

CUPRINS

INTRODUCERE.....	9
1 ANALIZA DOMENIULUI DE STUDIU.....	10
1.1 Importanța temei	10
1.2 Sisteme similare cu proiectul realizat	11
1.3 Scopul, obiectivele și cerințele sistemului	15
2 MODELAREA ȘI PROIECTAREA SISTEMUL INFORMATIC	17
2.1 Descrierea comportamentală a sistemului	18
2.1.1 Imaginea generală asupra sistemului	19
2.1.2 Modelarea vizuală a fluxurilor.....	20
2.1.3 Descrierea scenariilor de utilizare a aplicației	22
2.2 Descrierea structurală a sistemului.....	23
2.2.1 Descrierea structurii statice a sistemului	24
2.2.2 Relațiile de dependență între componentele sistemului.....	25
3 REALIZAREA SISTEMULUI.....	27
3.1 Structura sistemului	27
3.2 Integrarea serviciilor externe	28
3.3 Implementarea cerințelor funcționale	29
3.4 Arhitectura modulară a aplicației.....	30
3.5 Modelul de date și structura bazei de date.....	31
3.6 Fluxul de execuție în aplicație.....	32
3.7 Sistemul de alertare și detectare a comportamentelor suspecte.....	33
3.8 Generarea de rapoarte și vizualizări	34
3.9 Implementarea funcției Follow Stream pentru analiza comunicării în rețea	35
4 DOCUMENTAREA SISTEMULUI	38
4.1 Interfața de autentificare	38
4.3 Interacțiunea cu baza de date.....	39
4.4 Interfața principală a aplicației.....	40
4.5 Capturarea traficului de rețea.....	40
4.6 Vizualizarea hosturilor detectate în rețea	41
4.7 Analiza detaliată a fluxurilor de date (Follow Stream).....	42
4.8 Funcționalități disponibile pentru analiza pachetelor individuale.....	44
5 ESTIMAREA COSTURILOR ȘI EVALUAREA PROIECTULUI	46
5.1 Estimarea costurilor	46
5.2 Aspecte tehnice și riscuri întâmpinate	46
5.3 Evaluarea funcționalităților și testarea	47
BIBLIOGRAFIE.....	50

INTRODUCERE

În lumea digitală actuală, rețelele de calculatoare joacă un rol central în funcționarea instituțiilor, a întreprinderilor și a societății în ansamblu. Din ce în ce mai multe procese critice se bazează pe infrastructurile IT, iar fluxurile de date devin din ce în ce mai dense, variate. În acest context, protejarea traficului de rețea și asigurarea integrității comunicațiilor reprezintă provocări majore în materie de securitate cibernetică.

Lucrarea de față se concentrează pe proiectarea și implementarea unei aplicații software pentru interceptarea și analiza traficului de rețea în timp real [1], [2],[4], în vederea identificării comportamentelor suspecte, a posibilelor atacuri și a activităților anormale în cadrul unei rețele locale sau extinse. Aplicația asigură o interfață grafică intuitivă (GUI) prin intermediul căreia utilizatorul poate iniția sesiuni de captură, vizualiza pachetele interceptate, interpreta informații din diferite protocoale (TCP, UDP, HTTP, DNS etc.) și poate genera rapoarte și alerte bazate pe criterii predeterminate.

Motivul alegerii acestei teme provine din nevoia tot mai mare de a înțelege și contracara în timp util amenințările cibernetică. Având în vedere că atacurile de tip phishing, malware, port scanning sau DoS devin din ce în ce mai frecvente, implementarea unor soluții proactive de monitorizare a traficului devine practic pentru prevenirea incidentelor și protejarea datelor. Acest proiect contribuie în această direcție prin oferirea unui instrument capabil să ofere o imagine vizibilă asupra activității rețelei, inclusiv prin evidențierea protocoalelor utilizate, a surselor de trafic și a volumelor de date prelucrate.

Implementarea aplicației implică includerea tehnologiilor open-source și a metodelor eficiente de analiză, cum ar fi utilizarea bibliotecii Scapy pentru capturarea pachetelor în Python, prelucrarea și salvarea datelor relevante în format PCAP și JSON și implementarea unui sistem personalizat de detectare a anomaliilor bazat pe reguli. O parte importantă este capacitatea de a salva sesiuni și de a genera rapoarte PDF cu statistici detaliate.

Lucrarea evidențiază, de asemenea, procesul de proiectare modulară a aplicației și abordarea orientată spre utilizator, astfel încât sistemul să fie ușor de utilizat, dar suficient de flexibil pentru a putea fi extins în viitor. Prin această contribuție, se dorește oferirea unui exemplu concret de aplicație practică în domeniul securității cibernetică, care poate fi utilizată atât în scop educațional, cât și ca bază pentru soluții mai complexe de tip IDS (Intrusion Detection System) sau SIEM (Security Information and Event Management).

În concluzie, această lucrare se înscrie în eforturile curente de creștere a rezilienței rețelelor informatice și de susținere a securității infrastructurilor digitale, prin realizarea unei aplicații funcționale, bine documentate și cu un impact practic real în domeniul monitorizării traficului de rețea.

1 ANALIZA DOMENIULUI DE STUDIU

În realizarea proiectului propus, s-a acordat o atenție deosebită procesului de capturare și analiză a traficului de rețea, utilizând o serie de tehnologii și instrumente consacrate în domeniul securității cibernetice. Activitatea s-a concentrat pe înțelegerea funcționării protocoalelor de rețea și a mecanismelor prin care pachetele de date pot fi interceptate, clasificate și evaluate în timp real.

Pentru a construi o bază solidă în procesul de dezvoltare a aplicației, au fost analizate și comparate diverse soluții existente, precum Wireshark, tcpdump, Zeek și Suricata. Acestea au oferit repere importante în ceea ce privește abordările utilizate în captarea și interpretarea traficului, precum și în gestionarea volumului ridicat de date ce circulă prin rețele moderne. Prin testarea funcționalităților acestor instrumente, au fost identificate metode eficiente de prelucrare a informațiilor, dar și limitări care au stat la baza motivării dezvoltării unui instrument propriu.

Aplicația propusă în cadrul acestei lucrări a fost proiectată pornind de la nevoia de a crea o soluție personalizabilă, accesibilă și ușor de utilizat, care să permită monitorizarea activității din rețea, detectarea anomaliilor și înregistrarea evenimentelor relevante. Accentul a fost pus pe flexibilitate și extensibilitate, cu posibilitatea de adăugare a unor componente noi, precum sistemele de alertare sau modulele de export și raportare.

În urma procesului de testare și validare, s-a demonstrat că implementarea unei aplicații proprii aduce beneficii semnificative în ceea ce privește înțelegerea fluxurilor de date, localizarea surselor de trafic suspect și evidențierea tiparelor de comunicație. În perspectivă, soluția poate fi extinsă prin integrarea unor funcții avansate de analiză comportamentală, inclusiv prin utilizarea unor algoritmi de învățare automată pentru detecția proactivă a riscurilor.

1.1 Importanța temei

În contextul digitalizării accelerate a proceselor economice, administrative și sociale, rețelele de calculatoare au devenit coloana vertebrală a infrastructurilor moderne. Odată cu această expansiune, au crescut și riscurile asociate traficului de date, iar protecția rețelelor a devenit o preocupare centrală în domeniul securității cibernetice. Implementarea unui sistem capabil să intercepteze și să analizeze traficul de rețea oferă posibilitatea de a observa în timp real comportamentul comunicațiilor și de a identifica activitățile neobișnuite care pot semnală tentative de compromitere, abuzuri sau defecțiuni.

Monitorizarea atentă a fluxurilor de date permite nu doar identificarea rapidă a anomaliilor, dar și adoptarea unor măsuri de remediere înainte ca un incident să devină critic. Prin evidențierea comportamentelor suspecte ale utilizatorilor, dispozitivelor sau aplicațiilor, se pot preveni atacuri precum exfiltrarea de date, escaladarea privilegiilor, scanările de porturi sau atacurile de tip denial-of-service. Totodată, analiza traficului contribuie la optimizarea utilizării resurselor rețelei, ajutând la identificarea segmentelor congestionate sau a configurărilor ineficiente.

Importanța acestei teme derivă și din necesitatea de a respecta reglementările naționale și internaționale privind protecția datelor, auditul rețelilor și trasabilitatea evenimentelor digitale. Organizațiile sunt din ce în ce mai supuse verificărilor privind conformitatea cu standarde precum ISO/IEC 27001, NIS2 sau GDPR, iar un sistem eficient de monitorizare a traficului joacă un rol important în acest sens. În conformitate cu conceptele fundamentale prezentate de Stallings în [12], vizibilitatea și controlul asupra traficului sunt esențiale pentru prevenirea atacurilor și menținerea integrității rețelei.

Lucrarea de față propune o abordare aplicativă, axată pe compararea unor instrumente consacrate (precum Wireshark, tcpdump, Zeek și Suricata) în vederea identificării limitărilor acestora, precum și pe proiectarea și implementarea unei soluții proprii care să ofere o detecție mai rapidă, mai clară și mai ușor de integrat într-un mediu operațional. În acest mod, cercetarea contribuie la dezvoltarea de metode și instrumente moderne în sprijinul securității rețelilor informatice.

1.2 Sisteme similare cu proiectul realizat

În domeniul securității informatice, au fost dezvoltate numeroase aplicații destinate supravegherii și interpretării traficului de rețea. Aceste soluții sunt folosite atât în scopuri de monitorizare pasivă, cât și pentru identificarea în timp util a activităților neobișnuite sau potențial periculoase. Printre cele mai cunoscute și utilizate instrumente în acest sens se numără Wireshark[2], un software consacrat pentru inspecția detaliată a pachetelor transmise în cadrul rețelilor IP.

Wireshark oferă posibilitatea de a captura pachetele de date și de a le examina în profunzime, permițând utilizatorilor să aplice filtre personalizate pentru o analiză direcționată. Această funcționalitate este deosebit de utilă pentru depanarea problemelor de conectivitate, analiza performanței aplicațiilor sau investigarea traficului neobișnuit. Nivelul de detaliu disponibil este ridicat, iar interfața grafică ajută la structurarea datelor într-un mod accesibil. În **figura 1.1**, este ilustrat un exemplu de interfață Wireshark cu pachete capturate și decodificate la nivel de protocol.

Cu toate avantajele sale, Wireshark prezintă și unele limite, mai ales în situațiile în care este necesară o reacție rapidă la incidente. Procesul de analiză este predominant manual și poate deveni dificil de gestionat în rețele mari, unde volumul traficului este ridicat. În astfel de cazuri, interpretarea datelor brute necesită cunoștințe avansate din partea operatorilor și poate consuma timp important, ceea ce afectează capacitatea de reacție la evenimentele de securitate.

De aceea, în practică, Wireshark este adesea utilizat împreună cu alte sisteme care dispun de funcționalități automate de alertare și clasificare, cum ar fi sistemele de detectare și prevenire a intruziunilor (IDS/IPS). Acestea pot prelua sarcina de supraveghere continuă, în timp ce Wireshark este folosit pentru investigații detaliate post-eveniment sau pentru instruirea personalului în analiza comunicațiilor digitale.

Prin urmare, Wireshark reprezintă un instrument valoros în procesul de învățare și explorare a arhitecturii rețelilor, dar nu este suficient de eficient în mod izolat pentru a acoperi cerințele unui sistem de

apărare complet. Utilizarea sa într-un cadru mai larg, alături de alte soluții specializate, permite o abordare mai echilibrată a protecției infrastructurilor de rețea.

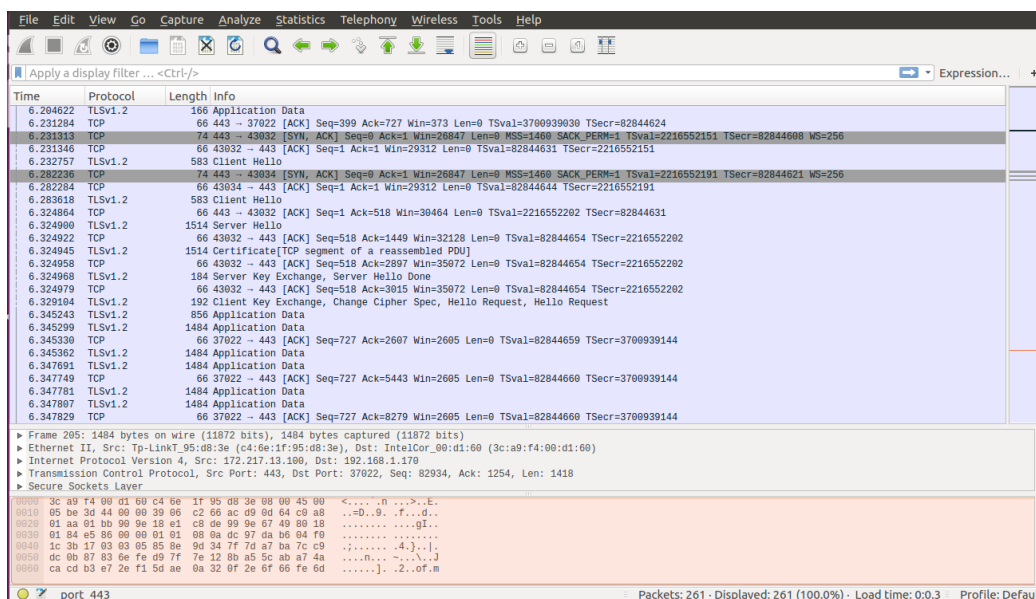


Figura 1.1 – Wireshark

Pe lângă instrumentele cu interfețe grafice, în analiza traficului de rețea sunt frecvent utilizate și soluții în linie de comandă, apreciate pentru flexibilitatea lor și pentru integrarea facilă în procese automatizate. Aceste unelte sunt preferate mai ales în medii server-based sau în situații în care se dorește monitorizarea discretă și consum redus de resurse. Una dintre cele mai cunoscute aplicații din această categorie este tcpdump, descrisă mai jos.

tcpdump este un utilitar în linie de comandă, frecvent utilizat pentru captarea și filtrarea traficului de rețea în timp real. Acesta funcționează prin monitorizarea pachetelor transmise prin interfețele de rețea ale sistemului și permite afișarea detaliată a acestora direct în consolă. Din cauza simplității sale și a consumului redus de resurse, tcpdump este adesea ales pentru diagnosticare rapidă în medii bazate pe Linux sau Unix, în special în scenarii în care interfețele grafice nu sunt disponibile sau nu sunt necesare.

Interfața sa text-based îl face potrivit pentru utilizatorii care dețin cunoștințe solide de rețea, deoarece interpretarea ieșirilor presupune înțelegerea structurii pachetelor și a protocoalelor implicate (IP, TCP, UDP, ICMP etc.). tcpdump suportă o varietate de filtre care permit selectarea traficului relevant, ceea ce îl transformă într-un instrument versatil pentru monitorizarea activității în rețele locale sau distribuite. În figura 1.2, este prezentat un exemplu de rulare a tcpdump într-un terminal, cu afișarea detaliată a pachetelor capturate.

Totuși, tcpdump nu include facilități grafice sau mecanisme automate de analiză, iar utilizarea sa eficientă presupune familiaritate cu expresiile de filtrare și structura internă a protocoalelor. Pentru investigații mai aprofundate sau pentru revizuirea capturilor într-un format vizual, fișierele .pcap generate

de tcpdump sunt adesea analizate ulterior în aplicații precum Wireshark, care oferă o reprezentare mai accesibilă a traficului.

Deși reprezintă o soluție viabilă pentru inspecția manuală și rapidă a comunicațiilor de rețea, tcpdump nu este conceput pentru a înlocui sistemele de detecție automată. Lipsa unor componente de alertare sau clasificare în timp real limitează utilizarea sa în strategii complexe de apărare. Astfel, în practica securității informatice, tcpdump este cel mai eficient atunci când este integrat într-un cadru mai larg de monitorizare, în combinație cu alte instrumente precum IDS/IPS sau platforme SIEM.

```
root@kali:~# tcpdump -i eth0 -s 0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
04:57:29.856624 IP 192.168.20.1.40816 > 192.168.20.255.40816: UDP, length 131
04:57:29.857265 IP kali.58048 > _gateway.domain: 22425+ PTR? 255.20.168.192.in-addr.arpa. (45)
04:57:29.858650 IP _gateway.domain > kali.58048: 22425 NXDomain 0/1/0 (80)
04:57:29.859271 IP kali.42264 > _gateway.domain: 27790+ PTR? 1.20.168.192.in-addr.arpa. (43)
04:57:29.860750 IP _gateway.domain > kali.42264: 27790 NXDomain 0/1/0 (78)
04:57:29.861150 IP kali.43676 > _gateway.domain: 36900+ PTR? 2.20.168.192.in-addr.arpa. (43)
04:57:29.862288 IP _gateway.domain > kali.43676: 36900 NXDomain 0/1/0 (78)
04:57:29.862621 IP kali.46656 > _gateway.domain: 19118+ PTR? 136.20.168.192.in-addr.arpa. (45)
04:57:30.638164 ARP, Request who-has _gateway tell kali, length 28
04:57:30.638292 ARP, Reply _gateway is-at 00:50:56:fd:dc:24 (oui Unknown), length 46
04:57:34.506805 IP 192.168.20.1.40816 > 192.168.20.255.40816: UDP, length 131
^C
11 packets captured
12 packets received by filter
1 packet dropped by kernel
```

Figura 1.2 – tcpdump

În analiza comportamentală a traficului de rețea, sunt necesare instrumente care să ofere mai mult decât capturarea brută a pachetelor. Unele soluții se concentrează pe interpretarea fluxurilor în context, generând informații utile despre activitatea desfășurată în rețea. Un exemplu reprezentativ în această categorie este **Zeek**, cunoscut anterior sub denumirea Bro[5].

Zeek este o platformă de monitorizare a rețelei care combină funcționalitatea clasică de captură a pachetelor cu un sistem avansat de analiză comportamentală. Pe lângă interceptarea datelor, Zeek poate interpreta protocoale de nivel superior (precum HTTP, DNS, SSL), extrăgând informații relevante care sunt ulterior organizate în fișiere jurnal structurate. Aceste fișiere descriu interacțiunile detectate și pot fi folosite pentru investigarea activităților suspecte sau pentru auditul comunicațiilor din rețea. În figura 1.3 este ilustrată interfața de lucru asociată jurnalelor generate de Zeek.

Spre deosebire de sistemele bazate exclusiv pe semnături, Zeek se concentrează pe observarea comportamentelor și pe corelarea evenimentelor. Astfel, poate surprinde modele de acțiune care nu corespund unor amenințări cunoscute, oferind o perspectivă mai extinsă asupra tacticilor și tehnicilor utilizate de actori rău-intenționați. De exemplu, analiza conținutului cererilor DNS, a sesiunilor SSL sau a tranzacțiilor HTTP poate indica prezența unui canal de comunicație ascuns sau a unei tentative de comandă și control.

Un alt avantaj important al Zeek constă în suportul pentru scripturi personalizate, care permit ajustarea regulilor și comportamentului aplicației în funcție de nevoile specifice ale administratorilor sau

ale politicilor de securitate adoptate. Prin aceste scripturi, utilizatorii pot defini condiții de alertare, extragere a anumitor câmpuri sau integrare cu alte sisteme, precum platforme SIEM sau soluții de orchestrare și automatizare.

Zeek este, așadar, potrivit pentru medii în care este necesară o înțelegere detaliată a traficului și o capacitate de adaptare continuă. Jurnalele și alertele pe care le produce pot sprijini analiza post-eveniment, oferind date clare despre tipurile de activitate care au avut loc într-o rețea într-un anumit interval de timp.

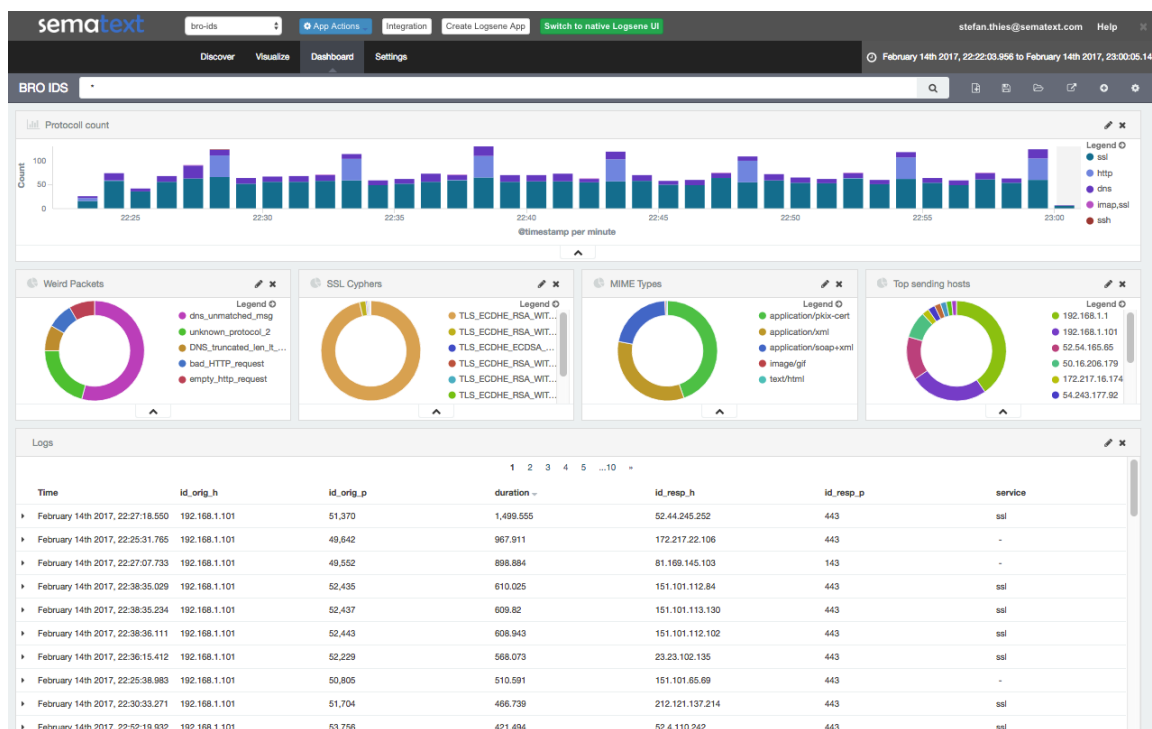


Figura 1.3 – Zeek

În completarea soluțiilor dedicate analizei pasive, există și instrumente care combină capacitățile de monitorizare cu acțiuni concrete de răspuns la incidente. Un exemplu în această direcție este **Suricata**, o platformă care funcționează atât ca sistem de detecție a intruziunilor (IDS), cât și ca sistem de prevenire (IPS), fiind capabilă să analizeze traficul de rețea în timp real și să aplice politici de filtrare sau blocare automată.

Spre deosebire de aplicațiile precum Wireshark sau tcpdump, care sunt axate în principal pe inspecția manuală a traficului capturat, Suricata poate interpreta fluxurile de date în mod automatizat, identificând semnale de atacuri cunoscute și comportamente suspecte, pe baza unor reguli configurabile. Această abordare permite o reacție mai rapidă în fața riscurilor detectate, fără a necesita intervenție directă imediată din partea unui operator uman.

Suricata folosește o bază extinsă de semnături, compatibilă cu formatul Snort, dar și mecanisme avansate de analiză, inclusiv suport pentru decodarea pachetelor la nivel de aplicație și pentru extragerea conținutului. În plus, oferă posibilitatea de a lucra cu fișiere jurnal structurate (JSON, EVE logs) care pot fi integrate ușor în soluții externe de analiză sau vizualizare, precum ELK Stack (Elasticsearch, Logstash, Kibana).

Sistemul propus în cadrul lucrării de față urmărește să combine punctele forte identificate în aplicații precum Wireshark, tcpdump, Zeek și Suricata, într-o platformă unificată, cu o interfață grafică accesibilă și funcționalități avansate de filtrare și clasificare. În locul unei analize brute, consumatoare de timp și expertiză, soluția implementată oferă o abordare orientată pe eficiență, care reduce efortul necesar pentru interpretarea datelor și scurtează timpul de reacție la evenimentele relevante.

Prin corelarea automată a informațiilor din trafic și generarea de rapoarte sintetice, sistemul oferă un sprijin consistent în procesul de decizie. Acest tip de integrare permite adaptarea mai bună la cerințele moderne de securitate, unde viteza de reacție și acuratețea detecției joacă un rol important în menținerea stabilității infrastructurii de rețea.

1.3 Scopul, obiectivele și cerințele sistemului

Proiectul de față are ca direcție principală dezvoltarea unei aplicații software capabile să intercepteze, analizeze și prezinte informațiile rezultate din traficul de rețea într-un mod accesibil și structurat. Prin această abordare, se urmărește construirea unui instrument funcțional care să contribuie la supravegherea comunicării dintre dispozitive și la identificarea comportamentelor care pot ridica semne de întrebare din perspectiva securității.

Soluția implementată trebuie să ofere o modalitate rapidă și automatizată de captare a datelor, să filtreze conținutul nerelevant și să evidențieze acele interacțiuni care pot indica nereguli, configurări greșite sau tentative de acces neautorizat. În plus, sistemul va permite păstrarea istoricului comunicațiilor și prezentarea acestora într-o formă care facilitează luarea de decizii.

Pentru atingerea acestor obiective, proiectul propune următoarele componente funcționale:

a) Captarea traficului de rețea, realizarea unui modul care să permită monitorizarea în timp real a datelor transmise între nodurile din rețea, cu ajutorul unor tehnologii consacrate precum tcpdump, Wireshark, Zeek sau Suricata[5],[6], în funcție de contextul de utilizare;

b) Filtrarea și clasificarea pachetelor, introducerea unor mecanisme care să selecteze doar acele informații relevante pentru analiză, înlăturând traficul de fundal sau datele irelevante pentru scopul propus. Aceasta presupune interpretarea pachetelor după protocol, port, adresă IP sau alte criterii specific;

c) Detecția activităților suspecte, integrarea unor metode de observare a comportamentului traficului, prin reguli statice sau modele configurabile, pentru a semnaliza posibile acțiuni nedorite sau ieșite din tiparele obișnuite;

d) Stocarea și gestionarea datelor, proiectarea unui sistem de înregistrare a informațiilor colectate, fie în fișiere de tip log, fie într-o bază de date structurată, pentru a facilita analiza ulterioară și generarea de statistici;

e) Interfață de utilizator, dezvoltarea unei componente vizuale care să permită utilizatorului să interacționeze facil cu datele procesate, să filtreze și să navigheze printre înregistrări, dar și să genereze rapoarte sintetice pentru documentare;

f) Automatizarea procesului, reducerea dependenței de intervenții manuale prin integrarea unor funcții automate de declanșare a capturilor, aplicare a filtrelor, clasificare și alertare, astfel încât utilizatorii să poată gestiona cu eficiență situațiile apărute.

Pentru ca acest sistem să funcționeze corespunzător, este necesar să îndeplinească o serie de cerințe care pot fi împărțite în două categorii: funcționale și nefuncționale.

Din punct de vedere funcțional, aplicația trebuie să capteze datele în timp real, să le interpreteze corect în funcție de contextul rețelei și să permită diferențierea între fluxuri normale și cele care pot indica riscuri. Aceasta presupune atât aplicarea unor filtre bazate pe protocoale, cât și capacitatea de a corela diferite pachete sau sesiuni pentru a înțelege intenția comunicației. Informațiile obținute trebuie să poată fi salvate și consultate ulterior, iar utilizatorul trebuie să dispună de un mediu vizual unde poate explora conținutul și genera documente utile pentru analiză sau raportare.

Din perspectiva cerințelor nefuncționale, sistemul trebuie să poată fi extins în viitor, în funcție de apariția unor nevoi noi sau de modificări în structura rețelei. De asemenea, aplicația trebuie să funcționeze stabil și să evite supraîncărcarea resurselor, în special în rețele mari sau cu trafic intens. Interfața trebuie să fie clară și bine organizată, pentru a permite o utilizare cât mai eficientă, inclusiv în condiții de presiune operațională. Nu în ultimul rând, sistemul trebuie să fie proiectat în conformitate cu normele legale privind supravegherea comunicațiilor, în special cele legate de confidențialitate și protecția datelor cu caracter personal.

Prin respectarea acestor cerințe, aplicația poate fi utilizată atât pentru activități de instruire și testare, cât și ca punct de plecare pentru soluții mai complexe dedicate mediilor operaționale în care vizibilitatea asupra traficului de rețea este indispensabilă pentru luarea deciziilor.

2 MODELAREA ȘI PROIECTAREA SISTEMUL INFORMATIC

În cadrul procesului de dezvoltare a unui sistem informatic complex, modelarea și proiectarea reprezintă etape fundamentale care permit structurarea logică și tehnică a soluției propuse. Aceste activități au un rol decisiv în definirea funcționalităților, stabilirea relațiilor dintre componente și anticiparea modului în care sistemul va reacționa în diverse scenarii de operare. În cazul de față, sistemul vizează interceptarea și analiza traficului de rețea, ceea ce presupune un nivel ridicat de coerență între modulele funcționale și o integrare eficientă a tehnologiilor implicate.

Modelarea sistemului se concentrează pe identificarea entităților care compun aplicația, precum și a fluxurilor de date care circulă între acestea. Este vorba despre o etapă în care se construiește o reprezentare conceptuală a arhitecturii, menită să asigure o înțelegere comună între toți membrii echipei de dezvoltare. În această reprezentare se includ atât elementele care preiau informațiile din rețea (modulele de captură), cât și cele care analizează, clasifică, salvează și prezintă aceste date (componentele de procesare, de stocare și de vizualizare).

Proiectarea, pe de altă parte, are în vedere transpunerea acestei arhitecturi conceptuale într-o structură tehnică clară. Se stabilesc astfel componentele software care vor alcătui sistemul, se determină modul în care acestea vor comunica între ele și se aleg tehnologiile corespunzătoare. Acest demers include, de asemenea, analiza cerințelor de performanță, scalabilitate și interoperabilitate, astfel încât soluția să fie adaptabilă și eficientă în contexte variate de utilizare.

Pentru a facilita această etapă, este utilizat **Limbajul Unificat de Modelare (UML)**, o metodă standardizată care permite descrierea vizuală a structurii și comportamentului sistemului. UML oferă un set de diagrame care ajută la documentarea precisă a procesului de dezvoltare, precum:

- **Diagramele de clasă**, care reflectă entitățile-cheie ale sistemului (de exemplu, modulele de captură, analiză, stocare și raportare) și relațiile dintre acestea. Aceste diagrame permit evidențierea proprietăților și metodelor fiecărei componente și modul în care acestea interacționează;
- **Diagramele de activitate**, care oferă o reprezentare a fluxurilor operaționale din cadrul sistemului. Acestea pot descrie pașii necesari pentru captarea traficului, prelucrarea acestuia, detecția anomaliilor și generarea alertelor;
- **Diagramele de secvență**, care ilustrează schimburile de mesaje între obiecte sau module într-un scenariu concret, cum ar fi inițierea unei sesiuni de analiză sau salvarea automată a unui fișier cu date procesate.

Utilizarea acestor diagrame permite nu doar documentarea clară a arhitecturii, ci și validarea deciziilor de proiectare. Ele pot fi analizate pentru identificarea timpurie a eventualelor probleme de compatibilitate, redundanță sau lipsă de coerență logică între componente. În plus, diagramele UML pot sta la baza

planificării activităților de testare, contribuind la stabilirea unor scenarii relevante de verificare a funcționării sistemului.

De asemenea, modelarea și proiectarea favorizează o mai bună colaborare între dezvoltatori, specialiști în rețelistică și securitate, și alți factori implicați în procesul decizional. Prin definirea clară a responsabilităților fiecărui modul și a interfețelor de comunicare, se reduce semnificativ riscul apariției neînțelegerilor în timpul implementării.

Un alt beneficiu al acestei abordări este că oferă o bază solidă pentru extinderea ulterioară a aplicației. Pe măsură ce nevoile sistemului evoluează sau apar cerințe suplimentare, o arhitectură bine definită permite adăugarea de funcționalități fără a afecta negativ comportamentul general al aplicației.

2.1 Descrierea comportamentală a sistemului

Funcționarea unui sistem informatic dedicat interceptării și analizei traficului de rețea poate fi înțeleasă cel mai bine prin analiza succesiunii de acțiuni desfășurate în cadrul procesului de monitorizare. În centrul acestui comportament se află interacțiunile dintre modulele componente și fluxurile de date care leagă aceste module într-o structură logică coerentă. Aceste activități formează un circuit operațional care pornește de la captarea traficului brut și se încheie cu prezentarea rezultatelor către utilizator într-un format structurat și interpretabil.

Procesul debutează cu interceptarea pachetelor de date la nivelul interfeței de rețea. Sistemul este configurat astfel încât să poată prelua traficul în timp real, fie prin ascultare promiscuă, fie prin aplicarea unor filtre definite anterior. Această primă etapă presupune gestionarea unui flux constant de date, într-un mod care să nu perturbe performanța generală a rețelei. Imediat după preluare, pachetele sunt transmise către componentele care se ocupă cu analizarea și filtrarea acestora.

În etapa de analiză, pachetele sunt supuse unui proces de segmentare logică în funcție de caracteristici precum protocolul, adresa sursă sau destinație ori portul utilizat. Datele considerate nerelevante sunt trecute în arhivă, fără a fi analizate în detaliu, pentru a evita aglomerarea inutilă a sistemului. În schimb, informațiile care indică un potențial comportament neobișnuit sunt procesate în profunzime. Acest proces se bazează fie pe reguli prestabilite, asemănătoare celor utilizate de sistemele de tip Suricata, fie pe modele de analiză comportamentală, inspirate de abordarea adoptată de Zeek. Datele considerate importante sunt apoi transmise către un sistem de stocare, unde sunt reținute pentru examinări ulterioare sau pentru generarea de rapoarte.

Interacțiunea cu utilizatorul se realizează prin intermediul unei interfețe grafice care permite vizualizarea datelor prelucrate, filtrarea acestora și exportul în diverse formate. Această componentă oferă o punte între logica internă a sistemului și utilizatorul final, permițându-i acestuia să consulte activitatea rețelei și să intervină, dacă este necesar, asupra unor fluxuri considerate suspecte.

Pentru a oferi o imagine clară și standardizată a comportamentului descris mai sus, sunt utilizate instrumente grafice specifice modelării, precum diagramele UML. Diagramele de activitate permit evidențierea traseului parcurs de pachetele de date în cadrul sistemului, de la momentul captării până la clasificarea lor ca relevante sau neimportante. În completare, diagramele de secvență descriu interacțiunile dintre modulele principale, punând accent pe ordinea schimburilor de informații și pe modul în care acestea colaborează pentru a susține procesul de monitorizare.

Această abordare comportamentală permite nu doar înțelegerea modului în care funcționează aplicația, ci și anticiparea comportamentului acesteia în fața diverselor scenarii de trafic. Totodată, contribuie la verificarea coerenței arhitecturii propuse, oferind un punct de plecare solid pentru etapa următoare de implementare.

2.1.1 Imaginea generală asupra sistemului

Modul în care utilizatorul final interacționează cu aplicația dedicată interceptării și analizei traficului de rețea poate fi reprezentat vizual printr-o diagramă de caz de utilizare. În **Figura 2.1 – Interacțiune cu utilizatorul/analistul**, este ilustrat setul principal de acțiuni pe care un analist le poate executa în cadrul sistemului. Aceste funcționalități reflectă relația directă dintre utilizator și componentele aplicației și sunt necesare pentru a configura, monitoriza și evalua activitatea rețelei.

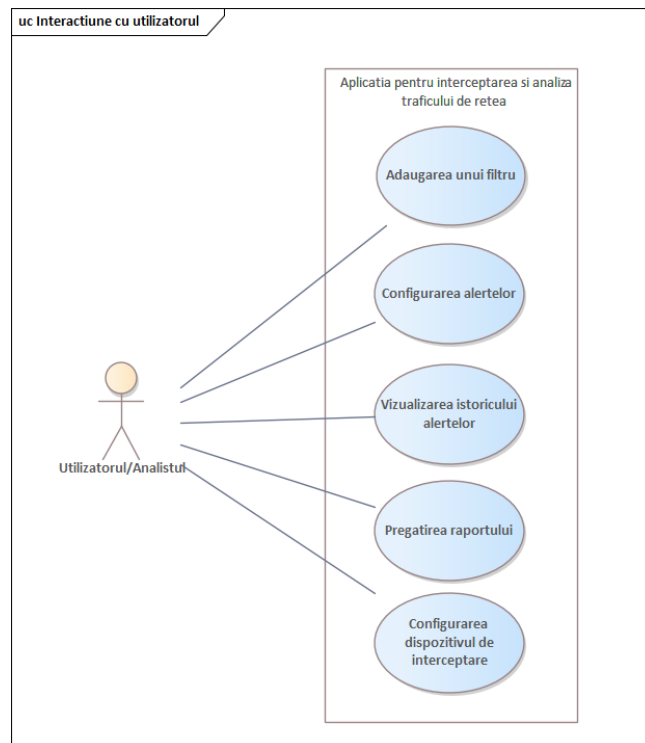


Figura 2.1 - Interacțiune cu utilizatorul/analistul

Una dintre opțiunile disponibile este **adăugarea unui filtru**, care permite selectarea traficului în funcție de criterii definite, precum adrese IP, protocoale utilizate sau porturi de destinație. Această funcție

contribuie la focalizarea procesului de captură asupra informațiilor relevante, reducând volumul de date inutile.

O altă funcționalitate importantă este **configurarea alertelor**, care oferă posibilitatea stabilirii unor condiții declanșatoare pentru semnalarea activităților anormale. Acest mecanism de alertare automată sprijină identificarea rapidă a incidentelor potențiale și îmbunătățește capacitatea de reacție a operatorului.

Vizualizarea istoricului alertelor permite revizuirea evenimentelor detectate anterior, oferind o imagine de ansamblu asupra comportamentului rețelei în timp și facilitând recunoașterea unor tipare recurente asociate cu riscuri sau abuzuri.

Prin pregătirea raportului, utilizatorul poate genera documente care sintetizează rezultatele sesiunilor de monitorizare. Aceste rapoarte pot conține statistici, diagrame și descrieri ale alertelor, fiind utile în contextul auditării, al raportării către management sau al investigațiilor ulterioare.

De asemenea, configurarea dispozitivului de interceptare oferă control asupra parametrilor tehnici ai capturii, cum ar fi interfața de rețea folosită, modul promiscuu de funcționare sau aplicarea de filtre la nivel hardware/software, asigurând astfel eficiența și acuratețea procesului de colectare a datelor.

Prin urmare, diagrama din Figura 2.1 oferă o perspectivă clară asupra punctelor de contact dintre utilizator și sistem, ilustrând modul în care analiza traficului este inițiată, controlată și documentată de către operator, în cadrul unei aplicații concepute pentru supraveghere și detecție.

2.1.2 Modelarea vizuală a fluxurilor

Pentru a înțelege funcționarea internă a sistemului informatic propus, este necesară reprezentarea grafică a modului în care datele sunt procesate în funcție de alegerile făcute la nivelul configurației inițiale. Un exemplu elocvent este oferit de Figura 2.2 – Procesul de interceptare, care prezintă o diagramă de activitate ce urmărește traseul logic parcurs de aplicație în procesul de captare și partajare a traficului de rețea.

Diagrama începe prin verificarea existenței unor filtre de captare definite de utilizator. În cazul în care aceste filtre sunt prezente, sistemul inițiază interceptarea doar a pachetelor care corespund criteriilor stabilite. Acest tip de filtrare direcționează captarea către fluxurile considerate relevante pentru analiza de securitate, reducând semnificativ volumul de date care necesită prelucrare ulterioară. Dacă nu au fost configurate filtre, sistemul trece automat la captarea integrală a traficului, asigurând astfel o acoperire completă a activității de rețea.

Ulterior, indiferent de tipul de captare inițial selectat, sistemul parcurge o a doua decizie logică, referitoare la aplicarea unei filtrări personalizate. Această etapă suplimentară permite rafinarea suplimentară a datelor, aplicând reguli mai detaliate în funcție de nevoile de investigare. Atunci când aceste reguli sunt active, traficul partajat este cel care respectă condițiile impuse, oferind un grad ridicat de

specificitate în analiza ulterioară. În lipsa unei astfel de filtrări, tot traficul capturat este partajat, păstrând caracterul complet al sesiunii de monitorizare.

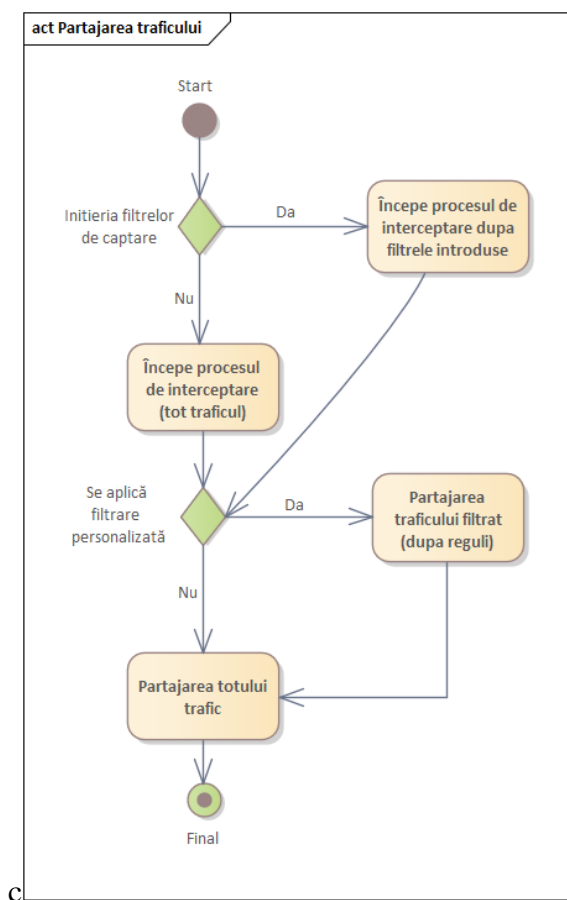


Figura 2.2 – Procesul de interceptare

Modelul prezentat în diagramă evidențiază flexibilitatea arhitecturii sistemului și permite o mai bună adaptare la diferite contexte operaționale. Alegerea între captarea completă și cea filtrată, urmată de opțiunea unei filtrări avansate, constituie o abordare care încurajează controlul progresiv asupra datelor procesate. În mod particular, această metodă este benefică în scenariile în care analiza trebuie să fie orientată fie pe trafic general, în scop de audit, fie pe fluxuri specifice, în cazul investigațiilor privind comportamente suspecte sau atacuri punctuale.

Prin intermediul acestei reprezentări grafice, este subliniată importanța definirii inițiale a regulilor de captare, precum și a capacității sistemului de a ajusta în mod dinamic volumul și tipul datelor analizate. Această abordare permite echipei de securitate să adapteze rapid setările aplicației la schimbările de context din rețea, fără a necesita intervenții complexe asupra structurii sistemului. În același timp, modelarea fluxurilor oferă o bază solidă pentru testare, întreținere și eventuală extindere a funcționalităților, sprijinind astfel un proces de dezvoltare sustenabil și bine documentat.

2.1.3 Descrierea scenariilor de utilizare a aplicației

Pentru a evidenția succesiunea logică a pașilor executați în cadrul aplicației și modul în care actorii interacționează cu componentele sistemului, este utilă reprezentarea procesului printr-o diagramă de secvență. În Figura 2.3 – Diagrama de secvență pentru configurarea, capturarea și analiza traficului de rețea, sunt ilustrate schimburile de mesaje dintre analist, sistem, componenta responsabilă de accesarea sursei de trafic și modulul de analiză.

Scenariul începe cu inițierea de către utilizator a procesului de configurare a filtrelor de capturare. Aceste filtre pot viza anumite protocoale, adrese IP sau alte criterii relevante în contextul supravegherii rețelei. Odată ce aceste setări sunt definite, sistemul validează configurația și trece la inițierea procesului de capturare a pachetelor.

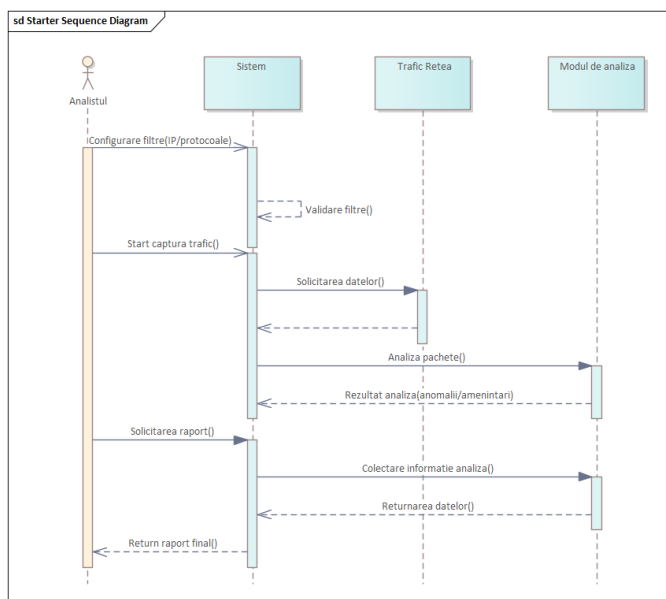


Figura 2.3 - Diagrama de secvență pentru configurarea, capturarea și analiza traficului de rețea

După pornirea capturii, sistemul formulează o solicitare către componenta care gestionează traficul de rețea, în vederea obținerii datelor corespunzătoare. Aceste date, odată colectate, sunt transmise către modulul de analiză, care le examinează pentru a identifica eventuale comportamente neobișnuite, semnale de alertă sau tentative de acces neautorizat. Rezultatul acestei analize este returnat sistemului, împreună cu informațiile relevante privind posibilele riscuri.

În etapa următoare, analistul solicită generarea unui raport, ceea ce determină sistemul să adune și să structureze datele obținute anterior. Modulul de analiză este implicat și în această etapă, furnizând date suplimentare sau agregate, care contribuie la realizarea unei sinteze finale a sesiunii de monitorizare. În cele din urmă, sistemul returnează raportul final, care este prezentat utilizatorului într-un format accesibil și interpretabil, contribuind la documentarea rezultatelor și la luarea unor decizii informate privind securitatea rețelei.

Structura acestei diagrame permite înțelegerea clară a fluxului de date și a rolului fiecărui component în cadrul sistemului. De asemenea, scoate în evidență caracterul secvențial și coordonat al acțiunilor, subliniind coeziunea dintre fazele de configurare, capturare, analiză și raportare. Acest mod de

reprezentare este util atât în documentarea arhitecturii aplicației, cât și în etapele ulterioare de testare și validare a funcționalităților implementate.

2.2 Descrierea structurală a sistemului

Structura unui sistem informatic destinat monitorizării și analizei traficului de rețea trebuie să fie bine definită și organizată astfel încât fiecare componentă să contribuie într-un mod coerent la atingerea scopurilor funcționale. În cazul proiectului de față, sistemul poate fi divizat în mai multe componente principale, fiecare având un rol specific și interacționând cu celelalte într-o manieră logică și previzibilă.

Componenta de **captură** reprezintă punctul de intrare a datelor în sistem și are ca sarcină interceptarea pachetelor de rețea în timp real. Aceasta trebuie să fie suficient de performantă pentru a gestiona volume mari de trafic, fără a pierde informații sau a produce întârzieri semnificative. Pe lângă funcția de colectare, modulul de captură oferă și suport pentru aplicarea filtrelor de selecție, permițând astfel extragerea numai a acelor fluxuri de date care corespund criteriilor definite de utilizator.

Odată ce datele au fost preluate, ele sunt transmise către **modulul de analiză**, responsabil pentru procesarea și interpretarea informațiilor colectate. Acest modul are rolul de a detecta comportamente neobișnuite, posibile amenințări sau abateri față de tiparele normale de trafic. Funcționalitățile sale pot include atât tehnici bazate pe semnături — care presupun compararea cu un set de reguli prestabilite — cât și tehnici de analiză comportamentală, care urmăresc identificarea anomaliilor prin observarea variațiilor în activitatea rețelei.

Pentru ca activitățile să poată fi revizuite, documentate și corelate în timp, sistemul include și o componentă de stocare, reprezentată de o bază de date sau un mecanism echivalent de logare. Aceasta reține informațiile relevante privind sesiunile de trafic, alertele generate, filtrele aplicate, precum și rapoartele produse. Accesul rapid la aceste date este important pentru investigarea ulterioară a evenimentelor și pentru realizarea unor audituri retrospective.

Legătura dintre utilizator și logica internă a aplicației este realizată prin intermediul **interfeței grafice**, care permite configurarea setărilor, inițierea procesului de captură, monitorizarea alertelor și generarea de rapoarte. Această interfață este concepută astfel încât să faciliteze utilizarea intuitivă, reducând curba de învățare și oferind un nivel ridicat de vizibilitate asupra activității desfășurate în cadrul rețelei. Prin intermediul acesteia, utilizatorii pot adapta sistemul la contextul operațional curent și pot reacționa rapid la semnalele generate de procesul de analiză.

Din punct de vedere logic, componentele descrise mai sus sunt interconectate într-o arhitectură modulară, în care fiecare parte contribuie la funcționarea generală a sistemului fără a depinde excesiv de celelalte. Această separare a responsabilităților permite atât dezvoltarea independentă a fiecărei componente, cât și întreținerea sistemului în timp. Comunicarea între module se realizează prin protocoale

bine definite sau prin interfețe programatice, ceea ce asigură flexibilitate în cazul în care se dorește extinderea sau înlocuirea unei părți a aplicației.

Pentru a documenta această structură, se utilizează frecvent instrumente vizuale de modelare, precum diagramele de componente sau diagramele de clasă din cadrul limbajului UML. Acestea permit descrierea arhitecturii dintr-o perspectivă abstractă, ușor de înțeles și interpretat atât de către dezvoltatori, cât și de către alți actori implicați în proiect, precum analiștii de securitate sau personalul tehnic de suport.

2.2.1 Descrierea structurii statice a sistemului

Pentru a înțelege organizarea internă a componentelor aplicației, este necesar să se analizeze structura statică a sistemului. Aceasta poate fi exprimată eficient printr-o diagramă de clasă UML, care descrie entitățile fundamentale, atributele și metodele aferente, precum și relațiile dintre aceste entități. În Figura 2.4 – Diagrama de clasă a sistemului de interceptare și analiză a traficului, este ilustrată arhitectura logică a sistemului sub forma unui model orientat pe obiecte.

Clasa Utilizator este responsabilă de autentificarea în sistem, configurarea filtrelor pentru captarea traficului și inițierea generării rapoartelor. Această clasă reprezintă interfața dintre operatorul uman și componentele aplicației, asigurând controlul și personalizarea procesului de monitorizare. Atribute precum ID, nume sau rol permit identificarea și diferențierea utilizatorilor, iar metodele disponibile permit realizarea acțiunilor principale asupra sistemului.

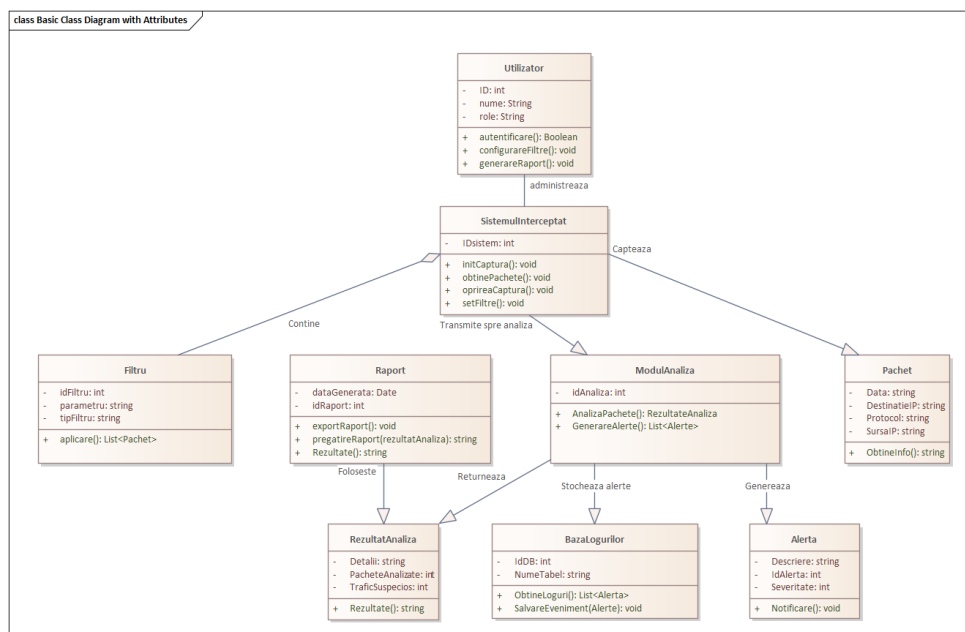


Figura 2.4 - Diagrama de clasă a sistemului de interceptare și analiză a traficului.

Componenta centrală a aplicației este reprezentată de clasa SistemInterceptat, care intermediază comunicarea dintre utilizator și procesele tehnice de interceptare. Această clasă cuprinde metodele de inițializare și oprire a capturii, precum și funcționalitatea de aplicare a filtrelor configurate. Ea joacă un rol cheie în transmiterea pachetelor captate către componentele de analiză.

Datele colectate din rețea sunt modelate prin clasa *Pachet*, care conține informații despre sursă, destinație, protocol și alte metadate asociate. Aceste instanțe sunt procesate ulterior de *ModulAnaliza*, componentă specializată în examinarea conținutului captat pentru identificarea unor comportamente neobișnuite. Modulul de analiză conține metode prin care se efectuează analiza propriu-zisă a pachetelor și se generează alerte atunci când sunt detectate amenințări sau abateri de la comportamentul normal.

Rezultatul procesului analitic este agregat în obiecte de tip *RezultatAnaliza*, care sintetizează datele obținute: numărul de pachete analizate, descrierea anomaliilor și volumul traficului suspect. Aceste rezultate pot fi transmise clasei *Raport*, responsabilă de generarea documentației aferente fiecărei sesiuni de monitorizare. Clasa respectivă include metode pentru exportul și pregătirea conținutului într-un format accesibil și utilizabil de către specialiști.

Un alt element important în structură este clasa *Filtru*, care determină criteriile de selecție a pachetelor procesate. Prin parametri precum adresa IP, tipul protocolului sau portul utilizat, această componentă contribuie la focalizarea procesului de captare asupra unor zone de interes și la reducerea volumului de date irelevante.

Pentru păstrarea informațiilor referitoare la alertele generate și la activitățile sistemului, se utilizează clasa *BazaLogurilor*. Aceasta asigură înregistrarea și accesarea ulterioară a evenimentelor, oferind un istoric valoros pentru audituri de securitate sau investigații. Evenimentele sunt stocate alături de datele asociate, iar metoda *SalvareEveniment()* permite persistarea acestora în mod controlat.

Clasa *Alerta* este generată în momentul în care sistemul detectează o amenințare. Aceasta cuprinde informații precum descrierea alertei, severitatea și identificatorul unic, fiind asociată automat cu evenimentul detectat. Sistemul poate notifica operatorul în baza acestor alerte, ceea ce sprijină luarea rapidă a unor măsuri de răspuns.

2.2.2 Relatiile de dependență între componentele sistemului

O componentă importantă în proiectarea unui sistem informatic o reprezintă modelarea stărilor interne prin care acesta trece în funcție de acțiunile utilizatorului și condițiile de execuție. Un exemplu relevant în acest sens este procesul de autentificare și autorizare, care determină accesul la funcționalitățile aplicației și stabilește drepturile de utilizare pentru fiecare categorie de utilizator.

În Figura 2.5 – Diagrama de stare pentru procesul de logare, este prezentat modul în care sistemul evoluează din momentul în care utilizatorul inițiază accesul până la finalizarea procesului de autentificare și tranziția către zona de funcționare a aplicației. Diagrama este structurată sub forma unei mașini de stare UML, care include atât stări simple, cât și un stat compozit, reflectând structura internă a sistemului în timpul autentificării.

Procesul debutează dintr-un nod de început, moment în care aplicația direcționează utilizatorul către interfața de logare. Aici, sistemul așteaptă introducerea unor date de identificare, precum nume de utilizator

și parolă. Această fază este encapsulată într-un stat compozit intitulat „Sistemul”, care cuprinde mai multe stări intermediare. După ce utilizatorul completează câmpurile de autentificare și inițiază procesul printr-un buton dedicat, sistemul trece în starea de verificare a datelor introduse.

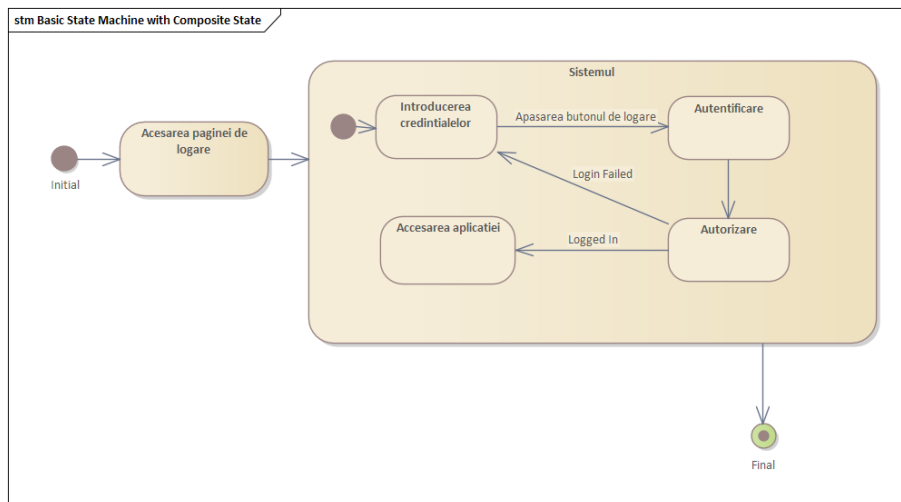


Figura 2.5 - Diagrama de stare pentru procesul de logare

În cazul în care informațiile furnizate nu corespund unor înregistrări valide, sistemul generează o tranziție către starea anterioară, semnalizată printr-un eveniment de tip „Login Failed”. Această revenire permite utilizatorului să încerce din nou, fără a părăsi cadrul aplicației. Dacă autentificarea este reușită, se inițiază automat procesul de autorizare, prin care se verifică nivelul de acces atribuit utilizatorului. Această etapă este necesară pentru a determina ce componente ale aplicației pot fi accesate și ce acțiuni pot fi efectuate.

Finalizarea cu succes a procesului de autorizare conduce la ieșirea din statul compozit și la trecerea într-o nouă stare activă, care corespunde utilizării normale a aplicației. În această etapă, utilizatorul are acces la funcțiile disponibile conform rolului său (de exemplu, vizualizarea rapoartelor, configurarea filtrelor, inițierea capturii de trafic etc.). Diagrama se încheie cu un nod final, care marchează încheierea procesului de autentificare și trecerea completă în mediul operațional al sistemului.

Această reprezentare evidențiază interdependențele dintre componentele de autentificare și celelalte module ale sistemului, precum și condițiile care trebuie îndeplinite pentru a permite accesul. Prin utilizarea stărilor și tranzițiilor, este oferită o imagine clară asupra modului în care sistemul reacționează la acțiunile utilizatorului și asupra logicii interne de control a accesului.

Modelarea procesului de logare în acest mod contribuie nu doar la înțelegerea mecanismului de control al accesului, ci și la identificarea punctelor în care pot fi aplicate politici de securitate, optimizări sau mecanisme suplimentare de verificare, cum ar fi autentificarea în doi pași sau înregistrarea tentativelor nereușite. În plus, această abordare poate sta la baza testării funcționale, asigurând că sistemul reacționează corect în toate scenariile posibile legate de identificarea utilizatorilor.

3 REALIZAREA SISTEMULUI

Procesul de construire a aplicației pentru interceptarea și analiza traficului de rețea a presupus parcurgerea mai multor etape succesive, de la proiectarea arhitecturii generale și definirea funcționalităților, până la implementarea concretă a componentelor și integrarea acestora într-un ansamblu coerent. Acest capitol prezintă în mod detaliat modul în care a fost realizată aplicația, punând accent pe organizarea modulară, alegerea tehnologiilor, structura internă și fluxul de funcționare.

Aplicația a fost concepută pentru a permite monitorizarea traficului din rețele locale, oferind utilizatorului posibilitatea de a vizualiza în timp real informații despre pachetele capturate. Aceasta este alcătuită din trei componente principale: motorul de captură și analiză a traficului, interfața grafică de utilizator (GUI) și un modul de stocare locală a datelor. Toate aceste componente au fost dezvoltate utilizând limbajul Python, datorită portabilității, suportului extins pentru biblioteci externe și ușurinței în implementarea funcțiilor specifice de rețea.

Pentru partea de captură a traficului, au fost utilizate bibliotecile `scapy`[3],[8] și `socket`, care permit interceptarea pachetelor la nivelul rețelei și extragerea caracteristicilor de bază ale acestora (IP sursă/destinație, porturi, protocoale, timestamps). Informațiile obținute sunt prelucrate și afișate prin intermediul unei interfețe grafice create cu Tkinter, organizată într-un mod intuitiv, sub forma unei ferestre de autentificare stilizate, însoțită de un panou principal de control și afișare a traficului.

Datele rezultate din sesiunile de monitorizare sunt salvate local într-o bază de date, permițând atât consultarea ulterioară, cât și exportul în formate standard, cum ar fi `.pcap` sau `.txt`. Acest mecanism oferă persistență și sprijină procesul de analiză post-eveniment, contribuind la o evaluare mai detaliată a comportamentului rețelei.

Aplicația a fost construită într-un mod flexibil și modular, astfel încât fiecare componentă să poată fi întreținută sau extinsă separat. În cadrul acestui capitol sunt prezentate în detaliu structura proiectului, modul de organizare a codului, integrarea cu servicii externe, și modul în care cerințele funcționale au fost transpuse în implementare concretă.

3.1 Structura sistemului

Aplicația „Sentinel Traffic Analyzer” a fost dezvoltată într-un mod modular, respectând o arhitectură clară și extensibilă. Codul sursă este organizat în mai multe fișiere Python, fiecare responsabil de un set specific de funcționalități. Această abordare permite întreținerea eficientă a aplicației și adăugarea ulterioară a unor noi componente fără a afecta funcționarea generală.

Captura traficului de rețea este realizată prin intermediul clasei `TrafficAnalyzerCore` din fișierul `core.py`. Metoda `start_capture()` utilizează biblioteca `scapy` pentru a intercepta pachetele în timp real, după cum se observă în fragmentul următor:

```
# core.py
def start_capture(self, iface, callback, bpf_filter=None):
    self.stop_sniff = False
    sniff(
        iface=iface,
        prn=lambda pkt: self._handle_packet(pkt, callback),
        stop_filter=lambda x: self.stop_sniff,
        filter=bpf_filter if bpf_filter else None)

```

După captură, pachetele sunt procesate intern de metoda `_handle_packet()`, care extrage date precum IP sursă, destinație, protocol, și le trimite spre interfața grafică:

```
# core.py
def _handle_packet(self, packet, callback):
    self.captured_packets.append(packet)
    callback(self.format_packet(packet))

```

Interfața grafică (GUI), implementată în fișierul `gui.py`, este construită folosind Tkinter și oferă utilizatorului funcționalități complete: pornirea și oprirea capturii, alegerea interfeței de rețea, salvarea sesiunilor, încărcarea fișierelor .pcap, generarea rapoartelor și vizualizarea alertelor. Tkinter a fost ales datorită integrării native cu Python, a simplității în utilizare și a documentației accesibile, conform [11].

După pornirea capturii, datele interceptate sunt afișate în timp real într-un widget de tip text, gestionat de clasa `PacketDisplay` din `packet_display.py`:

```
# packet_display.py
def write(self, line, packet_obj=None):
    self.packets.append(line)
    self.text_widget.insert(tk.END, line)
    self.text_widget.see(tk.END)

```

Toate datele capturate sunt păstrate într-o structură proprie de raportare (`ReportData` din `report_generate.py`), care permite generarea de fișiere PDF și JSON cu informații detaliate despre sesiunea analizată.

```
# report_generate.py
def export_pdf_report(self, path="raport_final.pdf"):
    self.generate_graphs()
    pdf.output(path)

```

3.2 Integrarea serviciilor externe

Pentru a oferi o funcționalitate extinsă și pentru a îmbunătăți experiența utilizatorului, aplicația *Sentinel Traffic Analyzer* integrează mai multe servicii și biblioteci externe care contribuie semnificativ la precizia și ușurința în utilizare a platformei.

Una dintre cele mai importante integrări este utilizarea bazei de date **GeoLite2-City.mmdb**, furnizată de MaxMind. Aceasta este o bază de date geografică care permite maparea adreselor IP către locații geografice estimative, incluzând informații precum țara, orașul, regiunea, coordonatele GPS și zona de timp.

Integrarea cu GeoLite2 este realizată în fișierul `network_discovery_view.py`, în cadrul metodei `load_geoip_data()`. Această metodă utilizează biblioteca `geoip2.database.Reader` pentru a accesa fișierul

.mmdb și pentru a parcurge toate adresele IP detectate în cadrul unei sesiuni de analiză. Informațiile obținute sunt apoi atașate fiecărui IP și utilizate ulterior în vizualizarea geografică a rețelei, precum și în tabelul de hosturi detectate.

```
reader = geoip2.database.Reader("PacketSentinel/GeoLite2-City.mmdb")
all_ips = set(report_data.sources.keys()) | set(report_data.destinations.keys())
for ip in all_ips:
    response = reader.city(ip)
    country = response.country.name
    city = response.city.name
    lat = response.location.latitude
    lon = response.location.longitude
```

Această integrare permite afișarea automată a locației IP-urilor implicate în sesiunile de trafic, oferind analistului o perspectivă suplimentară asupra distribuției geografice a dispozitivelor sau serverelor implicate în comunicare. În cazul în care o adresă IP nu poate fi localizată, sistemul completează automat cu valori „N/A”, asigurând astfel continuitatea afișării fără a întrerupe funcționarea aplicației.

Datele geografice sunt afișate într-un tab dedicat al ferestrei de descoperire a rețelei („Hartă Rețea”), unde IP-urile sunt reprezentate grafic ca noduri interconectate. Fiecare nod include, alături de IP, țara și orașul corespunzătoare, precum și – dacă este disponibil – organizația asociată adresei IP:

```
labels[node] = "\n".join(filter(None, [
    node,
    f"{loc.get('country', '')}, {loc.get('city', '')}",
    f"{loc.get('org')} " if loc.get('org', '') != "N/A" else ""
]))
```

Această reprezentare ajută la identificarea rapidă a legăturilor geografice între gazdele detectate și oferă un punct de plecare valoros în investigarea sursei unor activități suspecte.

3.3 Implementarea cerințelor funcționale

Pentru ca aplicația Sentinel Traffic Analyzer să răspundă obiectivelor propuse, s-a urmărit transpunerea concretă a unui set de cerințe funcționale în funcționalități directe, accesibile utilizatorului. Fiecare dintre aceste cerințe a fost abordată în mod practic, cu scopul de a asigura controlul complet asupra unei sesiuni de monitorizare, de la inițiere până la generarea de rapoarte și salvarea rezultatelor.

Un prim aspect urmărit a fost **captura traficului în timp real**, care se realizează la nivelul unei interfețe de rețea selectate de utilizator. Captura este gestionată asincron, ceea ce permite aplicației să funcționeze fluent indiferent de volumul de date. Procesul poate fi întrerupt și reluat fără pierderi, iar selecția traficului poate fi restrânsă prin aplicarea de filtre BPF.

Vizualizarea interactivă a datelor capturate a fost implementată pentru a facilita analiza în timp real. Pachetele interceptate sunt afișate progresiv într-un tabel grafic, cu actualizare dinamică, iar informațiile afișate sunt prezentate într-un format condensat, cu evidențierea protocoalelor. Acest mecanism permite utilizatorului să observe imediat variații în comportamentul rețelei sau prezența unor fluxuri neobișnuite.

Un element important îl constituie **sistemul de alertare**, activ automat în fundal. Acesta analizează traficul în mod continuu și declanșează notificări în cazul identificării unor modele care pot indica activități potențial periculoase. Printre aceste modele se regăsesc scanări de porturi, flood-uri de diverse tipuri, spoofing ARP[7] și trafic provenind de la IP-uri marcate ca periculoase. Alerte sunt prezentate vizual în interfață și salvate în fișiere de tip .log și .json.

O altă funcționalitate implementată este gestionarea sesiunilor de captură și exportul datelor. Utilizatorul poate salva sesiunile în format .pcap, utilizabil ulterior în instrumente de analiză precum Wireshark, dar și în format .txt sau .json, pentru prelucrare personalizată. La finalul fiecărei sesiuni, aplicația poate genera automat un raport în format PDF, care include grafice și sinteze ale sesiunii curente.

Aplicația include și o serie de **verificări și măsuri de protecție** în interfața grafică, care previn apariția unor erori de utilizare. De exemplu, nu este permisă pornirea capturii fără selectarea unei interfețe valide, iar dacă utilizatorul nu are drepturi suficiente, aplicația notifică acest lucru în mod explicit. Aceste mecanisme contribuie la fiabilitatea aplicației și reduc riscul de funcționare defectuoasă în condiții necontrolate.

3.4 Arhitectura modulară a aplicației

Aplicația Sentinel Traffic Analyzer a fost construită pe baza unui model arhitectural modular, în care fiecare componentă îndeplinește un rol distinct și interacționează cu celelalte prin mecanisme bine delimitate. Această abordare a permis o dezvoltare controlată și organizată, precum și posibilitatea extinderii funcționalităților fără a afecta stabilitatea aplicației.

Structura modulară este vizibilă în separarea logică a funcțiilor principale. Captura traficului de rețea este gestionată de un modul dedicat, care preia pachetele transmise prin interfața selectată, le interpretează și le transmite mai departe spre analiză și afișare. Procesarea acestora se desfășoară în paralel cu activitatea interfeței grafice, evitând astfel blocarea aplicației în timpul capturii.

Analiza comportamentală a traficului se realizează într-un modul distinct, care primește pachetele capturate și aplică asupra lor reguli de identificare a anomaliilor. Dacă sunt detectate comportamente suspecte, acestea sunt înregistrate sub formă de alerte și pot fi vizualizate în timp real, precum și stocate pentru analiză ulterioară. Interfața grafică joacă rolul de punct de interacțiune între utilizator și logica aplicației, permițând configurarea procesului de captură, declanșarea acțiunilor de export sau generare de rapoarte și monitorizarea în timp real a traficului.

Pe lângă componenta vizuală principală, aplicația conține un modul specializat pentru afișarea pachetelor interceptate, într-o formă structurată și lizibilă. Acesta menține o listă a datelor capturate și permite operațiuni precum marcare, comentarii sau salvarea individuală a pachetelor. Sistemul de raportare funcționează în strânsă legătură cu motorul de captură și cu baza de date, extrăgând informațiile relevante

și prezentându-le sub formă de documente PDF sau fișiere JSON, alături de reprezentări grafice generate automat.

Pentru a facilita procesul de selecție a sursei de trafic, aplicația include și o componentă de identificare a interfețelor de rețea, care extrage denumirile reale ale acestora, îmbunătățind astfel experiența de utilizare. Persistența datelor este asigurată printr-un sistem de stocare local, care permite salvarea sesiunilor și alertelor într-o bază de date, structurată astfel încât să poată fi interogată ulterior în funcție de mai mulți parametri.

Prin această distribuție clară a responsabilităților între componente, arhitectura aplicației reușește să asigure echilibrul între flexibilitate, performanță și mentenabilitate. Fiecare modul poate fi modificat, testat sau extins în mod independent, fără a afecta funcționarea celorlalte părți ale aplicației. Astfel, *Sentinel Traffic Analyzer* devine un cadru robust și extensibil, capabil să răspundă eficient cerințelor actuale și viitoare din domeniul analizei traficului de rețea.

3.5 Modelul de date și structura bazei de date

Persistența datelor în cadrul aplicației Sentinel Traffic Analyzer este asigurată printr-un sistem de stocare care combină două componente principale: o bază de date locală, utilizată pentru gestionarea sesiunilor, pachetelor și alertelor, și un serviciu API extern, prin care fișierele .pcap sunt salvate fizic și înregistrate într-o bază de date relațională (PostgreSQL).

Aplicația locală stochează temporar datele capturate în fișiere de tip .pcap, precum și metadatele asociate acestora (timestamp, interfață utilizată, durată, volum de trafic). Aceste date sunt ulterior disponibile pentru analiză post-eveniment, fie prin interfața grafică, fie prin exportul în formate standardizate. Pentru o utilizare avansată și centralizată, aplicația oferă și posibilitatea de a trimite aceste fișiere către un API extern, unde sunt gestionate în mod persistent.

Componenta de backend este realizată folosind frameworkul Flask și oferă un punct de acces HTTP la ruta /upload, unde fișierele .pcap sunt transmise prin metode POST. La primirea unui fișier valid, acesta este salvat local într-un director dedicat pe server (/server_storage/pcaps), iar referința către fișierul salvat este înregistrată într-o bază de date PostgreSQL. Acest proces asigură atât păstrarea fișierelor brute, cât și posibilitatea de a le identifica și regăsi prin interogări directe în baza de date.

Conexiunea cu baza de date este gestionată prin modulul psycopg2, iar parametrii de autentificare sunt încărcăți dintr-un fișier .env, pentru a păstra securitatea datelor de acces. Structura tabelului pcap_metadata este simplă și eficientă, incluzând câmpuri precum filename (numele generat al fișierului) și filepath (calea completă de stocare pe server).

Fragmentul de cod de mai jos ilustrează acest proces:

```
@app.route("/upload", methods=["POST"])
def upload_file():
    file = request.files['file']
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
```



```

filename = f"capture_{timestamp}.pcap"
filepath = os.path.join(UPLOAD_FOLDER, filename)
file.save(filepath)

conn = get_db_connection()
cur = conn.cursor()
cur.execute(
    "INSERT INTO pcap_metadata (filename, filepath) VALUES (%s, %s)",
    (filename, filepath)
)
conn.commit()

```

Această abordare permite scalarea aplicației pentru utilizare în rețele mai mari sau în scenarii distribuite, unde mai mulți clienți pot încărca capturi spre un punct centralizat de stocare și analiză. În același timp, sistemul rămâne compatibil cu rularea locală, pentru cazurile în care rețelele nu permit acces extern sau se dorește rularea aplicației în mod izolat.

În ceea ce privește modelul de date local utilizat în aplicație, structura este orientată pe sesiuni. Fiecare sesiune de monitorizare este identificată printr-un ID unic și conține o listă de pachete capturate, salvate în structuri serializabile. Alertele generate în timpul sesiunii sunt asociate cu aceste capturi și stocate în fișiere .log și .json, iar datele semnificative extrase din trafic pot fi interogate pe baza unor filtre precum IP sursă, porturi sau protocoale.

Prin această combinație între stocarea locală și integrarea cu servicii externe, aplicația oferă un sistem de persistență flexibil, sigur și scalabil, care răspunde cerințelor de audit, analiză și arhivare a traficului monitorizat.

3.6 Fluxul de execuție în aplicație

Funcționarea aplicației Sentinel Traffic Analyzer urmează un flux de execuție bine definit, care acoperă toate etapele de utilizare, de la inițializarea interfeței grafice până la încheierea unei sesiuni și generarea unui raport. Acest flux este organizat astfel încât utilizatorul să poată interacționa cu sistemul într-un mod intuitiv, iar toate acțiunile efectuate să se reflecte în mod direct asupra modului de procesare și afișare a traficului de rețea.

La lansarea aplicației, interfața grafică este inițializată și utilizatorului i se prezintă o fereastră principală în care sunt încărcate componentele de control, afișare și configurare. Printre primele acțiuni necesare se numără selectarea unei interfețe de rețea, care este identificată automat de un utilitar intern care utilizează WMI pentru a extrage numele reale ale plăcilor de rețea disponibile în sistem. Această selecție este fundamentală pentru inițierea unei sesiuni de captură.

Odată aleasă interfața, utilizatorul poate lansa procesul de captură. Acesta este pornit prin intermediul unui buton din interfață, care declanșează în fundal activarea modului de captură. Captura traficului se desfășoară asincron, astfel încât interfața să rămână responsivă. Pe măsură ce pachetele sunt interceptate, acestea sunt procesate în timp real și afișate în zona de vizualizare, într-un tabel cu actualizare progresivă.

În paralel, fiecare pachet procesat este transmis către modulul de analiză comportamentală. Acesta verifică pachetele conform unor reguli prestabilite, detectând activități suspecte precum scanări de porturi sau tentative de flood. Pentru fiecare pachet, sunt extrase câmpuri esențiale precum IP sursă, destinație, porturi și protocol, **conform specificațiilor definite în RFC 791 pentru Internet Protocol [9]**. În cazul identificării unui eveniment neobișnuit, sistemul generează automat o alertă, care este înregistrată și afișată în interfața grafică, precum și salvată pentru referință ulterioară. Pe tot parcursul sesiunii, utilizatorul are posibilitatea de a opri temporar capturarea, de a salva traficul sub formă de fișier .pcap și de a efectua filtrări după diverse criterii. La finalul unei sesiuni, toate datele pot fi exportate în diferite formate, iar aplicația oferă posibilitatea de a genera un raport automat. Acest raport conține statistici despre numărul de pachete, tipurile de protocoale întâlnite, alertele generate și o hartă a rețelei detectate, inclusiv cu locații geografice asociate IP-urilor (prin integrarea GeoLite2).

În cazul în care fișierul rezultat este considerat relevant pentru analiză centralizată sau pentru arhivare într-un sistem extern, utilizatorul poate transmite fișierul .pcap către o interfață API. Acest serviciu salvează fișierul pe server și înregistrează metadatele într-o bază de date PostgreSQL, oferind astfel un mecanism suplimentar de stocare și consultare ulterioară.

Întregul flux este însoțit de mecanisme de validare și notificare, care previn acțiunile eronate (de exemplu, pornirea capturii fără interfață selectată) și oferă feedback în timp real cu privire la starea operațiilor efectuate. Finalizarea unei sesiuni este marcată de oprirea capturii, salvarea datelor și revenirea aplicației într-o stare inactivă, în așteptarea unei noi comenzi.

Prin acest flux bine definit, aplicația permite controlul complet asupra unei sesiuni de analiză de rețea, punând la dispoziția utilizatorului instrumentele necesare pentru captură, filtrare, analiză, alertare și raportare, într-un mod unitar și eficient.

3.7 Sistemul de alertare și detectare a comportamentelor suspecte

Un element central al aplicației Sentinel Traffic Analyzer este capacitatea de a detecta în timp real comportamente de rețea neobișnuite, asociate potențial cu activități periculoase sau malițioase. Acest mecanism este implementat într-un modul dedicat, `detector.py`, care rulează în fundal în paralel cu procesul de captură și analiză a traficului. Funcția principală a acestuia este de a analiza fiecare pachet interceptat și de a semnaliza activități suspecte prin generarea de alerte.

La inițializare, detectorul creează mai multe structuri interne pentru urmărirea comportamentului IP-urilor din trafic. Aceste structuri folosesc colecții de tip `defaultdict`, unde se păstrează înregistrări temporale pentru fiecare activitate detectată. Această abordare permite o analiză în timp scurt (`sliding window`), fără stocarea persistentă a întregului istoric.

Fiecare linie analizată este procesată de metoda `process_packet()`, care extrage adresa IP sursă și eventual portul de destinație. Pe baza acestor date, sunt aplicate mai multe reguli de detecție care verifică:

- Trafic provenit de la adrese IP aflate pe o listă neagră (BLACKLIST). În acest caz, alerta este imediat marcată cu severitate maximă;
- Scanarea de porturi, identificată atunci când un IP accesează mai mult de zece porturi diferite într-un interval scurt;
- Atacuri de tip DNS flood, în care sunt detectate un număr mare de cereri către portul 53 într-o perioadă restrânsă;
- SYN flood, semnalat printr-un volum ridicat de pachete care conțin flag-ul SYN, dar nu și ACK[10fvf];
- ICMP flood, determinat pe baza numărului de mesaje ICMP primite de la un IP în cinci secunde.
- UDP flood, similar ca structură, dar axat pe traficul UDP;
- ARP spoofing, detectat prin analizarea intensității mesajelor ARP provenite de la o singură sursă.

Dacă oricare dintre aceste condiții este îndeplinită, se generează o alertă sub forma unui mesaj text și a unui nivel de severitate. Alerta este apoi înregistrată simultan în două fișiere: unul de tip .log și altul de tip .json. Această dublă salvare are rolul de a permite atât o inspecție rapidă a jurnalului într-un editor text, cât și o analiză automată ulterioară pe baza formatului structurat JSON.

Mecanismul de înregistrare este implementat în metoda `log_alert()`. Aceasta creează automat câte un fișier de alertă pentru fiecare zi de funcționare, utilizând ca denumire data curentă. Astfel, alertele pot fi revizuite și organizate cronologic, iar sistemul poate genera statistici privind frecvența și severitatea incidentelor de rețea.

Prin urmare, modulul `detector.py` contribuie la transformarea aplicației într-un instrument de detecție activă, capabil să identifice în mod automat semne ale unor atacuri sau disfuncționalități în rețea. Acest subsistem nu doar completează funcționalitatea de monitorizare pasivă, ci oferă și o bază pentru implementarea unor reacții automate în viitor, prin corelarea alertelor cu acțiuni de blocare, notificare sau escaladare.

3.8 Generarea de rapoarte și vizualizări

Pentru a oferi o imagine de ansamblu asupra traficului capturat și pentru a sprijini procesul de analiză post-eveniment, aplicația *Sentinel Traffic Analyzer* integrează un modul dedicat pentru generarea de rapoarte și vizualizări grafice. Acest modul, implementat în cadrul clasei `ReportData`, centralizează informațiile colectate în timpul sesiunilor de monitorizare și le transformă într-un format accesibil, lizibil și ușor de distribuit.

Pe durata unei sesiuni active, toate pachetele capturate sunt înregistrate împreună cu metadatele relevante, cum ar fi momentul capturii, IP-ul sursă, IP-ul destinație, protocolul utilizat sau numărul total de pachete. Modulul colectează în paralel informații despre frecvența protocoalelor, distribuția traficului pe IP-uri, alertele generate de modulul de detecție și volumul de trafic în timp. Aceste date sunt stocate în

structuri optimizate, cu acces rapid, și pot fi convertite într-un format JSON pentru procesare ulterioară automată sau stocare.

În vederea generării de vizualizări grafice, ReportData utilizează bibliotecile matplotlib și networkx[3]. Se creează astfel grafice de tip bară, care reflectă distribuția protocoalelor și volumul de trafic per IP, grafice liniare pentru evoluția traficului în timp și diagrame circulare care ilustrează proporția alertelor pe categorii. Pe lângă aceste elemente statistice, modulul construiește și o hartă a rețelei, în care IP-urile detectate sunt reprezentate ca noduri conectate în funcție de legăturile observate între ele. Această hartă oferă o perspectivă vizuală utilă pentru înțelegerea topologiei rețelei monitorizate.

Funcția principală de export este export_pdf_report, care generează un document PDF cuprinzător. Acesta include, într-o primă parte, un rezumat al sesiunii: numărul total de pachete, momentul de început și sfârșit al capturii, protocoalele utilizate, cele mai frecvente adrese IP sursă și destinație, precum și lista alertelor generate. Informațiile textuale sunt organizate clar, în ordine logică, cu fonturi și formatare specifice, asigurând astfel o prezentare profesională a datelor.

Ulterior, raportul PDF integrează imaginile grafice generate anterior. Acestea sunt inserate automat, pe pagini separate, și includ graficele statistice și harta de rețea. Fișierele grafice sunt selectate din directorul curent, iar denumirile acestora sunt validate pentru a evita introducerea de fișiere lipsă. Pentru a garanta compatibilitatea și lizibilitatea textului în cadrul PDF-ului, este aplicată o funcție de transliterare care înlocuiește caracterele specifice limbii române cu echivalente ASCII.

Raportul final astfel generat poate fi exportat în format .pdf și salvat local sau transmis către un serviciu extern, împreună cu fișierul brut .pcap, pentru arhivare sau audit. Procesul de generare este complet automatizat, astfel încât, odată finalizată sesiunea de monitorizare, utilizatorul să poată obține rapid o sinteză completă a activității de rețea analizate.

Acest sistem de raportare contribuie la extinderea funcționalității aplicației dincolo de analiza în timp real, adăugând o componentă importantă de informare, comunicare și analiză retrospectivă. Prin combinarea datelor brute cu reprezentări grafice intuitive și conținut textual bine structurat, rapoartele produse devin un instrument util atât pentru utilizatorii tehnici, cât și pentru factorii de decizie care evaluează starea rețelei sau investighează incidente de securitate.

3.9 Implementarea funcției Follow Stream pentru analiza comunicării în rețea

În cadrul dezvoltării aplicației *Sentinel Traffic Analyzer*, una dintre cerințele importante a fost oferirea unei funcționalități care să permită analiza logică și secvențială a pachetelor aparținând aceluiași flux[13] de rețea. Pentru aceasta, a fost implementată funcția **Follow Stream**, inspirată din abordări consacrate în aplicații precum Wireshark, dar adaptată arhitecturii și designului propriu aplicației.

Funcționalitatea a fost integrată în clasa PacketDisplay, care are responsabilitatea de a gestiona afișarea pachetelor capturate și interacțiunea utilizatorului cu acestea. Codul responsabil de această funcție

este definit în metoda `follow_stream(self)`, iar logica de declanșare este atașată la meniul contextual activat la clic dreapta pe un pachet (metoda `on_right_click(self, event)`).

Procesul de urmărire a fluxului începe cu identificarea pachetului selectat de către utilizator. În acest scop, aplicația determină poziția exactă în lista de pachete folosind coordonatele cursorului (`event.x`, `event.y`) și setează `self.selected_index`. În momentul în care utilizatorul alege opțiunea „Urmărește fluxul” din meniu, este apelată metoda `follow_stream`.

Această metodă verifică mai întâi existența pachetului selectat și a structurii `self.captured_packets`, care conține toate obiectele Scapy corespunzătoare pachetelor interceptate. Se validează faptul că pachetul are layer-ul IP, precum și unul dintre protocoalele de transport TCP sau UDP. În caz contrar, se notifică utilizatorul că fluxul nu este disponibil pentru acest pachet.

Pentru a identifica toate pachetele care aparțin aceluiași flux, se extrag adresa IP sursă, adresa IP destinație, portul sursă și portul destinație, precum și protocolul. Aceste informații sunt obținute din pachetul Scapy prin accesarea layer-ului IP (`packet["IP"]`) și layer-ului de transport (`packet["TCP"]` sau `packet["UDP"]`), după caz:

```
ip_layer = packet["IP"]
proto_layer = packet["TCP"] if packet.haslayer("TCP") else packet["UDP"]
ip_src, ip_dst = ip_layer.src, ip_layer.dst
port_src, port_dst = proto_layer.sport, proto_layer.dport
```

În implementarea acestei funcționalități, s-a urmărit nu doar eficiența algoritmică, ci și claritatea interfeței și coerența comportamentului în raport cu restul aplicației. Codul este organizat modular, iar accesul la metodele `show_raw_for_stream`, `show_payload_for_stream`, `mark_stream_as_suspect` și `export_stream_to_pcap` este realizat prin butoane cu design clar și etichete explicite, conforme cu stilul general al interfeței grafice Tkinter utilizate în aplicație.

3.10 Probleme întâmpinate și soluții aplicate

Pe parcursul dezvoltării aplicației *Sentinel Traffic Analyzer*, au existat mai multe provocări tehnice și arhitecturale care au necesitat identificarea unor soluții adaptate contextului platformei și specificului sistemului de operare. Procesul de realizare a unei aplicații complexe, care integrează captură de pachete, analiză în timp real, generare de alerte și interfață grafică reactivă, a implicat luarea unor decizii iterative, în funcție de limitările identificate în fazele de testare și implementare.

Unul dintre cele mai frecvente obstacole întâmpinate a fost legat de accesul la interfețele de rețea pe sistemele Windows. Captura de pachete la nivel scăzut necesită privilegii de administrator și o bibliotecă compatibilă cu driverul rețelei (precum Npcap). În absența acestor condiții, aplicația returna erori critice sau nu detecta nicio interfață disponibilă. Pentru a atenua aceste efecte, a fost integrat un sistem de notificare vizuală, care avertizează utilizatorul cu privire la lipsa privilegiilor sau la imposibilitatea de a accesa resursele necesare.

O altă dificultate a vizat menținerea performanței interfeței grafice în condițiile în care sunt interceptate volume mari de trafic. Inițial, actualizarea vizuală a pachetelor interceptate se realiza direct în threadul principal al interfeței, ceea ce ducea la blocaje și latențe. Soluția adoptată a constat în delegarea capturii către un fir de execuție paralel, cu sincronizare controlată între fluxul de captură și actualizarea interfeței, prin intermediul funcțiilor callback. Această decuplare a îmbunătățit considerabil responsivitatea GUI-ului, chiar și în condiții de trafic ridicat.

În etapa de dezvoltare a sistemului de alertare, s-a constatat că anumite condiții de detecție produceau alerte false pozitive, în special în cazul traficului local cu caracter repetitiv (de exemplu, cereri DNS legitime în buclă). Pentru a filtra acest comportament, s-au implementat ferestre de timp ajustabile și praguri dinamice de declanșare, care evaluează atât frecvența, cât și varietatea evenimentelor suspecte. Astfel, doar comportamentele neobișnuite într-un interval scurt de timp declanșează alerte, reducând zgomotul informațional.

Integrarea componentelor externe, precum biblioteca geoip2 și fișierul GeoLite2-City.mmdb, a adus provocări legate de gestionarea fișierelor lipsă sau corupte. În absența unei verificări corespunzătoare, aplicația putea întâmpina erori la generarea hărților de rețea. Pentru a preveni aceste situații, a fost introdus un mecanism de validare a prezenței fișierului înainte de procesare, iar în cazul lipsei acestuia, aplicația continuă execuția fără a întrerupe fluxul principal, înlocuind informațiile lipsă cu valori implicite (de tip „N/A”).

Un alt aspect important a fost proiectarea unui mecanism de stocare flexibilă și sigură a fișierelor capturate, care să permită transmiterea acestora către un serviciu API. În contextul acestei integrări, au fost necesare ajustări în gestionarea numelor de fișiere, pentru a preveni suprascrierea accidentală, și în validarea formatului .pcap. De asemenea, au fost identificate erori de comunicare între client și server în cazul unor fișiere corupte sau incomplete, care au fost tratate prin introducerea unui sistem de răspuns detaliat, cu coduri de eroare și mesaje informative returnate de API.

În procesul de generare a rapoartelor PDF, utilizarea fonturilor și caracterelor diacritice a ridicat probleme de randare, în special în unele versiuni ale bibliotecii FPDF. Soluția aleasă a fost implementarea unei funcții de transliterare care normalizează caracterele specifice limbii române, pentru a asigura compatibilitatea și lizibilitatea documentelor generate, fără a compromite conținutul semantic.

4 DOCUMENTAREA SISTEMULUI

Acest capitol prezintă modalitatea de utilizare a aplicației *Sentinel Traffic Analyzer*, explicând componentele vizuale, fluxurile funcționale și modul în care utilizatorul interacționează cu sistemul. Sunt descrise principalele interfețe ale aplicației, atât cele de conectare și înregistrare, cât și componentele operaționale pentru captură și afișare a traficului de rețea. Documentarea este însoțită de capturi de ecran care exemplifică funcționarea aplicației în diverse stări și acțiuni.

4.1 Interfața de autentificare

Pagina de autentificare oferă punctul de acces în aplicație pentru utilizatorii existenți. Interfața este realizată într-un stil modern și aerisit, cu accent pe claritate vizuală. Formularul include două câmpuri principale: adresa de email și parola. Acestea sunt însoțite de pictograme relevante, care facilitează recunoașterea rapidă a tipului de date solicitate.

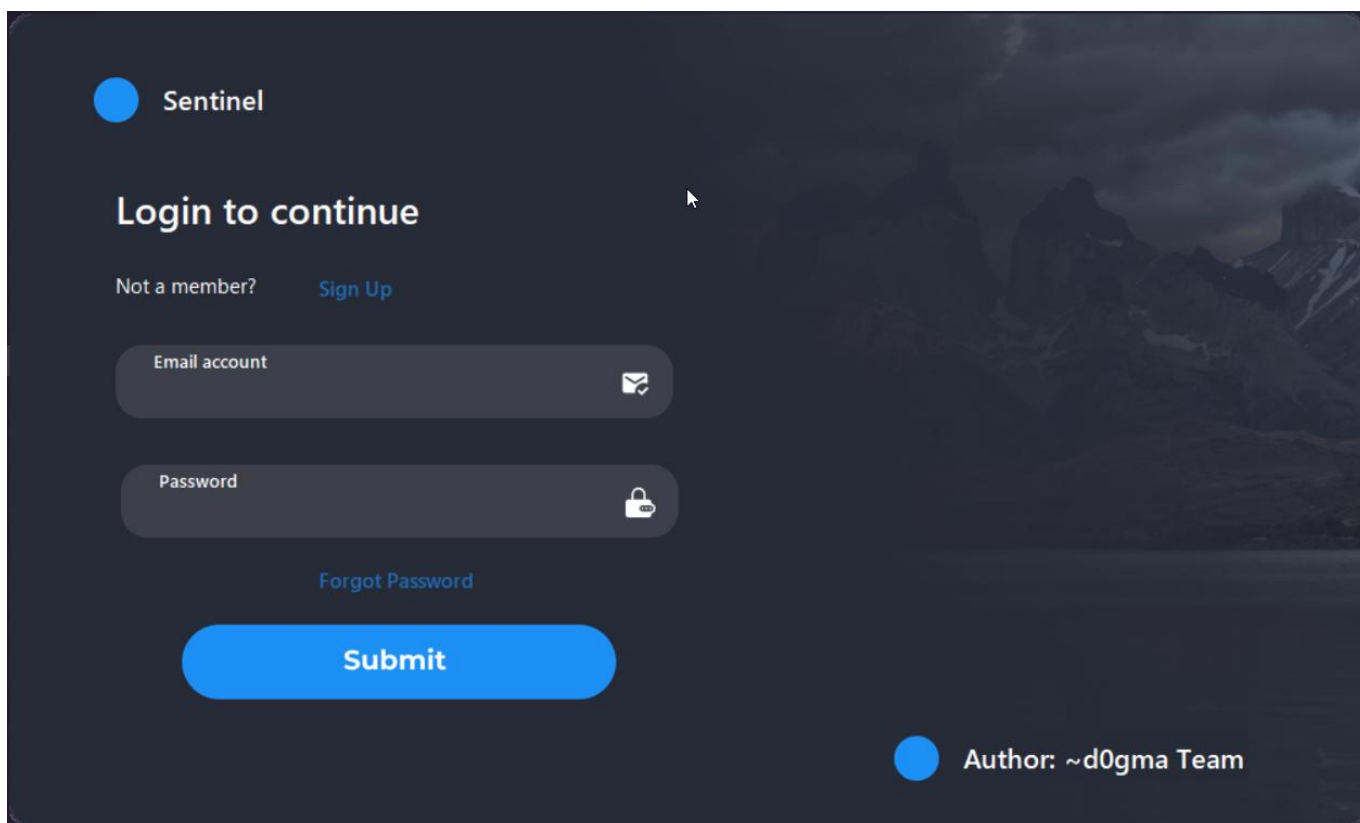


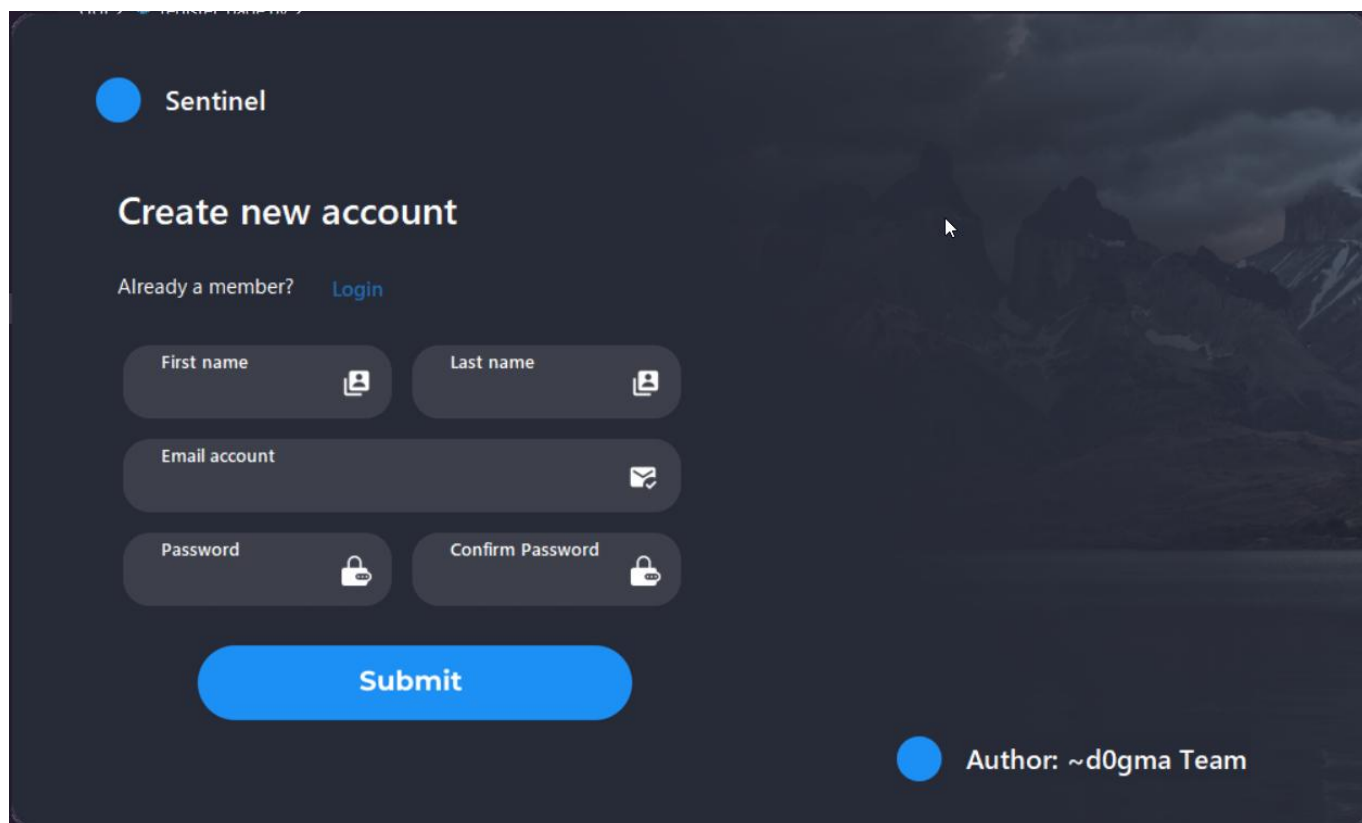
Figura 4.1 – Interfața de autentificare în aplicație

În partea superioară a paginii este prezentat logo-ul aplicației și denumirea completă. Sub formular se regăsesc legături rapide către pagina de înregistrare și funcționalitatea de resetare a parolei. Această interfață este redată în Figura 4.1, care ilustrează ecranul de logare în starea inițială.

4.2 Interfața de înregistrare

Formularul de înregistrare permite crearea unui cont nou. Acesta este accesibil prin butonul „Sign Up” de pe pagina de autentificare. Utilizatorul este invitat să completeze un set de câmpuri obligatorii,



printre care prenumele, numele, adresa de email, parola și confirmarea parolei. Interfața menține același stil vizual ca și pagina de autentificare, cu pictograme intuitive și un layout clar.






Sentinel

Create new account

Already a member? [Login](#)

First name  Last name 

Email account 

Password  Confirm Password 

Submit

Author: ~d0gma Team

Figura 4.2 – Formularul de creare a unui cont nou în aplicația Sentinel

Butonul de finalizare a înregistrării este evidențiat cromatic, pentru a sublinia funcționalitatea principală a formularului. Această pagină este ilustrată în **Figura 4.2**, care prezintă aspectul complet al interfeței de creare cont.

4.3 Interacțiunea cu baza de date

După completarea formularului de înregistrare, datele sunt validate și procesate în partea de backend. Acestea sunt inserate într-o bază de date PostgreSQL, într-o tabelă denumită users. Structura acestei tabele include un identificator de tip UUID, numele și prenumele utilizatorului, adresa de email și o parolă criptată cu algoritmul SHA-512. Acest model permite gestionarea sigură a datelor personale și a acreditivelor de conectare.

```
user_database=> select * from users;
```

id	first_name	last_name	email	password
79bc4fd8-153e-4a3f-9ffb-ca74918837b0	qq	qq	qq	d5ce2b19fbd14a25deac948154722f33efd37b369a32be8f03ec2be8ef7d3a5

(1 row)

Figura 4.3 - Conținutul bazei de date PostgreSQL după înregistrarea unui utilizator

Validarea datelor și înregistrarea reușită pot fi confirmate și prin interogări directe asupra bazei de date. **Figura 4.3** prezintă rezultatul unei astfel de interogări, evidențiind înregistrarea noului utilizator.

4.4 Interfața principală a aplicației

După autentificare, utilizatorul este redirecționat către fereastra principală a aplicației, unde sunt reunite toate modulele disponibile. Această interfață este organizată astfel încât să permită controlul complet asupra procesului de captură a traficului și accesul la funcționalitățile suplimentare.

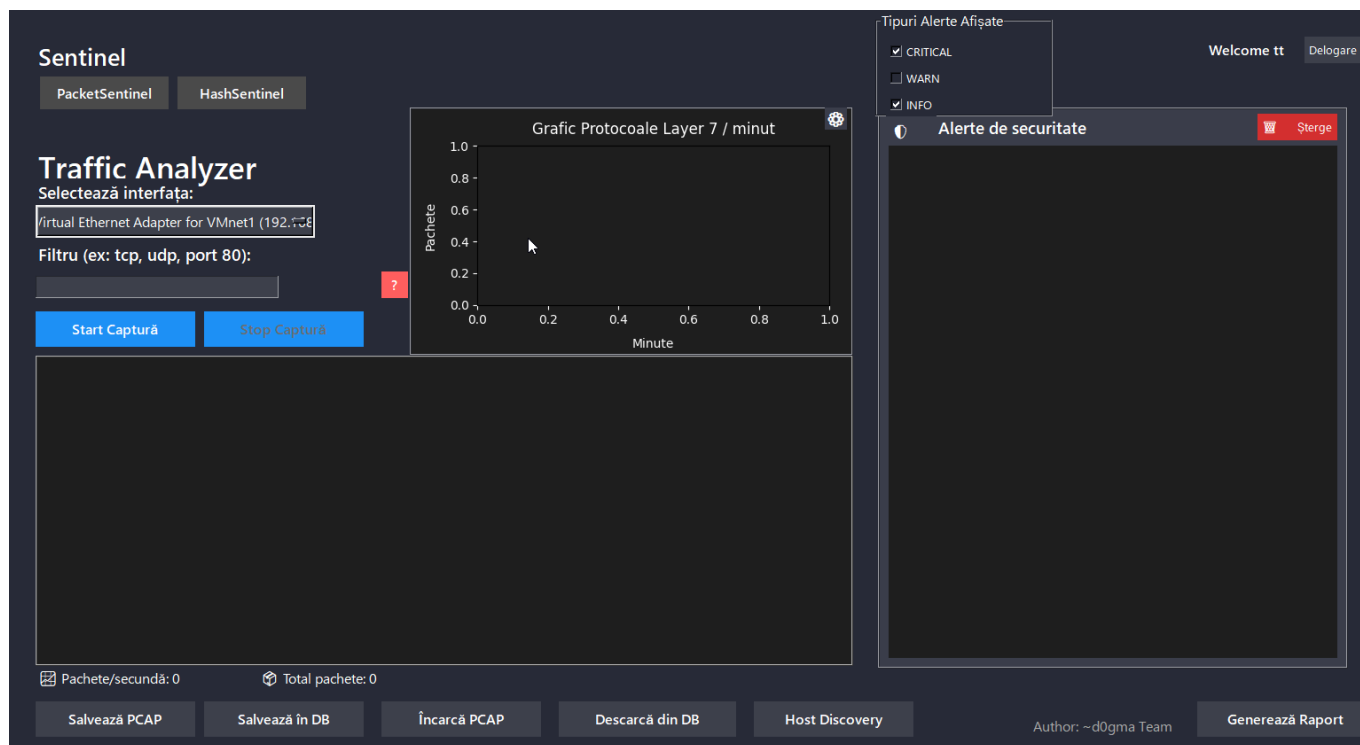


Figura 4.4 - Interfața principală a aplicației

În partea centrală a ecranului se află panoul de control pentru captură. Utilizatorul selectează interfața de rețea activă, apoi poate porni procesul de monitorizare. Traficul este afișat în timp real în zona de vizualizare. Interfața include butoane pentru oprirea capturii, salvarea sesiunii și generarea de rapoarte. Această zonă de lucru este prezentată în **Figura 4.4**, într-o stare pre-captură.

4.5 Capturarea traficului de rețea

Una dintre funcționalitățile centrale ale aplicației *Sentinel Traffic Analyzer* este capacitatea de a intercepta, analiza și afișa în timp real pachetele care circulă printr-o interfață de rețea selectată. Această funcție devine activă imediat după ce utilizatorul alege o interfață disponibilă din lista detectată de sistem și inițiază procesul de captură prin intermediul butonului dedicat din interfața principală.

Odată activată, aplicația utilizează un modul de captură de joasă nivel (scapy) care rulează într-un fir de execuție separat față de interfața grafică, asigurând astfel fluiditatea interacțiunii utilizatorului chiar și în condițiile în care traficul analizat este intens. Sistemul capturează fiecare pachet în momentul în care acesta trece prin interfață și îl procesează instantaneu, extrăgând metadatele relevante și afișându-le într-o formă structurată.

Pentru fiecare pachet interceptat, aplicația oferă următoarele informații: momentul exact al capturii (timestamp), adresa IP sursă, adresa IP destinație, protocolul utilizat (cum ar fi TCP, UDP sau ICMP), porturile implicate (acolo unde sunt aplicabile), precum și dimensiunea totală a pachetului. Aceste date sunt afișate într-un panou tip consolă, unde fiecare rând corespunde unui pachet individual, iar stilizarea vizuală a protocoalelor facilitează diferențierea rapidă a tipurilor de trafic.

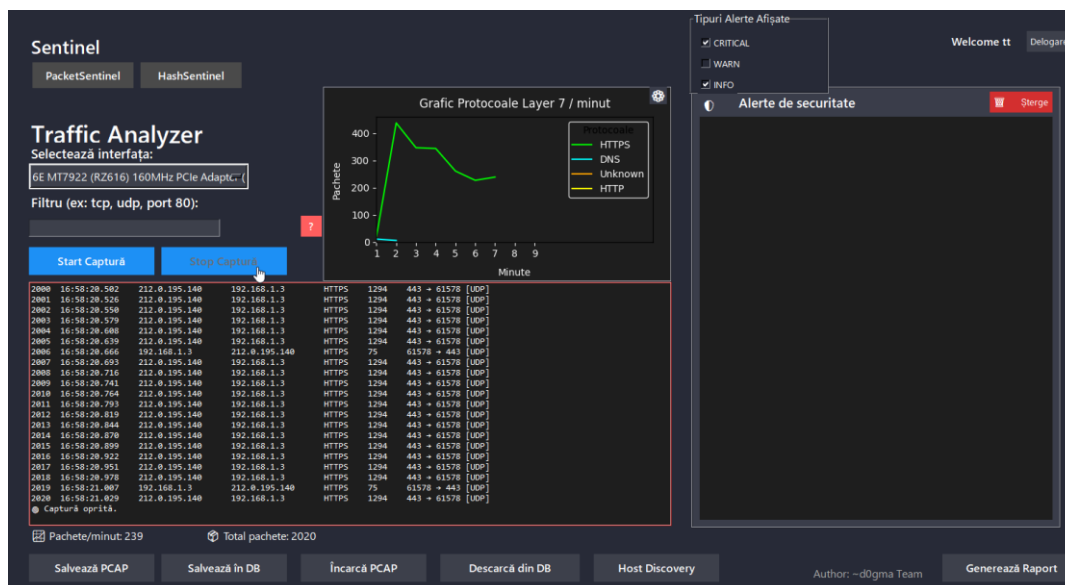


Figura 4.5 - Capturarea traficului în timp real și afișarea în interfață

Captura este realizată în timp real, dar datele afișate pot fi și analizate ulterior. Aplicația oferă posibilitatea de filtrare a pachetelor în funcție de protocoale sau de adrese IP, precum și opțiuni de căutare rapidă în conținutul afișat. În plus, utilizatorul are la dispoziție funcționalități de marcarea manuală a pachetelor suspecte, exportul selectiv al acestora, sau vizualizarea detaliilor brute ale conținutului, pentru investigații mai profunde.

Procesul de captură poate fi întrerupt în orice moment, fără pierderi de date sau coruperea sesiunii. Aplicația permite salvarea completă a sesiunii într-un fișier de tip .pcap, compatibil cu aplicații de analiză precum Wireshark. În paralel, există și opțiunea de a exporta datele sub formă text, într-un fișier .txt, pentru consultare directă, partajare sau documentare. Această flexibilitate în export facilitează adaptarea aplicației la diverse scenarii operaționale, fie ele tehnice sau educaționale.

Figura 4.5 prezintă o sesiune de captură în desfășurare, în care se observă interfața selectată (de tip loopback), zona de afișare a pachetelor, precum și starea activă a capturii, indicată vizual prin marcaje cromatice și butoane dinamice.

4.6 Vizualizarea hosturilor detectate în rețea

O funcționalitate suplimentară integrată în aplicația *Sentinel Traffic Analyzer* este componenta de **descoperire a hosturilor** active din rețea și vizualizarea relațiilor topologice dintre acestea. Această

opțiune este disponibilă prin intermediul unui tab dedicat intitulat „*Hartă Rețea*”, care face parte din secțiunea *Hosturi Detectate în Rețea*.

Funcționalitatea are la bază analiza adreselor IP sursă și destinație extrase din pachetele interceptate pe durata unei sesiuni de monitorizare. Adresele IP sunt grupate logic în funcție de conexiunile observate, iar rezultatul este afișat sub forma unei **hărți topologice interactive**, în care fiecare nod reprezintă un host detectat, iar legăturile dintre noduri corespund fluxurilor de date identificate între adrese.

Fiecare nod este etichetat cu adresa IP, locația geografică aproximativă (țară, oraș, dacă sunt disponibile) și, opțional, organizația asociată acelei adrese. În absența acestor informații, aplicația utilizează valori implicite de tip „*N/A*” pentru a menține coerența afișării. În figura 4.6 este prezentat un exemplu de rețea detectată, cu mai mulți hosturi din rețeaua locală și externă.

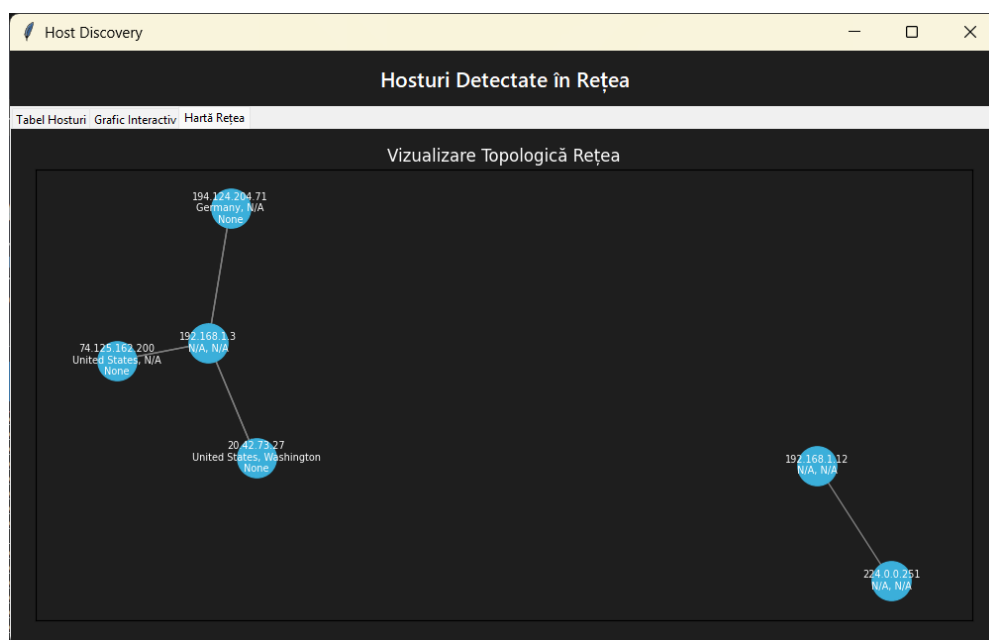


Figura 4.6 – Vizualizare topologică a hosturilor în rețea

Această vizualizare oferă utilizatorului o perspectivă sintetică asupra structurii traficului, facilitând identificarea rapidă a nodurilor centrale, a comunicațiilor neobișnuite sau a potențialelor surse externe implicate în activitatea rețelei. În special în cazul investigațiilor de securitate, această hartă poate sprijini procesul de corelare între fluxurile suspecte și entitățile implicate.

Interacțiunea cu harta este complet dinamică: utilizatorul poate deplasa nodurile, vizualiza detalii la trecerea cursorului peste un host sau exporta imaginea generată pentru documentare ulterioară. Această componentă contribuie la extinderea capacității aplicației de a oferi nu doar date brute, ci și reprezentări vizuale semnificative, utile în procesele de analiză, raportare și învățare.

4.7 Analiza detaliată a fluxurilor de date (Follow Stream)

În cadrul aplicației Sentinel Traffic Analyzer, una dintre funcționalitățile avansate care contribuie la înțelegerea completă a traficului monitorizat este opțiunea de analiză a fluxurilor, cunoscută în mod uzual

sub denumirea de Follow Stream. Această componentă permite utilizatorului să urmărească în mod logic și secvențial schimbul de date dintre două adrese IP care comunică în cadrul unei sesiuni, oferind o privire de ansamblu asupra comportamentului acestora în contextul unei conversații de rețea.

La activarea acestei funcții, este deschisă o fereastră distinctă care prezintă toate pachetele asociate fluxului selectat, în ordinea în care au fost interceptate. Această prezentare detaliată ajută utilizatorul să observe direcția de comunicare, porturile utilizate, protocolul implicat și eventualele pattern-uri repetitive sau anomalii. Fluxul este afișat sub formă textuală, structurată, cu detalii referitoare la nivelul de rețea și transport (cum ar fi IP, UDP, TCP), iar în partea superioară este menționat fluxul analizat, indicând atât adresele implicate, cât și porturile acestora.

Interfața oferă posibilitatea explorării mai detaliate a conținutului prin afișarea secțiunii *Raw*, care permite examinarea datelor brute ale pachetelor interceptate. Această vizualizare este utilă în special atunci când este necesară interpretarea manuală a formatului de date, mai ales în cazul protocoalelor aplicative nesupravegheate de analizator. În plus, pentru o înțelegere mai clară a conținutului util transmis, aplicația pune la dispoziție opțiunea de afișare a payload-ului, ceea ce facilitează extragerea esenței comunicației, eliminând detaliile de fundal care pot îngreuna interpretarea.

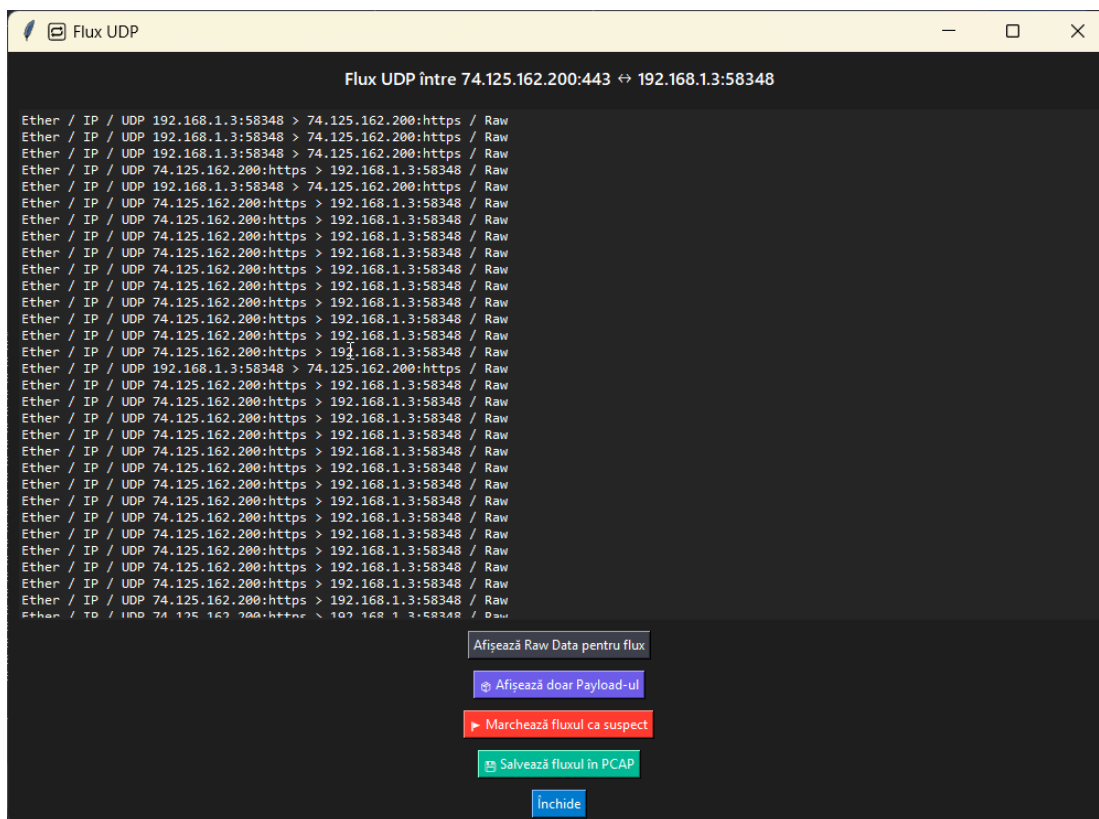


Figura 4.7 – Vizualizarea unui flux UDP între două adrese IP

În timpul analizei, dacă utilizatorul identifică un flux de date suspect sau neobișnuit, sistemul oferă posibilitatea de marcare explicită a acestuia ca fiind potențial periculos. Această marcare are efecte atât vizuale, cât și funcționale: fluxul respectiv este înregistrat în jurnalul intern de alerte și devine ușor de

urmărit în sesiunile ulterioare. Astfel, aplicația permite nu doar detectarea comportamentelor anormale, ci și documentarea și clasificarea lor pentru o analiză ulterioară mai riguroasă.

Pentru a sprijini nevoia de export și documentare externă, fluxurile pot fi salvate în format PCAP. Acest format este unul standard în industria securității informatice și poate fi deschis în aplicații consacrate precum Wireshark. Posibilitatea de a salva un flux individual într-un fișier separat facilitează transmiterea acestuia către terți, arhivarea pentru audituri viitoare sau includerea în rapoarte tehnice. Acest lucru contribuie semnificativ la caracterul profesional și reproductibil al analizelor efectuate cu ajutorul aplicației.

Funcționalitatea Follow Stream transformă aplicația dintr-un simplu captator de pachete într-un instrument complet de investigare și reconstrucție a comunicării în rețea. Prin integrarea unei astfel de componente, utilizatorii pot urmări în mod natural dialogul dintre două entități și pot corela evenimentele de rețea cu acțiuni concrete, ceea ce este esențial atât în cercetarea comportamentului rețelelor, cât și în investigarea incidentelor de securitate. Această abordare contextuală oferă un nivel de detaliu și control care o face potrivită atât pentru uz educațional, cât și pentru scenarii reale de analiză aplicată.

4.8 Funcționalități disponibile pentru analiza pachetelor individuale

Interfața aplicației Sentinel Traffic Analyzer include un sistem de interacțiune contextuală care permite analiza detaliată a fiecărui pachet capturat în cadrul unei sesiuni de monitorizare. Această funcționalitate este accesibilă direct din lista de pachete, printr-un clic dreapta pe oricare dintre înregistrări, ceea ce deschide un meniu contextual cu opțiuni suplimentare. Imaginea prezentată în Figura 4.8 ilustrează această interfață de analiză contextuală.

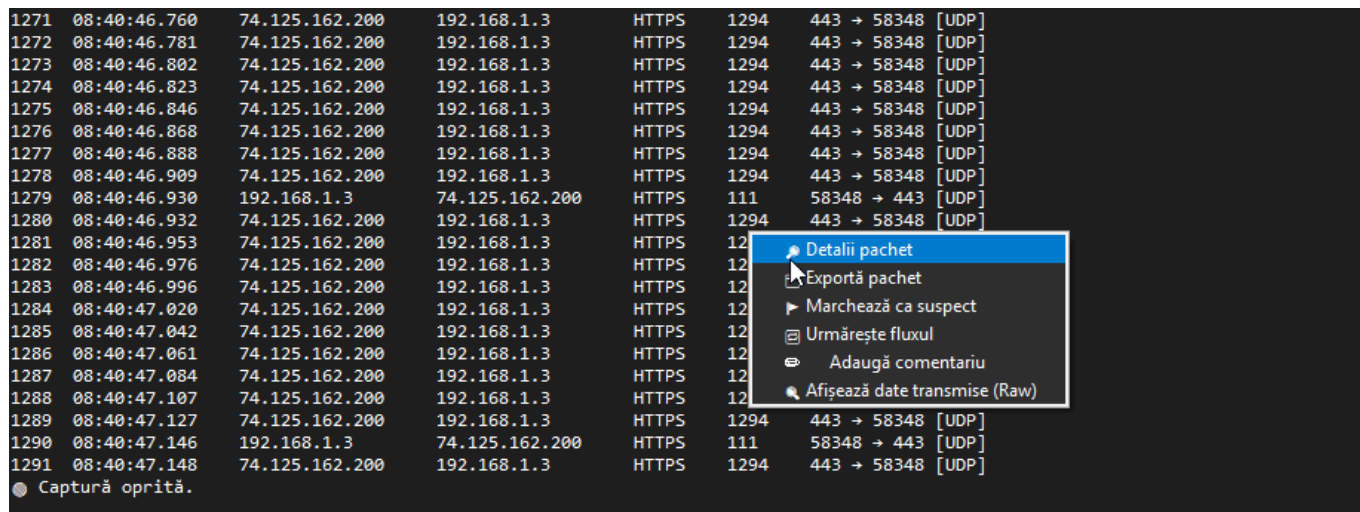


Figura 4.8 – Meniu contextual pentru interacțiunea cu pachete individuale

Prin intermediul acestui meniu, utilizatorul are posibilitatea de a consulta structura completă a pachetului selectat. Aceasta include toate nivelurile implicate în transmisia datelor – de la Ethernet, la IP și până la protocoalele de transport și aplicație – împreună cu toate valorile asociate câmpurilor de interes (adrese sursă și destinație, porturi, lungime, checksum etc.). Prezentarea acestor informații facilitează

identificarea exactă a conținutului și a eventualelor neconcordanțe, fiind utilă în special pentru identificarea traficului anormal sau pentru înțelegerea unei secvențe de comunicare.

Pe lângă consultarea detaliilor tehnice, aplicația permite și exportul selectiv al pachetului într-un fișier .pcap, care poate fi utilizat ulterior în instrumente externe precum Wireshark. Această funcționalitate este relevantă în situații de audit, arhivare sau trimitere către o echipă externă de analiză.

Dacă un pachet ridică suspiciuni în urma examinării, utilizatorul îl poate marca explicit ca suspect. Această acțiune activează mecanismul intern de alertare al aplicației, permițând ulterior filtrarea și prioritizarea acelor pachete în rapoarte și statistici. În contextul investigațiilor post-eveniment, acest proces contribuie la delimitarea clară între traficul benign și cel care necesită o atenție specială.

O altă opțiune semnificativă este cea care declanșează analiza fluxului din care pachetul face parte. Activarea acestei funcții conduce către vizualizarea detaliată a întregului dialog dintre cele două entități implicate, în cadrul ferestrei *Follow Stream*, ceea ce ajută la reconstruirea logicii comunicației și la înțelegerea întregului context.

Pentru facilitarea colaborării sau documentării ulterioare, aplicația permite și adăugarea de comentarii asociate direct cu pachetul selectat. Comentariile pot conține observații tehnice, ipoteze de investigare sau note personale, devenind vizibile ulterior în sesiuni de revizuire sau în rapoartele generate. În fine, pentru o inspecție în profunzime, utilizatorul are posibilitatea de a vizualiza conținutul efectiv transmis în format brut. Acest lucru este esențial în scenariile în care este necesară identificarea payload-ului unui protocol necunoscut sau pentru identificarea unor semnături digitale în secvența de octeți transmisă. Această vizualizare este redată într-un format interpretabil (de exemplu, hexazecimal sau ASCII), fiind utilă în special în analiza atacurilor la nivel de aplicație.

5 ESTIMAREA COSTURILOR ȘI EVALUAREA PROIECTULUI

Realizarea unei aplicații software precum *Sentinel Traffic Analyzer* presupune nu doar dezvoltarea tehnică propriu-zisă, ci și un proces atent de planificare a resurselor necesare pentru atingerea obiectivelor propuse. Estimarea costurilor are rolul de a asigura o utilizare echilibrată a timpului, efortului și resurselor materiale implicate în toate etapele proiectului. Pe lângă această dimensiune de management, dezvoltarea sistemului a fost însoțită de o analiză detaliată a eventualelor dificultăți tehnice, precum și de o evaluare a rezultatelor obținute din punct de vedere practic, educațional și științific.

5.1 Estimarea costurilor

Estimarea resurselor implicate a început încă din faza de inițiere, în care au fost identificate cerințele funcționale ale aplicației, mediul de execuție, tehnologiile alese și posibilele obstacole legate de rularea pe diverse sisteme de operare. Costurile sunt repartizate în principal între activitățile de analiză, proiectare, dezvoltare software, testare și documentare.

În ceea ce privește partea de dezvoltare, au fost avute în vedere orele de lucru dedicate scrierii codului pentru modulele principale: captură de trafic, interfață grafică, analiză de pachete, sistem de alertare și generare de rapoarte. Pentru fiecare modul, s-au calculat orele necesare în funcție de complexitate, nivelul de integrare cu celelalte componente și timpul estimat de testare. În medie, dezvoltarea fiecărui modul a presupus între 25 și 50 de ore de lucru individual, în paralel cu documentarea și testarea funcționalităților asociate.

Resursele materiale implicate sunt minime, aplicația fiind dezvoltată exclusiv cu tehnologii gratuite (Scapy, Tkinter, matplotlib, SQLite/PostgreSQL, FPDF etc.). Cu toate acestea, timpul investit în învățarea bibliotecilor și testarea acestora pe diferite platforme (Windows, Linux) este un factor semnificativ în planificarea globală.

În etapa de testare și validare, au fost incluse sesiuni de captură pe rețele locale și wireless, simularea traficului și analiza pachetelor generate. Rezultatele au fost salvate în formate diverse (.pcap, .json, .pdf) și au stat la baza evaluării performanței aplicației și a funcționalităților de alertare. Toate aceste activități au fost corelate cu obiectivele definite inițial, conducând la o distribuție echilibrată a costurilor de timp pe parcursul întregului proiect.

5.2 Aspecte tehnice și riscuri întâmpinate

În procesul de dezvoltare și testare, s-au evidențiat mai multe dificultăți de ordin tehnic, care pot influența funcționarea aplicației în anumite medii. De exemplu, pe platformele Windows, accesul la interfețele de rețea este condiționat de privilegii administrative și de instalarea corectă a componentelor externe, cum ar fi Npcap. În absența acestora, sistemul poate returna erori sau lista incompletă a interfețelor disponibile. Pentru a gestiona această situație, a fost introdus un mecanism de verificare automată a interfețelor și un sistem de notificare a utilizatorului.

Captura de pachete în timp real poate duce, în anumite condiții, la suprasolicitarea memoriei, mai ales în cazul unui volum ridicat de trafic. Pentru a atenua acest efect, s-a implementat un model de procesare paralelă, în care captura și afișarea se desfășoară în fire de execuție separate. De asemenea, utilizarea expresiilor de filtrare BPF (Berkeley Packet Filter) a permis reducerea volumului de date analizate.

Un alt obstacol a fost determinat de dificultatea în identificarea corectă a interfețelor de rețea, care sunt afișate uneori în format GUID, greu de interpretat. Pentru a remedia acest aspect, a fost introdusă o funcționalitate care corelează denumirile reale ale plăcilor de rețea cu adresele IP, astfel încât utilizatorul să poată selecta interfața dorită în mod intuitiv.

5.3 Evaluarea funcționalităților și testarea

Pentru a verifica conformitatea implementării aplicației *Sentinel Traffic Analyzer* cu cerințele inițiale, s-au desfășurat o serie de teste funcționale și observaționale în medii de rețea reale și simulate. Aceste teste au avut rolul de a valida stabilitatea, performanța și coerența logică a sistemului în diferite situații de utilizare.

Unul dintre obiectivele centrale ale testării a fost verificarea afișării în timp real a traficului interceptat. Acest aspect a fost analizat prin rularea aplicației pe mai multe interfețe de rețea (cablate și wireless), în timpul desfășurării unor activități obișnuite, precum navigarea web, actualizările de sistem sau transferurile de fișiere. Aplicația a demonstrat capacitatea de a intercepta corect pachetele, de a extrage metadatele relevante (IP sursă/destinație, port, protocol, dimensiune) și de a le afișa imediat în interfața grafică. Aceste informații au fost prezentate într-un format lizibil, cu actualizare constantă, fără blocaje sau întârzieri perceptibile.

O altă componentă testată a fost sistemul de filtrare. Aplicația oferă posibilitatea aplicării unor expresii de tip BPF (Berkeley Packet Filter), care permit restrângerea traficului afișat în funcție de protocol, port sau adresă IP. În cadrul testelor, au fost aplicate filtre precum tcp, udp port 53, icmp sau host 192.168.1.1, iar rezultatele au confirmat funcționarea corectă a acestora. Pachetele care corespundeau criteriilor definite erau capturate și afișate, în timp ce celelalte erau omise, ceea ce demonstrează eficiența mecanismului de selecție și utilitatea sa în analiza direcționată a traficului.

De asemenea, s-a testat comportamentul modulului de detecție a comportamentelor suspecte. Pentru aceasta, au fost simulate mai multe scenarii de atac, precum scanări de porturi (prin nmap), flood-uri de tip UDP și SYN, precum și atacuri de tip ARP spoofing. În toate cazurile, aplicația a reușit să genereze alerte corespunzătoare, semnalând în interfață apariția unor activități anormale. Aceste alerte au fost înregistrate simultan în fișiere de tip .log și .json, împreună cu detalii precum adresa IP sursă, tipul comportamentului detectat și nivelul de severitate estimat. Acest rezultat confirmă utilitatea reală a aplicației în activități de monitorizare și răspuns la incidente.

Funcționalitățile de export au fost și ele analizate atent. La finalul fiecărei sesiuni de captură, aplicația a permis salvarea datelor în mai multe formate: .pcap (pentru analiză ulterioară în aplicații precum Wireshark), .txt (pentru documentare simplificată), .json (pentru prelucrări automate), și .pdf (raport de sinteză). Exportul s-a realizat fără pierderi de informații, iar structura fișierelor rezultate a fost validă și completă. În cazul fișierului PDF, s-a remarcat includerea unor grafice generate automat (distribuția protocoalelor, numărul de pachete, alertele apărute), precum și sumarul sesiunii, ceea ce oferă o imagine clară și bine organizată asupra activității monitorizate.

Un aspect important remarcat pe parcursul testării a fost stabilitatea interfeței grafice, chiar și în situații cu trafic intens. Aplicația a reușit să mențină o experiență de utilizare fluidă, fără blocări sau consum excesiv de resurse. Aceasta este o caracteristică esențială pentru aplicațiile care rulează în timp real, unde întârzierile sau defecțiunile de afișare pot compromite valoarea analizei efectuate.

Concluzii

Lucrarea de față a urmărit proiectarea, implementarea și validarea unei aplicații software destinate interceptării și analizei traficului de rețea, cu accent pe vizibilitatea în timp real, detecția comportamentelor suspecte și generarea de rapoarte relevante pentru analiza ulterioară. Aplicația *Sentinel Traffic Analyzer* răspunde unei nevoi reale din domeniul securității informatice: aceea de a înțelege și evalua traficul de rețea în mod clar, controlabil și reproductibil, fără a necesita soluții comerciale costisitoare sau greu de adaptat.

Prin abordarea modulară și utilizarea unor tehnologii open-source (Python, Scapy, Tkinter, matplotlib, SQLite/PostgreSQL), sistemul a fost construit astfel încât să asigure atât funcționalitatea dorită, cât și posibilitatea extinderii ulterioare. Captura în timp real, filtrarea pachetelor, detecția unor tipare de atac (scanări, flood, spoofing), generarea de alerte și exportul rapoartelor în formate multiple au fost implementate și testate cu succes în rețele reale și simulate. Rezultatele obținute au confirmat stabilitatea aplicației, precum și utilitatea acesteia în activități de analiză și documentare.

Unul dintre avantajele majore ale aplicației este interfața sa intuitivă, care facilitează utilizarea chiar și în cazul persoanelor fără experiență avansată în rețelistică. În același timp, sistemul oferă funcționalități avansate de export, filtrare și vizualizare a traficului, care îl recomandă pentru utilizare în scopuri educaționale, dar și în scenarii de audit sau analiză incidentă. Integrarea componentelor de geolocalizare și clasificare a alertelor adaugă valoare practică în analiza contextuală a traficului.

Din punct de vedere tehnic, provocările întâmpinate – legate de accesul la interfețele de rețea, performanța interfeței grafice, evitarea alertelor false pozitive și compatibilitatea multiplatformă – au fost rezolvate prin soluții bine fundamentate, care au consolidat robustețea și funcționalitatea finală a aplicației. Testele efectuate au arătat că sistemul poate funcționa stabil în condiții variate de trafic, fără întreruperi sau degradări semnificative ale performanței.

Din perspectivă educațională și de cercetare, aplicația poate fi utilizată ca platformă pentru studierea traficului de rețea, pentru înțelegerea protocolului și a riscurilor asociate, dar și ca bază de plecare pentru extinderea funcționalităților în direcția sistemelor IDS/SIEM. Posibilitatea de integrare cu algoritmi de învățare automată pentru clasificarea traficului sau predicția anomaliilor deschide perspective promițătoare pentru viitoare dezvoltări.

BIBLIOGRAFIE

- [1] Bejtlich, R. (2005). *The Tao of Network Security Monitoring: Beyond Intrusion Detection*. Addison-Wesley.
- [2] Orebaugh, A., Ramirez, G., & Beale, J. (2006). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress.
- [3] Scapy Documentation. (2024). [Online]. Available: <https://scapy.readthedocs.io/> Accesat la: 25.05.2025.
- [4] Kurose, J. F., & Ross, K. W. (2021). *Computer Networking: A Top-Down Approach* (8th ed.). Pearson.
- [5] Paxson, V. (1999). "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks*, vol. 31, pp. 2435–2463.
- [6] Suricata IDS. *Suricata – Open Source Threat Detection Engine*. [Online]. Available: <https://suricata.io/>. Accesat la: 10.03.2025.
- [7] Nmap Network Scanning. (2024). *Nmap Guide to Network Discovery and Security Scanning*. [Online]. Available: <https://nmap.org/book/>. Accesat la: 15.03.2025.
- [8] Python Software Foundation. (2024). *Scapy Packet Manipulation Library*. [Online]. Available: <https://pypi.org/project/scapy/>. Accesat la: 25.03.2025.
- [9] RFC 791 – *Internet Protocol*. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc791>. Accesat la: 12.02.2025.
- [10] RFC 793 – *Transmission Control Protocol*. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc793>. Accesat la: 12.02.2025.
- [11] Tkinter GUI Docs. (2024). *Python Tkinter GUI Reference*. [Online]. Available: <https://docs.python.org/3/library/tkinter.html>. Accesat la: 10.01.2025.
- [12] Stallings, W. (2017). *Network Security Essentials* (6th ed.). Pearson.
- [13] Wireshark. (2024). *Wireshark User Guide*. [Online]. Available: <https://www.wireshark.org/docs/>. Accesat la: 25.01.2025.