Ministerul Educației și Cercetării al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Departamentul Informatică şi Ingineria Sistemelor

RAPORT

Lucru individual

la cursul „Proiectarea sistemelor informaționale”

A efectuat: **Chihai Adrian SI-211**
A verificat: **Magdei Octavian**

Chișinău 2024

Cuprins

1. Analiza și definirea domeniului de studiu

1.1 Identificarea și declarația problemei, scopul și obiectivele sistemului

În contextul actual al creșterii numărului de amenințări cibernetice și al utilizării pe scară largă a tehnicilor criptografice pentru securizarea informațiilor, există o nevoie crescândă pentru instrumente care să permită analiza și testarea siguranței algoritmilor de hash. Tehnicile de bruteforce pentru cracking-ul hash-urilor sunt esențiale pentru a identifica vulnerabilitățile sistemelor de securitate și pentru a proteja datele sensibile.

Multe companii și utilizatori individuali se confruntă cu provocări în analiza și testarea siguranței parolelor și a datelor criptate, deoarece nu dispun de instrumentele adecvate pentru a simula și evalua atacuri de tip bruteforce. În lipsa unor astfel de instrumente, evaluările de securitate devin incomplete, iar datele rămân expuse la riscuri majore.

Probleme identificate:

- Lipsa unei platforme accesibile care să permită efectuarea de teste de securitate pentru hash-uri.
- Necesitatea unui sistem care să simplifice procesul de cracking prin bruteforce, oferind și o interfață grafică ușor de utilizat.
- Absența unui instrument centralizat care să analizeze vulnerabilitățile hash-urilor și să ofere recomandări clare pentru îmbunătățirea securității.

Scopul principal al acestui proiect este de a crea o platformă care să integreze tehnici de bruteforce pentru cracking-ul hash-urilor, facilitând analiza vulnerabilităților acestora. Sistemul va fi prevăzut cu o interfață grafică intuitivă (GUI) care să fie accesibilă atât profesioniștilor din domeniul securității cibernetice, cât și utilizatorilor mai puțin experimentați.

Obiectivele principale ale sistemului sunt:

1. **Automatizarea** procesului de cracking al hash-urilor folosind tehnici de bruteforce, cu opțiuni de personalizare a tipului de atac.
2. **Crearea unei interfețe grafice intuitive (GUI)** care să permită utilizatorilor să efectueze teste de securitate fără a fi necesare cunoștințe tehnice avansate.

3. **Implementarea de algoritmi eficienți** pentru cracking-ul hash-urilor prin bruteforce, care să permită analiza rapidă și precisă a vulnerabilităților.
4. **Oferirea de rapoarte detaliate** despre starea securității hash-urilor, identificând punctele slabe și oferind recomandări pentru îmbunătățirea securității.
5. **Integrarea unui modul de suport pentru utilizatori**, oferind resurse și sfaturi pentru prevenirea amenințărilor cibernetice, cu accent pe protecția împotriva atacurilor fizice și psihologice.

Prin realizarea acestor obiective, proiectul propus va îmbunătăți capacitatea utilizatorilor de a evalua și îmbunătăți securitatea sistemelor bazate pe hash-uri, contribuind astfel la prevenirea atacurilor cibernetice și la protejarea datelor sensibile.

1.2 Sisteme existente

Unele dintre instrumentele de securitate informatică existente oferă soluții pentru analiza și spargerea hash-urilor prin tehnici brute-force. De exemplu, **Hashcat** este un instrument puternic care utilizează unități de procesare grafică (GPU) pentru a accelera procesul de brute-force și cracking. Acesta permite utilizatorilor să testeze milioane de combinații de parole într-un timp foarte scurt, utilizând funcții avansate de hashing, cum ar fi MD5, SHA-1 și multe altele.

Un alt instrument bine cunoscut în domeniul securității este **John the Ripper**. Acesta este un program open-source care suportă diferite tipuri de hash-uri și tehnici de spargere a parolelor, oferind utilizatorilor un mod eficient de a testa securitatea sistemelor de autentificare. Pe lângă cracking-ul bazat pe parole brute-force, John the Ripper poate efectua și atacuri pe bază de dicționar. Platforma este compatibilă cu multiple sisteme de operare, inclusiv Windows, Linux și macOS, oferind astfel o versatilitate ridicată.

Un alt exemplu important este **Ophcrack**, un program gratuit care utilizează tehnici bazate pe *Rainbow Tables* pentru a sparge hash-urile LM și NTLM folosite de sistemele de operare Windows. Acest software permite utilizatorilor să recupereze parole rapid și eficient, fără a fi necesară instalarea acestuia, deoarece poate rula de pe un live CD.

Produs	Puncte Tari	Puncte Slabe
Hashcat	- Suportă multiple tipuri de hash-uri și algoritmi; - Utilizează GPU-uri pentru performanță accelerată; - Instrument extrem de configurabil și eficient.	- Necesită cunoștințe tehnice avansate pentru configurare și utilizare; - Poate consuma multe resurse hardware.
John the Ripper	- Suport pentru cracking multi-platformă; - Permite atacuri bazate pe dicționare și brute-force; - Compatibil cu mai multe tipuri de hash-uri și sisteme.	- Performanță mai lentă în comparație cu Hashcat; - Interfață în linia de comandă, ceea ce poate fi dificil pentru utilizatorii non-tehnici.
Ophcrack	- Utilizare facilă datorită *Rainbow Tables*; - Nu necesită instalare, poate rula de pe un live CD; - Suport pentru Windows LM și NTLM hashes.	- Limitat la hash-urile LM și NTLM, deci nu poate fi utilizat pentru toate tipurile de sisteme; - Mai puțin flexibil și extensibil decât alte instrumente de cracking.

1.3 Domeniul de interes pentru sistemul propus

Tehnologiile de securitate informatică sunt într-o continuă evoluție, având un rol crucial în protejarea datelor sensibile. Crearea unui sistem informațional pentru analiza și implementarea tehnicilor brute-force este esențială pentru a oferi o soluție eficientă de spargere a hash-urilor în scopuri legale, precum audituri de securitate sau recuperarea parolelor uitate.

Sistemul propus va integra un GUI ușor de utilizat, care va permite atât utilizatorilor tehnici, cât și celor non-tehnici, să își desfășoare activitățile de spargere a hash-urilor fără a fi necesare cunoștințe extinse de programare sau comandă în linie. Utilizatorii vor putea introduce hash-uri, seta parametrii atacului brute-force și monitoriza progresul în timp real. Sistemul va folosi algoritmi avansați pentru a accelera procesul de spargere și va oferi rapoarte detaliate despre starea și rezultatele procesului de cracking.

2 Analiza antreprizei

În contextul tehnologiilor informaționale moderne, securitatea datelor reprezintă o prioritate fundamentală atât pentru organizații, cât și pentru utilizatorii individuali. Printre numeroasele metode de protecție a datelor, utilizarea algoritmilor de hashing pentru stocarea și protejarea parolelor și a altor informații sensibile este o practică comună. Hash-urile joacă un rol esențial în asigurarea securității infrastructurilor informatice, dar, în același timp, nu sunt invulnerabile. Tehnici precum atacurile brute-force rămân populare pentru a sparge aceste hash-uri, deși necesită resurse considerabile și un nivel înalt de expertiză tehnică pentru a fi eficient aplicate.

Sistemul propus urmărește să abordeze aceste provocări prin integrarea tehnicilor brute-force într-un instrument accesibil și ușor de utilizat, destinat recuperării parolelor și auditului securității. Acesta va include o interfață grafică (GUI) intuitivă, care va permite utilizatorilor, fie ei tehnici sau non-tehnici, să configureze și să controleze procesul de spargere a hash-urilor. În această analiză a antreprizei, vom examina aspectele cheie legate de oportunitatea dezvoltării unui astfel de sistem, avantajele competitive și provocările cu care se va confrunta.

Contextul Pieței și Nevoile Utilizatorilor

Din cauza creșterii exponențiale a atacurilor cibernetice, multe organizații și persoane fizice sunt preocupate de protejarea datelor lor. Parolele și alte informații sensibile, atunci când sunt hash-ate corespunzător, oferă un nivel de protecție suplimentar. Totuși, algoritmii de hashing nu oferă securitate absolută, și în cazuri de audit de securitate sau recuperare a datelor, devine necesară spargerea hash-urilor pentru accesarea informațiilor.

Atacurile brute-force sunt una dintre cele mai comune metode de spargere a hash-urilor, implicând testarea sistematică a tuturor combinațiilor posibile până la găsirea celei corecte. Deși aceste atacuri sunt eficiente în anumite situații, ele sunt consumatoare de resurse și implică un proces de lungă durată, în special pentru parolele complexe. În plus, multe soluții actuale necesită cunoștințe avansate de programare sau utilizarea liniei de comandă, ceea ce reprezintă o barieră pentru utilizatorii non-tehnici.

Avantajul soluției propuse

Soluția propusă, care integrează un atac brute-force eficient și un GUI prietenos, abordează aceste provocări, oferind atât accesibilitate, cât și eficiență. Utilizatorii vor avea la dispoziție o platformă care să le permită configurarea personalizată a atacurilor, monitorizarea progresului în timp real și obținerea de rapoarte detaliate despre rezultatele procesului. Totodată, utilizarea eficientă a resurselor hardware, cum ar fi procesorul și GPU-ul, va accelera procesul de spargere a hash-urilor, reducând astfel timpul necesar finalizării operațiunii.

Sistemul va contribui la satisfacerea unei nevoi reale pe piață, mai ales în rândul organizațiilor care au nevoie de soluții pentru auditarea securității sau recuperarea datelor. Totodată, prin accesibilitatea sa, va putea fi utilizat și de către persoane fără pregătire tehnică avansată, oferindu-le o modalitate de a-și recupera parolele fără a apela la experți în domeniu.

Provocările și Limitările soluției

Una dintre provocările majore ale acestei soluții constă în dependența sa de puterea hardware. Deși sistemul va fi optimizat pentru utilizarea eficientă a resurselor disponibile, performanța generală va fi limitată de echipamentele utilizatorilor. Cei care dispun de hardware mai puțin performant vor experimenta timpi de execuție mai mari, iar în cazul hash-urilor complexe, procesul de spargere poate dura foarte mult timp.

Un alt aspect ce trebuie luat în considerare este etica și legalitatea utilizării acestei tehnologii. Deși sistemul este destinat scopurilor legitime, cum ar fi auditul de securitate și recuperarea datelor, există riscul ca acesta să fie utilizat în scopuri neautorizate sau chiar ilegale. Este

important ca utilizarea acestui sistem să fie reglementată corespunzător pentru a preveni astfel de situații.

Oportunități de dezvoltare

Pe lângă utilizarea sa actuală, sistemul poate fi dezvoltat în viitor pentru a include funcționalități suplimentare, cum ar fi integrarea unor algoritmi mai eficienți de spargere, precum atacurile bazate pe dicționare sau rainbow tables. De asemenea, soluția ar putea fi adaptată pentru a rula în medii cloud, eliminând astfel limitările hardware și oferind acces la resurse de calcul mai mari.

Această abordare ar permite organizarea de sesiuni de cracking de hash-uri mult mai rapide și eficiente, fără a afecta performanța dispozitivelor locale ale utilizatorilor.

Automatizare și eficiență	Procesele de spargere a hash-urilor și generarea de rapoarte detaliate pot fi automatizate, economisind timp.	Necesitatea personalului calificat pentru întreținerea și utilizarea eficientă a sistemului.
Accesibilitate	Interfață grafică intuitivă, ușor de utilizat atât de tehnici, cât și de non-tehnici.	Limitări impuse de performanța hardware-ului utilizatorilor (în special pentru hash-uri complexe).
Configurare personalizată	Posibilitatea de a configura parametrii atacurilor brute-force în funcție de nevoile utilizatorului.	Consum ridicat de resurse hardware, cum ar fi CPU și GPU, care poate încetini alte procese pe dispozitiv.
Monitorizare în timp real	Oferă feedback în timp real privind progresul atacurilor și estimări de timp pentru finalizare.	Necesită un management adecvat al resurselor pentru optimizarea procesului de spargere.
Optimizare hardware	Capacitatea de a utiliza eficient resursele hardware pentru a accelera procesul de spargere a hash-urilor.	Lipsa unei versiuni bazate pe cloud, ceea ce limitează scalabilitatea pentru utilizatorii cu resurse locale.

3 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.

3.1 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.

4 Analiza cerintelor

4.1 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.

4.2 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).

5 Documentarea cerințelor colectate

5.1 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.

5.2 Cerințe de interfață externă

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

5.2.1 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).

5.2.2 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.

5.2.3 Software

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

5.2.4 Protocole de comunicare

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

5.3 Caracteristica sistemului de atac brute-force

5.3.1 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.

5.3 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.

5.3.3 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.

5.3.4 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.

5.3.5 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).

5.4 Caracteristica sistemului de management al datelor colectate

5.4.1 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.

5.4.2 Acțiune/Rezultat

Tabelul **5.4 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 5.4 - 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.

5.4.3 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.

5.4.4 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).

5.4.5 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.

5.5 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**

5.5.1 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.

5.5.2 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.

5.6 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.

5.6.1 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.

5.6.2 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.

5.6.3 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.

5.6.4 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.

6 Modelarea și Analiza Proceselor pentru Cracking-ul Hash-urilor

6.1 Descrierea procesului ales

Procesul selectat pentru realizarea acestei lucrări de laborator reprezintă procesul de analiză și implementare a tehnicilor brute-force pentru cracking-ul hash-urilor. Acesta este conceput pentru a evalua vulnerabilitățile sistemelor de securitate bazate pe algoritmi de hashing și pentru a oferi soluții practice și eficiente. Este un instrument valoros pentru profesioniștii în securitate cibernetică, deoarece le oferă posibilitatea de a înțelege mai bine limitele algoritmilor de hashing și de a dezvolta soluții pentru prevenirea accesului neautorizat.

Acesta poate implica următorii pași:

- analiza tehnicilor brute-force;
- determinarea tipului de hash;
- implementarea algoritmilor brute-force pentru decriptarea hash-urilor;
- optimizarea procesului de cracking;
- testarea eficienței tehnicilor brute-force;
- generarea rapoartelor de performanță.

Unul dintre primii pași este analiza tehnicilor brute-force, unde se selectează metodele aplicabile pentru cracking-ul diferitelor tipuri de hash-uri (MD5, SHA-1, SHA-256). După aceasta, utilizatorul poate introduce hash-ul țintă, iar aplicația determină automat tipul acestuia. Acest pas este crucial pentru alegerea algoritmului potrivit și creșterea eficienței procesului.

După determinarea tipului de hash, algoritmii brute-force sunt implementați pentru decriptare, utilizând wordlist-uri optimizate. Acest proces presupune generarea și verificarea unui număr mare de combinații de parole până la găsirea uneia care corespunde hash-ului introdus. Pe parcurs, aplicația optimizează procesul prin reducerea spațiului de căutare și utilizarea unor tehnici precum preprocesarea datelor sau utilizarea hardware-ului accelerat, cum ar fi GPU-uri.

Procesul de cracking generează date esențiale, precum timpul de execuție și rata de succes. Acestea sunt incluse într-un raport detaliat care ajută utilizatorul să înțeleagă eficiența tehnicilor

aplicate. În plus, aplicația oferă o interfață grafică intuitivă (GUI), care permite utilizatorilor să acceseze funcționalitățile fără a necesita cunoștințe avansate în securitate cibernetică.

Procesul de analiză și implementare a tehnicilor brute-force pentru cracking-ul hash-urilor contribuie la o mai bună înțelegere a vulnerabilităților asociate algoritmilor de hashing. Este un instrument esențial pentru profesioniștii din domeniu și ajută la identificarea măsurilor de protecție pentru îmbunătățirea securității datelor.

6.2 Elaborarea unui model logic al datelor pentru procesul ales

- **Diagrama de context**

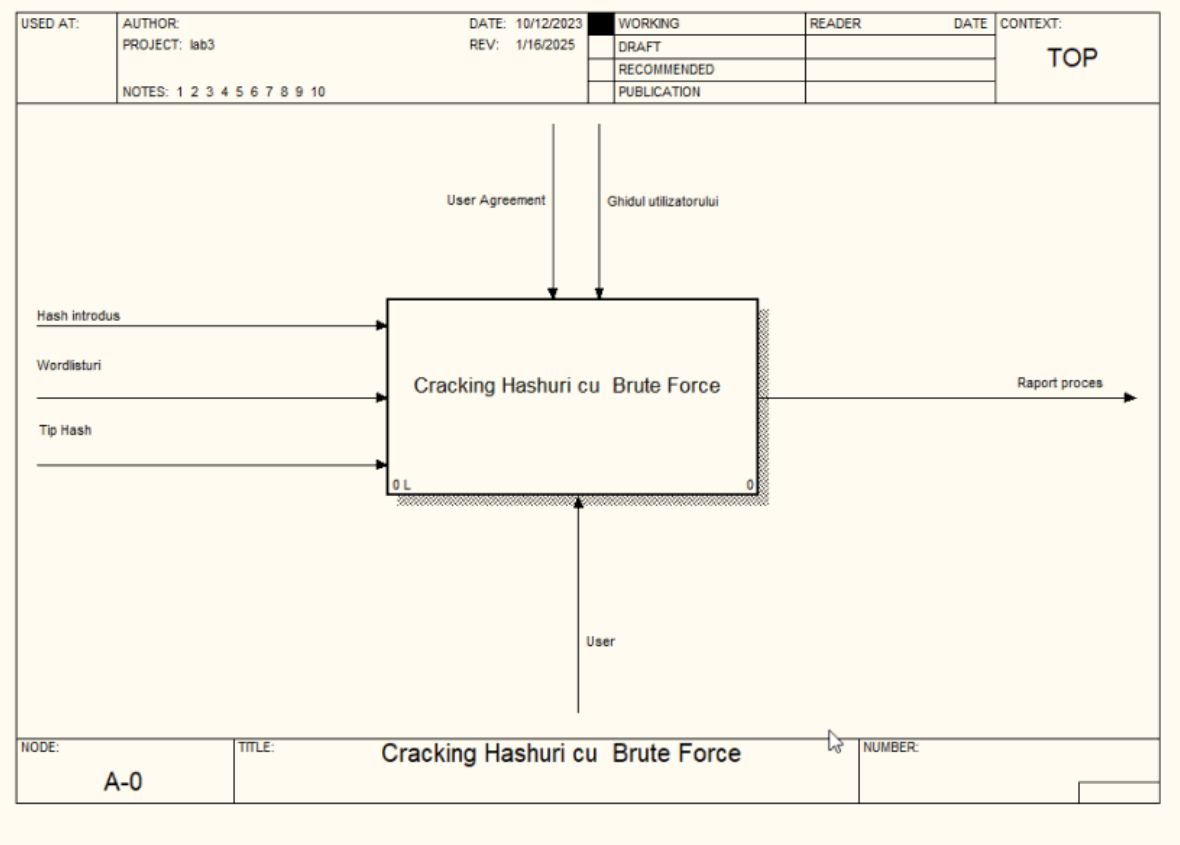

Figura 6.2.1 – Diagrama de context

Diagrama de context reprezentată în figura de mai sus oferă o viziune de ansamblu asupra procesului de cracking hash-uri cu brute force. Pentru realizarea acestui proces, utilizatorul trebuie

să introducă un hash țintă. De asemenea, este necesară furnizarea unui wordlist și, opțional, a tipului de hash, pentru a facilita aplicarea tehnicii brute-force în mod eficient.
Procesul este ghidat de instrucțiuni clare oferite prin intermediul aplicației, sub forma unui ghid al utilizatorului. Pe parcurs, utilizatorul este informat despre progresul procesului și poate vizualiza rapoarte detaliate despre performanța și eficiența tehnicilor aplicate.
Rezultatul acestui proces constă în:

- Determinarea parolei asociate hash-ului furnizat;
- Generarea unui raport detaliat cu timpul de execuție, numărul de combinații testate și rata de succes;

O mai bună înțelegere a vulnerabilităților algoritmilor de hashing folosiți.
Această diagrama subliniază elementele-cheie necesare pentru a facilita procesul și rolul utilizatorului în inițierea și monitorizarea cracking-ului hash-urilor.

- **Diagrama IDEF0**

Diagramele IDEF0, cunoscute sub denumirea de Integration Definition for Function Modeling, reprezintă un instrument esențial în domeniul ingineriei sistemelor și ingineriei software, folosit pentru a modela și analiza funcțiile și procesele din cadrul sistemelor complexe. Aceste diagrame au ca scop să ofere o reprezentare vizuală a structurii, funcțiilor, interacțiunilor și relațiilor dintre elementele care alcătuiesc un sistem sau proces specific.

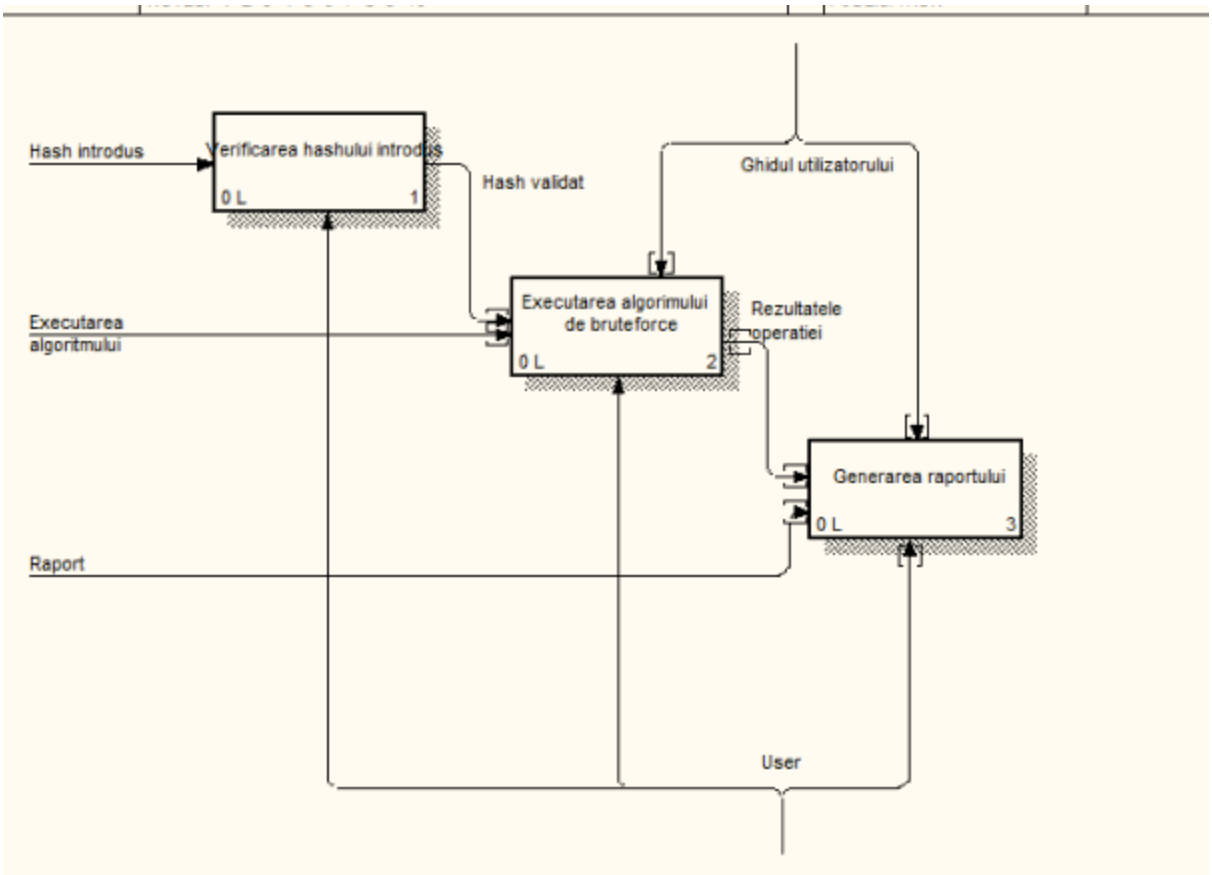

Figura 6.2.2 – Diagrama IDEF0

Diagrama IDEF0 care poate fi observată în figura 2.2 prezintă o reprezentare detaliată a procesului de **cracking al hash-urilor cu brute-force**. În primul rând, hash-ul introdus de utilizator este verificat pentru a asigura validitatea și compatibilitatea acestuia cu algoritmii disponibili. După validare, are loc procesul de executare a algoritmului de brute-force, utilizând datele furnizate, precum wordlist-uri și tipul hash-ului.

Rezultatele operației, incluzând statusul și eficiența algoritmului aplicat, sunt transmise utilizatorului. În etapa finală, un raport detaliat este generat, conținând informații despre timpul de execuție, parolele testate și succesul operației. Această diagramă evidențiază interacțiunea dintre utilizator, algoritm și aplicație, oferind o imagine clară a procesului etapizat.

- **Diagrama IDEF1**

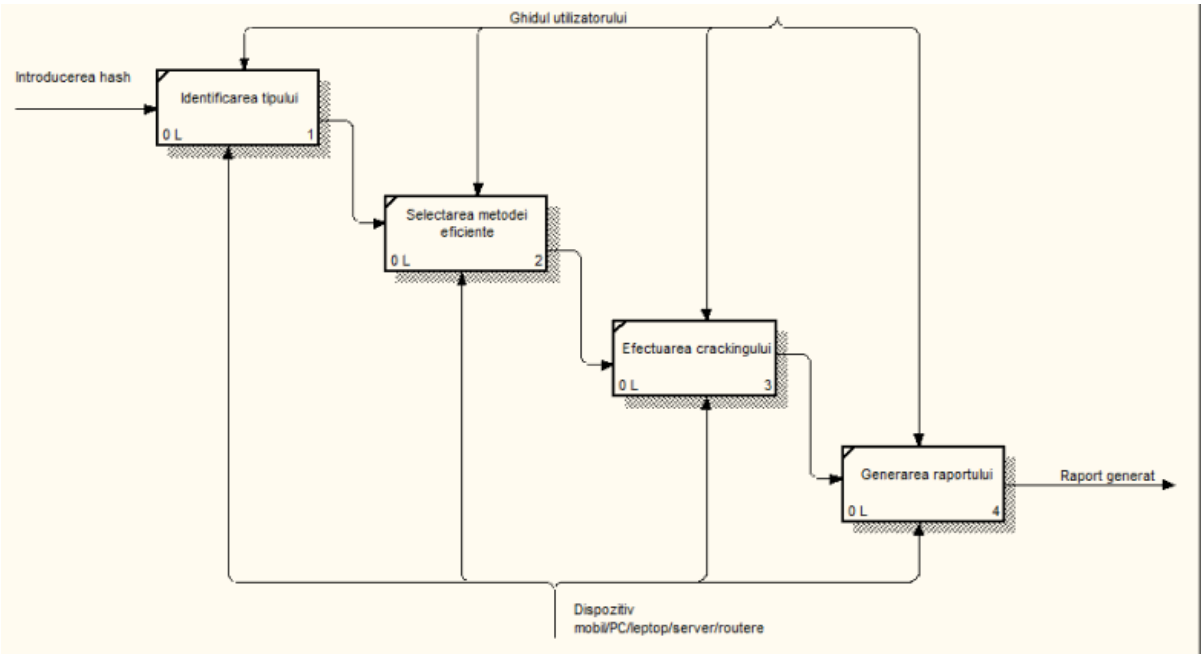

Figura 6.2.3 – Diagrama IDEF1

În figura de mai sus putem observa o reprezentare detaliată a procesului de cracking al hash-urilor. Acesta implică mai multe etape bine definite:

- Identificarea tipului de hash: Utilizatorul introduce hash-ul, iar sistemul determină automat tipul acestuia pentru a selecta algoritmul adecvat.
- Selectarea metodei eficiente: Sistemul alege metoda optimă de brute-force în funcție de tipul hash-ului și resursele disponibile.
- Efectuarea cracking-ului: Algoritmul selectat este aplicat, utilizând wordlist-uri și alte tehnici de optimizare pentru a găsi parola corespunzătoare hash-ului introdus.
- Generarea raportului: La final, un raport detaliat este generat și prezentat utilizatorului, incluzând informații despre timpul de execuție, eficiența metodei aplicate și rezultatele procesului.

Acest proces este ghidat de interfața utilizatorului (GUI) și poate fi realizat pe diferite dispozitive, precum laptopuri, servere sau alte echipamente dedicate. Raportul generat oferă utilizatorului o imagine completă asupra performanței și eficienței cracking-ului.

- **Diagrama IDEF3**

Scopul principal al diagramei IDEF3 este de a captura detaliile proceselor, inclusiv fluxurile de date, obiectivele, activitățile și relațiile dintre acestea. Prin intermediul acestei metodologii, se realizează o descriere comprehensivă a proceselor, ceea ce facilitează analiza și evaluarea acestora.

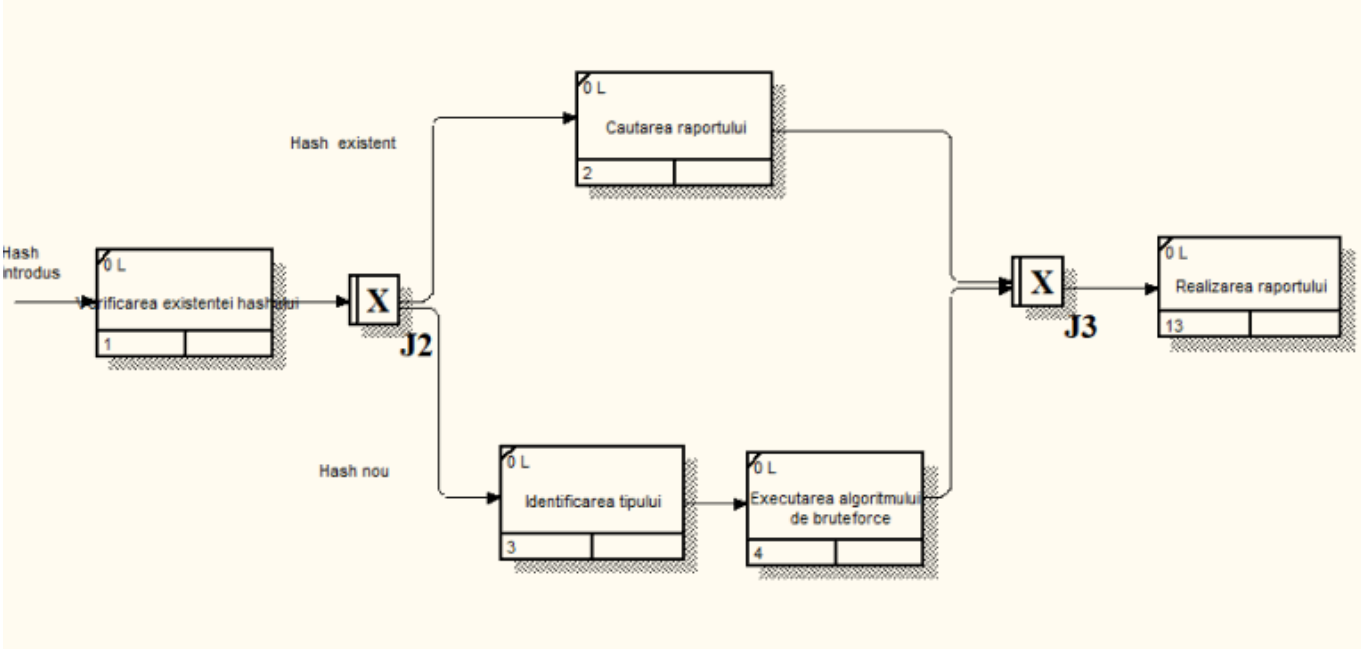

Figura 6.2.4 – Diagrama IDEF3

Diagrama din figura de mai sus este una de tip IDEF3 care reprezintă procesul de verificare a existenței unui hash. Dacă hash-ul introdus există, se caută raportul asociat. În caz contrar, are loc identificarea tipului de hash, executarea algoritmului de brute-force și generarea unui raport nou.

- **Diagrama DFD**

Diagrama Fluxului de Date (DFD) este o tehnică grafică utilizată pentru modelarea, analiza și documentarea proceselor și fluxurilor de date într-un sistem sau organizație. Această tehnică este utilizată pe scară largă în domeniul dezvoltării software, în ingineria sistemelor și în managementul proceselor pentru a oferi o înțelegere clară a modului în care datele și informațiile circulă în cadrul unui sistem sau organizație.Scopul unei diagrame DFD este de a ilustra fluxurile de date și procesele într-un sistem sau organizație.

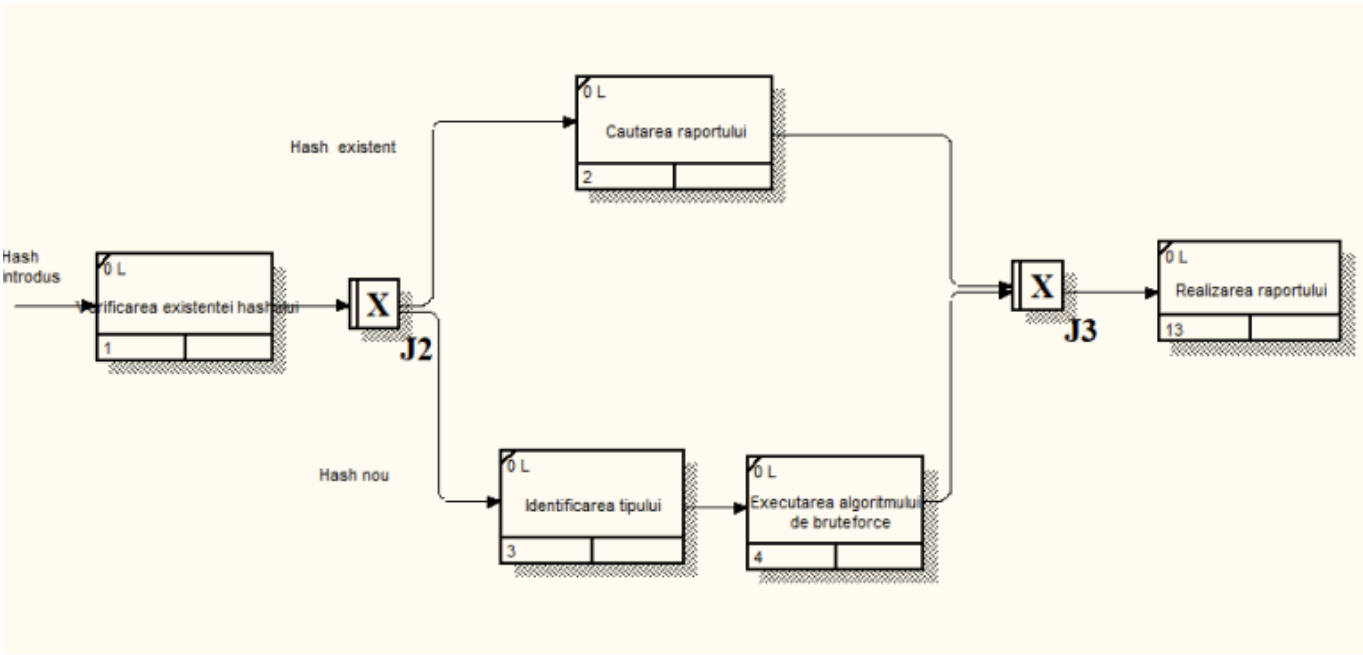

Figura 6.2.5 – Diagrama DFD

În ultima figură putem observa diagrama DFD pentru adăugarea hashului pentru cracking. În cazul dat aceasta ilustrează fluxurile de procese și date pentru procesul enunțat mai sus.

- **Diagrama IDEF1x**

Diagrama IDEF1x, cunoscută și sub denumirea de Integration Definition for Information Modeling, este o unealtă fundamentală utilizată în ingineria sistemelor, fiind special creată pentru a modela și descrie structura informațională a unui sistem. Dezvoltată inițial în cadrul programului IDEF de către Forțele Aeriene ale Statelor Unite, această diagramă se axează pe definirea și conturarea modelelor conceptuale pentru bazele de date și alte sisteme informaționale

complexe, contribuind la o înțelegere clară a relațiilor și atributelor esențiale între diferite entități.

Această formă de reprezentare vizuală are la bază diverse componente esențiale, fiecare contribuind la o ilustrare detaliată a organizării informaționale într-un sistem. Entitățile, reprezentate prin dreptunghiuri, constituie elementele fundamentale ale sistemului, descriind conceptele sau obiectele care stochează informații. Atributele, exprimate prin elipse, caracterizează entitățile, oferind detalii despre proprietățile și trăsăturile acestora. Relațiile, reprezentate prin linii ce leagă entități, indică interdependențele și conexiunile dintre elementele sistemului. Cheile, ilustrate prin simboluri specifice, oferă detalii despre modul de identificare unică a entităților în cadrul sistemului.

Scopul principal al acestei diagrame este de a facilita proiectarea și înțelegerea structurii informaționale a sistemului. Prin oferirea unei reprezentări clare a relațiilor dintre entități, atribute și chei, diagrama IDEF1x devine un instrument deosebit de eficient pentru comunicarea cerințelor și specificațiilor între membrii echipelor implicate în dezvoltarea sistemului. Aceasta contribuie la eliminarea ambiguităților și promovează o înțelegere comună a arhitecturii sistemului, sprijinind astfel eficientizarea procesului de proiectare și implementare a sistemelor informaționale complexe.

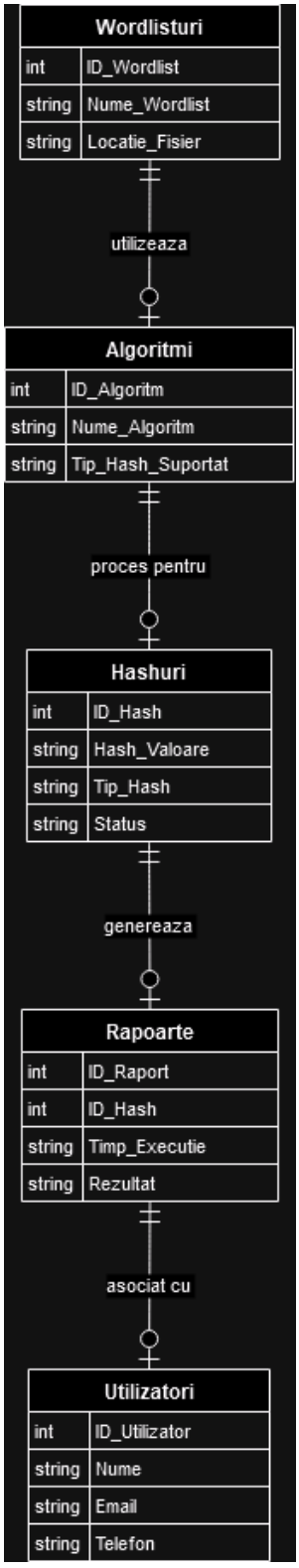

Figura 6.2.6 Diagrama IDEF1x

În cadrul diagramei IDEF1X pentru procesul de cracking al hash-urilor, sunt definite cinci entități principale: "Hashuri", "Algoritmi", "Rapoarte", "Wordlisturi" și "Utilizatori". Aceste entități sunt interconectate printr-un set de relații bine structurate conform notației IDEF1X.

Entitatea "Utilizatori" reprezintă utilizatorii aplicației, fiecare identificat printr-un ID unic. Acești utilizatori sunt legați de unul sau mai multe "Rapoarte", care reflectă rezultatele proceselor de cracking efectuate.

"Hashuri" este o entitate distinctă, caracterizată prin atribute precum valoarea hash, tipul și statusul acestuia. Fiecare hash poate avea un raport asociat, înregistrând rezultatele cracking-ului.

Entitatea "Rapoarte" acționează ca o punte între "Utilizatori" și "Hashuri", reflectând detalii despre procesul de cracking, precum timpul de execuție și rezultatul operației. Fiecare raport este conectat la un hash specific și asociat cu un utilizator.

"Algoritmi" reprezintă metodele utilizate pentru procesarea hash-urilor. Fiecare algoritm este caracterizat prin tipurile de hash-uri suportate și este utilizat în cadrul proceselor de cracking pentru a determina parolele asociate.

Entitatea "Wordlisturi" definește colecțiile de parole utilizate în timpul proceselor brute-force. Acestea sunt asociate cu algoritmi specifici pentru a optimiza cracking-ul hash-urilor.

Această diagramă IDEF1X furnizează o reprezentare clară și coerentă a relațiilor dintre entitățile din aplicația de cracking al hash-urilor, facilitând astfel proiectarea, dezvoltarea și gestionarea sistemului.

Bibliografie

2024. *Hashcat.* 10 13. https://hashcat.net/wiki/.
2024. *Hashcat wiki.* 10 13. https://hashcat.net/wiki/.
2024. *openwall jhon the ripper.* 10 13. https://www.openwall.com/john/.
2024. *ophcrack.* 10 13. https://ophcrack.sourceforge.io/.