Over the Wire – Bandit

Link to the challenge:
https://overthewire.org/wargames/bandit/

Level 0: first run 'ls' command to inepct file list, then cat the found file.

Level 1: here it is required to open dash file name, it is done with 'cat < -'

Level 2: here it is required to open file with spaces in it, so we have to put it in double quotes -> 'cat "spaces in this filename"'

Level 3: here it is required to a. enter 'inhere' directory with cd command, and b. run 'ls -a' within the directory that -a flag is for hidden items. C. cat the found file for the password.

Level 4: entry password:
2EW7BBsr6aMMoJ2HjW067dm8EgX26xNe

Here it is required to

a. Enter 'inhere' directory with 'cd inhere' command.
b. Run 'ls', here we see several dashed file.
c. Run 'find . -type f -exec file {} \; | grep -i text'
   In order to locate the human readable file.
   What this command does is to run 'file' command on all files in the directory and return the files which are of text type (readable)

Level 5: entry password:

lrIWWI6bB37kxfiCQZqUdOIYfr6eEeqR

Here it is required to run this command:

' find . -type f -size 1033c ! -executable -exec file {} \; | grep -i text'

Where:

- 'find .' is find file from current directory (.)
- '-type f' is searching for files
- '-size 1033c' is filter files by byte size of 1033, the c means bytesize.
- '! -executable' means non executable (! = not)
- '-exec file {} \; | grep -i text' means run file command on the files, and filter the output based on the word 'text', the '-i' flag is insensitive casing.

Level 6: entry password:

P4L4vucdmLnm8I7Vl7jG1ApGSfjYKqJU

Here it is required to find file that is owned by user bandit7, group bandit6, and has 33 bytes.

So the command to track the file is the following:

'find / -type f -size 33c -user bandit7 -group bandit6 2> /dev/null'

Where '2>/dev/null' means disregard errors (2 = std error file descriptor, > is the channeling, '/dev/null' is special file that discard all inputs)

Level 7: entry password:
z7WtoNQU2XfjmMtWA8u5rN4vzqu4v99S

The way to get the password is with the command

' cat data.txt | grep millionth' – filter the file content to the word 'millionth', and print the line.

Level 8: entry password:

TESKZC0XvTetK0S9xNwm25STk5iWrBvP

The command is:

sort data.txt | uniq -c | grep '1 '

where:

- 'sort data.txt' sorting all the lines within the file
- 'uniq -c' display only unique lines (no resentation of dupliqates) and add the number of times each line appears.
- grep '1 ' output only the lines which the counter value is 1.

Level 9: entry password:

EN632PlfYiZbn3PhVK3XOGSlNInNE00t

Here the command is ' strings data.txt | grep ='

After this the password is easy to find.

Level 10: entry password:

G7w8LIi6J3kTb8A7j9LgrywtEUlyyp6s

To decode the base64 data we use:

'base64 -d data.txt' where -d flag is decode.

Level 11: entry password:

6zPeziLdR2RKNdNYFNb6nVCKzphlXHBM

The way to solve this problem is this command:

"cat your_file.txt | tr 'a-zA-Z' 'n-za-mN-ZA-M'"

'tr' – translates, the linux command that is used for rotation operations.

Level 12: entry password:

JVNBBFSmZwKKOP0XbFXOoW8chDz5yVRv

The first action to do is to run xxd

' xxd -r data2.txt > data2.bin'

To convert the hexdump to binary file.

Then it is needed to check the output file with

'file outputfile'

Check the compression format

And rename the file to a file with the matching suffix.

Like 'mv outputfile file.gz' for gzip compression.

Then 'gzip -d file.gzip' where -d is for decompression.

Same with bzip2:

'bzip2 -d output.bz2'

And tar:

'tar -xf output.tar'

Level 13: entry password:

wbWdlBxEir4CaE8LaPhauuOo6pwRmrDw

The command to connect to bandit 14 is

'ssh -i sshkey.private
bandit14@bandit.labs.overthewire.org -p 2220'

Now that we connected to user bandit14 via ssh connection from bandit 13, we insert command

Command ' cat ../../etc/bandit_pass/bandit14'

This file can be accessed only by bandit14, this is why it was necessary to ssh connect to bandit 14 from bandit 13.

Level 14: entry password:

fGrHPx402xGC7U7rXKDaxiWFTOiF0ENq

the command is:

' echo "fGrHPx402xGC7U7rXKDaxiWFTOiF0ENq" | nc localhost 30000'

Where we send the current level password via netcat on localhost, port 30000.

And then we get the next level password from the server.

Level 15: entry password:

jN2kgmIXJ6fShzhT2avhotn4Zcka6tnt

the command to send ssl message:

' echo "jN2kgmIXJ6fShzhT2avhotn4Zcka6tnt" | openssl s_client -connect localhost:30001 -ign_eof'

Level 16: entry password:

JQttfApK4SeyHwDlI9SXGR50qclOAil1

First: nmap scan with -sV flag for service detection:

'nmap localhost -p 31000-32000 -sV'

Here we find the port '31790',

The next command is:

' echo "JQttfApK4SeyHwDlI9SXGR50qclOAil1" | openssl s_client -connect localhost:31790'

The output is RSA key, it needs to be saved in file, then its permissions set to 400 with the command:

'chmod 400 ../../../tmp/key1/key.pem'

Then access to bandit17 with the key with the command:

'ssh -i ../../../tmp/key1/key.pem bandit17@bandit.labs.overthewire.org -p 2220'

That gives access to bandit17

Level 17: no entry password.

First run:

'diff passwords.old passwords.new'

To notice the differences between the lines.

Then run 'cat passwords.new | grep {result}'

To confirm.

Level 18: entry password:

hga5tuuCLF6fFzUpnagiMN8ssu9LFrdg

when normally attempting to login, the server immediately throws the user out.

So to the connection command it is required to add 'cat readme', like this:

'ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme'

Level 19: entry password:

awhqfNnAbc1naukrpqDYcF95h7HoMTrC

in here there is a file with setuid active, meaning any user can run the file with owner privilges.

So it can be used to run the password file 'bandit20' in 'bandit_pass', with the command:

' ./bandit20-do cat ../../etc/bandit_pass/bandit20'.

Level 20: entry password:

VxCazJaVykI6W36BkBU0mJTCM8rR95XT

There is file 'suconnect' which used as client.

Basically it needs to receive from the server the current password, and then returns the next level password.

So a server has to be made.

It can be used with several ways, python socket, or netcat.

I used netcat with the command:

' nc -l -p 12345'

Then I ran the client './suconnect 12345'

Then on server I entered the current password, it got sent to client, and got the next level password.

Level 21 entry password:

NvEJF7oVjkddltPSrdKEFOllh9V1IBcq

a. run ' ls ../../etc/cron.d' to observe what run on cron.d (d=daily)
b. run ' cat ../../etc/cron.d/cronjob_bandit22'

the output of this command is this:

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

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

Meaning the cron.d process runs this:

'/usr/bin/cronjob_bandit22.sh'

c. run 'cat ../../usr/bin/cronjob_bandit22.sh'

the output is this: 'chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

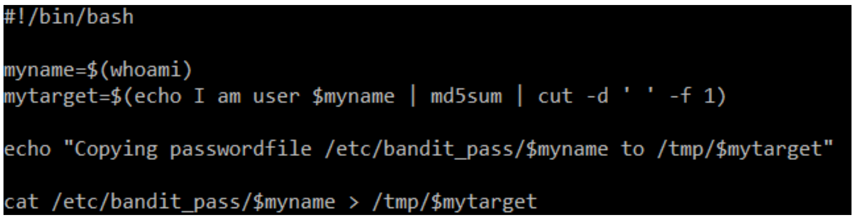
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv'

it means the password is being stored in the tmp file.

d. Run 'cat ../../tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv' And get the password.

Level 22: entry password: WdDozAdTM2z9DiFEQ2mGlwngMfj4EZff

a. Run 'ls ../../etc/cron.d'
b. Run 'cat ../../etc/cron.d/cronjob_bandit23'
   Just like the previous challenge – it tells us to go here: ' bandit23 /usr/bin/cronjob_bandit23.sh'
c. Run ' cat ../../usr/bin/cronjob_bandit23.sh'
   The output is:

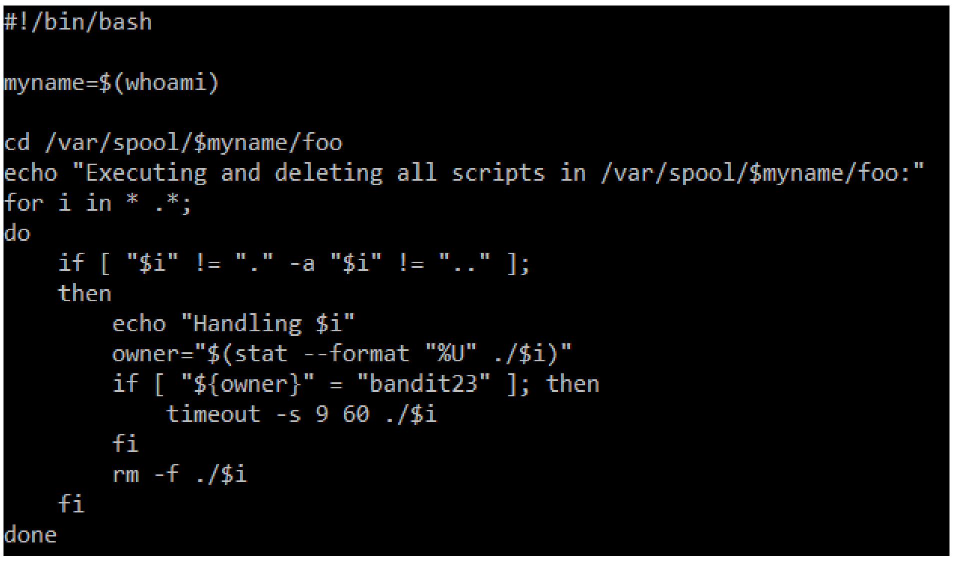


```
#!/bin/bash

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
```

   It takes the username and make a file on tmp based on the hash of said username, storing the next level password in it.
   The problem is running this script will work for the current user, not good. We need it for next user.
   We cant vim the vile so we will have to get the has manually by running manually the commands:
   'myname=bandit23'
   'echo I am user $myname | md5sum | cut -d ' ' -f 1'
   Then run
   'cat /tmp/8ca319486bfbbc3663ea0fbe81326349'
   Which the filename is the md5 hash output.

Level 23: entry password:
QYw0Y2aiA672PsMmh9puTQuhoz8SyR2G

a. Run 'cat ../../etc/cron.d/cronjob_bandit24'
   The output instructs ' bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null'
b. Run 'cat ../../usr/bin/cronjob_bandit24.sh'
   The output is:

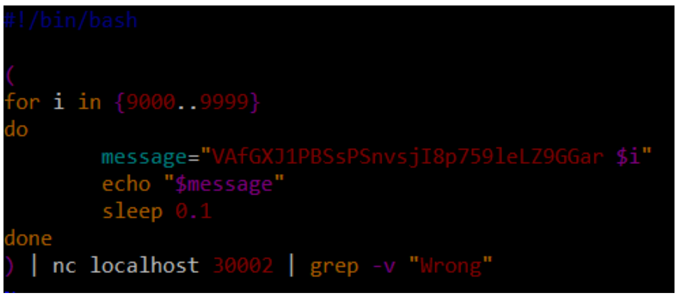


```
#!/bin/bash

myname=$(whoami)

cd /var/spool/$myname/foo
echo "Executing and deleting all scripts in /var/spool/$myname/foo:"
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
```

   It basically runs and delete everything placed in the path '/var/spool/banditxx/foo'
c. Create working directory environment: 'mkdir ../../tmp/amit_test'
d. Create files within the directory 'costumshell.sh' And 'password'
e. Within 'costumshell.sh' write with vim the following bash script: '#!/bin/bash
   cat /etc/bandit_pass/bandit24 > tmp/
   amit_test/password'
f. Give proper writing permissions to it with 'chmod 777 tmp/ amit_test/password'

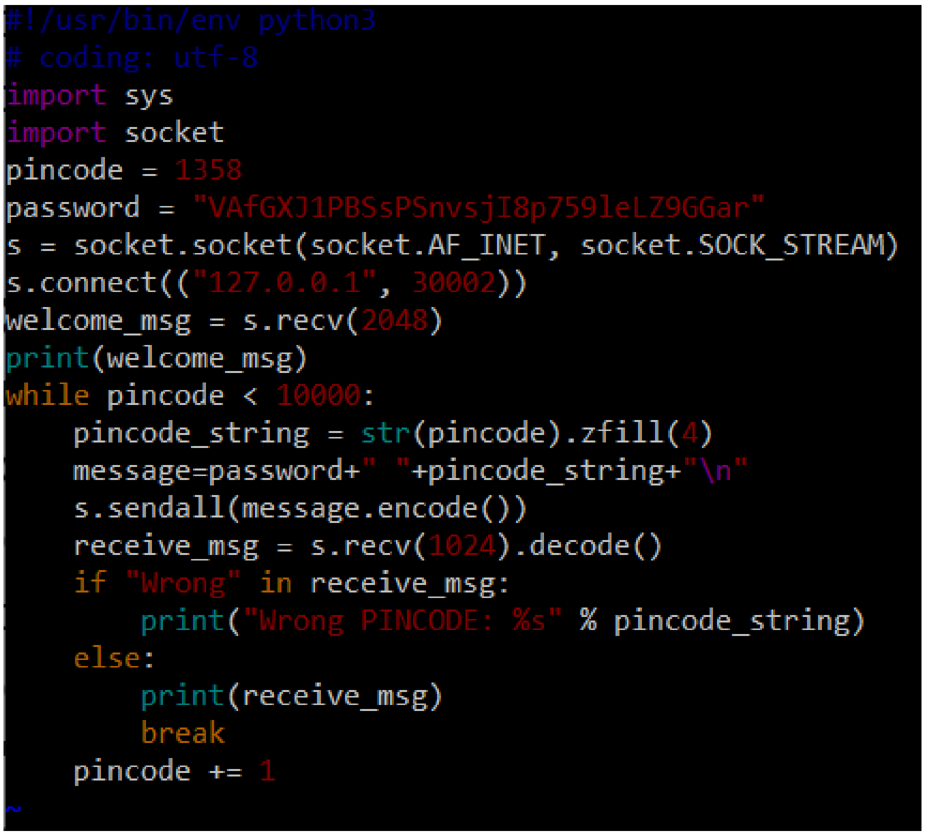
g. Give proper execution permissions to the customshell.sh
h. Copy the costumshell to spool/bandit24, where the cronjob_bandit24.sh will automatically run it at some point with: ' cp /tmp/amit_test/customshell.sh /var/spool/bandit24/foo'
i. Extract the password from '/amit_test/password' with cat command.

Level 24: entry password:
VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar
In here I used bash script that automatically sends the command
'echo "VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar 2648" | nc localhost 30002' from value 0000-9999.
It took some time, eventually the right number was 9014'
The bash script:



```
#!/bin/bash

(
for i in {9000..9999}
do
        message="VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar $i"
        echo "$message"
        sleep 0.1
done
) | nc localhost 30002 | grep -v "Wrong"
```

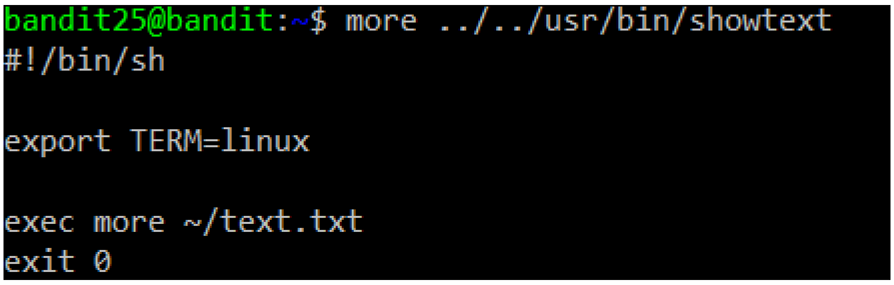
Also python can be used:



```
#!/usr/bin/env python3
# coding: utf-8
import sys
import socket
pincode = 1358
password = "VAfGXJ1PBSsPSnvsjI8p759leLZ9GGar"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1", 30002))
welcome_msg = s.recv(2048)
print(welcome_msg)
while pincode < 10000:
    pincode_string = str(pincode).zfill(4)
    message=password+" "+pincode_string+"\n"
    s.sendall(message.encode())
    receive_msg = s.recv(1024).decode()
    if "Wrong" in receive_msg:
        print("Wrong PINCODE: %s" % pincode_string)
    else:
        print(receive_msg)
        break
    pincode += 1
~
```

Level 25: entry password:
p7TaowMYrmu23Ol8hiZh9UvD0O9hpx8d

a. Go to /etc/passwd – there you can observer that the shell is not the usual /bin/bash but '/usr/bin/showtext'
b. There you can see



```
bandit25@bandit:~$ more ../../usr/bin/showtext
#!/bin/sh

export TERM=linux

exec more ~/text.txt
exit 0
```

c. It basically means that the window size must be reduced in order to log in to bandit26 with 'ssh -i bandit26.sshkey bandit26@localhost -p 2220'
d. Then vim must be opened with 'v' command, and then from vim we can open another file with the command ':e /etc/bandit_pass/bandit26'
   Where ':e' is the command itself
e. We still need to open shell, so we will enter on vim: ':shell' and then ':set shell=/bin/bash'
   And then again ':shell' to start the shell

Level 26 entry password:
c7GvcKlw9mC7aUQaPx7nwFstuAIBw1o1
Just like level 19 – run './bandit27-do cat /etc/bandit_pass/bandit27'

Level 27 entry password:

YnQpBuifNMas1hcUFk70ZmqkhUU2EuaS