BANDIT LV 0

TASK

- Log into the level with SSH.
- Server: bandit.labs.overthewire.org
- Port: 2220
- Username: bandit0
- Password: bandit0

BANDIT LV 0-1

TASK

The password for the next level is stored in a file called readme located in the home directory

Solution

1. We log in through SSH with the information above. And as explained in the theory section, we land in the home directory from user ‘bandit0’. (You can check this with the pwd command.)
2. Next, we can make sure that the readme file is actually in the folder.

Since this is the case, we can print the content of a file with the following command syntax: cat <file>.

BANDIT LV 1-2

TASK

Get the password from the file called ‘-’.

Solution

First, we make sure the file is in the folder by printing all the files.

Using the command cat - does not return anything. So instead of writing just - we add the path and write ./- and the command works as it should:

So we got the next password.

BANDIT LV 2-3

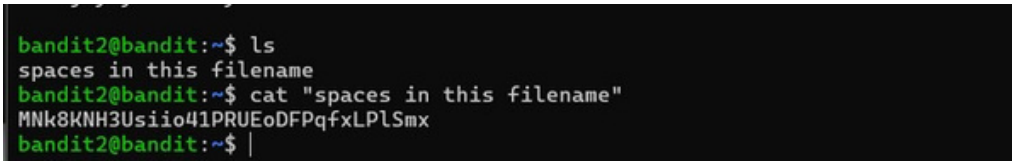
TASK

The password for the next level is stored in a file called spaces in this filename located in the home directory.

Solution

Similar to the previous level, simply trying to use the filename does not work:

This is because it assumes that we look at four files (or directories), which, however, do not exist.
Instead, we need to surround names with spaces with quotes (single or double) to indicate that all of it belongs to the name for one file:



And we can read the file and get the next password.

BANDIT LV 3-4

TASK

The password for the next level is stored in a hidden file in the inhere directory.

Solution

First, we go to the correct folder and then print all its files to find the filename

Therefore our file with the password is called .hidden, and we can read its content.

And we got the password for the bandit4 user.

BANDIT LV 4-5

TASK

The password for the next level is stored in the only human-readable file in the inhere directory

Solution

We again go into the ‘inhere’ directory and print out the files in the system:.

We can see that there are ten files.
We can use different methods to find the human-readable file and therefore, the password.

1. We could just print the contents of every file (cat). This is, however, not very efficient when we deal with more files.
2. Instead, we could use the method I mentioned in the theory part. The command structure is file <filename>. Instead of using a filename, we use a wildcard to get the type for all the files. Additionally, looking at the file names, specifically at the fact, the names start with ‘-’, gives us problems. Therefore we use the same method as in Level 2.

We can see that only ‘-file07’ is of type ‘ASCII text’, which is one of the encodings, that humans can read. (It is also the same file type as the files from previous levels.) Now we only need to print the file:
And get the next password.

BANDIT LV 5-6

TASK

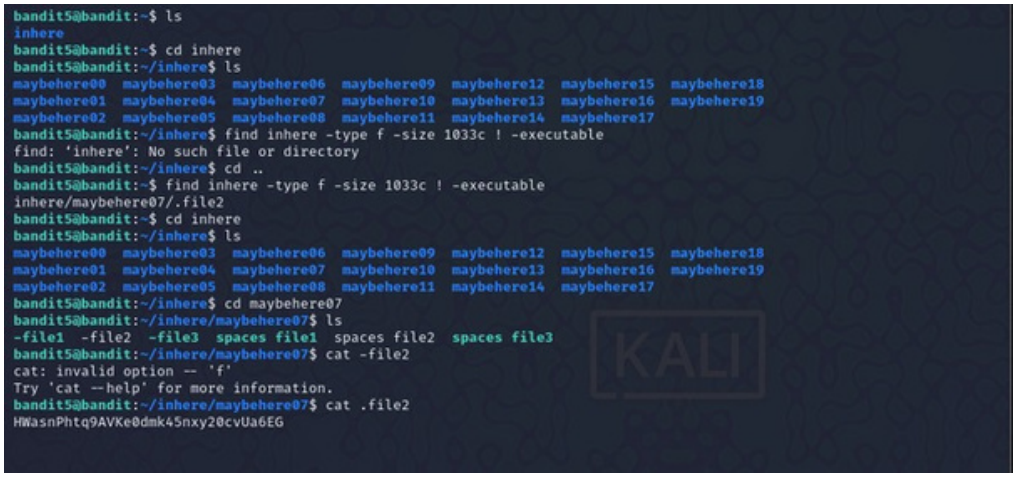
The password for the next level is stored in a file somewhere under the inhere directory and has all of the following properties:

- human-readable
- 1033 bytes in size
- not executable

Solution

We start again by going into the ‘inhere’ directory and getting an overview of what is inside it.

We get the file size, as mentioned in the Theory section, with help of the du command. Then we again use grep to filter for the correct size (‘1033’):



The most likely candidate would be find.

- We use -size 1033 to look for the file-size requirement.
- We use -type f to only look at files.
- We use -exec file '{}' \;, to execute the file command and get the file data type. After that, we simply need to filter the output for the file type ‘ASCII

And we got the password for the bandit6 user.

BANDIT LV 6-7

TASK

Find a file somewhere on the server. Properties:

1. owned by user bandit7
2. owned by group bandit6
3. 33 bytes in size

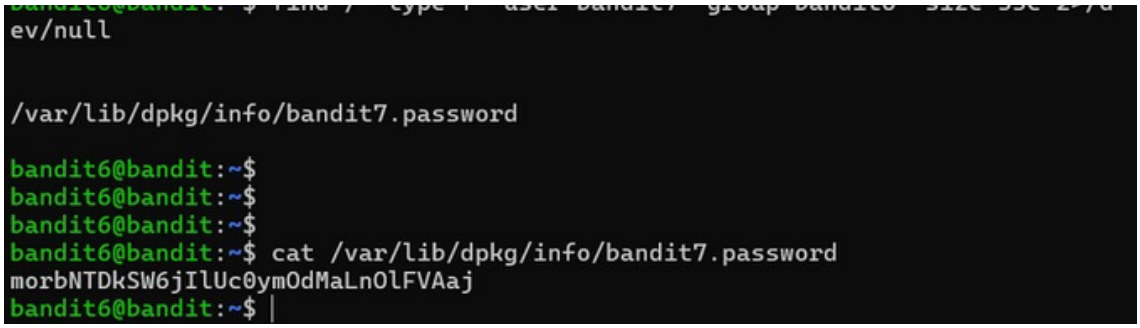
Solution

We use the find command with the following options:

- -type f, because we are looking for a file
- -user bandit7, to find files owned by the ‘bandit7’ user
- -group bandit6, to find files owned by the ‘bandit6’ group
- -size 33c, to find files of size 33 bytes

We need to run the command from the root directory to search the whole system. Running the command find / -type f -user bandit7 -group bandit6 -size 33c will, however, also print a Permission denied error for files that we do not have permission. We can append 2>/dev/null, which will ‘hide’ all error messages 1.

And we got the file and can read the next password.



And we got the password for the bandit7 user.

BANDIT LV 7-8

TASK

Get the password from a file, next to the word millionth

Solution

Checking the file size of data.txt, we can see it is huge:

So simply looking through the file, would take too long and be too much effort. Instead, we can try using grep, since the password is in the same line as the word ‘millionth’

And we got the password for the bandit8 user.

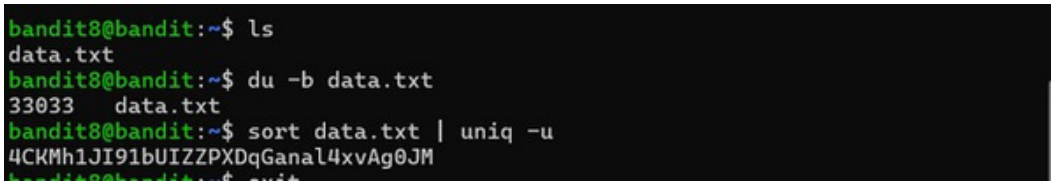
BANDIT LV 8-9

TASK

The password for the next level is stored in the file data.txt and is the only line of text that occurs only once

Solution

To find the line that occurs only once in the file, we first sort the lines and then filter for the unique one.



And we got the password for the bandit9 user.

BANDIT LV 9-10

TASK

The password for the next level is stored in the file data.txt in one of the few human-readable strings, preceded by several ‘=’ characters.

Solution

1. First, we need to distinguish human-readable strings in ‘data.txt’. We use the strings command.
2. Next, we want to filter that output by looking at lines that include more than one equal sign. - Assuming the equal signs and the password are on the same line, we can use grep again (as in Level 6). Since the task was unspecific regarding the number of equal signs, I used 3. However, next to the password, there are not too many lines with equal signs, so any amount between 1 and 10 would work (however, 2 to 10 would give the same result).

```
bandit9@bandit:~$ ls
data.txt
bandit9@bandit:~$ du -b data.txt
19379   data.txt
bandit9@bandit:~$ grep "password" data.txt
grep: data.txt: binary file matches
bandit9@bandit:~$ strings data.txt | grep ===
}========== the
3JprD========== passwordi
~fDV3========== is
D9========== FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey
```

And we got the password for the bandit10 user.

BANDIT LV 10-11

TASK

The password for the next level is stored in the file data.txt, which contains base64 encoded data.

Solution

The base64 command allows files as input, so we just need to use the command on the file.

```
bandit10@bandit:~$ ls
data.txt
bandit10@bandit:~$ du -b data.txt
69      data.txt
bandit10@bandit:~$ cat data.txt
VGhlIHBhc3N3b3JkIGlzIGR0UjE3M2ZaS2IwUlJzREZTR3NnMlJXbnBOVmozcVJyCg==
bandit10@bandit:~$ base64 -d data.txt
The password is dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr
bandit10@bandit:~$
```

And we got the password for the bandit11 user.

BANDIT LV 11-12

TASK

The password for the next level is stored in the file data.txt, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions.

Solution

- There are a lot of websites that offer ROT13 encryption/decryption, but sadly there is no build-in ROT13 command in Linux. However, I wanted a solution for the terminal, so I used the tr command for substitution.
- The substitution for ROT13 is A->N,...,Z->M. With tr it would be:

And we got the password for the bandit12 user.

BANDIT LV 12-13

TASK

The password for the next level is stored in the file data.txt, which is a hexdump of a file that has been repeatedly compressed. For this level, it may be useful to create a directory under /tmp in which you can work using mkdir.

Solution

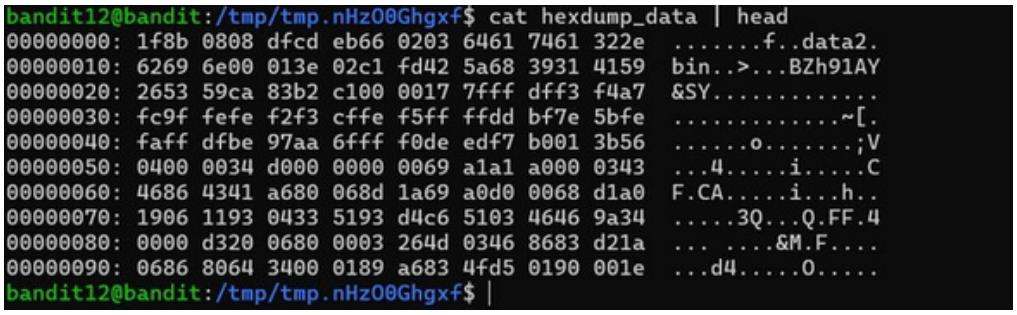
I have separated the task into three sub-tasks. Setting up a directory, reverting the hexdump and finally decompressing.

Create Directory and Move file

The first part of the task is to create a folder and copy the data to make further actions easier.

Revert hexdump of the file

Looking at the file, we see the format of the data. As stated it is a hexdump. It looks like this:



However, we want to operate on the actual data. Therefore, we revert the hexdump and get the actual data

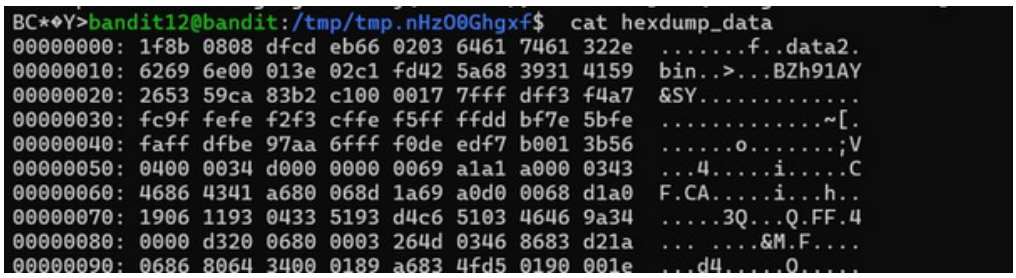
The actual data looks like this when printed to the console:

Repeatedly decompress

We now need to decompress the data. To figure out what decompression we need to use, look at the first bytes in the hexdump to find the file signature. We can search for these first bytes in a list of file signatures. An alternative would be to use the find command.

GZIP

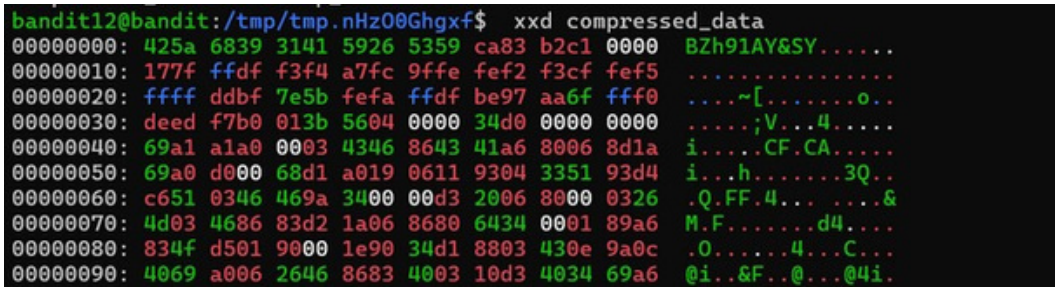
For gzip compressed files the header is \x1F\x8B\x08. Looking at the first line, we see that these are the first bytes of the file.



BZIP2

However, the data is still not fully decompressed, so we look at the first bytes again:

This time we have a different magic number. Quick googling tells us that BZ (= ‘425a’) is the file signature for bzip and the next two bytes h (= ‘68’) indicate the version, in this case, it is version 2. So we can rename the file with the appropriate file ending (.bz2) and decompress it with bzip2 -d

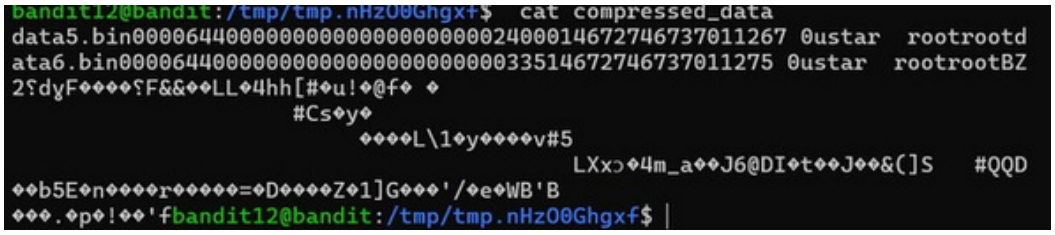


GZIP

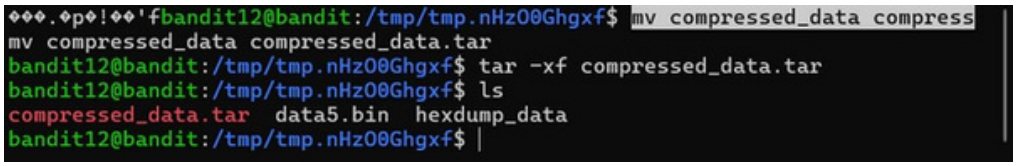
And the file is still compressed. xxd shows that it is ‘gzip’ compressed again. So we repeat the previous steps, renaming and decompressing

Tar archives

Using either cat compressed_data or xxd compressed_data | head (‘head’ to only get the first 10 lines), we can see the ‘data5.bin’ string, which is a filename.



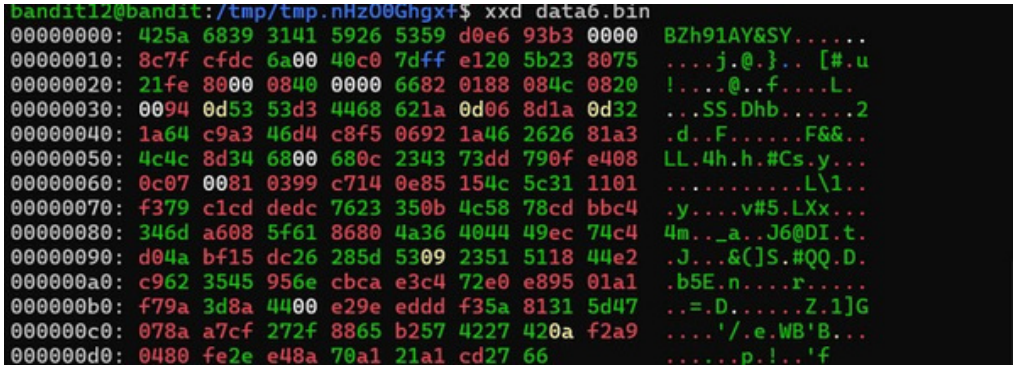
Using either cat compressed_data or xxd compressed_data | head (‘head’ to only get the first 10 lines), we can see the ‘data5.bin’ string, which is a filename.

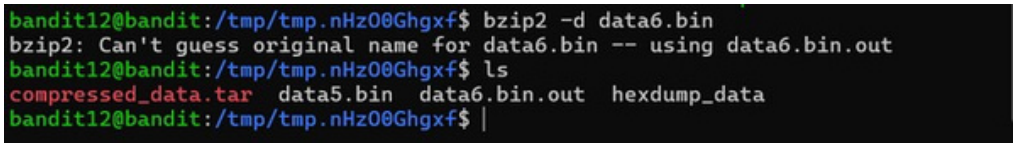


Now, ‘data5.bin’ seems to be another archive with a file called ‘data6.bin’. So we extract the file again.

BZIP2

The file ‘data6.bin’ seems to be bzip2 compressed again.





Tar Archive

‘data6.bin.out’ shows another file name ‘data8.bin’ again. So we extract this file.

```
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ tar -xf data6.bin.out
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ ls
compressed_data.tar  data5.bin  data6.bin.out  data8.bin  hexdump_data
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$
```

GZIP

Finally, we have to do one more decompression with gzip and we get a readable file with the password.

```
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ xxd data8.bin
00000000: 1f8b 0808 dfcd eb66 0203 6461 7461 392e  .......f..data9.
00000010: 6269 6e00 0bc9 4855 2848 2c2e 2ecf 2f4a  bin...HU(H,.../J
00000020: 51c8 2c56 70f3 374d 2977 2b4e 3648 4e4a  Q.,Vp.7M)w+N6HNJ
00000030: f4cc f430 c8b0 f032 4a0d cd2e 362a 4b09  ...0...2J...6*K.
00000040: 7129 77cc e302 003e de32 4131 0000 00    q)w....>.2A1...
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$
```

```
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ mv data8.bin data8.gz
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ gzip -d data8.gz
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ ls
compressed_data.tar  data5.bin  data6.bin.out  data8  hexdump_data
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$ cat data8
The password is FO5dwFsc0cbaIiH0h8J2eUks2vdTDwAn
bandit12@bandit:/tmp/tmp.nHzO0Ghgxf$
```

And we got the password for the bandit13 user.

BANDIT LV 13-14

Task

The password for the next level is stored in /etc/bandit_pass/bandit14 and can only be read by user bandit14. For this level, you don't get the next password, but you get a private SSH key that can be used to log into the next level.

Solution

I logged into the server as bandit13 and found the file 'sshkey.private' in the home directory. Knowing the location of the file, I can transfer it to my machine.

```
bandit13@bandit:~$ ls
sshkey.private
bandit13@bandit:~$ exit
logout
Connection to bandit.labs.overthewire.org closed.
PS C:\Users\Dhirsshin>
```

I used scp to connect to the remote machine and get the ssh key.

```
PS C:\Users\Dhirsshin> scp -P 2220 bandit13@bandit.labs.overthewire.org:sshk
ey.private .
                         _                     _ _ _
                        | |__   __ _ _ __   __| (_) |_
                        | '_ \ / _` | '_ \ / _` | | __|
                        | |_) | (_| | | | | (_| | | |_
                        |_.__/ \__,_|_| |_|\__,_|_|\__|


                      This is an OverTheWire game server.
            More information on http://www.overthewire.org/wargames

bandit13@bandit.labs.overthewire.org's password:
sshkey.private                                100% 1679     3.8KB/s   00:00
PS C:\Users\Dhirsshin>
```

Now that I had the private ssh key, I tried to log in with it. However, I got the following warning Permissions 0640 for 'sshkey.private' are too open., because it had the following writing permissions: -rw-r-----.

And we got into the server as bandit14.

BANDIT LV 14-15

Task

The password for the next level can be retrieved by submitting the password of the current level to port 30000 on localhost.

Solution

1.First, we need to find the password for bandit14. The previous levels stated that the password is in /etc/bandit_pass/bandit14.

2 .Next, we need to submit the password to port 30000 on localhost. I used nc to connect to localhost port 3000 and write the password.

BANDIT LV 15-16

Task

The password for the next level can be retrieved by submitting the password of the current level to port 30001 on localhost using SSL encryption.

Solution

Since the task states that the password can be retrieved using SSL encryption, I connect to the localhost server with the OpenSSL client and send the password from this level. The server then sends back the password for the next level.

```
bandit15@bandit:~$ openssl s_client -connect localhost:30001
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = SnakeOil
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN = SnakeOil
verify return:1
---
Certificate chain
 0 s:CN = SnakeOil
   i:CN = SnakeOil
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
   v:NotBefore: Jun 10 03:59:50 2024 GMT; NotAfter: Jun  8 03:59:50 2034 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
```

```
    Start Time: 1742919384
    Timeout   : 7200 (sec)
    Verify return code: 18 (self-signed certificate)
    Extended master secret: no
    Max Early Data: 0
---
read R BLOCK
8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo
Correct!
kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx

closed
```

BANDIT LV 16-17

Task

The credentials for the next level can be retrieved by submitting the password of the current level to a port on localhost in the range 31000 to 32000. First find out which of these ports have a server listening on them. Then find out which of those speak SSL and which don't. There is only 1 server that will give the next credentials, the others will simply send back to you whatever you send to it.

Solution

First, we need to find open ports between 31000 to 32000 on localhost and check what services are running on them. I used the service discovery from nmap. (This task could be split by first finding open ports and then doing the service discovery only on these ports. This could be faster.)

```
bandit16@bandit:~$ nmap -sV localhost -p 31000-32000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-25 16:26 UTC
Stats: 0:01:35 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 16:28 (0:00:24 remaining)
Stats: 0:01:40 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 16:28 (0:00:25 remaining)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00016s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
31046/tcp open  echo
31518/tcp open  ssl/echo
31691/tcp open  echo
31790/tcp open  ssl/unknown
31960/tcp open  echo
1 service unrecognized despite returning data. If you know the service/versi
on, please submit the following fingerprint at https://nmap.org/cgi-bin/subm
it.cgi?new-service :
SF-Port31790-TCP:V=7.94SVN%T=SSL%I=7%D=3/25%Time=67E2D938%P=x86_64-pc-linu
SF:x-gnu%r(GenericLines,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x2
SF:0current\x20password\.\n")%r(GetRequest,32,"Wrong!\x20Please\x20enter\x
SF:20the\x20correct\x20current\x20password\.\n")%r(HTTPOptions,32,"Wrong!\
SF:x20Please\x20enter\x20the\x20correct\x20current\x20password\.\n")%r(RTS
SF:PRequest,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20
SF:password\.\n")%r(Help,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x
SF:20current\x20password\.\n")%r(FourOhFourRequest,32,"Wrong!\x20Please\x2
SF:0enter\x20the\x20correct\x20current\x20password\.\n")%r(LPDString,32,"W
SF:rong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\.\n")
SF:%r(SIPOptions,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x20curren
SF:t\x20password\.\n");

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 163.65 seconds
```

So, nmap tells us that five ports are open. Only two ports (31518 and 31790) use SSL. Nmap also tells us that port 31518 runs only the echo service. The promising port seems to be port 31790, which runs an unknown service. Now we use OpenSSL again to connect to this port on localhost and send the password.

```
bandit16@bandit:~$ openssl s_client -connect localhost:31790 -ign_eof
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = SnakeOil
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN = SnakeOil
verify return:1
---
Certificate chain
 0 s:CN = SnakeOil
   i:CN = SnakeOil
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
   v:NotBefore: Jun 10 03:59:50 2024 GMT; NotAfter: Jun  8 03:59:50 2034 GMT
---
```

```
kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx
Correct!
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----

closed
```

The result is a private SSH key. So, we create a file (I called it ‘sshkey17.private’) to put the key into and like in Level 14, we need to make sure that the file only has permissions for the user.

BANDIT LV 17-18

Task

There are 2 files in the homedirectory: passwords.old and passwords.new. The password for the next level is in passwords.new and is the only line that has been changed between passwords.old and passwords.new

Solution

Find the one line that is different between two files.

```
bandit17@bandit:~$ diff passwords.old passwords.new
42c42
< w0Yfolrc5bwjS4qw5mq1nnQi6mF03bii
---
> kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd
```

BANDIT LV 18-19

Task

he password for the next level is stored in a file readme in the homedirectory. Unfortunately, someone has modified .bashrc to log you out when you log in with SSH.

Solution

Instead of logging into the machine with SSH, we execute a command through SSH instead. First, we use ls to make sure the readme file is in the folder then we can use cat to read it.

```
$ ssh bandit18@bandit.labs.overthewire.org -p 2220 ls
This is a OverTheWire game server. More information on http://www.overthewire.org/war

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

$ ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme
This is a OverTheWire game server. More information on http://www.overthewire.org/war

bandit18@bandit.labs.overthewire.org's password:
IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x
```

BANDIT LV 19-20

Task

To gain access to the next level, you should use the setuid binary in the homedirectory. Execute it without arguments to find out how to use it. The password for this level can be found in the usual place (/etc/bandit_pass), after you have used the setuid binary.

Solution

First, we check who the owner of the setuid binary is:

```
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
```

In this case, the owner is badit20 and the group is bandit19, this with '-rwsr-x—' means the user bandit19 can execute the binary, but the binary is executed as user bandit20.
Executing the binary says it simply executes another command as another user (as already explained, this user is bandit20). This means we can access the bandit20 users password file, which can only be read by the user bandit20.

```
bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
  Example: ./bandit20-do id
bandit19@bandit:~$ ./bandit20-do ls /etc/bandit_pass
bandit0   bandit12  bandit16  bandit2   bandit23  bandit27  bandit30  bandit4  bandit
bandit1   bandit13  bandit17  bandit20  bandit24  bandit28  bandit31  bandit5  bandit
bandit10  bandit14  bandit18  bandit21  bandit25  bandit29  bandit32  bandit6
bandit11  bandit15  bandit19  bandit22  bandit26  bandit3   bandit33  bandit7
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
GbKksEFF4yrVs6il55v6gwY5aVje5f0j
```

BANDIT LV 20-21

Task

TThere is a setuid binary in the homedirectory that does the following: it makes a connection to localhost on the port you specify as a commandline argument. It then reads a line of text from the connection and compares it to the password in the previous level (bandit20). If the password is correct, it will transmit the password for the next level (bandit21)

Solution

1. Using 'netcat', we can create a connection in server mode - which listens for inbound connection. To have netcat send the password, I use echo and pipe it into netcat. The -n flag is to prevent newline characters in the input. Lastly, we let the process run in the background with &.

```
bandit20@bandit:~$ echo -n 'GbKksEFF4yrVs6il55v6gwY5aVje5f0j' | nc -l -p 1234 &
[1] 24661
```

2 .Running the setuid binary with port 1234 means it will connect to our netcat server, receive the password inputted through echo and sends back the next password.

```
bandit20@bandit:~$ ./suconnect 1234
Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j
Password matches, sending next password
gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr
[1]+  Done
```