**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr.2

la cursul „Proiectarea Sistemelor Informaționale”

[Definirea specificațiilor tehnice pentru sistemul informațional](https://else.fcim.utm.md/course/view.php?id=1667)

A efectuat: **Chihai Adrian,** **SI-211**

A verificat: **Magdei Octavian**

**Chișinău 2024**

**CUPRINS**

[Introducere 3](#_Toc184050087)

[1. Descriere generală 4](#_Toc184050088)

[2. Colectarea cerintelor 5](#_Toc184050089)

[2.1. Cerințe funcționale și nefuncționale 6](#_Toc184050090)

[3. Analiza cerintelor 7](#_Toc184050091)

[3.1 Analiza Cerințelor Funcționale 7](#_Toc184050092)

[3.2 Analiza Cerințelor Nefuncționale 8](#_Toc184050093)

[4. Documentarea cerințelor colectate 10](#_Toc184050094)

[4.1 Descriere generală 10](#_Toc184050095)

[4.2 Cerințe de interfață externă 10](#_Toc184050096)

[4.3 Caracteristica sistemului de atac brute-force 11](#_Toc184050097)

[4.4 Caracteristica sistemului de management al datelor colectate 12](#_Toc184050098)

[4.5 Documentația proiectului 16](#_Toc184050099)

[4.5.1 Modulul de procesare bruteforce 16](#_Toc184050100)

[4.5.2 Modulul de analiză și raportare a progresului 16](#_Toc184050101)

[4.6 Documentația utilizatorului 17](#_Toc184050102)

[4.6.1 Ghid pentru procesarea hash-urilor 17](#_Toc184050103)

[4.6.2 Ghid pentru vizualizarea și exportarea rezultatelor 18](#_Toc184050104)

[4.6.3 Scenarii de utilizare comune 18](#_Toc184050105)

[4.6.4 Soluții pentru probleme frecvente 18](#_Toc184050106)

[Concluzie 20](#_Toc184050107)

[Referințe bibliografice 21](#_Toc184050108)

# Introducere

Proiectul se concentrează pe crearea unui sistem de gestionare și analiză a vânzărilor de produse cu scopul de a optimiza procesele operaționale și de a oferi recomandări bazate pe date pentru a ajuta companiile să ia mai bune decizii. Utilizatorii al sistemului sunt managerii de vânzări, echipele de marketing și administratorii sistemului, care lucrează cu platforma pentru a urmări vânzările și a face rapoarte adecvate.

Prin furnizarea unei soluții accesibile și ușor de înțeles, proiectul va garanta dezvoltarea coerentă și utilizarea eficientă a sistemului. Aceasta va ajuta la gestionarea și analiza vânzărilor ușor și eficient. Funcționalitățile sofisticate vor ajuta utilizatorii să facă rapoarte amănunțite, să analizeze tendințe de vânzări și să facă recomandări personalizate. Sistemul va fi construit pentru a satisface cerințele unice ale utilizatorilor, combinând funcționalități ușor de accesat cu funcționalități sofisticate care îmbunătățesc eficiența și ajută la luarea deciziilor bazate pe date.

Un obiectiv esențial al sistemului este simplificarea procesului de colectare și analiză a datelor de vânzări. Aceste acțiuni sunt esențiale pentru optimizarea stocurilor, monitorizarea performanței produselor și gestionarea eficientă a comenzilor. Gestionarea și actualizarea informațiilor în timp real vor oferi o imagine clară a progresului vânzărilor și vor permite crearea de rapoarte individualizate pentru evaluarea performanței echipelor de vânzări, analiza costurilor și satisfacția clienților. În plus, sistemul va oferi recomandări automate pentru optimizarea strategiilor de marketing și gestionarea optimă a inventarului pe baza tendințelor de vânzări și a comportamentului clienților.

Creșterea profitabilității și îmbunătățirea performanței generale vor rezulta din optimizarea sistemului a utilizării resurselor. Funcționalitățile vor permite monitorizarea în timp real a activităților de vânzări, oferind acces la informații amănunțite și analize importante. Pentru a face alegeri strategice care optimizează afacerea, echipele de vânzări vor putea evalua performanța produselor, administratorii vor primi rapoarte detaliate și recomandări personalizate bazate pe date.

# 

# Descriere generală

Proiectul **„Analiză și Implementare a Tehnicilor de Bruteforce pentru Cracking-ul Hash-urilor și integrarea unui GUI”** își propune să ofere o abordare educațională pentru înțelegerea proceselor și vulnerabilităților asociate algoritmilor de hash criptografic. Într-un context în care hash-urile sunt utilizate pe scară largă pentru verificarea integrității datelor și autentificarea parolelor, explorarea modului în care aceste funcții pot fi compromise este esențială pentru îmbunătățirea securității informatice.

Prin acest proiect, se va dezvolta o aplicație software ce implementează tehnici de **bruteforce**, o metodă de încercare sistematică a tuturor combinațiilor posibile pentru a descifra un hash. Aplicația va include o **interfață grafică intuitivă (GUI)**, care va permite utilizatorilor să configureze parametrii de atac, să urmărească progresul în timp real și să vizualizeze rezultatele obținute. Aceasta va oferi suport pentru algoritmi de hash precum **MD5, SHA-1 și SHA-256**, fiind astfel relevantă pentru majoritatea scenariilor educaționale și de cercetare.

În cadrul proiectului, se va pune un accent deosebit pe procesul de inginerie a cerințelor. Se va începe cu colectarea cerințelor de la părțile interesate, inclusiv utilizatori finali, profesori și studenți, urmând analiza acestora pentru a stabili specificațiile sistemului. Dezvoltarea va implica proiectarea și implementarea unor algoritmi eficienți pentru spargerea hash-urilor, optimizarea proceselor de calcul și crearea unei interfețe grafice funcționale și atractive.

Aplicația este destinată utilizării în scopuri educaționale și de cercetare, oferind o platformă prin care utilizatorii pot învăța despre provocările asociate securității criptografice într-un mediu controlat. Prin utilizarea acestei aplicații, utilizatorii vor putea observa direct cum funcționează atacurile de tip bruteforce și vor înțelege importanța utilizării unor algoritmi de hash puternici și a unor măsuri de protecție adecvate, cum ar fi crearea parolelor complexe.

Proiectul combină în mod armonios elementele teoretice și practice ale securității cibernetice, promovând înțelegerea profundă a tehnicilor de atac și a mecanismelor de apărare. Prin integrarea unui GUI, aplicația este accesibilă chiar și pentru utilizatorii mai puțin experimentați, consolidându-i utilitatea ca instrument educațional și contribuind la pregătirea viitorilor specialiști în securitate cibernetică.

# Colectarea cerintelor

Având în vedere modul complex în care interacționează diferitele componente și utilizatorii unui sistem de securitate cibernetică, colectarea cerințelor este un pas esențial în dezvoltarea unui software dedicat analizei și implementării tehnicilor de bruteforce pentru cracking-ul hash-urilor. Acest proces este crucial pentru definirea clară a obiectivelor și funcționalităților necesare pentru a satisface cerințele educaționale și de cercetare, precum și pentru a răspunde nevoilor utilizatorilor finali. Cerințele bine definite contribuie la crearea unei soluții eficiente, care facilitează înțelegerea proceselor criptografice și susține luarea deciziilor bazate pe date.

În primul rând, colectarea cerințelor este importantă deoarece sistemul trebuie să deservească mai multe categorii de utilizatori, fiecare cu nevoi și obiective diferite. Aplicația va fi utilizată de studenți, cercetători și profesori, iar fiecare dintre aceste categorii necesită un nivel specific de acces și funcționalități adaptate. Echipa de dezvoltare va putea crea un sistem ușor de utilizat, funcțional și eficient din punctul de vedere al accesului și utilizării datelor, dacă înțelege corect cerințele fiecărei categorii.

Implementarea și optimizarea tehnicilor de bruteforce reprezintă o altă componentă crucială. Funcționalitățile esențiale, cum ar fi selectarea tipului de hash, configurarea parametrilor algoritmului și afișarea progresului în timp real, sunt clarificate prin procesul de colectare a cerințelor. Echipa de dezvoltare va crea o soluție care nu doar îndeplinește aceste cerințe, ci și le face accesibile și ușor de utilizat, îmbunătățind astfel experiența utilizatorului.

O colectare adecvată a cerințelor garantează gestionarea eficientă și securizată a tuturor fluxurilor de informații generate de aplicație. Acest lucru este esențial pentru a oferi utilizatorilor o platformă fiabilă și pentru a asigura integritatea datelor. Totodată, stabilirea unor cerințe clare pentru performanță și scalabilitate contribuie la crearea unui sistem robust, capabil să gestioneze cantități mari de date și să răspundă rapid la cerințele utilizatorilor.

Definirea cerințelor de performanță este, de asemenea, o componentă importantă a acestui proces. În timpul sesiunilor educaționale sau de cercetare intensivă, aplicația trebuie să poată procesa un număr mare de combinații pe secundă și să ofere rezultate precise și în timp util. Colectarea cerințelor ajută la stabilirea unor standarde clare de performanță, astfel încât aplicația să funcționeze optim în orice scenariu.

Procesul de colectare a cerințelor garantează că sistemul este capabil să îndeplinească nevoile utilizatorilor, să optimizeze procesele educaționale și să susțină cercetarea în domeniul securității cibernetice. Prin urmare, acest pas devine esențial pentru dezvoltarea unui software eficient, care să ofere valoare adăugată prin demonstrarea practică a tehnicilor de bruteforce într-un mediu controlat.

## 2.1. Cerințe funcționale și nefuncționale

Obiectivele propuse trebuie să fie clar definite pentru a dezvolta o aplicație software dedicată spargerii hash-urilor prin tehnici de bruteforce și integrării unui GUI intuitiv. Cerințele funcționale stabilesc funcțiile necesare pentru colectarea, configurarea și monitorizarea atacurilor bruteforce, în timp ce cerințele nefuncționale asigură performanța, securitatea și scalabilitatea sistemului.

#### ****Cerințe funcționale****

1. Aplicația va permite încărcarea hash-urilor de tip **MD5, SHA-1** și **SHA-256**.
2. Utilizatorii vor putea configura parametrii algoritmului, inclusiv:
   * Lungimea parolelor testate.
   * Setul de caractere utilizat (litere, cifre, simboluri).
   * Numărul de fire de execuție (threads).
3. Sistemul va afișa în timp real progresul atacului, incluzând:
   * Combinația curentă testată.
   * Procentajul completat.
   * Timp estimativ rămas.
4. O interfață grafică interactivă va permite utilizatorilor să controleze și să monitorizeze procesul.
5. Sistemul va genera rapoarte detaliate cu rezultatele testelor și timpul de execuție.
6. Aplicația va trimite notificări pentru succesul sau eșecul atacurilor.

#### ****Cerințe nefuncționale****

1. Sistemul va utiliza mecanisme de autentificare pentru a limita accesul la funcționalitățile critice.
2. Performanța va fi optimizată pentru a procesa minim **10.000 combinații/secundă** pe un sistem cu specificațiile recomandate.
3. Arhitectura modulară va permite adăugarea de noi algoritmi de hash fără modificări majore.
4. Aplicația va fi compatibilă cu platformele **Windows** și **Linux**.
5. Interfața utilizator va fi simplă și intuitivă, pentru a reduce timpul necesar învățării și utilizării.
6. Sistemul va include mecanisme de backup pentru a preveni pierderea rezultatelor testelor în caz de întreruperi neprevăzute.

Prin definirea și implementarea acestor cerințe, proiectul va oferi o soluție software completă și eficientă, capabilă să răspundă atât nevoilor educaționale, cât și cerințelor de cercetare.

# Analiza cerintelor

## 3.1 Analiza Cerințelor Funcționale

Cerințele funcționale descriu acțiunile specifice pe care sistemul le va realiza pentru a satisface nevoile utilizatorilor.

**1. Încărcarea și gestionarea hash-urilor**

* Aplicația trebuie să permită utilizatorilor să introducă hash-uri fie manual, fie prin încărcarea unui fișier (e.g., .txt, .csv).
* Funcționalitatea de **identificare automată** a tipului de hash va analiza structura hash-ului (e.g., lungime, caracteristici specifice) și va determina tipul acestuia (MD5, SHA-1, SHA-256 etc.).
* Utilizatorii vor putea introduce hash-uri multiple pentru cracking simultan, fiecare având configurări dedicate.

**2. Configurarea algoritmilor de bruteforce**

* Utilizatorii vor putea personaliza parametrii algoritmului:
  + Lungimea minimă și maximă a parolelor (de exemplu, 4-12 caractere).
  + Seturi de caractere utilizate:
    - Litere mari și mici (e.g., A-Z, a-z).
    - Cifre (0-9).
    - Simboluri speciale (e.g., @, #, $, %).
  + Numărul de fire de execuție (threads) pentru procesare paralelă.
* Opțiunea de utilizare a **wordlist-urilor**:
  + Aplicația va include o bază de date predefinită cu cele mai utilizate wordlist-uri.
  + Utilizatorii vor putea încărca propriile wordlist-uri personalizate.

**3. Procesul de bruteforce**

* Implementarea de tehnici de bruteforce:
  + **Forță brută simplă:** Testarea tuturor combinațiilor posibile conform parametrilor configurați.
  + **Forță brută cu optimizare:** Utilizarea unor tehnici de preprocesare pentru reducerea spațiului de căutare.
  + Integrarea mai multor algoritmi de hashing (MD5, SHA-1, SHA-256).
* Posibilitatea de **pauză și reluare** a procesului de bruteforce fără pierderea progresului curent.

**4. Monitorizare și feedback în timp real**

* Progresul bruteforce va fi afișat în interfață, incluzând:
  + Combinația curentă de caractere testată.
  + Procentajul progresului total.
  + Timp estimativ rămas pentru finalizarea procesului.
* Afișarea utilizării resurselor sistemului (CPU, RAM) pentru optimizarea performanței.

**5. Raportare și rezultate**

* După finalizarea unui atac de bruteforce, aplicația va genera un **raport detaliat**, incluzând:
  + Hash-ul introdus.
  + Parola identificată (dacă există).
  + Parametrii utilizați (lungime, set de caractere, threads).
  + Timpul total de execuție.
* Posibilitatea de a exporta rapoartele în formate comune (e.g., PDF, CSV).

**6. Notificări și alerte**

* Alerte automate pentru:
  + Succesul procesului de cracking (hash decriptat).
  + Eșecul procesului (toate combinațiile testate fără succes).
* Notificări vizuale și sonore în interfață.

**7. Interfață grafică (GUI)**

* Interfața trebuie să includă:
  + Meniuri intuitive pentru configurarea parametrilor.
  + Vizualizări grafice pentru progresul atacurilor.
  + Tab-uri dedicate pentru rezultate și rapoarte.

## 3.2 Analiza Cerințelor Nefuncționale

Cerințele nefuncționale definesc calitățile și standardele care influențează performanța, fiabilitatea și utilizabilitatea aplicației.

**1. Performanță**

* Aplicația trebuie să suporte procesarea simultană a minim **10.000 de combinații/secundă** pe un sistem cu specificațiile recomandate:
  + Procesor: Intel Core i5 sau echivalent.
  + Memorie RAM: minim 4 GB.
* Sistemul trebuie să fie capabil să gestioneze până la **10 hash-uri simultan** fără întreruperi semnificative.

**2. Scalabilitate**

* Sistemul trebuie să permită adăugarea de noi algoritmi de hashing fără modificări majore ale arhitecturii.
* Aplicația va suporta creșterea dimensiunii bazei de date a wordlist-urilor (peste 1 milion de intrări).

**3. Compatibilitate**

* Aplicația va fi compatibilă cu:
  + Sistemele de operare: Windows 10 și mai recent, Linux.
  + Dispozitivele desktop și laptop.
* Utilizarea bibliotecilor Python (hashlib, tkinter etc.) pentru portabilitate și ușurința întreținerii.

**4. Securitate**

* Implementarea mecanismelor de autentificare pentru a limita accesul la funcționalitățile critice (e.g., acces pe baza unui cont).
* Datele utilizatorilor și informațiile procesate vor fi protejate prin criptare.
* Backup automat pentru datele salvate, pentru a preveni pierderea informațiilor.

**5. Fiabilitate**

* Sistemul trebuie să aibă o disponibilitate de minim **99%**.
* Funcționalitate de backup și recuperare automată în cazul unei întreruperi.

**6. Ușurința utilizării**

* Interfața grafică trebuie să fie:
  + Simplă și intuitivă, ușor de utilizat chiar și pentru utilizatorii nespecializați.
  + Dotată cu ghiduri și sugestii pentru configurarea corectă a parametrilor.
* Timp de învățare estimat: maxim 10 minute pentru utilizatori noi.

**7. Disponibilitate**

* Sistemul trebuie să fie disponibil 24/7 și să permită sesiuni prelungite de procesare.

**8. Adaptabilitate**

* Sistemul trebuie să permită integrarea cu soluții externe, precum:
  + Sisteme de stocare a datelor.
  + Alte instrumente de analiză criptografică.

**9. Documentație**

* Aplicația va include documentație detaliată pentru:
  + Utilizatori (manual de utilizare).
  + Dezvoltatori (descrierea arhitecturii și codului sursă).
* Ghidurile vor fi disponibile în format PDF și integrate în aplicație (secțiune de Help).

# Documentarea cerințelor colectate

## 4.1 Descriere generală

Proiectul urmărește dezvoltarea unei aplicații software care să implementeze tehnici brute-force pentru spargerea hash-urilor și să ofere o interfață grafică intuitivă (GUI) pentru utilizatorii finali. Scopul principal este analiza și evaluarea eficienței tehnicilor brute-force aplicate algoritmilor de hashing precum **MD5, SHA-1 și SHA-256**, oferind o soluție practică pentru înțelegerea vulnerabilităților de securitate.

Aplicația va permite utilizatorilor să configureze parametrii de bruteforce, să monitorizeze progresul procesului în timp real și să genereze rapoarte detaliate. Prin integrarea unei interfețe grafice prietenoase, sistemul va fi accesibil atât utilizatorilor tehnici, cât și celor mai puțin experimentați. Proiectul este structurat pe două componente principale:

* **Modulul de atac bruteforce:** Implementarea algoritmilor brute-force și configurarea parametrilor.
* **Modulul de raportare și analiză:** Vizualizarea progresului și generarea rapoartelor detaliate.

Aplicația va fi compatibilă cu platformele **Windows** și **Linux** și va utiliza tehnologii moderne precum Python și biblioteci asociate (e.g., Tkinter, hashlib) pentru performanță și scalabilitate.

## 4.2 Cerințe de interfață externă

Pentru a asigura o interacțiune eficientă între utilizatori, hardware și software, aplicația va include următoarele interfețe externe:

**4.2.1 Interfața utilizatorului**

* **Tehnologie utilizată:** Tkinter pentru GUI.
* **Funcționalități principale:**
  + Pagina principală pentru încărcarea hash-urilor și configurarea atacului brute-force.
  + Pagini dedicate pentru:
    - Vizualizarea progresului atacului.
    - Rapoarte detaliate și export în format PDF/CSV.
  + Secțiune de setări pentru configurarea parametrilor de atac (e.g., lungimea parolei, setul de caractere, threads).

**4.2.2 Hardware**

* **Cerințe minime:**
  + Procesor: Intel Core i5 sau echivalent.
  + RAM: minim 4 GB.
  + Spațiu pe disc: minim 1 GB pentru stocarea rapoartelor și a wordlist-urilor.

**4.2.3 Software**

* Sistem de operare: Windows 10 și mai recent, Linux.
* Python 3.9+ instalat cu suport pentru biblioteci necesare (hashlib, Tkinter, etc.).

**4.2.4 Protocole de comunicare**

* Local, fără interacțiune cu servere externe.
* Datele utilizatorilor și rezultatele brute-force vor fi stocate local pentru confidențialitate.

## 4.3 Caracteristica sistemului de atac brute-force

### 4.3.1 Descriere și prioritate

Sistemul va implementa tehnici de brute-force pentru a analiza hash-uri și a identifica parolele asociate acestora. Prioritatea este asigurarea unui proces eficient și configurabil, care să permită utilizatorilor să personalizeze parametrii atacurilor brute-force.

Sistemul va fi capabil să proceseze hash-uri pentru algoritmi precum **MD5, SHA-1 și SHA-256** și să genereze rapoarte detaliate ale rezultatelor obținute.

**4.3 Tabel - Acțiune/Rezultat sistemului de gestionare cracking**

|  |  |
| --- | --- |
| **Acțiune** | **Rezultat** |
| Utilizatorul încarcă hash-ul sau lista de hash-uri. | Sistemul identifică automat tipul hash-ului și îl afișează utilizatorului. |
| Utilizatorul configurează parametrii atacului brute-force. | Sistemul validează setările și inițiază procesul conform specificațiilor. |
| Utilizatorul monitorizează progresul. | |  | | --- | |  |  |  | | --- | | Sistemul afișează combinația curentă testată,  procentajul și timpul estimativ. | |
| Procesul brute-force se finalizează cu succes. | Sistemul afișează parola descoperită și salvează raportul generat. |
| Utilizatorul exportă raportul în format PDF. | Sistemul generează un fișier PDF cu toate detaliile procesului. |

### 4.3.3 Cerințe funcționale

### Sistemul trebuie să suporte algoritmi de hashing precum MD5, SHA-1 și SHA-256.

### Parametrii configurabili de către utilizator includ:

### Lungimea minimă și maximă a parolelor.

### Tipurile de caractere (litere, cifre, simboluri).

### Numărul de threads pentru procesare paralelă.

### Opțiunea de a utiliza wordlist-uri predefinite sau personalizate.

### Afișarea progresului în timp real.

### Generarea de rapoarte detaliate cu export în PDF/CSV.

### 4.3.4 Cerințe de performanță

 Sistemul trebuie să proceseze minim **10.000 combinații/secundă** pe un sistem standard.

 Aplicația trebuie să fie capabilă să gestioneze până la **10 hash-uri simultan** fără a afecta performanța.

### 4.3.5 Cerințe de securitate

 Datele și rapoartele vor fi stocate local, fără a fi trimise către servere externe.

 Accesul la funcționalități avansate va fi protejat prin autentificare (opțional).

## 4.4 Caracteristica sistemului de management al datelor colectate

### 4.4.1 Descriere și prioritate

Modulul de raportare și analiză reprezintă o componentă centrală a aplicației, având ca scop principal colectarea, structurarea și afișarea datelor rezultate din procesul brute-force. Acest modul este responsabil pentru furnizarea unui feedback detaliat utilizatorilor în timpul și după finalizarea procesului de spargere a hash-urilor.

Prioritățile modulului includ:

1. Generarea rapoartelor precise și structurate.
2. Exportul rapoartelor în formate utilizabile (e.g., PDF, CSV).
3. Oferirea unei vizualizări intuitive și detaliate a progresului și rezultatelor procesului brute-force.
4. Permisivitatea utilizatorilor de a accesa istoricul rapoartelor anterioare.

### 4.4.2 Acțiune/Rezultat

Tabelul **4.4.4 Acțiune/Rezultat** oferă o perspectivă clară asupra principalelor interacțiuni dintre utilizator și modulul de raportare și analiză, detaliind rezultatele obținute pentru fiecare acțiune întreprinsă în aplicație. Acest tabel ilustrează cum răspunde sistemul la comenzile utilizatorului și modul în care procesele sunt automatizate pentru a îmbunătăți experiența de utilizare.

**Tabel 4.4 - Acțiune/Rezultat sistemul de management al datelor colectate**

|  |  |
| --- | --- |
| **Acțiune** | **Rezultat** |
| Procesul brute-force este finalizat. | Sistemul generează automat un raport detaliat cu informațiile despre procesul finalizat. |
| Utilizatorul accesează istoricul rapoartelor. | Sistemul afișează o listă a proceselor anterioare, organizată cronologic. |
| Utilizatorul selectează un raport specific. | Sistemul afișează conținutul raportului selectat, incluzând detalii despre hash, algoritm și rezultate. |
| Utilizatorul exportă raportul în format PDF. | Sistemul salvează raportul într-un fișier PDF structurat, care include toate informațiile relevante. |
| Utilizatorul vizualizează progresul în timp real. | Sistemul afișează grafic (e.g., bară de progres) și textual detalii despre statusul procesului. |

### 4.4.3 Cerințe funcționale

1. **Generarea rapoartelor detaliate:**
   * Sistemul trebuie să creeze un raport după fiecare proces brute-force, care să includă:
     + Hash-ul introdus de utilizator.
     + Parola identificată (dacă există).
     + Algoritmul de hashing utilizat (e.g., MD5, SHA-1).
     + Timpul total de execuție.
     + Numărul de combinații testate.
     + Parametrii configurați (lungime parolă, set de caractere, threads).
2. **Exportul rapoartelor:**
   * Rapoartele trebuie să fie disponibile pentru export în cel puțin două formate: PDF și CSV.
   * Utilizatorul poate selecta locația de salvare a fișierului generat.
3. **Accesarea istoricului rapoartelor:**
   * Sistemul va organiza rapoartele într-o listă cronologică, afișând detalii esențiale (e.g., data procesului, hash, algoritm).
   * Utilizatorii vor putea căuta și filtra rapoartele în funcție de criterii precum algoritm sau dată.
4. **Vizualizarea progresului în timp real:**
   * Sistemul va afișa:
     + Bară de progres care indică procentul completat.
     + Timp estimativ rămas.
     + Combinația curentă testată.

### 4.4.4 Cerințe de performanță

Cerințele de performanță sunt fundamentale pentru asigurarea eficienței și vitezei proceselor brute-force aplicate cracking-ului hash-urilor. Aceste cerințe vizează optimizarea proceselor de decriptare, reducerea timpului de căutare și maximizarea utilizării resurselor hardware disponibile.

**Timpul de procesare pentru cracking-ul hash-urilor**

1. **Viteza minimă acceptabilă:**
   * Sistemul trebuie să proceseze cel puțin **50.000 combinații/secundă** pe algoritmi standard precum MD5 și SHA-1, utilizând un sistem cu specificațiile minime:
     + Procesor: Intel Core i5 @ 2.5 GHz sau echivalent.
     + Memorie RAM: 4 GB.
2. **Optimizare pe algoritmi diferiți:**
   * Algoritmii precum SHA-256, mai computațional intensivi, trebuie să fie procesați la o viteză de minim **30.000 combinații/secundă**, utilizând aceleași resurse hardware.
3. **Reducerea timpului necesar cracking-ului:**
   * Utilizarea tehnicilor de optimizare, cum ar fi reducerea spațiului de căutare și aplicarea unor strategii de preprocesare, trebuie să reducă timpul de procesare cu cel puțin **25%** față de metoda brute-force clasică.

**Utilizarea resurselor hardware**

1. **Procesare paralelă:**
   * Sistemul trebuie să suporte procesarea paralelă utilizând toate nucleele disponibile ale procesorului. De exemplu, un procesor quad-core trebuie să proceseze cel puțin **200.000 combinații/secundă**, distribuite proporțional pe fiecare nucleu.
2. **Utilizarea GPU-ului (opțional):**
   * Dacă hardware-ul include un GPU compatibil, sistemul trebuie să poată utiliza resursele acestuia pentru accelerarea procesării, crescând viteza cracking-ului cu **50%** comparativ cu procesarea exclusiv pe CPU.
3. **Managementul memoriei:**
   * Sistemul trebuie să folosească eficient memoria RAM, cu o limită de utilizare configurabilă de utilizator (e.g., maxim 75% din RAM disponibilă), pentru a evita impactul negativ asupra altor aplicații care rulează în paralel.

**Performanța algoritmilor brute-force**

1. **Wordlist-uri mari:**
   * Sistemul trebuie să fie capabil să proceseze wordlist-uri mari (e.g., peste **10 milioane de intrări**) fără o degradare semnificativă a performanței. Timpul de preluare al unui hash pentru un astfel de wordlist nu trebuie să depășească **5 minute**.
2. **Tehnici de optimizare:**
   * Implementarea de tehnici precum eliminarea combinațiilor redundante, utilizarea structurilor de date optimizate (e.g., Bloom filters) pentru căutarea rapidă și adaptarea automată a seturilor de caractere în funcție de tipul hash-ului.

**Capacitatea de gestionare a proceselor multiple**

1. **Cracking simultan:**
   * Sistemul trebuie să poată procesa simultan cel puțin **5 hash-uri diferite**, fiecare cu parametrii de configurare individuali, fără degradare semnificativă a performanței.
2. **Scenarii complexe:**
   * Sistemul trebuie să poată alterna între atacuri brute-force simple și atacuri optimizate pentru hash-uri complexe (e.g., PBKDF2, bcrypt) fără întârzieri notabile în inițierea proceselor.

**Viteza de actualizare în timp real**

1. **Intervale de actualizare:**
   * Progresul trebuie actualizat vizual în interfața utilizatorului la fiecare **250 ms**, incluzând:
     + Combinația curentă testată.
     + Numărul de combinații procesate.
     + Procentajul progresului și timpul estimativ rămas.
2. **Răspuns instant la comenzi:**
   * Comenzile utilizatorului (e.g., pauză, reluare, configurare) trebuie să fie aplicate instantaneu, cu un timp de răspuns sub **100 ms**.

**Testarea cerințelor de performanță**

1. **Scenarii de performanță:**
   * Testarea vitezei de cracking pe seturi de hash-uri pentru algoritmi diferiți (MD5, SHA-1, SHA-256).
   * Simularea utilizării intensive (e.g., 10 hash-uri procesate simultan cu un wordlist de 1 milion de intrări).
2. **Benchmark-uri hardware:**
   * Măsurarea performanței pe diferite configurații hardware (CPU, GPU, RAM).
3. **Testarea stabilității:**
   * Evaluarea funcționării sistemului pe perioade lungi de timp (e.g., 12 ore de procesare continuă).

### 4.4.5 Cerințe de securitate

Cerințele de securitate sunt esențiale pentru protejarea datelor utilizatorilor și asigurarea integrității sistemului. În contextul proiectului, aceste cerințe sunt orientate spre securitatea datelor procesate (hash-uri, parole) și prevenirea accesului neautorizat.

**Confidențialitatea datelor**

1. **Stocare locală:**
   * Toate datele utilizatorilor, inclusiv hash-urile și rezultatele proceselor brute-force, trebuie să fie stocate local, fără a fi partajate cu servere externe.
2. **Protecție împotriva accesului neautorizat:**
   * Utilizatorii trebuie să fie autentificați înainte de a accesa aplicația. Autentificarea poate include parole complexe sau autentificare cu doi factori (2FA).

**Integritatea datelor**

1. **Validarea inputurilor:**
   * Sistemul trebuie să valideze hash-urile introduse de utilizatori pentru a preveni introducerea de date incorecte sau potențial malițioase.
2. **Detectarea erorilor:**
   * Aplicația trebuie să includă mecanisme de verificare pentru a detecta și remedia eventualele coruperi ale datelor stocate local

**Securitatea comunicațiilor**

1. **Conexiuni sigure:**
   * Dacă sistemul oferă suport pentru procesare distribuită pe mai multe dispozitive, toate comunicațiile trebuie să fie criptate folosind protocoale sigure (e.g., TLS 1.3).
2. **Criptarea datelor sensibile:**
   * Rezultatele brute-force și hash-urile procesate trebuie criptate înainte de a fi salvate local.

**Prevenirea atacurilor brute-force împotriva aplicației**

1. **Limitarea numărului de încercări:**
   * Sistemul trebuie să implementeze limitări asupra numărului de încercări de autentificare, pentru a preveni atacurile brute-force asupra conturilor utilizatorilor.
2. **Jurnalizare:**
   * Toate activitățile sensibile trebuie înregistrate într-un jurnal de audit, pentru a permite monitorizarea activităților neobișnuite.

## 4.5 Documentația proiectului

Documentația tehnică a proiectului **"Analiză și Implementare a Tehnicilor de Bruteforce pentru Cracking-ul Hash-urilor și integrarea unui GUI"** include informații detaliate despre arhitectura sistemului, modulele principale și specificațiile necesare pentru implementare. Proiectul este organizat în două componente majore: **Modulul de procesare bruteforce** și **Modulul de analiză și raportare a progresului**

## ****4.5.1 Modulul de procesare bruteforce****

1. **Descrierea arhitecturii sistemului:**  
   Modulul este dezvoltat în Python și utilizează biblioteci precum hashlib pentru procesarea hash-urilor și multiprocessing pentru implementarea procesării paralele. Designul modular permite adăugarea ușoară de algoritmi noi (e.g., bcrypt, PBKDF2).
2. **Informații detaliate despre modulele individuale:**
   * **Modulul de inițializare a procesului:** Responsabil pentru încărcarea hash-urilor, selectarea algoritmului și configurarea parametrilor (e.g., lungimea parolei, seturi de caractere).
   * **Modulul de procesare paralelă:** Utilizează nucleele disponibile ale procesorului pentru testarea combinațiilor brute-force, crescând astfel eficiența procesului.
   * **Modulul de optimizare:** Include tehnici pentru reducerea spațiului de căutare, precum utilizarea structurilor de date eficiente și eliminarea combinațiilor redundante.
3. **Specificările hardware și software necesare:**
   * **Software:** Python 3.9+, pachete suplimentare precum hashlib, Tkinter, și multiprocessing.
   * **Hardware:**
     + Procesor: Minim Intel Core i5 (quad-core) sau echivalent.
     + RAM: Minim 4 GB (optimizat pentru 8+ GB pentru procese complexe).
4. **Proceduri de instalare, configurare și întreținere:**
   * **Instalare:** Utilizatorii vor descărca proiectul dintr-un repository GitHub și vor instala dependențele utilizând pip install -r requirements.txt.
   * **Configurare:** După instalare, utilizatorii pot modifica fișierul de configurare pentru a specifica setările implicite (e.g., seturi de caractere, algoritmi utilizați).
   * **Întreținere:** Sistemul trebuie actualizat periodic pentru a include suport pentru noi algoritmi de hashing și pentru remedierea problemelor identificate.

## 4.5.2 Modulul de analiză și raportare a progresului

1. **Descrierea arhitecturii sistemului:**  
   Acest modul este construit utilizând biblioteca Tkinter pentru GUI, care permite utilizatorilor să interacționeze ușor cu aplicația. Progresul procesului brute-force este actualizat în timp real și afișat grafic.
2. **Informații detaliate despre modulele individuale:**
   * **Modulul de afișare a progresului:** Prezintă procentajul procesului completat, combinația curentă testată, timpul estimat rămas și rata de procesare (combinații/secundă).
   * **Modulul de raportare:** Permite utilizatorilor să vizualizeze rezultatele procesului și să exporte informațiile în formate PDF sau CSV.
   * **Modulul de notificare:** Trimite notificări utilizatorului la finalizarea procesului sau în cazul apariției unor erori.
3. **Specificările hardware și software necesare:**
   * **Software:** Python, biblioteci suplimentare pentru GUI (Tkinter) și pentru export de rapoarte (reportlab, pandas).
   * **Hardware:** Aceleași cerințe ca pentru modulul de procesare bruteforce.
4. **Proceduri de instalare, configurare și întreținere:**
   * **Instalare:** Configurația GUI este inclusă în pachetul principal; utilizatorii vor rula aplicația direct din mediul Python.
   * **Configurare:** Personalizarea interfeței (e.g., culori, teme) se poate face printr-un fișier .config.
   * **Întreținere:** Actualizarea periodică a GUI-ului pentru a include noi funcționalități, precum suport pentru algoritmi adăugați ulterior.

## 4.6 Documentația utilizatorului

Documentația utilizatorului este destinată să ofere ghiduri clare și concise pentru utilizarea aplicației. Scopul principal este de a ajuta utilizatorii să configureze și să monitorizeze procesele brute-force, precum și să interpreteze rezultatele obținute.

## 4.6.1 Ghid pentru procesarea hash-urilor

1. **Încărcarea unui hash și inițierea procesului:**
   * Deschideți aplicația și selectați opțiunea „Adăugare Hash”.
   * Introduceți hash-ul manual sau încărcați un fișier conținând mai multe hash-uri.
   * Selectați algoritmul de hashing din lista disponibilă.
2. **Configurarea parametrilor brute-force:**
   * Selectați lungimea minimă și maximă a parolelor.
   * Alegeți setul de caractere (e.g., litere mari/mici, cifre, simboluri).
   * Configurați numărul de threads pentru procesare paralelă.
3. **Monitorizarea progresului:**
   * Progresul procesului va fi afișat în timp real, incluzând detalii precum:
     + Procentajul completat.
     + Combinația curentă testată.
     + Timpul estimat rămas.
4. **Finalizarea procesului:**
   * După identificarea parolei, sistemul va genera automat un raport detaliat.

## 4.6.2 Ghid pentru vizualizarea și exportarea rezultatelor

1. **Vizualizarea raportului:**
   * Accesați secțiunea „Rezultate” din interfață.
   * Selectați hash-ul procesat pentru a vedea detaliile (parola descoperită, timp total, algoritm utilizat).
2. **Exportarea raportului:**
   * Selectați formatul dorit (PDF/CSV).
   * Specificați locația de salvare pe dispozitiv.

## 4.6.3 Scenarii de utilizare comune

1. **Procesarea unui singur hash:**
   * Utilizatorul introduce un hash MD5, configurează parametrii brute-force, inițiază procesul și vizualizează rezultatul în GUI.
2. **Utilizarea unui wordlist personalizat:**
   * Utilizatorul încarcă un wordlist extern și inițiază procesul brute-force, vizualizând progresul în timp real.
3. **Gestionarea mai multor hash-uri simultan:**
   * Utilizatorul încarcă un fișier cu mai multe hash-uri și inițiază procesarea simultană, monitorizând progresul fiecărui proces individual.

## 4.6.4 Soluții pentru probleme frecvente

1. **Procesul se oprește brusc:**
   * Verificați logurile aplicației pentru mesaje de eroare.
   * Asigurați-vă că hardware-ul suportă configurările curente.
2. **Rezultate incorecte:**
   * Verificați dacă hash-ul introdus este compatibil cu algoritmul selectat.
   * Reconfigurați parametrii și reluați procesul.

# Concluzie

Proiectul **„Analiză și Implementare a Tehnicilor de Bruteforce pentru Cracking-ul Hash-urilor și integrarea unui GUI”** reprezintă un efort bine structurat și detaliat pentru abordarea unei probleme esențiale din domeniul securității cibernetice: evaluarea eficienței tehnicilor brute-force și identificarea vulnerabilităților de securitate asociate.

Documentația detaliată oferă o perspectivă completă asupra arhitecturii, funcționalităților și cerințelor proiectului, asigurând o implementare eficientă și robustă. Principalele realizări și beneficii ale acestui proiect sunt:

1. **Comprehensivitatea tehnică:**
   * Proiectul a integrat un modul de procesare bruteforce capabil să suporte algoritmi de hashing populari (MD5, SHA-1, SHA-256) și să optimizeze spațiul de căutare prin tehnici avansate.
   * Interfața grafică intuitivă (GUI) permite utilizatorilor, inclusiv celor cu experiență tehnică limitată, să utilizeze aplicația în mod eficient.
2. **Flexibilitatea și scalabilitatea:**
   * Sistemul este capabil să proceseze mai multe hash-uri simultan, gestionând cu succes procesele complexe și seturile mari de date.
   * Scalabilitatea arhitecturii permite integrarea ulterioară a unor noi algoritmi de hashing sau funcționalități adiționale.
3. **Relevanța practică:**
   * Proiectul oferă utilizatorilor și profesioniștilor un instrument util pentru identificarea vulnerabilităților legate de parole și algoritmi de hashing, contribuind astfel la îmbunătățirea securității informaționale.
4. **Performanța și eficiența:**
   * Prin utilizarea procesării paralele și optimizarea resurselor hardware, aplicația a atins un standard ridicat de performanță, reducând semnificativ timpul necesar pentru cracking-ul hash-urilor.
5. **Securitatea și confidențialitatea:**
   * Cerințele stricte de securitate asigură protejarea datelor utilizatorilor și a proceselor interne ale aplicației, menținând integritatea sistemului.

Proiectul nu doar că îndeplinește cerințele stabilite inițial, dar oferă și o bază solidă pentru extindere și adaptare. Funcționalitățile dezvoltate pot fi utilizate în scopuri educaționale, de cercetare sau profesionale, contribuind la o mai bună înțelegere a securității cibernetice.

**Recomandări viitoare:**  
Pe baza concluziilor acestui raport, sunt sugerate următoarele direcții de dezvoltare:

1. Integrarea de tehnici alternative la brute-force, cum ar fi atacurile bazate pe dicționare hibride sau rainbow tables.
2. Optimizarea aplicației pentru a suporta distribuirea proceselor pe rețele de calculatoare (distributed computing).
3. Crearea unui modul avansat de raportare care să includă analize statistice detaliate ale proceselor brute-force.

Această lucrare demonstrează valoarea tehnicilor brute-force în analiza securității cibernetice și oferă un instrument puternic pentru identificarea punctelor slabe și îmbunătățirea protecției datelor digitale.

# Referințe bibliografice

(n.d.). Retrieved from https://www.ssldragon.com/ro/blog/brute-force-attack/

(n.d.). Retrieved from https://www.guru99.com/ro/password-cracker-tools.html