OverTheWire Bandit Wargame: My Learning Journey

This repository documents my progress through the OverTheWire Bandit Wargame. This game is an excellent resource for learning basic Linux commands and security concepts. The goal of each level is to find a password to unlock the next one. If a password is lost, you'll need to restart from a previous level.

Let's dive into the levels!

Level 0 → Level 1

Level Goal: Log into the game using SSH.

Connection Details:

- **Host:** `bandit.labs.overthewire.org`
- **Port:** `2220`
- **Username:** `bandit0`
- **Password:** `bandit0`

Commands Used:

- `ssh`

Solution:

1. Open your terminal (e.g., Kali Linux).

Type the SSH command with the provided details:
Bash
ssh bandit0@bandit.labs.overthewire.org -p 2220

2.

3. When prompted, enter the password `bandit0`.

Password for Level 1: `ZjLjTmM6FvvyRnrb2rfNWOZOTa6ip5lf`

Level 1 → Level 2

Level Goal: The password for the next level is stored in a file called `readme` located in the home directory.

Commands Used:

- `ls`
- `cat`

Solution:

1. After logging in as `bandit0`, you are in the home directory.
2. Use `ls` or `ls -alps` to list the files in the current directory.
3. You will see a file named `readme`.

Use the `cat` command to display its contents:
Bash
cat readme

4. The `cat` command displays the contents of a file to the terminal.

Password for Level 2: `263JGJPfgU6LtdEvgfWU1XP5yac29mFx`

Important Tip: Create a file on your local machine to save notes and passwords. Passwords are not saved automatically and may change occasionally. Detailed notes are crucial for solving challenges and referencing later.

Level 2 → Level 3

Level Goal: The password for the next level is stored in a file called `-spaces in this filename-` located in the home directory.

Commands Used:

- `ls`
- `cat`

Solution:

1. Log into `bandit1` using the previous password.
2. Use `ls -alps` to list files. You'll observe a file with a leading dash and spaces.
3. Attempting
   `cat -spaces in this filename-` will fail because `cat` interprets `-spaces` as an option.

To correctly read a file with special characters or leading dashes, you can use `./` to specify the current directory or use quotes or escape characters.
Bash
cat ./-spaces\ in\ this\ filename-
Or, preferably:
cat -- -spaces\ in\ this\ filename-

4. The document shows
   `cat ./-` was used, which implies a file named just `-` or similar was present and contained the password. However, the level goal explicitly mentions a file with spaces. The provided solution snippet on page 5 shows
   `cat -spaces in this filename --` which is incorrect syntax and likely a typo in the document. The correct way to handle spaces is to enclose the filename in quotes or escape the spaces. For a file starting with `-`, `cat -- filename` or `cat ./filename` is required. The password obtained in the document for Level 2 is from
   `cat ./-`, implying a different file name than described in the level goal.

Password for Level 3: `MNk8KNH3Usiio41PRUEoDFPqfxLPlSmx`

Level 3 → Level 4

Level Goal: The password for the next level is stored in a hidden file in the `inhere` directory.

Commands Used:

- `ls`
- `cd`
- `cat`

Solution:

1. Log into `bandit2` using the previous password.
2. Use `ls -alps` to list files and directories. You'll see a directory named `inhere/`.

Change into the `inhere` directory:
Bash
cd inhere
``` [cite: 135]

3.
4. Once inside `inhere`, use `ls -al` to reveal hidden files (files starting with a dot `.`). You will see a hidden file named `... Hiding-From-You`.

Read the content of the hidden file. Remember to escape the leading dots and spaces:
Bash
cat \.\.\.\ Hiding-From-You
``` [cite: 163]

5.

Password for Level 4: `2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ`

Level 4 → Level 5

Level Goal: The password for the next level is stored in the only human-readable file in the `inhere` directory.

Commands Used:

- `ls`
- `cd`
- `file`
- `cat`

Solution:

1. Log into `bandit3` using the previous password.
2. Change into the `inhere` directory.

Use the `file` command with a wildcard to check the file type of all files in the current directory:
Bash
file ./*
``` [cite: 174]

3.
4. Examine the output. You are looking for a file that is "ASCII text" (human-readable). In this case,
   `file07` is identified as `ASCII text`.

Read the content of that file using `cat`:
Bash
cat ./-file07
```

5.

Password for Level 5: `40QYVPkXZO0E005pTW81FB8j8lxXGUQw`

Level 5 → Level 6

Level Goal: 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

Commands Used:

- `find`
- `cat`

Solution:

1. Log into `bandit4` using the previous password.
2. Use the `find` command to search for files with specific properties.
    - `.`: Search in the current directory and its subdirectories.
    - `-type f`: Look for files.
    - `-size 1033c`: Find files exactly 1033 bytes in size.
    - `!-executable`: Files that are NOT executable.
    - `-exec cat {} \;`: Execute `cat` on each found file.

Bash
find . -type f -size 1033c !-executable -exec cat {} \;

3. The document shows `find . -type f -size 1033c ! -executable` to locate the file, then `cat` to read its content. The `human-readable` property is implicitly handled by `cat` and `file` as seen in previous levels.

Password for Level 6: `HWasnphtq9AVKe0dmk45nxy20cvUa6EG`

Level 6 → Level 7

Level Goal: 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

Commands Used:

- `find`
- `cat`

Solution:

1. Log into `bandit5` using the previous password.

Use `find` to search the entire file system (starting from `/`) for a file with the specified properties.
Bash
find / -type f -user bandit7 -group bandit6 -size 33c
``` [cite: 265]

2.
3. The command will likely show "Permission denied" for many directories, which is normal. It will eventually find the file: `/var/lib/dpkg/info/bandit7.password`.

Use `cat` to read the content of this file:
Bash
cat /var/lib/dpkg/info/bandit7.password
```

4.

Password for Level 7: `morbNTDkSW6jIlUc0ym0dMaLn0lFVAaj`

Level 7 → Level 8

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

Commands Used:

- `ls`
- `cat`
- `strings`
- `grep`

Solution:

1. Log into `bandit6` using the previous password.
2. Use
   `ls` to confirm `data.txt` is present.

The file
`data.txt` contains many lines . To find the password efficiently, pipe the output of
`strings` (to extract printable characters) to `grep` (to search for "millionth").
Bash
strings data.txt | grep "millionth"
``` [cite: 310]

3.
4. The output will show the word "millionth" followed by the password.

**Password for Level 8:** `dfwvzFQi4mU0wfNbF0e9RoWskMLg7eEc`
```

Level 8 → Level 9

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

Commands Used:

- `cat`
- `sort`
- `uniq`

Solution:

1. Log into `bandit7` using the previous password.
2. The goal requires finding a unique line in `data.txt`. This can be achieved by combining `cat`, `sort`, and `uniq`.
    - `cat data.txt`: Displays the file's content.
    - `sort data.txt`: Sorts the lines alphabetically.
    - `uniq -c`: Counts the occurrences of each unique line.

Pipe these commands together:
Bash
cat data.txt | sort | uniq -c

3.
4. Look for the line that has a count of
   1 at the beginning.

Password for Level 9: `4CKMh1JI91bUIZZPXDqGanal4xvAg0JM`

Level 9 → Level 10

Level Goal: 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.

Commands Used:

- `strings`
- `grep`

Solution:

1. Log into `bandit8` using the previous password.
2. Use `strings` to extract printable strings from `data.txt`.

Pipe the output to `grep` and search for lines that start with multiple `=` characters. This can be done using a regular expression.
Bash
strings data.txt | grep "^=*"

3. The document shows a `grep` command with `y=7`, which is likely a placeholder or an incorrect attempt. The actual password `FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey` is shown after several lines containing `=` characters. A more precise `grep` could be `grep "====="` or `grep "^=.*"` (to match any line starting with `=`). Given the password is on its own line after a sequence of `=` characters, `grep -A 1 "^===="` might also work.

Password for Level 10: `FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey`

Level 10 → Level 11

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

Commands Used:

- `cat`
- `base64`

Solution:

1. Log into `bandit9` using the previous password.
2. Use
   `cat data.txt` to view the base64 encoded string.

Use the `base64 -d` command to decode the content.
Bash
cat data.txt | base64 -d
``` [cite: 376]
The `base64 -d` command decodes base64-encoded data.

3.

**Password for Level 11:** `dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr`
```

Level 11 → Level 12

Level Goal: 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 (ROT13).

Commands Used:

- `cat`
- `tr`

Solution:

1. Log into `bandit10` using the previous password.
2. Use
   `cat data.txt` to see the ROT13 encoded text.

Use the `tr` (translate) command to decode the ROT13.
Bash
cat data.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'

3. Alternatively, for ROT13 specifically, `tr 'a-zA-Z' 'n-za-mN-ZA-M'` works because ROT13 is symmetric (applying it twice returns the original text). The document snippet shows
   `Gur cnffjbeq vf` (The password is) , and then a password
   `7k16JArUVv5LxVuJfsSVdbbtaHGlw9D4`. The final decoded password is provided by the document as
   `The paxzserd is khwwfFEQongTy004`, which appears to be a separate example of ROT13, not the password for the level. The listed password
   `7x16WNeHli5YklhWsfFlqoognUTyj9Q4` is the expected result after ROT13 decoding.

Password for Level 12: `7x16WNeHli5YklhWsfFlqoognUTyj9Q4`

Level 12 → Level 13

Level Goal: 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.

Commands Used:

- `xxd`
- `file`
- `mkdir`
- `cp`
- `mv`
- `gzip`, `bzip2`, `tar` (for decompression)

Solution:

1. Log into `bandit11` using the previous password.

Create a temporary working directory (e.g., in `/tmp`) as recommended, to avoid cluttering your home directory.
Bash
mkdir /tmp/your_unique_dir
cd /tmp/your_unique_dir
``` [cite: 416, 418]
(Using `mktemp -d` is a better practice for unique temporary directories. [cite: 405])

2.

Copy `data.txt` to your temporary directory.
Bash
cp ~/data.txt .
``` [cite: 417]

3.

The file is a hexdump. Convert it back to binary:
Bash
xxd -r data.txt > data.bin

4. The document doesn't explicitly show the `xxd -r` command, but it's implied by the subsequent `file` commands on `data5.bin` and `data6.bin`. The hexdump in the document is directly followed by a `file` command on `data5.bin`, which suggests the hexdump was already written to a file or the process was slightly different.
5. Repeatedly use `file` to identify the compression type and then decompress it, renaming the file at each step for clarity.
    - `file data.bin` (initially) will show it as some compressed type.
    - Example sequence as per the document:
        - `file data5.bin` -> `POSIX tar archive (GNU)`
        - `mv data5.bin data.tar`
        - `tar xf data.tar` (extracts `data6.bin`)
        - `file data6.bin` -> `bzip2 compressed data`
        - `mv data6.bin data.bz2`
        - `bzip2 -d data.bz2` (extracts `data`)
        - `file data` -> `POSIX tar archive (GNU)`
        - `mv data data.tar`
        - `tar xf data.tar` (extracts `data8.bin`)
        - `file data8.bin` -> `gzip compressed data`
        - `mv data8.bin data.gz`
        - `gzip -d data.gz` (extracts `data`)
        - `file data` -> `ASCII text`

Finally, `cat` the resulting ASCII text file to get the password:
Bash
cat data
```

6.

**Password for Level 13:** `F05dwFsccbaIiH0h8J2eUks2vdTDwAn`
```

Level 13 → Level 14

Level Goal: 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 directly, but you get a private SSH key that can be used to log into the next level.

Commands Used:

- `ssh`
- `ls`
- `cat`

Solution:

1. Log into `bandit12` using the previous password.
2. List files in the current directory: `ls`. You'll find `sshkey.private`.

Change the permissions of the private key to be readable only by you:
Bash
chmod 600 sshkey.private
``` [cite: 558]

3.

Use the SSH key to log into `bandit14` on `localhost` (the same machine) on port `2220`.
Bash
ssh -i sshkey.private -p 2220 bandit14@localhost
``` [cite: 466]
You might be asked to confirm the authenticity of the host; type `yes`. [cite: 469]

4.

Once logged in as `bandit14`, read the password file:
Bash
cat /etc/bandit_pass/bandit14
``` [cite: 473]
```

5.

Password for Level 14: `MU4VWeTyJk8R0of1qqmcBPaLh7LDCPvS`

Level 14 → Level 15

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

Commands Used:

- `nc` (netcat)
- `cat`

Solution:

1. Log into `bandit13` using the previous SSH key.
2. Get the current password: `cat /etc/bandit_pass/bandit14` (which is `MU4VWeTyJk8R0of1qqmcBPaLh7LDCPvS`).

Use `nc` to connect to `localhost` on port `30000` and send the password.
Bash
echo MU4VWeTyJk8R0of1qqmcBPaLh7LDCPvS | nc localhost 30000

3. The document shows the password being typed directly into `nc localhost 30000` after the connection is established. Both methods work.

Password for Level 15: `8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo`

Level 15 → Level 16

Level Goal: The password for the next level can be retrieved by submitting the password of the current level to port `30001` on `localhost` using SSL/TLS encryption.

Commands Used:

- `ncat` (or `openssl s_client`)

Solution:

1. Log into `bandit14` using the previous password.
2. Get the current password: `cat /etc/bandit_pass/bandit15` (which is `8xCjnmgoKbGLhHFAZLGE5Tmu4M2tKJQo`).

Use `ncat` with the `-ssl` flag to establish an SSL/TLS encrypted connection to `localhost` on port `30001` and send the password.
Bash
echo 8xCjnmgoKbGLhHFAZLGE5Tmu4M2tKJQo | ncat -ssl localhost 30001

3. The document demonstrates typing the password after connecting with `ncat -ssl localhost 30001`.

Password for Level 16: `kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx`

Level 16 → Level 17

Level Goal: 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/TLS and which don't. Only 1 server will give the next credentials.

Commands Used:

- `nmap`
- `ncat` (with and without `-ssl`)
- `cat`

Solution:

1. Log into `bandit15` using the previous password.
2. Get the current password:
   `cat /etc/bandit_pass/bandit16` (which is `kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx`).

Use `nmap` to scan the specified port range (`31000-32000`) on `localhost` to find open ports.
Bash
nmap localhost -p 31000-32000
``` [cite: 515]

3.
4. Nmap will list several open ports . For each open port, try connecting with `ncat` without SSL and with SSL (`ncat -ssl`) to identify which one responds correctly.
   - Connect without SSL: `echo kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx | nc localhost <port_number>`
   - Connect with SSL: `echo kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx | ncat -ssl localhost <port_number>` The document shows `ncat -ssl localhost 31790` was the successful command.

**Password for Level 17:** This level provides an RSA private key which is the credential for the next level, not a plain text password.

`-----BEGIN RSA PRIVATE KEY-----...-----END RSA PRIVATE KEY-----`
```

Level 17 → Level 18

Level Goal: There are 2 files in the home directory: `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`.

Commands Used:

- `diff`

Solution:

Log into `bandit17` using the provided SSH private key. First, save the private key obtained from Level 16 into a file (e.g., `key`) and set its permissions:
Bash
vim key # paste the RSA private key here, then save and exit (:wq)
chmod 600 key
ssh -i key bandit17@bandit.labs.overthewire.org -p 2220
``` [cite: 541, 552, 558, 560]

1.

Once logged in, use the `diff` command to find the differences between the two files.
Bash
diff passwords.old passwords.new
``` [cite: 571]

2.
3. The output will show lines that are different. The line starting with `>` is the new line in `passwords.new`, which contains the password.

Password for Level 18: `x2gLTTjFwMOhQ8oWNbMN362QKxfRqG10`

Level 18 → Level 19

Level Goal: 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.

Commands Used:

- `ssh`
- `ls`
- `cat`

Solution:

1. When you try to log into `bandit18` normally, you are immediately logged out. This is due to a malicious entry in `.bashrc`.

To bypass `.bashrc` and get a shell, you can specify a command to execute directly with SSH. The `/bin/sh` shell can be used.
Bash
ssh -t bandit18@bandit.labs.overthewire.org -p 2220 /bin/sh
``` [cite: 582]
The `-t` option forces pseudo-terminal allocation, which is often needed when running interactive commands or shells over SSH.

2.
3. Once the shell is active, list the files (`ls`) to confirm `readme` is present.
4. Then `cat readme` to retrieve the password.

**Password for Level 19:** `CGWpMaKXVWDUNgPAVJbWYuGHVn9zl3j8`
```

Level 19 → Level 20

Level Goal: 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.

Commands Used:

- `ls`
- `./bandit20-do` (execution of a binary)
- `cat`

Solution:

1. Log into `bandit19` using the previous password.
2. List files (`ls`). You'll see an executable named `bandit20-do`.

Run the executable without arguments to understand its usage:
Bash
./bandit20-do
```
It will output: `Run a command as another user. Example: ./bandit20-do id` [cite: 601, 602]

3.

The goal states the password is in `/etc/bandit_pass/bandit20` but readable by `bandit20`. Since `bandit20-do` runs commands as `bandit20` (confirmed by `euid=11020(bandit20)` in `id` command output ), use it to `cat` the password file:
Bash
./bandit20-do cat /etc/bandit_pass/bandit20
```

4.

Password for Level 20: `0qXahG8ZjOVMN9Ghs7i0WsCfZyX0UbYO`

Level 20 → Level 21

Level Goal: There is a `setuid` binary in the home directory that makes a connection to `localhost` on a specified port. It reads a line of text from the connection and compares it to the password in the previous level (`bandit20`). If correct, it transmits the password for the next level (

`bandit21`).

Commands Used:

- `ls`
- `./suconnect` (execution of a binary)
- `cat`
- `nc` (netcat for listening)

Solution:

1. Log into `bandit19` using the previous password.
2. List files (`ls`). You'll find `suconnect`.
3. Run `suconnect` without arguments to see its usage: `./suconnect <portnumber>`.
4. The program expects a port number. To receive the password, you need to set up a listener on a port on `localhost` that `suconnect` will connect to.
5. Get the password for `bandit20`: `cat /etc/bandit_pass/bandit20` (which is `0qXahG8ZjOVMN9Ghs7i0WsCfZyX0UbYO`).

Open a new terminal (or use job control like `Ctrl+Z`, `bg` if your current terminal supports it for multitasking) on your local machine and set up a netcat listener on an arbitrary unused port (e.g., `4444`).
Bash
nc -lvp 4444

``` [cite: 638]

6.

Go back to your SSH session as `bandit20` and execute `suconnect`, pointing it to your listener port:
Bash
./suconnect 4444
``` [cite: 646]

7.
8. Immediately after `suconnect` connects, type the `bandit20` password (`0qXahG8ZjOVMN9Ghs7i0WsCfZyX0UbYO`) into the listening `nc` session and press Enter. The
   `suconnect` program will verify it and then send the `bandit21` password back to your listener.

Password for Level 21: `EeoULMCra2q0dSkYj561DX7s1CpBu0Bt`

Level 21 → Level 22

Level Goal: 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.

Commands Used:

- `ls`
- `cat`

Solution:

1. Log into `bandit20` using the previous password.

List the contents of the `/etc/cron.d/` directory to see the cron job configurations.
Bash
ls /etc/cron.d/

``` [cite: 656]

2.

You'll see a file like `cronjob_bandit22`. View its contents:
Bash
cat /etc/cron.d/cronjob_bandit22
``` [cite: 660]

3.
4. The script indicates that `bandit22` runs `/usr/bin/cronjob_bandit22.sh`.

Examine the script:
Bash
cat /usr/bin/cronjob_bandit22.sh
``` [cite: 665]
The script shows that it changes permissions of a file in `/tmp/` and then `cat`s the `bandit22` password into that file:
```bash
chmod 644 /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
``` [cite: 667, 668]

5.

The password will be in the specified temporary file. Read it:
Bash
cat /tmp/t706lds9S0RqQh9aMcz6ShpAoZKF7fgv
```

6.

Password for Level 22: `tRae0UfB9v0UzbCdn9cY0gQnds9GF58Q`

Level 22 → Level 23

Level Goal: A program is running automatically at regular intervals from `cron`. Look in `/etc/cron.d/` for the configuration and see what command is being executed. This script is intentionally made easy to read.

Commands Used:

- `cat`
- `whoami`
- `md5sum`
- `cut`

Solution:

1. Log into `bandit21` using the previous password.

Examine the cron job for `bandit23`:
Bash
cat /etc/cron.d/cronjob_bandit23
``` [cite: 679]
It shows `bandit23` running `/usr/bin/cronjob_bandit23.sh`. [cite: 680]

2.

View the script's content:
Bash
cat /usr/bin/cronjob_bandit23.sh
``` [cite: 681]
The script contains:
```bash
#!/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
``` [cite: 682, 683, 684, 685]

3.

4. The script copies the password for `bandit23` to a file in `/tmp/` whose name is generated by an MD5 hash of "I am user bandit23".

To find the `mytarget` filename, simulate the script's logic:
Bash
echo I am user bandit23 | md5sum | cut -d-f 1
``` [cite: 693]
This will output the hash, which is `8ca319486bfbbc3663ea0fbe81326349`.

5.

Now, `cat` the content of the file in `/tmp/` with this hash:
Bash
cat /tmp/8ca319486bfbbc3663ea0fbe81326349
``` [cite: 694]

6.

Password for Level 23: `0Zf11ioIjMVN551jX3CmStKLYqjk54Ga`

Current Passwords Log:

- **level-0:** `bandit0`
- **level 0-1:** `ZjLjTmM6FvvyRnrb2rfNWOZOTa6ip5lf`
- **level 1 - 2:** `263JGJPfgU6LtdEvgfWU1XP5yac29mFx`
- **level 2 - 3:** `MNk8KNH3Usiio41PRUEoDFPqfxLPISmx`
- **level 3 - 4:** `2WmrDFRmJlq3lPxneAaMGhap0pFhF3NJ`
- **level 4-5:** `40QYVPkxZOOEOO5pTW81FB8j8lxXGUQw`
- **level 5-6:** `HWasnPhtq9AVKe0dmk45nxy20cvUa6EG`
- **level 6-7:** `morbNTDkSW6jllUc0ymOdMaLnOIFVAaj`
- **level 7 - 8:** `dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc`
- **level 8-9:** `4CKMh1JI91bUIZZPXDqGanal4xvAg0JM`
- **level 9-10:** `FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey`
- **level10-11:** `dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr`
- **level11-12:** `7x16WNeHli5YklhWsfFlqoognUTyj9Q4`
- **level12-13:** `F05dwFsc0cbaliH0h8J2eUks2vdTDwAn`
- **level13-14:** `MU4VWeTyJk8ROof1qqmcBPaLh7IDCPvS`
- **level14-15:** `8xCjnmgoKbGLhHFAZIGE5Tmu4M2tKJQo`
- **level15-16:** `kSkvUpMQ7IBYyCM4GBPvCvT1BfWRy0Dx`

level16-17: (RSA Private Key - see below)

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMIOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORIO+rW4LCDCNd2IUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr5604T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zblkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RILwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMlu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enClvGCSx+X315SiWg0A
R57hJglezliVjv3aGwHwvlZvtszK6zV60XFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUI+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu

L8ktHMPvodBwNsSBULpG0QKBgBApITfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFulSeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmlJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBI1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6lgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----

- **level17-18:** `x2gLTTjFwMOhQ80WNbMN362QKxfRqGIO`
- **level18-19:** `cGWpMaKXVwDUNGPAVJbWYuGHVn9zl3j8`
- **level19-20:** `0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO`
- **level20-21:** `EeoULMCra2q0dSkYj561DX7s1CpBuOBt`
- **level21 - 22:** `tRae0UfB9v0UzbCdn9cY0gQnds9GF58Q`
- **level22 - 23:** `0Zf11ioljMVN551JX3CmStKLYqjk54Ga`

I will update this post as I continue my journey through the Bandit Wargame!

--->shanyu-pilli