OverTheWire Bandit Game Notes

Florian Legendre

August 17, 2020

Contents

Part I

Synthesis

Chapter 1

Hacking Methodology (aka Problem Solving Methodology)

En toutes occasions: Quoi (Nature de ce qu'on va décrire, peut être remis en cause après analyse)? Comment (Description et uniquement une description de ce qu'on a/voit)? Pourquoi comme ça? Ces trois questions seules devraient suffire à nourrir les questions suivantes:

1. Quelles sont les bonnes questions à se poser?
2. Quelles hypothèses peut-on avancer pour répondre à ces questions?
3. Comment peut-on raisonnablement exclure certaines hypothèses pour sélectionner la solution?
4. Le cas échéant, quels sont nos présupposés et quelles nouvelles hypothèses peut-on faire en remettant en cause ces présupposés?

Chapter 2

Useful Commands and their Syntaxis

Remember "-" is a special character for most bash commands, so e.g file ./-file00 to avoid that...

2.1 Files Manipulation

- find -¿ searches for files from the directory where it's called (by default) to all subdirectories (e.g find file1)

2.2 Text Manipulation

- diff –color [File1] [File2]
- uniq -¿ report or omit repeated lines. With no options, matching lines are merged to the first occurrence. -u, –unique only print unique lines. -w, –check-chars=N compare no more than N characters in lines.
- sort -¿ sort lines of text files
- base64 -¿ base64 encode/decode data and print to standard output. Syntax: base64 [option] [FILE] With no file, or if '-', reads standard input. -d option = decode.
- grep [pattern] [file] grep searches the named input FILEs for lines containing a match to the given PATTERN. If no files are specified, or if the file "-" is given, grep searches standard input. By default, grep prints the matching lines. E.G grep millionth data.txt
- tr -¿ translate or delete characters Syntax: tr [OPTION]... SET1 [SET2]

  option -d deletes characters.

  You can use special characters as ¸and much more in [SET2] (same meaning as in C++)

CHAR1-CHAR2 ¡=¿ all characters from CHAR1 to CHAR2 in ascending order.

-s option replaces each sequence of a repeated character that is listed in the last specified SET, with a single occurrence of that character.

Important: tr translates each character in set1 with each character in set2. Hence you can write: A-Za-z N-ZA-Mn-za-m, there are 26 characters from A-Z which are first mapped with the 13 characters from N-Z then the other 13 characters from A-M. Then it proceeds with the characters from a-z...

- xxd -r [file] reverses hexdump formatting and prints output to terminal (not to [file]!!) (xxd creates hexdumps with various format options.)

  To use content you have to create a new file and append it with xxd -r [file] output.

2.3 Output Redirections

- ¿ means that both stdout and stderr are redirected
- 2¿1, 2¿ means redirection of stderr (the "2") to whatever file is after. means that it's not what's after is not a filename it's a file descriptor (0 for stdin, 1 for stdout, 2 for stderr

2.4 Others

- "tmux is a terminal multiplexer: it enables a number of terminals to be created, accessed, and controlled from a single screen. tmux may be detached from a screen and continue running in the background, then later reattached." (from man tmux) Useful shortcuts (Crl+b before any shortcut) : " -¿ horizontal split, vertical split, arrows to move from panels, x to kill a panel Command: tmux attach-session ¡session$_name >$

Part II

Levels, Solutions and Lessons

Chapter 3

Level 0 → Level 10

3.1 Level 0

```
Solution = ssh bandit0@176.9.9.172 -p 2220

Lesson = syntax is ssh [username]@[host_ip] (option -p = specify port) (port)
         If no ip given but a web adress, it's possible to extract ip thanks
         to existing web applications, e.g bandit.labs.overthewire.org
```

3.2 Level 0 → Level 1

```
Solution = boJ9jbbUNNfktd78OOpsqOltutMc3MY1

Lesson = ls, cd, cat, file, du, find... New commands learnt.

 cat -> reads files
 its name means 'concatenate' as you can displays several files at once
 (they are concatenated.) Cat can also create new files with the redirection
 operator (e.g cat file1 > file2) WARNING! If file2 already exists it will be
 overwritten, to avoid that use the append operator (e.g cat file1 >> file2)

 du -> displays disk usage

 find -> searches for files
 from the directory where it's called (by default) to all subdirectories
 (e.g find file1)
```

3.3 Level 1 → Level 2

```
Solution = CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

Lesson = If a file is named "-" it can provoque strange behaviours from UNIX
 commands as "-" is a command special character (with cat it prints
 input to stdout). To avoid that we have to specify that it's a file
 from a directory (syntax is then, e.g cat ./-)
```

3.4 Level 2 → Level 3

```
Solution = UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK

Lesson = To open files with spaces in filenames you have to type in
 escape characters "\" with the escaped character next to it
 (here a blank space) e.g cat space\ in\ this\ filename
```

3.5 Level 3 → Level 4

```
Solution = pIwrPrtPN36QITSp3EQaw936yaFoFgAB

Lesson = cd stands for change directory and option -a in ls -a prints out hidden files
```

3.6 Level 4 → Level 5

```
Solution = koReBOKuIDDepwhWk7jZC0RTdopnAYKh

Lesson = Remember "-" is a special character for most bash commands,
 so e.g file ./-file00 to avoid that
```

3.7 Level 5 → Level 6

```
Solution = DXjZPULLxYr17uwoI01bNLQbtFemEgo7

Lesson = Command 'find' has options (read 'man' for all the options).
 -executable = Am I looking for an executable?
 -readable = Is the file readable by a human?
 -size = Specifies the size of the file you are looking for
 (Units: c -> bytes, k -> Kb, 'M' -> Mb, 'G' -> Gb)
   E.G find -readable -size 1033c
```

3.8 Level 6 → Level 7

```
Solution = HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

Lesson = find [directory from where to start search] [options],
 remember: '/' stands for root (= looking everywhere on the system).
 Read all of the output!! The answer was in it, if you had looked
 better you'd have found it. Don't cheat! You'll regret it.
 Other bunch of useful find options:
  -name 'name_pattern'
  -group groupe_pattern
  -user user_pattern
  -writable
```

3.9 Level 7 → Level 8

```
Solution = cvX2JJa4CFALtqS87jk27qwqGhBM9plV

Lesson = grep [pattern] [file] grep searches the named input FILEs
 for lines containing a match to the given PATTERN.
 If no files are specified, or if the file "-" is given,
 grep  searches  standard input. By default, grep prints the matching lines.
 E.G grep millionth data.txt
```

3.10 Level 8 → Level 9

```
Solution = UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR

Lesson = uniq -> report or omit repeated lines.
 With no options, matching lines are merged to the first occurrence.
 -u, --unique only print unique lines.
 -w, --check-chars=N compare no more than N characters in lines.

 sort -> sort lines of text files (-u option: misunderstanding...)
 | means "pipe down", it allows to add another transformation (operation)
 to the argument.
```

3.11 Level 9 → Level 10

```
Solution = truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk

Lesson = grep -a =* data.txt, -a option reads binary files as text.
```

Chapter 4

Level 10 → Level 20

4.1 Level 10 → Level 11

```
Solution = IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

Lesson = base64 -> base64 encode/decode data and print to standard output.
 Syntax: base64 [option] [FILE]
   With no file, or if '-', reads standard input.
   -d option = decode.

   Le protocole de base de transmission de courriels, SMTP,
   ne supporte que les caractères ASCII (qui font 7 bits).
   Cela limite les courriels aux messages qui n'incluent que
   ces caractères.

   MIME définit des mécanismes pour l'envoi d'autres sortes
   d'informations, comme des textes utilisant des codages de caractères
   autres que l'ASCII (et pouvant donc être dans une autre langue que
   l'anglais), ou des données binaires (dont des fichiers contenant des
   images, des sons, des films ou des programmes informatiques).
```

4.2 Level 11 → Level 12

```
Solution = 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

Lesson = cat data.txt | tr A-Za-z N-ZA-Mn-za-m

 tr -> translate or delete characters
 Syntax: tr [OPTION]... SET1 [SET2]

 option -d deletes characters.

 You can use special characters as \t, \n
 and much more in [SET2] (same meaning as in C++)

 CHAR1-CHAR2 <=> all characters from CHAR1 to CHAR2 in
 ascending order.

 -s option replaces each sequence of a repeated character
 that is listed  in the last specified SET,
 with a single occurrence of that character.

 Important: tr translates each character in set1 with each
 character in set2. Hence you can write: A-Za-z N-ZA-Mn-za-m,
 there are 26 characters from A-Z which are first mapped with
 the 13 characters from N-Z then the other 13 characters from A-M.
 Then it proceeds with the characters from a-z...
```

4.3 Level 12 → Level 13

```
Solution = 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

Lesson = xxd -r [file] reverses hexdump formatting and prints output
 to terminal (not to [file]!!) (xxd creates hexdumps with
 various format options.)

   To use content you have to create a new file and append it
   with xxd -r [file] output.

   gzip and bzip are two different compressing algorithms (.gz and .bz2)
   If file indicates that it is gzip (or bzip2) compressed data
   but gzip -d doens't work (wrong suffix error message) you have to
   add .gz (or .bz2) at the end of the file you want to decompress
   (Ex. mv data data.gz)

   tar is GNU archive utility, option -x means extract but you have to
   add -f option to indicate a file name.
```

4.4 Level 13 → Level 14

```
Solution = 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

Lesson = The client creates a key pair and then uploads the public key
 to any remote server it wishes to access. This is placed in a file
 called authorized_keys within the ~/.ssh directory in the user account's
 home directory on the remote server.

 After the symmetrical encryption is established to secure communications
 between the server and client, the client must authenticate to be allowed
 access. The server can use the public key in this file to encrypt a
 challenge message to the client. If the client can prove that it was able
 to decrypt this message, it has demonstrated that it owns the associated
 private key. The server then can set up the environment for the client.

 => Bandit14 created a pair of keys, a private and a public one, and put the
 public one on the server. When we get into the server under bandit13
 authentication (thanks to the password above) we steal the private key.

 ssh connects and logs into the specified hostname (with optional user
 name).  The user must prove his/her identity to the remote machine using
 one of several methods

 => Whether we decide to use ssh -i ~/sshkey.private bandit14@localhost
 (provided that we're still on the server under bandit13) or copy the
 private key in a file on our computer when we try to log in using -i, the
 server uses the public key associated to bandit14 to sends us a challenge.

 ssh -i identity_file [user@]hostname
 -i : Selects a file from which the identity (private key) for public key
 authentication is read.

 => As our private key is the right one, we're in!

 /!\ Don't forget -p 2220, otherwise you'll be trying to log in something
 else (some place on the server machine we aren't supposed to be.)

 /!\ Don't forget chmod 600 [filename]. The file we use to store the private
 key must have read-write authorization by the owner only. That's what the
 600 flag gives us (2 + 4 0 0, 2 corresponds to write, 4 to read, 1 is
 execute...* the place of the digit indicates who has what authorization.
 First digit is the owner, others are groups...)

 *0 { no permission
  1 { execute
  2 { write
  3 { write and execute
  4 { read
  5 { read and execute
```

```
 6 { read and write
 7 { read, write, and execute
```

4.5 Level 14 → Level 15

```
Solution = BfMYroe26WYalil77FoDi9qh59eK5xNr

Lesson = The objective is 'The password for the next level can be retrieved by
 submitting the password of the current level to port 30000 on localhost.'
 So we have to submit a password, a string of characters, to a listening
 server on port 30000.

 First way of doing this...

 => bandit14@bandit:~$ telnet localhost 30000
 Trying 127.0.0.1...
 Connected to localhost.
 Escape character is '^]'.
 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e
 Correct!
 BfMYroe26WYalil77FoDi9qh59eK5xNr

 Connection closed by foreign host.

 OR second way of doing this...
 => bandit14@bandit:~$ echo "4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e" | nc
 localhost 30000
 Correct!
 BfMYroe26WYalil77FoDi9qh59eK5xNr


 "You may be asking "why not just use telnet to connect to arbitrary ports?"
 Valid question, and here are some reasons.  Telnet has the "standard
 input  EOF"  problem,  so one must introduce calculated delays in driving
 scripts to allow network output to finish.  This is the main
 reason netcat stays running until the *network* side closes.  Telnet
 also will not transfer arbitrary binary data, because  certain  characters
 are  interpreted  as  telnet  options and are thus removed from the data
 stream.  Telnet also emits some of its diagnostic messages to standard
 output, where netcat keeps such things religiously separated from its
 *output* and will never modify any of the real data in  transit  unless
 you *really* want it to.  And of course telnet is incapable of listening
 for inbound connections, or using UDP instead.  Netcat doesn't have any
 of these limitations, is much smaller and faster than telnet, and has many
 other advantages." (from man nc)

 "Telnet is a client-server protocol based on character-oriented data
 exchange over TCP connections. Telnet enables remote control of computers
 via text-based inputs and outputs." [...] Another deployment scenario, for
```

which Telnet clients were typical, was the access to text-based programs on an application server. [...] The prerequisite for using Telnet is that the control device has user recognition. [...] Since Telnet connections are practically standard TCP connections, the client can be employed to use or test other services that rely on TCP as a transport protocol." (https://www.ionos.com/digitalguide/server/tools/telnet-the-system-wide-remote-protocol/)

"Telnet [is] often used to remotely log in to a remote server" (https://www.techwalla.com/articles/how-does-telnet-work) => the server can be something else than a telnet server...

"La commande telnet reste une commande très pratique pour tester des serveurs. Vu la flexibilité du programme, il est possible d'utiliser la commande telnet pour établir une connexion TCP interactive avec d'autres services tels que SMTP, HTTP, POP, IMAP, etc. en utilisant alors le port du protocole au lieu du port telnet standard. " (https://fr.wikipedia.org/wiki/Telnet)

=> The ndmps server accepts TCP connexions and text inputs so telnet and netcat (nc) both work

4.6 Level 15 → Level 16

```
Solution = cluFn7wTiGryunymYOu4RcffSxQluehd
```

```
Lesson = bandit15@bandit:~$ openssl s_client -connect localhost:30001
```

```
CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = localhost
verify return:1
---
Certificate chain
  0 s:/CN=localhost
    i:/CN=localhost
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICBjCCAW+gAwIBAgIEDU18oTANBgkqhkiG9w0BAQUFADAUMRIwEAYDVQQDDAls
b2NhbGhvc3QwHhcNMjAwNTA3MTgxNTQzWhcNMjEwNTA3MTgxNTQzWjAUMRIwEAYD
VQQDDAlsb2NhbGhvc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAK3CPNFR
FEypcqUa8NslmIMWl9xq53Cwhs/fvYHAvauyfE3uDVyyX79Z34Tkot6YflAoufnS
+puh2Kgq7aDaF+xhE+FPcz1JE0C2bflGfEtx4l3qy79SRpLiZ7eio8NPasvduG5e
pkuHefwI4c7GS6Y7OTz/6IpxqXBzv3c+x93TAgMBAAGjZTBjMBQGA1UdEQQNMAuC
CWxvY2FsaG9zdDBLBglghkgBhvhCAQ0EPhY8QXV0b21hdGljYWxseSBnZW5lcmF0
ZWQgYnkgTmNhdC4gU2VlIGh0dHBzOi8vbm1hcC5vcmcvbmNhdC8uMA0GCSqGSIb3
```

```
DQEBBQUAA4GBAC9uy1rF2U/OSBXbQJYuPuzT5mYwcjEEV0XwyiX1MFZbKUlyFZUw
rq+P1HfFp+BSODtk6tHM9bTz+p2OJRXuELG0ly8+Nf/hO/mYS1i5Ekzv4PL9hO8q
PfmDXTHs23Tc7ctLqPRj4/4qxw6RF4SM+uxkAuHgT/NDW1LphxkJlKGn
-----END CERTIFICATE-----
subject=/CN=localhost
issuer=/CN=localhost
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1019 bytes and written 269 bytes
Verification error: self signed certificate
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
     Protocol  : TLSv1.2
     Cipher    : ECDHE-RSA-AES256-GCM-SHA384
     Session-ID:
     76CC6CB7E839CC08417EF00100FCDDAF56B8E2DE9121D75CA7F36315497197EE
     Session-ID-ctx:
     Master-Key
     :96F9DA3420A7C3901995E364404C452C4DA199EFDA68F9965D9F29BDB
     8F3065238F7E0633F2097EBB3BFA8648768D695
     PSK identity: None
     PSK identity hint: None
     SRP username: None
     TLS session ticket lifetime hint: 7200 (seconds)
     TLS session ticket:
   0000 - aa 02 e6 3a 2e 0b c8 5d-6f 54 4a 1b 5a e0 2c 0e   ...:...]oTJ.Z.,.
   0010 - 48 e5 e2 b7 57 35 0c 8e-5a 6d 91 e9 f8 94 61 3b   H...W5..Zm....a;
   0020 - ce d4 30 39 a7 ab 67 b4-99 ab 6a 12 e7 18 9c dc   ..09..g...j.....
   0030 - 5f 3e 58 eb b6 df c0 24-bf b3 f5 73 5c 51 62 2d   _>X....$...s\Qb-
   0040 - 3f af aa 26 ed bf 26 43-5d c7 fd 1f d2 37 d0 40   ?..&..&C]....7.@
   0050 - 88 9c 10 c4 2d ae 5d 9c-36 a6 dd 63 76 56 27 c7   ....-.].6..cvV'.
   0060 - d1 97 f6 7d 39 1b e5 a3-ee cb 92 a1 34 7c 78 48   ...}9.......4|xH
   0070 - 2d cd 61 e6 36 b6 9b 00-75 31 39 df 44 25 3f cb   -.a.6...u19.D%?.
   0080 - 90 1c 73 c6 93 41 02 23-b5 35 95 65 c7 8e 99 42   ..s..A.#.5.e...B
   0090 - 78 86 4e be e3 ca 14 57-f3 8e 42 aa 47 2a 8e 71   x.N....W..B.G*.q

     Start Time: 1596284463
     Timeout   : 7200 (sec)
     Verify return code: 18 (self signed certificate)
     Extended master secret: yes
---
```

```
/* Program stopped here and waited with a blank. It took a moment before I
realized that it was waiting for an input (no cursor was blinking.
The CONNECTED(00000003) above helped me understand that. So I copy-pasted
the password as asked
by the challenge on the otw website and got the output below */

BfMYroe26WYalil77FoDi9qh59eK5xNr
Correct!
cluFn7wTiGryunymYOu4RcffSxQluehd

closed
```

4.7 Level 16 → Level 17

```
Solution =

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----


Lesson =

bandit16@bandit:~$ nmap -A -p 1-35000 localhost
```

```
Starting Nmap 7.40 ( https://nmap.org ) at 2020-08-01 14:43 CEST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00021s latency).
Not shown: 34990 closed ports
PORT      STATE SERVICE             VERSION
22/tcp    open  ssh                 OpenSSH 7.4p1 (protocol 2.0)
| ssh-hostkey:
|   2048 0f:b3:a9:49:cc:50:3e:72:98:aa:10:fb:33:12:72:c4 (RSA)
|_  256 0a:89:2a:f1:2e:2f:df:66:3e:fe:bb:49:10:1a:96:32 (ECDSA)
113/tcp   open  ident
30000/tcp open  ndmps?
| fingerprint-strings:
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, Help, Kerberos,
LDAPSearchReq, LPDString, RTSPRequest, SIPOptions, SSLSessionReq, TLSSessionReq:
|_    Wrong! Please enter the correct current password
30001/tcp open  ssl/pago-services1?
| fingerprint-strings:
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, Help, Kerberos,
LDAPSearchReq, LPDString, RTSPRequest, SIPOptions, SSLSessionReq, TLSSessionReq:
|_    Wrong! Please enter the correct current password
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost
| Not valid before: 2020-05-07T18:15:43
|_Not valid after:  2021-05-07T18:15:43
|_ssl-date: TLS randomness does not represent time
30002/tcp open  pago-services2?
| fingerprint-strings:
|   GenericLines:
|     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.
|     Fail! You did not supply enough data. Try again.
|     Fail! You did not supply enough data. Try again.
|   GetRequest, HTTPOptions, RTSPRequest:
|     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.
|     Wrong! Please enter the correct current password. Try again.
|     Fail! You did not supply enough data. Try again.
|   NULL, RPCCheck:
|_    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.
31046/tcp open  echo
31518/tcp open  ssl/echo
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost
| Not valid before: 2020-07-11T13:58:02
|_Not valid after:  2021-07-11T13:58:02
|_ssl-date: TLS randomness does not represent time
31691/tcp open  echo
31790/tcp open  ssl/unknown
```

```
| fingerprint-strings:
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, Help, Kerberos,
LDAPSearchReq, LPDString, RTSPRequest, SIPOptions, SSLSessionReq, TLSSessionReq:
|_    Wrong! Please enter the correct current password
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost
| Not valid before: 2020-07-11T13:56:28
|_Not valid after:  2021-07-11T13:56:28
|_ssl-date: TLS randomness does not represent time
31960/tcp open  echo
4 services unrecognized despite returning data. If you know the service/version,
please submit the following fingerprints at
https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port30000-TCP:V=7.40%I=7%D=8/1%Time=5F25638F%P=x86_64-pc-linux-gnu%r(Ge
[...]
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port31790-TCP:V=7.40%T=SSL%I=7%D=8/1%Time=5F25639A%P=x86_64-pc-linux-gn
SF:u%r(GenericLines,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20cur
SF:rent\x20password\n")%r(GetRequest,31,"Wrong!\x20Please\x20enter\x20the\
SF:x20correct\x20current\x20password\n")%r(HTTPOptions,31,"Wrong!\x20Pleas
SF:e\x20enter\x20the\x20correct\x20current\x20password\n")%r(RTSPRequest,3
SF:1,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\n
SF:")%r(Help,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x2
SF:0password\n")%r(SSLSessionReq,31,"Wrong!\x20Please\x20enter\x20the\x20c
SF:orrect\x20current\x20password\n")%r(TLSSessionReq,31,"Wrong!\x20Please\
SF:x20enter\x20the\x20correct\x20current\x20password\n")%r(Kerberos,31,"Wr
SF:ong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\n")%r(
SF:FourOhFourRequest,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20cu
SF:rrent\x20password\n")%r(LPDString,31,"Wrong!\x20Please\x20enter\x20the\
SF:x20correct\x20current\x20password\n")%r(LDAPSearchReq,31,"Wrong!\x20Ple
SF:ase\x20enter\x20the\x20correct\x20current\x20password\n")%r(SIPOptions,
SF:31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\
SF:n");

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 145.82 seconds



bandit16@bandit:~$ openssl s_client -connect localhost:31790
CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = localhost
verify return:1
---
Certificate chain
```

```
 0 s:/CN=localhost
   i:/CN=localhost
---
Server certificate
-----BEGIN CERTIFICATE-----
MIICBjCCAW+gAwIBAgIEOxBGEjANBgkqhkiG9w0BAQUFADAUMRIwEAYDVQQDDAls
b2NhbGhvc3QwHhcNMjAwNzExMTM1NjI4WhcNMjEwNzExMTM1NjI4WjAUMRIwEAYD
VQQDDAlsb2NhbGhvc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALjlJy1H
hxygfKR5X5QT8dbHVAqKBGZPWUutQJE5E7Ic+xKGl1BAVFzJmbGnJ8cxHgpSubDW
urtfkIPgu/vyyIhYn4jhmgkJOWuHc7mxRl64TVYfxMh6YpalOQ1aQeNsOtYgUoqA
+aG3Sa4eCaBNawS+CgV6EEnx0LICSN7cTRATAgMBAAGjZTBjMBQGA1UdEQQNMAuC
CWxvY2FsaG9zdDBLBglghkgBhvhCAQ0EPhY8QXV0b21hdGljYWxseSBnZW5lcmF0
ZWQgYnkgTmNhdC4gU2VlIGh0dHBzOi8vbm1hcC5vcmcvbmNhdC8uMA0GCSqGSIb3
DQEBBQUAA4GBAKHnag1vqMuJu3G3CTM/6pJWW14JvOoDwtTas8EgG6rLrBxNV8uU
HutrzqeW9EANLBnQyDytynWzU9fNh1TWtEVku1X/TLizuQb5EGF6pRE1n6LF9ptJ
CQkvW1CH8eOILuQcbPyjg+/43FM3ByVXtQmTEhORm7olAo8upbFLdTd0
-----END CERTIFICATE-----
subject=/CN=localhost
issuer=/CN=localhost
---
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: X25519, 253 bits
---
SSL handshake has read 1019 bytes and written 269 bytes
Verification error: self signed certificate
---
New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 1024 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: 3492073100BCC8052E37AA90DC0E502A70FF7210BDA7A1C3CC128EE1D3643759
    Session-ID-ctx:
    Master-Key: 3A65DCAC842CF4D3186F566DFF23C190DFC920650823CB7E7AFD7BDD994
    8AC68CBF2DE978BF9748237B8AA03930F55A0
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 7200 (seconds)
    TLS session ticket:
    0000 - 71 24 75 25 65 da ee 1d-04 f1 98 ed 3f 34 fb af   q$u%e.......?4..
    0010 - 38 96 16 8f a8 b7 65 ec-d4 e3 9b c9 a9 6b f4 f1   8.....e......k..
    0020 - 73 05 30 ab 59 a9 da 21-51 a2 ce b0 f8 0e ba 15   s.0.Y..!Q.......
    0030 - 97 98 67 22 ff 37 df 41-ee 2e 52 2d 96 ff 70 7d   ..g".7.A..R-..p}
    0040 - 31 64 56 5f 2b 06 f6 ed-97 77 12 40 51 21 01 f2   1dV_+....w.@Q!..
```

```
    0050 - 37 9d 40 5f f6 67 65 29-c7 66 2a bf e0 f8 1c fc   7.@_.ge).f*.....
    0060 - 29 68 e9 76 6b 78 91 60-90 ff c0 39 d9 4a 11 9d   )h.vkx.‘...9.J..
    0070 - 1d 6d 8b bd 8f 4d 8a a6-03 ed 71 1f 71 33 39 9d   .m...M....q.q39.
    0080 - d9 5e 53 2e 82 14 7d 62-25 6e 61 71 d3 89 ea 04   .^S...}b%naq....
    0090 - 8e b0 3a 88 67 b9 f4 65-fd 87 a3 b8 9e 4d 80 e0   ..:.g..e.....M..

    Start Time: 1596286101
    Timeout   : 7200 (sec)
    Verify return code: 18 (self signed certificate)
    Extended master secret: yes
---



cluFn7wTiGryunymYOu4RcffSxQluehd



Correct!
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKNOK5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjgOLWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----

closed
```

```
 => nmap by default ranges from port 1 to port 30000. This can be changed
 with -p option. -A allows to provide more information on what's listening
 on active ports (protocols, versions, etc.)

 => Remember: sudo chmod 600 on the new private key file otherwise it won't
 be identified as such. To verify : file <private_key_file_path>
```

4.8 Level 17 → Level 18

```
Solution = kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

Lesson =
 (base) florian@CRexJr:~$ ssh bandit17@176.9.9.172 -p 2220 -i ~/MEGA/Bureau
 BU/CS/Linux/bandit_game/ssh_rsa_lvl16

 This is a OverTheWire game server. [...] Enjoy your stay!

 bandit17@bandit:~$ diff --color passwords.old passwords.new
 42c42
 < w0Yfolrc5bwjS4qw5mq1nnQi6mF03bii
 ---
 > kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

 => Up = first argument of diff command, down = second argument. Color
 option for legibility.
 => Don't forget -p 2220 to log in any bandit server...
```

4.9 Level 18 → Level 19

```
Solution = IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

Lesson =

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

 Commands you may need to solve this level ssh, ls, cat

 => Someone added this at the end of .bashrc (and only this):
 echo 'Byebye !'
 exit 0

 ".bashrc is not meant to be executed but sourced."  (https:/
 stackoverflow.com/questions/19742005/bashrc-permission-denied)
```

"When you call source (or its alias .), you load and execute a bash script into the current bash process. So you can read variables set in the sourced script, use functions defined within it and even execute forks and/or subprocess if script do this.

When you call sh, you initiate a fork (sub-process or child) that runs a new session of /bin/sh, which is usually a symbolic link to [a] bash [shell]. In this case, environment variables set by the sub-script would be dropped when the sub-script finishes.

Caution: sh could be a symlink to another shell.
[...]
[for an executed script] you need to need to have execution permission to execute it (since it is forking)" (https://stackoverflow.com/questions/13786499/what-is-the-difference-between-using-sh-and-source)

=> .bashrc is sourced each time you log in a terminal which means that you have to interrupt the echo and exit processes it launches with Ctrl + C. You have to be very fast for this to work.

Another solution:
"In other words the pseudo-terminal(pty) is closed. Note that ssh uses a pseudo-terminal(pty) and not a text-terminal(tty).

So I needed to force a pseudo-terminal to open. This can be done with the -t flag while ssh-ing.

ssh -t bandit18@bandit.labs.overthewire.org /bin/sh

Here /bin/sh is the shell I used in my pty.

So now I have a shell I could use to interact with the server.
A simple cat readme gives us the password for the next level.

The pseudo-shell
Note that we didn't get all the welcome information we used to get in normal sshes since the normal commands didn't get executed this time, rather it forced open the pty I ordered.

There is another way to do it. The -T flag. -T flag disables pseudo-terminal allocation. This prevent the sourcing of .bashrc file.
We get just the shell without the terminal and then I used cat readme to get the password.

ssh -T bandit18@bandit.labs.overthewire.org" (https://www.akashtrehan.com/writeups/OverTheWire/Bandit/level18/)

"If you turn pty allocation off with -T, sshd will use a pair of pipes instead of a bi-directional pty to communicate with the process running the remote command." (https://unix.stackexchange.com/questions/509158/ssh-

```
 disable-pseudo-terminal-allocation)
```

4.10 Level 19 → Level 20

```
Solution = GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Lesson =
 => Level Goal
 To gain access to the next level, you should use the setuid binary in the
 homedirectory. 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.

 Helpful Reading Material
 setuid on Wikipedia

 => Setuid et Setgid pour les exécutables

 Il s'agit de bits de contrôle d'accès appliqués aux fichiers et répertoires
 d'un système d'exploitation UNIX. Grâce à eux, un processus exécutant un
 tel fichier peut s'exécuter au nom d'un autre utilisateur.

 Quand un fichier exécutable est la propriété de l'utilisateur root, et est
 rendu setuid, tout processus exécutant ce fichier peut effectuer ses tâches
 avec les permissions associées à root, ce qui constitue un risque de
 sécurité pour la machine, s'il existe une faille dans ce programme.

 [...]

 Il est possible de voir si un fichier est setuid ou setgid en tapant la
 commande :

 $ ls -l nom_du_fichier
 -rwsr-sr-x 1 propriétaire groupe 158998 mars 12 17:12 nom_du_fichier

 Le s dans la première partie (réservée au propriétaire) -rws indique
 que le fichier est setuid, le s dans la deuxième partie (réservée au group)
 r-s indique que le fichier est setgid. Si le s est en minuscule c'est que
 le fichier est exécutable par contre s'il est majuscule c'est qu'il n'est
 pas exécutable.

 Il est également possible de lister tous les fichiers setuid et setgid du
 système grâce à la commande find / -type f -perm /6000 -ls 2>/dev/null"
 (https://fr.wikipedia.org/wiki/Setuid)


 =>
 bandit19@bandit:~$ ls -l
 total 8
```

```
-rwsr-x--- 1 bandit20 bandit19 7296 May  7 20:14 bandit20-do

bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
Example: ./bandit20-do id

In man => id - print real and effective user and group IDs

=>
bandit19@bandit:~$ ./bandit20-do id
uid=11019(bandit19) gid=11019(bandit19) euid=11020(bandit20)
groups=11019(bandit19)

"Effective user ID
The effective UID (euid) of a process is used for most access checks."
(https://en.wikipedia.org/wiki/User_identifier)

=>
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
GbKksEFF4yrVs6il55v6gwY5aVje5f0j

=> setuid [is a file/directory bit] allow[ing] users to run an executable
with the file system permissions of the executable's owner or group
respectively [...]. They are often used to allow users on a computer
system to run programs with temporarily elevated privileges in order to
perform a specific task. bandit20-do is a setuid executable (setuid is an
adjective), this executable lets the user runs command as bandit20
an executable file. bandit20-do normally belongs to bandit20 user but it
has the setuid bit so it can be executed by any process from any user as
the bandit20 user (setuid gives owner's rights to all other users). So any
user can execute this file which allows the execution of commands as
bandit20 user, thus explaining why we can get the password from bandit_pass
file.

Note: Other files can't be read by user bandit20 so ./bandit20-do can only
cat bandit20's password...
```

Chapter 5

Level 20 → Level 30

5.1 Level 20 → Level 21

```
Solution = gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

Lesson =
 Level Goal
 There is a setuid binary in the homedirectory that does the following: it
 makes a connection to localhost on the port you specify as a commandline
 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).

 NOTE: Try connecting to your own network daemon to see if it works as you
 think

 Commands you may need to solve this level ssh, nc, cat, bash, screen,
 tmux, Unix 'job control' (bg, fg, jobs, &, CTRL-Z, ...)

 =>
 "tmux is a terminal multiplexer: it enables a number of terminals to be
 created, accessed, and controlled from a single screen.  tmux may be
 detached from a screen and continue running in the background, then later
 reattached." (from man tmux)
 Useful shortcuts (Crl+b before any shortcut) : " -> horizontal split, % ->
 vertical split, arrows to move from panels, x to kill a panel
 Command: tmux attach-session <session_name>

 =>
 In tmux, panel 1:
 bandit20@bandit:~$ nc -l -p 35000
 GbKksEFF4yrVs6il55v6gwY5aVje5f0j

 "Netcat can also function as a server, by listening for inbound connections
 on arbitrary ports and then doing the same reading and writing." (from man
```

nc)

"Listen for an incoming connection rather than initiating a connection to a remote host." (from man nc)

With nc -l -p <whatever free port we want> we create a daemon, ie. a program running in the background (here a server, so it's a 'network daemon'...) waiting for a connexion on this port. Once the connexion is done, nc can read/write data from/on the chosen port. Here I chose to send the password of bandit20

=>
"Les daemons sont souvent démarrés lors du chargement du système d'exploitation, et servent en général à répondre à des requêtes du réseau, à l'activité du matériel ou à d'autres programmes en exécutant certaines tâches." (https://fr.wikipedia.org/wiki/Daemon_(informatique))

=>
In tmux panel 2:
bandit20@bandit:~$ ./suconnect 35000

Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Password matches, sending next password
gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

As send in the statement of this level, ./suconnect <chosen port> makes a connection on the indicated port and then waits for an input from the connection. If what it gets from the connection matches the password of bandit20, ./suconnect sends the password of bandit21.

=>
The difficulty was to understand to make a connection, ./suconnect had to find a server ready to make connections and if this server doesn't exists then we could make our own!

5.2 Level 21 → Level 22

Solution = Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

Lesson =

=>
bandit21@bandit:~$ cat /etc/cron.d
cat: /etc/cron.d: Is a directory

bandit21@bandit:~$ ls /etc/cron.d
cronjob_bandit15_root cronjob_bandit17_root cronjob_bandit22

```
cronjob_bandit23  cronjob_bandit24  cronjob_bandit25_root


=>
bandit21@bandit:~$ cat /etc/cron.d/cronjob_bandit22
@reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null

"Whatever you write to /dev/null will be discarded, forgotten into
the void. It's known as the null device in a UNIX system." (https:/
linuxhint.com/what_is_dev_null/)

&> means that both stdout and stderr are redirected to /dev/null.
The long version of &> is:
/usr/bin/cronjob_bandit22.sh > /dev/null 2>&1, 2> means redirection of
stderr (the "2") to whatever file is after. & means that it's not
what's after is not a filename it's a file descriptor (0 for stdin, 1 for
stdout, 2 for stderr, cf. https://en.wikipedia.org/wiki/File_descriptor)
so here stderr is redirected to stdout which is already redirected to /dev/
null as the command is interpreted as a whole, not sequentially...

So @reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null means that
at each connection from bandit22, each "reboot", the script
/usr/bin/cronjob... is executed and all its outputs destroyed. That's
why I got interested in this script.

=>
bandit21@bandit:~$ file /usr/bin/cronjob_bandit22.sh
/usr/bin/cronjob_bandit22.sh: Bourne-Again shell script, ASCII text
executable

bandit21@bandit:~$ cat /usr/bin/cronjob_bandit22.sh
#!/bin/bash
chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

I wanted to know what was inside the script, chmod 644 was interesting
because it means than any other users can read /tmp/t706...
and, furthermore, the password from bandit22 is being written in /tmp
t706... So, the password must be there!

=>
bandit21@bandit:~$ ls -l /tmp
ls: cannot open directory '/tmp': Permission denied

(drwxrws-wt 3973 root root 782336 Aug  2 15:32 tmp)

"[Sticky bit => t flag:] When a directory's sticky bit is set, the
filesystem treats the files in such directories in a special way so only
the file's owner, the directory's owner, or root user can rename or delete
```

```
the file. Without the sticky bit set, any user with write and execute
permissions for the directory can rename or delete contained files,
regardless of the file's owner." (https://en.wikipedia.org/wiki/Sticky_bit)

bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

A simple check reveals that we can't read /tmp meaning we can't know
what's inside /tmp with a ls command for example. But its permissions
don't necessarily apply to all the files it contains (cf. chmod -R for
this...) an we know from cron and the script above the existence of the
file and its rights. t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv could be the
password but it doesn't hurt checking what's in
t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv file. After a cat we find
Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI which looks like the password while
t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv was just a way to protect the password
from users' guesses.
```

5.3 Level 22 → Level 23

```
Solution = jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n

Lesson =
 bandit22@bandit:~$ cat /etc/cron.d/cronjob_bandit23
 @reboot bandit23 /usr/bin/cronjob_bandit23.sh  &> /dev/null
 * * * * * bandit23 /usr/bin/cronjob_bandit23.sh  &> /dev/null

 bandit22@bandit:~$ 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

 bandit22@bandit:~$ whoami
 bandit22
 bandit22@bandit:~$ echo "I am user bandit23" | md5sum | cut -d ' ' -f 1
 8ca319486bfbbc3663ea0fbe81326349

 bandit22@bandit:~$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
 jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n

 =>
 $myname is defined by $(whoami) which is launched as bandit23 by cron
 (I've tried whoami to check if there wasn't any particular things to
 add to the user name). The rest is fairly self-explanatory.
```

5.4 Level 23 → Level 24

5.5 Level 24 → Level 25

5.6 Level 25 → Level 26

5.7 Level 26 → Level 27

5.8 Level 27 → Level 28

5.9 Level 28 → Level 29

5.10 Level 29 → Level 30

Chapter 6

Level 30 → Level 33

6.1 Level 30 → Level 31

6.2 Level 31 → Level 32

6.3 Level 32 → Level 33