Bandit Wargames - by Cyph3rRyx

Hey ya guys! I am Ryx and today we are going to do some great basic Linux terminal hacking with the wargame known as Bandit Wargames from OverTheWire

So this will be the brief walkthrough through each and every level so that if you get stuck in-between at any step then you can check this article for help. I am going to do a walkthrough video series too and I will update you with the link down below but before that lets jump into the first source level.

Source level-Level 1:

The goal of this level is for you to log into the game using SSH. The host to which you need to connect is bandit.labs.overthewire.org, on port 2220. The username is bandit0 and the password is bandit0. Once logged in, go to the Level 1 page to find out how to beat Level 1.

> 💡 **First thought**: Well lets search the terms and read the articles. So we can know what is SSH(secure shell) and how to connect to it.

Explanation: We can connect to ssh server with the following command:

```
ssh bandit0@bandit.labs.overthewire.org -p 2220
```

The password is bandit0 so then we can enter the shell and then we can use ls to list out the files in the directory and we get ‘readme’ file so we cat it by using ‘cat readme’ and we get the first pass for badnit1

💡 boJ9jbbUNNfktd78OOpsqOltutMc3MY1 bandit1 lv0

Note: ‘cd’ to change directory , ‘ls’ to list the files in the directory , ‘cat’ to read the file data.

Level 1->2 :

We have, the password for the next level is stored in a file called – located in the home directory

💡 **First thought**: we need to just dive into the home directory by using ‘cd ..’ and then we can see the files in the home directory.

Explanation: By using ‘*ls -al*’ command we can see the permissions and files located and then we can execute ‘*cat ./-*’ and we can have our password.

💡 CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9 bandit2 lv1

Level 2->3:

We have, the password for the next level is stored in a file called spaces in this filename located in the home directory

💡 **First thought:** We need to just cat ‘spaces in this filename’ and retrieve the file.

Explanation: Well absolutely correct we can just do ‘*cd ..*’ and then use

‘*cat spaces\ in\ this \filename*’

(just write ‘cat spaces’ and hit enter) and we get our password.

> 💡 UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK bandit3 lv2

Level 3->4:

We have, the password for the next level is stored in a hidden file in the inhere directory.

> 💡 **First thought**: We will try ‘cd ..’ to go into the home directory and ‘ls -al’ to see the hidden files.

Explanation: Well yeah we will do the same but we can forget going in home directory instead go directly to ‘inhere’ through ‘cd inhere’ and directly execute ‘*ls -al*’ and then we can see a ‘.hidden’ file and we can simply cat it out by using ‘*cat .hidden*’ and boom we got our password.

> 💡 pIwrPrtPN36QITSp3EQaw936yaFoFgAB bandit4 lv3

Level 4->5:

We have, the password for the next level is stored in the only human-readable file in the inhere directory.

> 💡 **First thought:** We can just search what does the ‘du’ command do and then execute ahead.

Explanation: Well du command, short for disk usage, is used to estimate file space usage.

- The du command can be used to track the files and directories which are consuming excessive amount of space on hard disk drive.
- So first we will do same steps as above where we will go in ‘inhere’ directory and see the files using ‘ls’ and then to find the human readable file we will try to open up every file by using ‘cat -file01' and whichever is human readable it is the password for next level.

koReBOKuIDDepwhWk7jZC0RTdopnAYKh bandit5 lv4

Level 5->6:

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

First thought: So we have to go in inhere directory and then check for human readable file and with du command we can see the file size and it must not be executable.

Explanation: Well we can use ‘find’ command here for a ez clap.

After some searching on google I found out that if you want to find a specific file whose size is mentioned then use ‘find -size <size>’ for bytes we use ‘c’ keyword and we can even see if its executable or not by appending ‘!-executable’ with the file command.

So final command will be:

find . -size 1033c ! -executable

DXjZPULLxYr17uwoI01bNLQbtFemEgo7 bandit6 lv5

Level 6->7:

The password for the next level is stored somewhere on the server and has all of the following properties:

owned by user bandit7

owned by group bandit6

33 bytes in size

> 💡 **First thought:** Dig more about find command and use it here.

Explanation: So yeah after more digging we can actually use this command:

'*find ./ -type f -user bandit7 -group bandit6*'

and boom we get our password file and you can cat it and read the passcode

> 💡 HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs bandit7 lv6

Level 7->8

The password for the next level is stored in the file data.txt next to the word millionth

> 💡 **First thought:** Well search for all the commands given in help section to know more.

▼ grep:

The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression

Explanation:

- Well we can use several commands like while searching everything I get to know about '*egrep*' command which treats the pattern as an extended regular expression and prints out the lines that match the pattern.

- If there are several files with the matching pattern, it also displays the file names for each line.
- We want to see the word next to ‘millionth’ so we can use the command

egrep ‘millionth’ data.txt

and hit enter and boom we get our password!

> 💡 cvX2JJa4CFALtqS87jk27qwqGhBM9plV bandit8 lv7

Level 8->9:

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

> 💡 **First thought:** Well we can use ‘*uniq’* keyword given in help section. The uniq command in Linux is a command line utility that reports or filters out the repeated lines in a file.

Explanation:

- Well we have a unsorted data so lets just sort it via ‘sort data.txt’ and then execute ‘uniq -c’ to see the one unique line.
- Here ‘-c’ will display the number of time the line is repeated so where ever we get 1 its the password.
- Well actually we can pipe both the command together and execute them at a same time.
- Piping can be done by using pipe operator ‘|’.

The final command : ‘*sort data.txt* | *uniq -c*’

> 💡 UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR bandit9 lv8

Level 9->10:

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.

Explanation:

- Well we will use strings to see the human readable strings and then pipe it with = so we can see the password.

Final command will be:

strings data.txt | grep “=”

> 💡 truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk bandit10 lv9

Level 10->11:

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

> 💡 **First thought:** We can just get that data out and convert it from base64 to plain from any online converter.

Explanation:

- You can do that …its absolutely correct but what if we need to do it in terminal then we can use the following command :

echo ‘<string>’ | base64 – decode

> 💡 IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR bandit11 lv10

> **Note**: If you wanna go more deep into knowing all the commands then you can use ‘*man*’ command(a Wikipedia inside terminal for

> any command) or can use google(recommended)

Level 11->12:

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.

> 💡 **First thought:** The change of 13 position indicates rot13 algorithm so we can cat the data and convert the data into plain text from online site.

Explanation:

- Well we can do that thing and its absolutely correct. We can use the ‘tr’ command to use to decode it in terminal.

Final command –

echo ‘Gur cnffjbeq vf 5Gr8L4qetPEsPk8htqjhRK8XSP6x2RHh’ | tr ‘A-Za-z’ ‘N-ZA-Mn-za-m’

- We can do this in rot13 to plain text converter too…but ‘tr’ command is used in terminal to do the same.

> 💡 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu bandit12 lv11

Level 12->13:

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.

For example: *mkdir /tmp/myname123*. Then copy the datafile using *cp*, and rename it using *mv*.

💡 **First thought:** Well the question will be enough to understand the initiating thought.

Explanation:

- Use of '*mkdir*' to make directory,
- '*cp*' to copy the file, '*mv*' to rename the file,
- '*bzip2 -d <file>*' to decompress the bz2 file,
- '*gzip2 -d <file>*' to decompress the gz file,
- '*tar xf <file>*' to decompress the tar file. Its the loop of file compression and decompression so be patient and solve it easily.

💡 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL bandit13 lv12

Level 13->14:

We get an private SSH key which can only be accessible to bandit14 so we can use that to log in to the bandit14 SSH network.

Explanation:

- SSH key is used to log in to the bandit14.(for more info you can google it)

Command :

ssh -i sshkey.private bandit14@localhost

- After logging in bandit14 you need to go into the directory mentioned in the question /etc/bandit_pass/bandit14 to see the password.

💡 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e bandit14 lv13

Level 14->15:

We have to submit our bandit14 password on local host 30000 port so that we can get the password for next level.

Explanation:

- I tried SSH command to connect to local host 30000 but as I didn't had a key to go through it so the path was blocked so after searching something on internet, I got a solution.

Command :

'nc localhost 30000'

- This will ask for your password to reveal the encrypted data on that host and guess what that data was the password for next level.

BfMYroe26WYalil77FoDi9qh59eK5xNr bandit15 lv14

Level 15->16:

First Thought: We have to submit our bandit15 password on localhost portnumber 30001 so that we can get the password for next level.

Explanation:

- I tried last level command of 'nc localhost 30001' but it didn't returned anything so I read the question again and it was said that use SSL encryption. So..
- I used the following command '*nc* | *grep ssl*' but it returned cmd line and then dead end so watched a walkthrough on YT and they used

Command:

'*ncat – ssl localhost 30001*'

and then same method as last level.

💡 cluFn7wTiGryunymYOu4RcffSxQluehd bandit16 lv15

Level 16 ->17:

We have to do some scanning. We have to scan port 31000 to 32000 on localhost server and we need to find that one port which speak SSL and returns back the pass. So several commands were used..

Command

To scan the port 31000–32000:

nmap localhost -p 31000–32000

- We get several ports which are open and we need to check each one using command : *ncat – ssl localhost <port>*

And whichever is fake will return same thing and one unique port 31790 will return you an SSH Private key.

Now when we get an SSH private key we can log in to bandit17 with this one but we need to store it first so we use command :

cd /tmp

to open the directory and then use

touch pvt16.key

to create a file named pvt16.key. Then we use nano editor to edit that file.

- So we use: ‘nano pvt16.key’ to enter the nano editor and there we paste our key in that file and save the file using ‘CTRL+X’ and then if we need to use the key for the level pass then we need to change the permission so that it can be only accessed for that particular pass and to do that we need to see who had the permissions?

command to see the permission of the file:

ls -l pvt16.key

and then after seeing the file permission we can change it using ‘*chmod*’ command :

chmod 700 pvt16.key

- Now we can use the key as a pass for bandit17 level.

command:

ssh -i pvt16.key bandit17@localhost

This was pretty much a hard level for me as I am new to nano editor and chmod command…i reffered some videos for passing this level so yeah I guess the levels are getting hard for a beginners but lets see the next one!

Level 17->18 :

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.

Command:

diff passwords.new passwords.old

Output:

42c42

< kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

– -

> w0Yfolrc5bwjS4qw5mq1nnQi6mF03bii

Explanation:

- Lines preceded by a < are lines from the first file.
- Lines preceded by > are lines from the second file.
- Now for 42c42 means we need to change line 42 in first line that need to be changed to match line number 42 in second file.
- As mentioned in the question the password for the next level is in file 1 (passwords.new) and thats the only line we need to change from file 2(passwords.old) so we have to submit the line from password.new

💡 i.e *kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd* for bandit18 and it will return “Byebye ! Connection to bandit.labs.overthewire.org closed.” Stay tuned for the next level!

Level 18->19:

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

💡 **First thought:** We need to remodify the .bashrc so we can log in the server and using cat command we can find readme file for password! But how can we remodify the bashrc file?

Explanation:

- Its a long process and I don’t know how to do that so we have one solution by ignoring the pseudo terminal which uses the bashrc file to log us in. We have a command called

ssh -T bandit18@bandit.labs.overthewire.org -p 2220

The above command will ignore the force allocation of psuedo terminal.

After that we can use ‘ls’ ‘cat readme’ to read the password.

💡 IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x – password for bandit19

Level 19->20:

To gain access to the next level, you should use the setuid binary in the home directory. 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.

💡 **First thought**: As mentioned in the question we need to use the setuidd binary in home directory.

Explanation:

- We get a file bandit20-do which can be executed by bandit20 user and its moded with some permissions so that bandit19 user can't use it to see the pass of the next level.
- So we run '*./bandit20-do*' and it returns that Run a command as another user like '*./bandit20-do id*'. I didn't get it what it is, so I run the command and it returned the euid of bandit20 which is already set in the directory so we can then go on using following command: ./bandit20-do cat /etc/bandit_pass/bandit20

and we get the password :

💡 GbKksEFF4yrVs6il55v6gwY5aVje5f0j bandit20 lvl19

Level 20->21:

There is a setuid binary in the home directory that does the following:

- It makes a connection to localhost on the port you specify as a command line 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).

💡 First thought: We need to see the help commands given which are: *ssh, nc, cat, bash, screen, tmux, Unix 'job control'* (*bg, fg, jobs, &, CTRL-Z*, …)

Explanation:

- So I connected to bandit20 and then used ls to see what we have there and then got file named '*suconnect*' which is an setuid in the home directory.

- We need to listen to the port for the request send to that ssh id and when we enter the password to login from 2nd terminal we will get the password for the next level in our main terminal where we are listening for the port request becuase it is mentioned in the question that it will read. a line of text from the connection and compares it to the password and if it is correct it will transmit the password for next level.

Lets try it now!

- To set up the listening port we will use netcat command with instructions of -l to listen -v to get more information -p to enter the port for listening.

The command will be as follow:

'*ncat -lvp 9999*'

here 9999 is a random port we need to listen and then we need to login in the other terminal or second terminal by

'*ssh bandit20@bandit.labs.overthewire.org -p 2220*'

- Now doing that will send the connection request on the first terminal where we are listening to the port 9999 then in the second terminal we need to use '*./suconnect 9999*' so that the port is connected on both end and now we when we enter the password for bandit20 in first terminal then in the second terminal as the password will match it will send the password for the next level which is as follow:

gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr bandit21 lv20

Level 21->22:

A program is running automatically at regular intervals from cron, the time-based job scheduler. Look in /etc/cron.d/ for the configuration and see what command is being executed.

💡 **First thought**: I don't know what cron is so search cron and see what it is and what it does…so I did searched cron on internet and 'Cron' allows Linux users to run commands or scripts at a given date and time. You can schedule scripts to be executed periodically. For commands that need to be executed repeatedly (e.g., hourly, daily, or weekly), you can use the crontab command. The crontab command creates a crontab file containing commands and instructions for the cron daemon to execute. You can use the crontab command with the following options:

'*crontab -a filename*'

Install filename as your crontab file. On many systems, this command is executed simply as crontab filename (i.e., without the -a option).

'*crontab -e*'

Edit your crontab file, or create one if it doesn't already exist.

'*crontab -l*'

Display your crontab file.

'*crontab -r*'

Remove your crontab file.

'*crontab -v'*

Display the last time you edited your crontab file. (This option is available on only a few systems.)

'*crontab -u user*'

Used in conjunction with other options, this option allows you to modify or view the crontab file of user. When available, only administrators can use this option.

Explanation:

- First command cd.. to go in home directory and then I wrote 'cd /etc/cron.d' to get into the directory. After getting there I checked the contents by using 'ls' command

and then I found a file named ‘cronjob_bandit22' so I tried to read it by using ‘cat cronjob_bandit22.sh’ but as I had no permission it wasn’t executing.

- After that I tried ‘cat /etc/cron.d/cronjob_bandit22' to see what’s in the folder and that is where I found this bunch of content:

“@reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null

- * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null”

here the cool thing is we got a directory path of bandit22.sh to read so then I executed ‘cat /usr/bin/cronjob_bandit22.sh’ as I entered the file the chmod command already got executed and it let me read the contents of the files:

- “#!/bin/bash
- chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
- cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv”
- here we can see in the third line that in we have a a directory path named ‘/tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv’ so I tried to read the content of the directory and then boom we got our password.

Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI bandit22 lv21

Level 22->23

We need to do exact same thing as last level but this time we need to run some shell script as mentioned in the question.

First thought: I did exactly same thing as before but I just changed bandit22 to bandit23 for password of bandit23. I got stuck at running scripts as I don’t know how to run them but still i tried. And my oh my I did get the password. (And I learnt how to run shell script)

Explanation:

1. ‘cat /etc/cron.d/cronjob_bandit23'
2. @reboot bandit23 /usr/bin/cronjob_bandit23.sh. &> /dev/null
3. * * * * bandit23 /usr/bin/cronjob_bandit23.sh. &> /dev/null
4. ‘cat /usr/bin/cronjob_bandit23.sh’

```
#!/bin/bash
myname=$(whoami)
mytarget=$(echo I am user $myname | md5sum | cut -d ‘ ‘ -f 1)
echo “Copying passwordfile /etc/bandit_pass/$myname to /tmp/$mytarget”
cat /etc/bandit_pass/$myname > /tmp/$mytarget
```

5. cat /etc/bandit_pass/bandit22 > /tmp/ryx
6. cat /tmp/ryx

Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

But wait isn’t this the same password of this level? Yep it is! So i recheck where I got this wrong and then I knew I am going all the way wrong executing the script. The script execution must be like this

‘*echo I am user bandit23* | *md5sum* | *cut -d ‘ ‘ -f 1*'

Return: 8ca319486bfbbc3663ea0fbe81326349

Now the output which returned from running script for user bandit23 instead of bandit22 we got the md5sum of bandit23 which means we need to manually do the next step which is…

Return:

bandit23 lv22 and this is the real password.

Level 23->24

We need to write our own shell script to run.

Now the catch really here is that after a minute the script will be executed and in the next minute it will be deleted. So lets see what can we do.

Explanation:

All the same steps like last one till step no.2(change from bandit23 to bandit24

After executing cat /usr/bin/cronjob_bandit24.sh we will get

```
#!/bin/bash
myname=$(whoami)
cd /var/spool/$myname
echo “Executing and deleting all scripts in /var/spool/$myname:”
for i in * .*;
do
 if [ “$i” != “.” -a “$i” != “..” ];
 then
 echo “Handling $i”
 owner=”$(stat – format “%U” ./$i)”
 if [ “${owner}” = “bandit23" ]; then
 timeout -s 9 60 ./$i
 fi
 rm -f ./$i
 fi
done
```

- This is the script stored in the bandit24.sh file which states that whichever script we will write will be executed for one minute and then get removed from the directory.
- So in last step we got the script and the script runner line too but here we have whole script but we need to mention the script runner line to run the script. How can we do that?
- First thought I got was that just run the scriptrunner in terminal in /var/spool but we need to make a directory first to get the password in that file after a minute of execution.

Lets do it in following steps.

Steps:

1. return = bandit23

2. myname=bandit24

return = makes myname to bandit24

3. cd /var/spool/bandit24

return = let you in the directory

4. mkdir /tmp/lvl23

return = makes a directory mentioned

5. chmod 777 /tmp/lvl23

return = gives every user permission to read that file in the directory mentioned as first 7 is for the author, second 7 is for group, third 7 is for everyone.

6. ls -l /tmp/lvl23

return= to check if the permissions are applied or not

7. nano get.sh

return = to create a file named ‘get.sh’ to be created so that we can run our script.

8.

```
#!/bin/bash
```

```
cat /etc/bandit_pass/bandit24 > /tmp/lvl23/passwords.txt
```

return = We created our script which will read data from bandit24 folder and returns or copy the data in passwords.txt file located in the directory we created in step 4 (to save it in nano CTRL+O,ENTER & CTRL+X)

9. chmod +x get.sh

return= to make it executable

10. date

return = the exact time with seconds too so when we hit new minute the data will be retrieved in the file named ‘passwords.txt’

11. cd /tmp/lvl23

return= go into that directory

12. ls

return = after writing of data gets completed we can cat passwords.txt to get the password for next level.

UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ bandit24 lvl23

Level 24->25

A daemon is listening on port 30002 and will give you the password for bandit25 if given the password for bandit24 and a secret numeric 4-digit pincode. There is no way to retrieve the pincode except by going through all of the 10000 combinations, called brute-forcing.

> 💡 **First thought:** We are going to encounter our first bruteforcing. So what is daemon? Lets search it.

▼ Daemon

A program that runs continuously and exists for the purpose of handling periodic service requests that a computer system expects to receive. The daemon program forwards the requests to other programs (or processes) as appropriate. Each server of pages on the Web has an HTTPD or Hypertext Transfer Protocol daemon that continually waits for requests to come in from Web clients and their users.A daemon is a program that runs by itself directly under the operating system whereas a demon is part of a larger application program.

Explanation:

So we can use netcat to connect to the server.

Command:

'*nc localhost 30002*'

Return: I am the pincode checker for user bandit25. Please enter the password for user bandit24 and the secret pincode on a single line, separated by a space.

- Now we need to enter that 4 digit combination code which we need to bruteforce. So that 4 digit code can be from 0–1000 so we need to write a script that checks every 4 digit combination and if that is right then that will return back the password for next level.

So lets write the script shall we?

- But before that create an active directory like last time and give the permission and all like last time and create a file ‘nano brute.sh’

In nano editor we can write the script.

```
!#/bin/bash
for i in {000..9999}
do
echo “UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ $i”
done
```

- Now here we are running a for loop where we are going from 1 to 9999 four digit numbers one by one and then echo line will write the password and the number simultaneously.
- Once we run it we can cat that all data into one file “bruteforce.txt”

Command:

./brute.sh > bruteforce.txt

Once done we can pipe the network host with bruteforce.txt in here to get next level pass.

Command:

cat bruteforce.txt | nc localhost 30002

The command will run on and when the 4 digit passcode match with the passcode checker it will return the password for bandit25

uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG bandit25 lv24

Level 25–26

We get to log in with the passcode we retrieved last time and not we need to log in to the server with the ssh key provided in the directory.

Command:

ssh -i bandit26.sshkey bandit26@localhost

- After connecting to bandit26 we get following error: Connection to localhost closed.

- Now as mentioned in the question “ The shell for user bandit26 is not /bin/bash, but something else. Find out what it is, how it works and how to break out of it.”
- We gotta do something here like breaking into the whole server? I don’t have any idea but after going in I did try to find the home directory by using command cd .. and ls -l to see the directories but we can’t access them. So what can we do here? We have some list of commands for help in the question but we already know what they do but there are 2 commands which highlights out and they are more and vi command which we can search on the google what they do.
- ‘more’ command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large (For example log files)
- ‘vi’ is the vi editor elaborated as visual editor. It is installed in every Unix system.

Explanation:

Now here is the truth…I literally watched 2 videos from Hackersploit and John Hammond and still have some doubts. I did all the steps shown in the video because this one was way tricky for me and that is why if you don’t get any of these then I highly recommend you to watch some videos. So here we go..

- Once we log into the server with ssh key we get an error that connection is closed to localhost but we can see an ASCII art just above it so we can do some steps here like:

S-1: Minimize the window so that you can see more command to see the files.

S-2: Now log in again via ssh key and then you can see the more command to see more text from ASCII image.

S-3: Now press ‘v’ to enter vim editor.

S-4: Now write :

:r /etc/bandit_pass/bandit26

hit ENTER.

S-5: Maximize the window and you can see the password in between the lines of ASCII image.

S-6: To get the shell for the ssh we need to set the shell by using the following command:’:set shell=/bin/bash’

S-7: To activate the shell we need to write ‘:shell’

5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z bandit26 lv25

> NOTE: This way indeed a CTF solving and was a bit tough for me but I am ready to grind more so lets go.

- Once we get the shell we can go right away to the next level 26->27 where we need to just retrieve the password.
- Now here we have a bandit27-do file which we need to cat as user bandit27 so we can use this command:

./bandit27-do cat /etc/bandit_pass/bandit27

💡 3ba3118a22e93127a4ed485be72ef5ea bandit27 lv26

Level 27->28

There is a git repository at ssh://bandit27-git@localhost/home/bandit27-git/repo. The password for the user bandit27-git is the same as for the user bandit27.

Clone the repository and find the password for the next level.

💡 **First thought:** So we need to clone the given repo and then cat the password. But how? Research about how to clone the repo.The git pull command is used to get updates from the remote repo. This command is a combination of git fetch and git merge which means that, when we use git pull, it gets the updates from remote repository (git fetch) and immediately applies the latest changes in your local (git merge)…blah blah blah so we can use git clone to clone the repo right? Lets see

Explanation:

S-1: We need to create an directory or we can just cd the previous directory but for the sake of git (lmao), I created a new directory by using *mkdir /tmp/newryx*

S-2: *cd /tmp/newryx*

S-3: Now for cloning so yeah I used

`git cloning ssh://bandit27-git@localhost/home/bandit27-git/repo`

and yeah I cloned the repo here(password is same as bandit27)

S-4: Now I got into the directory cd repo and then cat the README file located there and boom we got our password…The password to the next level is:

> 💡 0ef186ac70e04ea33b4c1853d2526fa2 bandit28 lv27

Level 28->29

There is a git repository at ssh://bandit28-git@localhost/home/bandit28-git/repo. The password for the user bandit28-git is the same as for the user bandit28.

Clone the repository and find the password for the next level.

> 💡 **First thought:** Isn't this the same problem like last time? Lets follow the last problem steps and see what happens.

Explanation:

S-1: All the steps just like last problem (don't forget to change the names & password)

S-2: Now the thing is we got a file named README.md and when we cat that file we get…

"

Bandit Notes

Some notes for level29 of bandit.

credentials

- username: bandit29
- password: xxxxxxxxxx
- "

- So the file just contain this data only and the password is very small and it has some kind of encryption involved (I guess)

So I tried to see the branch of our file by

g*it branch*

Return: I got a our origin branch name ‘master’.

- I tried several commands like ‘git status’ to view status of the branch, ‘git checkout -b master’ to switch the branch but I guess nothing was working.
- I even tried to see the head branch by using ‘*git branch -a*’ and it reverted back

`remotes/origin/HEAD -> origin/master`

`remotes/origin/master`

S-3: Well I am forgetting one big thing which I shouldn’t is that checking logs…by ‘*git logs*’

S-4: Now we have some logs and what we can do is see the first one which is latest commit and middle one is the change before releasing the latest commit and the last one is the initial commit.

So I opened up initial commit by using ‘*git show de2ebe2d5fd1598cd547f4d56247e053be3fdc38*' and in that file the password was to be decided.

S-5: I opened the middle commit by

git show c086d11a00c0648d095d04c089786efef5e01264' and boom we have our password.

💡 bbc96594b4e001778eee9975372716b2 bandit29 lv28

> NOTE: Sometime when we are doing hard stuff, don’t forget the basics XD.

Level 29->30

Its the same as last 2 problems with new address so lets reach till we cat the file and then think what to do next.

After doing all we get :

Bandit Notes

Some notes for bandit30 of bandit.

credentials

- username: bandit30
- password: <no passwords in production!>

 **First Thought:** So the obvious thing we can do is see the logs… so lets do it.

Explanation:

- So I opened the initial commit and I got the same thing just the username was bandit29 and in the next commit the username was changed from bandit29 to bandit30.
- The first thought got struck in my head was for both the bandit29 and bandit30 the passwords are same because the file is telling us that no password in production. So I tried doing that, but I was wrong and the passwords are indeed different.
- But what we were doing in last step maybe it can be useful here, like we can dive into new branch by using '*git branch -a*' and see what branch you are in and we can dive into the developer branch by using

'*git checkout /remote/origin/dev*'

and there we can check the logs and boom we got the password in the latest commit.

 5b90576bedb2cc04c86a9e924ce42faf bandit30 lv29

> NOTE: Use google as much as you can to search what one command does and try it practically in a test terminal.

Level 30->31

We have to follow same path as earlier till reading the file.

- After reading the file we get “just an epmty file… muahaha” …so I checked logs and it was the only commit and then I checked branch and no branch was attached here.

I checked status and it was working good. So what we have here is the file which is roasting us but what can we do here?

- I tried to use ‘push’ & ‘pull’ command to see if anything needs to be updated or not by using :

git pull origin HEAD:master

and

git push origin HEAD:master’

but it was already up to date. What now?

- So nothing was working out and the repo was empty but let me check is there any hidden file in the folder by ls -al and yup it had one and then I got the .git directory.
- I opened it and there was several folders and files but one that was catching my eye was ‘packed-ref’…so I read that file and it has indeed 2 hash one is initial commit and other one was tagged ‘secret’ so I chose to see the hash and then I got my password.

47e603bb428404d265f59c42920d81e5 bandit31 lv30

Level 31->32

We need to push the file to the remote directory (I am reading the content of README.md and before that you need to complete all the steps just like last time)

- So we will use the power of knowledge we used in last question(its always like that) So for pushing the content we can use

git pull origin HEAD:<directory>

- Lets do this fellas!

So README.md says this:

- This time your task is to push a file to the remote repository.

Details:

. File name: key.txt

. Content: ‘May I come in?’

. Branch: master

- So with ‘*nano key.txt*’ I created a file and then wrote ‘May I come in?’ and then tried to add the file by using command ‘*git add key.txt*’ but as .git-ignore was ignoring the .txt so we need to append -f at the end which says *‘git add key.txt -f’* and it will add it to the repository and now we need to push it but before that we need to do a commit so write ‘git commit’ and then write out whatever commit message you want to and then push it by using ‘*git push*’ and then add the passwords and then you’ll get this:

```
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 324 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files… ####
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
To ssh://localhost/home/bandit31-git/repo
! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to ‘ssh://bandit31-git@localhost/home/bandit31-git/repo’
```

In between the line we have the password and it is:

56a9bf19c63d650ce78e6ec0354ee45e bandit32 lvl31

Level 32->33

After all this git stuff its time for another escape. Good luck!

First thought: Its a CTF like last time and it will take immense time but try this time to break it on your own. Come on come on, chop chop!

We go in and we see Uppercase Shell and uhh… idk what uppercase shell is…(lol)

- Well I saw a video regarding this and found a solution where we can just write “$0" in the uppercase shell and we can then give the shell commands to it just like before.

So I then tried to see the password of bandit33 by command

‘*cat /etc/bandit_pass/bandit33*'

c9c3199ddf4121b10cf581a98d51caee bandit33 lvl32

> NOTE: If you don’t see a light in room then go search for sun outside the room (lol just made it out and idk why I put it here XD)

- I solved this all level on 24th of the July,2021 and we don’t have bandit33 level so I guess we need to wait but grind must not be stopped. So I will try some more wargames from OTW soon.

Huge Help:

Hackersploit, JohnHammond, Zulu CaPwn…I highly recommend you this 3 YouTube channels for any query regarding Bandit Wargames. I learnt a lot from them and yeah one more thing just don't see them blindly, try to search the command listed in the 'Helpful' section below the questions and once you get that thing then go for it.

GG! Bye Bye! See you all soon!