Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра

на тему: «Проєкт програмного рішення для виявлення аномалій безпеки трафіку систем на базі промислових Modbus-подібних протоколів»

Виконала: студентка 4-го курсу групи КБ-22-1
спеціальності 125 «Кібербезпека»
_____ Вікторія САВЧУК
Керівник: завідувач кафедри КІ та КБ, к.т.н, доцент
Андрій ЄФІМЕНКО
Рецензент: доцент кафедри КІ та КБ, к.т.н,
_____ Олексій ШЕЛУХА

Житомир – 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 125 «Кібербезпека»
Освітня програма «Кібербезпека»

ЗАТВЕРДЖУЮ
Завідувач
кафедри комп'ютерної
інженерії та кібербезпеки
Андрій Єфіменко
2 лютого 2026 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу бакалавра

Студентка: Савчук Вікторія Сергіївна

Тема роботи: «Проєкт програмного рішення для виявлення аномалій безпеки трафіку систем на базі промислових Modbus-подібних протоколів»
затверджена Наказом університету від 2 лютого 2026 р. № 58/с

Термін здачі студентом закінченої роботи 8 червня 2026 р.

Вихідні дані роботи:

Об'єктом дослідження є процес розробки програмного рішення для виявлення аномалій безпеки в мережах на базі Modbus-подібних протоколів.

Предметом дослідження є методи аналізу мережевого трафіку, виявлення аномалій безпеки, аналізу Modbus TCP-повідомлень та детекції кібератак у промислових мережах засобами Python.

Обладнання та програмне забезпечення:

1. Oracle VM VirtualBox.
2. Ubuntu 22.04 LTS, Kali Linux, Windows 10.
3. OpenPLC Runtime, ScadaBR.
4. Python 3.10.12, Scapy 2.5.0, Netmiko, PyModbus.
5. Wireshark, SSH.
6. Інструменти аналізу та автоматизації: Modbus-cli, Scapy, PyModbus, SSH/Netmiko.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Лайчук А.А.	6 березня 2026	6 березня 2026
2	Лайчук А.А.	2 квітня 2026	2 квітня 2026
3	Лайчук А.А.	29 квітня 2026	29 квітня 2026

Студент _____ Вікторія САВЧУК

Керівник _____ Андрій ЄФІМЕНКО

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання та аналіз об'єкта дослідження	06 березня 2026	Виконано
2	Дослідження архітектури та вразливостей протоколу Modbus/TCP	08 березня 2026	Виконано
3	Аналіз загроз кібербезпеки в промислових мережах(ICS/SCADA)	12 березня 2026 – 16 березня 2026	Виконано
4	Розробка моделі загроз для промислового контролера(PLC)	17 березня 2026 – 20 березня 2026	Виконано
5	Проєктування стенда для аналізу трафіку промислових протоколів	21 березня 2026 – 25 березня 2026	Виконано
6	Розробка алгоритму виявлення аномалій у трафіку Modbus та експериментальне дослідження ефективності засобів захисту	01 квітня 2026 – 27 квітня 2026	Виконано
8	Аналіз результатів та оцінка захищеності системи	01 травня 2026 – 03 травня 2026	Виконано
9	Оформлення пояснювальної записки	25 травня 2026 – 28 травня 2026	Виконано
10	Підготовка доповіді та презентації	29 травня 2026 – 31 травня 2026	Виконано
11	Отримання відгуку та рецензії на кваліфікаційну роботу		
12	Захист	10 червня 2026	

Студент _____ Вікторія САВЧУК

Керівник _____ Андрій ЄФІМЕНКО

АНОТАЦІЯ

Кваліфікаційна робота складається з проєкту віртуального полігону для виявлення аномалій безпеки та моделювання кібератак у мережах на базі промислового протоколу Modbus TCP, а також пояснювальної записки до нього. Пояснювальна записка містить 110 сторінок, 52 рисунки.

Метою роботи є розробка програмного рішення для виявлення аномалій безпеки мережевого трафіку в системах на базі протоколу Modbus TCP шляхом створення віртуального стенду, аналізу мережевого трафіку та моделювання кібератак на систему керування технологічним процесом.

У роботі обґрунтовано актуальність захисту автоматизованих систем управління технологічними процесами (АСУ ТП) та об'єктів критичної інфраструктури. Проаналізовано принцип роботи протоколу Modbus TCP, його особливості та ризики безпеки, пов'язані з відсутністю механізмів шифрування й автентифікації.

Також у роботі обґрунтовано вибір програмного забезпечення для віртуалізації, зокрема середовища VirtualBox, емулятора контролера OpenPLC, SCADA-системи ScadaBR та дистрибутива Kali Linux для моделювання атак. На базі зазначених компонентів розроблено та налаштовано модель промислової мережі.

Практична частина роботи містить реалізацію сценаріїв кібератак типу «Людина посередині» (MITM), перехоплення мережевого трафіку (Sniffing), підміни даних (ARP Spoofing) та моделювання аномальної мережевої активності. На основі аналізу мережевого трафіку розроблено IDS-модуль для виявлення аномалій безпеки у мережі Modbus TCP. Наведено результати тестування, які підтверджують небезпеку використання незахищених промислових протоколів у відкритих мережах та ефективність розробленого програмного рішення.

Ключові слова: MODBUS TCP, OPENPLC, SCADABR, KALI LINUX, MITM, ARPSPOOFING, IDS, SCAPY, NETMIKO, PYMODBUS.

					Державний університет «Житомирська політехніка» 26.125.14.000 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Савчук В.С.		4.06.25	Проєкт програмного рішення для виявлення аномалій безпеки трафіку систем на базі промислових Modbus-подібних протоколів	Літ.	Арк. 4 / Аркушів 110
Керівник		Єфіменко А.С.		4.06.25			
Реценз.		Шелуха О.О.		04.06.25		ФІКТ, Гр. КБ-22-1	
Затверд.		Єфіменко А.С.		4.06.25			

ANNOTATION

The qualification work consists of a project of a virtual testbed for analyzing security anomalies and modeling cyberattacks targeting the industrial Modbus TCP protocol, as well as an explanatory note. The explanatory note contains 110 pages and 52 illustrations.

The aim of the work is to develop a software solution for detecting security anomalies in network traffic of systems based on the Modbus TCP protocol through the creation of a virtual environment, network traffic analysis, and simulation of cyberattacks on a technological process control system.

The work substantiates the relevance of protecting industrial control systems (ICS) and critical infrastructure facilities. The operating principles of the Modbus TCP protocol, its features, and security risks related to the absence of encryption and authentication mechanisms are analyzed.

The paper also substantiates the choice of virtualization software, including the VirtualBox environment, the OpenPLC controller emulator, the ScadaBR SCADA system, and the Kali Linux distribution for attack simulation. Based on these components, a model of an industrial network was developed and configured.

The practical part of the work includes the implementation of Man-in-the-Middle (MITM) attack scenarios, traffic interception (Sniffing), data spoofing (ARP Spoofing), and simulation of anomalous network activity. Based on network traffic analysis, an IDS module for detecting security anomalies in a Modbus TCP network was developed. The testing results demonstrate the risks of using unsecured industrial protocols in open networks and confirm the effectiveness of the developed software solution.

Keywords: MODBUS TCP, OPENPLC, SCADABR, KALI LINUX, MITM, ARP SPOOFING, IDS, SCAPY, NETMIKO, PYMODBUS.

ЗМІСТ

ВСТУП 8

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ ПОБУДОВИ ТА ЗАБЕЗПЕЧЕННЯ КІБЕРЗАХИСТУ ПРОМИСЛОВИХ МЕРЕЖ ПОБУДОВАНИХ З ВИКОРИСТАННЯМ ПРОТОКОЛУ MODBUS 10

1.1 Аналіз принципів побудови промислових мереж та SCADA систем 10

1.2 Аналіз принципів організації функціонування та проблем безпеки протоколу Modbus 15

1.3 Аналіз вразливостей та кіберзагроз для промислових мереж та систем SCADA 22

1.4 Аналіз методів виявлення аномалій та мережевих вторгнень для промислових мереж та SCADA систем 27

РОЗДІЛ 2 ПРОЄКТ ВІРТУАЛЬНОГО СТЕНДУ ТА ПРОГРАМНОГО РІШЕННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ БЕЗПЕКИ ТРАФІКУ ПРОМИСЛОВОЇ СИСТЕМИ НА БАЗІ MODBUS ПОДІБНОГО ПРОТОКОЛУ .. 36

2.1 Проєкт віртуального стенду та програмного рішення для виявлення аномалій безпеки трафіку промислової системи на базі Modbus подібного протоколу 36

2.2 Проєктування протоколу промислової системи на базі Modbus подібного протоколу 41

2.3 Проєкт підсистеми захисту промислової системи на базі Modbus подібного протоколу 45

2.4 Проєкт програмного модуля для виявлення аномалій безпеки та моделювання атак у трафіку Modbus-подібного протоколу 48

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ВІРТУАЛЬНОГО СТЕНДУ ТА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ БЕЗПЕКИ В ТРАФІКУ MODBUS-ПОДІБНИХ ПРОТОКОЛІВ 53

3.1 Обґрунтування вибору середовища та практична реалізація віртуального стенду 53

3.2 Практична реалізація промислової системи програмного модуля та підсистеми захисту..... 70
3.3 Моделювання атак на промислову систему..... 74
3.4 Тестування стійкості розробленої системи підсистеми захисту..... 77
ВИСНОВКИ 87
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..... 89
ДОДАТКИ 92

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСУ ТП – Автоматизована система управління технологічним процесом

ОС – Операційна система

ВМ – Віртуальна машина

ПЗ – Програмне забезпечення

IDS – Intrusion Detection System

IPS – Intrusion Prevention System

IP – Internet Protocol

SSH – Secure Shell

MAC – Media Access Control

TCP – Transmission Control Protocol

DPI – Deep Packet Inspection

CNN – Convolutional Neural Network

SVM – Support Vector Machine

DDoS – Distributed Denial-of-Service

ARP – Address Resolution Protocol

MITM – Man-in-the-Middle

PLC – Programmable Logic Controller

SCADA – Supervisory Control And Data Acquisition

ПЛК – Програмований логічний контролер

ICS – Industrial Control Systems

ВСТУП

Інтеграція сучасних інформаційних технологій у промислові системи управління (ICS) стала невіддільною складовою розвитку енергетики. Однак цифровізація технологічних процесів не лише підвищує ефективність, а й створює нові вектори загроз. Зростання кількості кібератак на об'єкти критичної інфраструктури становить особливу небезпеку в умовах воєнного стану в Україні, де енергетична система є однією з головних цілей ворога. Актуальність даної роботи полягає у дослідженні аномалій безпеки та кібератак у мережах на базі протоколу Modbus TCP, який, попри свою поширеність в українській енергетиці, не має вбудованих механізмів захисту та шифрування, що робить його вразливим до кібервпливу.

Актуальність теми. Дослідження ризиків безпеки та методів виявлення аномальної активності в мережах Modbus TCP набуває критичного значення, оскільки саме на його базі функціонує значна частина автоматизованих систем керування енергетикою України. Кібератаки на енергетичну інфраструктуру мають деструктивний вплив не лише на саму галузь, а й провокують каскадні наслідки для суспільства та економіки. Порушення енергопостачання загрожує функціонуванню стратегічних підприємств, об'єктів охорони здоров'я та систем життєзабезпечення, що в умовах гібридної війни завдає прямих збитків національній безпеці держави.

Метою кваліфікаційної роботи є реалізація віртуального стенду та розробка програмного рішення для виявлення аномалій безпеки мережевого трафіку в системах на базі протоколу Modbus TCP.

Встановлена мета обумовлює наступні завдання:

1. Обґрунтувати актуальність захисту автоматизованих систем керування АСУ ТП в енергетиці, що використовують протокол Modbus.
2. Розробити архітектуру та розгорнути віртуальний стенд для емуляції технологічного процесу енергосистеми.
3. Провести моделювання кібератак на протокол Modbus TCP та реалізувати засоби моніторингу й виявлення аномалій мережевого трафіку.

Об’єктом дослідження є процес забезпечення інформаційної безпеки в АСУ ТП енергетичної галузі.

Предметом дослідження є методи аналізу мережевого трафіку та програмні засоби виявлення аномалій і кібератак на промисловий протокол Modbus TCP.

У процесі виконання кваліфікаційної роботи було реалізовано віртуальний стенд для моделювання кібератак на енергетичні системи та розроблено засоби їх виявлення на рівні протоколу Modbus.

РОЗДІЛ 1 ТЕОРЕТИЧНІ АСПЕКТИ ПОБУДОВИ ТА ЗАБЕЗПЕЧЕННЯ КІБЕРЗАХИСТУ ПРОМИСЛОВИХ МЕРЕЖ ПОБУДОВАНИХ З ВИКОРИСТАННЯМ ПРОТОКОЛУ MODBUS

1.1 Аналіз принципів побудови промислових мереж та SCADA систем

Промислові системи керування (Industrial Control Systems, ICS) являють собою комплекс програмних та апаратних засобів, призначених для автоматизації, моніторингу та керування технологічними процесами в реальному часі. До складу ICS входять SCADA-системи, програмовані логічні контролери (PLC), віддалені термінальні пристрої (RTU), операторські станції, мережеве обладнання та польові пристрої у вигляді датчиків і виконавчих механізмів[1].

На відміну від традиційних IT-систем, промислові мережі орієнтовані насамперед на забезпечення безперервності технологічного процесу, відмовостійкості та роботи в режимі реального часу. Навіть короткочасне порушення роботи промислової мережі може призвести до аварійних ситуацій, зупинки виробництва або пошкодження обладнання[19,23].

Сучасні ICS-системи будуються на основі мереж Ethernet та TCP/IP, що значно спрощує інтеграцію промислового обладнання в загальну мережеву інфраструктуру підприємства. При цьому взаємодія між компонентами системи може здійснюватися через локальні мережі (LAN), Wide Area Network (WAN), радіоканали, оптоволоконні лінії зв’язку або супутникові канали передачі даних.

Важливою особливістю промислових систем є використання централізованого диспетчерського керування. Центральний SCADA-сервер забезпечує збір, обробку та візуалізацію інформації, отриманої від PLC та RTU-пристроїв, а також дозволяє оператору контролювати технологічний процес через HMI-інтерфейс[19].

Одним із ключових компонентів ICS є програмований логічний контролер (ПЛК) – це промисловий цифровий комп’ютер, який має посилену конструкцію та пристосований для управління виробничими процесами, такими як конвеєрні лінії, роботизовані пристрої або будь-яка діяльність, що вимагає високої надійності управління, простоти програмування та діагностики несправностей у процесі. ПЛК були вперше розроблені в автомобільній промисловості для забезпечення гнучких,

посилених та легко програмованих контролерів, що замінили б жорстко підключені реле, таймери та секвенсори. З того часу вони широко застосовуються як високонадійні контролери автоматизації, придатні для роботи в суворих умовах. ПЛК є прикладом «жорсткої» системи реального часу, оскільки вихідні результати повинні генеруватися у відповідь на вхідні умови протягом обмеженого часу. Порушення часових обмежень може призвести до небажаної або аварійної роботи системи. Загальна схема взаємодії програмованого логічного контролера з вхідними та вихідними сигналами наведена на рис. 1.1 [19].

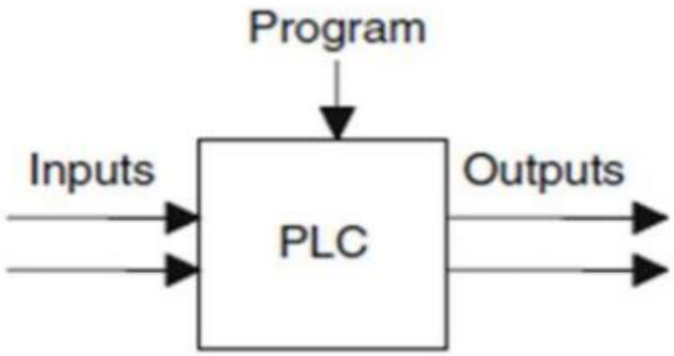


Рисунок 1.1 – Загальна схема взаємодії PLC із вхідними та вихідними сигналами

Він складається з центрального процесора (CPU), що містить системний мікропроцесор, пам'ять та схеми вводу-виводу. CPU керує та обробляє всі операції всередині ПЛК. Інформація всередині ПЛК передається за допомогою цифрових сигналів. Внутрішні шляхи, по яких протікають цифрові сигнали, називаються шинами. У фізичному сенсі шина – це просто низка провідників, по яких можуть протікати електричні сигнали. Це можуть бути доріжки на друкованій платі або дроти в плоскому кабелі. ЦП використовує шину даних для передачі даних між складовими елементами, шину адрес для передачі адрес місць для доступу до збережених даних, а також шину керування для сигналів, пов'язаних з внутрішніми діями керування. Системна шина використовується для зв'язку між портами вводу/виводу та блоком вводу/виводу. Структурна схема програмованого логічного контролера представлена на рис. 1.2[plc pdf].

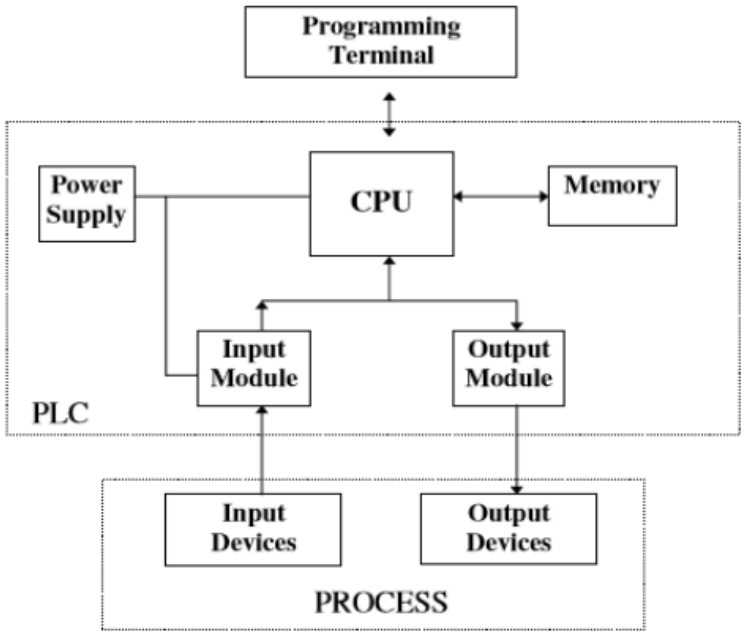


Рисунок 1.2 – Структурна схема програмованого логічного контролера

Переваги[23]:

1. Один і той самий базовий контролер можна використовувати з широким спектром систем управління.

2. Для модифікації системи управління потрібно змінити правила (що значно спрощує програмування та перепрограмування).

3. Немає потреби в переробці електропроводки.

4. У результаті ви отримуєте гнучку та економічно вигідну систему.

Оператор вводить послідовність команд (програму) у пам'ять ПЛК. Контролер відстежує вхідні сигнали та виконує правила керування. Контур керування – це безперервний цикл, під час якого ПЛК зчитує вхідні сигнали, виконує логічні операції, а потім змінює вихідні сигнали.

Програмовані логічні контролери є базовим елементом сучасних АСУ ТП та забезпечують безпосереднє керування технологічними процесами. Для централізованого моніторингу та диспетчеризації роботи ПЛК використовуються SCADA-системи.

В енергетичному секторі України ключовим компонентом АСУ ТП виступають SCADA-системи (Supervisory Control and Data Acquisition). Це комплекс програмного та апаратного забезпечення, що дозволяє збирати дані з віддалених пристроїв (сенсорів, контролерів), аналізувати їх та забезпечувати диспетчерам можливість управляти обладнанням у режимі реального часу. Сучасні

енергосистеми повністю керуються або активно використовують SCADA для моніторингу та оптимізації роботи всіх елементів – від генерації на електростанціях до розподільчих мереж. Це забезпечує надійність енергопостачання та мінімізує ризик операторських помилок[1,4].

SCADA-системи (Supervisory Control and Data Acquisition) використовуються для централізованого моніторингу та керування розподіленими промисловими об’єктами, де процес збору даних є таким же важливим, як і безпосереднє керування обладнанням. Подібні системи застосовуються в енергетичних мережах, системах водопостачання, газотранспортних системах, залізничній інфраструктурі та інших об’єктах критичної інфраструктури[3].

Основним призначенням SCADA-систем є збір даних із польових пристроїв, передача цієї інформації до центрального вузла керування та забезпечення оператора можливістю контролю технологічного процесу в режимі реального часу. Керування окремими процесами може виконуватись автоматично або за участю оператора через людино-машинний інтерфейс (HMI).

Типова SCADA-система складається з програмних та апаратних компонентів. До основних елементів належать центральний сервер керування (MTU), програмовані логічні контролери (PLC), віддалені термінальні пристрої (RTU), операторські станції, засоби комунікації та польове обладнання у вигляді датчиків і виконавчих механізмів. PLC та RTU виконують локальне керування технологічним процесом, тоді як центральний сервер здійснює збір, обробку та зберігання інформації[19].

Передача даних між компонентами SCADA-системи може здійснюватися через Ethernet-мережі, телефонні лінії, оптоволоконні канали, радіоканали або супутниковий зв’язок. Для обміну даними використовуються як стандартні, так і пропрієтарні промислові протоколи.

Важливою особливістю сучасних SCADA-систем є їхня відмовостійкість та наявність резервування основних компонентів. Це дозволяє підтримувати безперервність технологічного процесу навіть у випадку відмови окремих вузлів або виникнення аварійних ситуацій.

Загальна структура SCADA-системи наведена на рис. 1.3. Центральний вузол керування містить SCADA-сервер, HMI-інтерфейс, інженерні станції та систему

зберігання історичних даних (Data Historian). Взаємодія з польовими пристроями здійснюється через мережеву інфраструктуру глобальну мережу WAN(Wide Area Network). На польовому рівні використовуються PLC-контролери, RTU-пристрої та інтелектуальні електронні пристрої (IED), що забезпечують моніторинг та керування технологічними процесами.

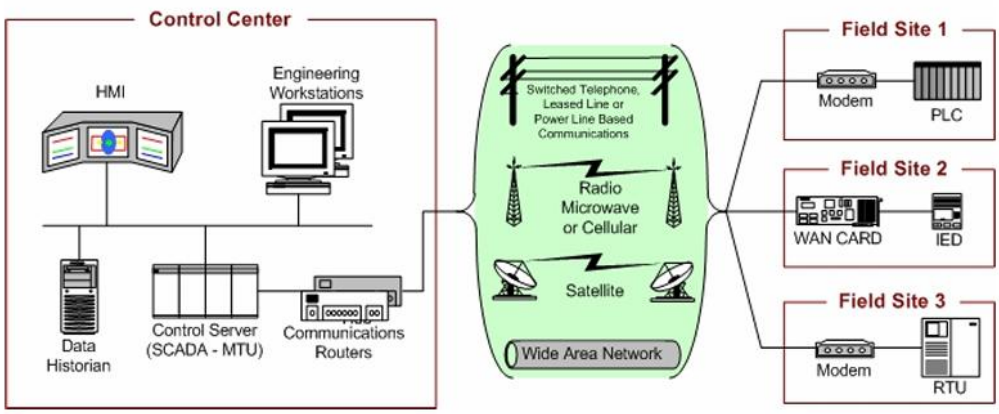


Рисунок 1.3 – Загальна структура SCADA-системи

Архітектура взаємодії між центральним вузлом керування та польовими пристроями може будуватись за різними топологіями, зокрема point-to-point, series, series-star та multi-drop. Основні варіанти організації зв'язку в SCADA-системах наведено на рис. 1.4[23,19].

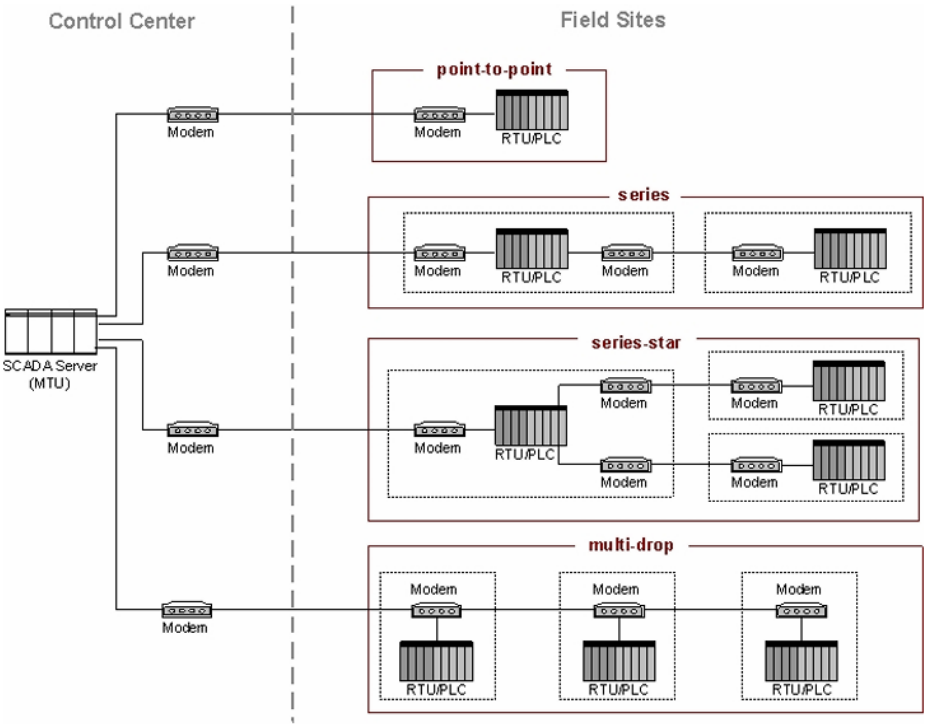


Рисунок 1.4 – Основні топології зв'язку в SCADA-системах

1.2 Аналіз принципів організації функціонування та проблем безпеки протоколу Modbus

Протокол Modbus був розроблений у 1979 році компанією Modicon Incorporated для систем промислової автоматизації та програмованих логічних контролерів Modicon. З того часу він став загальноприйнятим промисловим стандартом для передачі дискретних та аналогових даних вводу-виводу, а також даних регістрів у пристроях промислового керування та моніторингу. Наразі Modbus є широко поширеним відкритим протоколом, що належить до публічного домену. Для його використання не потрібно сплачувати роялті, що сприяло його широкому впровадженню в промислових системах[14,19].

Modbus – це протокол обміну повідомленнями прикладного рівня, який функціонує на 7-му рівні мережевої моделі OSI. Він забезпечує клієнт-серверну взаємодію між пристроями, підключеними до різних типів мереж та шин. На сьогодні протокол реалізується за допомогою:

1. TCP/IP через Ethernet.

2. Асинхронної послідовної передачі через різні середовища зв’язку (EIA/TIA-232-E, EIA-422, EIA/TIA-485-A, оптоволоконні лінії, радіоканали тощо).

3. Modbus Plus – високошвидкісної мережі з передачею маркера.

Протокол Modbus забезпечує простий та універсальний обмін даними в різних мережевих архітектурах. Різноманітні типи пристроїв, зокрема ПЛК, HMI-панелі, системи керування рухом, пристрої вводу-виводу та SCADA-системи, можуть використовувати Modbus для виконання дистанційних операцій та обміну даними як через послідовні інтерфейси, так і через Ethernet-мережі.

Протокол Modbus визначає просту одиницю даних протоколу (PDU), незалежну від базових рівнів зв’язку. Під час адаптації Modbus до конкретного середовища передачі даних до одиниці даних додатка (ADU) можуть додаватися додаткові поля. Обмін даними за протоколом Modbus може здійснюватися у двох режимах[14]:

1. Запит/відповідь – взаємодія між клієнтом та сервером.

2. Широкомовлення – клієнтський пристрій надсилає команду всім серверним пристроям одночасно.

Транзакція Modbus складається з одного кадру запиту або відповіді, а повідомлення містить адресу одержувача, код функції та дані, необхідні для виконання команди.

Для зв'язку між пристроями Modbus використовується архітектура «клієнт-сервер» (master/slave), у якій лише клієнтський пристрій може ініціювати транзакції. Серверні пристрої відповідають на запити, надаючи необхідні дані або виконуючи відповідні дії. У ролі серверних пристроїв можуть виступати перетворювачі вводу-виводу, датчики, клапани, накопичувачі даних та інші польові пристрої. Типовим клієнтським пристроєм є комп'ютер або SCADA-система, що виконує централізоване керування технологічним процесом[14].

Блок даних додатка Modbus (ADU) формується клієнтом, який ініціює транзакцію. Клієнт може звертатися як до окремого серверного пристрою, так і надсилати широкомовні повідомлення всім пристроям у мережі. Серверні пристрої повертають відповіді лише на індивідуально адресовані запити та не можуть самостійно ініціювати передачу повідомлень.

Запит клієнта містить адресу серверного пристрою, код функції, необхідні дані та поле перевірки помилок. Відповідь сервера містить підтвердження виконаної дії, дані у відповідь та поле контролю помилок. У випадку помилки або неможливості виконання команди сервер повертає повідомлення про виняток (Exception Response).

Традиційні повідомлення Modbus передаються послідовно, а для перевірки цілісності даних використовується контроль парності та механізми перевірки помилок у кадрі повідомлення[14].

Для реалізації обміну даними між пристроями в протоколі Modbus використовується клієнт-серверна архітектура взаємодії. Загальну схему обміну повідомленнями між клієнтом та сервером у протоколі Modbus наведено на рисунку 1.5.

Така модель взаємодії є зручною для промислових систем, оскільки дозволяє централізовано керувати обміном даними між SCADA-системою, PLC-контролерами і польовими пристроями. У типовій промисловій мережі клієнт надсилає запити на читання або запис даних, а сервер виконує відповідні операції з регістрами чи дискретними сигналами.

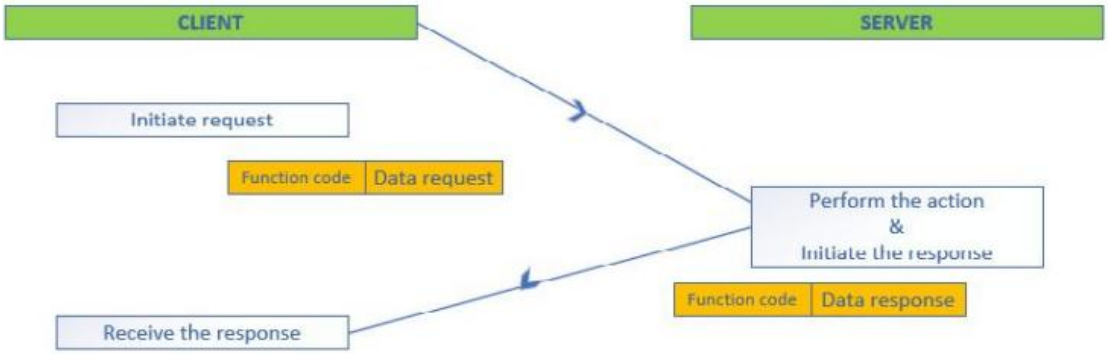


Рисунок 1.5 – Схема взаємодії клієнта та сервера у протоколі Modbus

У протоколі Modbus дані можуть бути представлені у вигляді дискретних входів, котушок, вхідних регістрів та регістрів зберігання. Дискретні значення використовуються для опису станів типу «увімкнено/вимкнено», а регістри застосовуються для передачі числових параметрів технологічного процесу.

Modbus, як прикладний протокол, визначає правила організації та інтерпретації даних, але залишається лише структурою обміну повідомленнями, незалежною від базового фізичного рівня. Як показано нижче на рисунку 1.6, він визначає протокольну одиницю даних (PDU), а іноді до одиниці прикладних даних (ADU) додаються додаткові поля, такі як додаткова адреса та перевірка на помилки. Загалом, він є простим для розуміння, вільно доступним для будь-кого та підтримується багатьма виробниками[14].

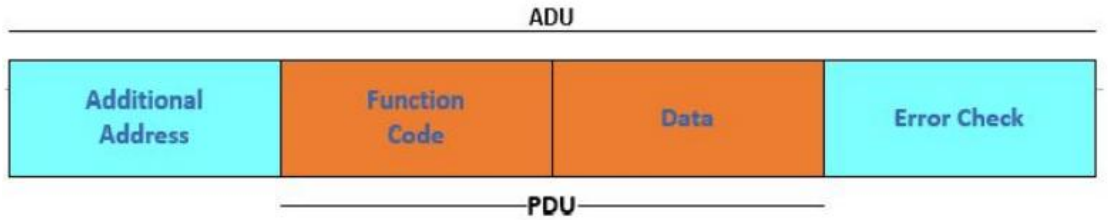


Рисунок 1.6 – Структура кадру Modbus ADU/PDU

Поле «Function code» кодується одним байтом, а допустимі значення коду знаходяться в діапазоні від 1 до 255 (у десятковій системі числення). У разі, якщо

клієнт надсилає повідомлення на сервер, код функції містить інформацію про тип дії, яку необхідно виконати. При цьому код функції «0» є недійсним. До деяких кодів функцій додаються коди підфункцій для визначення множинних дій[modbus].

Поле Data у повідомленні, надісланому від клієнта до серверного пристрою, містить додаткову інформацію, яку сервер використовує для виконання дії, визначеної кодом функції. Адреси регістрів, кількість елементів, що підлягають обробці, та кількість фактичних байтів даних у полі – це деякі елементи, що містяться в полі даних. У разі, якщо сервер не потребує жодної додаткової інформації, поле даних може бути відсутнім (нульової довжини)[14].

Modbus TCP/IP (також Modbus TCP) – це, по суті, протокол Modbus RTU з інтерфейсом TCP, що працює в мережі Ethernet. Структура обміну повідомленнями Modbus – це прикладний протокол, який визначає правила організації та інтерпретації даних незалежно від середовища передачі даних. TCP/IP означає Протокол управління передачею та Інтернет-протокол, які забезпечують середовище передачі для обміну повідомленнями Modbus TCP/IP. Іншими словами, TCP/IP дозволяє обмінюватися блоками двійкових даних між комп'ютерами. Це всесвітній стандарт, який слугує основою для Всесвітньої мережі. Основна функція TCP полягає в тому, щоб забезпечити правильний прийом усіх пакетів даних, тоді як IP гарантує, що повідомлення правильно адресовані та маршрутизовані. Комбінація TCP/IP є лише транспортним протоколом і не визначає, що означають дані або як їх слід інтерпретувати. Інтерпретація даних є завданням прикладного протоколу Modbus[14,20,5].

Підсумовуючи, Modbus TCP/IP використовує TCP/IP та Ethernet для передачі даних структури повідомлень Modbus між сумісними пристроями та поєднує фізичну мережу (Ethernet) зі стандартом мережевого зв’язку (TCP/IP) та стандартним методом представлення даних (Modbus як протокол додатка). Простіше кажучи, повідомлення Modbus TCP/IP – це комунікація Modbus, інкапсульована в оболонку Ethernet TCP/IP. На практиці Modbus TCP вбудовує стандартний кадр даних Modbus у кадр TCP без контрольної суми Modbus[modbus].

Команди Modbus та дані користувача інкапсулюються безпосередньо в контейнер даних телеграми TCP/IP без будь-яких змін. Однак поле перевірки помилок Modbus (контрольна сума) не використовується, оскільки для забезпечення цілісності даних застосовуються стандартні методи розрахунку контрольної суми на рівні канального зв'язку Ethernet TCP/IP. Крім того, поле адреси кадру Modbus замінюється ідентифікатором пристрою в Modbus TCP/IP і стає частиною заголовка протоколу додатків Modbus (MBAP). Код функції та поля даних залишаються у своєму первісному вигляді.

Таким чином, одиниця додатка Modbus TCP/IP (ADU) має вигляд 7-байтового заголовка, який складається з ідентифікатора транзакції, ідентифікатора протоколу, поля довжини та ідентифікатора пристрою, а також одиниці даних протоколу, яка складається з коду функції та даних. ADU вбудовується в поле даних стандартного кадру TCP і надсилається через TCP на системний порт 502. Цей порт спеціально зарезервовано для додатків Modbus, і всі сервери та клієнти отримують і прослуховують дані Modbus через нього[14].

Протокол Modbus визначає кілька функціональних кодів, кожен з яких відповідає певній команді. Зокрема, існує три категорії функціональних кодів MODBUS. Це:

1. Public Function Codes.

Це чітко визначені коди функцій, які:

1.1 Гарантовано є унікальними.

1.2 Перевірені спільнотою MODBUS.org.

1.3 Публічно задокументовані.

1.4 Мають доступні тести на відповідність.

Включають як визначені загальнодоступні коди функцій, так і незайняті коди функцій, зарезервовані для використання в майбутньому.

2. User-Defined Function Codes.

2.1 Існує два діапазони кодів функцій, що визначаються користувачем, а саме: від 65 до 72 та від 100 до 110 десятичний.

2.2 Користувач може вибрати та впровадити код функції, який не підтримується специфікацією.

2.3 Немає гарантії, що використання обраного коду функції буде унікальним.

2.4 Якщо користувач бажає перенести функціональність у категорію загальнодоступних кодів функцій, він повинен ініціювати RFC для внесення зміни до загальнодоступної категорії та отримання нового загальнодоступного коду функції.

3. Reserved Function Codes.

3.1 Коди функцій, які наразі використовуються деякими компаніями для застарілих продуктів і які не доступні для загального користування.

Протокол Modbus TCP був створений на основі Modbus RTU/ASCII, який спочатку розроблявся для використання в ізольованих послідовних мережах. Під час його створення питання кібербезпеки практично не враховувались, оскільки передбачалося, що доступ до пристроїв матимуть лише авторизовані технічні спеціалісти. Після інтеграції Modbus у TCP/IP-мережі ці недоліки стали суттєвими вразливостями, оскільки промислові пристрої отримали можливість віддаленого мережевого доступу[5,20,31].

У протоколі Modbus відсутні вбудовані механізми перевірки цілісності повідомлень на рівні прикладного протоколу. Унаслідок цього цілісність переданих даних залежить від нижчих рівнів мережевої моделі та не забезпечує належного захисту від модифікації пакетів.

Через відсутність механізмів захисту Modbus TCP дозволяє виконувати розвідувальні дії в мережі. Зловмисник може здійснювати сканування function codes, визначати підтримувані функції пристрою, тип обладнання та структуру регістрів PLC-контролера.

Однією з небезпечних загроз для промислових мереж на базі Modbus TCP є атаки типу Denial of Service (DoS). Такі атаки спрямовані на порушення доступності системи шляхом створення надмірного навантаження на мережеві пристрої або PLC-контролери. Перевантаження мережі може призводити до

збільшення часу відповіді Modbus-пристроїв, втрати пакетів та порушення роботи систем моніторингу й керування технологічним процесом[20,31].

Однією з небезпечних загроз для мереж на базі Modbus TCP є розподілені атаки типу DDoS, спрямовані на створення надмірного мережевого навантаження. Велика кількість запитів до PLC-контролерів або SCADA-серверів може призводити до збільшення затримок обробки пакетів, втрати мережевих повідомлень та перевантаження компонентів системи[8].

Особливістю промислових мереж є необхідність роботи в режимі реального часу, тому навіть короткочасне перевантаження мережі може негативно впливати на стабільність обміну даними між SCADA-системою та PLC-контролерами.

Основні проблеми безпеки протоколу Modbus TCP та їх можливі наслідки наведено в таблиці 1.1[5,20,31].

Таблиця 1.1

Основні проблеми безпеки протоколу Modbus TCP

Проблема	Наслідок
Відсутність шифрування	Перехоплення трафіку
Відсутність аутентифікації	Несанкціонований доступ
Відсутність контролю цілісності	Модифікація пакетів
Відкритий порт 502	Можливість сканування
Function code scanning	Розвідка мережі

Використання стандартних IT-мереж для промислових протоколів керування створило нові проблеми безпеки, пов’язані з особливостями функціонування промислових систем та їх вимогами до надійності й безперервності роботи. Одним із найбільш поширених промислових протоколів є MODBUS TCP/IP, який походить від застарілих систем послідовного зв’язку та не має вбудованих механізмів автентифікації, шифрування й контролю цілісності переданих даних.

Відсутність вбудованих засобів захисту робить протокол Modbus TCP вразливим до різних типів мережевих атак, зокрема перехоплення трафіку, модифікації пакетів, несанкціонованого доступу та DoS-атак. Оскільки MODBUS TCP/IP функціонує на прикладному рівні мережевої моделі, для його аналізу та моніторингу можуть використовуватися системи виявлення вторгнень, засоби Deep Packet Inspection та методи аномалійного аналізу мережевого трафіку.

У сучасних дослідженнях для виявлення підозрілої активності в мережах Modbus TCP застосовуються механізми аналізу function codes, кодів винятків та послідовності мережевих повідомлень. Такі підходи дозволяють виявляти нетипову поведінку системи та потенційні спроби реалізації кібератак[29,31,8].

Протокол Modbus TCP широко використовується у промислових системах, зокрема в енергетичних, водопостачальних та теплових мережах. Тому забезпечення належного рівня кіберзахисту та моніторингу мережевого трафіку є важливим завданням для підвищення безпеки SCADA-систем та PLC-контролерів[19,31].

1.3 Аналіз вразливостей та кіберзагроз для промислових мереж та систем SCADA

Загрози для систем управління можуть надходити з численних джерел, зокрема з боку ворожих урядів, терористичних угруповань, промислових шпигунів, незадоволених співробітників, кіберзлочинців, а також з природних джерел, таких як складність систем, людські помилки та нещасні випадки, відмови обладнання та природні катастрофи. Для захисту від ворожих загроз (а також відомих природних загроз) необхідно розробити стратегію багаторівневого захисту для ICS[19].

Зловмисники проникають у мережі заради гострих відчуттів від подолання виклику або щоб похвалитися перед спільнотою хакерів. Якщо раніше для віддаленого злому потрібні були чималі навички та знання в галузі інформатики, то зараз зловмисники можуть завантажувати з Інтернету скрипти та протоколи атак і запускати їх проти сайтів жертв. Отже, хоча інструменти для атак стали більш досконалими, вони також стали простішими у використанні. Багато зловмисників не мають необхідного досвіду, щоб загрожувати складним цілям, таким як критично важливі мережі США. Проте, загальна кількість зловмисників у світі становить відносно високу загрозу поодиноких або короткочасних зривів роботи, що можуть спричинити серйозні збитки.

Злочинні угруповання прагнуть атакувати системи з метою отримання грошової вигоди. Зокрема, угруповання організованої злочинності використовують

спам, фішинг та шпигунське/шкідливе програмне забезпечення для крадіжки особистих даних та здійснення інтернет-шахрайства[19].

Спамери – це особи або організації, які розсилають небажані електронні листи зі прихованою або неправдивою інформацією з метою продажу товарів, здійснення фішингових атак, розповсюдження шпигунського/шкідливого програмного забезпечення або здійснення атак на організації (наприклад, DoS).

Особи або організації зі злісними намірами здійснюють атаки на користувачів шляхом створення та поширення шпигунського та шкідливого програмного забезпечення. Низка руйнівних комп'ютерних вірусів та черв'яків завдала шкоди файлам і жорстким дискам, зокрема макровірус «Мелісса», черв'як «Explore.Zip», вірус CIH («Чорнобиль»), «Німда», «Код Ред», «Сламмер» та «Бластер».

Терористи прагнуть знищити, вивести з ладу або використати об'єкти критичної інфраструктури з метою створення загрози національній безпеці, спричинення масових жертв, ослаблення економіки США та підриву морального духу та довіри населення. Терористи можуть використовувати фішинг або шпигунське/шкідливе програмне забезпечення для залучення коштів або збору конфіденційної інформації. Терористи можуть атакувати одну ціль, щоб відвернути увагу або ресурси від інших цілей.

Промислове шпигунство має на меті здобуття інтелектуальної власності та ноу-хау таємними методами.

Необхідний метод оцінки та класифікації ризику, пов'язаного з можливою вразливістю на конкретному об'єкті. Ризик є функцією ймовірності того, що визначений суб'єкт загрози (супротивник) зможе скористатися конкретною вразливістю та спричинити наслідки. На ризик, спричинений будь-якою конкретною вразливістю, впливає низка пов'язаних показників, зокрема[19]:

1. Архітектура та стан мережі й комп'ютерів.
2. Встановлені засоби захисту.
3. Технічна складність атаки.

4. Імовірність виявлення (наприклад, час, протягом якого супротивник може залишатися на зв’язку з цільовою системою/мережею, не будучи виявленим).

5. Наслідки інциденту.

6. Вартість інциденту.

У системах промислового контролю значна частина вразливостей пов’язана не лише з технічними недоліками, але й з організаційними проблемами, зокрема відсутністю належної документації, політик безпеки, аудитів та планів аварійного відновлення. Основні організаційні вразливості ICS наведено в таблиці 1.2[19].

Таблиця 1.2

Вразливості політик та процедур безпеки ICS

Вразливість	Опис
Недосконала політика безпеки для ICS	Уразливості в системах промислового контролю (ICS) часто виникають через неналежні правила або відсутність правил, спеціально розроблених для забезпечення безпеки систем управління.
Відсутність офіційної програми навчання та інформування з питань безпеки ICS	Документально оформлена офіційна програма навчання та підвищення обізнаності з питань безпеки призначена для того, щоб персонал був у курсі останніх змін у політиках та процедурах безпеки організації, а також галузевих стандартів кібербезпеки та рекомендованих практик. Без навчання щодо конкретних політик та процедур ICS не можна очікувати, що персонал зможе забезпечити безпеку середовища ICS.
Недосконала архітектура та структура системи безпеки	Інженери з систем управління традиційно отримували мінімальну підготовку з питань безпеки, і аж до нещодавнього часу виробники не включали функції безпеки у свої продукти
На основі політики безпеки для ICS не було розроблено жодних конкретних або задокументованих процедур безпеки	Необхідно розробити спеціальні процедури безпеки та провести навчання співробітників щодо ІКС. Саме вони є основою надійної програми безпеки.
Відсутність або недостатність обладнання ICS рекомендації щодо впровадження	Інструкції з експлуатації обладнання повинні бути актуальними та легко доступними. Ці інструкції є невід’ємною частиною процедур безпеки на випадок порушення роботи ІКС.
Відсутність адміністративних механізмів забезпечення безпеки	Співробітники, відповідальні за забезпечення безпеки, повинні нести відповідальність за дотримання задокументованих політик та процедур безпеки.

Продовження табл.1.2

Вразливість	Опис
Аудити безпеки в ICS проводяться рідко або взагалі не проводяться	Незалежні аудити безпеки повинні передбачати перевірку та аналіз записів і операцій системи з метою визначення адекватності засобів контролю системи та забезпечення дотримання встановлених політик і процедур безпеки ІКС. Аудити також повинні використовуватися для виявлення порушень у роботі служб безпеки ІКС та надання рекомендацій щодо змін, які можуть включати посилення існуючих засобів контролю та/або впровадження нових засобів контролю.
Відсутність спеціального плану забезпечення безперебійної роботи або плану відновлення після аварій (DRP)	План відновлення після аварій (DRP) має бути розроблений, перевірений та доступний на випадок серйозної аварії апаратного чи програмного забезпечення або руйнування об’єктів. Відсутність спеціального плану відновлення після аварій для ІКС може призвести до тривалих простоїв та втрат у виробництві.
Відсутність управління змінами, специфічного для ICS	Слід запровадити процедуру контролю змін у апаратному забезпеченні, вбудованому програмному забезпеченні, програмному забезпеченні та документації, щоб забезпечити захист ІКС від неналежних або неправильних змін до, під час та після впровадження системи. Відсутність процедур управління змінами конфігурації може призвести до недоглядів у сфері безпеки, вразливостей та ризиків.

Однією з поширених проблем безпеки SCADA-систем є недостатньо надійні механізми автентифікації. Значна частина промислових систем спочатку проєктувалася для роботи в ізольованих мережах, тому сучасні механізми захисту доступу не враховувались під час їх розробки. Це створює ризик несанкціонованого доступу до компонентів системи та можливість виконання несанкціонованих операцій над технологічним процесом[24,20].

Для зменшення ризику компрометації SCADA-систем рекомендується використовувати багатофакторну автентифікацію, рольову модель доступу (RBAC) та політики складних паролів із регулярною зміною облікових даних.

Багато SCADA-систем використовують промислові протоколи без вбудованих механізмів шифрування, що створює ризик перехоплення або модифікації мережевого трафіку. Передача даних у відкритому вигляді дозволяє зловмиснику аналізувати мережеві пакети, змінювати команди керування та здійснювати атаки типу Man-in-the-Middle[5,31].

Для підвищення рівня безпеки рекомендується використовувати VPN-з'єднання, сегментацію мережі, міжмережеві екрани та засоби виявлення вторгнень (IDS), здатні аналізувати аномальну мережеву активність у промислових мережах[26,8,29].

Додатковою проблемою безпеки SCADA-систем є використання застарілого програмного забезпечення та legacy-компонентів, які можуть містити невиправлені вразливості. У багатьох випадках оновлення або модифікація таких систем є складною через необхідність забезпечення безперервності технологічного процесу.

Крім технічних недоліків, суттєвий вплив на безпеку ICS має людський фактор. Недостатній рівень підготовки персоналу, помилки конфігурації, використання слабких паролів та відсутність постійного моніторингу мережевої активності можуть значно підвищити ризик компрометації системи[24,19].

Одними з найбільш поширених атак на промислові мережі є атаки типу Man-in-the-Middle (MITM), DoS-атаки та підміна мережевих пакетів. У випадку MITM-атаки зловмисник перехоплює мережевий трафік між SCADA-системою та PLC-контролером, що дозволяє аналізувати або змінювати передані дані.

DoS-атаки спрямовані на порушення доступності компонентів ICS шляхом створення надмірного мережевого навантаження. Для промислових систем подібні атаки є особливо небезпечними, оскільки навіть короткочасне порушення зв'язку може призвести до аварійних ситуацій або порушення технологічного процесу.

Основні кіберзагрози, характерні для SCADA та ICS-систем, наведено в таблиці 1.3.

Таблиці 1.3

Основні кіберзагрози для SCADA та ICS-систем

Тип загрози	Опис загрози	Можливі наслідки
MITM	Перехоплення та модифікація мережевого трафіку	Підміна команд керування
DoS/DDoS	Створення надмірного мережевого навантаження	Порушення доступності системи
ARP Spoofing	Підміна ARP-відповідей у локальній мережі	Перенаправлення трафіку через зловмисника
Replay Attack	Повторна передача перехоплених пакетів	Повторне виконання команд

Продовження табл.1.3

Тип загрози	Опис загрози	Можливі наслідки
Несанкціонований доступ	Отримання доступу до SCADA або PLC	Зміна параметрів технологічного процесу
Malware	Використання шкідливого ПЗ	Порушення роботи ICS-систем

1.4 Аналіз методів виявлення аномалій та мережних вторгнень для промислових мереж та SCADA систем

Система виявлення вторгнень (Intrusion Detection System, IDS) – це програмне або програмно-апаратне рішення, призначене для моніторингу мережевого трафіку та виявлення несанкціонованої або підозрілої активності в інформаційній системі. Основним завданням IDS є аналіз мережевих подій з метою виявлення спроб вторгнення, порушень політик безпеки та потенційних кібератак[25].

Системи IDS здійснюють контроль вхідного та вихідного мережевого трафіку, аналізують поведінку вузлів мережі та виявляють аномальні дії, які можуть свідчити про наявність загроз безпеці. У разі виявлення підозрілої активності система формує відповідне повідомлення або сигнал тривоги для адміністратора безпеки[26].

Правильне налаштування IDS є важливим фактором ефективності її функціонування. Для коректного виявлення атак система повинна аналізувати нормальний режим роботи мережі та формувати модель типової поведінки мережевого трафіку. У разі неправильного налаштування IDS може генерувати хибні спрацювання (False Positive), помилково визначаючи легітимний трафік як шкідливий.

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) є важливим елементом забезпечення кібербезпеки промислових мереж та SCADA-систем. Основним призначенням IDS є моніторинг мережевого трафіку або стану окремих вузлів з метою виявлення підозрілої активності, порушень політик безпеки та потенційних кібератак. Використання IDS у промислових мережах дозволяє своєчасно виявляти аномалії, що можуть негативно впливати на стабільність технологічного процесу[6].

Залежно від способу функціонування та області моніторингу системи IDS поділяються на декілька типів.

Основні типи систем виявлення вторгнень наведено на рис. 1.7[25].

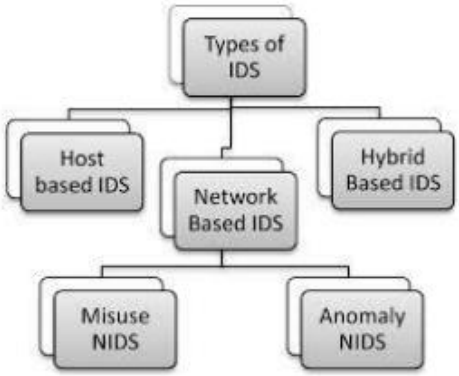


Рисунок 1.7 – Класифікація систем виявлення вторгнень IDS

Як показано на рис. 1.7, системи IDS можуть класифікуватися залежно від способу моніторингу мережі та методів аналізу мережевого трафіку.

Мережеві системи виявлення вторгнень (Network Intrusion Detection System, NIDS) здійснюють аналіз мережевого трафіку в окремих сегментах мережі. Такі системи встановлюються у ключових точках мережевої інфраструктури та контролюють пакети даних, що передаються між пристроями. NIDS аналізує вміст пакетів, мережеві заголовки та інші параметри трафіку для виявлення підозрілої активності. У разі виявлення аномалії система формує повідомлення для адміністратора безпеки. Використання NIDS є особливо актуальним для промислових мереж на базі Modbus TCP, оскільки дозволяє виявляти несанкціоновані запити, сканування function codes, ARP Spoofing та аномальне мережеве навантаження[20,26,31].

Хостові системи виявлення вторгнень (Host Intrusion Detection System, HIDS) встановлюються безпосередньо на окремих вузлах мережі та здійснюють моніторинг їхнього внутрішнього стану. HIDS аналізує журнали подій, зміни системних файлів, процеси та локальний мережевий трафік. У разі виявлення несанкціонованих змін або підозрілих дій система генерує відповідне попередження. Такі рішення можуть використовуватися для контролю стану SCADA-серверів або робочих станцій операторів.

Протокольно-орієнтовані системи виявлення вторгнень (Protocol-based IDS, PIDS) призначені для аналізу мережевих протоколів та контролю правильності їх функціонування. PIDS здійснює перевірку мережевих повідомлень на відповідність специфікації протоколу та дозволяє виявляти аномальні або модифіковані пакети. Для промислових мереж особливо важливим є контроль промислових протоколів, зокрема Modbus TCP, оскільки зловмисник може використовувати некоректні function codes або змінювати параметри технологічного процесу[20,31].

Окремим різновидом IDS є гібридні системи виявлення вторгнень (Hybrid IDS), які поєднують можливості декількох методів моніторингу. Такі системи можуть одночасно аналізувати мережевий трафік, стан окремих вузлів та специфіку роботи промислових протоколів. Використання гібридного підходу дозволяє підвищити точність виявлення кібератак та зменшити кількість помилкових спрацювань[8,29].

Для виявлення вторгнень у сучасних IDS використовуються сигнатурні та аномалійні методи аналізу.

Сигнатурний метод виявлення вторгнень (Signature-based Detection) базується на використанні заздалегідь визначених шаблонів або сигнатур атак. Система IDS аналізує мережевий трафік та порівнює його з базою відомих шаблонів шкідливої активності. У разі збігу формується повідомлення про виявлення атаки. Основною перевагою сигнатурного аналізу є висока точність виявлення відомих загроз. Однак цей метод має недолік, оскільки не дозволяє ефективно виявляти нові або модифіковані атаки, сигнатури яких відсутні в базі даних.

Аномалійний метод виявлення вторгнень (Anomaly-based Detection) базується на аналізі поведінки мережі та визначенні відхилень від нормального режиму функціонування системи. IDS формує модель нормального мережевого трафіку та контролює зміну його характеристик. До ознак аномальної активності можуть належати різке збільшення кількості пакетів, поява нетипових запитів Modbus TCP, використання нехарактерних function codes або несанкціонована взаємодія між вузлами мережі. Перевагою аномалійного аналізу є можливість

виявлення нових типів атак, однак недоліком є більша кількість хибних спрацювань[6,8,29,31].

Основні типи систем виявлення вторгнень наведено в табл. 1.4.

Таблиця 1.4

Основні типи систем виявлення вторгнень IDS

Тип IDS	Призначення	Особливості
NIDS (Network IDS)	Аналіз мережевого трафіку	Контролює пакети в мережі та виявляє мережеві атаки
HIDS (Host IDS)	Моніторинг окремого вузла	Аналізує системні файли, журнали подій та процеси
PIDS (Protocol IDS)	Контроль мережевих протоколів	Аналізує коректність роботи протоколів
APIDS (Application Protocol IDS)	Контроль прикладних сервісів	Виявляє атаки на рівні застосунків
Hybrid IDS	Комбінований аналіз	Поєднує декілька методів моніторингу та аналізу

Як наведено в табл. 1.4, різні типи IDS відрізняються способом моніторингу мережі та рівнем аналізу мережевого трафіку[25,26].

У сучасних системах кіберзахисту промислових мереж часто застосовується гібридний підхід, який поєднує сигнатурний та аномалійний аналіз. Це дозволяє забезпечити ефективне виявлення як відомих, так і нових загроз у мережах SCADA та системах промислової автоматизації.

Розроблений у кваліфікаційній роботі IDS-модуль використовує елементи аномалійного аналізу мережевого трафіку шляхом контролю інтенсивності пакетів, аналізу function codes протоколу Modbus TCP та виявлення нехарактерної мережевої активності[31,29].

Окрім систем виявлення вторгнень IDS, у сучасних мережах також використовуються системи запобігання вторгненням IPS (Intrusion Prevention System). IPS є розвитком IDS-технологій та поєднує функції моніторингу мережевого трафіку з можливістю активного реагування на виявлені загрози[26].

Основною відмінністю IPS від IDS є здатність не лише виявляти підозрілу активність, а й автоматично блокувати потенційні атаки або змінювати параметри безпеки системи. Таким чином, IPS дозволяє зменшити ризик успішної реалізації мережевих атак та мінімізувати негативний вплив інцидентів на роботу інформаційної системи.

Системи IPS можуть використовувати різні механізми реагування на загрози. До основних способів протидії атакам належать:

1. Блокування мережевого з'єднання або користувацької сесії.
2. Блокування IP-адреси джерела атаки.
3. Заборона доступу до цільового сервера або сервісу.
4. Автоматична зміна конфігурації мережевих пристроїв безпеки.
5. Фільтрація або модифікація шкідливого мережевого трафіку.

У промислових мережах IPS можуть використовуватися для захисту PLC-контролерів, SCADA-серверів та мережевої інфраструктури від атак типу DoS/DDoS, ARP Spoofing, несанкціонованих Modbus-запитів та інших загроз безпеці[20,31].

Однією з важливих особливостей систем IDS/IPS є наявність помилкових спрацювань. У випадку, коли система помилково визначає легітимний трафік як шкідливий, виникає False Positive. Якщо ж IDS або IPS не виявляє реальну атаку, виникає False Negative. Для зменшення кількості таких помилок виконується налаштування та оптимізація параметрів системи виявлення вторгнень[26].

Сучасні IDS/IPS-системи також здатні інтегруватися з централізованими системами моніторингу та SIEM-рішеннями, що дозволяє здійснювати централізований аналіз подій безпеки та оперативно реагувати на кіберінциденти[8].

У сучасних інформаційних та промислових мережах використовується значна кількість програмних рішень для виявлення та запобігання мережевим вторгненням. Такі системи забезпечують моніторинг мережевого трафіку, аналіз подій безпеки та виявлення підозрілої активності в режимі реального часу.

Одним із поширених рішень є SolarWinds Security Event Manager (SEM), який поєднує можливості мережевих та хостових систем виявлення вторгнень. Система дозволяє здійснювати централізований збір журналів подій, аналіз мережевого трафіку та автоматичне реагування на інциденти безпеки.

Компанія McAfee також пропонує IDS/IPS-рішення, які використовують сигнатурний та аномалійний методи аналізу мережевого трафіку. Такі системи

здатні виявляти шкідливу активність у режимі реального часу та забезпечують масштабованість для великих корпоративних мереж[26].

Серед відкритих систем виявлення вторгнень особливе місце займає Suricata. Це безкоштовна система мережевого моніторингу класу NIDS, яка забезпечує аналіз мережевого трафіку та виявлення відомих загроз за допомогою сигнатурного аналізу. Suricata підтримує роботу в режимі реального часу та широко використовується для моніторингу мережевої безпеки.

Для централізованого аналізу подій безпеки та виявлення складних атак використовуються SIEM-рішення. Одним із прикладів є платформа Blumira, яка дозволяє здійснювати безперервний моніторинг IT-інфраструктури, аналіз подій безпеки та виявлення підозрілої активності як у хмарних середовищах, так і у локальних мережах[8].

Ще одним прикладом сучасної системи моніторингу є Cisco Stealthwatch, яка поєднує можливості мережевого та хостового аналізу. Система використовує методи машинного навчання для формування моделей нормальної поведінки мережі та здатна виявляти аномальну активність навіть у зашифрованому мережевому трафіку.

Таким чином, сучасні IDS/IPS та SIEM-системи є важливими компонентами захисту промислових мереж та SCADA-систем, оскільки дозволяють своєчасно виявляти кіберзагрози та реагувати на потенційні інциденти безпеки.

У сучасних промислових мережах все більшого поширення набувають методи аномалійного аналізу мережевого трафіку, які дозволяють виявляти нові та невідомі типи атак. На відміну від сигнатурних систем, аномалійні методи не потребують попередньо сформованої бази сигнатур та базуються на аналізі відхилень від нормальної поведінки мережі[8,9,29].

Для мереж на базі протоколу Modbus TCP характерною особливістю є відносно передбачуваний характер обміну даними між SCADA-системою та PLC-контролерами. Завдяки цьому можливо формувати модель нормального мережевого трафіку та виявляти аномалії шляхом аналізу інтенсивності пакетів,

структури мережевих повідомлень, частоти використання function codes та інших параметрів протоколу.

Сучасні дослідження у сфері захисту промислових мереж також розглядають використання методів машинного навчання для виявлення мережевих вторгнень. Для аналізу мережевого трафіку Modbus TCP застосовуються алгоритми Random Forest, Support Vector Machine (SVM), Decision Tree та штучні нейронні мережі. Такі методи дозволяють автоматично формувати модель нормальної поведінки мережі та виявляти нетипові події, які можуть свідчити про реалізацію кібератаки[9].

Окремим напрямом розвитку систем виявлення вторгнень є використання методів Deep Packet Inspection (DPI), які дозволяють аналізувати вміст мережевих пакетів на прикладному рівні. Для промислових протоколів, зокрема Modbus TCP, це дає можливість контролювати function codes, параметри запитів, значення регістрів та інші елементи протоколу з метою виявлення підозрілої активності[20,31].

Поєднання сигнатурного аналізу, аномалійного моніторингу та методів машинного навчання дозволяє підвищити ефективність виявлення загроз у промислових мережах та забезпечити більш надійний захист SCADA-систем і PLC-контролерів.

Сучасні системи виявлення аномалій можуть використовувати методи машинного навчання для аналізу мережевого трафіку та формування моделі нормальної поведінки системи. Такі підходи дозволяють виявляти невідомі типи атак та аномальні відхилення у роботі промислової мережі.

Для аналізу вразливостей протоколу Modbus TCP використовуються спеціалізовані засоби тестування безпеки, одним із яких є Modbus/TCP Fuzzer (MTF). Даний інструмент призначений для виявлення помилок та вразливостей у реалізації протоколу Modbus шляхом генерації некоректних або аномальних мережевих пакетів[20,16].

MTF формує модифіковані пакети Modbus TCP із помилковими параметрами та надсилає їх до PLC-контролерів або SCADA-систем з метою перевірки стійкості

системи до некоректних запитів. Одним із основних завдань такого тестування є виявлення можливих відмов у роботі пристроїв, зависань, помилок обробки пакетів та DoS-атак.

Використання fuzzing-підходу дозволяє оцінити рівень захищеності промислових систем та виявити потенційні проблеми безпеки ще до впровадження системи в реальне виробниче середовище.

Для виявлення мережевих атак у промислових мережах можуть використовуватися сигнатурні правила IDS-систем. Одним із найбільш поширених рішень є система Snort, яка забезпечує аналіз мережевого трафіку та виявлення порушень безпеки на основі сигнатурного аналізу[26].

Для протоколу Modbus TCP були розроблені спеціалізовані правила виявлення вторгнень, що дозволяють виявляти:

1. Сканування function codes.
2. Аномальні запити до регістрів PLC.
3. Атаки типу DoS.
4. Malformed Modbus ADU packets.
5. Надмірне мережеве навантаження.
6. Несанкціоновані операції запису в регістри.

Сигнатурний аналіз дозволяє ефективно виявляти відомі типи атак, однак його ефективність залежить від актуальності бази сигнатур та правильності налаштування правил IDS.

Для аналізу можливих сценаріїв атак на промислові мережі можуть використовуватися дерева атак (Attack Tree). Даний підхід дозволяє моделювати можливі дії зловмисника та визначати найбільш критичні вразливості системи[31].

Метод Attack Tree використовується для аналізу протоколів, мережевої інфраструктури та SCADA-систем з метою оцінки потенційних шляхів реалізації кібератак. Використання такого підходу дозволяє визначати найбільш ефективні механізми захисту та мінімізувати ризики порушення роботи промислової системи.

Протокол Modbus TCP є вразливим до flooding-атак, які спрямовані на створення надмірного мережевого навантаження. Велика кількість запитів до PLC-

контролерів або SCADA-систем може призводити до збільшення затримок, втрати пакетів та порушення стабільності технологічного процесу.

Для виявлення flooding-атак використовуються сигнатурні та аномалійні методи аналізу мережевого трафіку. Сигнатурні методи базуються на контролі інтенсивності мережевого трафіку та порогових значень кількості пакетів, тоді як аномалійний аналіз дозволяє виявляти нетипову поведінку мережі та відхилення від нормального режиму функціонування системи[8,31].

РОЗДІЛ 2 ПРОЄКТ ВІРТУАЛЬНОГО СТЕНДУ ТА ПРОГРАМНОГО РІШЕННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ БЕЗПЕКИ ТРАФІКУ ПРОМИСЛОВОЇ СИСТЕМИ НА БАЗІ MODBUS ПОДІБНОГО ПРОТОКОЛУ

2.1 Проєкт віртуального стенду та програмного рішення для виявлення аномалій безпеки трафіку промислової системи на базі Modbus подібного протоколу

Віртуальний стенд призначений для моделювання промислової мережі на базі протоколу Modbus TCP, генерації мережевого трафіку, реалізації атак та тестування програмного рішення для виявлення аномалій безпеки[19,23].

Для реалізації практичної частини роботи було обрано метод віртуалізації. Це дозволяє створити безпечне, ізольоване середовище для моделювання кібератак без ризику пошкодження реального фізичного обладнання або порушення роботи діючих мереж.

Для побудови віртуального полігону (стенду) було обрано наступний стек програмних продуктів:

1. Середовище віртуалізації - Oracle VM VirtualBox. Вибір даного гіпервізора обумовлений такими факторами[23]:

1.1 ПЗ розповсюджується за ліцензією GNU GPL, що робить його безкоштовним та доступним для освітніх цілей.

1.2 VirtualBox дозволяє створювати віртуальні мережі (Internal Network, Host-only), що дає змогу емулювати реальну топологію промислової мережі, ізольовану від Інтернету.

1.3 Має функцію знімків стану, що є критично важливо для досліджень, яка дозволяє миттєво повернути систему до початкового стану після проведення деструктивної атаки.

2. Емулятор ПЛК – Ubuntu 22.04 LTS з OpenPLC В якості платформи для розгортання емулятора контролера (OpenPLC Runtime) обрано операційну систему Ubuntu 22.04 LTS (Jammy Jellyfish)[2]:

2.1 OpenPLC – це перше повністю відкрите програмне забезпечення, що підтримує стандарт IEC 61131-3, який використовується в реальній промисловості.

2.2 OpenPLC має нативну підтримку протоколу Modbus TCP, виступаючи в ролі Modbus Server (Slave), що є необхідною умовою для дослідження вразливостей цього протоколу[15].

2.3 Індекс LTS (Long Term Support) гарантує довгострокову підтримку та стабільність ядра системи, що є критично важливим для імітації безперервного технологічного процесу.

2.4 Ця версія має повну сумісність із бібліотеками, необхідними для коректної компіляції та роботи OpenPLC v3.

3. Атакуючі вузли – Kali Linux 2021 Для виконання функцій тестування на проникнення (Penetration Testing) використано спеціалізований дистрибутив Kali Linux версії 2021[16]:

3.1 Це галузевий стандарт для фахівців з кібербезпеки, що містить сотні попередньо встановлених інструментів для тестування на проникнення.

3.2 Дана версія містить стабільний та перевірений набір утиліт для аудиту безпеки (Metasploit Framework, Nmap, Wireshark), версії яких гарантовано підтримують скрипти для тестування протоколу Modbus.

3.3 Обрана версія відзначається збалансованим споживанням ресурсів віртуальної машини при виконанні активного сканування мережі.

4. Робоче місце оператора (SCADA-клієнт) – Windows 10 Для емуляції автоматизованого робочого місця (АРМ) диспетчера обрано операційну систему Windows 10[3,17,15]:

4.1 Вона повноцінно підтримує протокол Modbus TCP, що дозволяє отримувати дані з емулятора OpenPLC.

4.2 Дозволяє побудувати мнемосхему процесу (графіки, індикатори, кнопки керування), що необхідно для наочної демонстрації наслідків кібератак (наприклад, втрати керування або підміни показників датчиків).

4.3 На цій ОС розгорнуто клієнтське програмне забезпечення (ScadaBR), яке виступає в ролі Modbus Master, надсилаючи запити до контролера.

4.4 Переважна більшість клієнтського програмного забезпечення SCADA та HMI-інтерфейсів у реальній енергетиці функціонує саме в середовищі Windows.

Для побудови віртуального стенду було обрано набір програмних компонентів, що забезпечують емуляцію роботи промислової мережі, SCADA-системи, PLC-контролера та вузлів атакуючої сторони. Характеристика основних компонентів стенду наведена в табл. 2.1[2,3,16,22].

Таблиця 2.1

Характеристика програмного забезпечення стенду

Компонент стенду	Програмний засіб	Роль у дослідженні
Гіпервізор	Oracle VM VirtualBox	Забезпечення віртуалізації та мережевої ізоляції
Жертва(ПЛК)	OpenPLC (на Ubuntu)	Емуляція логіки технологічного процесу та протоколу Modbus
Атакуючі вузли	Kali Linux	Генерація аномального трафіку, MITM-атаки, ARP Spoofing та аналіз мережевої активності
Моніторинг	Wireshark	Перехоплення та аналіз мережевих пакетів, виявлення атак
Операторська станція(SCADA)	ScadaBR	Візуалізація технологічного процесу, відправка команд керування, роль "жертви" на рівні оператора

Узагальнена структура віртуального стенду дозволяє моделювати взаємодію між SCADA-системою, PLC-контролером та атакуючими вузлами в ізольованому мережевому середовищі[19,23].

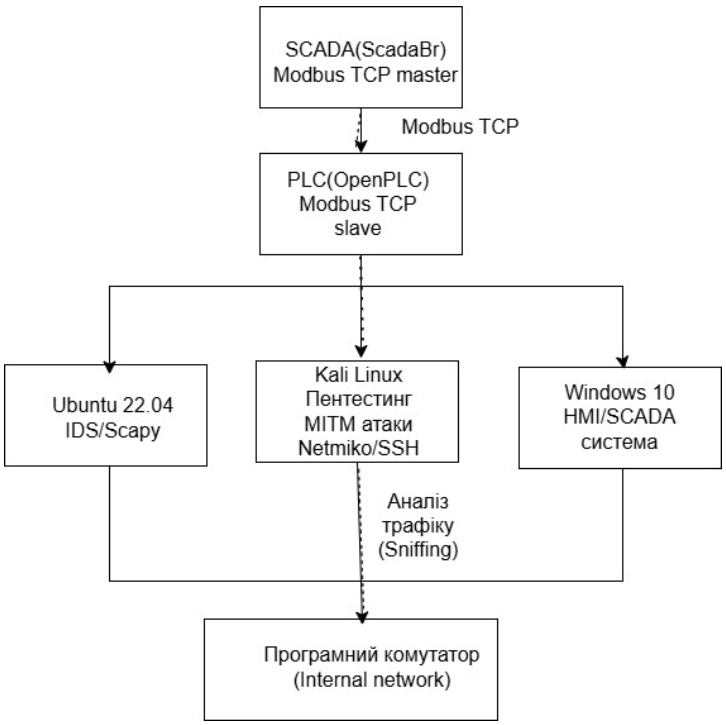


Рисунок 2.1 – Архітектура віртуального стенду промислової системи

Архітектура віртуальної мережі, структурна схема якої представлена на рис. 2.1, мережі побудована на базі середовища VirtualBox та включає кілька сегментів, що імітують реальну промислову інфраструктуру. На верхньому рівні розташована SCADA-система (ScadaBR), яка виконує функції моніторингу та диспетчеризації. Взаємодія з контролером здійснюється через протокол Modbus TCP[19,15].

Центральним елементом є PLC (OpenPLC), що реалізує логіку технологічного процесу за допомогою програми на Ladder Diagram[2].

Для моделювання атак використовуються вузли Kali Linux, які виконують функції генерації аномального трафіку, реалізації MITM-атак та ARP Spoofing. Ubuntu 22.04 виступає як вузол IDS, де реалізовано інструменти Scapy та Netmiko для аналізу трафіку та виявлення аномалій[21,22].

Windows 10 використовується як клієнтська станція HMI для візуалізації технологічного процесу. Усі компоненти взаємодіють у межах сегментованої віртуальної мережі, що забезпечує ізоляцію та контроль доступу між різними рівнями системи[3,17].

Проєктування архітектури віртуального полігону є фундаментальним етапом, оскільки від коректності налаштування мережевого середовища залежить

не лише успішність проведення атак, а й достовірність отриманих результатів моніторингу.

Топологія досліджуваної мережі побудована за схемою «Зірка». Це дозволяє централізувати обмін даними та імітувати підключення всіх вузлів до одного комутатора технологічного рівня. Графічне представлення розробленої архітектури наведено на рис. 2.2[23].

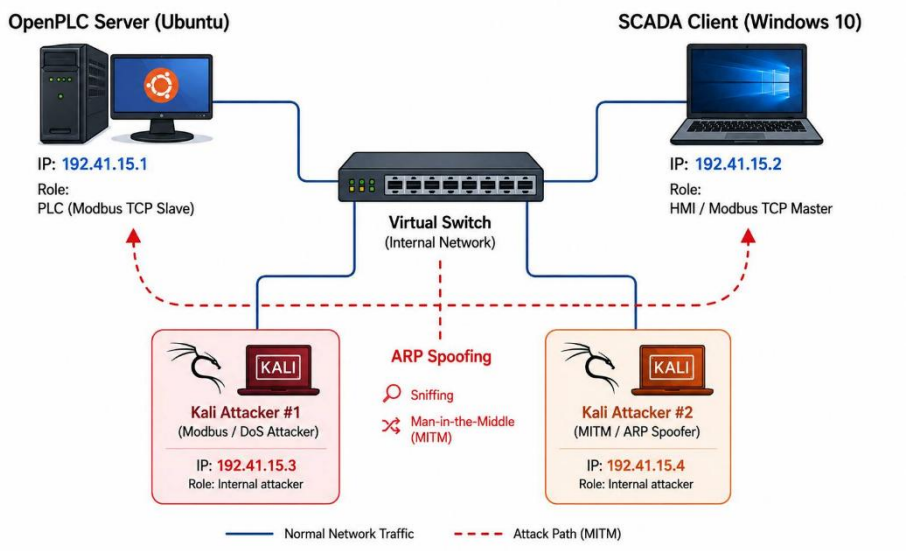


Рисунок 2.2 – Топологія досліджувальної мережі

Відповідно до принципів сегментації промислових мереж, описаних у моделі Purdue та стандарті IEC 62443, мережа була сегментована на логічні зони. Попри те, що всі вузли функціонували в межах одного широкомовного домену Internal Network, логічно мережа була сегментована відповідно до функціонального призначення вузлів[19,23]:

1. Зона виконання (Process Zone).

Представлена вузлом OpenPLC (IP: 192.41.15.1). Тут генерується основний технологічний трафік.

2. Зона управління (Basic Control).

Представлена системою ScadaBR (IP: 192.41.15.2), яка виконує роль Modbus-клієнта.

3. Зона безпеки та аудиту.

Представлена двома вузлами Kali Linux. Перший вузол (IP: 192.41.15.3) використовувався для генерації аномального Modbus TCP-трафіку та DoS/flood-активності. Другий вузол (IP: 192.41.15.4) виконував функції аналізу трафіку, перехоплення пакетів та реалізації MITM-атаки шляхом ARP Spoofing[16,22].

2.2 Проєктування протоколу промислової системи на базі Modbus подібного протоколу

У розробленому віртуальному стенді взаємодія між компонентами промислової системи реалізована на базі протоколу Modbus TCP. OpenPLC виконує роль Modbus Server (Slave), тоді як ScadaBR функціонує як Modbus Client (Master)[14,15].

Взаємодія між SCADA-системою та PLC-контролером у межах віртуального стенду реалізована за допомогою протоколу Modbus TCP. Схему обміну даними наведено на рис. 2.3.

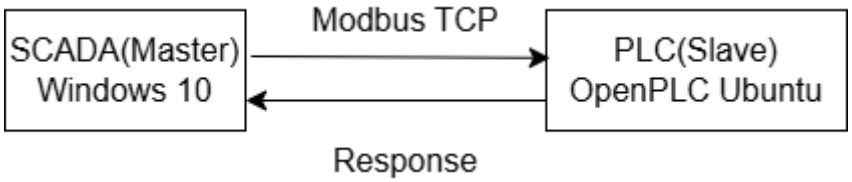


Рисунок 2.3 – Схема обміну даними між SCADA-системою та PLC за протоколом Modbus TCP

SCADA-система функціонує в ролі Modbus Master та періодично надсилає запити до PLC-контролера, який виконує роль Modbus Slave та повертає відповідні дані.

Для передачі даних між SCADA-системою та PLC-контролером у протоколі Modbus TCP використовуються function codes (коди функцій), які визначають тип операції, що виконується сервером. Кожен function code відповідає за окрему дію, наприклад читання стану дискретних виходів, отримання значень регістрів або запис параметрів технологічного процесу[14,15].

У межах розробленого віртуального стенду використовувалися стандартні function codes протоколу Modbus TCP. Основні функції наведено в табл. 2.2.

Таблиця 2.2

Основні function codes протоколу Modbus TCP

Код функції	Назва функції	Призначення
01	Read coils	Зчитування стану дискретних виходів
05	Write Single Coil	Запис значення одного виходу
06	Write Single Register	Запис значення одного регістра
15	Write Multiple Coils	Запис декількох дискретних виходів
16	Write Multiple Registers	Запис декількох регістрів

Аналіз function codes є важливим елементом систем виявлення вторгнень для промислових мереж. Нетипове використання кодів функцій, аномальне використання function codes 15 та 16 може свідчити про спроби несанкціонованої зміни параметрів технологічного процесу[31,20,29].

У межах розробленого стенду використовувались основні function codes протоколу Modbus TCP, необхідні для обміну даними між SCADA-системою та PLC-контролером. Для зчитування стану дискретних виходів використовувався function code 01 (Read Coils), для запису одиночних значень – function codes 05 (Write Single Coil) та 06 (Write Single Register), а для групового запису параметрів – function codes 15 (Write Multiple Coils) і 16 (Write Multiple Registers). Аналіз аномальної активності базувався на контролі нетипових або підозрілих запитів до зазначених функціональних кодів.

Для емуляції роботи комутаційного апарату (високовольтного вимикача підстанції) було розроблено програму керування мовою релейно-контактних схем (LD – Ladder Diagram). Ця мова стандартизована міжнародним стандартом IEC 61131-3 і є найбільш поширеною в енергетиці завдяки своїй наочності[2,18].

Алгоритм функціонування системи передбачає наступну логіку:

1. Увімкнення.

При надходженні сигналу логічної одиниці на вхід Start_PB (натискання кнопки), активується вихідна котушка Breaker_Switch (замикання контактів вимикача).

2. Самопідхват (Latching).

Оскільки кнопка запуску не має фіксації, для утримання вимикача у ввімкненому стані використовується паралельний контакт, який шунтує кнопку пуску станом виходу. Це дозволяє системі залишатися активною після відпускання кнопки.

3. Вимкнення.

При надходженні сигналу на нормально-замкнений вхід Stop_PB, коло розривається, і вихідна котушка деактивується (вимкнення живлення).

Логічну послідовність прийняття рішень контролером під час виконання описаного циклу сканування відображено на блок-схемі алгоритму (рис. 2.4).

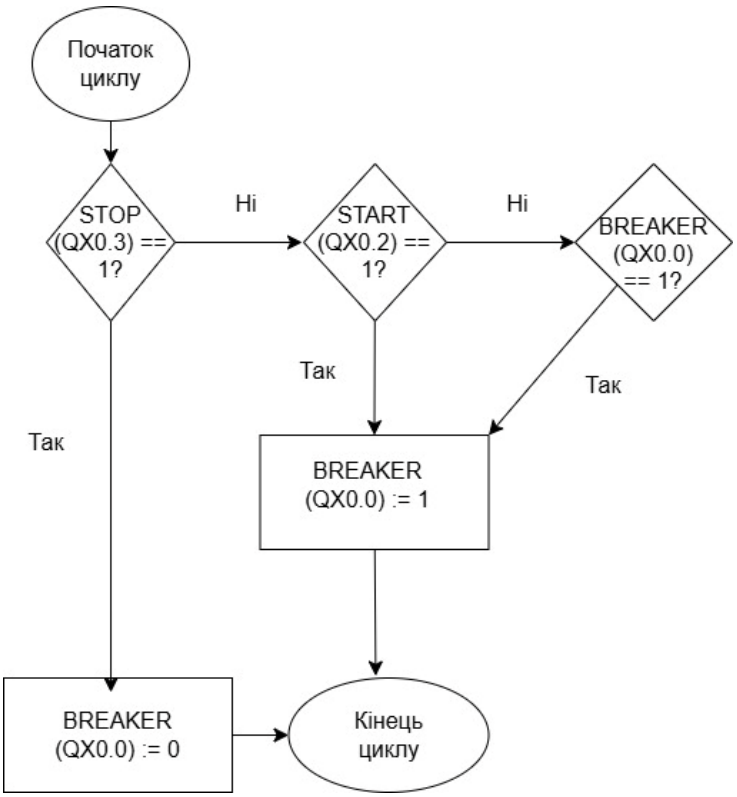


Рисунок 2.4 – Блок-схема алгоритму роботи логіки керування вимикачем

Як показано на рис. 2.4, алгоритм реалізує принцип пріоритетного переривання. Система в першу чергу перевіряє стан регістра аварійної зупинки %QX0.3. Якщо команда на вимкнення відсутня, контролер аналізує запит на активацію %QX0.2. Третій етап перевірки (стан %QX0.0) виконує роль програмного самопідхвату, що забезпечує стабільність роботи системи при імпульсному керуванні.

2.3.1. Механізм обробки логіки в OpenPLC.

Для забезпечення детермінованості роботи системи захисту OpenPLC працює в циклічному режимі (Scan Cycle). Це гарантує, що стан високовольтного вимикача перевіряється та оновлюється через рівні проміжки часу, незалежно від навантаження на мережу[2,18].

Цикл сканування складається з трьох основних етапів:

1. Зчитування образів входів.

ПЛК фіксує стан регістрів Modbus, які відповідають командам Start та Stop.

2. Обробка програми (Scan).

Процесор виконує логіку Ladder Diagram (LD). На цьому етапі реалізується функція «самопідхвату», яка дозволяє утримувати вимикач у ввімкненому стані навіть після припинення сигналу на вході Start.

3. Оновлення виходів.

Результат логічної операції записується в котушку %QX0.0(Breaker_Switch), яка імітує замикання силових контактів.

2.3.2. Обґрунтування вибору типів даних та системи адресації.

При розробці програми керування особлива увага приділялася відповідності внутрішньої адресації ПЛК та протоколу Modbus TCP. Вибір адрес типу %QX та типу даних BOOL зумовлений архітектурними особливостями побудови пам'яті контролера.

1. Синтаксис адресації за стандартом IEC 61131-3.

Використання префікса %Q позначає, що змінна належить до області вихідних даних (Outputs). Другий символ X вказує на розмір даних – Bit (одиничний біт)[2,15].

1.1 У системі OpenPLC змінні, які призначені для взаємодії зі SCADA-системою через Modbus TCP, найчастіше оголошуються в області %Q. Це пов'язано з тим, що змінні типу %QX автоматично мапуються (відображаються) на Modbus Coils (регістри типу 0x), які підтримують як читання, так і запис.

1.2 Адреса %QX0.1 означає: область виходів (Q), розмір біт (X), нульовий байт (0), перший біт (.1).

2. Обґрунтування типу даних BOOL.

Для керування комутаційним апаратом (вимикачем) було обрано логічний тип даних BOOL (Boolean). Це рішення базується на дискретній природі об’єкта керування[18]:

2.1 Бінарний стан.

Високовольтний вимикач має два сталих стани: «Увімкнено» (логічна 1) та «Вимкнено» (логічний 0). Використання складніших типів даних (наприклад, INT або REAL) було б надлишковим і призвело б до нераціонального використання пам'яті ПЛК та збільшення обсягу мережевого трафіку.

2.2 Оптимізація передачі даних.

Один біт даних BOOL у протоколі Modbus TCP передається мінімально можливим обсягом інформації всередині MBAP-заголовка. Це критично важливо для систем РЗА (релейного захисту та автоматики), де швидкість реакції системи на команду має бути максимальною.

3. Взаємодія з Modbus Register Map.

Вибір адрес %QX0.0, %QX0.2 та %QX0.3 дозволяє створити послідовну карту регістрів (Modbus Register Map). Це спрощує групове зчитування даних (функція Modbus 01 – Read Coils). Замість того, щоб опитувати кожен регістр окремим запитом, SCADA-система може одним запитом зчитати стан усіх трьох змінних, що значно знижує навантаження на мережевий канал зв'язку. Взаємозв'язок між описаними типами даних, внутрішніми адресами ПЛК та відповідними реєстрами Modbus TCP систематизовано та наведено в табл. 2.2[15].

Таблиця 2.2

Таблиця відповідності змінних програми та регістрів Modbus

Ім’я змінної	Тип даних	Адреса OpenPLC	Адреса Modbus	Призначення
Start_PB	BOOL	%QX0.0	Coil 2	Кнопка «Пуск»
Stop_PB	BOOL	%QX0.1	Coil 3	Кнопка «Стоп»
Breaker_Switch	BOOL	%QX0.0	Coil 0	Стан вимикача

2.3 Проєкт підсистеми захисту промислової системи на базі Modbus подібного протоколу

Для підвищення рівня безпеки промислової системи була спроєктована підсистема виявлення аномальної мережевої активності. Основним призначенням

підсистеми є моніторинг мережевого трафіку Modbus TCP у режимі реального часу, виявлення підозрілих мережевих подій та фіксація потенційних інцидентів безпеки[26,31].

Необхідність створення такої підсистеми обумовлена особливостями протоколу Modbus TCP, який не має вбудованих механізмів автентифікації, шифрування та контролю цілісності повідомлень. Це створює можливість реалізації атак типу ARP Spoofing, DoS/DDoS, MITM, а також несанкціонованої зміни параметрів технологічного процесу шляхом модифікації Modbus-пакетів[5,20].

Розроблена підсистема захисту базується на аналізі мережевого трафіку промислової системи та контролі аномальної активності у мережі. Основним джерелом даних для аналізу є мережеві пакети, що передаються між SCADA-системою та PLC-контролером у межах протоколу Modbus TCP[29].

У процесі роботи підсистеми здійснювався контроль мережевих параметрів, які можуть свідчити про реалізацію мережевих атак або порушення нормального режиму функціонування системи. Основні параметри, що аналізувалися підсистемою захисту, наведено в табл. 2.3.

Таблиця 2.3

Параметри, що аналізуються

Параметр	Призначення
IP	Ідентифікація вузлів
MAC	Виявлення MITM
Function codes	Аналіз Modbus
Packet rate	Flood detection

Для виявлення flooding-атак використовувався механізм контролю інтенсивності мережевого трафіку. Підсистема здійснювала підрахунок кількості пакетів, що передавалися протягом визначеного проміжку часу. У випадку перевищення встановленого порогового значення трафік класифікувався як потенційно аномальний, після чого формувалося повідомлення про можливу атаку типу DoS/DDoS[8,31].

Окрему увагу було приділено виявленню атак типу ARP Spoofing, які використовуються для реалізації сценаріїв MITM (Man-in-the-Middle). Для цього

здійснювався контроль відповідності між MAC- та IP-адресами вузлів мережі. У разі виявлення невідповідності або появи дубльованих ARP-відповідей підсистема формувала попередження про можливу спробу перехоплення мережевого трафіку[31,22].

Крім аналізу мережевих параметрів, підсистема здійснювала контроль Modbus function codes. Нетипове використання кодів функцій, велика кількість запитів на запис регістрів або спроби використання нехарактерних команд могли свідчити про реалізацію несанкціонованого впливу на PLC-контролер[29,31].

Усі виявлені інциденти безпеки фіксувалися у журналі подій та додатково зберігалися у форматі pcap для подальшого аналізу мережевого трафіку. Це дозволяло проводити детальний аналіз атак, досліджувати поведінку мережевих пакетів та оцінювати ефективність роботи підсистеми захисту[22].

Загальну структуру роботи підсистеми захисту наведено на рис. 2.5.

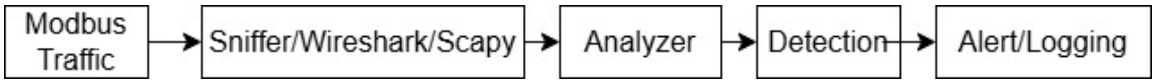


Рисунок 2.5 – Структура роботи підсистеми виявлення аномалій мережевого трафіку

Основними перевагами розробленої підсистеми є:

1. Можливість моніторингу мережевого трафіку у режимі реального часу.

2. Виявлення аномальної активності у промисловій мережі.

3. Фіксація мережевих інцидентів та підозрілих подій.

4. Можливість подальшого аналізу мережевого трафіку за допомогою pcap-файлів.

5. Простота інтеграції у віртуальне середовище промислової системи.

Разом із перевагами розроблена підсистема має певні обмеження. Зокрема, використання сигнатурного та порогового аналізу може призводити до виникнення помилкових спрацювань (False Positive) або пропуску окремих атак (False Negative). Крім того, ефективність виявлення аномалій залежить від правильності налаштування порогових значень мережевого трафіку та особливостей функціонування промислової системи[26].

Перехоплення мережевого трафіку здійснюється у режимі promiscuous mode, що дозволяє аналізувати всі пакети, які передаються в межах віртуальної мережі. Такий підхід забезпечує можливість контролю мережевої активності між PLC-контролером, SCADA-системою та вузлами атакуючої сторони.

Для аналізу пакетів Modbus TCP використовуються елементи Deep Packet Inspection (DPI), які дозволяють здійснювати перевірку вмісту мережевих пакетів на прикладному рівні. Це дає можливість аналізувати function codes, параметри запитів та аномальну активність у мережевому трафіку промислової системи[20,31].

Схему реалізації атаки типу MITM у межах досліджуваної мережі наведено на рис. 2.6.



Рисунок 2.6 – Схема реалізації MITM-атаки у промисловій мережі

2.4 Проєкт програмного модуля для виявлення аномалій безпеки та моделювання атак у трафіку Modbus-подібного протоколу

Pymodbus – це Python-бібліотека для реалізації взаємодії з пристроями, що використовують протокол Modbus TCP. Бібліотека підтримує формування запитів, читання та запис регістрів, а також дозволяє реалізувати обмін даними між вузлами промислової мережі[15].

У розробленій системі бібліотека використовувалась для моделювання мережевої взаємодії з PLC-контролером, генерації тестового трафіку та реалізації сценаріїв атак, пов’язаних зі зміною значень регістрів Modbus TCP. Це дозволило імітувати несанкціоновані зміни параметрів промислового процесу та виконати тестування IDS-модуля в умовах наближених до реального середовища.

Крім того, використання Pymodbus дозволило реалізувати передачу Modbus-запитів у автоматизованому режимі та забезпечити гнучке налаштування параметрів взаємодії між вузлами віртуального стенду.

Netmiko – це Python-бібліотека для автоматизації мережевої взаємодії через SSH-з’єднання. Бібліотека базується на Paramiko та забезпечує спрощене керування

мережевими пристроями шляхом автоматизації виконання команд і обробки відповідей від віддалених вузлів[21].

У програмній реалізації бібліотека Netmiko використовувалась для віддаленого керування PLC-контролером через SSH-з'єднання. За допомогою бібліотеки виконувалась передача команд до вузла OpenPLC, запуск окремих процесів та автоматизація керування компонентами віртуального стенду.

Крім того, використання Netmiko дозволило реалізувати альтернативний спосіб керування PLC-контролером у випадку виникнення проблем із SCADA-системою або необхідності прямого доступу до вузла.

Scapy – це Python-бібліотека для роботи з мережевими пакетами, яка забезпечує можливість їх формування, перехоплення, аналізу та модифікації. Бібліотека підтримує роботу з різними мережевими протоколами та використовується для аналізу трафіку, тестування мереж і дослідження мережевих атак[22].

Scapy дозволяє здійснювати надсилання мережевих пакетів і отримання відповідей від вузлів мережі, що забезпечує можливість аналізу структури пакетів та дослідження мережевої взаємодії. Крім того, бібліотека підтримує перехоплення трафіку та роботу з pcap-файлами. Основні можливості бібліотеки Scapy наведені на рис.2.7.

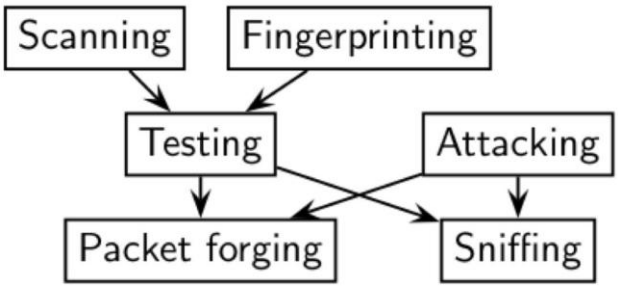


Рисунок 2.7 – Основні можливості бібліотеки Scapy

На основі бібліотеки Scapy у програмній реалізації було створено функції аналізу мережевого трафіку, реалізовано snifing пакетів, аналіз pcap-файлів та модуль виявлення аномалій у трафіку Modbus TCP. Також бібліотека використовувалась для виявлення ARP spoofing-атак, контролю інтенсивності

мережевого трафіку та запису перехоплених пакетів у pcap-файл для подальшого аналізу у Wireshark[22,31].

Для реалізації часового аналізу мережевого трафіку та визначення інтенсивності пакетів використовувалась стандартна бібліотека Python time. Бібліотека застосовувалась для формування часових інтервалів аналізу трафіку, підрахунку кількості пакетів за визначений проміжок часу та виявлення аномальної активності, характерної для DoS-атак.

Бібліотека subprocess використовувалась для запуску системних команд та автоматизації окремих процесів моделювання атак у віртуальному середовищі. За допомогою бібліотеки виконувалось керування окремими процесами операційної системи, запуск допоміжних утиліт та автоматизація сценаріїв тестування IDS-модуля.

Бібліотека collections використовувалась для зберігання та підрахунку статистичних даних мережевого трафіку. Зокрема, за допомогою структури Counter реалізовувався підрахунок кількості пакетів від окремих IP-адрес з метою виявлення аномальної мережевої активності.

Використання Python-бібліотек у програмній реалізації наведено в таблиці 2.4.

Таблиця 2.4

Використання Python-бібліотек у програмній реалізації модуля виявлення аномалій

Бібліотека	Призначення	Використання у програмній реалізації
Scapy	Аналіз і формування мережевих пакетів	Захоплення трафіку, аналіз ARP/Modbus-пакетів, виявлення підозрілих змін, запис pcap
Pymodbus	Робота з протоколом Modbus	Опис логіки Modbus-взаємодії, підтримка читання/запису регістрів, тестування Modbus-комунікації
Netmiko	SSH-автоматизація	Дистанційне керування вузлом/PLC, виконання команд, запуск сценаріїв навантаження

Продовження табл.2.4

Бібліотека	Призначення	Використання у програмній реалізації
time	Робота з часовими інтервалами	Формування часових вікон для підрахунку пакетів і виявлення DoS
subprocess	Запуск системних команд	Автоматизація запуску окремих команд або процесів під час моделювання атак
collections	Зручне зберігання й підрахунок даних	Підрахунок кількості пакетів за IP/MAC, формування статистики трафіку

Структура програмної реалізації IDS-модуля наведена на рисунку 2.8[26,31].

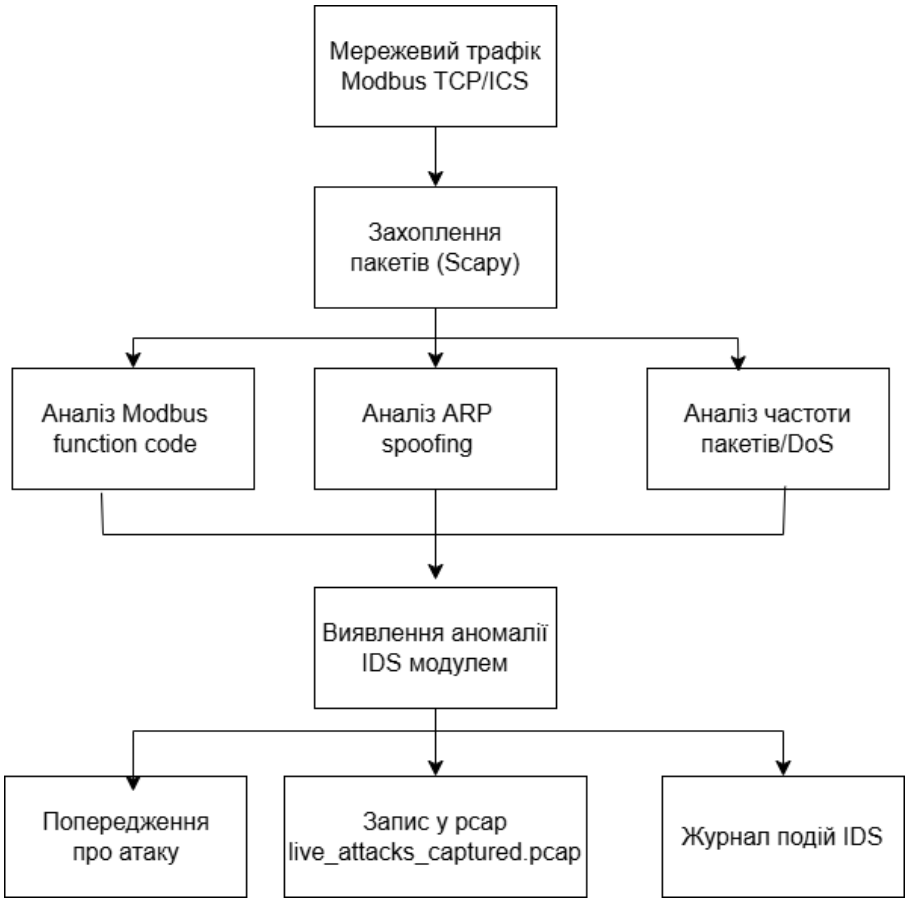


Рисунок 2.8 – Логіка роботи IDS-модуля

Структуру модуля моделювання атак та взаємодію з IDS-модулем наведено на рисунку 2.9[31,16].

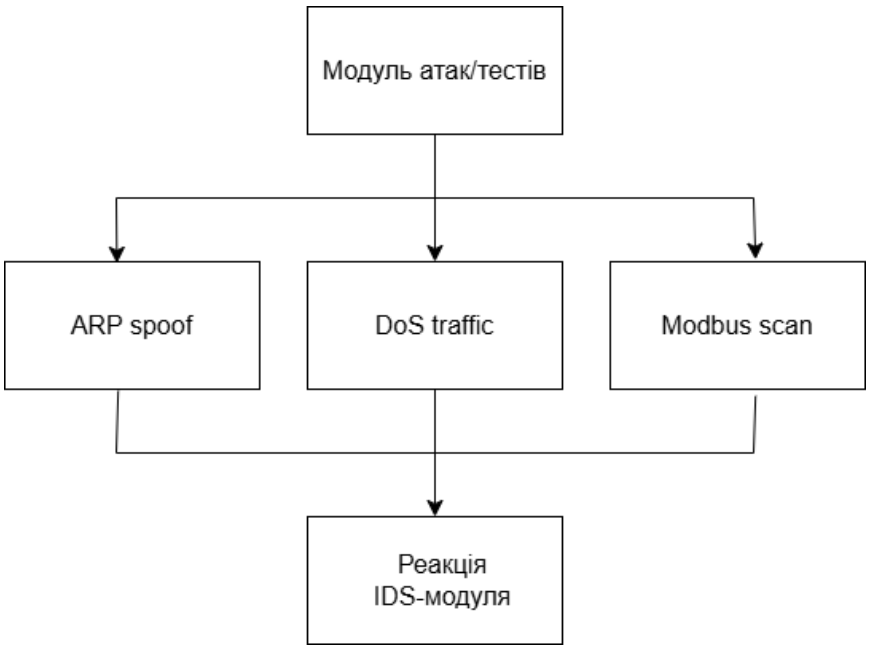


Рисунок 2.9 – Структура модуля моделювання атак

РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ВІРТУАЛЬНОГО СТЕНДУ ТА ПРОГРАМНОГО МОДУЛЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ БЕЗПЕКИ В ТРАФІКУ MODBUS-ПОДІБНИХ ПРОТОКОЛІВ

3.1 Обґрунтування вибору середовища та практична реалізація віртуального стенду

Для моделювання процесів інформаційного обміну в АСУ ТП та реалізації сценаріїв кібератак було спроєктовано віртуальну мережеву інфраструктуру на базі середовища віртуалізації Oracle VirtualBox. З метою забезпечення цілісності даних та запобігання виходу шкідливого трафіку за межі лабораторії, обрано режим "Internal Network" (Внутрішня мережа).

Технічна перевага цього режиму полягає у наступному:

1. Повна ізоляція.

Трафік інкапсулюється всередині програмного комутатора VirtualBox і не доступний для мережевих інтерфейсів хост-машини.

2. Відсутність DHCP.

Всі адреси призначаються статично, що імітує реальні промислові об'єкти, де динамічна зміна IP-адрес є неприпустимою через ризик втрати зв'язку з ПЛК.

3. Емуляція L2-рівня.

Режим дозволяє вузлам обмінюватися Ethernet-фреймами безпосередньо, що є критично важливим для реалізації атак на протокол ARP.

Вибір адресного простору 192.41.15.0 з маскою підмережі /28 (255.255.255.240) є стратегічним рішенням для створення ізольованого сегмента. Використання маски такої довжини дозволяє обмежити широкомовний домен лише 14-ма корисними адресами (192.41.15.1 – 192.41.15.14), що цілком достатньо для функціонування вузлів стенду та забезпечує високий рівень контролю над трафіком.

Математично це обґрунтовується необхідністю мінімізації "мережевого шуму" та запобігання конфліктам з іншими віртуальними інтерфейсами хост-

системи. Повний перелік налаштувань мережевих адаптерів для кожного компонента віртуального полігону представлено в табл. 3.1.

Таблиця 3.1

Адресація вузлів віртуального стенду

Мережа/Пристрій	Інтерфейс/Мережний адаптер/Шлюз	IP-адреса	Маска	Префікс
Мережа A	-	192.41.15.0	255.255.255.240	/28
Ubuntu 22.04	Мережний адаптер	192.41.15.1	255.255.255.240	/28
Windows 10	Мережний адаптер	192.41.15.2	255.255.255.240	/28
Kali Linux #1	Мережний адаптер	192.41.15.3	255.255.255.240	/28
Kali Linux #2	Мережний адаптер	192.41.15.4	255.255.255.240	/28

Для забезпечення стабільної роботи віртуального полігону та запобігання конфліктам ресурсів хостової операційної системи, було визначено оптимальні конфігурації для кожної віртуальної машини. При виборі характеристик враховувалися системні вимоги розгорнутого ПЗ та роль кожного вузла в моделі техпроцесу.

Узагальнені параметри конфігурації віртуальних машин наведено в таблиці 3.2.

Таблиця 3.2

Технічні характеристики вузлів віртуального полігону

Назва ВМ	ОС	vCPU	RAM	Дисковий простір	Мережний адаптер
Ubuntu-PLC	Ubuntu 22.04 LTS	2	6 GB	35 GB	Internal network
Win10-Scada	Windows 10	1	4,5 GB	70 GB	Internal network
Kali-Linux #1	Kali linux 2021	2	4 GB	80 GB	Internal network
Kali-Linux #2	Kali linux 2021	2	3,5 GB	80 GB	Internal network

Для реалізації віртуального полігону було обрано конфігурації вузлів, що забезпечують високу швидкодію під час імітації мережевих навантажень під час моделювання атак. Розподіл ресурсів було виконано наступним чином:

1. OpenPLC_Server (Ubuntu 22.04).

Виділення 2 віртуальних процесорів та 6 ГБ оперативної пам'яті обумовлено необхідністю швидкої компіляції логіки в OpenPLC Editor та стабільної роботи

середовища виконання Runtime. Такий обсяг пам'яті дозволяє системі без затримок обробляти велику кількість Modbus-запитів одночасно.

2. SCADA_Client (Windows 10).

Для АРМ оператора виділено 4,5 ГБ оперативної пам'яті та 70 ГБ дискового простору. Це забезпечує стабільну роботу графічної оболонки та бази даних ScadaBR, що забезпечує стабільну роботу веб-інтерфейсу та збереження журналів технологічного процесу.

3. Kali Linux #1 (Attacker Node).

Конфігурація з 2 vCPU та 4 ГБ пам'яті є оптимальною для проведення активного сканування та виконання скриптів аналізу та генерації мережевого трафіку на Python для детекції атак. Обсяг диску у 80 ГБ дозволяє накопичувати значні масиви перехопленого трафіку (pcap-файли) для подальшого аналізу.

4. Kali Linux #2 (MITM Node).

Конфігурація другого вузла Kali Linux використовувалась для реалізації ARP Spoofing, перехоплення мережевого трафіку та моделювання MITM-атаки. Виділені ресурси (2 vCPU та 3,5 ГБ RAM) забезпечували стабільну роботу засобів аналізу трафіку та перехоплення пакетів у реальному часі.

Всі вузли функціонують у замкненому мережевому контурі Internal Network, що дозволяє безпечно моделювати деструктивні впливи на протокол Modbus TCP без ризику для зовнішньої мережевої інфраструктури.

Графічна реалізація описаної логіки LD та конфігурація відповідних адрес у середовищі розробки OpenPLC Editor представлена на рис. 3.1.

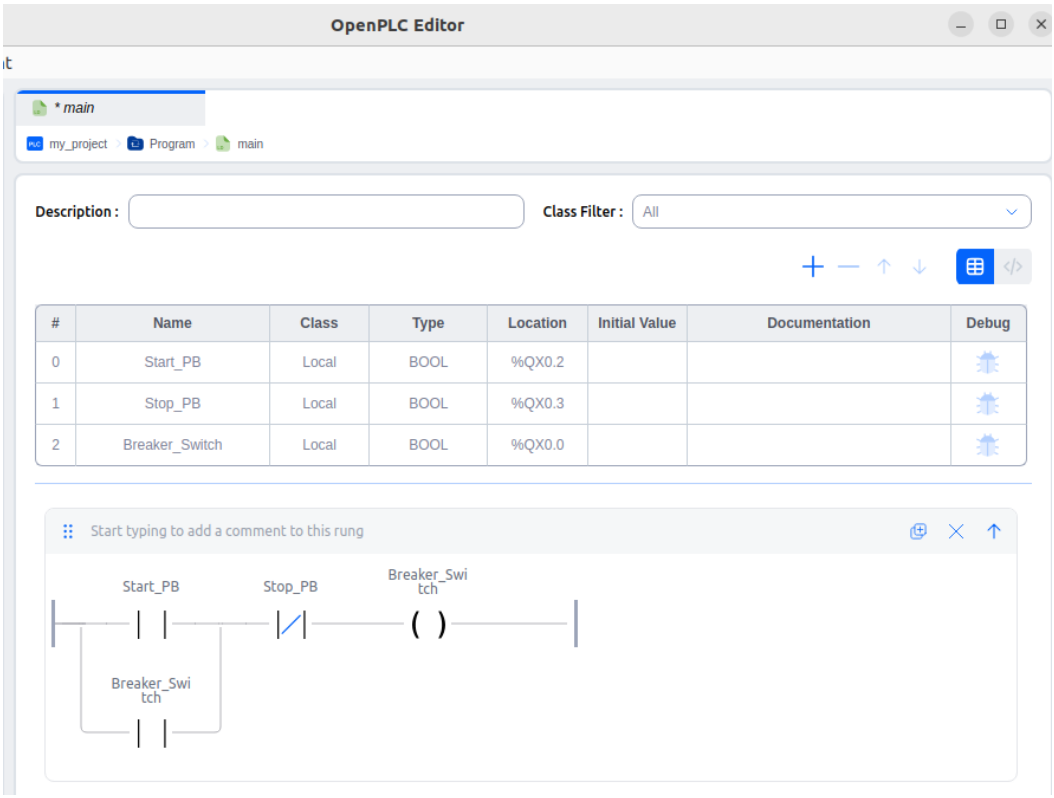


Рисунок 3.1 – Реалізація логіки керування вимикачем у середовищі OpenPLC Editor

3.1.1 Конфігурування мережевого середовища у VirtualBox

Після визначення логічної структури мережі було проведено етап фізичного налаштування віртуальних адаптерів для кожного вузла стенда. Основним завданням на цьому етапі було забезпечення стабільного зв'язку між компонентами за умови повної ізоляції від зовнішніх мереж.

Для кожної віртуальної машини (OpenPLC, ScadaBR, Kali Linux) у налаштуваннях мережевого інтерфейсу було обрано тип підключення «Внутрішня мережа» (Internal Network). В якості ідентифікатора мережі використано назву intnet. Вибір цього режиму обґрунтований необхідністю створення замкненого середовища, де пакети передаються виключно через програмний комутатор віртуалізатора, що унеможливлює вихід шкідливого трафіку на хост-машину.

Особлива увага була приділена налаштуванню вузла Kali Linux, який виконує роль станції моніторингу та тестування на проникнення. На відміну від інших вузлів, для його мережевого адаптера було активовано «Змішаний режим» (Promiscuous Mode) у значенні «Allow All» (Дозволити всім).

Технічна необхідність такого налаштування полягає в наступному:

1. Захоплення трафіку (Sniffing).

У стандартному режимі мережева карта ігнорує всі кадри Ethernet, MAC-адреса отримувача яких не збігається з її власною.

2. Реалізація MITM-атак.

Активація нерозбірливого режиму дозволяє аналізаторам трафіку (наприклад, Wireshark або Ettercap) приймати та обробляти всі пакети, що проходять через віртуальний комутатор, що є критично важливим для перехоплення Modbus-запитів між SCADA-системою та ПЛК.

3. Дзеркалювання трафіку.

Це налаштування є програмним аналогом порту SPAN (Switched Port Analyzer) на фізичних керованих комутаторах.

Приклад налаштування мережевого інтерфейсу для вузла Kali Linux наведено на рис. 3.2.

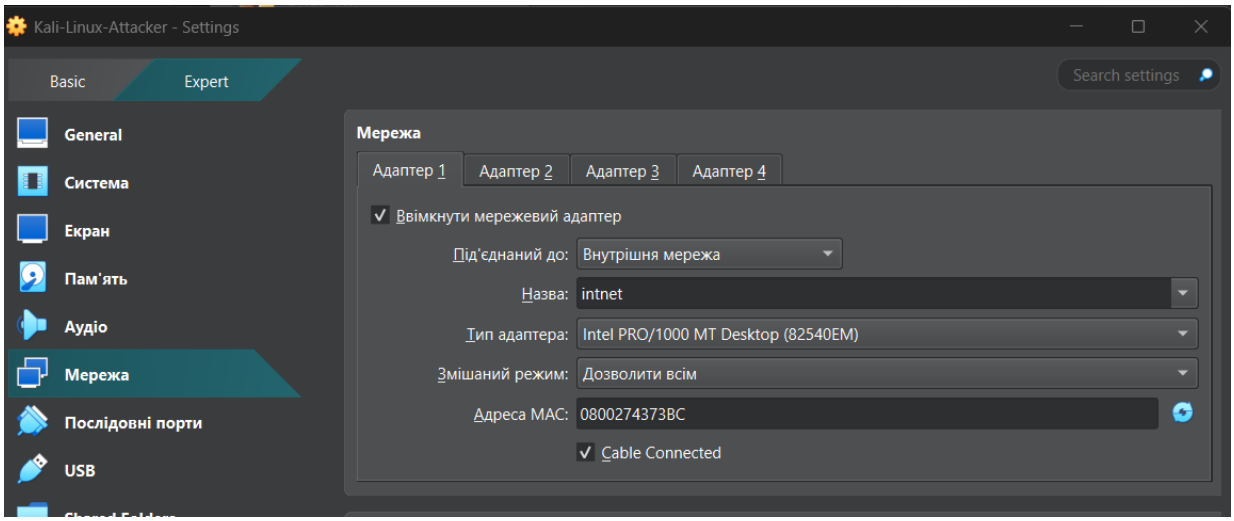


Рисунок 3.2 – Конфігурація мережевого адаптера у VirtualBox для станції моніторингу

Для вузлів OpenPLC та Windows 10 (ScadaBR) налаштування виконані аналогічно, за виключенням змішаного режиму, який для них залишено у стані «Deny» (Заборонено) з метою дотримання стандартної поведінки легітимних вузлів мережі.

3.1.2 Налаштування параметрів протоколу IP та мережевої адресації

Для забезпечення зв’язку між компонентами віртуального полігону було проведено конфігурування мережевих стеків операційних систем. Оскільки всі вузли стенда (OpenPLC, ScadaBR та Kali Linux) знаходяться в межах одного логічного сегмента мережі (192.41.15.0/28), взаємодія між ними відбувається виключно на канальному (Data Link) та мережевому (Network) рівнях моделі OSI без залучення засобів маршрутизації.

Особливістю конфігурації є відсутність шлюзу за замовчуванням (Default Gateway). У промислових мережах (OT-сегментах) такий підхід часто використовується для підвищення безпеки, оскільки він унеможливлює автоматичне пересилання пакетів за межі локального сегмента. Технічне обґрунтування відсутності шлюзу базується на наступних аспектах:

1. Локальна доставка.

При відправці IP-пакета вузол перевіряє, чи знаходиться адреса отримувача в тій самій підмережі (використовуючи маску підмережі). Оскільки всі вузли належать до діапазону 192.41.15.1–14, доставка здійснюється напряму через ARP-запит.

2. Ізоляція трафіку.

Відсутність шлюзу гарантує, що трафік Modbus TCP не буде випадково перенаправлений у зовнішні мережі, навіть якщо в системі з'явиться додатковий мережевий інтерфейс.

Конфігурування проводилося шляхом редагування параметрів мережевих адаптерів у відповідних операційних системах. На рис. 2.3–2.5 продемонстровано результати успішного призначення статичних адрес.

1. На вузлі Ubuntu (OpenPLC) параметри були прописані через утиліту Netplan (або Network Manager), що забезпечує стабільність адреси 192.41.15.1 після перезавантаження системи (рис. 3.3).

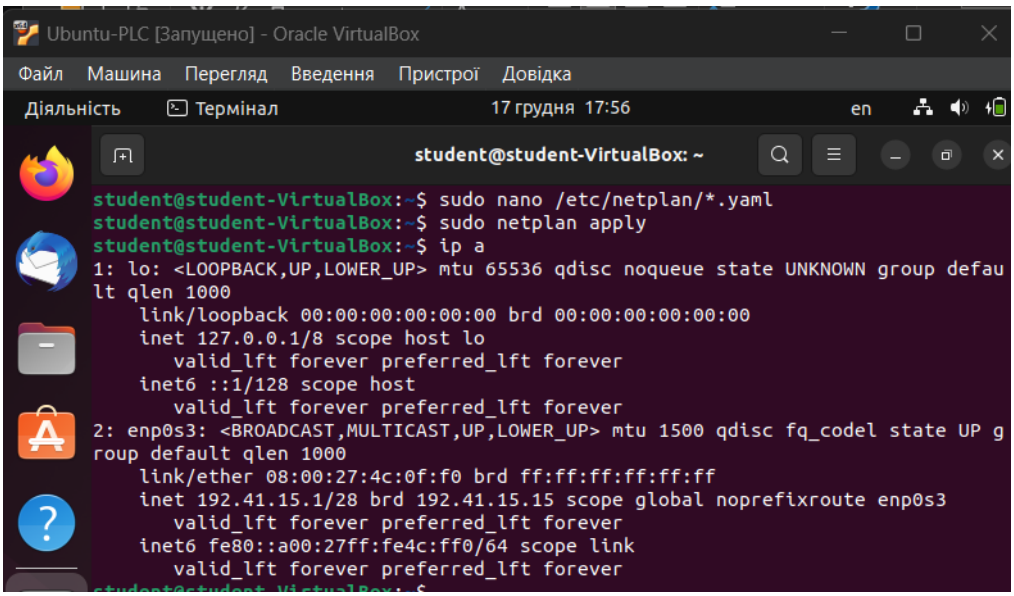


Рисунок 3.3 – Перевірка конфігурації мережевого інтерфейсу вузла OpenPLC

2. На вузлі Windows 10 (ScadaBR) налаштування виконані через властивості протоколу IPv4, де чітко зафіксовано адресу 192.41.15.2 та маску 255.255.255.240 (рис. 3.4).

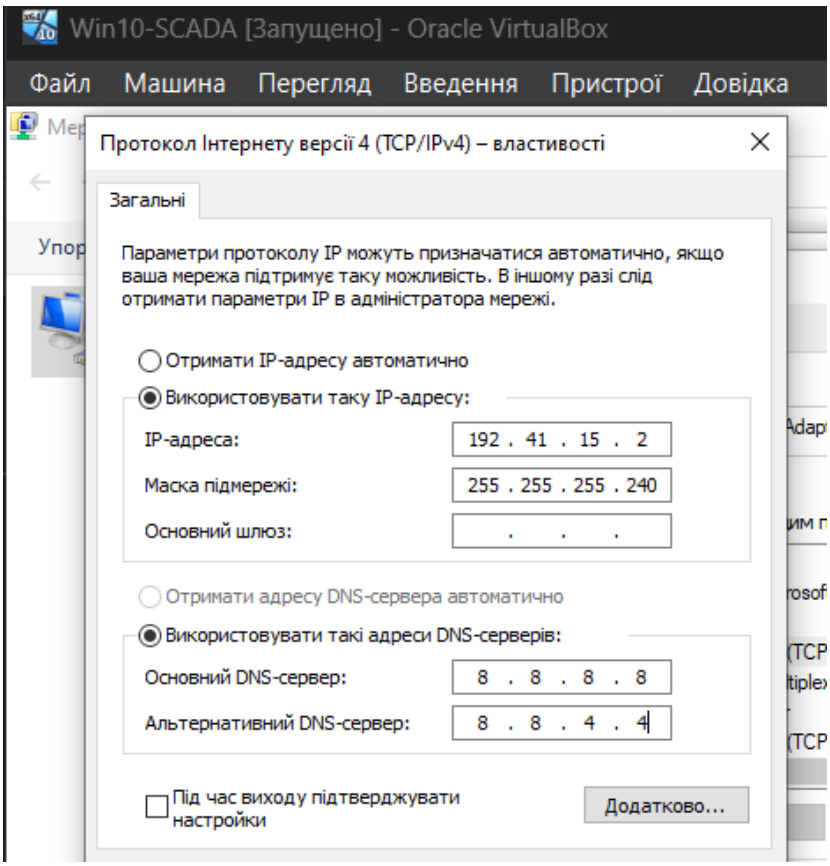


Рисунок 3.4 – Перевірка конфігурації мережевого інтерфейсу вузла OpenPLC

3. На вузлі Windows 10 (ScadaBR) налаштування виконані через властивості протоколу IPv4, де чітко зафіксовано адресу 192.41.15.2 та маску 255.255.255.240 (рис. 3.5).

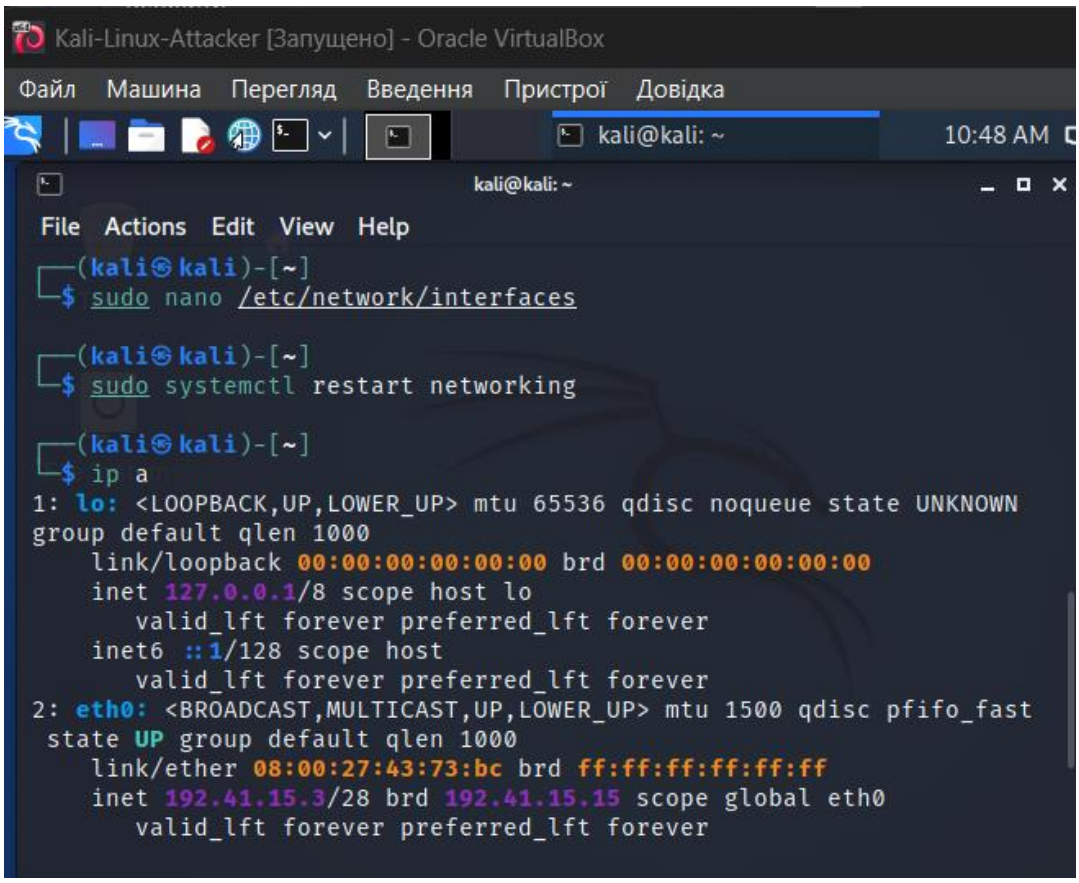


Рисунок 3.5 – Верифікація мережевих параметрів станції моніторингу

Для забезпечення безперешкодного проходження дослідницького трафіку та проведення атак на вищі рівні моделі OSI (прикладний рівень Modbus TCP), необхідно було провести попереднє налаштування мережевих екранів на вузлах-жертвах.

На вузлі Windows 10 (ScadaBR) стандартні налаштування безпеки можуть блокувати вхідні з'єднання на порт 502 або ICMP-запити, що викривило б результати досліджень. Тому в середовищі «Брандмауер Захисника Windows» було проведено деактивацію фільтрації для всіх типів мереж:

1. Доменних мереж.
2. Приватних мереж.
3. Спільних (загальнодоступних) мереж.

Стан вимкненого брандмауера, як показано на рис. 3.6, підтверджується відповідними індикаторами («Вимкнуто»), що свідчить про готовність операційної системи до прийому будь-яких пакетів з віртуального сегмента мережі.

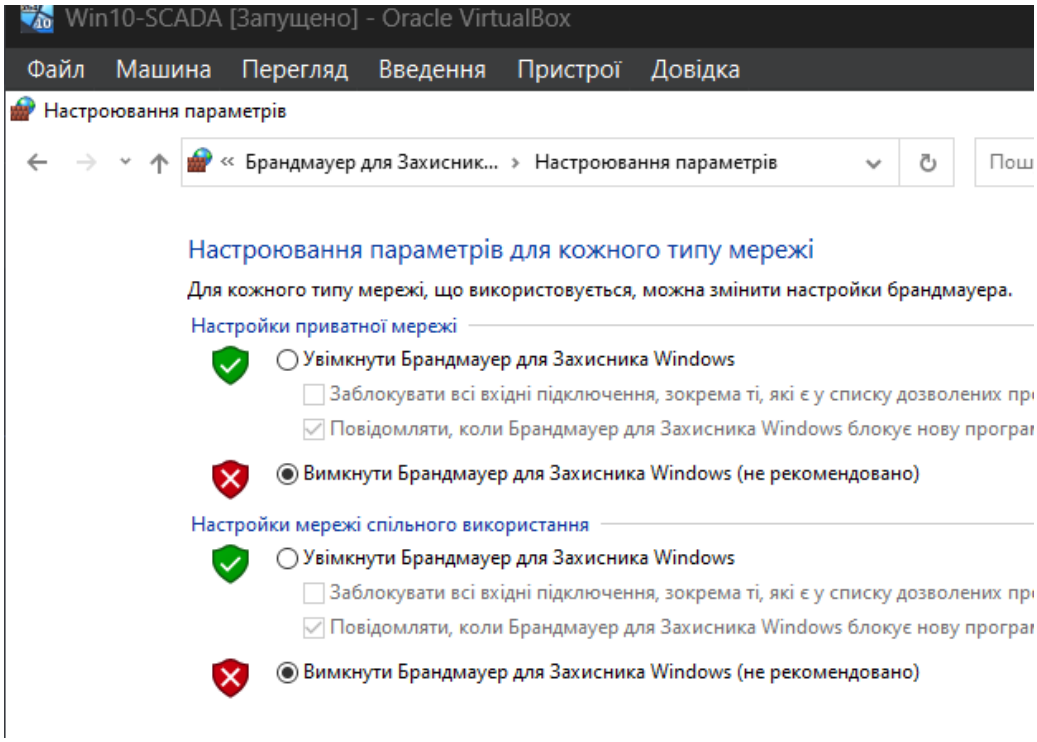


Рисунок 3.6 – Деактивація стандартних засобів фільтрації трафіку в ОС Windows 10

Важливо зауважити, що повне деактивування брандмауера є виключно дослідницьким кроком, обумовленим специфікою бакалаврської роботи. У реальних умовах експлуатації АСУ ТП дотримання політик кібербезпеки вимагає використання принципу мінімальних привілеїв. Це передбачає не відключення захисних механізмів, а створення суворих «дозволяючих правил» (Allow Rules) для конкретних портів. Зокрема, для забезпечення роботи промислового протоколу відкривається лише порт 502/tcp.

Проте, для нашого лабораторного стенда було обрано шлях повної деактивації мережевого екрана. Це зроблено з метою аналізу максимально широкого спектра вразливостей (включаючи ICMP-сканування, виявлення ОС та верифікацію відкритих портів), а також для усунення будь-яких сторонніх чинників, що могли б вплинути на проходження аномального трафіку під час

тестування на проникнення. Такий підхід дозволяє отримати "чисті" результати виявлення атак системою моніторингу без їх превентивного блокування засобами ОС.

Фінальним етапом розгортання мережевої інфраструктури є верифікація цілісності каналів зв’язку. Для цього було використано утиліту ping, яка базується на протоколі ICMP. Перевірка проводилася зі станції моніторингу Kali Linux шляхом надсилання ехо-запитів на адреси ПЛК (192.41.15.1) та SCADA-сервера (192.41.15.2).

Як видно з результатів, наведених на рис. 3.7, час відгуку (RTT) є мінімальним (менше 1 мс), що характерно для внутрішньої віртуальної мережі. Відсутність втрат пакетів (0% packet loss) підтверджує коректність роботи програмного комутатора VirtualBox та готовність стенда до проведення атак на вищі рівні моделі OSI.

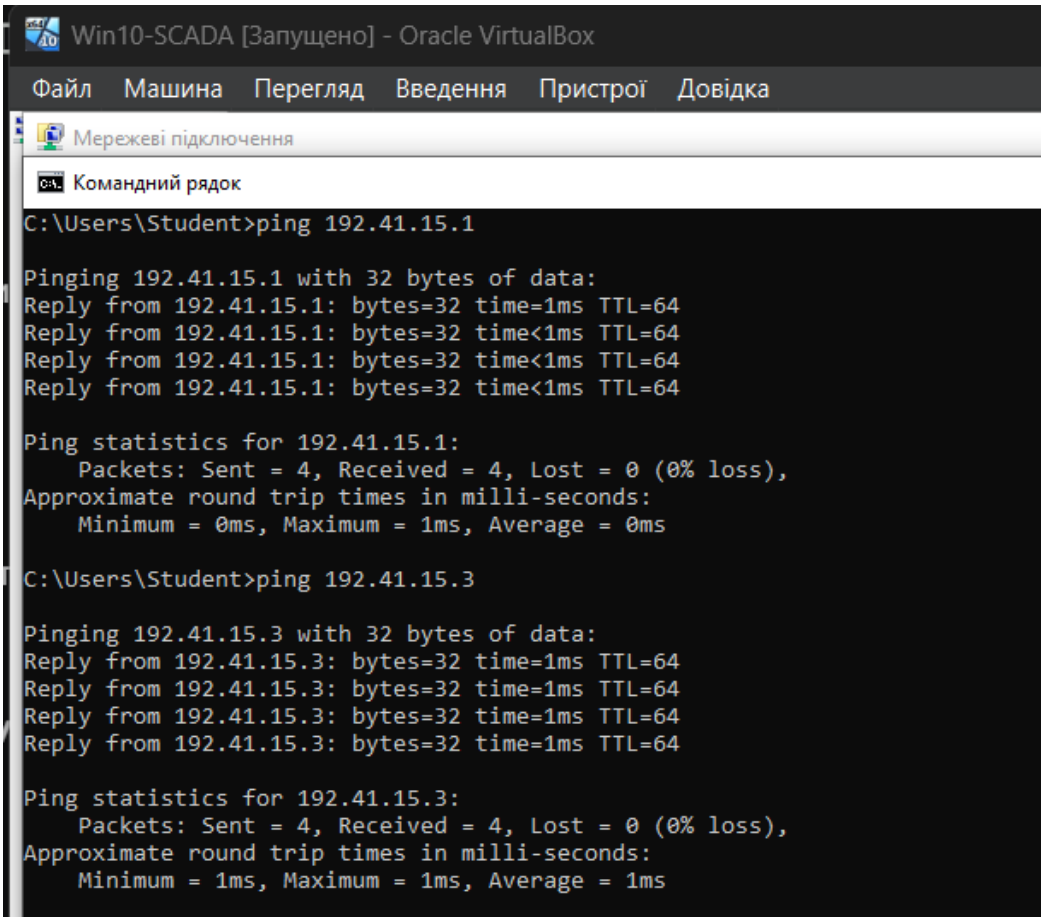


Рисунок 3.7 – Результати тестування мережевої зв’язності між вузлами стенда

3.1.3 Інсталяція та конфігурація OpenPLC Runtime

Оскільки віртуальний стенд функціонує в ізольованому середовищі, для етапу інсталяції програмного забезпечення було тимчасово організовано доступ до мережі Інтернет через NAT-інтерфейс. Після завантаження репозиторіїв та встановлення залежностей (Git, Python, OpenPLC Runtime) віртуальну машину було повернуто в режим "Внутрішня мережа" (Internal Network).

На вузлі з ОС Ubuntu 22.04 було розгорнуто середовище OpenPLC Runtime, яке виступає програмним емулятором промислового контролера. Вибір даної платформи обумовлений її повною відповідністю стандарту IEC 61131-3, що дозволяє перенести результати досліджень на реальне промислове обладнання без суттєвих змін у логіці.

1. Процес інсталяції та компіляції ядра

Процес розгортання програмного ПЛК здійснювався шляхом запуску автоматизованого інсталяційного скрипта в терміналі ОС Ubuntu (рис. 3.8).

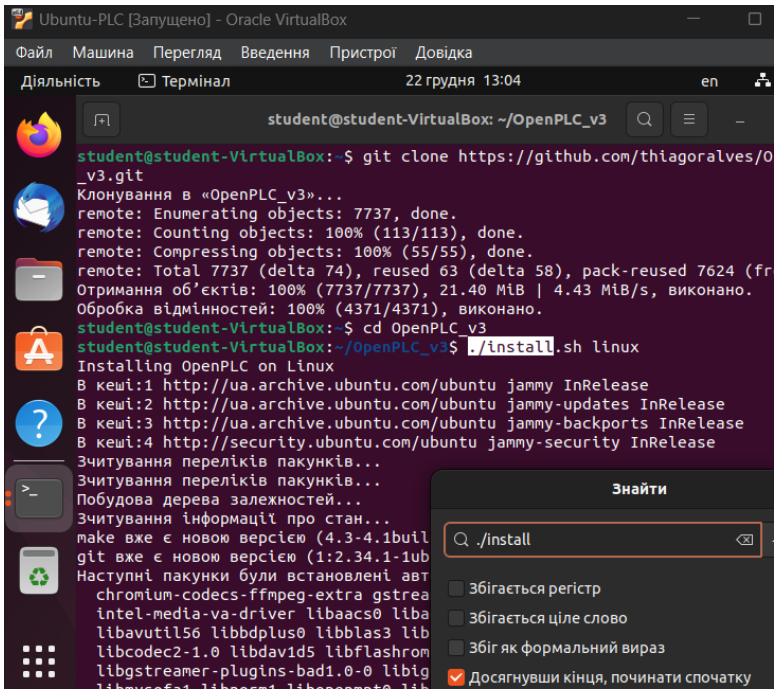


Рисунок 3.8 – Клонування репозиторію та запуск процесу інсталяції OpenPLC

Під час встановлення система виконує такі критичні операції:

1.1 Перевірка наявності необхідних системних залежностей.

1.2 Збірка внутрішніх модулів середовища виконання.

1.3 Налаштування мережевих сервісів для роботи протоколу Modbus TCP.

Використання консольного методу дозволяє контролювати цілісність процесу та впевнитися у відсутності помилок лінкування. Після завершення компіляції запуск сервера виконується командою ./start_openplc.sh. Доступ до панелі керування здійснюється через веб-інтерфейс за адресою http://localhost:8080 (рис. 3.9). Конфігурація Modbus TCP сервера.

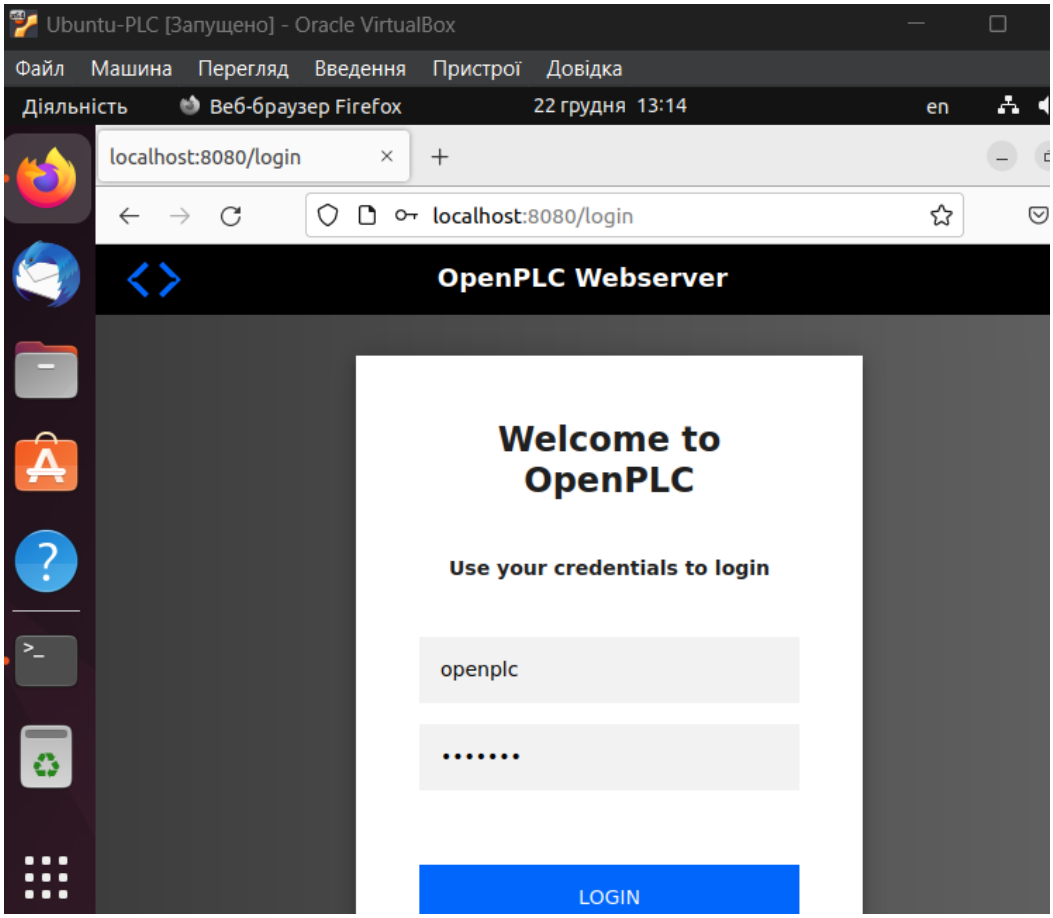


Рисунок 3.9 – Вікно автентифікації у веб-інтерфейсі OpenPLC Runtime

2. Налаштування середовища розробки OpenPLC Editor.

Окрім серверної частини (Runtime), на віртуальній машині було розгорнуто OpenPLC Editor для можливості локального редагування та налагодження коду мовою Ladder Diagram (LD).

При розгортанні редактора виникла необхідність коригування прав доступу до виконуваного файлу, оскільки політика безпеки Linux блокує запуск сторонніх бінарних файлів без явного дозволу. За допомогою терміналу було надано атрибут виконання (команда chmod +x) та здійснено запуск середовища (рис. 3.10)

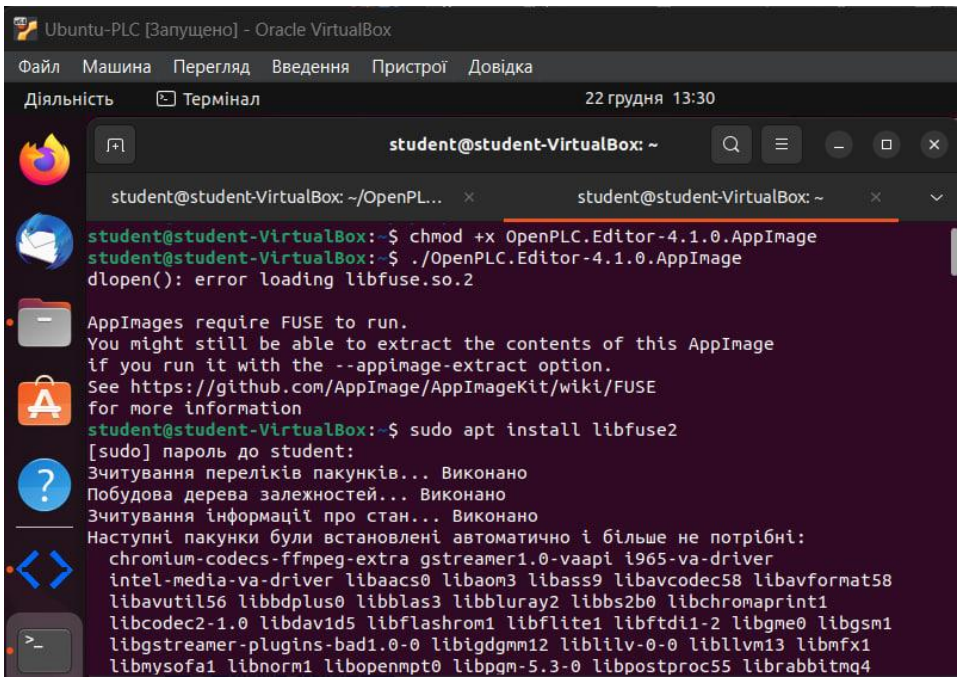


Рисунок 3.10 – Надання прав доступу та запуск OpenPLC Editor

Інтерфейс встановленого редактора (рис. 3.11) дозволяє створювати складні алгоритми керування та проводити їх попередню верифікацію безпосередньо в середовищі Linux.

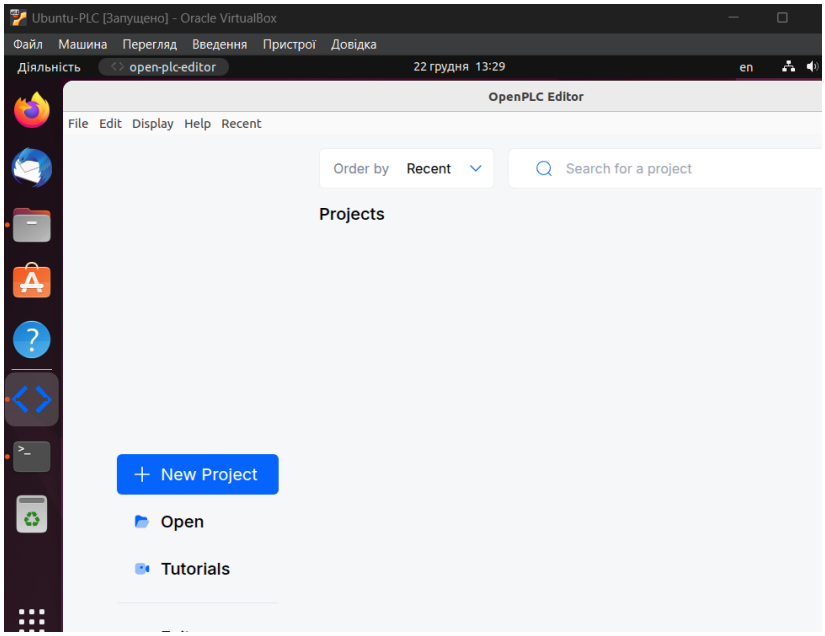


Рисунок 3.11 – Графічний інтерфейс середовища розробки логіки ПЛК

3. Інтеграція та запуск логіки керування.

На завершальному етапі було виконано інтеграцію розробленої логіки в середовище виконання. Файл програми (program.st) було імпортовано через веб-

інтерфейс. Після успішної компіляції та активації режиму «Start PLC», контролер перейшов у робочий стан.

На головній панелі (Dashboard) статус змінився на «Running» (рис. 3.12), що підтверджує готовність віртуального ПЛК до виконання циклу керування та взаємодії зі SCADA-системою по протоколу Modbus TCP.

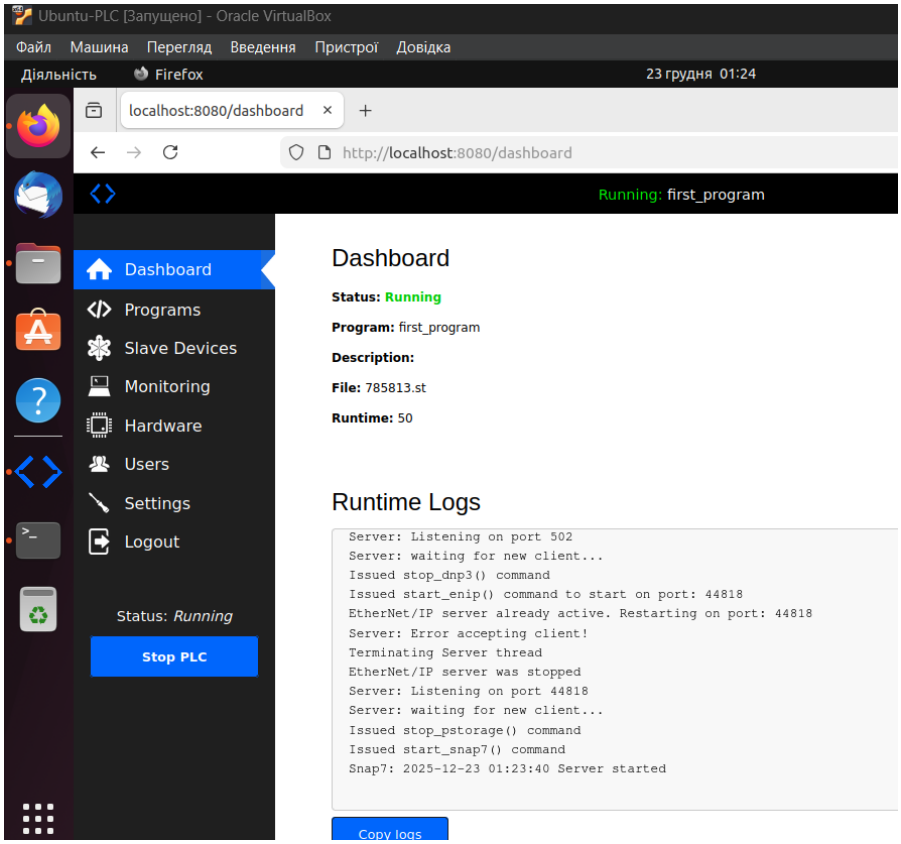


Рисунок 3.12 – Моніторинг активного стану віртуального ПЛК (режим Running)

3.1.4 Розгортання та інтеграція SCADA-системи ScadaBR.

Для забезпечення людино-машинного інтерфейсу (HMI) та реалізації функцій диспетчерського керування на вузлі Windows 10 було розгорнуто систему ScadaBR. Аналогічно до етапу налаштування ПЛК, для завантаження пакетів Java Runtime Environment (JRE) та компонентів SCADA було використано тимчасове NAT-підключення, після чого систему було остаточно ізольовано в межах підмережі 192.41.15.0/28.

Процес розгортання та інтеграції складався з наступних етапів:

1. Керування службами сервера додатків.

Оскільки ScadaBR базується на технології Java, її функціонування забезпечується сервером додатків Apache Tomcat. Для запуску системи було ініціалізовано відповідну службу (рис. 3.13), що дозволило відкрити доступ до веб-консолі управління.

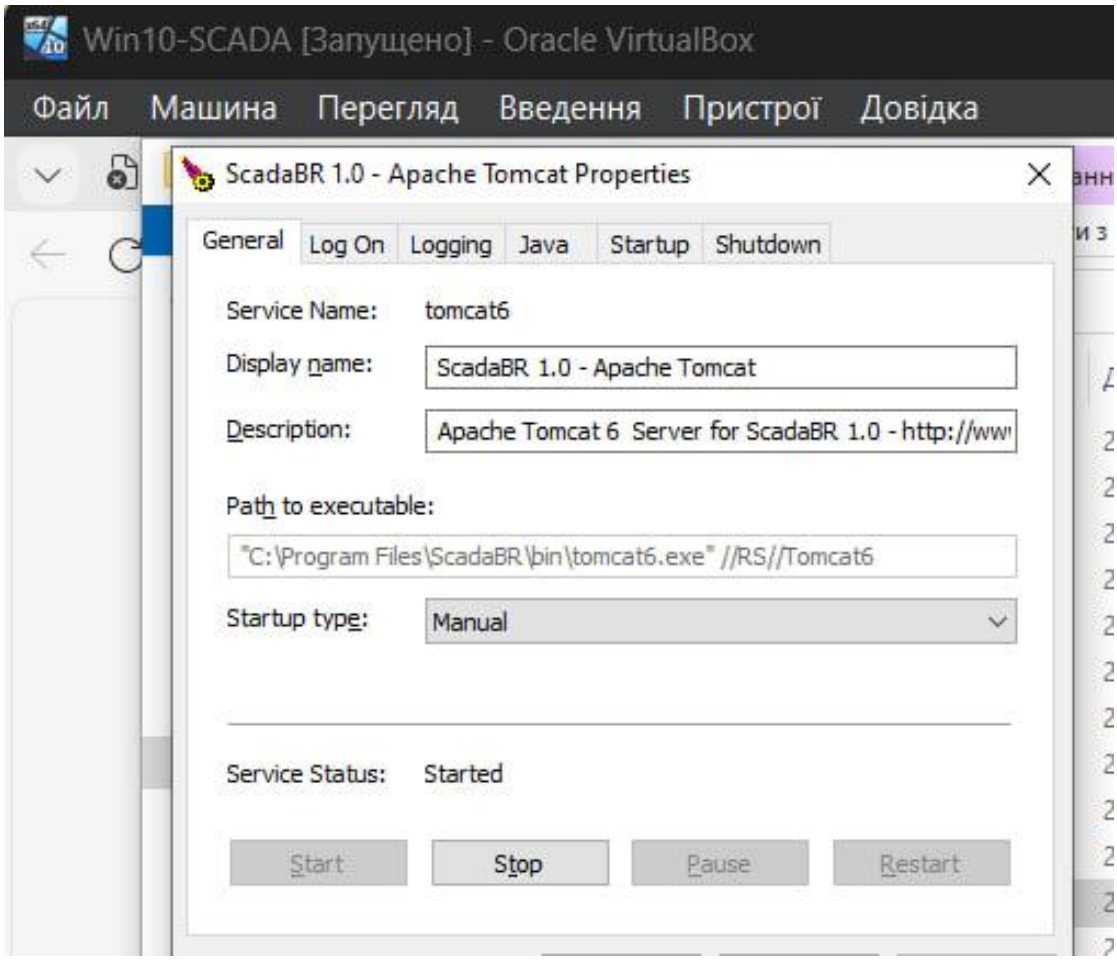


Рисунок 3.13 – Вікно керування службою ScadaBR (Apache Tomcat)

2. Автентифікація та доступ до веб-інтерфейсу.

Доступ до адміністративної панелі здійснюється через веб-браузер за локальною адресою http://localhost:8080/ScadaBR. На етапі входу (рис. 3.14) виконується перевірка прав доступу, що є першим рівнем захисту SCADA-системи від несанкціонованого втручання.

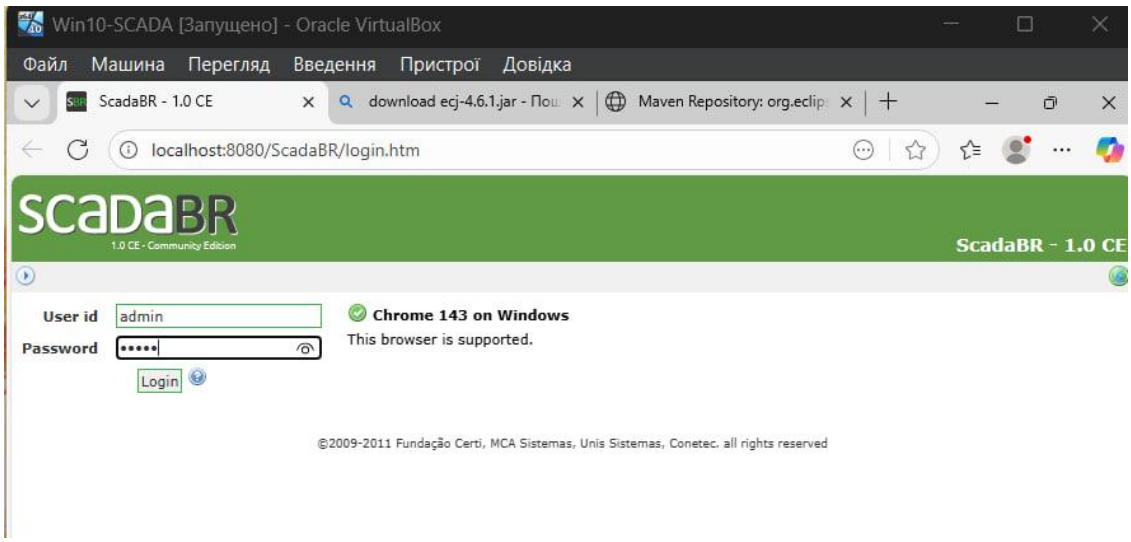


Рисунок 3.14 – Вікно автентифікації в системі ScadaBR

3. Створення джерела даних Modbus IP.

Фундаментальним етапом інтеграції є налаштування комунікаційного драйвера. Було створено нове джерело даних (Data Source) типу «Modbus IP». У параметрах конфігурації (рис. 3.15) вказано:

1.1 Host: 192.41.15.1 (адреса віртуального ПЛК).

1.2 Port: 502 (стандартний порт протоколу Modbus TCP).

1.3 Update period: 1 секунда (оптимальний інтервал для моніторингу станів вимикача без перевантаження мережевого каналу).

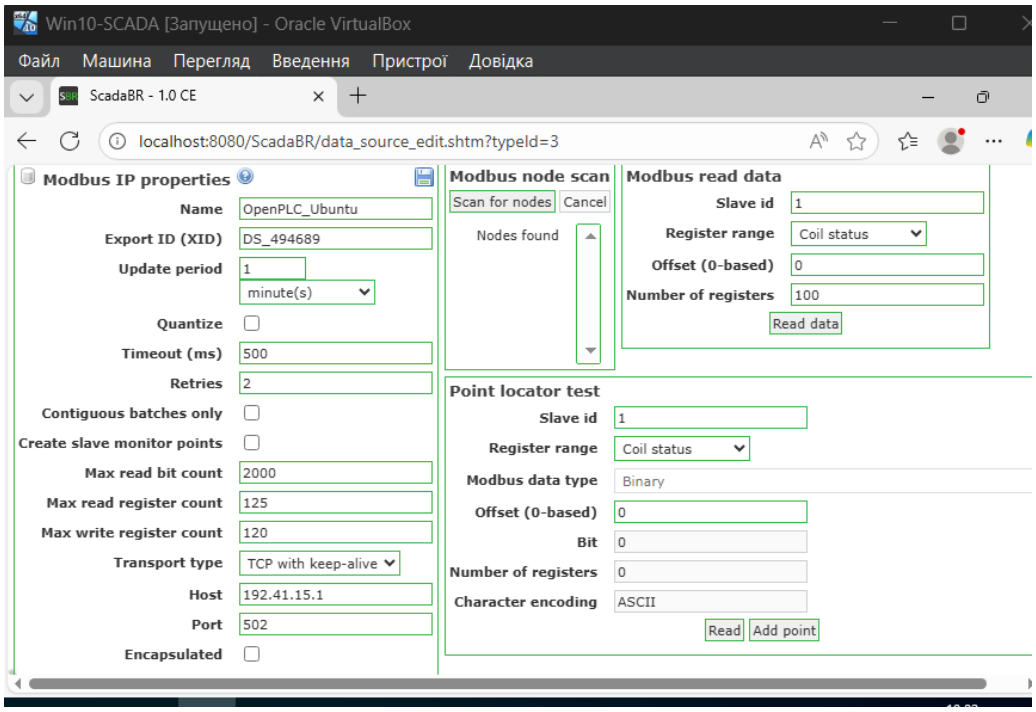


Рисунок 3.15 – Налаштування комунікаційного вузла Modbus IP

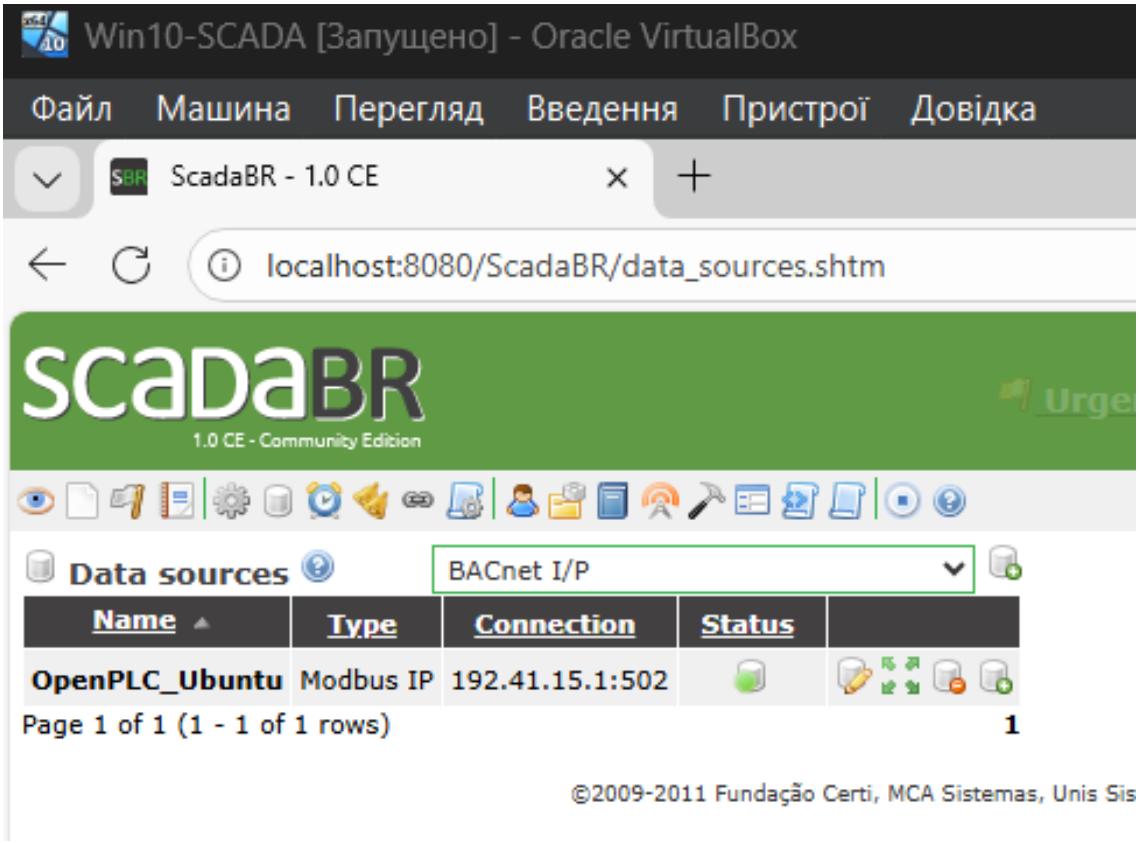


Рисунок 3.16 – Налаштування комунікаційного вузла Modbus IP

4. Верифікація мережевого обміну та сканування вузлів.

Для підтвердження коректності встановленого зв'язку було використано інструмент «Node Scan». Успішне виявлення активного Slave-вузла (рис. 3.17) підтверджує, що віртуальний комутатор VirtualBox та брандмауери вузлів налаштовані вірно.

Статус джерела даних у головному вікні змінився на зелений індикатор, що свідчить про успішну ініціалізацію сесії Modbus TCP та стабільний обмін пакетами між Windows 10 та Ubuntu.

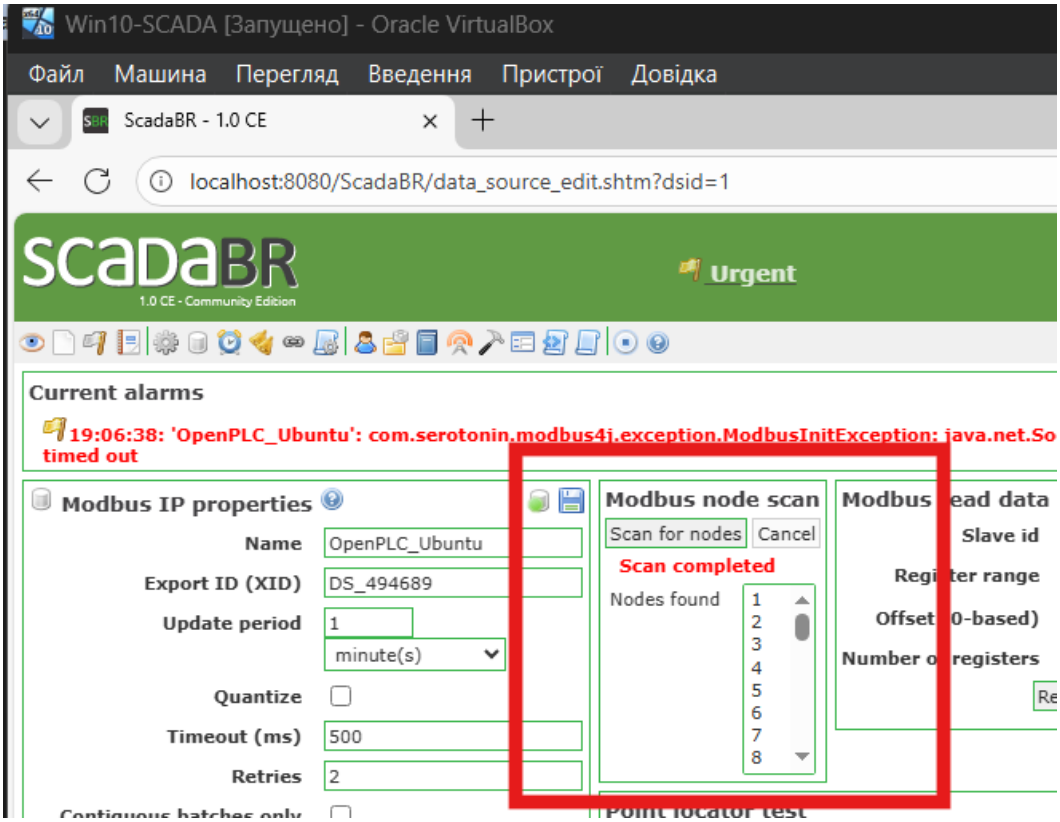


Рисунок 3.17 – Верифікація доступності ПЛК через функцію Node Scan

3.2 Практична реалізація промислової системи програмного модуля та підсистеми захисту

Для забезпечення пасивного захисту промислового сегмента та оперативного виявлення кіберзагроз було розроблено спеціалізоване програмне забезпечення системи виявлення вторгнень (IDS) мовою програмування Python. Розгорнутий програмний комплекс функціонує на захисному вузлі під керуванням ОС Ubuntu та реалізує комбінований аналіз мережевого трафіку : евристичний моніторинг інтенсивності запитів (для детекції DoS-атак) та сигнатурний аналіз прикладного рівня протоколу Modbus TCP (для виявлення несанкціонованих промислових команд та аномальної активності). Повний лістинг розробленого програмного забезпечення наведено у Додатку Б.

Оскільки промисловий протокол Modbus TCP функціонує поверх транспортного рівня TCP (порт 502) та за замовчуванням не має вбудованих засобів автентифікації, розроблений скрипт використовує аналізатор пакетів бібліотеки Scapy. Для того, щоб система могла коректно розбирати інформативні поля

прикладного рівня, утиліта виконує прив'язку портів за допомогою вбудованих інструментів. Цю конфігурацію зв'язування шарів та ініціалізацію аналізу представлено на лістингу програми (рис. 3.18).

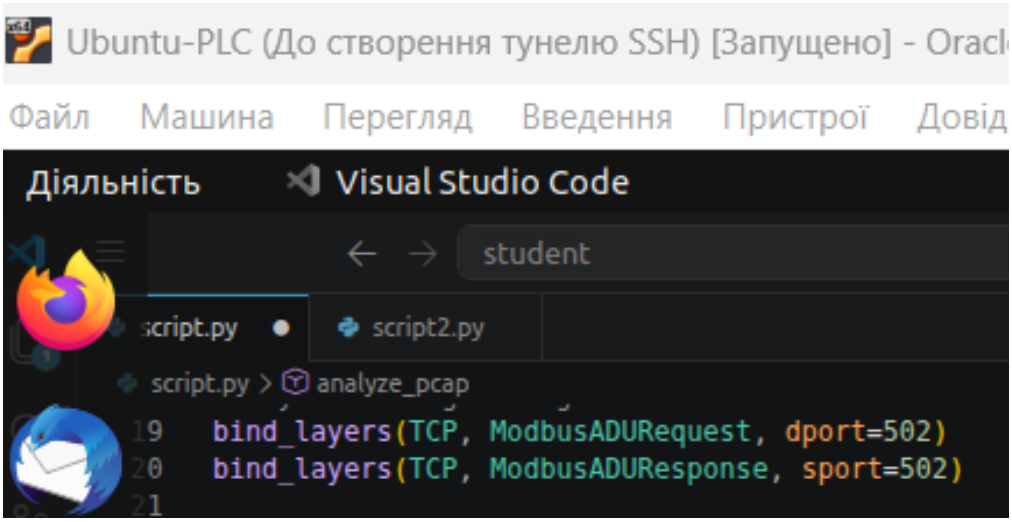


Рисунок 3.18 – Налаштування прив’язки рівнів протоколу Modbus TCP

Завдяки технології глибокої інспекції пакетів (DPI), реалізованій через конструкцію bind_layers(), здійснюється автоматичне мапування сирих байтів транспортного рівня на змінні прикладного рівня промислового стека без ручного парсингу байтових зрушень.

Важливою складовою евристичного аналізу системи є функція моніторингу частоти запитів, програмний код якої наведено на рисунку 3.19. Вона працює за моделлю «ковзного часового вікна» тривалістю time_window = 1.0 секунда. Алгоритм динамічно очищає пам'ять від застарілих часових міток для запобігання витоку оперативної пам'яті (Memory Leak) під час тривалого моніторингу. Якщо ліміт інтенсивності з однієї унікальної IP-адреси перевищує константу threshold = 20, функція повертає значення True, що є тригером для генерації сповіщення.

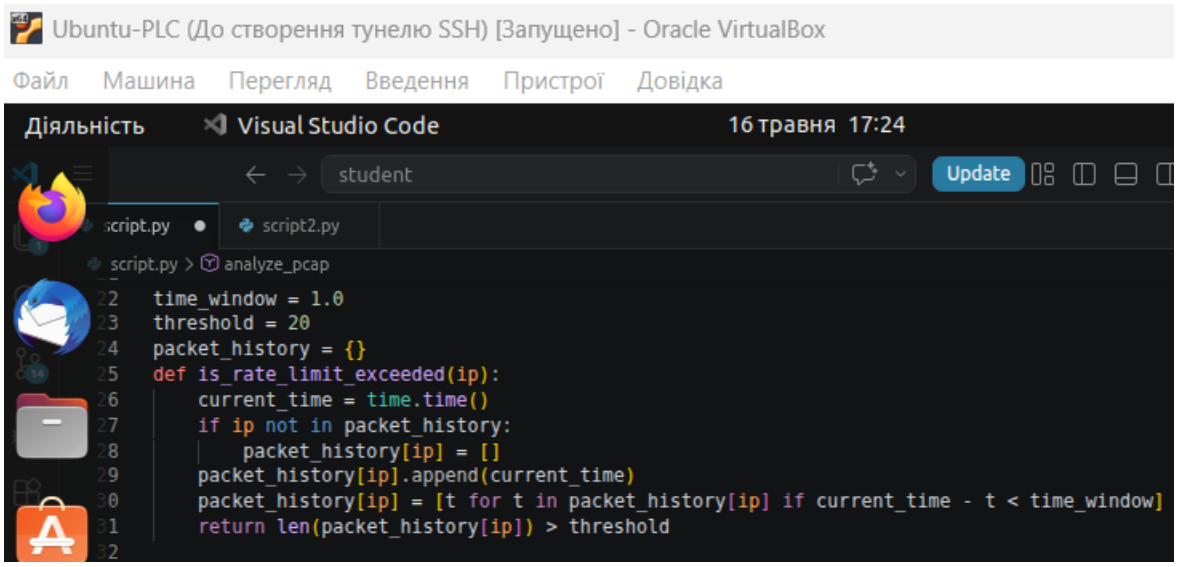


Рисунок 3.19 – Програма реалізації алгоритму DPI для виявлення аномалій мережевого трафіку та DoS-атак

Головний модуль системи process_packet(), логіку якого представлено на рисунках 3.20 та 3.21, здійснює послідовну багаторівневу сигнатурну перевірку рівнів загрози:

1. Рівень 1 (Мережева автентифікація).

Скрипт перевіряє джерело запиту src_ip. Будь-який пакет, відправлений з адреси, що не входить до «білого списку» легітимних пристроїв стенда (SCADA-система ScadaBR та контролер OpenPLC), класифікується системою як потенційна спроба несанкційонованого доступу.

2. Рівень 2 (Прикладний сигнатурний аналіз).

За допомогою методу haslayer() скрипт шукає протокол Modbus. Якщо його виявлено, аналізатор перевіряє код функції (layer.funcCode). До списку критичних функцій внесено команди модифікації пам'яті: 5 (запис одного коїла), 6 (запис одного регістра) та 15, 16 (множинний запис). Моніторинг цих кодів у межах модуля DPI дозволяє зафіксувати та заблокувати спроби несанкціонованої зміни параметрів технологічного процесу зловмисником. Також контролюється код 1 (Read Coils) для виявлення спроб розвідки мережі та сканування регістрів.

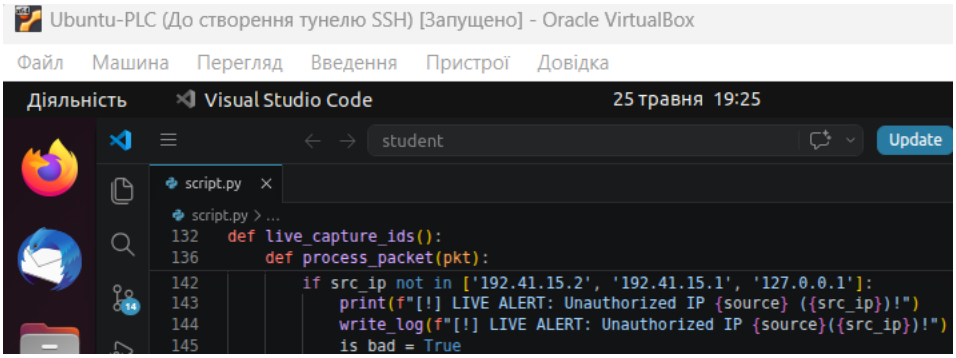


Рисунок 3.20 – Програмна реалізація головного модуля обробки пакетів та перевірки мережевої автентифікації вузлів

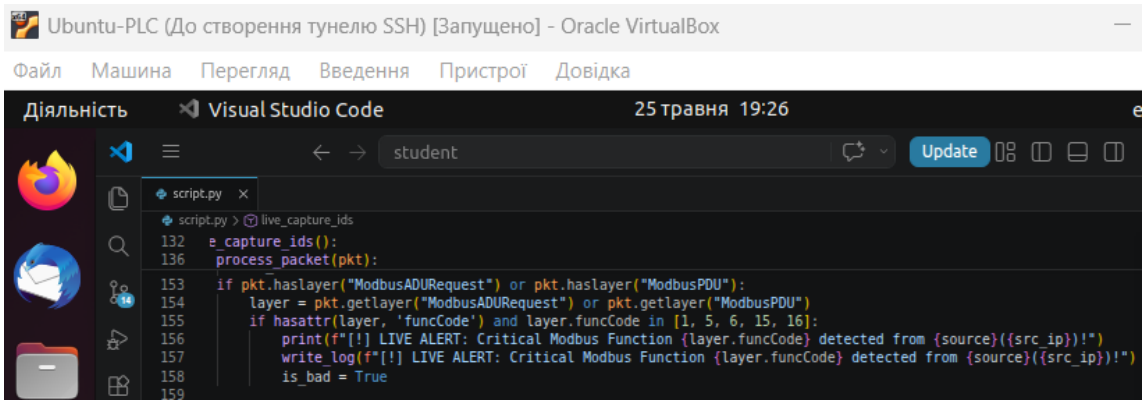


Рисунок 3.21 – Логіка прикладного сигнатурного аналізу та фільтрації критичних кодів функцій Modbus TCP

Окрім генерації попередження, система реалізує механізм автоматичного реагування на атаку шляхом блокування IP-адреси порушника засобами iptables. Фрагмент програмної реалізації наведено на рисунку 3.22.

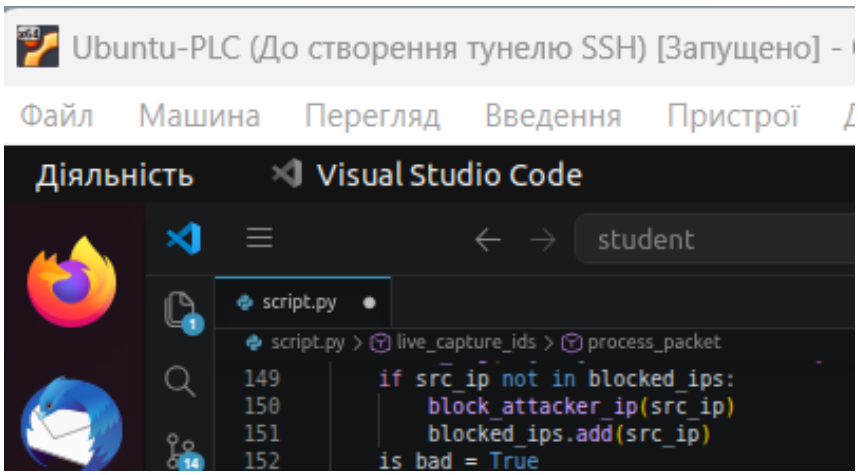


Рисунок 3.22 – Фрагмент реалізації автоматичного блокування IP-адреси порушника

3.3 Моделювання атак на промислову систему

3.3.1 Моделювання легітимної взаємодії з PLC через SSH/Netmiko

Для комплексної перевірки працездатності розробленої системи IDS на атакуючому вузлі Kali Linux (IP-адреса 192.41.15.3) було реалізовано спеціалізовану утиліту автоматизованого тестування мовою Python. Програмний модуль поєднує функції дистанційного керування PLC-контролером та генерації аномального Modbus TCP-трафіку.

Для реалізації дистанційного керування використовується клас NetCon, який ініціює SSH-з'єднання з вузлом OpenPLC під керуванням Ubuntu 22.04 за допомогою бібліотеки Netmiko та механізму ключової автентифікації. Такий режим дозволяє моделювати легітимну взаємодію адміністратора або інженера з PLC-контролером.

Фрагмент програмної реалізації SSH-з'єднання наведено на рисунку 3.23.

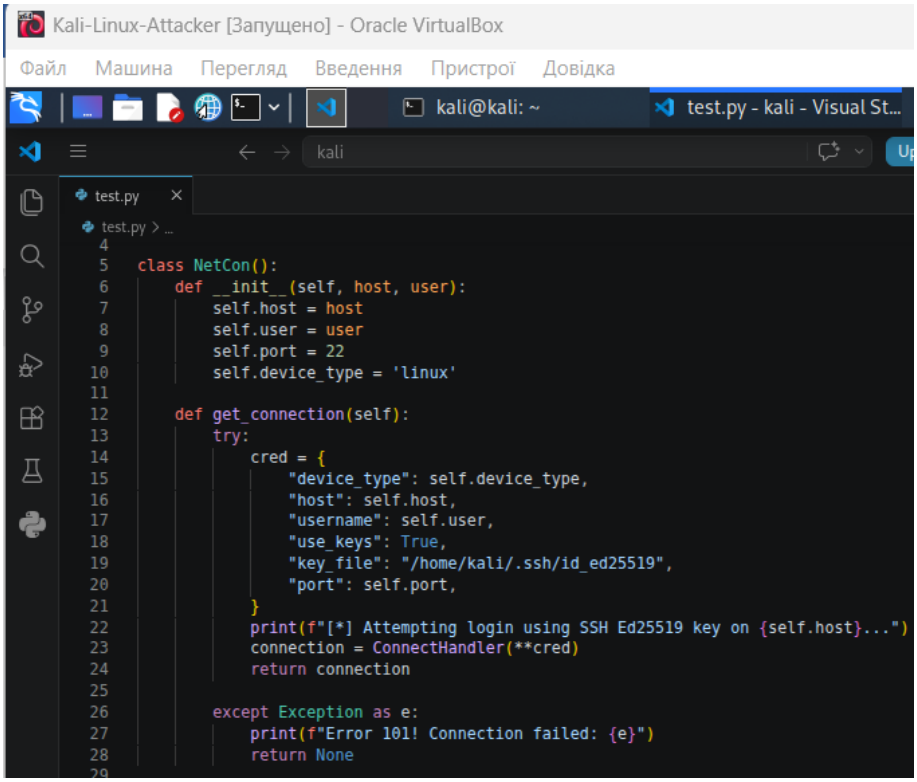


Рисунок 3.23 – Фрагмент програмної реалізації SSH-з'єднання з OpenPLC через Netmiko

Функція connection_to_plc() реалізує інтерактивний режим керування PLC-контролером. Через SSH-з'єднання виконується запуск утиліти modpoll, яка надсилає Modbus-команди на локальний інтерфейс OpenPLC. У межах тестування

користувач може виконувати команди запуску, зупинки та перемикання станів технологічного процесу.

Фрагмент програмної реалізації функції дистанційного керування наведено на рисунку 3.24.

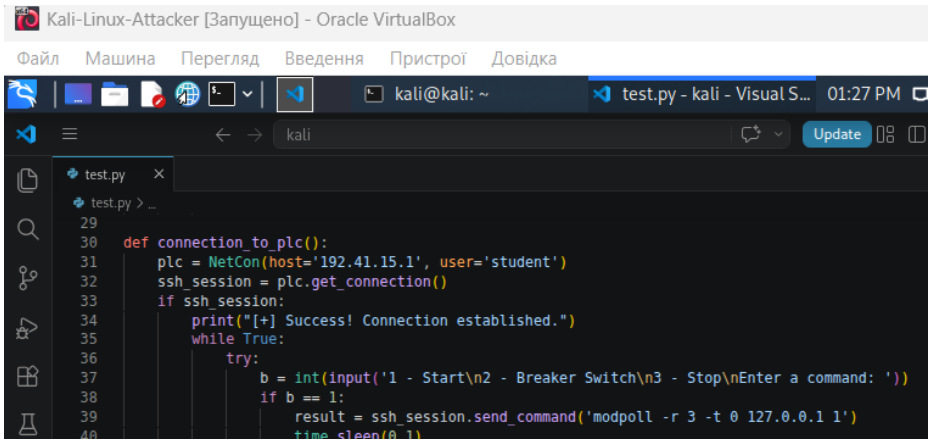


Рисунок 3.24 – Реалізація функції дистанційного керування OpenPLC через SSH/Netmiko

3.3.2 Генерація аномального Modbus TCP-трафіку

Окрім режиму легітимного керування, у програмній утиліті реалізовано функцію attacks(), призначену для генерації аномального мережевого трафіку та моделювання активних кібератак на промисловий сегмент.

У даному режимі підключення до PLC-контролера здійснюється безпосередньо через TCP-порт 502 за допомогою бібліотеки PyModbus. Після встановлення з’єднання скрипт багаторазово надсилає команди запису до coils та holding registers, зокрема:

1. write_coil().
2. write_register().
3. write_coils().
4. write_registers().

Такий підхід дозволяє сформувати інтенсивний потік критичних Modbus-команд та змоделювати аномальну активність у мережі промислового контролера.

Фрагмент програмної реалізації генерації аномального Modbus TCP-трафіку наведено на рисунку 3.25.

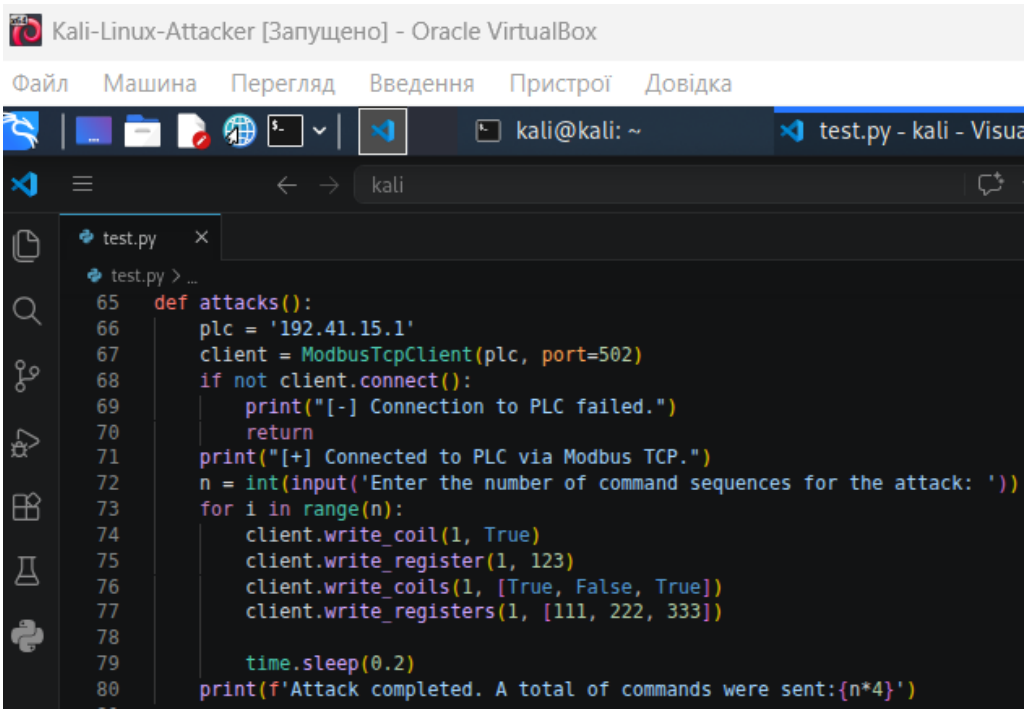


Рисунок 3.25 – Реалізація функції генерації аномального Modbus TCP-трафіку

У межах одного циклу виконуються чотири команди запису, тому загальна кількість сформованих Modbus-запитів визначається як n × 4, де n – кількість циклів, введена користувачем. Для запобігання повному перевантаженню SCADA-системи між циклами використовується затримка time.sleep(0.2).

3.3.3 Моделювання MITM-атаки методом ARP Spoofing

У рамках дослідження вразливостей промислового протоколу Modbus TCP було проведено моделювання атаки типу «людина посередині» (Man-in-the-Middle, MITM). Даний тип атаки дозволяє зловмиснику перехоплювати мережевий трафік між SCADA-системою та PLC-контролером, виконуючи його аналіз або модифікацію.

Реалізація атаки здійснювалась методом ARP Spoofing. Суть даного підходу полягає у надсиланні підроблених ARP-відповідей вузлам мережі з метою зміни ARP-таблиць та перенаправлення мережевого трафіку через атакуючий вузол.

Для підвищення реалістичності сценарію тестування у віртуальному стенді використовувались два вузли Kali Linux:

1. Вузол 192.41.15.3 – генерація аномального Modbus TCP-трафіку та DoS/flood-активності;

2. Вузол 192.41.15.4 – реалізація MITM-атаки та перехоплення трафіку.

Такий підхід дозволив змоделювати комплексний сценарій мережевого впливу на промисловий сегмент.

Для реалізації MITM-атаки використовувався інструмент arpspoof, який циклічно надсилав підроблені ARP-відповіді вузлам OpenPLC (192.41.15.1) та ScadaBR (192.41.15.2). Паралельно на атакуючому вузлі було активовано механізм IP Forwarding для пересилання мережних пакетів між вузлами.

Результат реалізації атаки ARP Spoofing у середовищі Kali Linux наведено на рисунку 3.26.

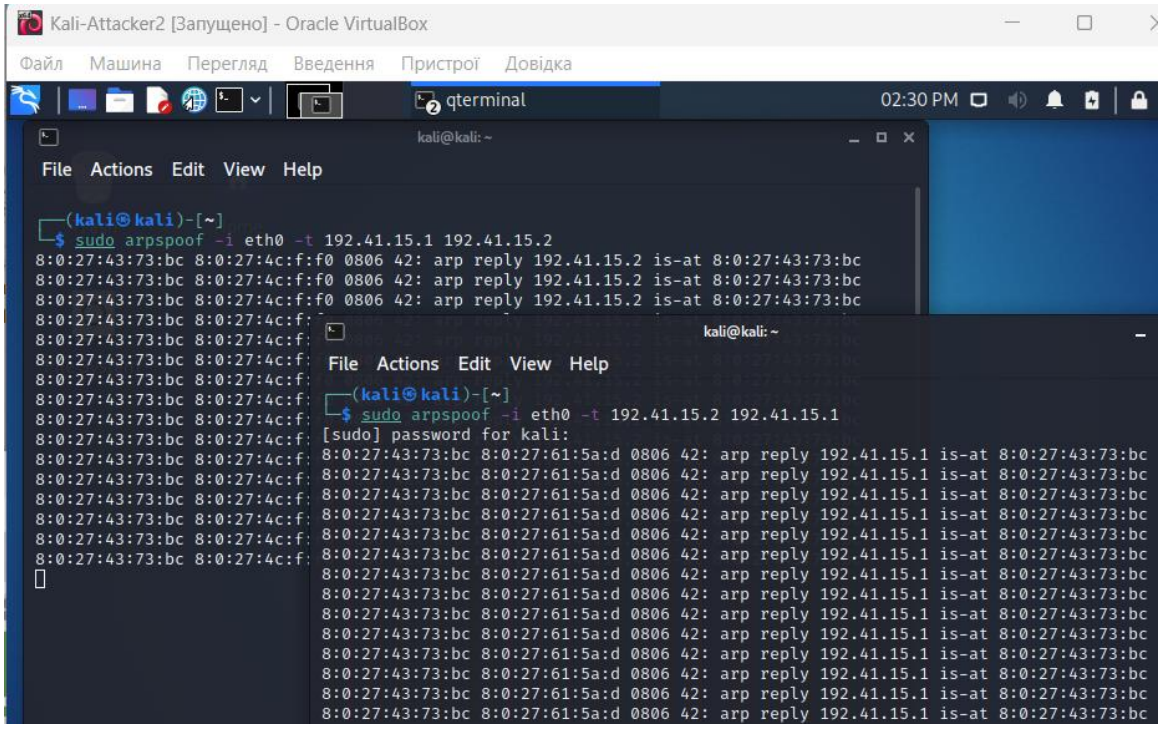


Рисунок 3.26 – Реалізація атаки ARP-spoofing у середовищі Kali Linux

3.4 Тестування стійкості розробленої системи підсистеми захисту

Для перевірки стійкості розробленої системи захисту та оцінки можливостей IDS було проведено серію експериментів із моделюванням мережних атак на промисловий сегмент Modbus TCP.

Під час проведення MITM-атаки було зафіксовано зміну ARP-таблиці вузлів мережі, у результаті чого декілька IP-адрес були асоційовані з MAC-адресою вузла Kali Linux, який виконував ARP spoofing. Це підтверджує успішне

перенаправлення мережевого трафіку через атакуючий вузол та можливість його перехоплення. Результат наведено на рисунку 3.27.

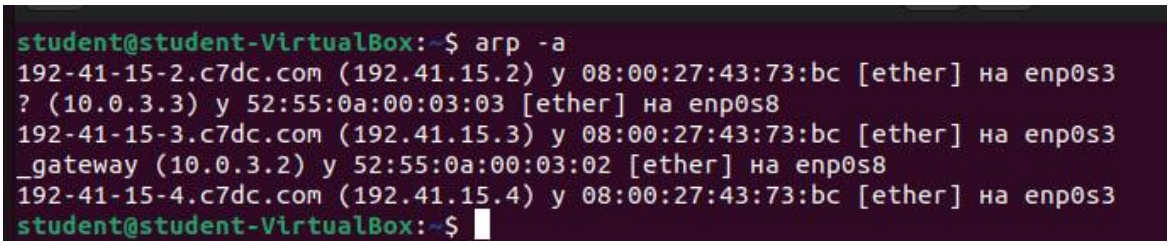


Рисунок 3.27 – Результат підміни ARP-таблиці під час MITM-атаки

Для аналізу перехопленого мережевого трафіку було використано програмний аналізатор Wireshark. Захоплення пакетів здійснювалося в режимі реального часу. Завдяки увімкненому раніше змішаному режиму (Promiscuous Mode) на адаптері, Kali Linux забезпечував перехоплення мережевого трафіку в режимі реального часу. Для фільтрації було застосовано дисплейний фільтр modbus, що дозволило відокремити службовий трафік від іншого мережевого шуму (рис. 3.28).

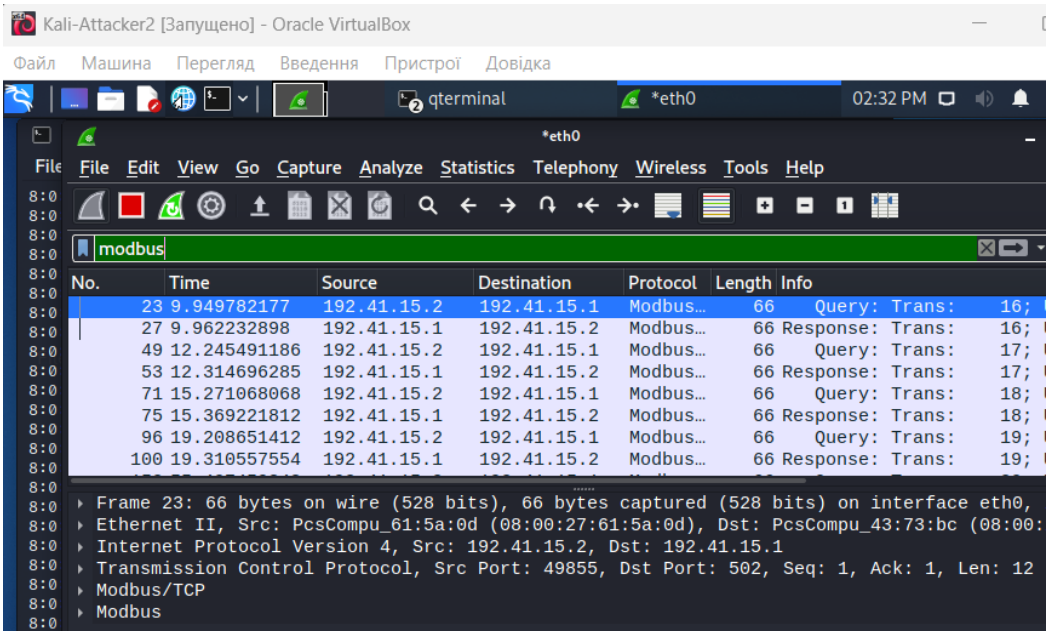


Рисунок 3.28 – Перехоплення технологічного трафіку Modbus TCP у реальному часі

Як показано на рис. 3.28, у перехопленому трафіку чітко ідентифікуються пакети протоколу Modbus TCP, що безперервно передаються між вузлами з періодом опитування в 1 секунду.

Для детального аналізу структури промислового пакета було досліджено структуру запиту (Query) та відповіді (Response). На рис. 3.29 представлено детальний розбір запиту типу Query.

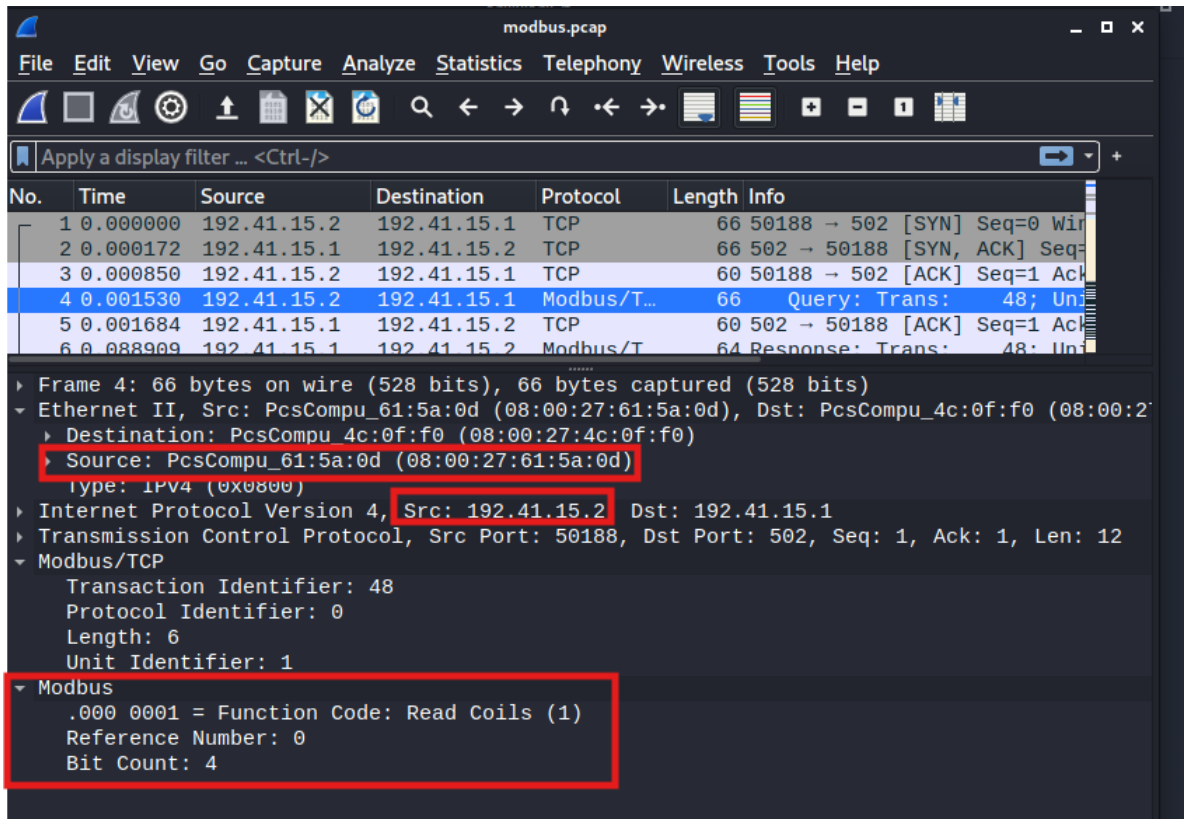


Рисунок 3.29 – Структура перехопленого пакета Modbus TCP Query

При аналізі заголовків пакета (рис. 3.29) чітко видно аномалію канального (L2) рівня моделі OSI. Пакет, який логічно призначений для IP-адреси ПЛК (192.41.15.1), на канальному рівні містить Destination MAC-адресу (08:00:27:4c:ef:f0), яка через утиліту arpspoof була асоційована у SCADA-системи із контролером. Це забезпечує фізичне перенаправлення технологічного трафіку через транзитний інтерфейс станції атакуючого (Kali Linux), підтверджуючи успішність реалізації атаки типу «Людина посередині» (MITM), при цьому IP-адреси відправника та отримувача в полях третього рівня (L3) залишаються незмінними для збереження працездатності сесії.

Аналогічним чином було проаналізовано пакет-відповідь (Response) від контролера, який повертався до SCADA-системи через транзитний вузол зловмисника (рис. 3.30).

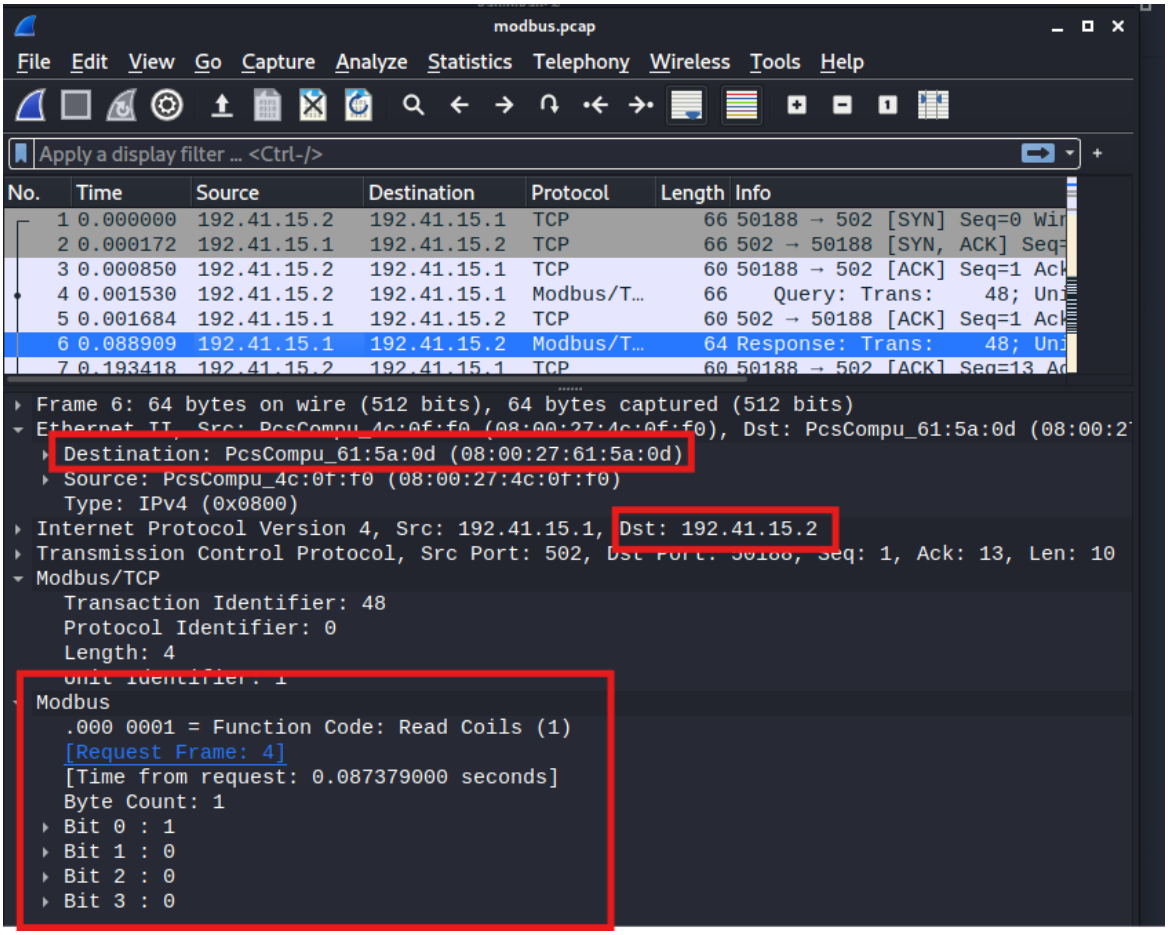


Рисунок 3.30 – труктура перехопленого пакета Modbus TCP Response

На рис. 3.30 представлено відповідь сервера, що містить значення бітів технологічного процесу. Дані передаються у відкритому вигляді без будь-якого шифрування чи контролю цілісності на прикладному рівні. Це дозволяє зловмиснику безперешкодно отримати точну інформацію про поточний стан об’єкта керування.

Для верифікації зчитаних даних було проведено зіставлення перехоплених бітових значень із реальними інтерфейсами оператора та контролера. На рис. 3.31 представлено поточний стан елемента керування у ScadaBR та OpenPLC Runtime.

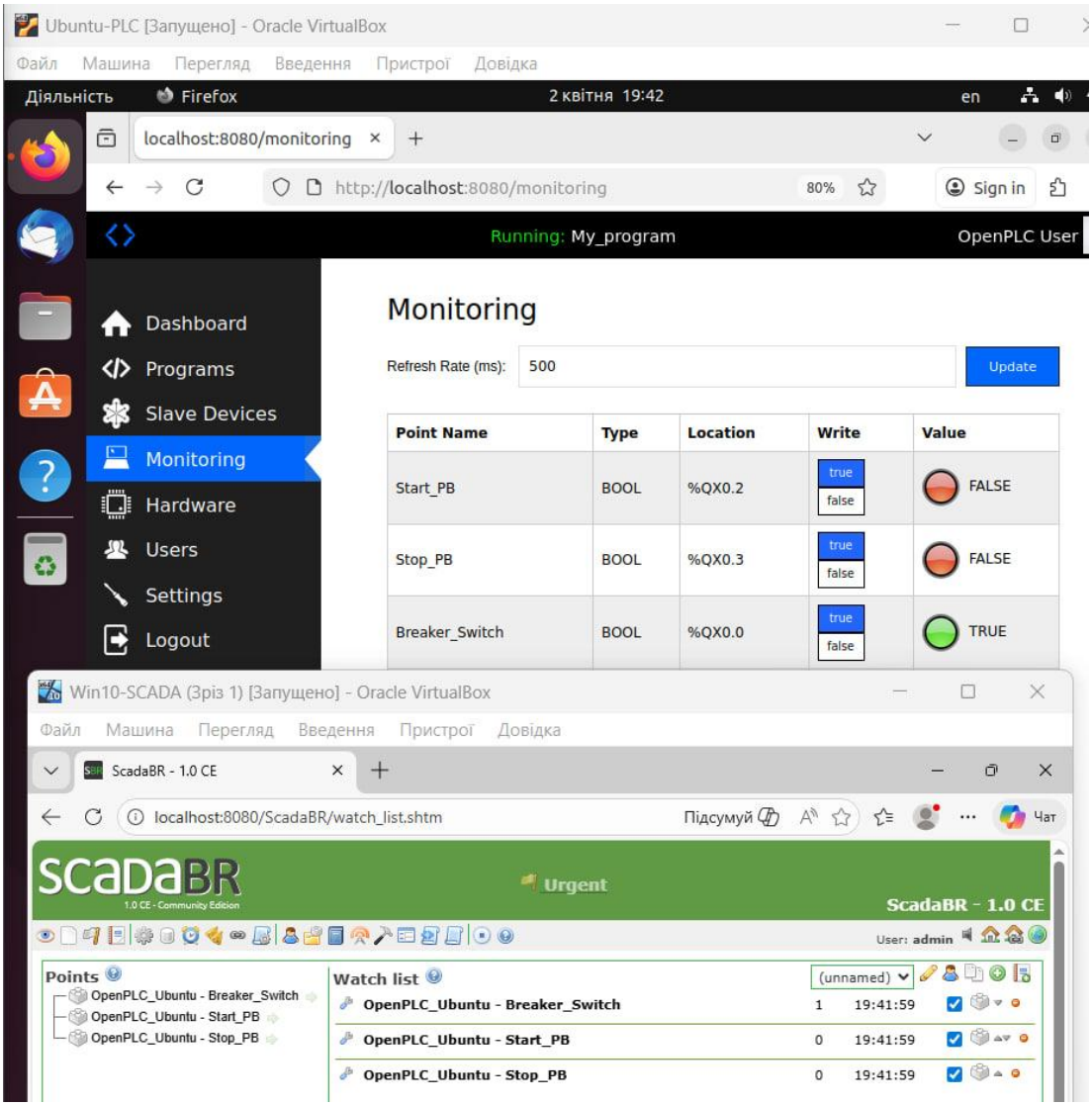


Рисунок 3.31 – Синхронізація технологічних станів у вікнах ScadaBR та OpenPLC Runtime

Зміна стану елемента Breaker_Switch (значення TRUE/FALSE) на рис. 3.30 призводить до миттєвого формування відповідних запитів у протоколі Modbus TCP. Це підтверджує повну працездатність стенда та доводить можливість віддаленого контролю над станом об’єкта.

У процесі дослідження було остаточно підтверджено та систематизовано критичні недоліки протоколу Modbus TCP. У перехоплених пакетах можуть бути ідентифіковані такі ключові поля:

1. Функціональні коди (Function Code), що визначають тип операції (читання або запис).

2. Адреси регістрів (котушок пам'яті), які безпосередньо вказують на конкретні виконавчі механізми.

3. Значення змінних (Data), що відображають фізичний стан техпроцесу.

Наприклад, у запитах типу Read Coils передаються адреси дискретних входів/виходів, тоді як у відповідях контролера містяться їхні поточні значення (0 або 1), що відображає стан кнопок керування або силових вимикачів.

Аналіз перехопленого трафіку експериментально підтвердив, що за умов відсутності додаткових засобів захисту, зловмисник в ізольованому сегменті має повну видимість процесу обміну даними (пасивна атака). Це створює передумови для реалізації більш небезпечних активних атак, таких як Modbus Injection (підміна команд керування в обхід SCADA) або атак типу «відмова в обслуговуванні» (DoS-атака на порт 502).

Таким чином, проведене моделювання атаки дозволило наочно продемонструвати базову вразливість архітектури Modbus TCP (відсутність автентифікації та шифрування) та обґрунтувати критичну необхідність впровадження систем виявлення вторгнень (IDS) для захисту технологічного периметра.

Для підтвердження працездатності розробленої системи виявлення вторгнень та верифікації логіки, наведеної на лістингах коду (рис. 3.19 – 3.21), було проведено запуск скрипта на вузлі моніторингу в режимі реального часу. Нижче наведено протокольовані результати виконання коду (вивід консолі IDS) під час симуляції легітимної роботи та кібератак:

Результат 1. Моніторинг штатного режиму (Легітимна взаємодія).

Під час проведення експерименту було здійснено перевірку реакції системи виявлення вторгнень на штатні та потенційно небезпечні запити. Як відображено на рисунку 3.31, модуль моніторингу успішно фіксує інформаційні пакети (повідомлення класу [INFO]), які надходять від робочої станції SCADA з IP-адресою 192.41.15.2 на адресу ПЛК 192.41.15.1.

Проте, під час спроби відправки команд модифікації пам'яті (читання/запис прапорців через коди функцій 5 та 1), модуль глибокого аналізу пакетів (DPI) система зафіксувала використання критичних функцій прикладного рівня. Система було сформовано повідомлення на подію та згенерувала тривожне сповіщення

[LIVE ALERT: Critical Modbus Function 5/1 detected!]. Це підтверджує здатність розробленої IDS здійснювати точний сигнатурний аналіз індустріального трафіку та виявляти потенційно деструктивні команди керування в реальному часі.

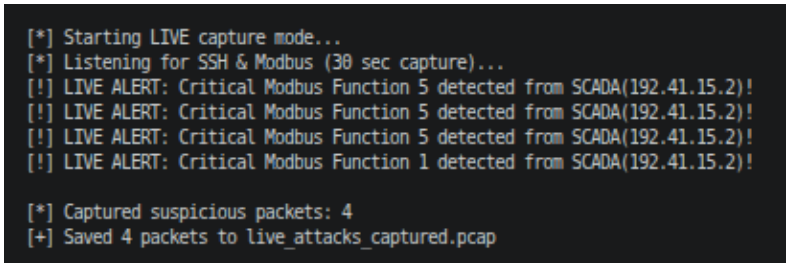


Рисунок 3.32 – Фіксація інциденту перевищення ліміту інтенсивності трафіку [LIVE ALERT] у консолі моніторингу

Результат 2. Детекція дій внутрішнього порушника (Спроба несанкційованого підключення за лістингом рис. 3.19).

При спробі стороннього вузла (атакуючої станції Kali Linux з IP 192.41.15.3) надіслати запит у промисловий сегмент, головний модуль відповідно до логіки коду на рисунку 3.33 ідентифікує невідому адресу та генерує тривожний сигнал LIVE ALERT.

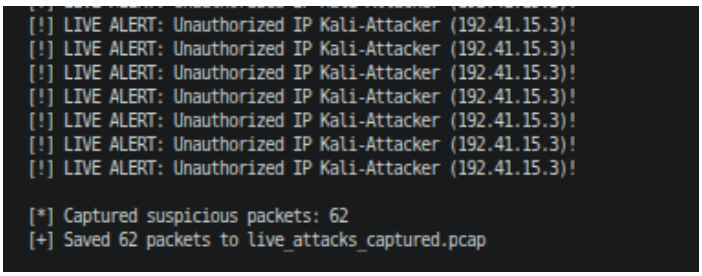


Рисунок 3.33 – Результат детекції інцидентів безпеки [LIVE ALERT] під час імітації дій внутрішнього зловмисника

Результат 3. Виявлення DoS-атаки та флуду пакетів (Евристичний аналіз).

Для підтвердження ефективності роботи евристичного модуля аналізу трафіку було проведено симуляцію DoS-атаки (генерації інтенсивного потоку запитів) з неавторизованого пристрою. Алгоритм ковзного часового вікна, програмну реалізацію якого наведено у попередніх лістингах (рис. 3.18 та рис. 3.19), здійснює підрахунок кількості пакетів, що надходять від кожної IP-адреси за одиницю часу.

Результат успішного виявлення та динамічного блокування DoS-флуду в реальному часі відображено на рисунку 3.34. Як видно з консолі моніторингу системи IDS, у момент різкого збільшення інтенсивності надходження пакетів з хоста порушника 192.41.15.3, система миттєво фіксує перевищення встановленого безпечного порогу (20 пакетів/сек). Евристичний аналізатор миттєво ідентифікує спробу перевантаження стеку протоколів ПЛК та генерує IDS виявила перевищення порогу [!!!] DOS ALERT: 192.41.15.3 sends too many packets! Це демонструє здатність розробленої IDS виявляти атаки типу «відмова в обслуговуванні».

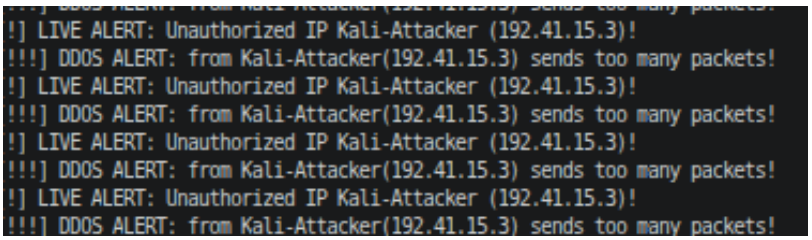


Рисунок 3.34 – Консольний лог системи IDS під час комплексної детекції DoS-атаки та несанкціонованої активності порушника

Результат автоматичного додавання правила блокування до таблиці iptables наведено на рисунку 3.35.

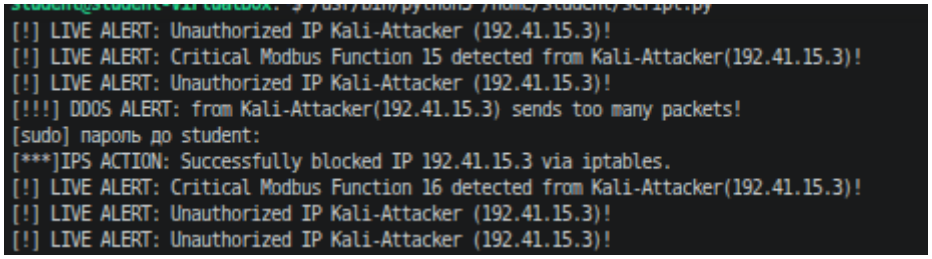


Рисунок 3.35 – Результат автоматичного додавання правила блокування IP-адреси засобами iptables

Після спрацювання механізму реагування система додає правило фільтрації до таблиці Linux Firewall, що підтверджується результатом виконання команди iptables -L -n, наведеним на рисунку 3.36.

```
student@student-VirtualBox:~$ sudo iptables -L -n
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.41.15.3          0.0.0.0/0

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  192.41.15.3          0.0.0.0/0
```

Рисунок 3.36 – Результат автоматичного додавання правила блокування IP-адреси до iptables

Результат 4. Комплексне тестування IDS при мультиатаці.

Для перевірки працездатності IDS в умовах комбінованого мережевого впливу було реалізовано сценарій мультиатаки, який поєднував генерацію надлишкового Modbus TCP-трафіку, надсилання критичних команд та несанкціоновану активність з боку атакуючого вузла Kali Linux. У результаті система IDS одночасно зафіксувала перевищення інтенсивності трафіку, використання критичних Function Code та появу неавторизованої IP-адреси. Результат роботи системи моніторингу наведено на рисунку 3.37.

```
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Unauthorized IP Kali-Attacker (192.41.15.3)!
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Unauthorized IP Kali-Attacker (192.41.15.3)!
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Unauthorized IP Kali-Attacker (192.41.15.3)!
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Critical Modbus Function 16 detected from Kali-Attacker(192.41.15.3)!
[!] LIVE ALERT: Unauthorized IP Kali-Attacker (192.41.15.3)!
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Unauthorized IP Kali-Attacker (192.41.15.3)!
[!!!] DDOS ALERT: from Kali-Attacker(192.41.15.3) sends too many packets!
[!] LIVE ALERT: Critical Modbus Function 5 detected from SCADA(192.41.15.2)!
[!] LIVE ALERT: Critical Modbus Function 5 detected from SCADA(192.41.15.2)!
[!] LIVE ALERT: Critical Modbus Function 5 detected from SCADA(192.41.15.2)!

[*] Captured suspicious packets: 1052
[+] Saved 1052 packets to live_attacks_captured.pcap

---------- MAIN MENU ----------
```

Рисунок 3.37 – Результат комплексного виявлення мультиатаки системою IDS

Для узагальнення результатів тестування розробленої системи IDS та оцінки ефективності виявлення різних типів аномальної активності було сформовано порівняльну таблицю, наведену в таблиці 3.3.

Таблиця 3.3

Порівняльний аналіз результатів виявлення кібератак системою IDS

Тип атаки	Інструмент	Ознаки в трафіку	Реакція IDS	Результат
ARP Spoofing	arpspoof	Зміна MAC	LIVE ALERT	Виявлено
DoS/Flood	PyModbus	>20pkt/sec	DDOS ALERT+ iptables	Заблоковано
Unauthorized Access	PyModbus	Критичні funcCode	LIVE ALERT	Виявлено
MITM	arpspoof+sniffing	Дублювання MAC	Аномалія	Виявлено

Як видно з таблиці 3.3, розроблена система IDS підтверджує можливість виявлення мережеві аномалії, пов’язані з MITM-атаками, генерацією надлишкового трафіку та несанкціонованими Modbus-командами, а також реалізує автоматичне реагування шляхом блокування IP-адрес порушника.

ВИСНОВКИ

У даній кваліфікаційній роботі розроблено та реалізовано віртуальний дослідницький полігон для аналізу кібербезпеки промислових систем керування та дослідження вразливостей протоколу Modbus TCP.

У першому розділі було проведено аналіз основних аспектів забезпечення кібербезпеки в промислових автоматизованих системах керування технологічними процесами (АСУ ТП). Досліджено специфіку та архітектуру промислового протоколу Modbus TCP, а також детально розглянуто його критичні вразливості, зумовлені відсутністю вбудованих механізмів шифрування та автентифікації. Обґрунтовано необхідність розробки спеціалізованих засобів виявлення вторгнень (IDS), здатних здійснювати глибокий аналіз промислового трафіку для захисту критичної інфраструктури від внутрішніх та зовнішніх порушників.

У другому розділі було спроектовано та розгорнуто комплексне імітаційне середовище на базі платформи Oracle VM VirtualBox. Створено структуру лабораторного стенда, що включає програмний логічний контролер OpenPLC, SCADA-систему для візуалізації технологічного процесу, робочу станцію оператора та атакуючий вузол на базі Kali Linux. Описано принципи взаємодії компонентів стенда та сформовано модель комбінованих деструктивних впливів на промислову мережу.

У третьому розділі було програмно реалізовано прототип системи виявлення вторгнень (IDS) мовою Python з використанням бібліотеки Scapy. Виконано налаштування модулів глибокого аналізу пакетів (DPI) для сигнатурної детекції критичних функцій Modbus TCP та евристичного аналізатора на основі алгоритму ковзного часового вікна для виявлення DoS-атак. Проведено серію експериментів та тестування системи за допомогою автоматизованих скриптів Netmiko та PyModbus. Верифікація результатів через аналізатор трафіку Wireshark підтвердила високу точність фіксації інцидентів (Unauthorized IP, DoS Alert) та стабільність роботи розробленого засобу захисту.

Проведена кваліфікаційна робота успішно вирішила поставлені завдання. Було здійснено комплексний аналіз загроз для протоколу Modbus TCP, побудовано

віртуалізований стенд АСУ ТП для дослідження аномалій безпеки та успішно реалізовано власну систему сигнатурно-евристичного моніторингу трафіку. Розроблена IDS забезпечує миттєве виявлення несанкційованих дій внутрішнього порушника, надійно фіксує спроби перевантаження мережевого стеку ПЛК та нелегальні модифікації критичних функцій керування. Проведене моделювання підтвердило ефективність запропонованих рішень та їх можливість використання як додаткового засобу моніторингу та виявлення аномалій у промислових мережах.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Інжиніринг АСУ ТП: основні поняття та визначення [Електронний ресурс] – Режим доступу до ресурсу: https://www.ksimex.ua/subprojects/apcs-engineering

2. Installing OpenPLC Runtime on Linux Systems [Електронний ресурс] – Режим доступу до ресурсу: https://autonomylogic.com/docs/installing-openplc-runtime-on-linux-systems/

3. ScadaBR – Open Source SCADA System [Електронний ресурс] – Режим доступу до ресурсу: https://sourceforge.net/projects/scadabr/

4. Програмне забезпечення WebAccess: система SCADA [Електронний ресурс] – Режим доступу до ресурсу: https://www.proxis.ua/uk/catalog/webaccess/

5. Development of Secure Modbus TCP (Modbus-S) using Encryption, Hashing and Digital Signatures [Електронний ресурс] / Journal of Physics: Conference Series – Режим доступу до ресурсу: https://iopscience.iop.org/article/10.1088/1742-6596/1213/5/052058/pdf

6. Anomaly Detection in SCADA Systems [Електронний ресурс] / D. Gamez – Режим доступу до ресурсу: https://www.davidgamez.eu/papers/BighamGamezLu03_SCADAAnomalyDetection.pdf

7. Інформаційна безпека інформаційно-комунікаційних систем [Електронний ресурс] / Державний університет телекомунікацій – Режим доступу до ресурсу: https://dut.edu.ua/uploads/l_488_73227512.pdf

8. Advancements in Anomaly Detection for SCADA Networks [Електронний ресурс] / IEEE Xplore – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/10461058

9. Deep Learning-Based Anomaly Detection in Industrial Control Systems [Електронний ресурс] / MDPI Electronics – Режим доступу до ресурсу: https://www.mdpi.com/2079-9292/11/14/2184

10. Töpke C. Network Programming and Automation Essentials: Get started in the realm of network automation using Python and Go. Birmingham : Packt Publishing, 2023. 296 p.

11. Ballmann B. Understanding Network Hacks: Attack and Defense with Python. Berlin : Springer, 2021. 229 p.

12. Scapy and Python Part 1 – Install, Run, Sniff : video // YouTube channel “danscourses”. – 2020. – Available at:

https://www.youtube.com/watch?v=EuTAmtMGdNU

13. Intro to Scapy – Sniffing : video // YouTube channel “Brief”. – 2022. – Available at: https://www.youtube.com/watch?v=f0vpwwNAcdI

14. Evangeliou G. Intrusion Detection in Industrial Control Systems Based on Modbus TCP Protocol. – University of Piraeus, Department of Digital Systems, 2021.

15. PyModbus Documentation. – The Python Modbus Stack Documentation, 2024.

16. Khawaja G. Kali Linux Penetration Testing Bible. – Indianapolis : Wiley, 2022. – 944 p.

17. ScadaBR User Manual. – ScadaBR Documentation.

18. Purdue University. Industrial Control Systems (ICS) Security Modules. – Purdue University, Virtual Assessment of Attack Methodologies in ICS (VAAMI).

19. Stouffer K., Falco J., Scarfone K. Guide to Industrial Control Systems (ICS) Security. – NIST Special Publication 800-82 Revision 1. – Gaithersburg : National Institute of Standards and Technology, 2011.

20. Detection and Analysis of Modbus TCP/IP Vulnerabilities in Industrial Networks // IEEE Xplore Digital Library, 2024.

21. Network Automation with Netmiko. – Network Automation Documentation.

22. Scapy Documentation. – Scapy Interactive Packet Manipulation Program Documentation.

23. Industrial Control Systems and SCADA Networks : Module 6. – Department of Electrical and Electronics Engineering.

24. 9 SCADA System Vulnerabilities and How to Secure Them // Automation.com.

25. Intrusion Detection Systems : Laboratory Manual. – MRCET Department of Computer Science and Engineering.

26. Scarfone K., Mell P. Guide to Intrusion Detection and Prevention Systems (IDPS). – NIST Special Publication 800-94. – Gaithersburg : National Institute of Standards and Technology, 2007.

27. Ramalho L. Fluent Python: Clear, Concise, and Effective Programming. – 2nd ed. – Sebastopol : O'Reilly Media, 2022. – 1011 p.

28. Simon T. Python for Beginners: Master Python Programming from Basics to Advanced Level. – 2024. – 133 p.

29. Varghese S. A., Ghadim A. D., Balador A., Alimadadi Z., Papadimitratos P. Digital Twin-based Intrusion Detection for Industrial Control Systems. – 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). – 2022. – P. 323–328.

30. Nurhaida I. Python for Cyber Security Manual. – Cybersecurity for All (CS4ALL), ERASMUS-EDU-2022-CBHE. – 2024. – 1276 p. URL: https://cs4all.studentscenter.in/assets/Python%20CS/Python%20for%20Cyber%20Security%20Manual.pdf

31. Ghadim A. D. Identification of Cyberattacks in Industrial Control Systems. – Mälardalen University Licentiate Thesis. – 2023. – 74 p. URL: https://www.diva-portal.org/smash/get/diva2:1755154/FULLTEXT02.pdf

ДОДАТКИ

Додаток А

Лістинг керуючої програми ПЛК мовою Structured Text

```
PROGRAM main
  VAR
    Start_PB AT %IX0.0 : BOOL;
    Stop_PB AT %IX0.1 : BOOL;
    Breaker_Switch AT %QX0.0 : BOOL;
  END_VAR

  Breaker_Switch := NOT(Stop_PB) AND (Breaker_Switch OR Start_PB);
END_PROGRAM


CONFIGURATION Config0

  RESOURCE Res0 ON PLC
    TASK task0(INTERVAL := T#20ms,PRIORITY := 1);
    PROGRAM instance0 WITH task0 : main;
  END_RESOURCE
END_CONFIGURATION

(*DBG:char md5[] = "043cea30f3dc6fca8743bfe26fb790f2";*)
(*DBG:*)
(*DBG:void set_endianness(uint8_t value)*)
(*DBG:{*)
(*DBG: if (value == SAME_ENDIANNESS || value == REVERSE_ENDIANNESS)*)
(*DBG: {*)
(*DBG: endianness = value;*)
(*DBG: }*)
(*DBG:}*)
(*DBG:*)
(*FILE:c_blocks.h // Empty file *)
(*FILE:c_blocks.h *)
(*FILE:c_blocks_code.cpp #include <stdint.h> *)
(*FILE:c_blocks_code.cpp *)
(*FILE:c_blocks_code.cpp #ifdef ARDUINO *)
(*FILE:c_blocks_code.cpp #include <Arduino.h> *)
(*FILE:c_blocks_code.cpp #endif *)
(*FILE:c_blocks_code.cpp *)
```

(*DBG:/**)

(*DBG: * This file is part of OpenPLC Runtime*)

(*DBG: **)

(*DBG: * Copyright (C) 2023 Autonomy, GP Orcullo*)

(*DBG: * Based on the work by GP Orcullo on Beremiz for uC*)

(*DBG: **)

(*DBG: * This program is free software; you can redistribute it and/or*)

(*DBG: * modify it under the terms of the GNU General Public License*)

(*DBG: * as published by the Free Software Foundation; either version 2*)

(*DBG: * of the License, or (at your option) any later version.*)

(*DBG: **)

(*DBG: * This program is distributed in the hope that it will be useful,*)

(*DBG: * but WITHOUT ANY WARRANTY; without even the implied warranty of*)

(*DBG: * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the*)

(*DBG: * GNU General Public License for more details.*)

(*DBG: **)

(*DBG: * You should have received a copy of the GNU General Public License*)

(*FILE:c_blocks_code.cpp /**********/ *)

(*FILE:c_blocks_code.cpp /* IEC Types defs */ *)

(*FILE:c_blocks_code.cpp /**********/ *)

(*FILE:c_blocks_code.cpp *)

(*FILE:c_blocks_code.cpp typedef uint8_t IEC_BOOL; *)

(*FILE:c_blocks_code.cpp *)

(*FILE:c_blocks_code.cpp typedef int8_t IEC_SINT; *)

(*FILE:c_blocks_code.cpp typedef int16_t IEC_INT; *)

(*FILE:c_blocks_code.cpp typedef int32_t IEC_DINT; *)

(*FILE:c_blocks_code.cpp typedef int64_t IEC_LINT; *)

(*FILE:c_blocks_code.cpp *)

(*FILE:c_blocks_code.cpp typedef uint8_t IEC_USINT; *)

(*FILE:c_blocks_code.cpp typedef uint16_t IEC_UINT; *)

(*FILE:c_blocks_code.cpp typedef uint32_t IEC_UDINT; *)

(*FILE:c_blocks_code.cpp typedef uint64_t IEC_ULINT; *)

(*FILE:c_blocks_code.cpp *)

```
(*DBG: * along with this program; If not, see <http://www.gnu.org/licenses/>.*)

(*DBG: **)

(*DBG: */*)

(*DBG:*)

(*DBG:#include <stdbool.h>*)

(*DBG:*)

(*DBG:#include "iec_types_all.h"*)

(*DBG:#include "POUS.h"*)

(*DBG:*)

(*DBG:#define SAME_ENDIANNESS      0*)

(*DBG:#define REVERSE_ENDIANNESS   1*)

(*DBG:*)

(*DBG:char plc_program_md5[] =
"043cea30f3dc6fca8743bfe26fb790f2";*)

(*DBG:*)

(*DBG:uint8_t endianness;*)

(*DBG:*)

(*DBG:*)

(*DBG:extern MAIN RES0__INSTANCE0;*)

(*DBG:*)

(*DBG:static const struct {*)

(*DBG:    void *ptr;*)

(*FILE:c_blocks_code.cpp
typedef uint8_t   IEC_BYTE; *)

(*FILE:c_blocks_code.cpp
typedef uint16_t  IEC_WORD; *)

(*FILE:c_blocks_code.cpp
typedef uint32_t  IEC_DWORD;
*)

(*FILE:c_blocks_code.cpp
typedef uint64_t  IEC_LWORD;
*)

(*FILE:c_blocks_code.cpp  *)

(*FILE:c_blocks_code.cpp
typedef float   IEC_REAL; *)

(*FILE:c_blocks_code.cpp
typedef double  IEC_LREAL; *)

(*FILE:c_blocks_code.cpp  *)

(*FILE:c_blocks_code.cpp
#ifndef STR_MAX_LEN *)

(*FILE:c_blocks_code.cpp
#define STR_MAX_LEN 126 *)

(*FILE:c_blocks_code.cpp #endif
*)

(*FILE:c_blocks_code.cpp  *)

(*FILE:c_blocks_code.cpp
#ifndef STR_LEN_TYPE *)

(*FILE:c_blocks_code.cpp
#define STR_LEN_TYPE int8_t
*)

(*FILE:c_blocks_code.cpp #endif
*)
```

```
(*DBG:    __IEC_types_enum type;*)

(*DBG:} debug_vars[] = {*)

(*DBG:    {&(RES0__INSTANCE0.START_PB),
BOOL_P_ENUM},*)

(*DBG:      {&(RES0__INSTANCE0.STOP_PB),
BOOL_P_ENUM},*)

(*DBG:
{&(RES0__INSTANCE0.BREAKER_SWITCH),
BOOL_O_ENUM},*)

(*DBG:};*)

(*DBG:*)

(*DBG:#define VAR_COUNT              3*)

(*DBG:*)

(*DBG:uint16_t get_var_count(void)*)

(*DBG:{*)

(*DBG:    return VAR_COUNT;*)

(*DBG:}*)

(*DBG:*)

(*DBG:size_t get_var_size(size_t idx)*)

(*DBG:{*)

(*DBG:    if (idx >= VAR_COUNT)*)

(*DBG:    {*)

(*DBG:        return 0;*)

(*DBG:    }*)

(*DBG:    switch (debug_vars[idx].type) {*)
```

```
(*FILE:c_blocks_code.cpp  *)

(*FILE:c_blocks_code.cpp
typedef          STR_LEN_TYPE
__strlen_t; *)

(*FILE:c_blocks_code.cpp
typedef struct { *)

(*FILE:c_blocks_code.cpp
__strlen_t len; *)

(*FILE:c_blocks_code.cpp
uint8_t   body[STR_MAX_LEN];
*)

(*FILE:c_blocks_code.cpp        }
IEC_STRING; *)

(*FILE:c_blocks_code.cpp  *)
```

```
(*DBG:    case BOOL_O_ENUM:*)

(*DBG:    case BOOL_P_ENUM:*)

(*DBG:        return sizeof(BOOL);*)

(*DBG:    default:*)

(*DBG:        return 0;*)

(*DBG:    }*)

(*DBG:}*)

(*DBG:*)

(*DBG:void *get_var_addr(size_t idx)*)

(*DBG:{*)

(*DBG:    void *ptr = debug_vars[idx].ptr;*)

(*DBG:*)

(*DBG:    switch (debug_vars[idx].type) {*)

(*DBG:    case BOOL_O_ENUM:*)

(*DBG:    case BOOL_P_ENUM:*)

(*DBG:        return (void *)((((__IEC_BOOL_p *)
ptr)->flags & __IEC_FORCE_FLAG)*)

(*DBG:                    ? &(((__IEC_BOOL_p *)
ptr)->fvalue)*)

(*DBG:                    : ((__IEC_BOOL_p *) ptr)-
>value);*)

(*DBG:    default:*)

(*DBG:        return 0;*)

(*DBG:    }*)
```

```
(*DBG:}*)

(*DBG:*)

(*DBG:void force_var(size_t idx, bool forced,
void *val)*)

(*DBG:{*)

(*DBG:    void *ptr = debug_vars[idx].ptr;*)

(*DBG:*)

(*DBG:    if (forced) {*)

(*DBG:        size_t var_size = get_var_size(idx);*)

(*DBG:        switch (debug_vars[idx].type) {*)

(*DBG:    *)

(*DBG:        case BOOL_O_ENUM: {*)

(*DBG:                memcpy((((__IEC_BOOL_p *)
ptr)->value), val, var_size);*)

(*DBG:                memcpy(&((__IEC_BOOL_p *)
ptr)->fvalue, val, var_size);*)

(*DBG:            ((__IEC_BOOL_p *) ptr)->flags |=
__IEC_FORCE_FLAG;*)

(*DBG:            break;*)

(*DBG:        }*)

(*DBG:        case BOOL_P_ENUM: {*)

(*DBG:                memcpy(&((__IEC_BOOL_p *)
ptr)->fvalue, val, var_size);*)

(*DBG:            ((__IEC_BOOL_p *) ptr)->flags |=
__IEC_FORCE_FLAG;*)
```

```
(*DBG:          break;*)
(*DBG:      }*)
(*DBG:      default:*)
(*DBG:          break;*)
(*DBG:      }*)
(*DBG:  } else {*)
(*DBG:      switch (debug_vars[idx].type) {*)
(*DBG:      case BOOL_O_ENUM:*)
(*DBG:      case BOOL_P_ENUM:*)
(*DBG:          ((__IEC_BOOL_p *) ptr)->flags
&= ~__IEC_FORCE_FLAG;*)
(*DBG:          break;*)
(*DBG:      default:*)
(*DBG:          break;*)
(*DBG:      }*)
(*DBG:  }*)
(*DBG:}*)
(*DBG:*)
(*DBG:void swap_bytes(void *ptr, size_t size)*)
(*DBG:{*)
(*DBG:  uint8_t *bytePtr = (uint8_t *)ptr;*)
(*DBG:  size_t i;*)
(*DBG:  for (i = 0; i < size / 2; ++i)*)
```

```
(*DBG:    {*)

(*DBG:        uint8_t temp = bytePtr[i];*)

(*DBG:        bytePtr[i] = bytePtr[size - 1 - i];*)

(*DBG:        bytePtr[size - 1 - i] = temp;*)

(*DBG:    }*)

(*DBG:}*)

(*DBG:*)

(*DBG:void trace_reset(void)*)

(*DBG:{*)

(*DBG:        for  (size_t  i=0;  i  <  VAR_COUNT;
i++)*)

(*DBG:    {*)

(*DBG:        force_var(i, false, 0);*)

(*DBG:    }*)

(*DBG:}*)

(*DBG:*)

(*DBG:void set_trace(size_t idx, bool forced, void
*val)*)

(*DBG:{*)

(*DBG:    if (idx >= 0 && idx < VAR_COUNT)*)

(*DBG:    {*)

(*DBG:                    if    (endianness    ==
REVERSE_ENDIANNESS)*)

(*DBG:        {*)
```

```
(*DBG:            // Prevent swapping for STRING
type*)

(*DBG:                if (debug_vars[idx].type ==
STRING_ENUM)*)

(*DBG:          {*)

(*DBG:            // Do nothing*)

(*DBG:            ;*)

(*DBG:          }*)

(*DBG:          else*)

(*DBG:          {*)

(*DBG:                              swap_bytes(val,
get_var_size(idx));*)

(*DBG:          }*)

(*DBG:       }*)

(*DBG:*)

(*DBG:       force_var(idx, forced, val);*)

(*DBG:    }*)

(*DBG:}*)
```

Додаток Б

Лістинг програмного коду мовою програмування Python на вузлі Ubuntu:

```
from scapy.all import load_contrib, bind_layers, rdpcap, sniff, wrpcap
load_contrib('modbus')
from scapy.contrib.modbus import ModbusADURequest, ModbusADUResponse
from scapy.layers.inet import IP, TCP
import os
import time
import subprocess

# Centralized file list
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
FILES = {
    '1': os.path.join(BASE_DIR, 'modbus.pcap'),
    '2': os.path.join(BASE_DIR, 'ssh_modbus_eth0.pcap'),
    '3': os.path.join(BASE_DIR, 'modbus2.pcap'),
    '4': os.path.join(BASE_DIR, 'ssh_modbus_eth0(2).pcap'),
    '5': os.path.join(BASE_DIR, 'live_attacks_captured.pcap')
}

# Layer binding configuration
bind_layers(TCP, ModbusADURequest, dport=502)
bind_layers(TCP, ModbusADUResponse, sport=502)

time_window = 1.0
threshold = 20
packet_history = {}
blocked_ips = set()
def is_rate_limit_exceeded(ip):
    current_time = time.time()
    if ip not in packet_history:
        packet_history[ip] = []
    packet_history[ip].append(current_time)
    packet_history[ip] = [t for t in packet_history[ip] if current_time - t < time_window]
    return len(packet_history[ip]) > threshold

def write_log(message):
    with open("alerts.log", "a") as log:
        log.write(f"[{time.ctime()}] {message}\n")

def block_attacker_ip(ip):
    try:
        subprocess.run(f'sudo iptables -I INPUT -s {ip} -j DROP', shell=True, check=True)
```

```
        subprocess.run(f"sudo iptables -I FORWARD -s {ip} -j
DROP",shell=True,check=True)
        print(f'[***]IPS ACTION: Successfully blocked IP {ip} via iptables.')
    except subprocess.CalledProcessError as e:
        print(f'[-]Failed to block IP: {e}')

def classify_source(src_ip):
    if src_ip == "192.41.15.2":
        return "SCADA"
    elif src_ip == "192.41.15.3":
        return "Kali-Attacker"
    elif src_ip == "192.41.15.1":
        return "PLC"
    else:
        return "Unknown"

def get_file_choice():
    """Universal file selection function for all modules"""
    print('\n--- Available PCAP Files ---')
    for key, path in FILES.items():
        print(f"{key} - {path}")

    choice = input('Choose a file number (or "q" to cancel): ').strip()
    return FILES.get(choice)

def analyze_pcap():
    """Module for detailed manual packet analysis"""
    file_path = get_file_choice()
    if not file_path: return

    try:
        print(f"[*] Loading {file_path}...")
        d = rdpcap(file_path)
        print(f'[+] Success. Number of packets: {len(d)}')

        while True:
            cmd = input('\nEnter a row number (or "q" to back to menu): ').strip()
            if cmd.lower() == 'q': break

            try:
                a = int(cmd)
                if 0 <= a < len(d):
                    print('\n1 - Structured (Summary)\n2 - Hierarchical (Detailed)')
                    mode = input('Your choice: ').strip()
```

```
                print("-" * 50)
                if mode == '1':
                    print(d[a].summary())
                else:
                    d[a].show()
                print("-" * 50)
            else:
                print(f'[!] Error: Packet #{a} not found. Max index: {len(d)-1}')
        except ValueError:
            print("[!] Please enter a numeric value!")
    except Exception as e:
        print(f"[!] Error reading file: {e}")

def detect_attacks():
    """Module for automated attack detection (IDS)"""
    file_path = get_file_choice()
    if not file_path: return

    try:
        print(f"[*] Analyzing {file_path} for attacks...")
        packets = rdpcap(file_path)
        suspicious_count = 0

        for pkt in packets:
            # Modbus check
            if pkt.haslayer("ModbusADURequest") or pkt.haslayer("ModbusPDU"):
                layer = pkt.getlayer("ModbusADURequest") or pkt.getlayer("ModbusPDU")

                if hasattr(layer, 'funcCode'):
                    f_code = layer.funcCode
                    # Detecting critical function codes (Write/Control)
                    if f_code in [1, 5, 6, 15, 16]:
                        print(f"[!] Write command detected! Function Code: {f_code}")
                        suspicious_count += 1

                # IP spoofing/Suspicious source check
                if pkt.haslayer(IP):
                    src_ip = pkt[IP].src
                    source = classify_source(src_ip)
                    if src_ip not in ['192.41.15.2', '192.41.15.1', '127.0.0.1']:
                        print(f'[!] WARNING! Suspicious IP detected: {source} {src_ip}')

        print(f'\n[+] Analysis complete. Total incidents detected: {suspicious_count}')
    except Exception as e:
        print(f"[!] Analysis error: {e}")
```

```
def live_capture_ids():
    print("\n[*] Starting LIVE capture mode...")
    print("[*] Listening for SSH & Modbus (30 sec capture)...")
    captured_data = []
    def process_packet(pkt):
        if not pkt.haslayer(IP):
            return
        src_ip = pkt[IP].src
        source = classify_source(src_ip)
        is_bad = False
        if src_ip not in ['192.41.15.2', '192.41.15.1', '127.0.0.1']:
            print(f"[!] LIVE ALERT: Unauthorized IP {source} ({src_ip})!")
            write_log(f"[!] LIVE ALERT: Unauthorized IP {source}({src_ip})!")
            is_bad = True
        if src_ip != '192.41.15.1' and is_rate_limit_exceeded(src_ip):
            print(f"[!!!] DDOS ALERT: from {source}({src_ip}) sends too many packets!")
            write_log(f"[!!!] DDOS ALERT: from {source}({src_ip}) sends too many
packets!")
            if src_ip not in blocked_ips:
                block_attacker_ip(src_ip)
                blocked_ips.add(src_ip)
            is_bad = True
        if pkt.haslayer("ModbusADURequest") or pkt.haslayer("ModbusPDU"):
            layer = pkt.getlayer("ModbusADURequest") or pkt.getlayer("ModbusPDU")
            if hasattr(layer, 'funcCode') and layer.funcCode in [1, 5, 6, 15, 16]:
                print(f"[!] LIVE ALERT: Critical Modbus Function {layer.funcCode} detected
from {source}({src_ip})!")
                write_log(f"[!] LIVE ALERT: Critical Modbus Function {layer.funcCode}
detected from {source}({src_ip})!")
                is_bad = True

        if is_bad:
            captured_data.append(pkt)

    try:
        sniff(
            iface="enp0s3",
            prn=process_packet,
            filter="tcp port 22 or tcp port 502",
            timeout=30
        )
    except KeyboardInterrupt:
        print("\n[*] Capture stopped by user.")
```

```
    print(f"\n[*] Captured suspicious packets: {len(captured_data)}")

    if captured_data:
        output_file = "live_attacks_captured.pcap"
        wrpcap(output_file, captured_data)
        print(f"[+] Saved {len(captured_data)} packets to {output_file}")
    else:
        print("[!] No suspicious packets captured.")

while True:
    print('\n========== MAIN MENU ==========')
    print('1. Detailed PCAP Analysis')
    print('2. Automated Attack Detection')
    print('3. LIVE Capture & IDS Mode (Sniffing)')
    print('4. Exit program')
    print('==============================')

    choice = input('Make your choice: ').strip()

    if choice == '1':
        analyze_pcap()
    elif choice == '2':
        detect_attacks()
    elif choice == '3':
        live_capture_ids()
    elif choice == '4':
        print('Exiting... Good bye!')
        break
    else:
        print('[!] Invalid choice. Please try again.')
```

Додаток В

Лістинг програмного коду мовою програмування Python на вузлі Kali Linux:

```
 from netmiko import ConnectHandler
from pymodbus.client import ModbusTcpClient
import time

class NetCon():
    def __init__(self, host, user):
        self.host = host
        self.user = user
        self.port = 22
        self.device_type = 'linux'

    def get_connection(self):
        try:
            cred = {
                "device_type": self.device_type,
                "host": self.host,
                "username": self.user,
                "use_keys": True,
                "key_file": "/home/kali/.ssh/id_ed25519",
                "port": self.port,
            }
            print(f"[*] Attempting login using SSH Ed25519 key on {self.host}...")
            connection = ConnectHandler(**cred)
            return connection

        except Exception as e:
            print(f"Error 101! Connection failed: {e}")
            return None

def connection_to_plc():
    plc = NetCon(host='192.41.15.1', user='student')
    ssh_session = plc.get_connection()
    if ssh_session:
        print("[+] Success! Connection established.")
        while True:
            try:
                b = int(input('1 - Start\n2 - Breaker Switch\n3 - Stop\nEnter a command: '))
                if b == 1:
                    result = ssh_session.send_command('modpoll -r 3 -t 0 127.0.0.1 1')
                    time.sleep(0.1)
                    result = ssh_session.send_command('modpoll -r 3 -t 0 127.0.0.1 0')
                    break
                elif b == 2:
```

```
                result = ssh_session.send_command('modpoll -r 1 -t 0 127.0.0.1 1')
                break
            elif b == 3:
                result = ssh_session.send_command('modpoll -r 4 -t 0 127.0.0.1 1')
                time.sleep(0.1)
                result = ssh_session.send_command('modpoll -r 4 -t 0 127.0.0.1 0')
                break
            else:
                print('Invalid option! Please enter 1, 2, or 3.')
        except ValueError:
            print('Error! Please enter a correct number (1, 2, or 3).')

    print("Response from OpenPLC:")
    print("-" * 30)
    print(result)
    print("-" * 30)
    ssh_session.disconnect()
    print("Connection closed.")
  else:
    print("[-] Connection failed. Please check the IP address, credentials, or SSH service
status.")

def attacks():
  plc = '192.41.15.1'
  client = ModbusTcpClient(plc, port=502)
  if not client.connect():
    print("[-] Connection to PLC failed.")
    return
  print("[+] Connected to PLC via Modbus TCP.")
  n = int(input('Enter the number of command sequences for the attack: '))
  for i in range(n):
    client.write_coil(1, True)
    client.write_register(1, 123)
    client.write_coils(1, [True, False, True])
    client.write_registers(1, [111, 222, 333])

    time.sleep(0.2)
  print(f'Attack completed. A total of commands were sent:{n*4}')

while True:
  print('\n========== MAIN MENU ==========')
  print('1. Attacks')
  print('2. Manage OpenPLC (SSH)')
  print('3. Exit program')
  print('==============================')
```

```
try:
    choice = int(input('Make your choice: '))
    if choice == 1:
        attacks()
    elif choice == 2:
        connection_to_plc()
    elif choice == 3:
        print('Exiting... Good bye!')
        break
    else:
        print('[!] Invalid choice. Please select 1, 2, or 3.')
except ValueError:
    print('[!] Error: Input must be a number.')
```