Raul Pastor

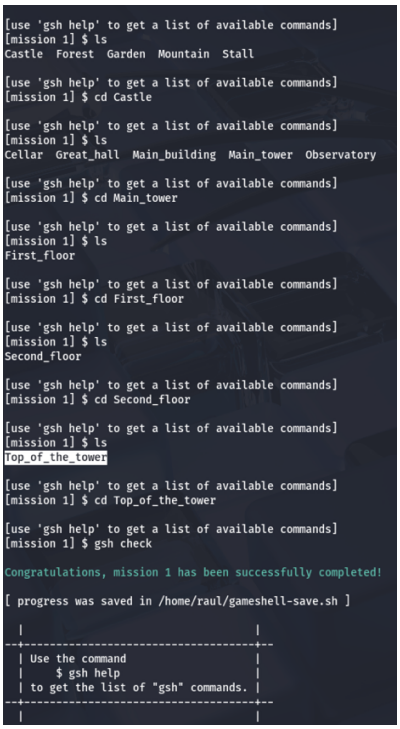
PROGETTO FINALE M2

Il progetto finale del secondo modulo sarà diviso in 2 parti ben distinte, la prima parte riguarda l'utilizzo dei comandi Linux attraverso GameShell installato sulla macchina Kali; Mentre la seconda parte riguarda la creazione di un programma (in C o Python…io ho scelto Python) che permetta l'esecuzione di un brute force attack ad un servizio SSH su una macchina Debian/Ubuntu, il programma verrà descritto nella seconda parte di questo file mentre lo script si trova qui: https://github.com/rauligno6/ConsegneCorsoCS/tree/main/PROGETTI_FINALI/M2.

Prima Parte:

la prima parte sarà divisa per ogni livello completato contenente le seguenti informazioni:
- screenshot di verifica dei comandi
- piccolo commento esplicativo del livello

Lv1)



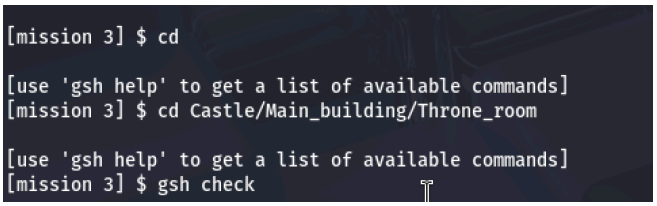
Il livello 1 consisteva solo nel raggiungere la cima della torre utilizzando il comando cd ed ls per spostarsi e capire "dove andare".

Raul Pastor

Lv2)

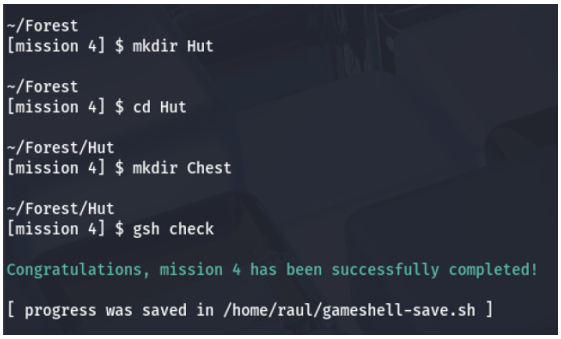
Il livello 2 aveva sempre lo stesso scopo del livello 1 ma in questo caso dovevamo raggiungere il 'Cellar'.

Lv3)



Il livello 3 chiedeva di arrivare alla 'Throne_room' solo con 2 comandi, quindi utilizzando cd per raggiungere la repository principale e successivamente inserendo il path completo nel secondo comando.

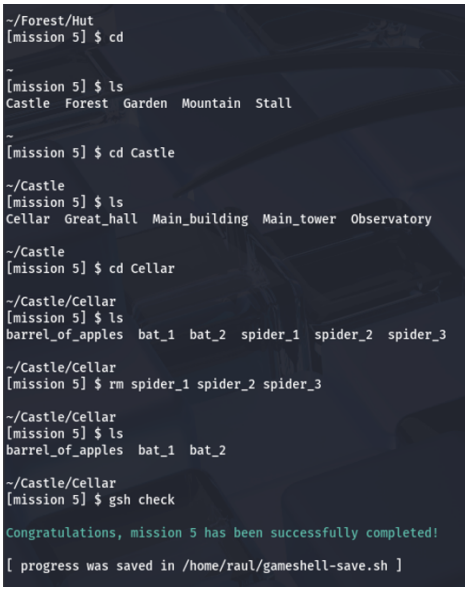
Lv4)



Il livello 4 richiedeva l'utilizzo di mkdir per creare le directory 'Hut' e 'Chest'.

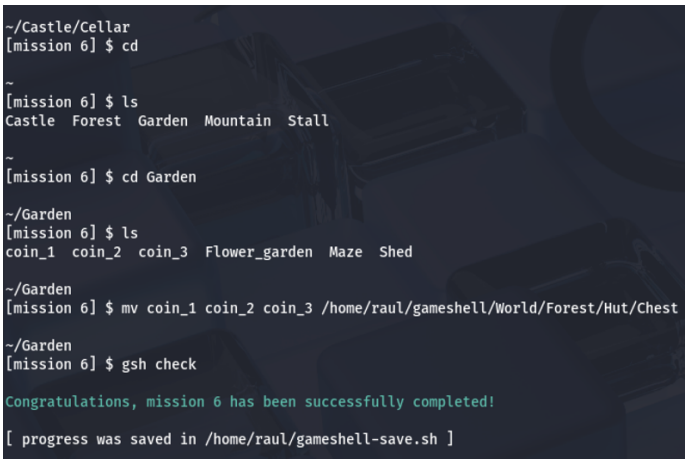
Raul Pastor

Lv5)



Il livello 5 richiedeva di usare rm per rimuovere i ragni dal ‘Cellar’.

Lv6)



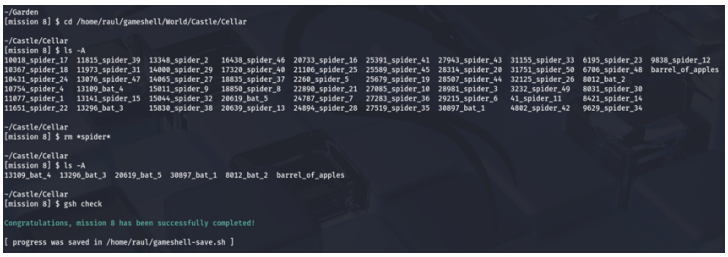
Il livello 6 richiedeva di spostare con mv i ‘coin’ dal garden alla nostra chest.

Lv7)

Il livello 7 richiedeva l’utilizzo di ls -A per verificare l’esistenza di ‘coin’ nascosti e spostarli nella nostra chest.

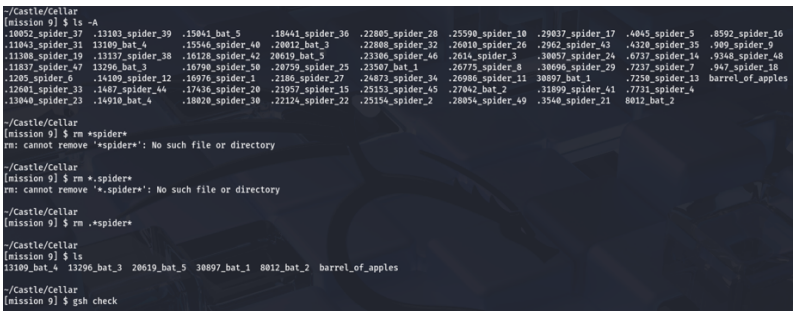
Raul Pastor

Lv8)



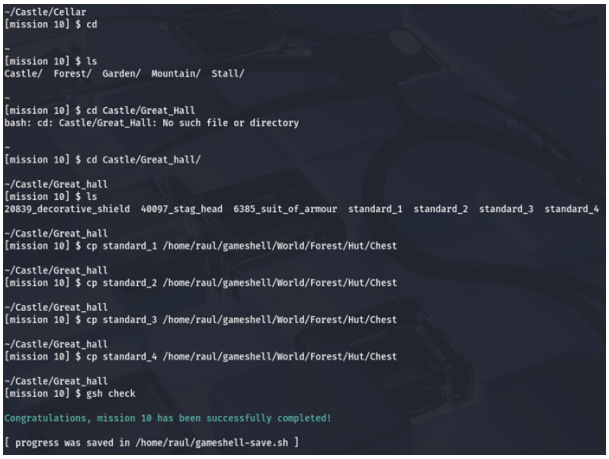
Il livello 8 richiedeva di tornare al ‘Cellar’ e rimuovere con rm i ragni nascosti utilizzando ‘*’ o ‘?’ per filtrare i nomi.

Lv9)



Il livello 9 aveva lo stesso obiettivo del precedente solamente applicando il carattere ‘.’ davanti al nome del file per mostrare la differenza nel filtro.

Lv10)



Il livello 10 richiedeva di copiare gli ‘standard’ dalla sala del trono alla nostra chest.

Lv11)

Il livello 11 richiedeva di copiare le ‘tapestry’ dalla Hall alla nostra chest utilizzando il carattere ‘*’.

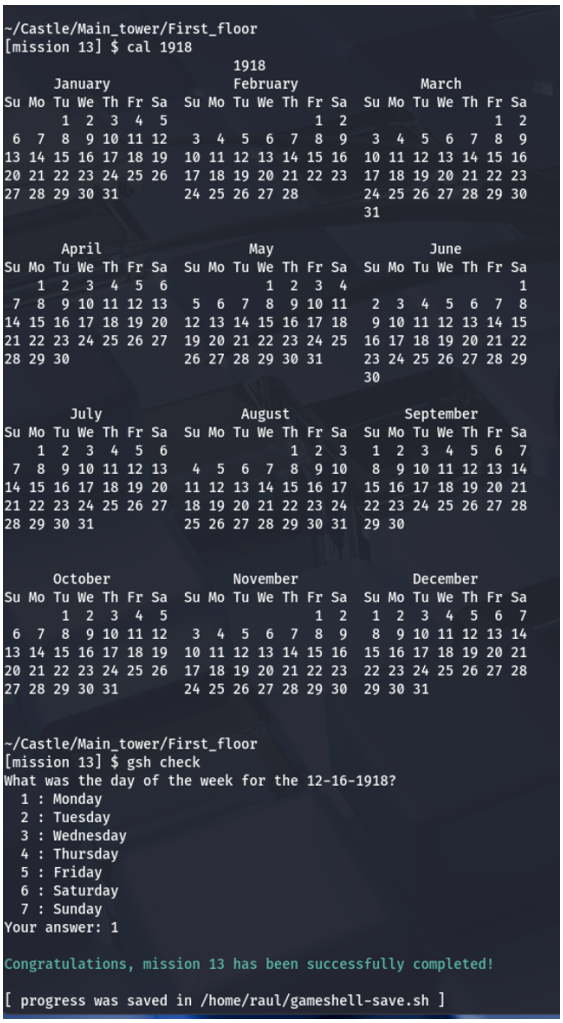
Raul Pastor

Lv12)

Il livello 12 insegna l'utilizzo dei comandi 'cat' e 'ls -l' per verificare il contenuto di un file e per verificare informazioni aggiuntive rispetto al semplice ls (come permessi e data creazione).

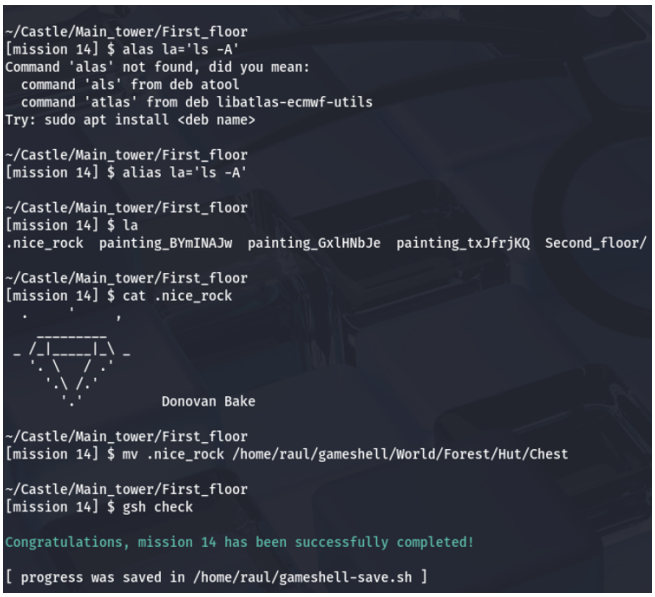
Raul Pastor

Lv13)



Il livello 13 insegna l'utilizzo di cal.

Lv14)



Il livello 14 insegna l'utilizzo di alias e come semplificare dei comandi assegnandoli a delle stringhe più semplici.

Raul Pastor

Lv15)

```
~/Castle/Main_tower/First_floor
[mission 15] $ cd /home/raul/gameshell/World/Forest/Hut/Chest

~/Forest/Hut/Chest
[mission 15] $ nano journal.txt

~/Forest/Hut/Chest
[mission 15] $ gsh check

Congratulations, mission 15 has been successfully completed!

[ progress was saved in /home/raul/gameshell-save.sh ]
```

Il livello 15 insegna ad utilizzare nano come editor di testo.

Lv16)

```
~/Forest/Hut/Chest
[mission 16] $ alias journal='nano /home/raul/gameshell/World/Forest/Hut/Chest/journal.txt'

~/Forest/Hut/Chest
[mission 16] $ journal

~/Forest/Hut/Chest
[mission 16] $ gsh check

Congratulations, mission 16 has been successfully completed!
```

Nel livello 16 ho creato un alias per poter modificare facilmente il ‘journal.txt’.

Lv17)

```
~/Castle/Cellar
[mission 17] $ cd .Lair_of_the_spider_queen\ UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu/

~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 17] $ ls
wSIAwCmRSOaGZIYE_baby_bat_jpbxtekxQfrUqHNP  zSXvjYrqwkLMMpPg_spider_queen_GExiPthIohTzRfck

~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 17] $ rm zSXvjYrqwkLMMpPg_spider_queen_GExiPthIohTzRfck

~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 17] $ gsh check
Perfect, it took you only 16 seconds to complete this mission!

Congratulations, mission 17 has been successfully completed!

[ progress was saved in /home/raul/gameshell-save.sh ]
```

Il livello 17 implicava di entrare nel covo del ragno e rimuovere il ragno regina, tutto in meno di 20s quindi implica l’utilizzo del tasto tab (essendo il livello contiene stringhe inutilmente lunghe).

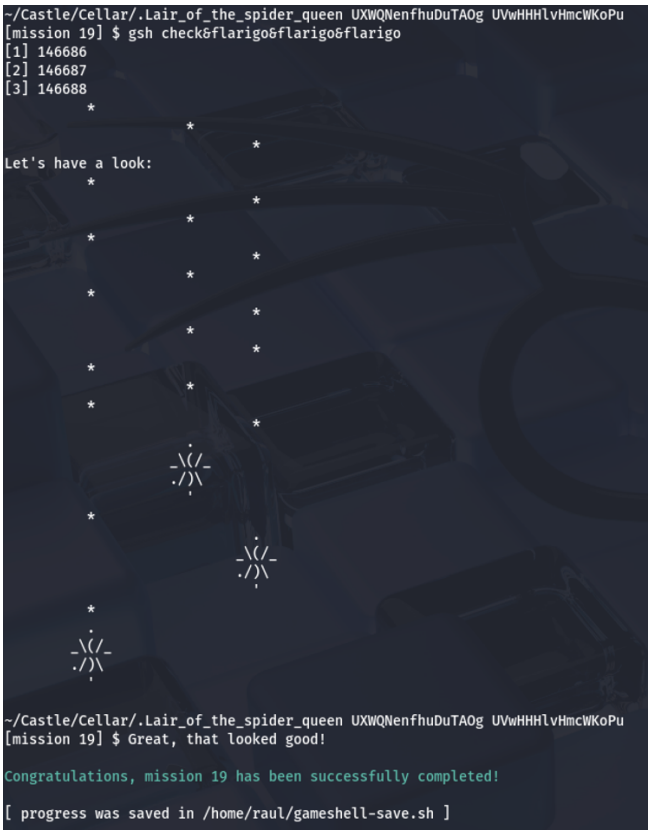
Lv18)

```
~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 18] $ xeyes
^C

~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 18] $ xeyes &
[1] 139637

~/Castle/Cellar/.Lair_of_the_spider_queen UXWQNenfhuDuTAOg UVwHHHlvHmcWKoPu
[mission 18] $ gsh check

Congratulations, mission 18 has been successfully completed!

[ progress was saved in /home/raul/gameshell-save.sh ]
```

Il livello 18 insegna l’utilizzo di ‘xeyes’.

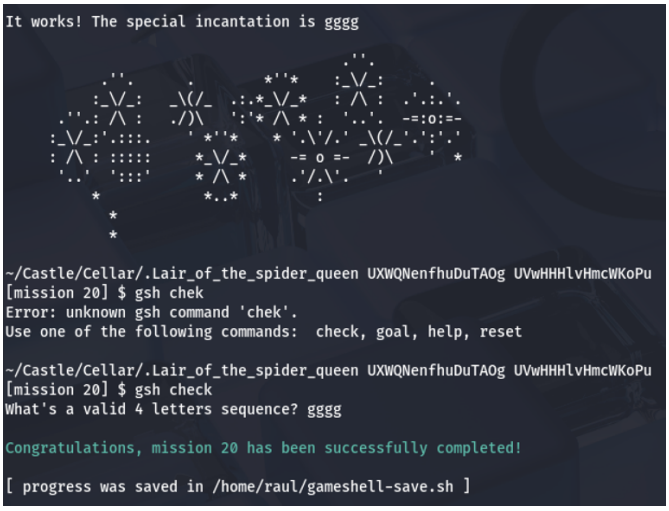
Raul Pastor

Lv19)



Il livello 19 insegna la concatenazione di diversi comandi attraverso il carattere‘&’.

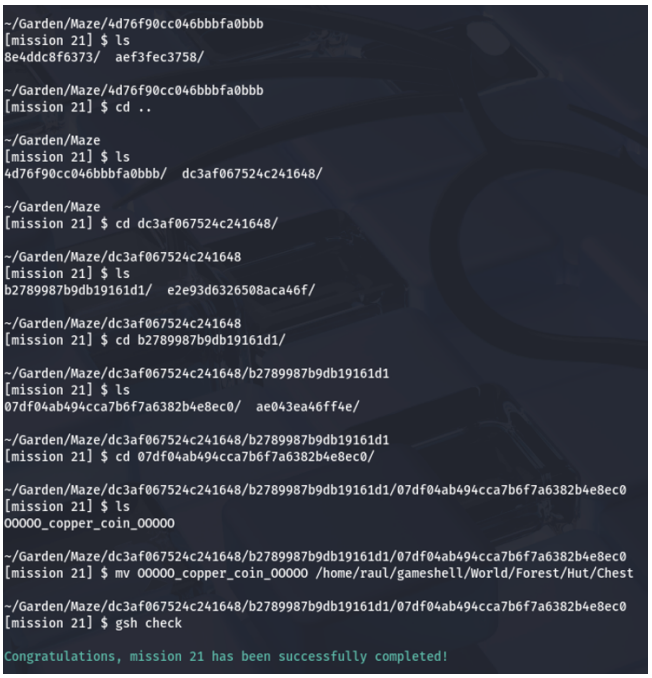
Lv20)



Nel livello 20 ho dovuto trovare una concatenazione di 4 caratteri che facesse funzionare l’incantesimo, dopo vari tentativi falliti (che dovevo fermare con ctrl+c) ho trovato ‘gggg’.

Raul Pastor

Lv21)



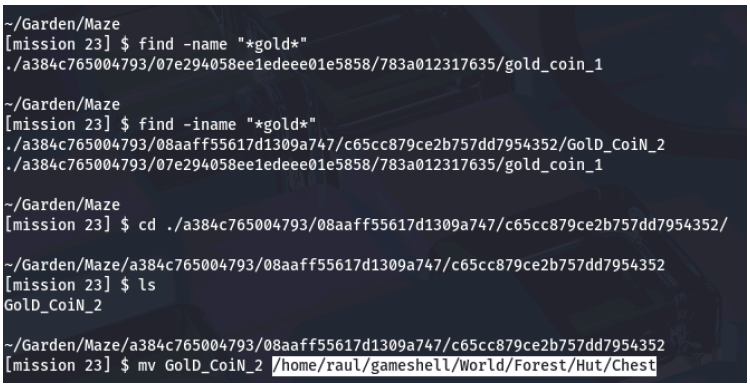
Il livello 21 implicava una semplice ricerca del ‘copper_coin’ dentro al labirinto utilizzando cd ed ls.

Lv22)

Il livello 22 implicava l’utilizzo del comando ‘ls -R’ per cercare la moneta d’argento e spostarla nella nostra solita chest.

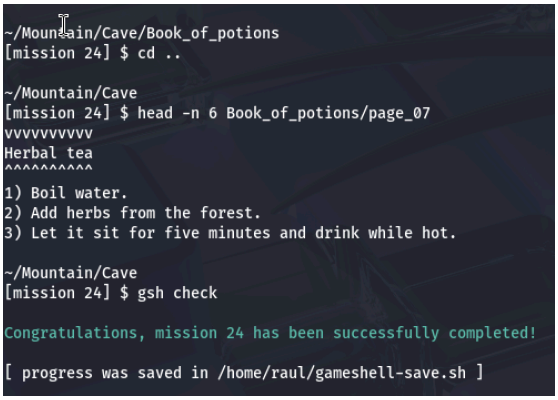
Raul Pastor

Lv23)



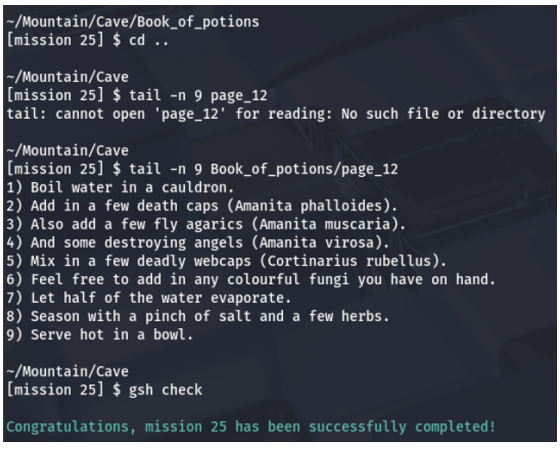
Il livello 23 consisteva nell'utilizzare il comando 'find' con l'opzione -iname per trovare tutti i nomi di file che contenessero ciò che si è inserito nell'argomento, nel mio caso *gold*.

Lv24)



Il livello 24 prevedeva l'utilizzo di head con l'opzione -n per vedere le prime K(6) righe, per trovare la ricetta dell'herbal tea.

Lv25)



Il livello 25 prevedeva l'utilizzo di tail con l'opzione -n per vedere le ultime K(9) righe, per trovare la ricetta dello stufato.

Raul Pastor

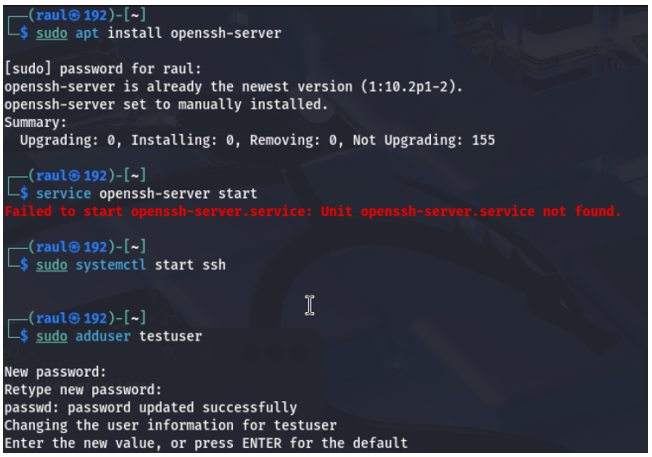
Lv26)

Il livello 26 prevedeva l’utilizzo di cat per concatenare due pagine (file) diverse(i).

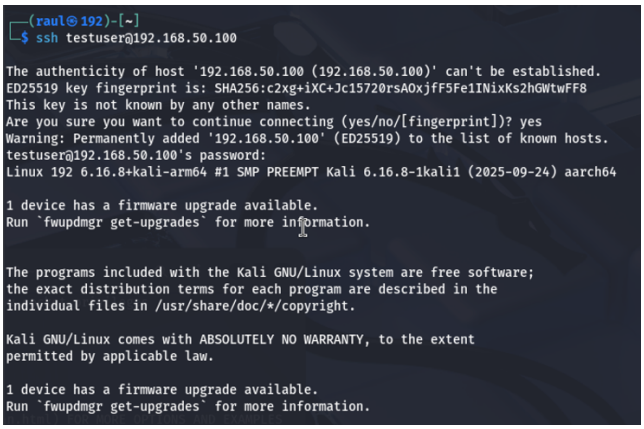
Raul Pastor

Seconda Parte:

Nella seconda parte, per prima cosa, ho configurato la macchina di destinazione installando e avviando il servizio OpenSSH Server. Ho creato un utente dedicato per il test come si vede dall'immagine successiva.



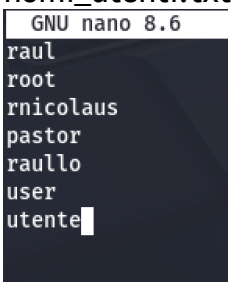
Dopo aver verificato manualmente la connessione tramite il comando ssh, ho proceduto allo sviluppo dello script brute-force.



Per implementare l'attacco ho utilizzato la libreria Python **Paramiko**. All'interno del programma ho importato le classi principali necessarie, in particolare SSHClient, AutoAddPolicy, e le eccezioni AuthenticationException e SSHException per identificare il risultato di ogni tentativo.

Lo script esegue un brute-force combinando username e password prelevati da due file di testo, nomi_utenti.txt e password.txt contenenti le liste di nomi utenti e password create.

nomi_utenti.txt



Raul Pastor

password.txt

```
  GNU nano 8.6
password
1234567
12345
ciao
admin
123456789
```

Una volta caricati questi file, l'utente deve inserire l'indirizzo IP della macchina target. A questo punto, il programma avvia due cicli for: per ogni username viene provata ogni password della lista. Ad ogni tentativo viene creato un nuovo client SSH tramite SSHClient(), così da garantire che ogni autenticazione sia indipendente dalle precedenti. Prima di connettersi, imposto la politica AutoAddPolicy(), che permette di accettare automaticamente la chiave host del server nel caso non sia ancora presente nel file known_hosts.

La connessione alla macchina avviene usando il metodo .connect() su porta 22 (precedentemente controllata con nmap) passando username e password.

```
┌──(raul㉿192)-[~]
└─$ nmap -sS 192.168.50.100 -p 22
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-18 20:04 GMT
Nmap scan report for 192.168.50.100 (192.168.50.100)
Host is up (0.000088s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Se le credenziali sono errate, Paramiko solleva un'AuthenticationException, che lo script cattura e segnala chiaramente come tentativo fallito. In caso di molti tentativi in rapida successione, il server SSH applica un ritardo crescente per contrastare attività anomale: questa situazione genera una SSHException, che lo script interpreta come rate-limiting, quindi attendendo qualche istante prima di riprovare. Se invece la connessione va a buon fine, significa che è stata trovata una combinazione valida di username e password. In questo caso il programma interrompe i tentativi, segnala il successo e mantiene aperta la sessione SSH, permettendo eventualmente altre operazioni fino alla chiusura manuale da parte dell'utente.

Durante lo sviluppo è stato necessario inserire delle piccole attese (time.sleep) tra un tentativo e l'altro, sia per non sovraccaricare il server SSH, sia per rendere l'esercizio più realistico.

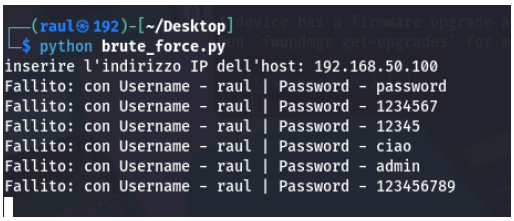
Di seguito il lancio del programma con gli output dell'attacco:

Richiesta di inserimento dell'indirizzo IP dopo aver lanciato lo script:

```
┌──(raul㉿192)-[~/Desktop]
└─$ python brute_force.py
inserire l'indirizzo IP dell'host: 192.168.50.100
```

Raul Pastor

Una serie di prove con username e password fallite:



Il messaggio di login e il print di username e password: