# Capitolul 4. Analiză și fundamentare Teoretică

## 4.1 Fundamentele capturii de pachete în rețea

Capturarea pachetelor de rețea presupune interceptarea datelor care tranzitează o interfață de rețea într-un sistem informatic. Acest proces are la bază o caracteristică esențială a interfeței de rețea, și anume posibilitatea de a fi comutată în mod promiscuu. În acest mod, interfața nu mai filtrează pachetele adresate altor noduri, ci transmite mai departe întregul trafic către aplicația de captură, oferindu-i acesteia acces complet la tot ce circulă în rețea.

Această capacitate de observare pasivă este indispensabilă în analiza securității, a performanței sau a comportamentului unei rețele. Prin captură, aplicația devine capabilă să observe în timp real comunicațiile între dispozitive, fără a interveni sau a modifica traficul respectiv, ceea ce garantează acuratețea și neutralitatea analizei.

Captura efectivă se face utilizând biblioteca scapy, care interacționează direct cu dispozitivul de rețea. Prin funcția sniff(), aplicația inițiază procesul de monitorizare continuă a interfeței selectate.

## 4.2 Modele logice de procesare a pachetelor

Funcționarea aplicației se bazează pe următorul model logic:  
  
- Captură: Interfața selectată transmite toate pachetele către procesorul de captură (packet\_callback), unde sunt extrase date relevante.  
- Analiză: Pe baza protocoalelor detectate (IP, TCP, UDP, ARP, ICMP), aplicația construiește un tablou cu informații precum sursă, destinație, porturi, lungime, tip pachet.  
- Filtrare: Aplicația oferă filtrare personalizată la nivel de aplicație și conversie în expresii BPF pentru compatibilitate cu scapy.  
- Vizualizare: Datele analizate sunt afișate în interfața grafică, iar la dublu-click, se oferă o detaliere stratificată și în format hexazecimal.

## 4.3 Protocoale suportate și logica de identificare

Pentru fiecare pachet, aplicația identifică protocolul pe baza straturilor Scapy. Protocolul este determinat în ordinea:  
  
- ARP – identificat prin packet.haslayer(scapy.ARP)  
- TCP – oferă cele mai multe detalii (SYN, ACK, SEQ, etc.)  
- UDP – se detectează porturile și dimensiunea  
- ICMP – mesaje de tip „echo request/reply”  
- IPv6 – afișat ca tip generic, fără inspecție detaliată



## 4.4 Structura logică a aplicației

Aplicația este structurată în următoarele module logice:  
  
- Captură și analiză (packet\_callback): determină protocol, extrage metadate și construiește linia de afișare.  
- Filtrare (apply\_local\_filter, convert\_expression\_to\_bpf): traduce expresii logice în filtre aplicabile local sau BPF.  
- Export (wrpcap): permite salvarea pachetelor individual sau în grup.  
- Interfață grafică: bazată pe Tkinter, organizează meniul de control, tabelul principal și ferestrele de detalii.

## 4.5 Argumentare logică a alegerilor

Alegerea scapy:  
- Permite o manipulare foarte granulară a pachetelor și este portabil între sisteme.  
- Comparativ cu pyshark sau pcapy, scapy oferă flexibilitate superioară la filtrare, vizualizare și extragere de metadate.  
  
Alegerea Tkinter:  
- Este inclus nativ în Python și permite rapid prototipare de interfețe grafice.  
- În combinație cu ttk.Treeview, oferă posibilitatea afișării dinamice și colorate a conținutului.

## 4.6 Considerații privind securitatea și accesul

Pentru a capta pachete de rețea, aplicația necesită permisiuni ridicate (root/administrator), deoarece interacționează direct cu interfața de rețea. În medii Linux, acest lucru este gestionat prin rularea cu sudo, iar în Windows, prin executarea aplicației ca Administrator.

## 4.7 Integrarea cu ELK Stack pentru vizualizarea traficului

Pentru a îmbunătăți vizibilitatea și capacitatea de analiză a datelor capturate, aplicația este integrată cu ecosistemul ELK Stack (Elasticsearch, Logstash, Kibana), o soluție populară de procesare și vizualizare a logurilor și fluxurilor de date.  
  
Funcționarea acestui pipeline este următoarea:  
  
- Captura pachetelor este realizată cu scriptul Python (sniffer.py) care extrage metadate semnificative.  
- Transformarea în loguri se face printr-un modul custom care scrie datele în fișiere .log structurate (ex: JSON, text delimitat).  
- Logstash preia aceste fișiere, le parsează conform unui pipeline.conf configurabil, și le trimite către Elasticsearch.  
- Elasticsearch stochează și indexează datele într-un format optim pentru căutare și agregări.  
- Kibana oferă interfață vizuală pentru explorarea și dashboard-ul traficului de rețea.  
  
Această integrare transformă aplicația dintr-un tool de captură local într-o platformă scalabilă de analiză și securitate de rețea.



Figura 4.2. Pipeline-ul de analiză: de la captură locală la vizualizare web.

Avantajele integrării ELK:

Centralizarea datelor – toate capturile de trafic sunt stocate într-un motor de căutare performant.

Vizualizare interactivă – utilizatorii pot filtra, căuta și crea dashboard-uri în timp real.

Extensibilitate – sistemul permite adăugarea de metadate, alerte sau reguli personalizate.

Această integrare transformă aplicația dintr-un tool de captură local într-o platformă scalabilă de analiză și securitate de rețea.