Filename -:

- cat ./-
- cat < -

Spaces within the filename:

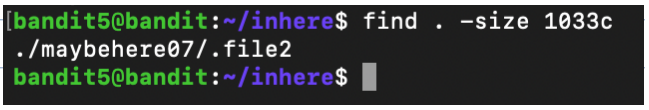
- Wrap the entire filename between quotes
- Escape every space using backslash key (do not put it next to cat or at the end if there are no spaces)

By default, when you log in, you're in the home directory. If you're not sure, use:

- cd ~
- cd

Level 4 human readable (determine the file type):

- ASCII text = human readable



```
[bandit5@bandit:~/inhere$ find . -size 1033c
./maybehere07/.file2
bandit5@bandit:~/inhere$ 
```

Level 5 Many files:

- You can execute find to narrow down your options
- find . -type f -size 1033c ! -executable

. = start search from current directory

-type f = search for only files

-size 1033c = files with exactly 1033 bytes (btw c = bytes)

! executable = not executable

Level 6 New command grep:

- grep = print lines that match patterns
- cd / = this makes it so you navigate to the top-level directory (search across entire server by starting at root directory)
- -user <username>: Search for files owned by a specific user.
- -group: Search for groups owned by specific group
- Add 2>/dev/null to hide "Permission Denied" errors for directories you can't access.
    - I didn't use grep but I were to:

find . -size 33c -user bandit7 -group bandit6 2>/dev/null

(returns --> ./var/lib/dpkg/info/bandit7.password)

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

Level 7 Next to the word millionth:

- grep "millionth" data.txt
- grep --> helps to find pattern

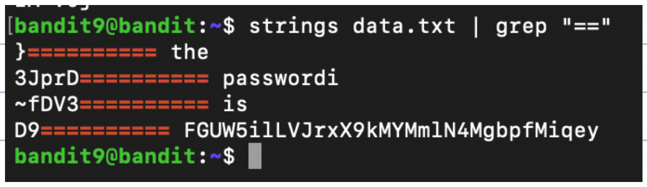
Level 8 Looking to remove duplicates:

- The password is the only line that appears exactly once in the file.

- To extract this password, you can use a combination of piping, redirection, and commands like grep, sort, and uniq.
- Piping (|): Sends the output of one command as the input to another.
- Redirection (> and >>): Save output to a file or append to an existing file.
- Use uniq -c to count occurrences of each line:
  - sort data.txt | uniq -c
- Filter the Line That Occurs Once: Use grep to find lines with a count of 1:
  - sort data.txt | uniq -c | grep " 1 "
    - The grep "1" accidentally account for 10 duplicates or passwords with a 1
- Try grep "^ *1 " instead: ^ *1 Matches lines where the first non-space character is 1 and it's followed by a space.

Note:

- Commands like uniq -c don't work "just by themselves" effectively because uniq relies on consecutive duplicate lines to perform its operations. If the input data is not sorted or structured properly, uniq will not detect duplicates accurately.



```
bandit9@bandit:~$ strings data.txt | grep "=="
}========== the
3JprD========== passwordi
~fDV3========== is
D9========== FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey
bandit9@bandit:~$ 
```

Level 9:

Binary File: file that contains data in a format that is not human-readable (encoded in binary format)

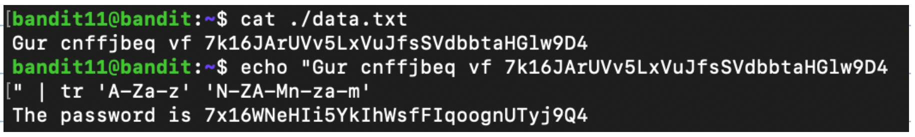
- different to text file (that contains readable characters like ASCII)
- Use xxd to generate a hex dump to inspect the file's content in detail
- You tried using the xxd command to view the content of data.txt in a hexadecimal format. This command translates the binary data into a human-readable representation (hex dump).

Level 10:

- Contains base64 encoded data
- base64 --> encodes or decodes data in base64 format
- Example --> echo "SGVsbG8sIFdvcmxkIQ==" | base64 --decode
- echo "VGhlIHBhc3N3b3JkIGlzIGR0UjE3M2ZaS2IwUlJzREZTR3NnMlJXbnBOVmozcVJyCg==" | base64 --decode
- The password is dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr

Level 11: All lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions:

- ROT13 is a substitution cipher, replacing each letter with the 13th one in the alphabet. It's a reciprocal cipher: encoding twice returns the original text.
- Simple obfuscation for jokes, spoilers, or non-sensitive data.
- Still readable if someone deciphers it, making it unsuitable for real encryption.
- Because ROT13 is its own inverse (applying it twice reverses the transformation), decoding is as simple as encoding --> because there is 26 letters in the alphabet
- With tr, you don't need to manually calculate the character rotations. The mappings you provide handle everything automatically.
  - 'A-Z' maps directly to 'N-ZA-M'
  - 'a-z' maps directly to 'n-za-m'



```
bandit11@bandit:~$ cat ./data.txt
Gur cnffjbeq vf 7k16JArUVv5LxVuJfsSVdbbtaHGlw9D4
bandit11@bandit:~$ echo "Gur cnffjbeq vf 7k16JArUVv5LxVuJfsSVdbbtaHGlw9D4
" | tr 'A-Za-z' 'N-ZA-Mn-za-m'
The password is 7x16WNeHIi5YkIhWsfFIqoognUTyj9Q4
```

    - tr 'A-Za-z' 'N-ZA-Mn-za-m'

Level 12:

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. For this level it may be useful to create a directory under /tmp in which you can work. Use mkdir with a hard to guess directory name. Or better, use the command “mktemp -d”. Then copy the datafile using cp, and rename it using mv (read the manpages!)

- mkdir: creates a new directory
- mktemp -d: can create a uniquely named directory for you
- cp: copies files or directories (copy file into new directory so you don't affect the original file)
- xxd: converts a file into a hexdump or vice versa
- file: to check current type so you can decompress (with gzip, bzip2, or tar)
- Note, ascii text is consistent with it being hexdump
- Note, xxd will create a new file for the conversion

Level 12 Steps:

1. tempdir=$(mktemp -d) --> to make dictionary
2. cd "$tempdir"t
    - Use $tempdir to access the value stored in the variable tempdir as opposed to referring to the literal string tempdir (if you didn't use $).

3. cp ~/data.txt "$tempdir" --> copies data.txt into this dictionary (you can verify that this file is in this directory using ls)

4. xxd -r data.txt new.data.file
    - Note, you can name the second argument as anything (I named it new.data.file). Note -r means reverse (will create a new file based off that name you give it)

5. Gzip detected --> decompress using (gzip -d --force new.data.file)
    - Rename the file if it doesn't detect the suffix .gz (new.data.file.gz)
    - mv new.data.file new.data.file.gz
    - gzip -d new.data.file.gz
    - file new.data.file (don't include gz anymore since it got decompressed)

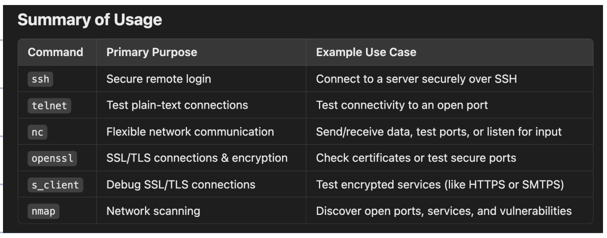
- Original file (new.data.file was gzip compressed) --> I decompressed it and got a new file called new.data.fille.out (which was bzip2 compressed) --> I decompressed it and now the new.data.file.out is identified as a tar archive POSIX format.

6. Tar archive detected --> tar -xvf new.data.file.out (extract contents of tar archive into the directory)

Level 13:

- Private key --> secure file to establish SSH session (only client has access to private key)
    - You will be provided private key to log in as bandit14 user
- ls to obtain file with private key
- Note: localhost is a hostname that refers to the machine you are working on
- Use ssh command with the ‘-i’ switch and specify the private key as a parameter.
    - bandit13@bandit:~$ ssh -i sshkey.private bandit14@localhost -p2220
    - Common Options:
    - -p [port]: Specify a custom port (default is 22).

- -i [file]: Use a private key file for authentication.


Summary of Usage

Command	Primary Purpose	Example Use Case
`ssh`	Secure remote login	Connect to a server securely over SSH
`telnet`	Test plain-text connections	Test connectivity to an open port
`nc`	Flexible network communication	Send/receive data, test ports, or listen for input
`openssl`	SSL/TLS connections & encryption	Check certificates or test secure ports
`s_client`	Debug SSL/TLS connections	Test encrypted services (like HTTPS or SMTPS)
`nmap`	Network scanning	Discover open ports, services, and vulnerabilities

Level 14:

- The password for the next level can be retrieved by submitting the password of the current level to port 30000 on localhost.
- nc localhost 30000
- The password for the current level is stored in the file /etc/bandit_pass/bandit14 (hence why you do cat)
- bandit14@bandit:~$ cat /etc/bandit_pass/bandit14

MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS

Level 15:

- 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.
  - ncat --> similar to nc, but SSL suport too (allows for secure connects using TLS/SSL (--ssl)
  - openssl --> implements SSL/TLS encryption (allows you to connect to SSL/TLS server)
    - openssl s_client -connect hostname:port
    - openssl s_client -connect www.google.com:443
  - s_client (openSSL subcommand) --> tests and debugs SSL/TLS connections to a server
  - ss --> modern replacement for netstat
- Port 30001 on localhost using SSL/TLS encryption

Note for Level 15: You can use either:

- ncat --ssl localhost 30001
- openssl s_client -connect localhost:30001

Level 16:

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

- Find which port has server listening on them
- Find which server speaks SSL/TLS and which doesn't
  - ncat --> to connect to the server
  - telnet --> connect to remote machines/servers over unsecured connections (allows you to send and receive raw text)(often replaced by modern tools such as nc and openssl)
  - socat --> redirecting and transfering data between two endpoints (transfers data between two addresses)
  - nmap --> network discovery (identifies open ports)
    - scan specific range too --> for example --> nmap -p 22-80 localhost

Level 16 Steps:

Step 1: namp range:

- Not shown: 996 closed tcp ports (conn-refused)
- PORT   STATE SERVICE

- 31046/tcp open unknown
- 31518/tcp open unknown
- 31691/tcp open unknown
- 31790/tcp open unknown
- 31960/tcp open unknown

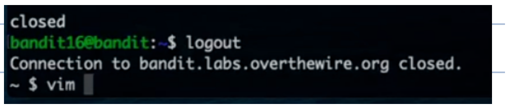
Step 2: See if it speaks SSL/TLS: openssl s_client -connect localhost:31790

Step: Returns private key so now i need to log into bandit17 (but i need to save the private key as a file so i can read the password)

- chmod 0400 bandit17_key
- ssh -i bandit17_key bandit17@bandit.labs.overthewire.org -p 2220

Level 17:

- 2 files in homedirectory: passwords.old and passwords.new
- next level password in password.new (only line that has been changed)
- diff (Compare Files Line by Line)
  - Compares two files line by line and outputs the differences.
    - <: Lines present only in the first file (e.g., file1.txt)
    - >: Lines present only in the second file (e.g., file2.txt)



```
closed
bandit16@bandit:~$ logout
Connection to bandit.labs.overthewire.org closed.
~ $ vim
```

Level 18:

- Password stored in a file readme in the homedirectory.
- Unfortunately, someone has modified .bashrc to log you out when you log in with SSH.
  - The issue you're facing occurs because the .bashrc file has been modified to automatically log you out when you log in via SSH. This means that the default behaviour of opening an interactive shell (ssh) is disrupted, making it difficult to perform standard commands like ls or cat to retrieve the file.
  - You need to retrieve the readme file in the home directory without triggering .bashrc.
    - ssh bandit18@bandit.labs.overthewire.org -p 2220

```
~
~
:wq
```

Level 18 More Explanation:

- Interactive SSH session (default behaviour, no command added at the end):
  - ssh bandit18@bandit.labs.overthewire.org -p 2220
  - This will start an interactive session (spawns an interactive shell) and configuration files are executed
- Non-interactive SSH session (when you add command at the end):
  - ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme
  - SSH skips opening an interactive shell and will directly execute the cat readme command on then the SSH will exit once the command has been completed.
    - Because no interactive shell was executed, .bashrc file was not run and didn't log me out immediately.

Level 19:

- Use setuid binary in the homedirectory
- Password found in usual place --> /etc/bandit_pass

ssh bandit19@bandit.labs.overthewire.org -p 2220

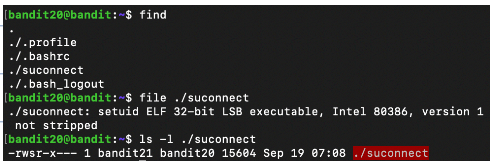
Level 20:

- setuid stands for "Set User ID"
- Allows program or executable to run with the permissions of the file's owner (instead of the user who runs the program)
    - Allows regular users to perform tasks that normally require higher privileges

-rwsr-xr-x 1 root root 12345 Oct 5 10:30 /usr/bin/passwd

- The s in rws indicates the setuid bit is set
    - The setuid bit allows the program bandit20-do to run as bandit20.
    - Your command (cat /etc/bandit_pass/bandit20) is executed with bandit20's privileges via the bandit20-do program.
    - This lets you bypass the permissions that would otherwise block bandit19.

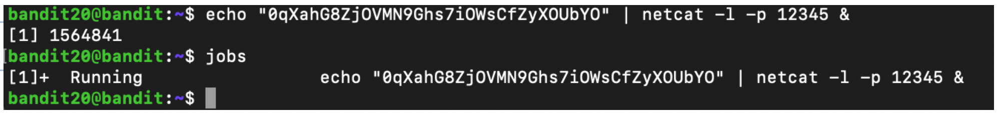
./bandit20-do cat /etc/bandit_pass/bandit20



```
bandit20@bandit:~$ find
.
./.profile
./.bashrc
./suconnect
./.bash_logout
bandit20@bandit:~$ file ./suconnect
./suconnect: setuid ELF 32-bit LSB executable, Intel 80386, version 1
 not stripped
bandit20@bandit:~$ ls -l ./suconnect
-rwsr-x--- 1 bandit21 bandit20 15604 Sep 19 07:08 ./suconnect
```

Level 20:

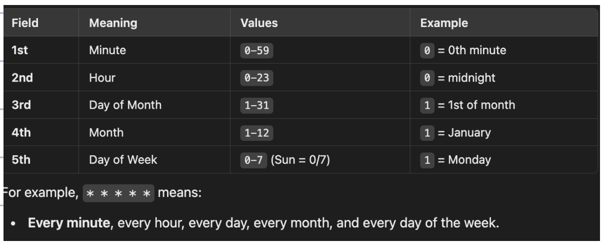
- When you run ./suconnect directly it will explain the purpose and usage of it
    - ls -l ./suconnect



```
bandit20@bandit:~$ echo "0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO" | netcat -l -p 12345 &
[1] 1564841
bandit20@bandit:~$ jobs
[1]+  Running                 echo "0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO" | netcat -l -p 12345 &
bandit20@bandit:~$
```

Level 20:

- Create a network listener (a daemon) on a port
    - Listener created using netcat
    - echo -n "0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO" | nc -l -p 1234 &
    - & makes it so you can run the command in the background so you can continually type commands in the same terminal
        - You can verify the listener is runing by using (jobs)
        - Note, port 1234 is an arbitrary port
        - -n prevents newline characters
        - netstat -tuln shows you which are available and not available (if it is listed, its not available)
- Use the binary to connect to that port (./suconnect 1234)
- Send the correct password when the binary connects
- Receive the next password



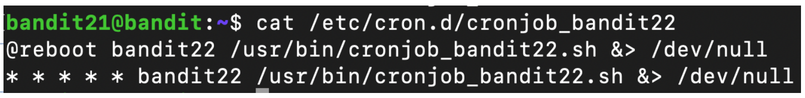
Field	Meaning	Values	Example
1st	Minute	0–59	0 = 0th minute
2nd	Hour	0–23	0 = midnight
3rd	Day of Month	1–31	1 = 1st of month
4th	Month	1–12	1 = January
5th	Day of Week	0–7 (Sun = 0/7)	1 = Monday

For example, `* * * * *` means:

- **Every minute**, every hour, every day, every month, and every day of the week.

Level 21:

- 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.



```
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
```

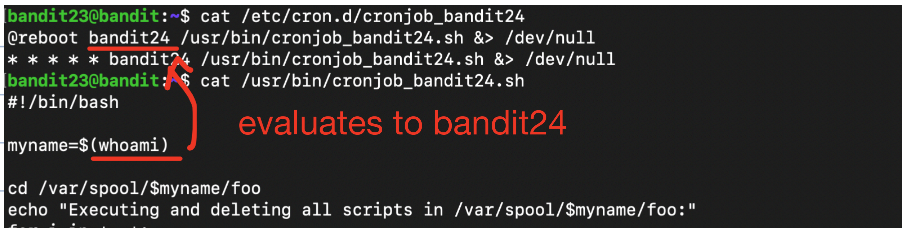
- The files in /etc/cron.d/ are system-wide configuration files that define tasks for specific users. Each file can have scheduled tasks with precise information about when they run and what commands they execute. These files might hold valuable information about how the system operates or, in this case, how the password for the next level can be accessed

- Note, /etc/cron.d/ is just a pathway (directory)
- Each file in /etc/cron.d/ is essentially a cron table --> must follow cron job syntax
- The path to the script or command being executed (e.g., /usr/bin/cronjob_bandit22.sh)
- Scripts often store their output in a temporary file (e.g., /tmp/)
- ls -l /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
- >: Redirects the output of the cat command to a file
- The chmod command ensures that the file can be read by other users

Level 22:

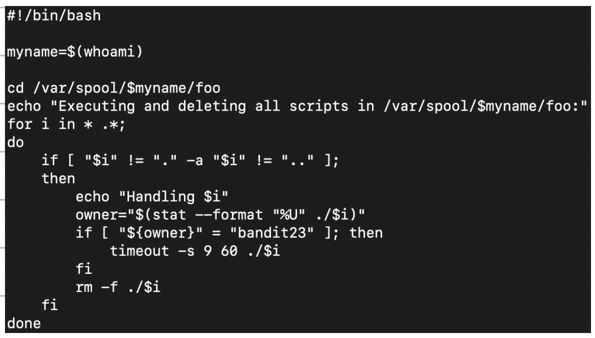
- Look in /etc/cron.d/ for the configuration. Did the easy aspects:
  - ls /etc/cron.d/
  - cat /etc/cron.d/cronjob_bandit23
  - cat /usr/bin/cronjob_bandit23.sh
  - whoami command --> used to display current user's username and their computer
  - md5sum computes the MD5 hash of the input text
  - echo I am user bandit23 | md5sum | cut -d ' ' -f 1
    - 8ca319486bfbbc3663ea0fbe81326349
  - cat /tmp/8ca319486bfbbc3663ea0fbe81326349

Level 23:



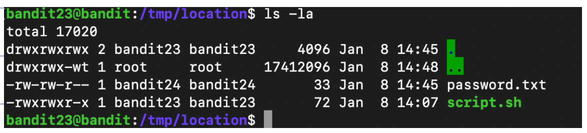
```
bandit23@bandit:~$ cat /etc/cron.d/cronjob_bandit24
@reboot bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
* * * * * bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
bandit23@bandit:~$ cat /usr/bin/cronjob_bandit24.sh
#!/bin/bash

myname=$(whoami)

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

- cat /usr/bin/cronjob_bandit24.sh
- The output will run all scripts in /var/spool/bandit24/foo if the file's owner is bandit23 but it will delete the script after execution.



```
#!/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
```

- You notice in the script that we are changing directory cd /var/spool/bandit24/foo
- The for loop means that it will iterate through all the files in the directory (* = anything)
- You also see that if the owner = bandit23 --> timeout (causes after certain period of time will kill command
- So it's doing exactly as it mentioned --> executing and deleting the scrips in this directory /var.



```
bandit23@bandit:/tmp/location$ ls -la
total 17020
drwxrwxrwx 2 bandit23 bandit23     4096 Jan  8 14:45 .
drwxrwx-wt 1 root     root     17412096 Jan  8 14:48 ..
-rw-rw-r-- 1 bandit24 bandit24       33 Jan  8 14:45 password.txt
-rwxrwxr-x 1 bandit23 bandit23       72 Jan  8 14:07 script.sh
bandit23@bandit:/tmp/location$
```

- Let's write a bash command that copies the password from bandit24 to a temporary location
  - mkdir /tmp/location then cd /tmp/location to access --> then make a script using text editor nano script.sh --> we now want to make a shell script
    - Note, scripts start with a shebang (#!) and we want to use bash so --> #!/bin/bash
  - Now, as bandit24 we want to read the password file:
    - #!/bin/bash
    - cat /etc/bandit_pass/bandit24 > /tmp/location/password.txt
    - Note, the script must be executable (note, that when we first create the script, it is infact not executable)
      - ls -l script.hs (-rw-rw-r--) --> means no execute permissions

            - chmod +x script.hs (this will make it executable for everyone) --> -rwxrwxr-x
    - Another thing, in our script, we made it so that the output of our cat command is written to the /tmp/ directory. We need to make sure that bandit24 has write permissions to that /tmp/ directory (check with ls -la)
        - chmod 777 . (where 777 = give permissions for absolutely everyone)
    - So now we will try to move the script and get it to execute
        - cp script.hs /var/spool/bandit24/foo
        - (ls /tmp/place to check for password.txt)

Note:

- You're exploring the /etc/cron.d directory using ls. This is a special system directory where cron jobs (automated tasks) are defined.
- You're reading the cronjob_bandit24.sh script (cat /usr/bin/cronjob_bandit24.sh)

What Happened Behind the Scenes?

- Once your script was in the /var/spool/bandit24/foo directory, the bandit24 cron job noticed it.
- The cron job ran your script with bandit24's permissions, meaning it could access /etc/bandit_pass/bandit24.
- Your script executed as follows:
    - It read the password from /etc/bandit_pass/bandit24.
    - It wrote the password to /tmp/place/password.txt, which you (as bandit23) could access.

Level 24:

- Daemon listening on port 30002 --> will give you password for bandit25 if given the password for bandit24 and a secret numeric 4-digit pin-code.
- There is no way to retrieve the pin-code except by going through all of the 10000 combinations, called brute-forcing (test all from 0000 to 9999)
    - nc localhost 30002 --> "I am the pin-code checker for user bandit25. Please enter the password for user bandit24 and the secret pin-code on a single line, separated by a space."
        - Note, nc creates connection with daemon on port 30002 and sends inputs
    - You can write a loop in a shell script to automate the process (make sure the script is executable)
        - Note after the loop generates all 10000 combinations, they are piped into a single connection to the daemon via nc (avoids opening new connection each attempt)
        - The grep -v command is used to filter out unwanted lines based on a pattern (like the reverse of normal grep)
        - Done = end of a for, while, or until loop
    - Note when running pwd to see which directory i live in --> if I'm in the (e.g., /home/banditXX, /etc), I won't be able to create or write files there unless I have sufficient permissions. Best create a temporary directory /tmp/directory.
    - In bash, variable assignments must not have spaces around the = sign

For Loop (solve for level 25):

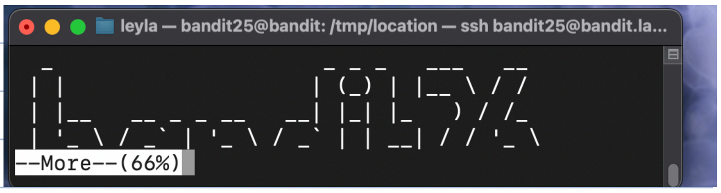
```
#!/bin/bash
```

```
password24="gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8"
for pin in {0000..9999}; do
    echo "$password24 $pin"
done | nc localhost 30002 | grep -v "Wrong!"
```

New commands:

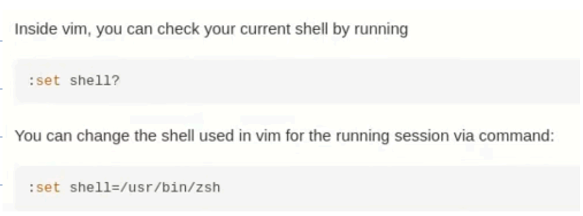
- more --> displays file contents page by page; handy for larger files
- vi --> this is a text editor
- id --> displays your user ID (UID) and group ID (GID) information
- pwd --> shows the full path of your current location in the filesystem
- Each user entry in /etc/passwd includes a field specifying the user's default shell
- More is like a way to display text

Level 25 --> 26:



```
leyla — bandit25@bandit: /tmp/location — ssh bandit25@bandit.la...
 _                     _ _ _   ___   __
| |                   | (_) | |__ \ / /
| |__   __ _ _ __   __| |_| |_   ) / /_
| '_ \ / _` | '_ \ / _` | | __| / / '_ \
--More--(66%)
```

- The shell for user bandit26 is not /bin/bash, but something else. Find out what it is, how it works and how to break out of it.
    - Found file bandit26.sshkey with RSA private key
    - cat /etc/passwd | grep bandit26 (This filters the /etc/passwd file to find only the line for bandit26, revealing the shell).
        - bandit26:x:11026:11026:bandit level 26:/home/bandit26:/usr/bin/showtext
        - Instead of a regular shell like /bin/bash --> the user has /usr/bin/showtext as their default shell. This is the restricted shell we need to investigate and bypass.



Inside vim, you can check your current shell by running

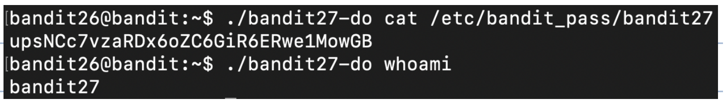
```
:set shell?
```

You can change the shell used in vim for the running session via command:

```
:set shell=/usr/bin/zsh
```

- The more command is taking a text document and it is displaying it (the Bandit26 style text). the way to think about this is that we want to stop the more ~/text.txt from going through. And we need to make it so that the BANDIT26 isn't displayed in one go --> so make the terminal really tiny. So now when we do that, we haven't been kicked out and it cannot fully display that ascii art.
- Now use the -v command to execute commands to use editor vi (now look at shell commands inside vi)
    - You can spawn the shell using :shell
    - :set shell? (this allows you to check what shell you are in)
    - :set shell=/bin/bash (allows you to change the shell)

Level 26:



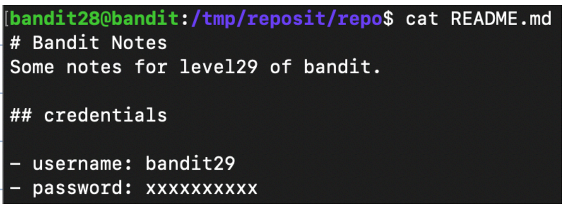
```
bandit26@bandit:~$ ./bandit27-do cat /etc/bandit_pass/bandit27
upsNCc7vzaRDx6oZC6GiR6ERwe1MowGB
bandit26@bandit:~$ ./bandit27-do whoami
bandit27
```

- bandit26@bandit:~$ file bandit27-do

bandit27-do: setuid ELF 32-bit LSB executable,
Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2,
BuildID[sha1]=368cd8ac4633fabdf3f4fb1c47a250634d6a8347, for GNU/Linux 3.2.0, not stripped

Level 27:

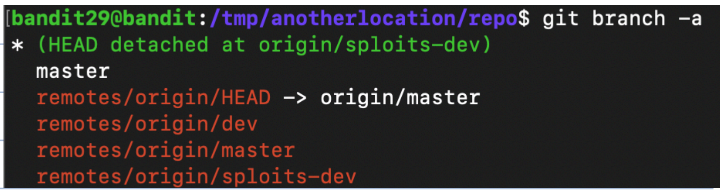
- There is a git repository at ssh://bandit27-git@localhost/home/bandit27-git/repo via the port 2220. The password for the user bandit27-git is the same as for the user bandit27. Clone the repository and find the password for the next level.
    - git clone = copying all the files from a repository from someone else to my computer
    - The last part of the URL (repo) is the default name for the directory that will hold the project on your local machine.
        - git clone in a directory where you have permission (so /tmp/)
- This wasn't too hard since because I didn't have write permissions I made a temporary directory. Then I cloned the git repository into the temporary directory and then cd repo and then cat README to get the password.



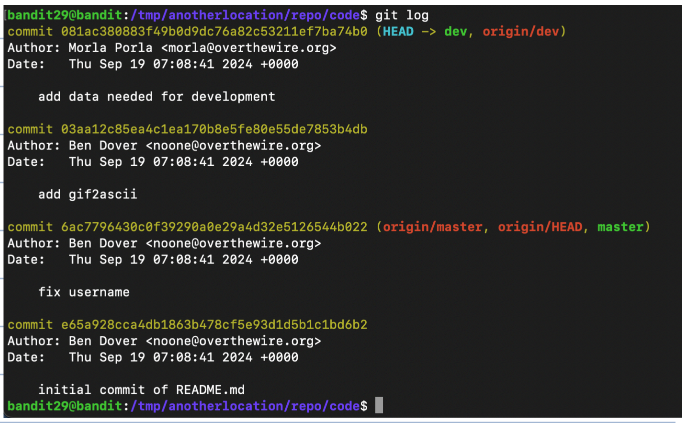
```
bandit28@bandit:/tmp/reposit/repo$ cat README.md
# Bandit Notes
Some notes for level29 of bandit.

## credentials

- username: bandit29
- password: xxxxxxxxxx
```

Level 28:

- Same as level 27 but make sure to specify it is 2220 port in there too (or else error thrown)
    - git clone ssh://bandit28-git@localhost:2220/home/bandit28-git/repo
    - Because it was a git repository, there could have been many versions of this file
    - git uses hashing as its way of storing things.
        - git log
        - git checkout <hash>
            - This will cause the pointer to change



```
bandit29@bandit:/tmp/anotherlocation/repo$ git branch -a
* (HEAD detached at origin/sploits-dev)
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/dev
  remotes/origin/master
  remotes/origin/sploits-dev
```



```
bandit29@bandit:/tmp/anotherlocation/repo/code$ git log
commit 081ac380883f49b0d9dc76a82c53211ef7ba74b0 (HEAD -> dev, origin/dev)
Author: Morla Porla <morla@overthewire.org>
Date:   Thu Sep 19 07:08:41 2024 +0000

    add data needed for development

commit 03aa12c85ea4c1ea170b8e5fe80e55de7853b4db
Author: Ben Dover <noone@overthewire.org>
Date:   Thu Sep 19 07:08:41 2024 +0000

    add gif2ascii

commit 6ac7796430c0f39290a0e29a4d32e5126544b022 (origin/master, origin/HEAD, master)
Author: Ben Dover <noone@overthewire.org>
Date:   Thu Sep 19 07:08:41 2024 +0000

    fix username

commit e65a928cca4db1863b478cf5e93d1d5b1c1bd6b2
Author: Ben Dover <noone@overthewire.org>
Date:   Thu Sep 19 07:08:41 2024 +0000

    initial commit of README.md
bandit29@bandit:/tmp/anotherlocation/repo/code$
```

Level 29:

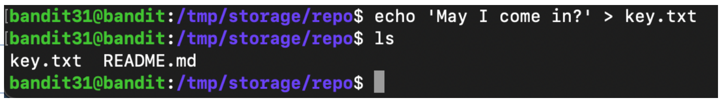
- A branch represents an independent line of development; you can move between branches to work on different tasks.
    - Use command git branch -a (to see all local and remote branches)
    - git checkout <branch_name> (to switch to existing branch)
        - There was a lot more information and logs in this branch then there were in the main
        - git checkout dev
- ssh://bandit29-git@localhost:2220/home/bandit29-git/repo
    - The password was present in the README.md file on the dev branch but not on the master branch. This is because different branches can contain different versions of files.

Level 30:

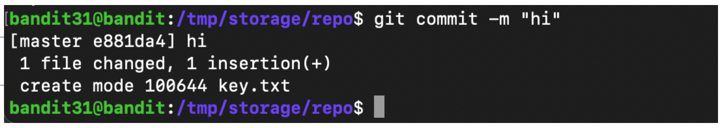
- git clone ssh://bandit30-git@localhost:2220/home/bandit30-git/repo
- Use git tag and git show to explore tags --> tags might point to commits containing hidden information like passwords (note, direct commit history doesn't show all commits)

- For example, ff a commit is created, then detached (e.g., it's not on any branch), it may become "orphaned" and won't appear in the typical commit history.

Level 31:



```
bandit31@bandit:/tmp/storage/repo$ echo 'May I come in?' > key.txt
bandit31@bandit:/tmp/storage/repo$ ls
key.txt  README.md
bandit31@bandit:/tmp/storage/repo$ 
```



```
bandit31@bandit:/tmp/storage/repo$ git commit -m "hi"
[master e881da4] hi
 1 file changed, 1 insertion(+)
 create mode 100644 key.txt
bandit31@bandit:/tmp/storage/repo$ 
```

- git clone ssh://bandit31-git@localhost:2220/home/bandit31-git/repo
- This time your task is to push a file to the remote repository (File name: key.txt, Content: 'May I come in?', Branch: master)
  - So I've made the key.txt, and when I go to add it to the master branch, there is a .gitignore file with an instruction to refuse the addition of the key.txt (the instruction being *.txt --> meaning all files with the .txt extension are being ignored by git). So instead of modifying that file, I'm just fonna force it anyway (git add -f key.txt) or you can (rm .gitignore)
  - Now we need to do the git commit -m "hello" (can be any message since it will show up in the commit log)

Level 32 (Last Level):

- Presented to the uppercase shell --> converts all user inputs to uppercase --> prevents linux commands (which are case sensitive and lowercase) from being executed
  - Ex. typing ls --> LS --> which it can't register as a command
- Typing $0 however, calls the shell without any case transformation (allowing ls, find etc)
- cat /etc/bandit_pass/bandit33 (whoami command to find bandit33)