

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

**Realizarea unei aplicatii destinate capturarii si analizei**

**traficului dintr-o retea de calculatoare**

LUCRARE DE LICENȚĂ

Absolvent:

Coordonator:

**Cuprins**

[**Capitolul 2. Obiectivele Proiectului** 3](#_Toc195051488)

[**2.1. Definirea temei** 3](#_Toc195051489)

[**2.2. Obiectivele specifice ale proiectului** 3](#_Toc195051490)

[**2.3. Cerințele funcționale** 4](#_Toc195051491)

[**2.4. Cerințele non-funcționale** 4](#_Toc195051492)

[**Capitolul 3: Studiu Bibliografic** 5](#_Toc195051493)

[**3.1. Introducere în capturarea și analiza traficului de rețea** 5](#_Toc195051494)

[**3.2. Tehnici tradiționale de analiză a traficului** 5](#_Toc195051495)

[**3.3. Protocoale fundamentale in analiza traficului de retea** 5](#_Toc195051496)

[**3.4. Tehnologii și framework-uri pentru capturarea traficului** 6](#_Toc195051497)

[**3.5. Utilizarea învățării automate în analiza traficului de rețea** 6](#_Toc195051498)

[**3.6. Aplicații ale învățării profunde în analiza traficului** 7](#_Toc195051499)

[**3.7. Analiza traficului criptat** 7](#_Toc195051500)

[**3.8. Soluții existente pentru captură și analiză** 7](#_Toc195051501)

[**3.9. Compararea soluțiilor existente** 11](#_Toc195051502)

[**3.10. Limitările soluțiilor existente** 11](#_Toc195051503)

[**3.11. Concluzii asupra soluțiilor existente** 11](#_Toc195051504)

# **Capitolul 2. Obiectivele Proiectului**

## **2.1. Definirea temei**

Tema proiectului presupune crearea unei aplicații software care să permită capturarea și analiza traficului de rețea în timp real, furnizând statistici și informații utile pentru un administrator de rețea. Aplicația va folosi un framework pentru captarea pachetelor de rețea și va oferi opțiuni pentru filtrarea acestora, generarea de rapoarte și vizualizarea acestora într-o interfață web accesibilă de la distanță.

**Obiectivele principale ale proiectului sunt:**

* **Capturarea traficului de rețea:** Identificarea interfețelor de rețea disponibile pe dispozitivul gazdă și capturarea pachetelor de pe aceste interfețe.
* **Analiza traficului:** Procesarea pachetelor pentru identificarea protocoalelor de rețea (IPv4, IPv6, TCP, UDP, ICMP), împărțirea traficului pe fluxuri și generarea de statistici.
* **Stocarea datelor:** Salvarea informațiilor despre pachetele capturate (header-e, metadate) într-un format accesibil (CSV, XML, PCAP) pentru analiză ulterioară.
* **Vizualizarea datelor:** Oferirea unei interfețe grafice web pentru vizualizarea în timp real a statisticilor și compararea acestora cu datele istorice.

## **2.2. Obiectivele specifice ale proiectului**

1. **Identificarea și selecția interfețelor de rețea:** Aplicația va trebui să permită administratorului să identifice interfețele de rețea disponibile (Ethernet, Wi-Fi, etc.) și să selecteze cele pe care dorește să le monitorizeze.
   * **Tehnici de implementare:** Utilizarea librăriei libpcap (sau JPcap pentru Java) pentru identificarea interfețelor de rețea.
   * **Provocări tehnice:** Gestionarea permisiunilor de acces la interfețele de rețea (administrator/root), gestionarea eroilor și a întreruperilor de conexiune.
2. **Captura și analiza pachetelor:** Sistemul va capta pachetele de pe interfețele selectate și le va analiza pentru a extrage informații utile.
   * **Tehnici de implementare:** Analiza pachetelor folosind protocoale de bază (TCP, UDP, ICMP, IPv4, IPv6).
   * **Provocări tehnice:** Identificarea fluxurilor de trafic (fiecare flux fiind definit prin IP sursă, port sursă, IP destinație, port destinație și protocol) și analiza eficientă a unui trafic de rețea voluminos.
3. **Stocarea datelor:** Pachetele capturate vor fi salvate în fișiere pentru o analiză ulterioară. Datele vor include metadatele fiecărui pachet (header-e).
   * **Tehnici de implementare:** Utilizarea formatului PCAP (format standardizat pentru capturarea pachetelor) sau CSV/XML pentru stocarea datelor.
   * **Provocări tehnice:** Crearea unei structuri eficiente pentru stocarea și recuperarea datelor din fișiere mari.
4. **Generarea și vizualizarea statisticilor:** În timp real, aplicația va crea statistici despre trafic, fluxuri, protocoale, utilizare a rețelei și altele.
   * **Tehnici de implementare:** Utilizarea unui server web (de exemplu, Flask pentru Python) pentru a transmite statistici într-o interfață web.
   * **Provocări tehnice:** Gestionarea volumului mare de date și furnizarea de informații în timp real cu o latență minimă.

## **2.3. Cerințele funcționale**

1. **Identificarea interfețelor de rețea disponibile pe dispozitivul gazdă.**
   * Implementarea unui mecanism de detectare a interfețelor de rețea disponibile.
   * Permisiuni adecvate pentru accesul la interfețele de rețea.
2. **Capturarea pachetelor de pe interfețele de rețea.**
   * Filtrarea traficului pe baza protocoalelor de interes (IPv4, IPv6, TCP, UDP, ICMP).
   * Suport pentru capturarea pe mai multe interfețe simultan.
3. **Analizarea pachetelor capturate.**
   * Analiza headere-lor pentru extragerea informațiilor relevante: adrese IP, porturi, protocoale.
   * Calcularea statisticilor pe fluxuri de trafic.
4. **Filtrarea și clasificarea traficului.**
   * Posibilitatea de a filtra pachetele pe baza unor criterii precum IP sursă/destinație, porturi sursă/destinație, tipul de trafic.
5. **Stocarea datelor.**
   * Salvarea în fișiere PCAP pentru analiza ulterioară.
   * Salvarea metadatelor pachetelor în formate ușor accesibile (CSV/XML).
6. **Vizualizarea de la distanță.**
   * Crearea unei interfețe web care să permită accesul la statistici și informații în timp real.

## **2.4. Cerințele non-funcționale**

1. **Performanța:** Aplicația trebuie să proceseze și să salveze datele într-un timp scurt, cu un impact minim asupra performanței rețelei.
2. **Scalabilitate:** Sistemul trebuie să suporte monitorizarea mai multor interfețe de rețea și să poată scala pe măsură ce volumul de trafic crește.
3. **Flexibilitate:** Aplicația trebuie să permită extinderea cu noi protocoale sau funcționalități.
4. **Robustete:** Sistemul trebuie să gestioneze erorile și excepțiile, inclusiv întreruperile de rețea.
5. **Persistența datelor:** Datele trebuie să fie stocate eficient și să poată fi recuperate oricând, chiar și după oprirea aplicației.

# **Capitolul 3: Studiu Bibliografic**

## **3.1. Introducere în capturarea și analiza traficului de rețea**

Analiza traficului de rețea reprezintă un proces esențial în menținerea securității, performanței și integrității comunicațiilor în rețelele moderne. În esență, aceasta presupune monitorizarea, înregistrarea și interpretarea fluxurilor de date care circulă printr-o rețea de calculatoare, în scopul identificării de modele de utilizare, anomalii sau atacuri. Odată cu digitalizarea accelerată și extinderea rețelelor, analiza traficului a evoluat de la o activitate reactivă, de diagnostic, la un mecanism proactiv de securitate cibernetică și optimizare a resurselor [1].

Această practică este indispensabilă în centrele de date, infrastructurile critice, precum și în medii enterprise sau academice, contribuind la detectarea timpurie a intruziunilor, identificarea utilizării ineficiente a resurselor și stabilirea de politici de management al traficului.

## **3.2. Tehnici tradiționale de analiză a traficului**

Printre metodele tradiționale se numără analiza la nivel de pachet (packet-based) și analiza la nivel de flux (flow-based). Analiza la nivel de pachet permite inspectarea detaliată a fiecărui pachet transmis, inclusiv conținutul efectiv al acestuia, folosind metode de Deep Packet Inspection (DPI). Această metodă oferă o granularitate ridicată, însă ridică probleme legate de confidențialitate, resurse consumate și scalabilitate [2].

Pe de altă parte, analiza bazată pe flux presupune agregarea datelor în funcție de sursă, destinație, porturi și protocol, fără a examina conținutul pachetelor. Aceasta oferă o perspectivă mai eficientă și agregată asupra comportamentului rețelei, fiind utilizată în instrumente precum NetFlow sau sFlow.

## **3.3. Protocoale fundamentale in analiza traficului de retea**

Protocoalele de rețea definesc regulile prin care dispozitivele comunică în cadrul unei rețele de calculatoare. În contextul analizei traficului de rețea, este esențială înțelegerea modului în care aceste protocoale funcționează, deoarece permit interpretarea corectă a datelor capturate.

1. **IPv4 și IPv6 – Protocoale de rețea**

* **IPv4 (Internet Protocol Version 4)** este protocolul cel mai utilizat pentru identificarea și rutarea pachetelor de date în rețea. Acesta folosește adrese pe 32 de biți, permițând aproximativ 4,3 miliarde de adrese unice. Header-ul unui pachet IPv4 conține informații precum adresa IP sursă și destinație, TTL (Time to Live) și opțiuni de fragmentare [3].
* **IPv6** a fost introdus pentru a depăși limitările IPv4, extinzând spațiul de adresare la 128 de biți și adăugând capabilități suplimentare precum IPSec pentru securitate și un mecanism mai simplu de rutare [4].

1. **TCP și UDP – Protocoale de transport**

* **TCP (Transmission Control Protocol)** este un protocol orientat pe conexiune, care asigură livrarea fiabilă a datelor. Folosește mecanisme de retransmisie, confirmare și ordonare a pachetelor, fiind ideal pentru aplicații precum HTTP, e-mail sau transferuri FTP [5].
* **UDP (User Datagram Protocol)**, spre deosebire de TCP, este un protocol fără conexiune și nu oferă garanții privind livrarea. Este preferat în aplicații care necesită viteză ridicată și unde pierderea de pachete este tolerabilă, cum ar fi streamingul audio/video sau jocurile online [6].

1. **ICMP – Protocol de diagnosticare**

* **ICMP (Internet Control Message Protocol)** este utilizat pentru trimiterea de mesaje de control și diagnostic în rețea. Comenzi precum ping și traceroute se bazează pe ICMP pentru a evalua conectivitatea și traseele pachetelor în rețea [4].

1. **DSCP – Prioritizarea traficului**

* **DSCP (Differentiated Services Code Point)** este un mecanism implementat în header-ul IPv4/IPv6 pentru clasificarea și prioritizarea pachetelor. Acesta este util în rețele unde anumite tipuri de trafic, precum vocea sau video, trebuie să aibă prioritate față de alte tipuri (best effort) [7].

În procesul de captură și analiză a traficului, identificarea protocoalelor implicate este primul pas în înțelegerea comportamentului rețelei, detectarea anomaliilor și aplicarea de filtre utile pentru investigarea eficientă.

## **3.4. Tehnologii și framework-uri pentru capturarea traficului**

Capturarea pachetelor este posibilă datorită unor framework-uri care permit aplicațiilor să acceseze interfețele de rețea la nivel de kernel. Cele mai utilizate tehnologii includ:

1. **libpcap/WinPcap**

* Descriere: Biblioteca de bază utilizată pentru capturarea pachetelor. libpcap este disponibilă pe Unix/Linux, în timp ce WinPcap este varianta sa pentru Windows.
* Funcționalități:
  + Oferă acces la nivel scăzut la interfețele de rețea.
  + Permite capturarea pachetelor și aplicarea de filtre.
* Exemplu de utilizare: Wireshark folosește libpcap pentru capturarea traficului.

1. **NFQueue și Netfilter**

* Descriere: Tehnologii specifice Linux care permit manipularea pachetelor de rețea direct din kernel.
* Aplicații: Firewall-uri și sisteme de prevenire a intruziunilor.

1. **Apache Kafka**

* Descriere: O platformă distribuită pentru procesarea în timp real a datelor.
* Exemplu: Utilizarea Apache Kafka pentru stocarea și analiza fluxurilor mari de trafic în timp real.

## **3.5. Utilizarea învățării automate în analiza traficului de rețea**

Progresele recente în domeniul inteligenței artificiale (AI) și, în special, în învățarea automată (ML) au permis dezvoltarea unor sisteme mai eficiente de detecție a anomaliilor și clasificare a traficului. Aceste metode nu necesită neapărat inspecția conținutului pachetelor și pot fi antrenate să recunoască tipare de trafic asociate cu comportamente legitime sau malițioase [8].

Modelele ML, precum Random Forest, Support Vector Machines (SVM) sau K-Nearest Neighbors (KNN), sunt folosite pentru clasificarea traficului pe baza caracteristicilor statistice extrase din pachete (dimensiune, timp între pachete, număr de pachete etc.). Aceste metode permit o analiză rapidă și scalabilă, dar necesită date de antrenament de calitate și pot fi sensibile la modificările în comportamentul traficului [8].

## **3.6. Aplicații ale învățării profunde în analiza traficului**

În comparație cu metodele clasice ML, învățarea profundă (Deep Learning - DL) oferă o capacitate superioară de a extrage automat caracteristici relevante, fără inginerie manuală a datelor. Rețelele neuronale convoluționale (CNN) sunt eficiente în recunoașterea tiparelor spațiale, iar rețelele neuronale recurente (RNN) și variantele LSTM sunt potrivite pentru secvențe temporale de pachete [9].

Studiile arată că modelele bazate pe DL pot depăși performanțele celor tradiționale în sarcini precum clasificarea aplicațiilor, detectarea botnet-urilor sau analiza traficului criptat, având rate de detecție ridicate și o rezistență mai bună la variații ale datelor [9].

## **3.7. Analiza traficului criptat**

Unul dintre cele mai mari provocări actuale o constituie analiza traficului criptat (ex. HTTPS, VPN). Deși criptarea sporește securitatea comunicațiilor, ea limitează semnificativ posibilitățile de inspecție directă. În acest context, analiza trebuie să se bazeze pe metadate și pe caracteristici precum mărimea pachetelor, timpii dintre ele sau distribuția temporală a fluxului [10].

Studiile recente propun utilizarea învățării automate pentru identificarea serviciilor web criptate, detecția malware-ului în trafic TLS și clasificarea aplicațiilor mobile, toate realizate fără acces la conținutul efectiv al pachetelor [10].

## **3.8. Soluții existente pentru captură și analiză**

În acest subcapitol, vom analiza câteva dintre cele mai populare soluții existente pentru capturarea și analiza traficului de rețea, incluzând atât instrumente open-source, cât și soluții comerciale. În acest caz, vom discuta despre **Wireshark**, **tcpdump**, **Zeek (Bro)**, **SolarWinds Network Performance Monitor (NPM)** și **NetworkMiner**. Acestea oferă diferite abordări în funcție de cerințele rețelelor și scopurile utilizatorilor.

1. **Wireshark**

**Descriere:** Wireshark este unul dintre cele mai populare și utilizate instrumente pentru analiza și capturarea pachetelor de rețea. Este un **tool open-source** care permite capturarea pachetelor de date de pe interfețele de rețea și analiza lor detaliată, inclusiv decodificarea protocoalelor de rețea. Wireshark este folosit de administratorii de rețea, cercetătorii în securitate și dezvoltatorii de aplicații pentru a înțelege comportamentele rețelelor și pentru a rezolva probleme de performanță sau securitate.

**Funcționalități:**

* **Capturarea live a pachetelor:** Wireshark poate captura pachete de pe orice interfață de rețea disponibilă pe sistem.
* **Analiză detaliată a pachetelor:** Permite analiza fiecărui pachet capturat în detaliu, oferind informații despre header, protocoale și datele transmise.
* **Filtrare avansată:** Oferă un limbaj de filtrare puternic care permite selecția precisă a pachetelor de interes.
* **Decodare a protocoalelor:** Wireshark poate decoda sute de protocoale de rețea (IP, TCP, UDP, HTTP, DNS, și multe altele).
* **Analiză vizuală:** Grafice și statistici vizuale pentru a analiza traficul în funcție de diverse criterii (ex: volume de trafic pe fiecare protocol).
* **Exportul datelor:** Permite exportarea datelor capturate în formate precum **PCAP**, **CSV**, și **XML** pentru analize ulterioare.

**Avantaje:**

* **Open-source și gratuit.**
* Suport pentru un număr mare de protocoale și formate.
* Interfață grafică prietenoasă, ușor de utilizat.
* Comunitate activă și actualizări frecvente.
* Permite salvarea și analiza ulterioară a datelor capturate (întotdeauna poți analiza aceleași capturi multiple ori).

**Dezavantaje:**

* Poate consuma multe resurse, mai ales pe rețele mari.
* Este mai puțin eficient în capturarea traficului în timp real pentru rețele mari sau cu lățime de bandă mare.
* Interfața grafică poate deveni copleșitoare pentru începători din cauza complexității opțiunilor.

**Cazuri de utilizare:** Wireshark este ideal pentru depanarea rețelelor, analiza traficului de aplicație și diagnosticul de performanță. Este folosit frecvent pentru a înțelege comportamentele de rețea, pentru a detecta atacuri sau anomalii și pentru a analiza aplicațiile care rulează pe rețea.

1. **tcpdump**

**Descriere:** tcpdump este un alt instrument de capturare a pachetelor de rețea, dar spre deosebire de Wireshark, tcpdump funcționează din linia de comandă. Este un instrument extrem de rapid și eficient pentru capturarea pachetelor în rețelele de calculatoare, iar datorită simplității și eficienței sale, este folosit adesea pe servere și în scenarii automate sau de scripting.

**Funcționalități:**

* **Captura de pachete:** Poate capta pachete de pe orice interfață de rețea, inclusiv Wi-Fi.
* **Filtrare BPF (Berkeley Packet Filter):** Permite utilizatorilor să aplice filtre extrem de precise pentru a capta doar pachetele care îi interesează (de exemplu, toate pachetele dintr-o anumită adresă IP sau port).
* **Exportul datelor:** Pachetele capturate pot fi salvate în fișiere **PCAP** pentru analize ulterioare cu instrumente ca Wireshark.
* **Analiză de bază:** Poate decoda și analiza protocoale precum **IP, TCP, UDP, ICMP** și altele, dar cu mai puțină detaliere față de Wireshark.

**Avantaje:**

* **Rapid și eficient:** Este un instrument ușor de utilizat în medii de producție unde performanța este critică.
* **Consum redus de resurse:** Deoarece rulează în linie de comandă, consumă mult mai puține resurse comparativ cu Wireshark.
* **Ideal pentru scripting:** Poate fi folosit într-un mod complet automatizat și integrat în scripturi pentru capturi de pachete programate.
* **Flexibilitate:** Permite capturarea și filtrarea precisă a pachetelor, cu un control detaliat asupra ceea ce se captează.

**Dezavantaje:**

* **Lipsa interfeței grafice:** Nu are interfață grafică, ceea ce poate face analiza datelor mai dificilă pentru cei care nu sunt familiarizați cu linia de comandă.
* **Capacități limitate de analiză vizuală:** Deși poate analiza protocoale de rețea, nu oferă aceleași funcționalități vizuale avansate ca Wireshark.

**Cazuri de utilizare:** tcpdump este folosit frecvent pentru capturi rapide de pachete, diagnosticul problemelor de rețea și analize automate. Este de asemenea popular pe servere sau în medii în care resursele sunt limitate sau când se dorește integrarea sa într-un sistem automatizat de monitorizare.

1. **Zeek (fost Bro)**

**Descriere: Zeek**, cunoscut anterior sub denumirea de **Bro**, este un sistem open-source pentru monitorizarea și analiza rețelelor. Spre deosebire de Wireshark și tcpdump, Zeek este un **framework de securitate și monitorizare** a rețelelor care se concentrează pe analiza și detecția evenimentelor din rețea la nivel înalt. Este folosit pentru a examina și detecta activitățile de rețea, pentru a identifica comportamente anormale sau potențial dăunătoare și pentru a genera loguri detaliate ale activităților de rețea.

**Funcționalități:**

* **Monitorizarea rețelei la nivel de aplicație:** Zeek analizează fluxurile de rețea și detectează anomalii la nivelul aplicațiilor.
* **Detectarea evenimentelor de securitate:** Zeek poate fi configurat să detecteze diverse tipuri de atacuri, cum ar fi scanările de porturi, atacurile DDoS, injectarea de comenzi SQL, etc.
* **Scripturi personalizate:** Oferă un limbaj propriu de scripting, **Zeek scripting language**, care permite utilizatorilor să creeze reguli personalizate pentru a analiza evenimente și protocoale specifice.
* **Loguri detaliate:** Generează loguri detaliate care pot fi folosite pentru analiza ulterioară sau pentru investigarea incidentelor de securitate.
* **Suport pentru protocoale diverse:** Analizează o gamă largă de protocoale (HTTP, DNS, FTP, SSH, SMTP etc.) și poate integra datele cu alte soluții de securitate (de exemplu, SIEM).

**Avantaje:**

* **Puternic în securitate:** Zeek este un instrument de securitate avansat, excelent pentru detectarea atacurilor și monitorizarea comportamentului rețelelor.
* **Extensibilitate:** Posibilitatea de a crea scripturi personalizate pentru a analiza protocoale specifice sau pentru a implementa politici de securitate.
* **Loguri detaliate:** Oferă o gamă largă de loguri pentru a urmări activitățile din rețea și pentru a răspunde la incidente de securitate.
* **Scalabilitate:** Zeek poate fi scalat pentru a monitoriza rețele mari și complexe, fiind potrivit pentru organizații mari și centre de date.

**Dezavantaje:**

* **Complexitatea configurării:** Zeek necesită o configurare inițială și un învățământ detaliat pentru a fi utilizat eficient.
* **Consum mare de resurse:** Monitorizarea rețelelor mari poate consuma resurse semnificative, fiind necesare hardware puternice pentru a procesa traficul în timp real.
* **Curba de învățare:** Zeek poate fi mai dificil de utilizat pentru cei care nu au experiență în scripting sau administrarea rețelelor.

**Cazuri de utilizare:** Zeek este utilizat adesea în medii de securitate avansată, cum ar fi centrele de securitate a rețelelor, pentru a detecta comportamente anormale, pentru a implementa reguli personalizate de monitorizare și pentru a răspunde la incidente de securitate în timp real.

1. **SolarWinds Network Performance Monitor (NPM)**

**Descriere:** SolarWinds Network Performance Monitor (NPM) este o soluție comercială de monitorizare a rețelei, care include și funcționalități pentru capturarea și analiza traficului de rețea. Este o soluție completă destinată organizațiilor mari și medii, care doresc o platformă centralizată pentru a monitoriza performanța rețelelor, dispozitivelor și aplicațiilor.

**Funcționalități:**

* **Monitorizare continuă:** SolarWinds NPM monitorizează rețeaua în timp real, identificând întreruperile și problemele de performanță.
* **Captură și analiză a traficului:** Permite monitorizarea fluxurilor de trafic pe baza protocolului, a adresei IP și a porturilor.
* **Vizualizare a performanței rețelei:** Grafice și hărți detaliate ale rețelei pentru a vizualiza performanța în timp real.
* **Alertare și rapoarte:** Sistem de alerte în timp real și rapoarte detaliate care ajută administratorii să identifice rapid problemele.
* **Analiză a aplicațiilor:** SolarWinds NPM poate urmări traficul aplicațiilor pentru a determina performanța acestora și impactul asupra rețelei.
* **Configurare ușoară:** Configurare și implementare rapidă printr-o interfață web intuitivă.
* **Scalabilitate:** Suportă rețele mari, scalându-se ușor în funcție de nevoile organizației.

**Avantaje:**

* Suport excelent pentru rețele complexe și mari.
* Ușor de utilizat, cu o interfață grafică prietenoasă.
* Integrare cu soluții de management ale rețelei și de securitate.
* Capabilități de monitorizare a aplicațiilor și a performanței rețelei.

**Dezavantaje:**

* Este o soluție costisitoare, potrivită pentru organizații mari.
* Poate fi mai puțin flexibil în comparație cu soluțiile open-source personalizabile.
* Necesită licențiere anuală, ceea ce adaugă costuri suplimentare pe termen lung.

1. **NetworkMiner**

**Descriere:** NetworkMiner este un instrument de analiză forensică a traficului de rețea, special conceput pentru a extrage informații utile din pachetele de rețea capturate. Spre deosebire de Wireshark, NetworkMiner se concentrează pe recuperarea și prezentarea datelor importante din capturile de pachete, cum ar fi fișierele transferate, utilizatorii autentificați și sesiunile de rețea.

**Funcționalități:**

* **Analiză forensică:** NetworkMiner este folosit pentru a reconstruirea sesiunilor de rețea și extragerea datelor transmise (fișiere, parole, etc.).
* **Detectarea sesiunilor și utilizatorilor:** Poate extrage date din sesiunile HTTP, FTP, POP3, IMAP, SMB și multe altele.
* **Decodificarea și analiza protocoalelor:** Permite decodificarea completă a protocoalelor și analizarea fișierelor transferate.
* **Captura offline:** Poate analiza fișierele de captură deja realizate (PCAP) pentru a extrage informațiile relevante.
* **Interfață grafică ușor de utilizat:** Interfață intuitivă care permite o analiză detaliată a traficului capturat.

**Avantaje:**

* Ideal pentru analiza incidentelor de securitate și investigații forensice.
* Permite extragerea detaliilor sensibile din traficul de rețea, cum ar fi parolele și fișierele transmise.
* Ușor de utilizat pentru cei care nu sunt experți în analiza pachetelor.

**Dezavantaje:**

* Nu este la fel de complet în ceea ce privește analiza în timp real.
* Suportă mai puține protocoale comparativ cu Wireshark sau tcpdump.
* Poate fi mai limitat pentru analize de performanță în rețele mari.

## **3.9. Compararea soluțiilor existente**

Un tabel comparativ între soluțiile existente.

## **3.10. Limitările soluțiilor existente**

* **Wireshark**: Nu este ideal pentru rețele mari sau pentru capturarea traficului în timp real din cauza consumului mare de resurse.
* **tcpdump**: Deși eficient și rapid, nu oferă capabilități avansate de analiză vizuală.
* **Zeek**: Necesită un hardware puternic și o configurare detaliată, ceea ce îl face dificil de implementat fără experiență.
* **SolarWinds NPM**: Costisitor și mai puțin personalizabil în comparație cu soluțiile open-source.
* **NetworkMiner**: Limitează analiza în timp real și nu poate decoda toate protocoalele.

## **3.11. Concluzii asupra soluțiilor existente**

Analiza traficului de rețea este într-o continuă schimbare, iar tehnologiile emergente, precum **inteligenta artificială** și **machine learning-ul**, vor juca un rol tot mai important. Aceste tehnologii vor putea ajuta la detectarea automată a anomaliilor și a atacurilor sofisticate în timp real. Totodată, **tehnologiile 5G** și **internetul obiectelor (IoT)** vor genera noi provocări, deoarece vor aduce o creștere semnificativă a volumului și diversității traficului de rețea.

În concluzie, capturarea și analiza traficului de rețea reprezintă un domeniu esențial pentru menținerea securității și performanței rețelelor moderne. Soluțiile existente variază în funcție de complexitatea și scopul utilizării, iar alegerea celor mai potrivite instrumente depinde de cerințele fiecărei rețele în parte.