UNIVERSITATEA DIN
BUCUREȘTI

FACULTATEA DE
MATEMATICĂ ȘI
INFORMATICĂ

SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

Proiect de diplomă

DETECȚIE AUTOMATĂ DE ATACURI IN REȚELE WI-FI

Absolvent
Fărcășanu Tudor Andrei

Coordonator științific
Conf. Dr. Paul Irofti

București, iunie 2024

Rezumat

Această lucrare are ca scop elaborarea de metode de detectare și clasificare a atacurilor asupra rețelelor Wi-Fi utilizând tehnici din domeniul *deep learning*, folosind AWID3, un set de date conceput pentru dezvoltarea de sisteme de detectare a intruziunilor în rețele Wi-Fi. Am prelucrat acest set de date, pentru a îi reduce dimensiunea și a elimina informațiile redundante sau irelevante pentru detectarea atacurilor la nivelul 2 pe stiva OSI.

Având în vedere lucrări din același domeniu ce utilizează versiuni precedente ale AWID, am dezvoltat și antrenat rețele neurale clasice dar și de tip *autoencoder* pentru scenariile de clasificare binară a traficului și de clasificare a atacurilor pe 8 categorii, folosind mai multe strategii.

În final, am evaluat modelele și strategiile elaborate pe baza scorului F1 mediu obținut în scenariul de clasificare a atacurilor pe categorii, demonstrând experimental performanța superioară a abordării bazată pe rețele neurale clasice.

Abstract

This thesis aims to develop methods for detecting and classifying Wi-Fi attacks using deep learning techniques, employing AWID3, a dataset designed for developing Wi-Fi intrusion detection systems. This dataset was processed to reduce its size and eliminate redundant or irrelevant information for detecting layer 2 attacks on the OSI stack.

Considering works in the same field that used earlier versions of AWID, both classical and autoencoder neural networks were developed and trained for binary traffic classification and multi-class attack classification scenarios using several strategies.

Finally, the models and strategies were evaluated based on the average F1 score obtained in the multi-class attack classification scenario, experimentally demonstrating the superior performance of the approach based on classical neural networks.

Cuprins

Listă de figuri

Listă de tabele

Capitolul 1

Introducere

Detecția de anomalii reprezintă un domeniu vast în analiza datelor, cu aplicații diverse de la securitatea cibernetică și diagnosticarea echipamentelor industriale până la prevenirea fraudelor și monitorizarea sănătății. Anomaliile (*outliers*) sunt acele observații sau evenimente care prezintă diferențe semnificative față de restul majorității observațiilor. Identificarea acestora poate semnala evenimente rare dar critice, cum ar fi atacuri cibernetice, defecțiuni ale echipamentelor sau tranzacții frauduloase.

Detectarea anomaliilor implică utilizarea unei varietăți de tehnici și algoritmi pentru a analiza seturi mari de date și a identifica tipare neobișnuite. Abordările tradiționale în acest sens includ metode statistice, bazate pe calculul de distanțe sau pe algoritmi de învățare automată supervizată și nesupervizată. În era *Big Data*, aceste metode au fost extinse și optimizate prin utilizarea algoritmilor de învățare profundă (*deep learning*), care pot oferi performanțe superioare în identificarea anomaliilor complexe și subtile.

Rețelele Wi-Fi au devenit o componentă indispensabilă a vieții moderne, asigurând conectivitatea esențială pentru dispozitivele mobile, computerele personale, echipamentele smart home și multe altele. În contextul în care mobilitatea și accesul constant la internet sunt critice, rețelele Wi-Fi asigură o infrastructură flexibilă și accesibilă, susținând atât activitățile personale cât și cele profesionale.

1.1 Importanța securității rețelelor Wi-Fi

Creșterea accelerată a numărului de dispozitive conectate, adesea prin intermediul rețelelor Wi-Fi, a condus la mărirea riscurilor de expunere la atacuri cibernetice prin intermediul lor, securitatea acestor rețele devenind astfel o prioritate pentru utilizatori, organizații și furnizorii de servicii de internet.

Pătrunderea unui atacator într-o rețea Wi-Fi privată poate cauza daune semnificative. Prin interceptarea și manipularea comunicațiilor, acesta poate iniția acțiuni cum ar fi furtul de credențiale și date personale prin imitarea site-urilor web legitime, realizarea de configurații pe echipamentele compromise din interiorul rețelei pentru menținerea contro-

lului de la distanță (de exemplu prin configurarea de rețele private virtuale) sau furnizarea de pachete software și firmware malițioase prin manipularea traficului pentru înlocuirea celor autentice, atunci când sunt detectate solicitări de descărcare de la utilizatori sau dispozitivele lor. Impactul acestor atacuri poate fi devastator, nu doar la nivel individual, ci și pentru companii și instituții.

Pentru elaborarea de metode de detectare și clasificare a atacurilor asupra rețelelor Wi-Fi vom folosi setul de date AWID3 [4] (*Aegean Wi-Fi Intrusion Dataset*), care este conceput pentru dezvoltarea de sisteme de detectare a intruziunilor în rețele Wi-Fi. Acesta conține atacuri asupra protocolului de securitate WPA2, în rețele ce au activat PMF (*Protected Management Frames*) și în medii care folosesc EAP (*Extensible Authentication Protocol*).

Deși WPA2 nu mai este cel mai nou standard de securitate folosit în rețelele Wi-Fi, fiind înlocuit de WPA3, metodele elaborate în această lucrare dar și atacurile asupra WPA2 vor rămâne relevante, conform [7], ce evidențiază vulnerabilități critice în modul de tranziție al WPA3, care permite utilizarea concomitentă a WPA2 și WPA3 pe aceeași rețea logică, cu aceeași parolă. În lucrare se arată că dispozitivele client ce sunt conectate folosind WPA3 la o rețea legitimă pot fi forțate să se conecteze folosind WPA2 la o copie ilegitimă a rețelei (*Rogue AP*), permițând capturarea parțială a *4-way handshake* specific WPA2 și recuperarea parolei prin atacuri tradiționale, *brute-force* sau de tip dicționar, ca în figura 1.1.

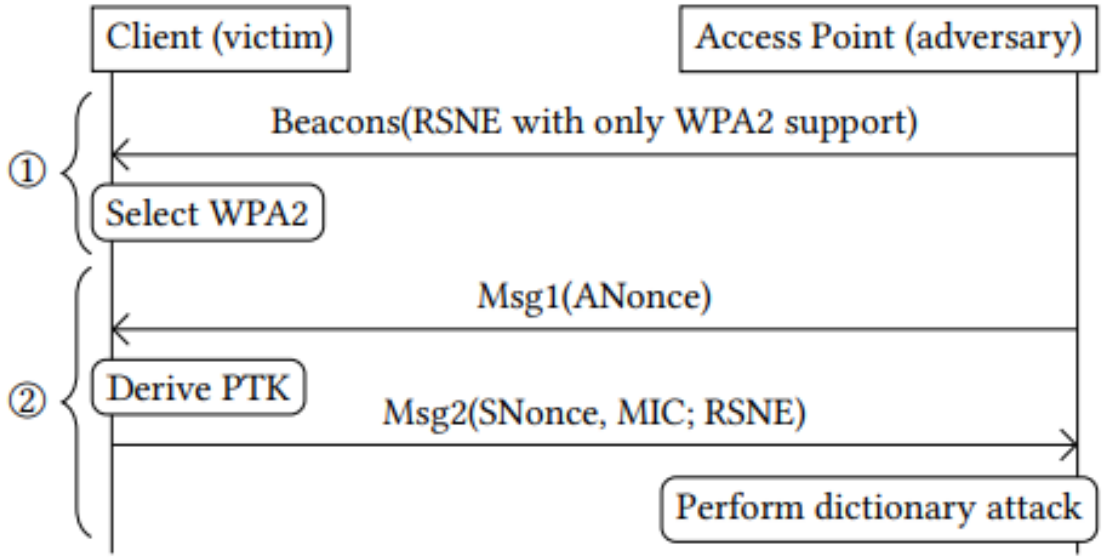

Figura 1.1: Atac de tip *downgrade* împotriva *WPA3 Transition mode* [7]

Studiul subliniază că aceste probleme vor persista cât timp rețelele WPA3 operează în mod de tranziție, menținând vulnerabilitatea la atacuri tipice WPA2.

1.2 Contribuții

Această lucrare contribuie la domeniul detectării anomaliilor prin:

1. Elaborarea de metode de detectare și clasificare a atacurilor asupra rețelelor Wi-Fi, prin dezvoltarea și antrenarea unor rețele neurale clasice dar și unor rețele de tip *autoencoder*.

2. Analizarea și prelucrarea setului de date AWID3 pentru a îi reduce dimensiunea și a elimina informațiile redundante sau irelevante pentru detectarea atacurilor doar la nivelul 2 pe stiva OSI.

Capitolul 2

Noțuni teoretice

2.1 Rețele Neurale

Rețelele neurale artificiale (*ANN - Artificial Neural Networks*) sunt modele computaționale inspirate de structura și funcționarea creierului uman. Acestea sunt utilizate pe scară largă în diverse domenii ale inteligenței artificiale datorită capacității lor de a învăța și generaliza din date complexe.

Fiecare neuron dintr-o rețea neurală primește una sau mai multe intrări, le procesează prin intermediul unei funcții de activare și produce o ieșire. Funcțiile de activare comune includ funcția sigmoidă, ReLU (*Rectified Linear Unit*) și tangenta hiperbolică, fiecare având proprietăți specifice care influențează modul în care rețeaua învață și generalizează.

Rețelele neurale sunt alcătuite din noduri interconectate, numite neuroni artificiali, organizate în straturi succesive, fiecare având un rol specific:

- **Stratul de intrare**: Primește datele inițiale, cum ar fi pixelii unei imagini, și le transmite mai departe către straturile ascunse.
- **Straturile ascunse**: Straturi intermediare care efectuează transformări și extrag caracteristici relevante din datele de intrare. Fiecare neuron din aceste straturi primește informații de la neuronii din stratul anterior, le procesează folosind o funcție de activare și transmite rezultatul către următorul strat.
- **Stratul de ieșire**: Stratul final al rețelei care furnizează predicțiile sau deciziile rețelei neurale, cum ar fi clasificarea unei imagini.

Antrenarea unei rețele neurale implică ajustarea ponderilor sinaptice dintre neuroni pentru a minimiza eroarea dintre ieșirea rețelei și valorile așteptate. Algoritmul de învățare cel mai comun este *backpropagation*, care propagă înapoi eroarea prin rețea și ajustează ponderile pe baza gradientului funcției de cost. Funcția de cost măsoară diferența dintre predicțiile rețelei și valorile reale și poate lua forme precum eroarea medie pătratică (*Mean Squared Error, MSE*) sau *cross-entropy*.

2.2 Autoencoder

Un autoencoder este un tip specific de rețea neurală care are ca obiectiv reproducerea datelor de intrare la ieșire, după ce le-a comprimat într-o reprezentare codificată de dimensiune redusă în straturile ascunse ale rețelei. Această reprezentare codificată are rolul de a captura caracteristicile esențiale ale datelor de intrare, permițând reconstrucția lor ulterioară. Arhitectura unui autoencoder este alcătuită din două părți principale:

- **Encoderul:** Această parte a rețelei neurale comprimă datele de intrare într-o reprezentare codificată de dimensiune redusă, trecând prin straturi ascunse cu un număr descrescător de neuroni. Funcția encoderului poate fi considerată o mapare $h = f(x)$, unde x reprezintă inputul și h codificarea latentă. Scopul encoderului este de a extrage caracteristicile esențiale din datele brute, reducând astfel dimensiunea acestora.

- **Decoderul:** Această parte reconstruiește datele de intrare inițiale pornind de la reprezentarea codificată, trecând prin straturi ascunse cu un număr crescător de neuroni. Funcția decoderului este o mapare $x' = g(h)$, unde x' este outputul reconstruit. Decoderul încearcă să refacă datele inițiale cât mai precis posibil din codificarea latentă.

Funcția de pierdere (*loss*) a autoencoderelor este proiectată pentru a minimiza diferența dintre intrările originale și reconstrucțiile lor.

Antrenarea unui autoencoder se realizează în mod nesupervizat, prin minimizarea unei funcții de pierdere care măsoară diferența dintre datele de intrare inițiale și ieșirea reconstruită de rețea. O funcție de pierdere comună este eroarea pătratică medie (MSE), care penalizează diferențele mari între valorile inițiale și cele reconstruite. Scopul este ca autoencoderul să învețe o reprezentare codificată care captează caracteristicile esențiale ale datelor, permițând reconstrucția lor cât mai fidelă.

Există mai multe tipuri de autoencoder, fiecare cu specificitățile sale:

- **Autoencodere Simple:** Acestea utilizează rețele neuronale obișnuite pentru encoder și decoder.

- **Denoising Autoencoders:** Acestea sunt antrenate pentru a reconstrui intrările inițiale dintr-o versiune coruptă a acestora, ceea ce le conferă robustețe în prezența zgomotului în date.

- **Variational Autoencoders (VAE):** Extind autoencoderele prin învățarea probabilistică, reprezentările latente fiind învățate ca distribuții probabilistice, permițând generarea de date noi care sunt similare cu datele inițiale.

2.3 Tipurile de atac asupra rețelelor Wi-Fi din AWID3

În urma prelucrării setului de date așa cum este detaliat în capitolul 3, am selectat doar datele ce conțineau atacuri specifice Wi-Fi. Natura acestor atacuri este detaliată mai jos:

Krack (Key Reinstallation Attack)

Krack (Key Reinstallation Attack) este un atac care exploatează o vulnerabilitate în protocolul WPA2, permițând atacatorului să reinstaleze cheia criptografică utilizată în timpul *4-way handshake*. Prin manipularea mesajelor de *handshake*, atacatorul poate face ca cheia de criptare să fie reinstalată, fapt ce permite reinițializarea valorilor a unor parametri criptografici, care pot fi apoi folosiți pentru decriptarea traficului de date.

Kr00k

Kr00k este un atac ce exploatează o vulnerabilitate prezentă în anumite cipuri Wi-Fi produse de companii precum Broadcom și Cypress. Această vulnerabilitate permite unui atacator să decripteze pachetele de date Wi-Fi chiar și atunci când acestea sunt criptate folosind WPA2.

Pentru a exploata această vulnerabilitate, un atacator trimite pachete de deautentificare către clientul victimă. Când dispozitivele se deconectează de la rețea, cheia temporară de criptare este resetată la zero, iar pachetele care se aflau în *buffer*-ul cipului sunt criptate cu cheia nulă și trimise. Astfel, atacatorul poate induce aceste deconectări repetate pentru a obține un volum mare de date decriptate.

RogueAP (Rogue Access Point)

Rogue Access Point se referă la un punct de acces neautorizat instalat într-o rețea. Acest punct de acces poate fi folosit pentru a intercepta traficul, pentru a lansa atacuri de tip *Man-in-the-Middle* sau pentru a compromite securitatea rețelei. Acestea pot fi instalate de atacatori în interiorul unei zone private, direct în rețeaua locală. Astfel, pot monitoriza traficul din rețeaua locală (care folosește doar cabluri Ethernet de exemplu) dintr-o zonă publică învecinată.

Evil Twin

Atacul de tip Evil Twin implică crearea unui punct de acces fals care imită unul legitim. Utilizatorii sunt păcăliți să se conecteze la acest punct de acces, permițând astfel atacatorului să intercepteze datele transmise între utilizator și rețea. Atacatorul poate configura un punct de acces cu același nume (SSID) și setări de securitate ca și rețeaua autentică, iar utilizatorii, nefiind conștienți, se conectează la rețeaua falsă, expunându-și

astfel datele sensibile. De obicei, acest tip de atac are ca țintă o rețea nesecurizată dintr-un spațiu public, fiindcă poate păcăli utilizatorii prin faptul că poate avea putere mai mare de transmisie, și astfel acoperire mai bună decât rețeaua legitimă.

Disas (Disassociation) și Deauth (Deauthentication)

Atacurile de tip *Disassociation* și *Deauthentication* implică trimiterea de cadre de tip *disassoc* și respectiv *deauth* către clientul victimă, forțându-l să se deconecteze de la rețeaua Wi-Fi, fiind de obicei folosite ca un atac de tip DoS (*Denial of Service*). Diferența dintre cele două este că pachetele de tip *deauth* transmit dispozitivului țintă să se deconecteze complet de la rețea, în timp ce *disassoc* comunică dispozitivului doar să se deasocieze de la un anume *access point*, dar să rămână autentificat în rețea, eventual prin alt *access point*.

În cazul *Disassociation* atacatorul poate încerca să forțeze utilizatorul să se conecteze la un *access point* ilegitim care este setat astfel încât să pară că face parte din rețeaua țintă, asemănător unui atac de tip Evil Twin.

Atacul *Deauthentication* este de obicei utilizat fie pentru a întrerupe conexiunea victimei în mod repetat (DoS), sau pentru a forța victima să se reautentifice, astfel încât atacatorul să poată captura *4-way handshake*, care poate fi folosit mai târziu cu un atac de tip dicționar sau *brute-force* pentru recuperarea parolei. De asemenea, *Deauthentication* este folosit pentru a facilita și alte tipuri de atacuri, cum ar fi Evil Twin, Krack sau Kr00k.

(Re)Assoc ((Re)Association)

Atacurile de tip *(Re)Assoc* [3] au scopul de a manipula dispozitivele Wi-Fi de tip client pentru a se asocia cu un *access point* malițios, prin exploatarea unor funcționalități de scanare și conectare automată la rețele Wi-Fi cunoscute, implementate în managerul de rețea al unui sistem de operare. Acest lucru se poate realiza prin trimiterea repetată a pachetelor de tip *Beacon* ce aparțin unor rețele deja memorate în dispozitivul client.

Capitolul 3

Pregătirea setului de date

3.1 Date inițiale și prelucrare preliminară

AWID3 [4] este un set de date conceput pentru a fi folosit în dezvoltarea de sisteme de detectare a intruziunilor în rețele Wi-Fi, disponibil public atât sub formă de capturi de rețea, cât și în format CSV. În formatul său CSV, cuprinde 254 proprietăți, dintre care o singură coloană indică tipul eșantionului, fiecare aparținând uneia dintre cele 13 categorii (Figura 3.1).

File Name	Attacks
Deauth	Deauth. Flooding
	Deauth. Flooding Broadcast
Disass	Disass. Flooding
	Disass. Flooding Broadcast
	Amok
(Re)Assoc	Assoc. Request Flooding
	Reassoc. Request Flooding
	Beacon Flooding
Rogue_AP	Rogue AP
Krack	Channel MitM attack
	Key Reinstallation
Kr00k	TK Reinstallation
SSH	SSH Brute Force
Botnet	Botnet
Malware	WannaCry Plus
	TeslaCrypt
SQL_Injection	SQL Injection
SSDP	SSDP Amplification
Evil_Twin	Captive Portal
	Eaphammer
Website_spoofing	ARP & DNS Poisoning

Figura 3.1: Atacurile disponibile în AWID3 [1]

Pentru a reduce cantitatea de memorie necesară folosirii setului de date, am prelucrat și folosit mai departe doar fișierele ce conțineau eșantioane din 8 categorii (Tabela 3.1), care reprezintă doar tipuri de atacuri specifice Wi-Fi.

Categorie	Număr total	Proporție
Normal	15107404	96.99%
Kr00k	191803	1.23%
Evil_Twin	104827	0.67%
Disas	75131	0.48%
Krack	49990	0.32%
Deauth	38942	0.25%
(Re)Assoc	5502	0.035%
RogueAP	1310	0.0084%

Tabela 3.1: Distribuția tipurilor de atacuri în datele folosite

Am șters inițial 181 de coloane care conțineau date ce corespund straturilor superioare nivelului 2 pe stiva OSI sau erau complet nule în toate fișierele. Am prelucrat și manual, ștergând eșantioane ce fie aveau eticheta nulă, fie aveau majoritatea proprietăților nule. Apoi, am umplut valorile lipsă din coloanele rămase cu elemente nule, sau care să aibă un înțeles similar în contexul fiecărei proprietăți.

3.2 Feature engineering

Printre coloanele rămase s-au regăsit proprietăți care conțineau text, sau reprezentări textuale ale diverselor câmpuri din pachetele de rețea, date redundante, dar și erori în colectarea acestora.

3.2.1 Transformări aplicate

Proprietăți convertite în valori booleane

Coloanele *wlan_rsna_eapol.keydes.data* și *data.data* conțineau text în format hexazecimal. Având în vedere că informațiile din aceste coloane prezentau lungimi variabile și nu contribuiau direct la clasificarea traficului, au fost convertite în valori booleane: adevărat (1) pentru conținut nenul și fals (0) în caz contrar.

Proprietăți codificate

Coloanele *wlan.bssid*, *wlan.da*, *wlan.ra*, *wlan.sa* și *wlan.ta* au fost codificate folosind modulul Python *macaddress*[1], după ce au fost prelucrate pentru a reține numai prima adresă MAC din fiecare celulă, în cazul în care conțineau adrese multiple separate prin '-' (unele înregistrări conțineau multiple adrese MAC concatenate, în loc de o singură adresă per coloană).

Coloanele *wlan.analysis.kck*, *wlan.analysis.kek*, *wlan.rsn.ie.gtk.key*, *wlan.rsn.ie.igtk.key*, *wlan.rsn.ie.pmkid* și *wlan_rsna_eapol.keydes.nonce* conțin diverse elemente criptografice

[1]https://github.com/mentalisttraceur/python-macaddress

specifice WPA2. Acestea au fost transformate în format numeric prin aplicarea secvenței următoare, conservând unicitatea valorilor inițiale și reducând necesarul de memorie comparativ cu o conversie directă din hexazecimal:

```
df[col+'_hashed'] = df[col].apply(lambda val: int(hashlib.sha1(str(val).encode(
    ↪ 'utf-8')).hexdigest(), 16) % (10 ** (16 if len(str(val)) == 32 else 32)))
```

Alte transformări

1. Proprietate eliminată: *frame.encap_type* conținea exclusiv valoarea 23 în cadrul întregului set de date, fiind prin urmare redundantă.

2. Proprietate convertită: *wlan.fc.ds* conținea numere în format hexazecimal, a fost convertită în format numeric.

3. Proprietate modificată: *Label* conținea două variante pentru categoria *Kr00k*, care au fost unificate sub o singură denumire.

4. Proprietate modificată: *data.len* conținea în unele înregistrări valori multiple concatenate cu '-' în loc de o singură valoare. A fost modificată pentru a reține doar prima valoare în acest caz.

În final, setul de date folosit conține 15574909 înregistrări, din care 467505 (aproximativ 3%) sunt anomalii.

Pe parcursul utilizării acestuia, a fost împărțit în subseturi de antrenare, validare și testare în mod stratificat după etichetă pentru a menține distribuția claselor în mod proporțional în fiecare subset, folosind *train_test_split* și standardizat folosind *StandardScaler* din *scikit-learn* [6]. Un singur obiect *StandardScaler* a fost antrenat pe întregul set de date, excluzând coloana cu etichetele țintă, pentru a obține o prelucrare uniformă a datelor.

Capitolul 4

Implementare

Am ales să folosesc metode ce aparțin domeniului învățării profunde (*deep learning*), o abordare ce folosește câteva arhitecturi diferite de rețea neurală pentru a obține clasificarea traficului atât binar cât și pe categorii de trafic, dar și o abordare ce folosește mai multe rețele neurale de tip *autoencoder* și diferite strategii de categorisire a informațiilor reconstituite de acestea pentru a obține clasificarea pe categorii de trafic.

4.1 Rețea neurală clasică

Prima abordare a constat în folosirea unor arhitecturi de rețele neurale, inspirate din [2], care au fost antrenate supervizat pe setul de date etichetat.

Arhitectura adaptată inițial conținea 5 straturi, din care am modificat primul strat pentru a avea dimensiunea de intrare corespunzătoare numărului de proprietăți (39 fără etichetă), dar și ultimul strat în funcție de tipul de clasificare abordat: (Figurile 4.1 și 4.2).

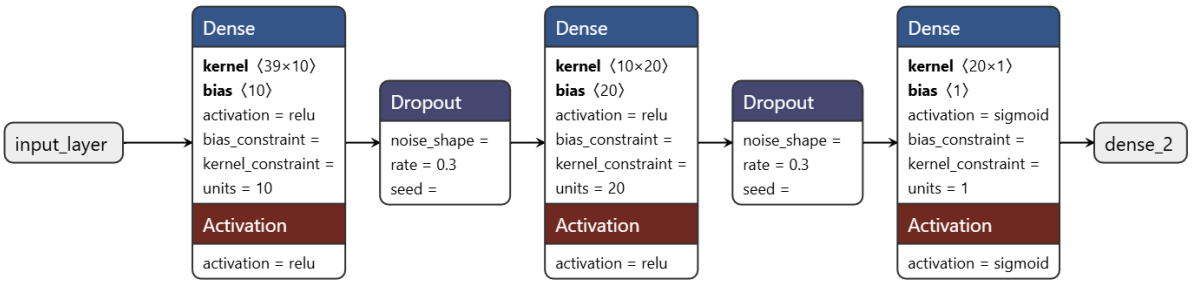

Figura 4.1: Arhitectura inițială - Dimensiune de ieșire 1, pentru clasificarea binară a traficului, funcție de activare sigmoidă

De asemenea, datele au fost prelucrate puțin diferit pentru fiecare tip de clasificare; pentru clasificarea binară etichetele traficului normal au fost înlocuite cu 0 iar restul cu 1, în timp ce pentru clasificarea pe categorii etichetele au fost prelucrate folosind *LabelEncoder* și *to_categorical* din *scikit-learn* [6] și respectiv *TensorFlow* [5].

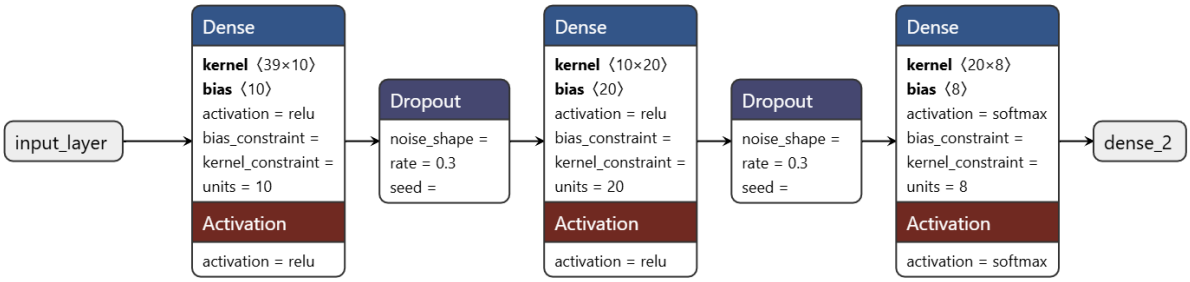

Figura 4.2: Arhitectura inițială - Dimensiune de ieșire 8, egală cu numărul de categorii, pentru clasificarea atacurilor pe categorii, funcție de activare softmax

Pentru clasificarea binară, ieșirea modelului este un singur neuron care utilizează funcția de activare sigmoid pentru a obține probabilitatea ca instanța să aparțină clasei anomaliilor, mapând scorul calculat pe un interval între 0 și 1. Apoi, pragul de 0.5 este folosit pentru a converti aceste scoruri la etichete de clasă 0 sau 1. Funcția sigmoid are forma:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

Pentru clasificarea atacurilor pe categorii, având 8 categorii în setul de date, stratul de ieșire al modelului conține 8 neuroni, fiecare reprezentând scorul pentru o categorie. Funcția de activare folosită este softmax, care normalizează scorurile pentru fiecare clasă astfel încât suma lor să fie 1, permițând interpretarea ca probabilități. Predicția finală este obținută prin găsirea indicelui clasei cu probabilitatea maximă. Funcția softmax este definită ca:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{4.2}$$

Experimentând cu diferite configurații pentru arhitecura rețelelor neurale folosite, am realizat că rezultatele sunt din ce în ce mai bune pe măsură ce scad ponderea straturilor *Dropout*. Inițial, aceste straturi au fost incluse în arhitectură cu scopul de a preveni fenomenul de *overfitting* și de a îmbunătăți capacitatea de generalizare a modelului.

Tabela 4.1: Raport de clasificare a atacurilor pe categorii

Clasă	Suport	Model fără starturi *Dropout*			Model cu rata *Dropout* 0.3		
		Precizie	Recall	F1-score	Precizie	Recall	F1-score
Normal	1510740	1.00	1.00	1.00	1.00	1.00	1.00
Kr00k	19181	1.00	1.00	1.00	1.00	1.00	1.00
Evil_Twin	10483	1.00	1.00	1.00	0.99	0.96	0.98
Disas	7513	1.00	1.00	1.00	1.00	0.99	0.99
Krack	4999	0.96	0.93	0.94	0.86	0.79	0.82
Deauth	3894	1.00	1.00	1.00	0.78	1.00	0.88
(Re)Assoc	550	1.00	1.00	1.00	0.79	0.63	0.70
RogueAP	131	1.00	1.00	1.00	0.00	0.00	0.00
medie		0.99	0.99	0.99	0.80	0.80	0.80
medie ponderată		1.00	1.00	1.00	1.00	1.00	1.00

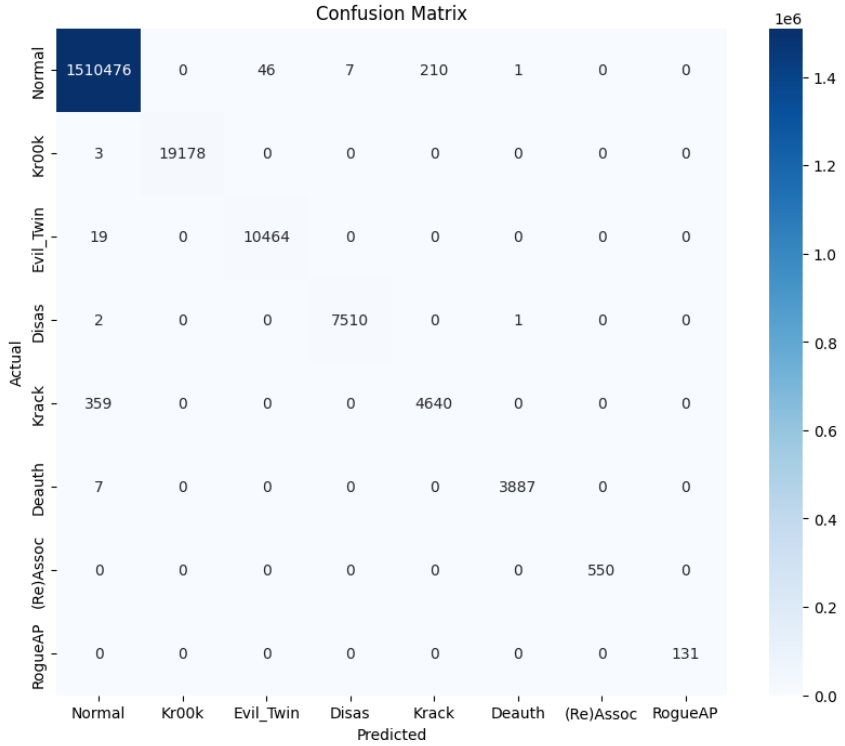

Figura 4.3: Matrice confuzie - Model fără starturi *Dropout*

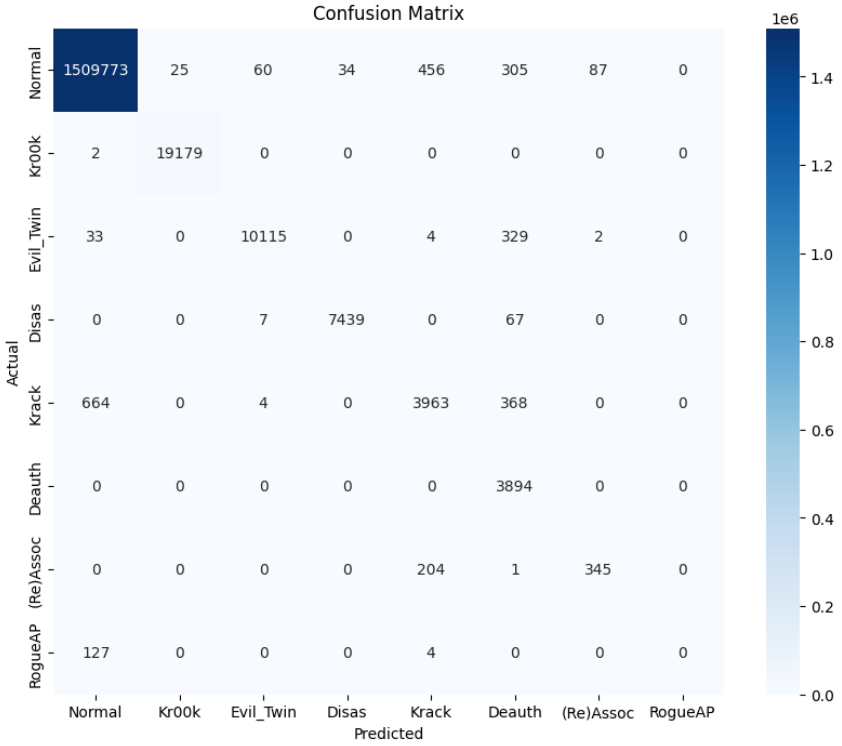

Figura 4.4: Matrice confuzie - Model cu rata *Dropout* 0.3

Tabela 4.2: Raport de clasificare binară a traficului

Clasă	Suport	Model fără starturi *Dropout*			Model cu rata *Dropout* 0.3		
		Precizie	Recall	F1-score	Precizie	Recall	F1-score
Normal	1510741	1.00	1.00	1.00	1.00	1.00	1.00
Malicious	46750	0.99	1.00	0.99	0.98	0.99	0.98
medie		1.00	1.00	1.00	0.99	0.99	0.99
medie ponderată		1.00	1.00	1.00	1.00	1.00	1.00

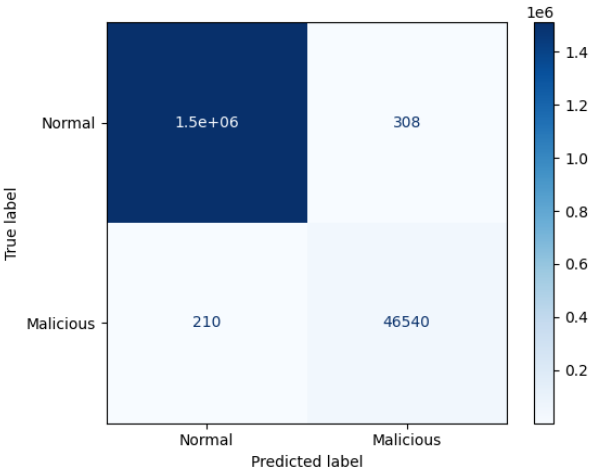

Figura 4.5: Model fără starturi *Dropout*

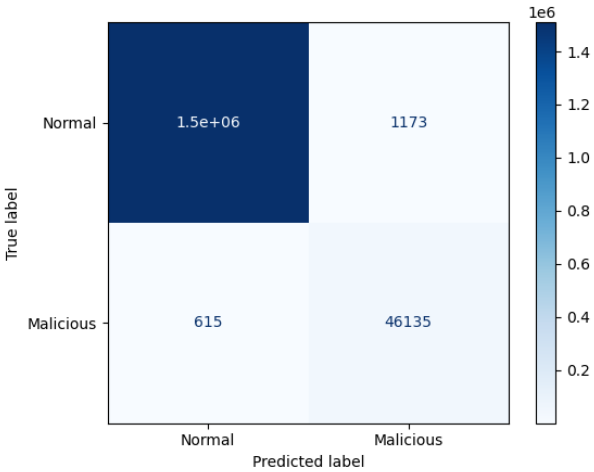

Figura 4.6: Model cu rata *Dropout* 0.3

Cele mai bune rezultate pentru ambele tipuri de clasificare au fost obținute prin eliminarea completă a straturilor *Dropout* din arhitecturile rețelelor neurale (Tabelele 4.1, 4.2; Figurile 4.3, 4.4, 4.5, 4.6). Astfel, arhitecturile finale conțin doar 3 straturi: (Figurile 4.7, 4.8)

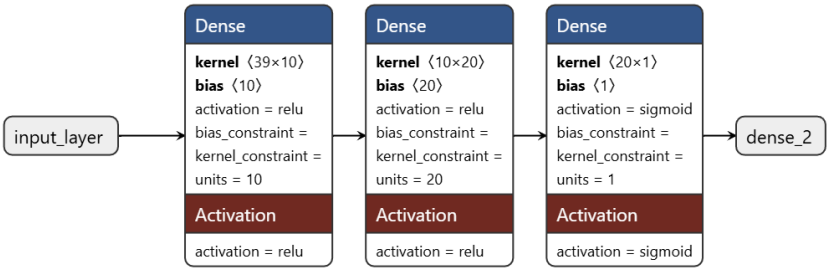

Figura 4.7: Arhitectura finală - Clasificare binară, fără *Dropout*

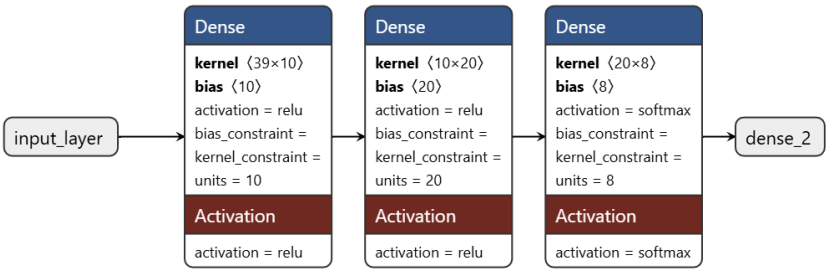

Figura 4.8: Arhitectura inițială - Clasificare pe categorii, fără *Dropout*

Hiperparametrii folosiți pentru antrenare au fost:

- *batch_size* = 2048
- *epochs* = 50
- *loss* = 'categorical_crossentropy' pentru clasificarea pe categorii sau 'binary_crossentropy' pentru clasificarea binară
- *optimizer* = 'adam'
- *EarlyStopping*:
  - Valoare monitorizată: acuratețea pe setul de validare (*val_accuracy*)
  - *patience* = 3
  - *restore_best_weights* = True

4.2 Autoencoder

A doua abordare a implicat antrenarea nesupervizată a mai multor rețele de tip autoencoder, câte una pentru fiecare categorie de trafic, doar pe datele etichetate ca și categoria respectivă.

Astfel, după ce datele au fost împărțite în subseturi de antrenare, validare și test în mod stratificat după etichete, subseturile de antrenare și validare au fost divizate suplimentar, în funcție de categoria exemplelor cuprinse. Prin urmare, pentru fiecare categorie de trafic existentă în set, am obținut două subseturi separate, unul pentru antrenare și altul pentru validare. Acest proces a permis antrenarea modelelor pe fiecare subset corespunzător categoriei respective, în mod izolat, independent de celelalte categorii.

Inițial, fiindcă numărul de proprietăți al setului de date este 39, am experimentat cu o arhitectură cu 15 straturi, în care partea codificatoare conținea 7 straturi (6 de tip *Dense* intercalate cu straturi de tip *Dropout*) ce comprimă datele de la dimensiunea inițială la 32, 16 și 8 unități, iar apoi la 4 unități în stratul codificării, și invers în partea decodificatoare, cu încă 7 (din care 6 straturi intercalate identic), pentru a ajunge înapoi la dimensiunea inițială. Această arhitectură a avut performanțe scăzute, iar experimental am realizat că pe măsură ce numărul de straturi scade, performanțele se îmbunătățesc, erorile de reconstrucție fiind din ce în ce mai mici.

Pentru a verifica aceste observații, am implementat o arhitectură intermediară în care am eliminat și straturile *Dropout*. Aceasta conținea 7 straturi, în care partea codificatoare conține 3 straturi (ce comprimă datele de la dimensiunea inițială la 32 și respectiv 16 unități) și apoi la 8 unități în stratul codificării, cu încă 3 straturi identice în partea decodificatoare pentru a ajunge înapoi la dimensiunea inițială. (Figura 4.9)

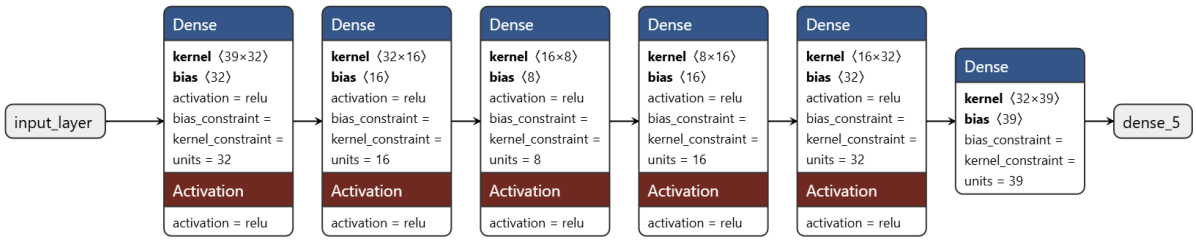

Figura 4.9: Arhitectura intermediară - Fără straturi *Dropout*

Așa am ajuns la arhitectura finală ce conține doar 5 straturi, în care partea codificatoare conține 2 straturi (ce comprimă datele de la dimensiunea inițială la 32 de unități) și apoi la 16 unități în stratul codificării, cu încă 2 straturi identice în partea decodificatoare pentru a ajunge înapoi la dimensiunea inițială. (Figura 4.10)

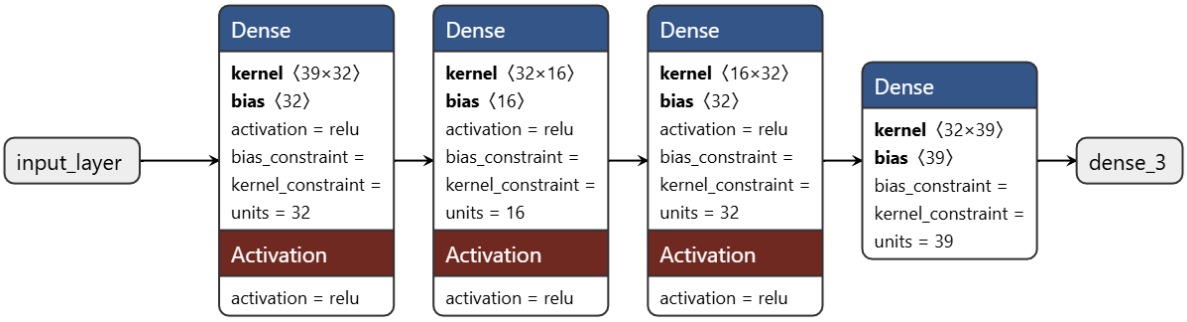

Figura 4.10: Arhitectura finală

Din tabelul 4.3 se poate observa că modelul cu arhitectura finală a avut pentru toate categoriile erorile de reconstrucție cele mai mici, acesta fiind utilizat mai departe pentru clasificare.

Tabela 4.3: Compararea erorilor de reconstrucție

Clasă	Best Epoch	Model intermediar	Best Epoch	Model final
Normal	1746	1.18550e-02	1560	2.08367e-03
Kr00k	5577	7.20589e-05	4664	3.52272e-05
Evil_Twin	6377	5.00988e-03	4352	9.29793e-04
Disas	9994	4.08687e-04	9989	4.09299e-05
Krack	10000	2.41950e-02	9999	8.35681e-03
Deauth	10000	6.51598e-04	10000	4.82967e-05
(Re)Assoc	10000	1.05994e-03	10000	6.16459e-04
RogueAP	9500	4.72845e-06	7556	8.07956e-06

Hiperparametrii folosiți pentru antrenare au fost:

- *batch_size* = 32768
- *epochs* = 10000
- *loss* = 'mse'
- *optimizer* = tf.keras.optimizers.Adam(learning_rate=1e-4)
- *EarlyStopping*:
  - Valoare monitorizată: eroare de reconstrucție pe setul de validare (*val_loss*)
  - *patience* = 35
  - *mode* = 'min'
  - *restore_best_weights* = True

4.2.1 Clasificare folosind praguri

Prima metodă de clasificare a constat în căutarea unui prag pentru valorile reconstituite de fiecare autoencoder pentru fiecare categorie în parte. Am realizat acest lucru prin parcugerea a 250 de valori pentru praguri, de la 0 la maximul erorii de reconstrucție a fiecărei clase pe setul de validare, și selectarea celor care maximizează scorul F1 pe tot setul de date de validare.

Pentru a obține predicțiile, am calculat erorile de reconstrucție pe întregul set de testare folosind fiecare autoencoder antrenat. Astfel, pentru fiecare instanță din setul de testare, am stocat câte o eroare de reconstrucție pentru fiecare categorie. Aceste erori au fost ulterior utilizate pentru a determina clasa finală a fiecărei instanțe, comparându-le cu pragurile optime pentru fiecare categorie în parte, până când se găsește o clasă pentru care eroarea este mai mică decât pragul. (Figura 4.11)

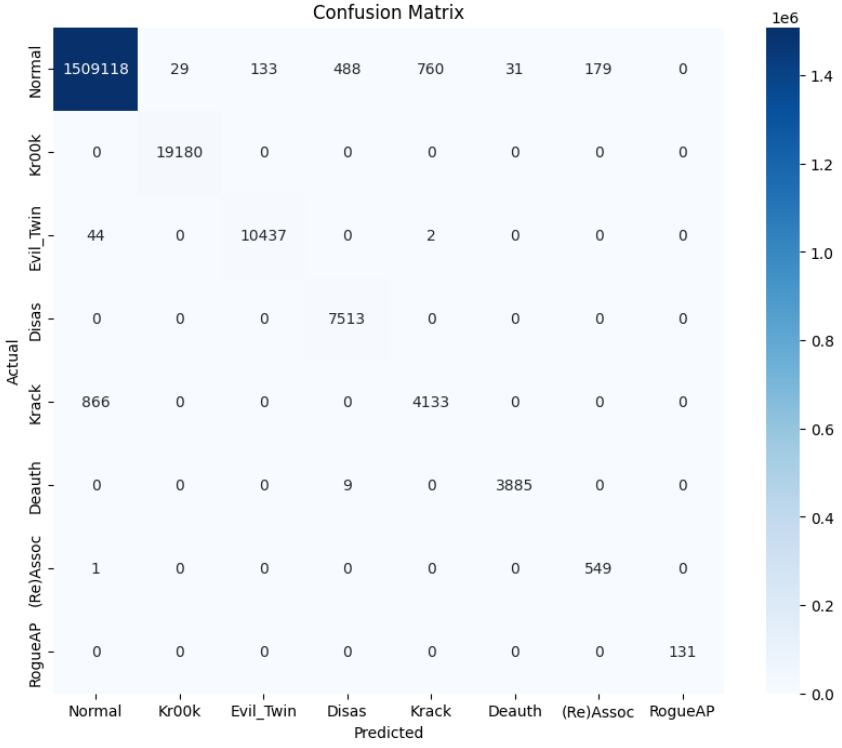

Figura 4.11: Matrice confuzie - Clasificare folosind praguri

4.2.2 Clasificare folosind distibuția de probabilitate normală

Pentru acest tip de clasificare setul de validare a fost divizat în funcție de categoria exemplelor cuprinse, astfel încât pentru fiecare categorie de trafic să avem câte un set de validare, doar cu instanțe din acea categorie.

Pentru aceste date, am calculat parametrii distribuțiilor normale folosind funcția *norm.fit()* din biblioteca SciPy[8] pe datele din fiecare clasă, ca să poată fi folosiți mai târziu pentru a calcula probabililtatea ca noi date să aparțină fiecărei clase.

Apoi, am calculat erorile de reconstrucție pe întregul set de testare folosind fiecare autoencoder antrenat, și am utilizat parametrii distribuțiilor calculați anterior pentru a determina probabilitățile ca fiecare instanță de test să aparțină fiecărei clase, utilizând funcția de densitate a probabilității *norm.pdf()*[8]

Pentru a obține predicțiile am ales pentru fiecare instanță din setul de test clasa cu probabilitatea cea mai mare. (Figura 4.12)

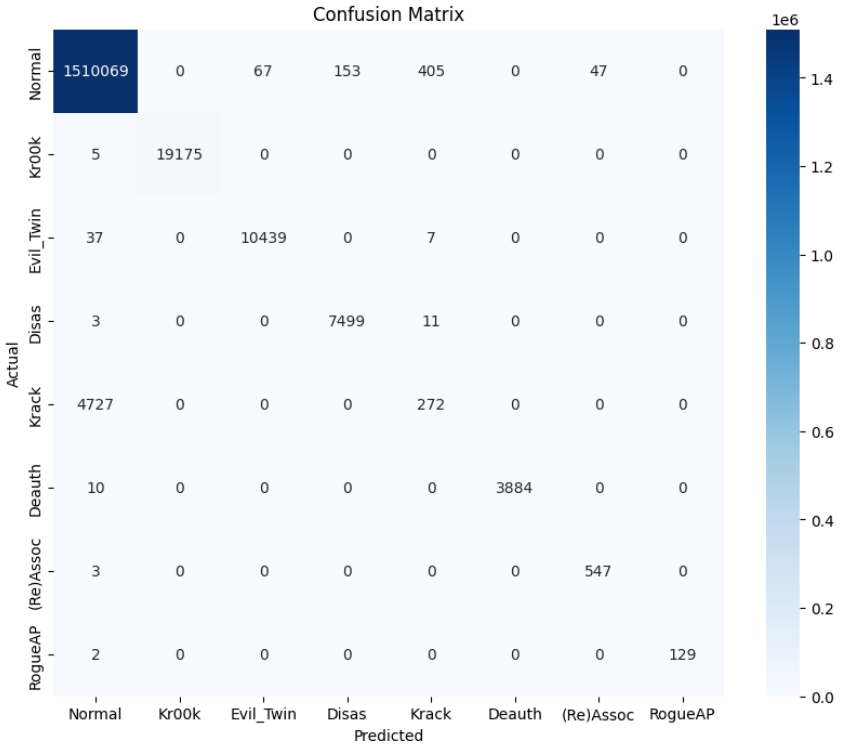

Figura 4.12: Matrice confuzie - Clasificare folosind distibuția normală

Din tabelul 4.4 și figurile 4.11, 4.12 reiese că metoda de clasificare cu praguri este metoda mai performantă din cele 2 studiate folosind rețele de tip autoencoder, conform mediei scorurilor F1 pe toate clasele.

Tabela 4.4: Raport de clasificare a atacurilor pe categorii

Clasă	Suport	Clasificare praguri			Clasificare dist. normală		
		Precizie	Recall	F1-score	Precizie	Recall	F1-score
Normal	1510741	1.00	1.00	1.00	1.00	1.00	1.00
Kr00k	19180	1.00	1.00	1.00	1.00	1.00	1.00
Evil_Twin	10483	0.99	1.00	0.99	0.99	1.00	0.99
Disas	7513	0.94	1.00	0.97	0.98	1.00	0.99
Krack	4999	0.84	0.83	0.84	0.39	0.05	0.10
Deauth	3894	0.99	1.00	0.99	1.00	1.00	1.00
(Re)Assoc	550	0.75	1.00	0.86	0.92	0.99	0.96
RogueAP	131	1.00	1.00	1.00	1.00	0.98	0.99
medie		0.94	0.98	0.96	0.91	0.88	0.88
medie ponderată		1.00	1.00	1.00	0.99	1.00	1.00

Capitolul 5

Concluzii

În această lucrare, am prezentat metode de detectare a atacurilor Wi-Fi utilizând tehnici avansate din domeniul învățării profunde (*deep learning*). Fiindcă tehnologiile de învățare profundă au cunoscut o dezvoltare rapidă în ultimii ani, am investigat folosirea atât a rețelelor neurale clasice, cât și de tip autoencoder pentru clasificarea traficului și identificarea anomaliilor.

Motivația principală a alegerii acestei teme a fost creșterea exponențială a dispozitivelor conectate la internet prin Wi-Fi și, implicit, creșterea numărului de utilizatori expuși la atacurile cibernetice care vizează aceste conexiuni. Atacurile asupra rețelelor Wi-Fi sunt relativ ușor de realizat și pot avea consecințe grave asupra securității utilizatorilor și a datelor lor.

Studiul a fost realizat pe setul de date AWID3 [4], un set de date conceput pentru a fi folosit în dezvoltarea de sisteme de detectare a intruziunilor în rețele Wi-Fi, care conține atacuri asupra protocolului de securitate WPA2, pe rețele ce au activat PMF (*Protected Management Frames*) și în medii care folosesc EAP (*Extensible Authentication Protocol*).

Setul de date a fost prelucrat pentru a reduce dimensiunea și a elimina informațiile redundante sau irelevante pentru detectarea atacurilor doar la nivelul 2 pe stiva OSI. Transformările aplicate asupra datelor, cum ar fi codificarea adreselor MAC și a altor elemente relevante reprezentate ca string-uri sau conversia valorilor cu relevanță mică în valori booleane au fost esențiale pentru îmbunătățirea performanței modelelor.

Din rezultatele prezentate în secțiunea practică, putem trage concluzia că metoda cea mai performantă de clasificare a atacurilor este utilizând rețele neurale clasice, care au obținut cele mai bune scoruri F1 în medie, așa cum reiese din tabelul 5.1.

Tabela 5.1: Compararea scorurilor F1

Clasă	Rețea neurală clasică F1-score	AE - praguri F1-score	AE - dist. normală F1-score
Normal	1.00	1.00	1.00
Kr00k	1.00	1.00	1.00
Evil_Twin	1.00	0.99	0.99
Disas	1.00	0.97	0.99
Krack	0.94	0.84	0.10
Deauth	1.00	0.99	1.00
(Re)Assoc	1.00	0.86	0.96
RogueAP	1.00	1.00	0.99
medie	0.99	0.96	0.88

Pe parcursul realizării acestei lucrări, am întâmpinat dificultăți legate de lipsa memoriei, necesară pentru prelucrarea întregului set de date în stagiile inițiale. Acest lucru a dus la dedicarea unei porțiuni foarte mari din timpul de realizare a lucrării la prelucrarea datelor, fiind nevoit să analizez și modific setul de date în forma lui inițială, împărțit în peste 300 de fișiere.

5.1 Direcții viitoare

Direcții viitoare de dezvoltare pentru această temă ar putea fi:

1. Aplicarea și a altor metode de detectare a anomaliilor pe acest set de date, atât supervizate cât și nesupervizate, cum ar fi *Support Vector Machines*, *Isolation Forest* sau *Local Outlier Factor*, existând posibilitatea să se obțină rezultate la fel de bune, eventual cu o complexitate a implementărilor mai redusă.

2. Dezvoltarea pe baza celor prezentate în această lucrare sau a altor metode a unui sistem de detecție și clasificare a atacurilor în timp real, care să poată fi rulat pe dispozitive IoT cu capacitate redusă de procesare, cum ar fi un router sau hub pentru soluții *Smart Home*.

În final, această lucrare demonstrează eficiența metodelor *deep learning* pentru clasificarea atacurilor în rețele Wi-Fi, și poate servi ca punct de plecare pentru dezvoltarea unui sistem de detecție și clasificare a acestor atacuri în timp real.

Bibliografie

[1] *AWID - Aegean Wi-Fi Intrusion Dataset*, https://icsdweb.aegean.gr/awid/awid3, (Accesat la data 11.06.2024).

[2] Diwash Bikram Basnet și Hisham A.; Advisor Kholidy, *An Empirical Wi-Fi Intrusion Detection System*, 2020, URL: http://hdl.handle.net/20.500.12648/1603.

[3] George Chatzisofroniou și Panayiotis Kotzanikolaou, „Association Attacks in IEEE 802.11: Exploiting WiFi Usability Features", în *Socio-Technical Aspects in Security and Trust*, ed. de Thomas Groß și Theo Tryfonas, Cham: Springer International Publishing, 2021, pp. 107–123, ISBN: 978-3-030-55958-8.

[4] E. Chatzoglou, G. Kambourakis și C. Kolias, „Empirical evaluation of attacks against IEEE 802.11 enterprise networks: The AWID3 dataset", în *IEEE Access* 9 (2021), pp. 34188–34205, DOI: 10.1109/ACCESS.2021.3061609.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu și Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015, URL: https://www.tensorflow.org/.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot și E. Duchesnay, „Scikit-learn: Machine Learning in Python", în *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] Mathy Vanhoef și Eyal Ronen, „Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd", în *IEEE Symposium on Security & Privacy (SP)*, IEEE, 2020.

[8] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt și SciPy 1.0 Contributors, „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", în *Nature Methods* 17 (2020), pp. 261–272, DOI: 10.1038/s41592-019-0686-2.