

Brut3-Forc3 ad un servizio SSH

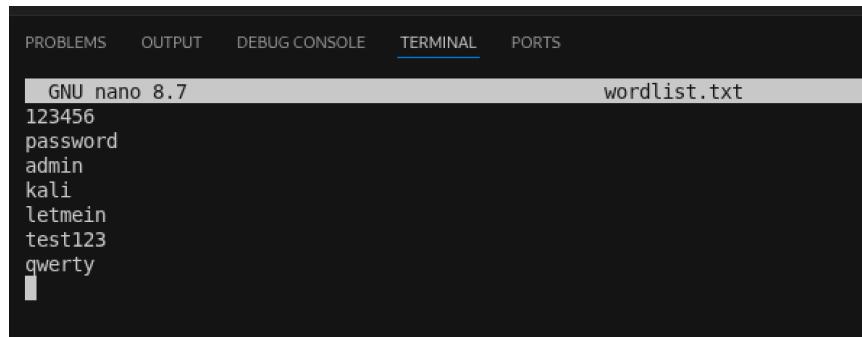
Task 2:

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 x wordlist.txt
W8D4task.py > ...
1 import random
2 import time
3
4
5 def load_usernames(path: str) -> list[str]:
6
7     with open(path, "r", encoding="utf-8") as f:
8         names = [
9             line.strip()
10            for line in f
11            if line.strip() and not line.strip().startswith("#")
12        ]
13
14    if not names:
15        raise ValueError("Il file non contiene username validi.")
16
17    return names
18
19
20 def main() -> None:
21
22     path = input("Path file username (es. users.txt): ").strip()
23
24     usernames = load_usernames(path)
25
26     # Scegli automaticamente il target dalla lista
27     target_username = random.choice(usernames)
28
29     print("\n[INFO] Username target selezionato automaticamente.")
30     print("[INFO] Avvio simulazione...\n")
31
32     attempts = 0
33     start = time.time()
34
35     # Mischia la lista per simulare ordine casuale
36     random.shuffle(usernames)
37
38     for guess in usernames:
39         attempts += 1
40
41         print(f"Tentativo {attempts}: {guess}")
42
43         time.sleep(0.3) # delay realistico
44
45         if guess == target_username:
46             elapsed = time.time() - start
47
48             print("\nTROVATO!")
49             print(f"Username: {guess}")
50             print(f"Tentativi: {attempts}")
51             print(f"Tempo: {elapsed:.4f} secondi")
52
53         return
54
55     print("\n[-] Username non trovato.")
56
57
58 if __name__ == "__main__":
59     main()
```

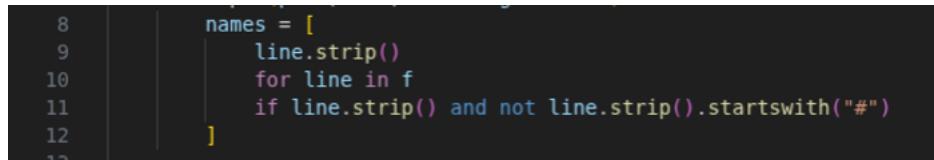
File wordlist.txt: dobbiamo compilare un file con delle parole random tra le maggiori utilizzate per le password.



```
GNU nano 8.7 wordlist.txt
123456
password
admin
kali
letmein
test123
qwerty
```

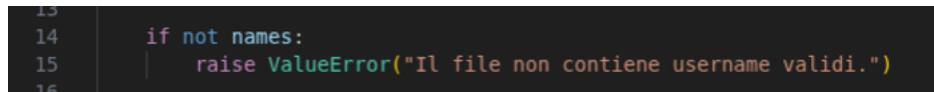
Spiegazione del codice:

- **import random:** libreria che 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.



```
8     names = [
9         line.strip()
10        for line in f
11        if line.strip() and not line.strip().startswith("#")
12    ]
13
14    if not names:
15        raise ValueError("Il file non contiene username validi.")
```

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.



```
13
14    if not names:
15        raise ValueError("Il file non contiene username validi.")
```

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 verrà 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.

```
31
32     attempts = 0
33     start = time.time()
34
```

- **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 volta 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
49         print("\nTROVATO!")
50         print(f"Username: {guess}")
51         print(f"tentativi: {attempts}")
52         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().

Esecuzione del programma:

```
(kali㉿kali)-[~/Desktop/codice/CodicePython]
$ python W8D4task.py
Path file username (es. users.txt): wordlist.txt

[INFO] Username target selezionato automaticamente.
[INFO] Avvio simulazione...

Tentativo 1: password
Tentativo 2: test123
Tentativo 3: letmein

TROVATO!
Username: letmein
Tentativi: 3
Tempo: 0.9088 secondi

(kali㉿kali)-[~/Desktop/codice/CodicePython]
```

Conclusioni:

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.