PassHackCheck

1. Introduzione ed Obiettivo

Questo documento descrive lo sviluppo di un'applicazione per la valutazione della sicurezza delle password, implementata in linguaggio C con supporto alla parallelizzazione tramite OpenMP. L'obiettivo principale è determinare se una password è vulnerabile a diversi tipi di attacchi, come brute force, attacchi a dizionario e social engineering. L'obiettivo è la realizzazione di un sistema che tenta di crackare le password con metodi comuni così che si sappia se la password è compromessa.

Si noti che il superamento del test non implica che la password sia sicura al 100% e che non ci sia possibilità che venga compromessa in futuro.

2. Metodo di Sviluppo: Code and Fix

Per lo sviluppo del progetto si è adottato il modello **Code and Fix**, un approccio informale e iterativo. Tale scelta è motivata dalla natura del progetto, che consiste in un programma compatto con funzionalità ben definite e un focus sull'implementazione pratica piuttosto che su una progettazione dettagliata a priori.

Motivazione della Scelta

- Il progetto ha una **complessità limitata**, rendendo superflua un'analisi estensiva prima della codifica.
- L'obiettivo principale è **testare la fattibilità tecnica** e sperimentare l'uso di OpenMP per la parallelizzazione.
- Il modello permette una **rapida iterazione**, utile per testare velocemente gli algoritmi e ottimizzare le prestazioni.

3. Analisi dei Requisiti

3.1 Requisiti Funzionali

L'applicazione deve:

1. **Verificare la vulnerabilità di una password** attraverso diversi attacchi:
    - **Brute Force**: prova tutte le combinazioni possibili.
    - **Dizionario**: confronta la password con un elenco di parole comuni.
    - **Social Engineering**: utilizza informazioni note per generare password probabili.
2. **Fornire un'indicazione del tempo stimato** necessario per attaccare la password.
3. **Supportare l'esecuzione parallela** per ottimizzare le prestazioni.

3.2 Requisiti Non Funzionali

- **Efficienza**: il programma deve ridurre al minimo i tempi di calcolo utilizzando più thread.
- **Portabilità**: il codice deve essere eseguibile su sistemi basati su Linux.
- **Scalabilità**: l'implementazione deve poter essere estesa con nuovi algoritmi di attacco.

4. Scelta della Parallelizzazione

La decisione di utilizzare **OpenMP** per la parallelizzazione è stata presa per ottimizzare i tempi di esecuzione, soprattutto per gli attacchi brute force e dizionario.

Vantaggi della Parallelizzazione

- **Riduzione del tempo di calcolo** suddividendo il lavoro tra più core CPU.
- **Scalabilità**: possibilità di sfruttare macchine con più thread hardware.
- **Semplicità di implementazione** grazie alle direttive di OpenMP.

Attacchi brute Force possono essere molto onerosi. parallerizzare consente di ridurre drasticamente i tempi di attesa.

5. Scenari e Casi d'Uso

Scenario 1: Utente Generico

1. L'utente inserisce una password da testare.
2. Il sistema esegue gli attacchi selezionati.
3. Il sistema restituisce se è riuscito a crakkare la password o meno.

6. Tecnologie Utilizzate

1. **Linguaggio di Programmazione: C** Il linguaggio C è stato scelto per lo sviluppo di questo progetto principalmente per la sua compatibilità con **OpenMP**, una libreria di parallelismo che permette di parallelizzare le operazioni e ottimizzare il calcolo delle password. La scelta del linguaggio non si basa sull'efficienza intrinseca, ma sulla facilità con cui C supporta il parallelismo, che è fondamentale per accelerare gli attacchi alle password.

2. **OpenMP (Open Multi-Processing) OpenMP** è una libreria che consente di eseguire il parallelismo su sistemi multi-core. È stato integrato nel progetto per migliorare le prestazioni nell'esecuzione di attacchi come **brute-force** e **dizionario**, che richiedono un gran numero di calcoli. Utilizzando direttive speciali all'interno del codice C, OpenMP consente di suddividere il carico di lavoro tra i core disponibili, riducendo significativamente i tempi di esecuzione. Ad esempio, durante l'esecuzione

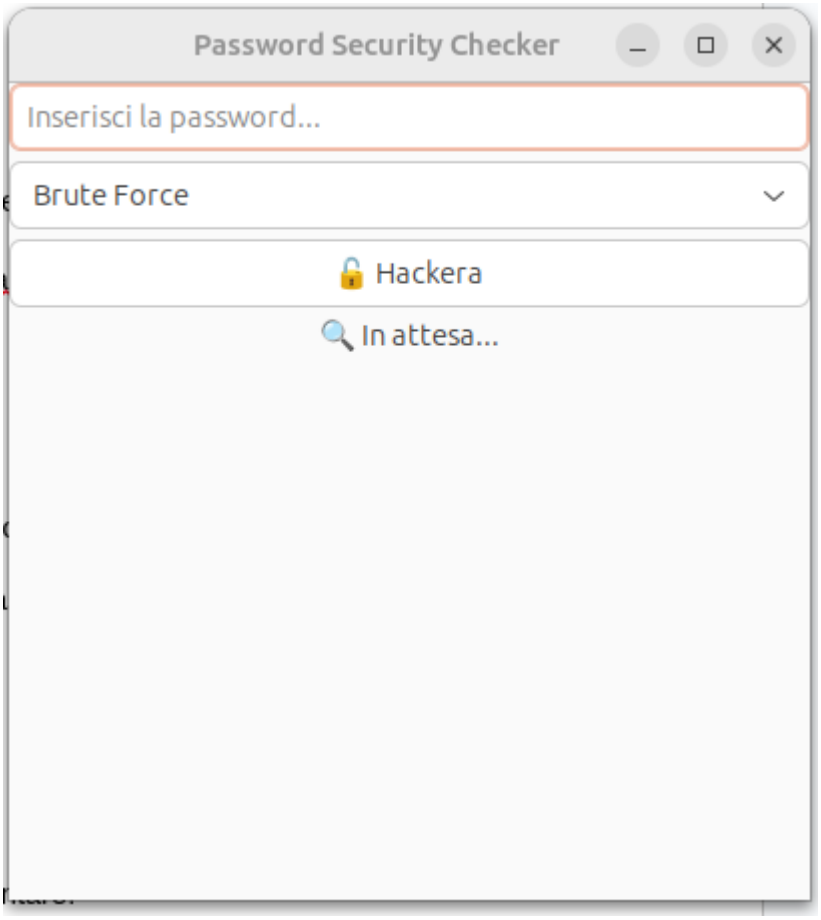
di attacchi con dizionari o brute force, la possibilità di suddividere il lavoro tra più thread consente di testare simultaneamente più combinazioni, migliorando la velocità del processo.

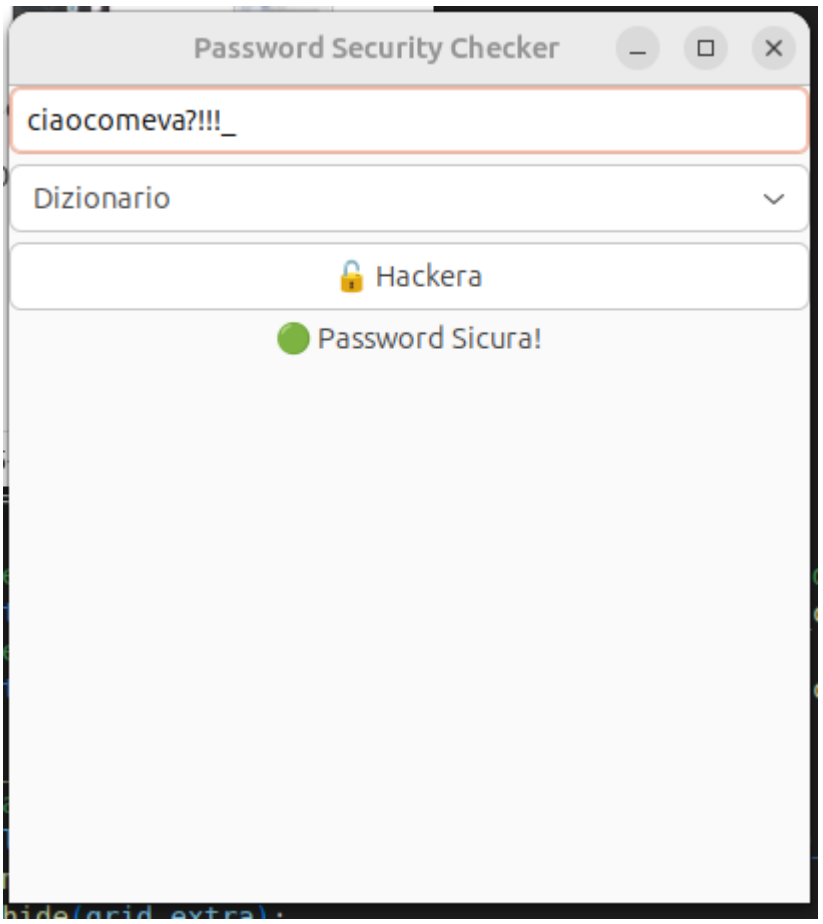
3. **GTK+ 3.0** Per l'interfaccia grafica, è stato utilizzato **GTK+ 3.0**, una libreria di sviluppo di interfacce utente in C. GTK è ampiamente usata per creare applicazioni con interfacce grafiche su piattaforme Linux, ma è anche compatibile con altri sistemi operativi come Windows e macOS. La libreria offre numerosi widget, come finestre, pulsanti, etichette e menu a tendina, che sono stati utilizzati per raccogliere l'input dell'utente (ad esempio, la password e il tipo di attacco) e visualizzare i risultati.

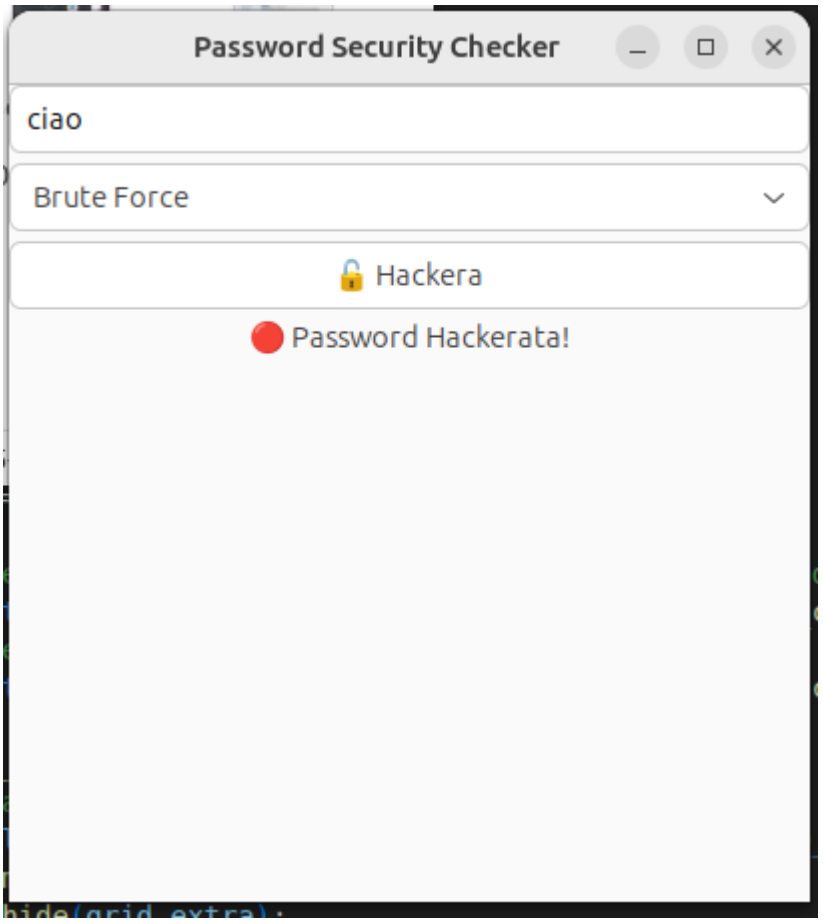
4. **GCC (GNU Compiler Collection)** Il progetto è stato compilato utilizzando **GCC**, uno dei compilatori più diffusi per il linguaggio C. GCC è noto per la sua compatibilità con diverse librerie, la sua efficienza nel generare codice ottimizzato e la possibilità di abilitare il supporto per il parallelismo con OpenMP tramite l'opzione `-fopenmp`.

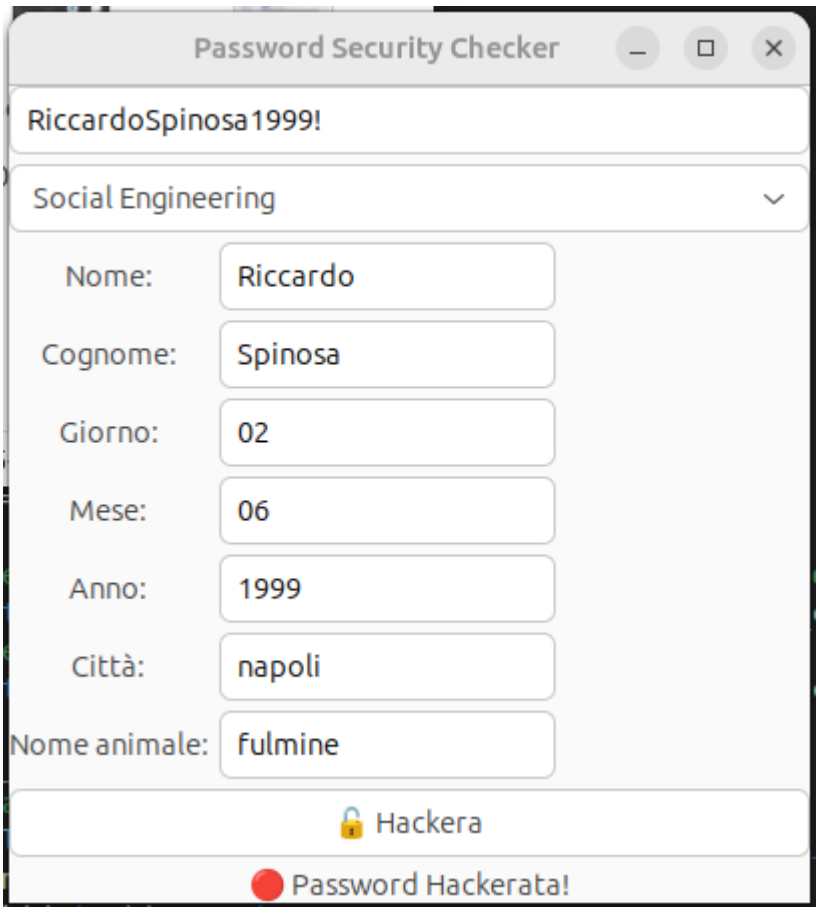
5. **Librerie per gli Attacchi (Brute Force, Dizionario, Social Engineering)** Le funzioni per i vari attacchi alle password (brute force, dizionario, social engineering) sono state implementate separatamente in moduli e integrate nel programma principale. Ogni attacco esegue una serie di operazioni specifiche, come il confronto della password con un dizionario o l'analisi di informazioni legate a un attacco di social engineering.

7. Mockup Interfaccia Grafica


Password Security Checker
Inserisci la password...
Brute Force
🔓 Hackera
🔍 In attesa...


Password Security Checker
ciaocomeva?!!!_
Dizionario
🔓 Hackera
🟢 Password Sicura!


Password Security Checker
ciao
Brute Force
🔓 Hackera
🔴 Password Hackerata!



8. Future Implementazioni

Per espandere il progetto, si potrebbero implementare:

- **Attacchi basati su hash** con tecniche di rainbow table.
- **Supporto a GPU computing** per velocizzare brute force.
- **Integrazione con database di password compromesse**.
- **Interfaccia utente completa** con statistiche dettagliate.

9. Conclusione

Il progetto ha dimostrato la fattibilità di un sistema per la valutazione della sicurezza delle password, con particolare attenzione alla parallelizzazione. L'obiettivo era di realizzare un sistema semplice che ci consentisse di scegliere tra gli attacchi disponibili e testare la sicurezza della password inserita.