





OverTheWire — Bandit(11–20) Walkthrough


 Vighnesh Srinivas · Follow


8 min read · Apr 19, 2021

 51









This is the second part of my bandit walkthrough. You can find the first part [here](#). In this part, we will cover levels 11 through 20. So let’s jump right in!

Level 10 -> Level 11

Here the password for the next level is stored in the file “**data.txt**” in the home directory. The data in this file is base64 encoded. We can “*cat*” the file and see that there’s just one word in it. So we run the command ‘*base64 -d data.txt*’ and get the password for the next level.

```
bandit10@bandit:~$ wc data.txt
 1 1 69 data.txt
bandit10@bandit:~$ cat data.txt
VGhlIHBhc3N3b3JkIGlzIElGdWt3S0dzRlc4TU9xM0lSRnFyeEUxaHhUTkViVVBSCg==
bandit10@bandit:~$ base64 -d data.txt
The password is IFukwKGSFW8MOq3IRFqrxE1hxTNEbUPR
bandit10@bandit:~$
```

Password for bandit11

Level 11 -> Level 12

Here we see another “**data.txt**” file in the home directory. The webpage for this level says that there are upper and lower case characters in this file which are rotated by 13 positions. So basically each letter in a string is replaced by a character at the 13th position from it. So a will be replaced by n, b will be replaced by o, and so on. This is basically a special type of Ceaser's cipher. After reading around for a bit, I found a StackOverflow link explaining to decrypt this cipher using the ‘*tr*’ command which stands for translate. This command translates a set of characters into another set of characters. So we can then run the command “*tr '[a-z][A-Z]' '[n-za-m][N-ZA-M]*””. This command will basically rearrange the characters decrypt the data in the file.

```
bandit11@bandit:~$ ROT13
-bash: ROT13: command not found
bandit11@bandit:~$ cat data.txt | tr '[a-z][A-Z]' '[n-za-m][N-ZA-M]'
The password is 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu
bandit11@bandit:~$
```

Password for bandit12

Level 12 -> Level 13

In this level, the password is stored in “**data.txt**” which is a hexdump of a file that’s been repeatedly compressed. The webpage for this level suggests that we make a folder in the /tmp directory, copy the file in this directory and work on it. So we can create a directory using ‘*mkdir /tmp/shirahoshi*’ and copy the file in this location by running the command ‘*cp data.txt /tmp/shirahoshi*’. We then rename the file using the command ‘*mv data.txt hex.txt*’.

```
File Actions Edit View Help
bandit12@bandit:~$ mkdir /tmp/shirahoshi
bandit12@bandit:~$ cp data.txt /tmp/shirahoshi
bandit12@bandit:~$ cd /tmp/shirahoshi
bandit12@bandit:/tmp/shirahoshi$ mv data.txt hexdump.txt
bandit12@bandit:/tmp/shirahoshi$
```

Here we need to first decrypt the hex dump file and for that, we use a tool called `xxd`. `xxd` is a tool that can make a hex dump and reverse it. So we use `'xxd -r hexdump.txt > new'`. This should reverse the hex dump file and store its content in the file “new”. Now we know that this file has been repeatedly compressed. On running `'file new'`, we find out that it has been zipped by using `gzip`. Now before we start decompressing the file, we need to make sure that it has the proper extension else the tools won't work. So we first change the extension and rename the file `new.gz`. We can now unzip the file using `'gzip -d new.gz'`.

```
File Actions Edit View Help
bandit12@bandit:/tmp/shirahoshi$ xxd -r hexdump.txt > new
bandit12@bandit:/tmp/shirahoshi$ file new
new: gzip compressed data, was "data2.bin", last modified: Thu May  7 18:14:30 2020, max compression, from Unix
bandit12@bandit:/tmp/shirahoshi$ mv new new.gz
bandit12@bandit:/tmp/shirahoshi$ gzip -d new.gz
bandit12@bandit:/tmp/shirahoshi$ ls
hexdump.txt  new
bandit12@bandit:/tmp/shirahoshi$
```

Now after decompressing the file, we check the file type again using the `'file'` command. We see that it is of file type `bzip2`. So we use the tool `bzip2` to decompress this file by using the command `'bunzip2 new'`.

```
bandit12@bandit:/tmp/shirahoshi$ ls
hexdump.txt  new
bandit12@bandit:/tmp/shirahoshi$ file new
new: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/shirahoshi$ bunzip2 new
bunzip2: Can't guess original name for new -- using new.out
bandit12@bandit:/tmp/shirahoshi$ ls
hexdump.txt  new.out
```

Now we have another compressed file! We run the `'file'` command again and see that it is a `gzip` file. So we rename it again decompress it using `gunzip` as described above. Now when we check the file type, we find out that it's a `tar` file. So we rename the file with `.tar` extension and then run the command `'tar -xf new.tar'`. The `'-x'` flag tells `tar` to extract from the file and `'-f'` flags tells it that it is a regular file.

```
bandit12@bandit:/tmp/shirahoshi$ mv new.out new.gz
bandit12@bandit:/tmp/shirahoshi$ gunzip new.gz
bandit12@bandit:/tmp/shirahoshi$ ls
hexdump.txt  new
bandit12@bandit:/tmp/shirahoshi$ file new
new: POSIX tar archive (GNU)
bandit12@bandit:/tmp/shirahoshi$ mv new new.tar
bandit12@bandit:/tmp/shirahoshi$ tar -xf new.tar
bandit12@bandit:/tmp/shirahoshi$
```

On running the `'ls'` command, we see a new file named “`data5.bin`”. We check the file type and figure out that it is a `.tar` file. We know that this file has been compressed several times, so we go through the same process of finding the file, changing the extension in case it's a `gzip` file, and then decompress it using the methods described above. We continue doing this until we finally get the text file. We then `'cat'` out the “`data8`” file which has our password. Sweet!

```
File Actions Edit View Help
bandit12@bandit:/tmp/shirahoshi$ ls
data5.bin  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/shirahoshi$ mv data5.bin data5.tar
bandit12@bandit:/tmp/shirahoshi$ tar -xf data5.tar
bandit12@bandit:/tmp/shirahoshi$ ls
data5.tar  data6.bin  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ file data6.bin
data6.bin: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/shirahoshi$ bunzip2 data6.bin
bunzip2: Can't guess original name for data6.bin -- using data6.bin.out
bandit12@bandit:/tmp/shirahoshi$ file
Usage: file [-bcChiklLNnprsvzZ0] [--apple] [--extension] [--mime-encoding] [--mime-type]
        file [-e testname] [--F separator] [--f namefile] [--m magicfiles] file ...
        file -C [-m magicfiles]
        file [--help]
bandit12@bandit:/tmp/shirahoshi$ ls
data5.tar  data6.bin.out  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ file data6.bin.out
data6.bin.out: POSIX tar archive (GNU)
bandit12@bandit:/tmp/shirahoshi$ mv data6.bin.out data6.bin.tar
bandit12@bandit:/tmp/shirahoshi$ tar -xf data6.bin.tar
bandit12@bandit:/tmp/shirahoshi$ ls
data5.tar  data6.bin.tar  data8.bin  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Thu May  7 18:14:30 2020, max compression, from Unix
bandit12@bandit:/tmp/shirahoshi$ mv data8.bin data8.gz
bandit12@bandit:/tmp/shirahoshi$ ls
data5.tar  data6.bin.tar  data8.gz  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ gunzip data8.gz
bandit12@bandit:/tmp/shirahoshi$ ls
data5.tar  data6.bin.tar  data8  hexdump.txt  new.tar
bandit12@bandit:/tmp/shirahoshi$ file data8
data8: ASCII text
bandit12@bandit:/tmp/shirahoshi$ cat data8
The password is 8ZjyCR1BWfYkneahHwxCv3wb2a10RpYL
bandit12@bandit:/tmp/shirahoshi$
```

Password for bandit13

Level 13 -> Level 14

Here, to get access to the next level, we will need an SSH key which can be used to get access to level 14. In the home directory, we can find the said ssh key by running 'ls'.

```
bandit13@bandit:~$ ls
sshkey.private
bandit13@bandit:~$
```

We can then copy this file to our local system and change its permission to 600, which is required for ssh private keys to work. We can then use this key to login as *bandit14* using the command '*ssh -i id_rsa bandit14@bandit.labs.overthewire.org -p 2220*'. Here '*id_rsa*' file has our private key.

Level 14 -> Level 15

The webpage for this level says “The password for the next level can be retrieved by submitting the password of the current level to **port 30000 on localhost**.” So we first need to find the password of *bandit14* as we only have a private key and not the actual password for this user. Since we are the owner of the file “*etc/bandit_pass/bandit14*”, we can read the contents of this file and retrieve the password.

```
File Actions Edit View Help
bandit14@bandit:~$ cat /etc/bandit_pass/bandit14
4wcYUJFw0k0XLShLDzztnTBHiqxU3b3e
bandit14@bandit:~$
```

Password for bandit14

Now that we have *bandit14*'s password, all we have to do is connect to the localhost at port 30000 and supply the password. We can use **netcat** to open a connection to our localhost. **Netcat** is a tool that can be used for a variety of things including port scanning and listening for traffic. We can connect to the localhost by running the command '*nc localhost 30000*' and we get the password for *bandit15*!

```
bandit14@bandit:~$ nc localhost 30000
4wcYUJFw0k0XLShLDzztnTBHiqxU3b3e
Correct!
BfMYroe26WYalil77FoDi9qh59eK5xNr
bandit14@bandit:~$
```

Password for bandit15

Level 15 -> Level 16

To get the password for the next level, we will have to connect to the localhost over SSL on port 30001 and submit the current user's password. We can do this by using “**openssl**”. We run the command '*openssl s_client -connect localhost:30001*'. This opens up the connection to the localhost. We then provide the password and retrieve *bandit16*'s password.

```
bandit15@bandit:~$ openssl s_client -connect localhost:30001
CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = localhost
verify return:1
---
Certificate chain
 0 s:/CN=localhost
  i:/CN=localhost
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICBjCCAW+gAwIBAgIEc/JorzANBgkqhkiG9w0BAQUFADAUMRIwEAYDVQDDAlsb2NhbgHhvc3QwHhcnMjEwMzI2MTExNzE2WhcnMjEwMzI2MTExNzE2WjAUMRIwEAYDVQDDAlsb2NhbgHhvc3QwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAL4S7GJ6MMKd9sGjXvizaYVKxlXF2yR6rEfo2lm4WLI/oJqAl2ZImgrFR0tCATWRgSjSuGTFbQRLlXzMYXtGNUOnU3cYncuJYwXFUCCTD7FcqTjS3y7x/YXcLANzQPL42SEJDTQ53AYXMEcP2rzv7z2H6qMZxSVJEUq6FFaZS4GPAgMBAAGjZTBjMBQGA1UdEQQNMAUCWxxyV2FsaG9zdDBLBglghkgBhvhCAQ0EPhy8QXV0b21hdGJjYWxseS8nZW5lcmF0ZWQgYnkgTmNhdc4gU2VlIGh0dHBzOi8vbm1hcC5vcmcvbmNhdc8uMA0GCsq6SIB3DQEBBQUAAAGBAAhub5wBj/cjV07NGPA02dntrDCct8De4jXMRM+WW8UtBu03yQEtFQuzp2BkeuYkXWbLjVTJ10Y2h13y3eMpEp5wwEgdxWg11z0AyQYwzXqzxmB96mBrfyEZn8S83bvty7M6ig4psS1FtWe0XQmR27Zet1Exzz/AuNagbJjppBQq
-----END CERTIFICATE-----
subject=/CN=localhost
issuer=/CN=localhost
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1019 bytes and written 269 bytes
Verification error: self signed certificate
```

```
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 7E8C270B3968695F9E35DA55FF1798A895EE61479D2C295A0C83A89D3BE71B
    Session-ID-ctx:
    Master-Key: 386A1683C1618E4A2E93FC631B3E52249D213DC32D0D77ACDBBC9B68A08148EE3DF950F8D3354B6DFA4D0EAD508B37
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - 1f 3a 31 dc eb 5d f2 89-9c 01 f2 14 3c 29 8d d7      .:1..].....<)..
    0010 - 55 1d 06 b6 19 1c c1 d4-be af 9d 15 56 54 c4 b2      U.....VT..
    0020 - 75 1d e1 96 b6 05 cb d2-70 1f a1 bd 05 3e f8 39      u.....p....>.9
    0030 - 25 95 c6 7d 11 36 7b 8e-81 33 b9 2d 9d a2 76 58      %..}.6{..3--..vX
    0040 - 53 96 cf bf 9b 97 55 ca-92 95 a2 47 81 84 63 58      S.....G...cX
    0050 - b6 af 32 21 ca da 41 42-a8 b5 e1 2e a9 37 9d 61      ..21..A8.....7..a
    0060 - c2 52 3d 20 29 b3 5d 54-f8 2f 37 43 36 e7 1e 24      .R-).]T./7/6...$
    0070 - 93 a6 63 44 09 a8 bd 27-2c 24 5d b7 c2 d8 da 2a      ...cD... '$)....*
    0080 - a1 aa 91 a4 75 50 b4 e6-bd c2 8f bc b1 87 4b a6      ....UP.....K.
    0090 - d7 90 07 6a 2c e5 38 d1-a9 bd c7 9a 1d 37 7c 93      ...j,.8.....7[.
    Start Time: 1617279126
    Timeout    : 7200 (sec)
    Verify return code: 18 (self signed certificate)
    Extended master secret: yes
---
BfMYroe26WYalil77FoDi9qh59eK5xNr
Correct!
cluFn7wTiGryunymY0u4Rcff5xQluehd
```

Password for bandit16

Level 16 -> Level 17

On this level, we first have to find the correct port in the range 31000 to 32000, connect to it on localhost over SSL, and submit the current user’s password to get the password for the next level. First, we need to find the correct port to connect to. We can do this using **nmap**. Nmap is a port scanner that tells us about all the open ports and the services they are running. So we do a quick nmap scan by running the command ‘*nmap -p 31000–32000 localhost*’. This shows us the following output.

```
bandit16@bandit:~$ ls
bandit16@bandit:~$ ls -la
total 24
drwxr-xr-x  2 root    root      4096 May  7  2020 .
drwxr-xr-x 41 root    root      4096 May  7  2020 ..
-rw-r----- 1 bandit16 bandit16   33 May  7  2020 .bandit15.password
-rw-r----- 1 root     root       220 May 15  2017 .bash_logout
-rw-r----- 1 root     root     3526 May 15  2017 .bashrc
-rw-r----- 1 root     root       675 May 15  2017 .profile
bandit16@bandit:~$ nmap -p 31000-32000 localhost

Starting Nmap 7.40 ( https://nmap.org ) at 2021-04-01 16:04 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00033s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
31046/tcp  open  unknown
31518/tcp  open  unknown
31691/tcp  open  unknown
31790/tcp  open  unknown
31960/tcp  open  unknown
```

nmap output.

We can then manually connect to each port till we find the correct one. Once we have the correct port, we can use the same openssl tool to connect to the localhost and retrieve the password in the similar fashion. The command is ‘*openssl s_client -connect localhost:31790*’. Here we submit the current user’s password and it then gives us an RSA private key for the next user.


```
-----
cluFn7wTiGryunymYOU4RcffSxQluehd
Correct!
-----BEGIN RSA PRIVATE KEY-----
MIIEogI8AAKCAQEAvM0kuiFmMg6HL2YPI0jon6iWfbp7c3jx34YkYWqUH57SudyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMl0Jf7+BrJ0bArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZL870RiO+rW4LCDcNd2LuvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW30ekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XF0JuaQIDAQABAoI8ABagxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQw1k7xfw+24pRNUDE6SFthOar69jp5RLLwD1NhPx3i8l
J9n0M8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKuFD52y0Q9qOkwFTEQpjtf4uNtJom+asvlpM58A
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRB2G82so8vUHK/fur850Efc9TncnCY2crpoqsgghifKLxrlGtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2Mx3NaeSDm75Lsm+tBbAiyC9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KasZx+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCivGCSx+X3L5SiWg0A
R57hJglezIiVjv3aGwHwvLZvtszK6zV6oXFau0ECgYAbjo46T4hyP5tJi93V5HDi
TtieK7xRVxUL+iU7rWkGAXFpMLFteQEsRr7P3/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB30hYimtiG2Cg5JCIZFHxD6MjEG0iu
L8ktHMPvodBwNsSBULpG0QK8gBApLTfC1H0nWiMG0U3KPwYwT006CdTkmJ0mL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7Yfz0KU4ZxEnabvXnvWkU
Y0djHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyZRqaM
77pBAoGAMmjMIJdjp+Ez8duyn3ieo36yrttF5NSsJLABxPpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEB1104f7HVm6EpTscdXU+bCXWkfjuRb7Dy9G0tt9JPsX8MBTakzh3
v8gsyi/sN3RqRBcGU40F0oZyFAMT8s1m/uYv5206IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----

closed
bandit16@bandit:~$ █
```

Private key for bandit17

Level 17 -> Level 18

In the home directory of *banditi17*, there are two files with almost the same content except for one line. That line has the password for the next level. We can find this line by using a tool called “diff”. We can run the command ‘*diff passwords.old passwords.new*’. This will give us the password for the next level.

```
File  Actions  Edit  View  Help
bandit17@bandit:~$ ls
passwords.new  passwords.old
bandit17@bandit:~$ diff passwords.old passwords.new
42c42
< w0Yfolrc5bwjs4qw5mq1nnQi6mF03bii
---
> kfBf3eYk5BPBRzwjqutbbfE887SVC5Yd
bandit17@bandit:~$ █
```

Password for bandit18

Level 18 -> Level 19

This is an interesting challenge! When you login with *bandit18*’s credentials, we get access, see a “Byebye !” message and the session terminates. This is because the configuration in .bashrc is such that it will log you out when you try to log in. After looking around for a while, I figured out that ssh lets you execute a command when trying to open a connection. Amazing! So even if our shell terminates, it will retrieve the results of the command we execute. So Let’s try to list files in the home directory first. We run the command ‘*ssh bandit18@bandit.labs.overthewire.org -p 2220 ls*’. Here as you can see, I am trying to execute the ‘*ls*’ command. Let’s see what this does.

```
(root@kali)-[~]
# ssh bandit18@bandit.labs.overthewire.org -p 2220 ls
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit18@bandit.labs.overthewire.org's password:
readme
```

ssh executed the ‘*ls*’ command and displayed the “readme” file in the home directory. Sweet! So this means we can also read the contents of this file. We run the command ‘*ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme*’ and voila! We get the password for *bandit19*.

Level 19 -> Level 20

The webpage for this level says that we need to use the *setuid* binary in the home directory to get the password for the user *bandit20*. Now *setuid* binary are files which let you run the file as if you were the owner of that file. It is a special type of user privilege. If you run the command ‘*ls -la*’ in the home directory, you will see “s” in the user permissions. You can also see that the owner of this *suid* binary is *bandit20*. So we can execute this binary as *bandit20*.

```
bandit19@bandit:~$ ls -la
total 28
drwxr-xr-x  2 root    root    4096 May  7 2020 .
drwxr-xr-x 41 root    root    4096 May  7 2020 ..
-rwsr-x---  1 bandit20 bandit19 7296 May  7 2020 bandit20-do
-rw-r--r--  1 root    root     220 May 15 2017 .bash_logout
-rw-r--r--  1 root    root   3526 May 15 2017 .bashrc
-rw-r--r--  1 root    root    675 May 15 2017 .profile
bandit19@bandit:~$
```

Now to figure out what this binary does, we have to execute it. So we run the command `./bandit20-do`. On running this command, the binary tells us to “run a command as another user”. So we basically have to execute this binary along with a command of another user. So let’s try to print *bandit20*’s password as that user is the owner of this binary. we run the command `./bandir20-do cat /etc/bandit_pass/bandit20`. This gives us the password for the user *bandit20*.

```
File Actions Edit View Help
bandit19@bandit:~$ ls
bandit20-do
bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
Example: ./bandit20-do id
bandit19@bandit:~$ ./bandit20-do ls -la
total 28
drwxr-xr-x  2 root    root    4096 May  7 2020 .
drwxr-xr-x 41 root    root    4096 May  7 2020 ..
-rwsr-x---  1 bandit20 bandit19 7296 May  7 2020 bandit20-do
-rw-r--r--  1 root    root     220 May 15 2017 .bash_logout
-rw-r--r--  1 root    root   3526 May 15 2017 .bashrc
-rw-r--r--  1 root    root    675 May 15 2017 .profile
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
GbKksEFF4yrVs6il55v6gwY5aVje5f0j
bandit19@bandit:~$
```

Password for bandit20.

- Linux
- War Games
- Privilege Escalation
- Overthewire
- Bandit



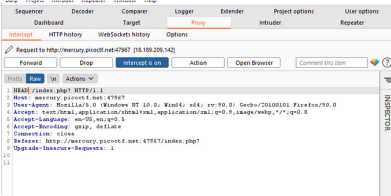
Written by Vighnesh Srinivas

38 Followers

A Cybersecurity enthusiast looking to make his career in offensive security.

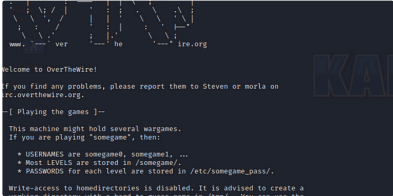
Follow

More from Vighnesh Srinivas



picoCTF—GET aHEAD
Hello Hackers,

Jun 7, 2021 8



OverTheWire—Bandit(0-10) Walkthrough
Hello hackers,

Apr 4, 2021 33 2

```
leviathan@leviathan:~$ ls
leviathan$ ./check
0x5530, 0, 0xfffff784, 0x040010 unfinished ...
/etc, 0x540f6700 = 10
/etc, 0x540f6700 = 10
/etc, 0x540f6700 = 10
)
Good Bye ... "Wrong password, Good Bye ... = 20
+++
$ ./check
pass/leviathan
```