Kali Linux for Ethical Hackers

Basic Commands

ls -la = show all files + all hidden files

pwd = displays the directory you currently are in

locate file or locate filetype = finds and lists all files with similar names and/or file types

updatedb = updates the database

passwd = updates the user's password to a new password

man = instructions for any command you are trying to use

Users & Priveleges Commands

Command: ls -la

```
Output: d rwx r-x  r-x 14 root root 4096 May 11 17:41
           | 1 |  2  | 3 |
```

1) Permissions for Owner of the file
2) Permissions for Group Members that own a copy of the file
3) Permissions for all other users

d = directory
r = read
w = write
x = execute

----

chmod (change mode/permission)

chmod +rwx or chmod 777 (gives full permissions to read/write/execute for the user)
chmod +rw
chmod +x ...

----

adduser (creates a new user in the group)

----

cat /etc/passwd (shows all the users in the group)

----

cat /etc/shadow - (shows all the user's hashed passwords)

----

su user - (switches to different user account)

su -   - (switches to root user)

We can navigate to the log files to view which users have tried to access unauthorized information,
or if there has been failed attempts with their logins by going the the /var/log directory,

cd /var/log

and then outputting the auth.log file by using,

cat auth.log

Alternatively, we could combine these two commands and instead use,

cat /var/log/auth.log

Network Commands

ifconfig - (shows connected interfaces and IPs addressed to them)

----

iwconfig - (shows wireless interfaces information, if connected)

----

arp -a - (shows IP address the interface talks to & the MAC address associated with it)

----

netstat -ano - (shows the active connections running on the machine)

*** Useful to see if your machine is talking to another machine/service on a specific port number ***

----

route - (shows routing table / shows where traffic is exiting)

Viewing, Creating, & Editing Files

echo “hi” > hey.txt - (Writes a string “hi” to a .txt file named “hey”)

----

cat hey.txt - (Prints out what is stored in the file)

----

echo “bye” >> hey.txt - (Appends a string to the file)

*** Append is useful for creating lists: ***

Ex:
- IP addresses you might be collecting
- A series of commands you want to execute all at once

Different text editors within the terminal

nano
gedit
vi
vim

Starting & Stopping Services in Kali

service apache2 start - (starts the apache2 service so that you can host a local html webpage)

service apache2 stop   - (stops the apache2 service)

----

python -m SimpleHTTPServer 8080 - (starts up a simple HTTP web server running on port 8080)

****************** Question *******************

** How to spin up a FTP server using python? **

----

systemctl enable ssh   - (Persistently enables ssh on your machine when it starts up)
systemctl enable postgresql   - (Persistently enables postgresql for Metasploit framework)

systemctl disable ssh   - (Disables ssh on your machine when it starts up)

Installing & Updating Tools

apt-get update && apt-get upgrade - (checks all the sytems packages for updates & updates them; then upgrade the packages to the newest packages)

----

apt-get install git   - (installs git into your machine / used for github/gitlab and other git repositories)

Scripting with Bash

ping 192.168.1.90 -c 1 > ip.txt   - (sends 1 packet to the IP address and stores the output into “ip.txt”)

Now we want to narrow down the output of this file to extract the IP address by using,

cat ip.txt | grep “64 bytes”

Output:

```
    64 bytes from 192.168.1.90: icmp_seq=1 ttl=128 time=2.21 ms
```

----

grep - narrows down results and displays only lines containing search term or file type

----

We can narrow down further by using,

cat ip.txt | grep “64 bytes” | cut -d “ ” -f 4

cut -

    -d (delimiter) : what you want to remove, in this case, “ ", or space is our delimiter

    -f (field) : which field position you want to return, in this case, we want field 4 where the IP address is

Output:
```
    192.168.1.90:
```

Now we want to remove the colon at the end of the IP address by using,

```
cat ip.txt | grep “64 bytes” | cut -d “ ” -f 4 | tr -d “:”
```

tr - (translate)
    -d(delimeter) : what you want to remove, in this case, “:”

Output:
```
    192.168.1.90
```

Now, we can automate obtaining IP addresses by writing this into a script by typing,

```
gedit ipsweep.sh
```

Afterwards type the following:

```
#!/bin/bash
if [ "$1" == "" ]
then
echo "Must enter IP address"
echo "Syntax: ./ipsweep.sh 10.0.2"

else
for ip in $(seq 1 254);
do
ping -c 1 $1.$ip | grep "64 bytes" | cut -d " " -f 4 | tr -d ":" &
done
fi
```

Save and close the file.

-c 1 - (Used to send just one packet per address)
$1   - (Used to obtain user input)
$ip - (Variable used to hold ip addresses)
&   - (Used to enable threading on the process)

We have to then give our file the correct permissions to execute by using,

```
chmod +x ipsweep.sh
```

Now we can execute our script by typing,

```
./ipsweep.sh IP_ADDR

    Ex: ./ipsweep.sh 192.168.1
         ./ipsweep.sh 10.0.2
```

We can output this to a file by using,

```
./ipsweep.sh 10.0.2 > iplist.txt
```

and then view our results by typing,

```
cat iplist.txt
```

Output:
```
    10.0.2.2
    10.0.2.4
    10.0.2.3
    10.0.2.15
```

Now that we have IP addresses in a list, we can use them to perform penetration tests like nmap

To do this, we can type in the console:

```
for ip in $(cat iplist.txt); do nmap -sS -p 80 -T4 $ip; done    (end with semi-colon to run one at a time = single threaded)
```

or

```
for ip in $(cat iplist.txt); do nmap -sS -p 80 -T4 $ip & done    (end with ampersand to run simultaneously = multithreaded)
```

Metasploit Framework

Armitage - GUI for Metasploit Framework

** Metasploit Console **

First need to start postgresql database to run metasploit on,

In terminal, type to start the database service:
```
    service postgresql start
```

Then to run metasploit type,

```
    msfconsole
```

Basic Commands

help - (Gives you access to a menu to explain the commands you can use with the Framework)

search [<options>] [<keywords>] - (Allows you to find a module based on what keyword knowledge you give)
Ex:
```
search type:exploit platform:windows flash
```

use <name | term | index> - (Allows you to load a module (exploit))
Ex:
```
use exploit/windows/browser/adobe_flash_avm2
```

show all - (Displays all the information about the module loaded)

show options - (Displays the options we can change about the module, changes depend on the how and method of exploitation)
- (Options include: Number of Retries, Server Host, Server Port, SSL/SSLCert (enable/disable), URI Path)

set [option] [value] - (Allows you to change a specific option about the module)
- (Options include: Retries, SRVHOST, SRVPORT, SSL, SSLCert, URIPATH)

show payloads - (Displays all of the payloads that you can load, different way of approaching an attack)

show targets - (Displays the exploitable targets)

show info - (Gives you information about the exploit)

back - (Takes you a step back in the msfconsole) exit - (Terminates the msfconsole framework)

Understanding Metasploit Modules

Modules usually saved in the following directory:

```
cd /usr/share/metasploit-framework/modules
```

Exploits - A module that will take advantage of a system's vulnerability and it will install/plant a payload onto the victim machine

Payloads - Files left onto the victim machine that gives you access or control over the system
- Ex: Rootkits / reverse shell

- Divided into 3 groups:
  1. Singles - (Designed to do one single action)
     Ex: Keylogging
  2. Stagers - (Used for creating a communication link between the attacker and the victim machine)
     - (Can be used to deliver another payload)
  3. Stages - (Very large payloads, can give very good control over the target)
     Ex: VNC connections, meterpreter shell, reverse shell

Auxiliary - Unique types of modules that perform scanning, fuzzing, sniffing, etc. to find vulnerabilities about the target

Encoders - Used to re-encode payloads and exploits to get by security systems such as anti-viruses

Nops (No Operation) - Causes the system's CPU to do nothing for 1 clock cycle which can allow you to remotely execute a specific file/code after you've exploited the buffer overflow

Post - Used after the target system has been exploited, allows you to perform further attacks once the system has been owned
Ex: Keyloggers, webcam, browser information

Information Gathering - Aux. Scanners

nmap - A network exploitation tool and security / port scanner
- Nmap can provide further information on targets, including reverse DNS names, operating system guesses, device types, and MAC addresses

Ex: nmap -A -T4 scanme.nmap.org
- A, to enable OS and version detection, script scanning, and traceroute
- T4, for faster execution

SCAN TECHNIQUES:
```
-sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
-sU: UDP Scan
-sN/sF/sX: TCP Null, FIN, and Xmas scans
--scanflags <flags>: Customize TCP scan flags
-sI <zombie host[:probeport]>: Idle scan
-sY/sZ: SCTP INIT/COOKIE-ECHO scans
-sO: IP protocol scan
-b <FTP relay host>: FTP bounce scan
```

Scanning ssh if port is open

command: search ssh_version

We want to use the scanner auxiliary so from this list select ‘auxiliary/scanner/ssh/ssh_version’ by typing,

command: use auxiliary/scanner/ssh/ssh_version

Next we want to show options to display target host, port, threads, & timeout settings by typing,

command: show options

Now we can set our target IP by using,

command: set RHOSTS 10.0.2.15

As well as increase the number of threads we want to run concurrently,

command: set THREADS 100

Finally, we can run this scanner by entering,

command: run

This will return information about the target's ssh service vender/product & OS information