OverTheWire Bandit Wargame Write-up

January 17, 2016
This is one of the easier wargames out there. The first ~22 levels have very few gotchas for experienced UNIX developers, and can take 30 minutes to a day. There are a few that I got caught up on though. It's a good way to learn the shell, or get more experienced with it.

Usually each stage will give you a password (and increment the username) to the next stage that you use to login with to get the next password and so on.

Check it out yourself here: http://overthewire.org/wargames/bandit/

Bandit Wargame

Bandit 0 → 1

Simply `ssh` into the server, with the username and password provided, and `cat readme`.

```
ssh bandit0@bandit.labs.overthewire.org # password bandit0
cat readme
```

Bandit 1 → 2

You need to give the full file path to read it when the file has a '-' in it.

```
cat /home/bandit1/-
```

Bandit 2 → 3

You can use autocomplete w/ the tab key to help you out with hard file names.

```
cat "spaces in this file name"
```

Bandit 3 → 4

You can see hidden dot files with `ls -la`.

```
cd /inhere/
ls -la
cat .hidden
```

Bandit 4 → 5

No need to think too much, you can `cat` out all the files and look for something that looks like a password.

```
cat /inhere/*
```

Bandit 5 → 6

I can't remember all the flags and options to `find`, so I had to look in `man find` to see how to specify a file size in bytes. Find is recursive by default.

```
find  . - size 1033c
```

Bandit 6 → 7

Just more flags to `find`, at the file system root.

```
find . -size 33c -user bandit7 -group bandit6
```

Bandit 7 → 8

Simple use of the pipe, `|` and `grep`.

```
cat data.txt | grep millionth
```

Bandit 8 → 9

You have to `sort` before using `uniq`, as `uniq` only looks at contiguous matches. Use the `-c` flag to display counts. Looking for the entry with only 1 count.

```
cat data.txt | sort | uniq -c
```

Bandit 9 → 10

Just get a list of the strings in the data set and look for the string beginning with ========.

```
cat data.txt | strings
```

Bandit 10 → 11

Simply use of base64 tool.

```
base64 --decode data.txt
```

Bandit 11 → 12

Simply using cat and copy paste into a rot13 convert convert online yields the answer. You can also use `tr` to do rot13.

Bandit 12 → 13

This one's a real drag. First get the hexdump into actual hex data and then do a series of `file` and different extractions like 7 times.

```
xxd -r data.txt
file <output file>
gunzip/bunzip2/tar xf over and over
```

Bandit 13 → 14

Simply specify the key to use to `ssh` with the `-i` flag.

```
ssh -i sshkey.private bandit14@localhost
```

Bandit 14 → 15

This one's tricky, because it requires you to read the MOTD that says "passwords are stored in /etc/somegame_pass". After that it's just nc localhost 30000 and enter the password of bandit14 from the directory.

Bandit 15 → 16

I had to google how to use `nc` with SSL for this one. Turns out `ncat`, another clone of `nc`, from `nmap` allows easy ssl.

```
ncat --ssl localhost 30001
```

Bandit 16 → 17

For this one, you get an ssh key that you have to feed to a port running on localhost in the range 31000-32000. There's only 5 servers, so just list them all and try them all. You get an ssh key in return, save it to a file and `chmod 600` it to use for the next level.

```
nmap localhost -p 31000-32000
```

Bandit 17 → 18

Just a simple `diff`.

```
diff passwords.old passwords.new
```

Bandit 18 → 19

Copy the file over using `scp`.

```
scp bandit18@bandit.labs.overthewire.org:/home/bandit18/readme .
```

Bandit 19 → 20

The setuid bin runs as bandit20 and executes anything given as them.

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

Bandit 20 → 21

Run a socket listener on one terminal, and use the `suconnect` binary as a socket client.

```
nc -l -p 6161 # On one terminal
./suconnect 6262 # On other terminal
```

Bandit 21 → 22

This one involves some guessing of what they want from you. Basically it's just look in `/etc/cron.d` and the `cronjob_bandit22.sh` file. The `cronjob` runs `/usr/bin/cronjob_bandit22.sh` which simply creates a tmp file with bandit22's pw. Read that file to get the next password.

Bandit 22 → 23

Same thing as the previous level, but look into `cronjob_bandit23.sh`, this one creates a file as well, but it's more obfuscated. You can simply redo what it does and get the filename.

```
 cat `echo I am user bandit23 | md5sum | cut -d ' ' -f 1`
```

Bandit 23 → 24

Another cronjob challenge. This one uses `cronjob_bandit24.sh` and executes all scripts in `/var/spool/bandit24`. Simply create a shell script that writes a file with bandit24's password. Be sure to `chmod +x` it.

```
#!/bin/bash
cat /etc/bandit_pass/bandit24 > /tmp/somefile
```

Bandit 24 → 25

These last two are the most exciting. This one involves bruteforcing all 4 digit pins to a remote server. The slow way is to loop doing `echo foo | nc`, but that creates a new connection each time. The faster way is to keep the connection open and loop. I used my `shoe.py` script for easy reading and writing to remote servers.

```
#!/usr/bin/python
import shoe

s = shoe.Shoe('localhost', 30002)
for i in range(0, 9999):
    cmd = "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ {}".format(i)
    s.write(cmd + "\n")
    r = s.read_until_end(.001)
    if "Wrong!" not in r and len(r) > 10:
            print(r)
```

Bandit 25 → 26

This one took me a bit figure out. I'll leave it as a challenge to the reader.

Congratulations on solving the last level of this game!

Share this post