CRYTONITE WRITEUP

OverTheWire BANDIT Writeup

Level 0

Dual booted with ubuntu

Installed ssh and enabled it

Then used ssh command to connect with bandit

ssh [bandit0@bandit.labs.overthewire.org](mailto:bandit0@bandit.labs.overthewire.org) -p 2220

Level 0 to level 1

Used cat to display password in content of readme

cat readme

Logged in into bandit1 using ssh command again using password in readme

Level 1 to level 2

Googled about dash command

Used cat ./- to display the content in file -

Level 2 to level 3

Used / before every file name after searching about it on google

Level 3 to level 4

Used ls to list directories

Then changed current directory to inhere

Viewed the hidden file in inhere using -a command

ls

cd inhere

ls -a

cat .hidden

Level 4 to level 5

Found file type using file ./\*

Found ascii text file

Used Cat ./-file07 for showing password

Level 5 to level 6

Changed current directory to inhere

Used find -size 1033c to find file

Changed to directory maybehere07

Used find -size 1033c ! -executable

Used file command to find file type

By above 2 statement confirmed all 3 conditions

Level 6 to level 7

 Googled for how to search in whole server

Used find / -size 33c -user bandit7 group bandit6

Used cat /var/lib/dpkg/info/bandit7.password

Level 7 to level 8

Used grep command

grep ‘millionth’ data.txt

Level 8 to level 9

Learned about piping

Tired cat data.txt| uniq -c but failed as got to know uniq only search for unique  adjacent  to it

Hence used sort for finding it

Sort data.txt | uniq -c

Level 9 to level 10

Used strings to find human readable strings in binary file

Strings data.txt | grep ‘=’

Level 10 to level 11

Searched on google about base64

Decoded the file using base64 -d data.txt

Level 11 to level 12

Learned about rot13 through link given on level 11 to level 12 page

Used tr to rotate

Cat data.txt | tr ‘A-Za-z’ ‘N-ZA-Mn-za-m’

Level 12 -13

xxd program is used to make a hexdump or to do the reverse. Option -r convert hexdump into the binary. File myfile.txt is a hexdump and convert it into a binary file myfile1.bin using command

Using command file myfile1.bin , we found that myfile1.bin is a *gzip compressed data*.

zcat is a program supplied with gzip and is used to decompress *gzip compressed files*.

Again using file command on myfile2, we found that it is *bzip2 compressed data*.

bzcat program is supplied with bzip2 and is used to decompress *bzip2 compressed files*.

myfile3 is gzip compressed file so use zcat program to decompress it in myfile4. myfile4 is a POSIX tar archive.

tar program is used for archiving file and options x is used to extract an archive, f is used to specify name of the tar archive and v is used for more detailed listing.

This command outputs file data5.bin which is again a tar archive. Again use tar program on data5.bin which outputs data6.bin. data6.bin is a bzip2 compressed file and use bzcat program to decompress it to myfile7.

myfile7 is a tar archive and use tar program which outputs data8.bin. data8.bin is a gzip compressed file and use zcat to decompress it to file myfile9

l

level 13-14

Command to connect remote host : ssh bandit13@bandit.labs.overthewire.org -p 2220

To access password we need to login as bandit14 and for that the host is localhost. In the directory /home/bandit13 file sshkey.private contains SSH key to login into bandit14.

ssh command is used to login and execute commands on a remote machine. Option -i selects a file from which the identity (private key) for RSA or DSA authentication is read and the file sshkey.private

The password is stored in /etc/bandit\_pass/bandit14.

The command is cat /etc/bandit\_pass/bandit14

Level 14-15

netcat is a simple unix utility which reads and writes data across network connections, using TCP or UDP protocol.

nc host port creates a TCP connection to the given port on the given target host. Your standard input is then sent to the host, and anything that comes back across the connection is sent to your standard output

level 15-16

OpenSSL comes with a client tool that you can use to connect to a secure server. The tool is similar to telnet or nc, in the sense that it handles the SSL/TLS layer but allows you to fully control the layer that comes next.

To connect to a server, you need to supply a hostname and a port. For example: $ openssl s\_client -connect [www.feistyduck.com:30001](http://www.feistyduck.com:30001)

level 16-17

Nmap (“Network Mapper”) will help us to scan for ports. Option -p provides the port ranges to the tool.

We found two ports and port 31790 is open. Open port means server on that port is listening.

Now to connect to port 31790 we will use openssl with s\_client tool, which we have done in previous challenge.

The output we received is a RSA private key.

Save this key locally on your computer with name bandit17.key.

level 17-18

Command to login ssh -i bandit17.key bandit17@bandit.labs.overthewire.org -p 2220

diff program compare files line by line and option --normal output a normal diff.

Command to find difference between passwords.new and passwords.old

level 18-19

We receive Byebye ! as the output. In the question it is mentioned that someone has modified **.bashrc** to log you out when you log in with SSH.

With option -t in ssh command we can force psuedo-tty allocation.

“Pseudo Terminals” emulates Terminal hardware, handling input and output in the same way a physical device would so that the software connected is not aware there’s not a real device attached.

We have forced a psuedo terminal and we know that password is in the readme file, so we can view password using command.

level 19-20

**Setuid**, which stands for set user ID on execution, is a special type of file permission in Unix and Unix-like operating systems such as Linux and BSD. It is a security tool that permits users to run certain programs with escalated privileges.

When an executable file’s setuid permission is set, users may execute that program with a level of access that matches the user who owns the file.

When viewing a file’s permissions with the ls -l command, the setuid permission is displayed as an “**s**” in the “user execute” bit position.

Here setuid is set for the binary file (-rwsr-x--) and the user is bandit20. So if execute ./bandit20-do it will run as user bandit20 and password for bandit20 is stored in /etc/bandit\_paas/bandit20 which is only accessible through user bandit20.

level 20-21

First lets see how to create a TCP listener connection on our local machine. For this purpose we will use netcat program and option -l specify that nc should listen for an incoming connection rather than initiate a connection to a remote host. We will open two terminal and in first terminal the command is nc -l localhost -p 5000. The netcat server is listening to port 5000. -p specifies the source port nc should use.

Now in second terminal netcat client establishes connection to the server on port 5000. The command to do that is nc -v localhost 5000. Option -v give more verbose output

Now login to a remote machine as user bandit20.

Binary file suconnect has escalated privilege as user bandit21. After running command ./suconnect it shows how to use command i.e. Usage: ./suconnect

It also shows “This program will connect to the given port on localhost using TCP. If it receives the correct password from the other side, the next password is transmitted back

First we have to establish a TCP connection like we did above. We can connect on any port for eg. port 12345. Then we pass the password of current level (bandit20). And if the password is correct, it will display password for level 21.

In terminal 1 server will listen on port 12345 and password of the current level is passed

In terminal 2 binary file suconncet will connect to server on port 12345 and if password matches, password for next level is displayed on the terminal 1.

Here connection is also established on port 12346.

level 21-22

Cron is a system daemon used to execute desired tasks (in the background) at designated times.

A crontab file contains instructions for the cron(8) daemon in the following simplified manner: “run this command at this time on this date”. Each user can define their own crontab. Commands defined in any given crontab are executed under the user who owns that particular crontab.

In directory /etc/crond.d/, file important to us is cronjob\_bandit22. It is a crontab file and contain commands. The content can be viewed using cat cronjob\_bandit22

The command is executing the file cronjob\_bandit22.sh which is in the directory /usr/bin/. Lets see the content of file cronjob\_bandit22.sh using command cat /usr/bin/cronjob\_bandit22.sh

Now we know that file /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv contains the password for the next level. We can view the password using command cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

level 22-23

File cronjob\_bandit23.sh copies password of bandit22 in another file and it is readable and executable by user bandit22

So we cannot modify it. But we want to copy password of bandit23 instead of bandit

After executing cronjob\_bandit23.sh using ./cronjob\_bandit23.sh it generates a file in tmp directory and the name of the file is md5 hash containing bandit22.

If we change bandit23 to bandit23 then new hash is generated which is the name of new file in tmp directory which may contain password for level 23.

To generate new hash run echo I am user bandit23 | md5sum | cut -d ' ' -f 1 on terminal. The new hash is 8ca319486bfbbc3663ea0fbe81326349 and this is the name of file in tmp directory.

To see password run cat /tmp/8ca319486bfbbc3663ea0fbe81326349

level 23-24

Execute these commands to know the content of crontab file and then content of the script file.

cd /etc/cron.d/

ls

cat cronjob\_bandit24

cat /usr/bin/cronjob\_bandit24.sh

The command cd /vars/pool/$myname will change to bandit24 directory and then all scripts in directory bandit24 will be executed and then deleted. This is done in every 60 seconds as timeout is provided

So if we put our script in /var/spool/bandit24 then it will be executed and then deleted. So we will write our script such that it will output password for next level.

We will create a directory pcdir123 in /tmp directory and change the permissions to readable, writeable and executable for pcdir123.

Now create a script file pass24script.sh in /tmp/pcdir123 which will contain our script

pass24script.sh is not executable, so make it executable for all users using command chmod 777 pass24script

Copy this script in /var/spool/bandit/24 using command cp pass24script.sh /var/spool/bandit24

When new minute starts, file pass24file is produced with password for the next level if the directory pcdir123 has write permission enabled