**Fatlind Lici**
**17/02/2026**

**W8_D4_CSPT25**

# Brut3-Forc3 ad un servizio SSH

**Task 2:**

Si richiede allo studente di scrivere un programma, con un linguaggio a sua scelta tra Python e C, che permetta l'esecuzione di un attacco Brute-Force ad un servizio SSH su una macchina Debian/Ubuntu (kali va benissimo come macchina di test).

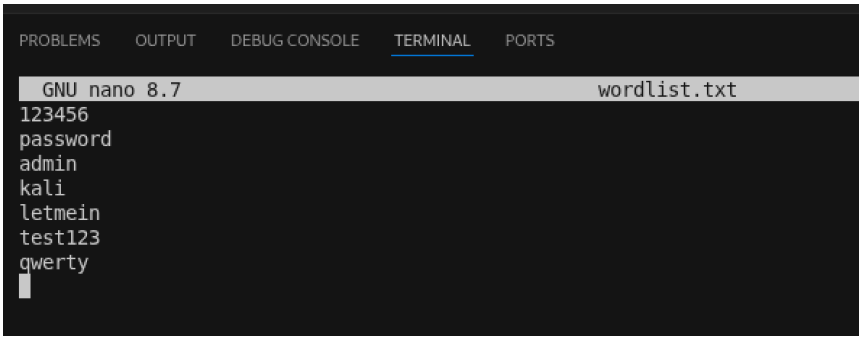
Codice:

W8D4task.py wordlist.txt

W8D4task.py > ...

```python
import random
import time


def load_usernames(path: str) -> list[str]:

    with open(path, "r", encoding="utf-8") as f:
        names = [
            line.strip()
            for line in f
            if line.strip() and not line.strip().startswith("#")
        ]

    if not names:
        raise ValueError("Il file non contiene username validi.")

    return names


def main() -> None:

    path = input("Path file username (es. users.txt): ").strip()

    usernames = load_usernames(path)

    # Scegli automaticamente il target dalla lista
    target_username = random.choice(usernames)

    print("\n[INFO] Username target selezionato automaticamente.")
    print("[INFO] Avvio simulazione...\n")

    attempts = 0
    start = time.time()

    # Mischia la lista per simulare ordine casuale
    random.shuffle(usernames)

    for guess in usernames:
        attempts += 1

        print(f"Tentativo {attempts}: {guess}")

        time.sleep(0.3)  # delay realistico

        if guess == target_username:
            elapsed = time.time() - start

            print("\nTROVATO!")
            print(f"Username: {guess}")
            print(f"Tentativi: {attempts}")
            print(f"Tempo: {elapsed:.4f} secondi")

            return

    print("\n[-] Username non trovato.")


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

File wordlist.txt: dobbiamo compilare un file con delle parole random tra le maggiori utilizzate per le password.

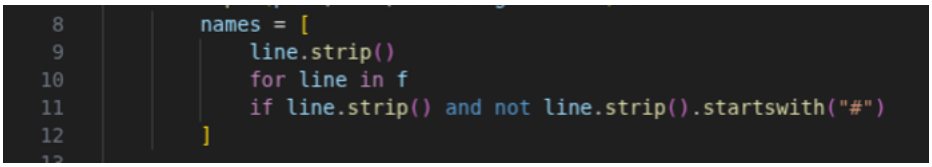



Spiegazione del codice:

- **import random:** libreria ceh serve per scegliere un riferimento a caso.
- **Import time:** libreria per misurare il tempo oppure creare ritardi.

- **def load_usernames(path: str) -> list[str]:** questa funzione prende il percorso del nostro file .txt e restituisce una lista di username.

- **with open(path, "r", encoding="utf-8") as f:** “r” significant read (lettura), per aprire il nostro file .txt.




Il codice sopra riportato è una **list comprehension,** controlla il contenuto del file .txt assicurandosi che non sia vuoto, e che le parole non inizino con # commenti dando come risultato una lista pulita.


Il codice sopra esegue il comando dove se il file .txt fosse vuoto, ci dia come risposta un errore.

- **def main() -> None:** è il contenuto del programma che verra eseguito.
- **path = input(“Path file username: “).strip():** comando che chiede dove si trova il file.txt ( il file .txt dovrà essere presente all’interno della cartella del programma.py altrimenti non lo trova.)
- **usernames = load_usernames(path):** legge il file e crea la lista.
- **target_username = random.choice(usernames):** sceglie a caso uno username e lo salva come password da trovare.
- **print("[INFO] Username target selezionato automaticamente.”):** funzione per chiarire il comando.

```
32      attempts = 0
33      start = time.time()
```

- **attempts = 0:** conta I tentativi.
- **start = time.time():** salva l'orario di partenza del programma e calcola il tempo.

- **random.shuffle(usernames):** funzione per mescolare la lista.
- **for guess in usernames:** prende uno username alla votla e lo prova.
- **attempts += 1:** ogni giro +1
- **print(f"Tentativo {attempts}: {guess}"):** per mostrare che cosa sta provando.
- **time.sleep(0.3):** aspetta 0.3 secondi.
- **if guess == target_username:** se quello provato sia uguale a quello corretto.
- **elapsed = time.time() - start:** ora attuale - ora iniziale = tempo totale.

```
48          print("\nTROVATO!")
49          print(f"Username: {guess}")
50          print(f"Tentativi: {attempts}")
51          print(f"Tempo: {elapsed:.4f} secondi")
```

Questo output mostra chi è, quanti tentativi e quanto tempo.

- **return:** fine del programma.

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

Ultima funzione, serve per dire se questo file.py viene eseguito direttamente avvialo dal main().

Conclusione:

Il programma legge una wordlist da file, seleziona automaticamente un username target e simula un attacco brute-force provando tutti i nomi in ordine casuale.
Per ogni tentativo introduce un delay, conta i tentativi e misura il tempo.
Quando trova il valore corretto, termina mostrando i risultati.
*Il programma è stato sviluppato solamente a scopo didattico, utilizzarlo al di fuori dall'ambiente didattico non è legale. Non ci assumiamo alcuna responsabilità del cattivo utilizzo.*