Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

"Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Network (DQN)"

Виконав: ст. 2-го курсу, групи КІТС-23м
спеціальності 125– Кібербезпека
Освітня програма – Кібербезпека
інформаційних технологій та систем

(шифр і назва напряму підготовки, спеціальності)

Савшак А.С.
(прізвище та ініціали)

Керівник: к.т.н., доц., доцент каф. МБІС
Грицак А.В.
(прізвище та ініціали)
«____» _____ 2024 р.

Опонент: к.т.н., доц., доцент каф. ОТ
Тарновський М.Г.
(прізвище та ініціали)
«____» _____ 2024 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"09" грудня 2024 р.

Вінниця ВНТУ - 2024 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____ **Юрій ЯРЕМЧУК**
"20" вересня 2024 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту

Савшаку А.С.
(прізвище, ім`я, по-батькові)

1. Тема роботи : " Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Network (DQN)"

Керівник роботи
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "17" вересня 2024 року № 310

2. Строк подання студентом роботи

3. Вихідні дані до роботи: Електронні джерела, наукові статті, книги та документи, пов'язані з темою. Існуюче програмне забезпечення для аналізу мережевого трафіку, інструменти для машинного навчання та глибокого виявлення аномалій, технічна документація, вимоги та обмеження до програмного забезпечення.

4. Зміст текстової частини: Для досягнення мети роботи були поставлені наступні задачі: дослідити актуальність загроз для IoT мереж та вразливостей; проаналізувати існуючі методи виявлення аномалій у мережевому трафіку IoT, зокрема алгоритми машинного навчання; дослідити можливості використання нейронних мереж і глибокого навчання для виявлення аномалій у мережах IoT; розробити вдосконалений алгоритм для виявлення аномалій з адаптивним порогом і застосуванням Deep Q-Network (DQN); реалізувати програмний засіб для аналізу мережевого трафіку IoT з використанням Python та бібліотек глибокого навчання; здійснити тестування та оцінити ефективність системи на реальних даних.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)

Зображення поетапної реалізації та налаштування веб-додатку

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	к.т.н., доц., кафедри МБІС Грицак А.В.	20.09.2024	20.09.2024
Економічна частина	проф.кафедри ЕПВМ, д.е.н Бурєннікова Н.В.		

7.Дата видачі завдання 20 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
		початок	закінчення	
1	Аналіз предметної області IoT мереж та їх вразливостей	20.09.2024	23.09.2024	Виконано
2	Огляд існуючих методів захисту IoT мереж та виявлення аномалій в мережевому трафіку	24.09.2024	28.09.2024	Виконано
3	Постановка задачі та визначення мети роботи	29.09.2024	02.10.2024	Виконано
4	Дослідження алгоритмів машинного навчання для виявлення аномалій в IoT мережах	02.10.2024	08.10.2024	Виконано
5	Розробка алгоритму для глибокого виявлення аномалій в мережевому трафіку IoT	08.10.2024	16.10.2024	Виконано
6	Вибір і адаптація алгоритму навчання з підкріпленням (Deep Q-Network, DQN) для IoT мереж	18.10.2024	29.10.2024	Виконано
7	Розробка вдосконаленого алгоритму виявлення аномалій з адаптивним порогом	30.10.2024	05.11.2024	Виконано
8	Практична реалізація системи для виявлення аномалій в IoT мережах на основі DQN	06.11.2024	13.11.2024	Виконано
9	Тестування та оцінка ефективності розробленої системи для виявлення аномалій в мережевому трафіку IoT	14.11.2024	20.11.2024	Виконано
10	Оформлення матеріалів до захисту	21.11.2024	21.11.2024	Виконано

Студент _____ _____
(підпис)

АНОТАЦІЯ

УДК 621.374.415

Савшак А.С. Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2024. 105 с.

На укр. мові. Бібліогр.: 50 назв; рис.: 27; табл. 6.

У магістерській кваліфікаційній роботі розроблено програму для підвищення захисту IoT мережі на основі аналізу мережевого трафіку і мащинного навчання з використання вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net. У загальній частині роботи розглянуто особливості IoT мереж, проблеми безпеки IoT мереж, вимоги безпеки IoT, розглянуто методологічні аспекти проблеми інформаційної безпеки IoT мереж, приведена класифікація загроз, атак, уразливостей, які можуть бути направлені на компрометацію IoT мережі, наведені основні кроки розробки алгоритму, показана UML-діаграма. побудови сучасних радіоприймачів, а також обґрунтована доцільність їх розробки. У технологічній частині розроблений програмний засіб для підвищення захищеності IoT мереж.

В економічному розділі обґрунтовано доцільність створення даного програмного засобу.

Ключові слова: IoT мережі, аналіз мережевого трафіку, алгоритм виявлення аномалій мережевого трафіку, машинне навчання, Deep Q-Net, Python.

ABSTRACT

Savshak A.S. Improving the security of IoT networks based on network traffic analysis and machine learning using an advanced deep anomaly detection algorithm using adaptive detection threshold and a deep Q-Net reinforcement learning algorithm. Bachelor's thesis in specialty 125 - Cybersecurity, educational program-cybersecurity of information technologies and systems. Vinnitsa: VNTU, 2020. – 105 p.

In Ukrainian language. Bibliographer: 50 titles; fig.: 27; tabl. 6.

In the master's qualification work, a program was developed to improve the protection of the IoT network based on network traffic analysis and mast training using an advanced deep anomaly detection algorithm using adaptive detection threshold and a deep Q-Net reinforcement learning algorithm. In the general part of the work, the features of IoT networks, security problems of IoT networks, IoT security requirements are considered, methodological aspects of the problem of information security of IoT networks are considered, the classification of threats, attacks, vulnerabilities that can be aimed at compromising the IoT network is given, the main steps of algorithm development are given, and the UML diagram is shown. construction of modern radio receivers, as well as the expediency of their development is justified. In the technological part, a software tool has been developed to improve the security of IoT networks.

In the economic section, the expediency of creating this software tool is justified.

Keywords: IoT networks, network traffic analysis, network traffic anomaly detection algorithm, machine learning, Deep QNet, Python.

Зміст

Вступ

Актуальність. У сучасному взаємопов’язаному світі Інтернет речей (IoT) змінив спосіб життя та роботи суспільства. Саме тому, виникає критична потреба в надійній безпеці мережі в пристроях IoT.

У взаємопов’язаному ландшафті Інтернету речей (IoT) забезпечення надійної безпеки мережі має першочергове значення для захисту конфіденційних даних і підтримки цілісності систем. Розповсюдження пристроїв Інтернету речей відкрило безліч точок входу для кібератак, що змусило організації впровадити комплексні заходи безпеки.

Від розумної побутової техніки до промислових систем керування, кожен пристрій, підключений до Інтернету, може бути використаний зловмисниками. Ця вразливість підкреслює необхідність постійного моніторингу, регулярних оновлень програмного забезпечення та розгортання протоколів шифрування для захисту даних під час передачі та зберігання.

Однією з ключових проблем безпеки Інтернету речей є складність керування різними пристроями з різними протоколами безпеки. Це розмаїття не тільки ускладнює впровадження стандартизованих заходів безпеки, але й створює потенційну вразливість, якщо безпеку мережі в IoT будь-якого пристрою буде скомпрометовано.

Різні методи захисту систем IoT мають вирішальне значення для захисту пристроїв і мереж IoT від кіберзагроз, охоплюючи надійні протоколи шифрування, суворі заходи безпеки мережі та проактивну оцінку вразливості.

Підвищення безпеки мережі передбачає впровадження брандмауерів, систем виявлення/запобігання вторгненням і безпечних каналів зв’язку, таких як VPN (віртуальна приватна мережа), для захисту від зловмисних атак, спрямованих на інфраструктуру IoT.

Впровадження додаткових заходів безпеки для IoT виходить за рамки традиційних підходів, вимагаючи інтеграції з операційною технологією, застарілими протоколами та бездоганного узгодження з корпоративними операціями.

Зі швидким розширенням ландшафту Інтернету речей (IoT) складність захисту цих взаємопов’язаних пристроїв і систем різко зросла.

Розширені заходи безпеки необхідні для захисту конфіденційних даних і забезпечення цілісності роботи в різних мережах.

Ці заходи поширюються на інтеграцію з операційною технологією для захисту критично важливої інфраструктури, забезпечення безпеки застарілих протоколів для пом’якшення вразливостей і узгодження стратегій безпеки з основними операціями підприємства для узгодженого управління ризиками.

Саме тому, впровадження нових методів підвищення захищеності IoT мереж є одним із пріоритетних напрямків захисту інформації.

Метою кваліфікаційної роботи є розробка заходів підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net.

Об’єктом кваліфікаційної роботи є засоби підвищення захисту IoT мереж на сучасному етапі розвитку.

Предметом кваліфікаційної роботи є впровадження методів підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net.

Для досягнення поставленої мети потрібно виконати наступні **завдання**:

- Розглянути поняття IoT мережі та проблем безпеки, пов’язаних з даним видом мереж;
- Виділити основні вимоги безпеки, які висуваються IoT мережам;
- Провести огляд методологічних аспектів проблем інформаційної безпеки IoT мереж;
- Навести класифікацію загроз, атак, розголошень, уразливостей, які можуть бути застосовані до IoT мереж;
- Провести розробку алгоритму підвищення захищеності IoT мереж;

– Провести тестування розробленого алгоритму та навести результати його тестування;
– Оцінити економічну доцільність реалізації даного проекту.

Наукова новизна кваліфікаційної роботи полягає в поглиблені вже існуючих методів захисту IoT мереж, з використанням методів машинного навчання та алгоритмів глибокого навчання, що дає можливість більш ефективно виявляти та нейтралізувати загрози, які виникають при роботі з IoT мережами.

Розділ 1. Теоретичні відомості про IoT мережі

1.1 Поняття IoT мережі

Інтернет речей, або IoT, – це мережа взаємопов'язаних пристроїв, які з'єднуються та обмінюються даними з іншими пристроями IoT та хмарою. Пристрої IoT зазвичай вбудовані в такі технології, як датчики та програмне забезпечення, і можуть включати механічні та цифрові машини та предмети споживання.

Ці пристрої охоплюють все: від повсякденних предметів домашнього вжитку до складних промислових інструментів. Все частіше організації в різних галузях використовують IoT для більш ефективної роботи, забезпечення покращеного обслуговування клієнтів, покращення прийняття рішень і підвищення вартості бізнесу [1].

Завдяки IoT дані можна передавати через мережу, не вимагаючи взаємодії людини з людиною або людиною з комп'ютером.

Річчю в Інтернеті речей може бути людина з імплантатом кардіомонітора, сільськогосподарська тварина з біочіпом-транспондером, автомобіль, який має вбудовані датчики для попередження водія про низький тиск у шинах, або будь-який інший природний або штучний об'єкт, якому можна призначити адресу інтернет-протоколу і який може передавати дані через мережу.

Системи IoT функціонують шляхом збору даних із датчиків, вбудованих у пристрої IoT, які потім передаються через шлюз IoT для аналізу програмою або внутрішньою системою.

Наступні чотири елементи включені в екосистему IoT для її функціонування [1]:

Екосистема IoT складається з розумних пристроїв з веб-доступом, які використовують вбудовані системи, такі як процесори, датчики та комунікаційне обладнання, для збору, надсилання та дій на основі даних, отриманих із їхніх середовищ.

Пристрої IoT можуть спілкуватися один з одним через мережу через Інтернет. Ці пристрої обмінюються даними датчиків, підключаючись до шлюзу

IoT, який діє як центральний концентратор, куди пристрої IoT можуть надсилати дані. Перш ніж поділитися даними, їх також можна надіслати на периферійний пристрій, де вони аналізуються локально [2].

Лише відповідні дані використовуються для виявлення закономірностей, надання рекомендацій і виявлення потенційних проблем до їх ескалації. Аналіз даних локально зменшує обсяг даних, що відправляються в хмару, що мінімізує споживання пропускної здатності [3].

Іноді ці пристрої зв'язуються з іншими пов'язаними пристроями та діють на основі інформації, яку вони отримують один від одного. Більшу частину роботи пристрої виконують без участі людини, хоча з пристроями можуть взаємодіяти люди. Наприклад, вони можуть налаштувати їх, дати їм інструкції або отримати доступ до даних. Протоколи підключення, мережі та зв'язку, які використовуються з цими веб-пристроями, значною мірою залежать від конкретних розгорнутих додатків IoT.

IoT також може використовувати штучний інтелект і машинне навчання, щоб зробити процеси збору даних простішими та динамічнішими.

Графічний інтерфейс користувача (UI) зазвичай використовується для керування пристроями IoT. Наприклад, веб-сайт або мобільний додаток можна використовувати як інтерфейс користувача для керування, контролю та реєстрації розумних пристроїв [3].

Інтернет речей допомагає людям жити та працювати розумніше. Споживачі, наприклад, можуть використовувати вбудовані в Інтернет речей пристрої, такі як автомобілі, розумні годинники або термостати, для поліпшення свого життя.

Окрім пропозиції розумних пристроїв для автоматизації будинків, IoT має важливе значення для бізнесу. Він надає організаціям можливість у режимі реального часу спостерігати за тим, як працюють їхні системи, надаючи інформацію про все, від продуктивності машин до ланцюжка поставок і логістичних операцій.

Інтернет речей дозволяє машинам виконувати виснажливі завдання без втручання людини. Компанії можуть автоматизувати процеси, скоротити

витрати на оплату праці, скоротити відходи та покращити надання послуг. Інтернет речей допомагає зробити виробництво та доставку товарів дешевшим і забезпечує прозорість транзакцій клієнтів.

Інтернет речей продовжує розвиватися, оскільки все більше компаній усвідомлюють потенціал підключених пристроїв для підтримки їх конкурентоспроможності.

Інтернет речей пропонує організаціям кілька переваг. Це заохочує компанії переосмислити свій підхід до бізнесу та надає їм інструменти для покращення бізнес-стратегій [4].

Деякі переваги IoT залежать від галузі, тоді як інші застосовні в кількох галузях. Як правило, промисловий Інтернет речей (IoT) найбільш поширений у виробничих, транспортних та комунальних організаціях, які використовують датчики та інші пристрої IoT; Однак він також має варіанти використання для організацій у галузях сільського господарства, інфраструктури та домашньої автоматизації, що веде деякі організації до цифрової трансформації.

Поширені приклади додатків IoT включають наступне:

Сільське господарство. Інтернет речей може принести користь фермерам, полегшуючи їхню роботу. Наприклад, датчики можуть збирати дані про кількість опадів, вологість, температуру та вміст ґрунту, а IoT може допомогти автоматизувати методи ведення сільського господарства. Крім того, пристрої IoT можна використовувати для нагляду за здоров'ям худоби, моніторингу обладнання та оптимізації управління ланцюжком поставок [5].

Будівництво. Інтернет речей може допомогти контролювати операції навколо інфраструктури. Датчики, наприклад, можуть відстежувати події або зміни в структурних будівлях, мостах та іншій інфраструктурі, які потенційно можуть поставити під загрозу безпеку. Це забезпечує такі переваги, як покращення управління інцидентами та реагування на них, зниження операційних витрат та покращення якості обслуговування [6].

Домашня автоматизація. Компанія, що займається домашньою автоматизацією, може використовувати IoT для моніторингу та маніпулювання механічними та електричними системами в будівлі. Домовласники також

можуть дистанційно керувати та автоматизувати своє домашнє середовище за допомогою пристроїв IoT, включаючи розумні термостати, системи освітлення, камери безпеки та голосові помічники, такі як Alexa та Siri, для підвищення комфорту та енергоефективності [7].

Розумні будівлі та міста. Розумні міста можуть допомогти громадянам зменшити кількість відходів та споживання енергії. Вони можуть зменшити витрати на електроенергію за допомогою датчиків, які визначають, скільки людей знаходиться в кімнаті, і включення кондиціонера, якщо датчики виявляють, що конференц-зал заповнений, або зниження температури, якщо всі в офісі розійшлися по домівках [8].

Міські системи споживання. Технології IoT також можуть використовуватися для моніторингу та управління міським споживанням, такими як світлофори, паркомати, системи управління відходами та мережі громадського транспорту [9].

Моніторинг охорони здоров'я. Пристрої IoT, такі як системи віддаленого моніторингу пацієнтів, розумні медичні пристрої та трекери ліків, дозволяють медичним працівникам контролювати стан здоров'я пацієнтів, керувати хронічними захворюваннями та надавати своєчасні втручання. IoT дає постачальникам можливість більш уважно стежити за пацієнтами, аналізуючи згенеровані дані. Лікарні також часто використовують системи IoT для виконання таких завдань, як управління запасами як фармацевтичних препаратів, так і медичних інструментів [10].

Роздрібні. Датчики та маяки IoT у роздрібних магазинах можуть відстежувати рух клієнтів, аналізувати моделі покупок, керувати рівнем запасів та персоналізувати маркетингові повідомлення. Це покращує досвід покупок для клієнтів та оптимізує роботу магазину [10].

Транспорт. Пристрої IoT допомагають транспортній галузі, відстежуючи продуктивність транспортних засобів, оптимізуючи маршрути та відстежуючи відправлення. Наприклад, можна контролювати паливну ефективність підключених автомобілів, щоб зменшити витрати на паливо та підвищити

екологічність. Пристрої IoT також можуть відстежувати стан вантажу, щоб він дістався до місця призначення в оптимальному стані [11].

Пристрої, що носяться. Носимі пристрої з датчиками та програмним забезпеченням можуть збирати та аналізувати дані користувачів, надсилаючи повідомлення іншим технологіям про користувачів, щоб зробити їхнє життя простішим та комфортнішим. Носимі пристрої також використовуються для громадської безпеки – наприклад, шляхом збільшення часу реагування служб швидкого реагування під час надзвичайних ситуацій шляхом забезпечення оптимізованих маршрутів до місця або відстеження життєвих показників будівельників чи пожежників на небезпечних для життя об'єктах [11].

Управління енергією. Інтелектуальні мережі з підтримкою Інтернету речей, інтелектуальні лічильники та системи управління енергією дозволяють комунальним підприємствам та споживачам контролювати та оптимізувати використання енергії, керувати програмами реагування на попит та ефективніше інтегрувати відновлювані джерела енергії. Наприклад, дані, зібрані пристроями та датчиками IoT, допомагають виявляти закономірності, час пікового використання та області неефективності [12].

1.2 Фундаментальні проблеми безпеки IoT

В загальному розумінні, IoT являє собою багатогранну систему, яка включає в себе велику кількість різноманітних послуг та додатків, а в той же час, вона вимоглива до наявності чітко сформульованого наповнення. В даному випадку, мається на увазі, наявність загального протоколу, максимально наповнену методологію, архітектуру, якісно сформовані стандарти, а також, наявності механізмів безпеки, які дадуть можливість поєднати віртуальний і реальний світ в одній платформі. Глобально, виділяють всього три групи проблем, повз’язаних з безпекою IoT, а саме, архітектура, стандартизація, безпека та конфіденційність [13].

Проблема архітектури полягає в тому, що кількість смарт-пристрої нестримно росте. Збільшення кількості пристроїв вимагає генерування великої кількості дани, щоб забезпечити надання великої кількості послуг, в будь-якому

куточку світу та в будь-який час. Надання представленої кількості послуг потребує наявності інфраструктури, в пергу чергу дя того, щоб проводити такі дії як інтеграція та аналіз згенерованих даних. Саме тому, IoT мережа потребує наявності гнучкої архітектури, щоб охарактеризувати цілісну картину того, яким чином здійснюється підтримка пристроїв та додатків.

Що стосується архітектурної проблеми, кількість різноманітних смарт-об'єктів з кожним днем збільшується. Кожен розумний пристрій генерує дані для надання багатьох послуг у будь-який часі в будь-якому місці. Ці послуги вимагають певної інфраструктури для підтримки, інтеграції та аналізу згенерованих даних для того, щоб передбачити майбутнє рішення. Тому Інтернету речей потрібна динамічна архітектура, щоб представити цілісний план, який використовується для підтримки різних об'єктів і додатків [14].

Варто зазначити, IoT обладнання великою кількістю датчиків, потужість яких досить сильно обмежена. Відповідно, застосування традиційних заходів безпеки не зможе забезпечити зв'язок IoT між різними пристроями. Саме тому, вирішення проблеми полягає в розробці таких систем безпеки, які могли б забезпечувати низьку обчислювальну потужність і простоту в користуванні. Отже, провідну роль в розробках IoT мереж відіграє забезпечення безпеки даних.

В будь-якому випадку, в основі IoT лежить передача та обмін інформацією або даними між різними пристроями. Дані, які передаються між пристроями, як правило, складаються з особистої інформації користувачі, знаючи яку, зловмисники можуть ідентифікувати користувача, та скористатись його особистими даними. Особисті дані користувачів не проходячи засоби аутентифікації постійно підпадають під дію різних атак та загроз. Тому, забезпечення конфіденційності і захисту персональної інформації також є важливим чинником IoT мереж.

Основу конфіденційності становлять три базові принципи, а саме, секретність, анонімність, автономність. Секретність розуміється як шифрування повідомлень, зміст яких зрозуміли кінцевому отримувачу. Анонімність розуміється як можливість не розкривати особистість відправників та

одержувачі, при передачі повідомлень. Автономність передбачає нівелювання зустрічей із зловмисниками [15].

Управління довірою є ще одною важливою проблемою, яка повз'язана з безпекою IoT. Управління довірою є одним із базових принципів взаємодії між розумними об'єктами в плані обміну інформацією та її використанням.

Через обмеженість протоколів, ресурсів і потужності різних розумних об'єктів існує надзвичайно велика проблема управління довірою IoT. Управління довірою є важливою частиною безпеки Інтернету речей, інформаційної безпеки, послуг, програм і конфіденційності користувачів. Управління довірою є важливим елементом взаємодії між розумними об'єктами для обміну даними та керування ними.

Всередині IoT присутня велика кількість об'єктів, яка генерує різні об'єми даних. Кожен об'єкт має проходити процес аутентифікації і бути спорядженим різними механізмами безпеки, які зможуть надати захист користувачам. Варто зазначити, що роль механізмів безпеки полягає у здійснені запобігання загрозам і атакам на особисті дані користувачів. Наукова термінологія називає даний механізм наскрізною безпекою. До складу домену наскрізної безпеки входять: пристрої IoT, шлюз IoT, доступ і мережеве підключення, додаток IoT, платформа та користувачі. Базовими принципами роботи наскрізної безпеки є аутентифікація, контроль доступу та процеси шифрування [16].

Аутентифікація може бути запроваджена з допомогою різних методів, таких як ідентифікатор, пароль та інфраструктура відкритих ключів (PKI). Варто зазначити, що усталені методи аутентифікації не є ефективними для IoT, причиною є неоднорідність і складність об'єктів. Керування ідентифікацією потрібно для встановлення з'єднань смарт - об'єктів. Відповідно, ідентифікація та аутентифікая досить сильно пов'язані між собою.

Відповідно засоби авторизації та контролю доступу дають можливість користувачам здійснювати вхід до своїх ресурсів та користуватись наданими послугами. Проведення аутентифікації та контролю доступу майже повністю нівелюють здійснення несанкціонованого доступу до ресурсів користувачів.

Механізми авторизації та контролю доступу надають користувачам доступ до мережевих ресурсів і послуг. Аутентифікація та контроль доступу запобігають доступу неавторизованих користувачів до ресурсів мережі [28].

Основним призначенням системи управління є акцентуація методичного забезпечення безпеки рівнів IoT. Дана система потрібна для реагування на атаки та загрози в мережі. Серед великої кількості механізмів безпеки виділяють безпеку протоколу Інтернету (IPsec), асиметричну та симметричну криптографію, аутентифікацію тощо.

На глобальному рівні, варто зазначити, що для забезпечення якісного захисту різних рівнів даних не існує чітко розробленої концепції, яка б могла це гарантувати.

1.3 Вимоги безпеки IoT мереж

Для підвищення ефективності використання IoT потрібно дотримуватись базових вимог системи безпеки. До таких вимог відносять доступність, підзвітність, аудит, аутентифікація та авторизація, контроль доступу, приватність, конфіденційність, цілісність, безвідмовність.

В основі визначення доступномі лежить забезпечення неперервного надання до всіх видів послуг незалежно від часу та місцезнаходження користувача. Наявність комунікації між користувачем та ресурсом є важливою запорукою дотримання принципу доступності [17].

Підзвітність не приймає участі у запобіганні атак та загроз в IoT, але вона нерозривно пов’язана з цілісністю та конфіденційністю. Підзвітність потрібна для відстеження пристроїв, які приймають та відправляють дані, для відстеження підозрілої активності, тим самим надаючи правила пристроям та користувача.

Аудит є важливим принципом вимог безпеки, який дає можливість знаходити уразливості в системі безпеки IoT. В основі аудиту покладена система оцінок, призначенням якої є дослідження відповідності IoT стандартам створення додатків [18].

Аутентифікація відіграє провідну роль у формуванні системи безпеки, так як вона допомогає ідентифікувати користувача, використовуючи крипто логічні алгоритми. Авторизація виступає в ролі дозволу на взаємодію між користувачем та мережевим ресурсом.

Контроль доступу повністю підпорядкований адміністратору мережі, для надання користувачам методів аутентифікованого доступу до мережевих ресурсів.

Приватність виступає захистом особистих даних користувачів від несанкціонованого доступу.

Конфіденційність є мірою запобігання появі неавторизованих користувачів. Конфіденційні включає в себе ідентифікацію, аутентифікацію та авторизацію для учасників мережі IoT [19].

Цілісність дає можливість авторизованим користувачам робити маніпуляції з особистими даними, але при цьому маючи ряд певних обмежень. Цілісність є засобом запобігання внутрішніх атак, які є більш небезпечними ніж зовнішні.

Безвідмовність. Даний принцип полягає в тому, щоб довести, що адресат та адресант не можуть заперечити, що відправлені та отримані ними повідомлення належать їм.

1.4 Висновки

Інтернет речей (IoT) трансформує життя, забезпечуючи автоматизацію та нові можливості, але стикається з викликами безпеки, конфіденційності та інтеграції. Для захисту IoT-мереж потрібні нові підходи, зокрема аналіз трафіку та машинне навчання, адаптовані до ресурсних обмежень пристроїв.

Розділ 2. Методологічні аспекти проблеми інформаційної безпеки IoT мереж

2.1 Основна інформація

За час дослідження проблеми виникнення загроз середовища IoT було сформовано велику кількість класифікацій. Автори дослідження пропонують свої персоналізовані моделі, які відображають всі складові безпеки окремо взятої технології. Складові безпеки за однією із класифікацій поділяються на ідентифікацію, агрегацію даних, прийняття рішень, об'єднання елементів у єдину систему.

Ідентифікацію пов'язують з інтеграцією RFID-датчиків та зчитувачів, які повинні чітко ідентифікувати пристрій у мережі, а також вказати його місцезнаходження [20].

Агрегація даних передбачає збір великих масивів даних з подальшою передачею їх у вказаному форматі, такому я JSON або XML, на веб-сервер або інший пристрій.

Прийняття рішень являє собою процес фільтрації отриманого обсягу даних, виділяючи найбільш інформативні. В залежності від виду взаємодії, можливі наступні варіанти. При взаємодії «машина-машина» на основі зібраної інформації реалізується прийняте рішення, а при взаємодії «людина-машина», користувач отримує вибір дій.

Відповідно до вищезазначеного, архітектура безпеки умовно ділиться на три виміри, а саме, служба безпеки, мережевий рівень та домен безпеки. В основу першого виміру входить аутентифікація, управління доступом, конфіденційність, цілісність, доступність та ін. Третій вимір складається з домену виконавчих та сенсорних пристроїв, домену доступу, мережевого домену, та внутрішнього домену [21].

Альтернативна модель безпеки включає в себе рівень виконавчих пристроїв, транспортний рівень та прикладний рівень. Структура моделі представлена у таблиці 2.1

Таблиця 2.1 – Складові безпеки середовища Інтернету речей

Рівень виконавчих пристроїв	Безпека пристроїв	RFID-пристрої, бездротові сенсорні пристрої, GPS-пристрої
	Безпека мережі пристроїв	
Транспортний рівень	Доступ до мережі	WiFi-мережі, Ad hoc- мережі
	Мережа Інтернет	Мобільний Інтернет, Інтернет
	Локальна мережа	Безпека локальної мережі
Прикладний рівень	Додатки Інтернету речей	Інформаційна логістика, інтелектуальна мережева безпека, моніторинг середовища
	Підтримка додатків	Безпека середовища розробки, платформи хмарних обчислень, проміжні технології безпеки

Отже, можна констатувати повну відсутність єдиного підходу до формалізації безпеки середовищ IoT мереж. Всі представлені концепції або стосуються проблеми поверхнево, або концентруються на окремих технологіях

2.2 Класифікація негативних чинників, які можуть впливати на роботу IoT

Одним із найважливіших чинників формування безпеки IoT є ідентифікація різних загроз, уразливостей, впливів та атак. У випадку загрози, то вони вводяться в дію на основі знання про слабкості, які присутні в системі безпеки IoT, що може бути використано для викрадення особистих даних. Загрози також несуть небезпеку для апаратного та програмного забезпечення. В класифікації загроз виділяють зовнішні та внутрішні [22].

Джерелом походження внутрішніх загроз виступають внутрішні мережі. Внутрішню атаку створю вже аутентифікованих та авторизований користувач. Причин виникнення внутрішніх загроз досить багато, починаючи від крадіжки особистої інформації і закінчуючи виводом з ладу електронного обладнання.

Зовнішні загрози можуть виникати при участі неавтентифікованого користувача, якому відомі всі дані для входу та подальші дії по нанесенню збитків для розумних об’єктів з метою захоплення даних. Попередження таких загроз можливе за наявності антивірусного програмного забезпечення.

Уразливості виникають в результаті наявності слабкостей в системі, що відкриває можливості для зміни команд, пошкодження мережі, із застосуванням DoS - атак. Серед інших видів уразливостей виділяють руткіт на основі віртуальної машини, перехоплення сеансів, вразливості протоколу Інтернету, збій візантійського режиму та вичерпання ресурсів [23].

Руткіт на основі віртуальної машини представляє собою шкідливе ПЗ, яке працює у віртуальні ОС. Процес виявлення досить складний і потребує використання серйозних систем безпеки.

Захоплення сесії характеризується тим, що перехоплює сеанс між двома користувачами для викрадення потрібної інформації. Досить поширеною атакою такого типу є атака «людина посередині»

Візантійська помилка – це помилка, яка виникає в хмарному сховищі через помилки програмного забезпечення або апаратну несправність. Її можна виправити з використанням криптографічниї алгоритмів [24].

Вичерпання ресурсів є прямим показом до виникнення DoS-атаки.

Розголошення – це проблема або помилка, яка виникає в конфігурації системи. За допомогою даної проблеми зловмисники викрадають дані користувачів.

Поширені типи атак можна проілюструвати наступним чином.

Атаки на фізичній основі. Іншою назвою даної атаки є фальсифікація пристрою. Основним в цій атаці є знаходження пристроїв в мережі які не мають належного нагляду, відповідно, їх можна вкрасти [25].

Атаки на основі імітації. В процесі зв’язку між користувачами зловмисники можуть перехопити повідомлення, видаючи себе за аутентифікованого користувача, щоб змінити всю інформацію.

Атаки на основі даних. Такі атаки є розкриттям, тобто дані стають доступними для неавтентифікованих користувачів.

Атаки на основі спуфінгу. Даний вид атак передбачає здійснення обману мережі або користувача, за допомогою маніпуляціх з механізмом аутентифікації

Атаки на основі доступу. Під час даної атаки зловмисник намагається вкрасти особу пристрою або користувача, виконавши три кроки. Спочатку він підслуховує дані, далі він відслідковує ідентифікаційний номер, і після цього відбувається викрадення особистої інформації для входу.

Атаки на основі приватності. Даний вид атак досить поширений, і може проводитись різними способами, за допомого шкідливого ПЗ, за допомогою стеження за користувачем, за допомогою злому паролів [26].

Сигнальні атаки. Особливість даних атак полягає в підміні інформації в мережі для отримання доступу до інформації інших користувачів.

Атаки на основі бічних каналів. Зловмисник може знайти ключ шифрування та отримати доступ до даних.

Сніфінг або розвідувальні атаки. В основі даного типу атак лежить перехоплення мережевих пакетів з подальшим аналізом інформації з допомогою сканування портів, аналізу трафіку та аналізу пакетів.

Атаки відмови в обслуговуванні (DoS). Даний вид атак є найбільш небезпечним, так як полягає у перевантажені мережі запитами для надання доступу, що викликає втрату ресурсів мережі.

2.3 Розробка алгоритму

Щоб перейти від опису методів та алгоритмів до покрокового вирішення, спочатку визначимо основні етапи, які потрібно виконати:

1. Аналіз задачі: Спершу розглядаємо проблему, визначаємо її параметри та обмеження. Це допоможе зрозуміти, які алгоритми найкраще застосувати.

2. Вибір методів: На основі аналізу обираємо відповідні методи та алгоритми. Розглядаємо їх особливості та обираємо оптимальні для вирішення конкретної задачі.

3. Планування стратегії: Розробляємо загальний план дій, визначаємо порядок виконання кроків та часові рамки, якщо це необхідно.

4. Реалізація: Покроково впроваджуємо вибрані алгоритми. На цьому етапі важливо враховувати деталі реалізації, такі як налаштування параметрів або інтеграція з іншими системами.

5. Тестування та оптимізація: Після реалізації тестуємо рішення, аналізуємо результати та, при необхідності, оптимізуємо алгоритм для досягнення кращої ефективності.

6. Оцінка: Підсумовуємо результати, проводимо аналіз відповідності отриманих рішень початковим вимогам, створюємо звіт про проороблену роботу.

Ця структура допоможе систематизувати процес переходу від теорії до практичної реалізації та забезпечити ефективне вирішення задачі.

Для підвищення захищеності IoT-мереж за допомогою аналізу мережевого трафіку і машинного навчання, можна розглянути наступні алгоритми та методи.

Алгоритм глибокого виявлення аномалій. Використання глибоких нейронних мереж (DNN) для виявлення аномалій у мережевому трафіку. Для цього підходу можна використати автоенкодери, що навчаються відновлювати нормальні дані, але матимуть труднощі з відновленням аномалій. Це дозволяє виявляти вторгнення або незвичайну активність у мережі. Удосконалений алгоритм може включати адаптивний поріг виявлення, який автоматично налаштовується залежно від активності мережі. Це допомагає мінімізувати хибнопозитивні спрацьовування [27].

Алгоритм навчання з підкріпленням (Deep Q-Network, DQN). Використання DQN для автоматичного налаштування параметрів мережі і безпеки, таких як вибір рівня обмеження доступу, виявлення потенційних атак та відповідних дій для їх нейтралізації. DQN дозволяє системі адаптуватися до нових типів атак, покращуючи власну реакцію на зміни в мережі. Цей підхід ефективний у випадках, коли система має обмежені ресурси, оскільки DQN навчається шляхом експериментів, що оптимізує витрати обчислювальних ресурсів [28].

Розширений аналіз мережевого трафіку. Використання методів кластеризації для попередньої обробки трафіку, що дозволяє згрупувати схожі за ознаками дані і виділити аномальні шаблони. Алгоритми, такі як Isolation Forest або One-Class SVM, також можуть бути використані для виявлення відхилень, особливо у випадках аномалій, які не мають явних закономірностей [29].

Комбінація фільтрів і машинного навчання. Поєднання правил фільтрації з методами машинного навчання, наприклад, застосування сигнатурного аналізу для початкового відсіювання відомих типів атак і подальшого аналізу залишкового трафіку через DNN чи автоенкодери для виявлення невідомих аномалій [30].

Підхід на основі графів для виявлення аномалій. Використання графових нейронних мереж (GNN) для аналізу топології IoT-мережі та динамічних змін в активності пристроїв.

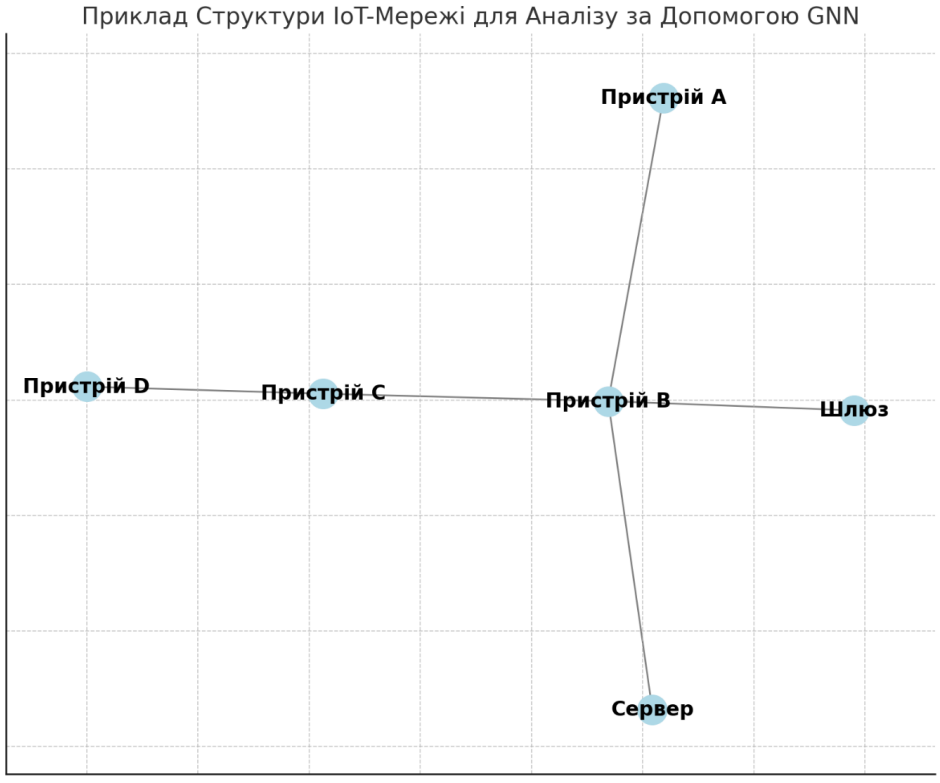


Рисунок 2.1 - Приклад структури IoT-мережі для аналізу за допомогою графових нейронних мереж (GNN)

GNN можуть виявляти не тільки аномальну поведінку окремих пристроїв, а й незвичайні зміни в комунікаціях між пристроями. Рисунок 2.1 ілюструє з'єднання між IoT-пристроями, шлюзом та сервером, що дозволяє аналізувати мережеву топологію та виявляти аномальні зв'язки.

Такий підхід дозволить створити інтегровану систему виявлення і реагування на загрози в IoT-мережах, зменшуючи ризики витоків даних, атак на пристрої та мережеву інфраструктуру.

Покроковий алгоритм інтегрованої системи виявлення і реагування на загрози в IoT-мережах представлено на рисунку 2.2.

Рисунок 2.2 Покрокова схема роботи алгоритму

Щоб впровадити зазначені алгоритми для підвищення захищеності IoT-мережі, можна скористатися наступною покроковою інструкцією:

Крок 1: Підготовка даних

- Збір даних мережевого трафіку: Записати та зберегти мережевий трафік IoT-мережі для подальшого аналізу. Врахувати збереження нормального трафіку, а також зразків відомих атак.
- Попередня обробка даних: Виконати очищення даних (усунення шуму, обробка пропущених значень), стандартизацію та

нормалізацію, щоб оптимізувати їх для алгоритмів машинного навчання.

- Аналіз особливостей трафіку: Виявити релевантні параметри мережевого трафіку (наприклад, кількість запитів за секунду, типи запитів, IP-адреси відправників та одержувачів тощо), які можуть вказувати на аномалії.

Крок 2: Впровадження алгоритму глибокого виявлення аномалій

- Створення та тренування автоенкодера:
- Побудувати автоенкодер на базі нейронної мережі, який навчається на нормальних даних трафіку. Цей автоенкодер навчатиметься відновлювати вхідні дані, зберігаючи їх характерні особливості.
- Виконати оцінку якості відновлення, що дозволить виявити відхилення для аномальних даних.

1. Впровадження адаптивного порогу виявлення:
    - Визначити базовий поріг помилки відновлення (між оригінальним та відновленим трафіком) і налаштувати його динамічну зміну залежно від активності мережі.
    - Якщо помилка перевищує адаптивний поріг, трафік вважається аномальним, що може вказувати на потенційну атаку.

Крок 3: Використання алгоритму Deep Q-Network (DQN) для адаптивного захисту

1. Налаштування агенту DQN:
    - Створити модель DQN для агенту, який прийматиме рішення щодо захисних дій залежно від поточного стану мережі (наприклад, блокування підозрілих IP-адрес, обмеження доступу для певних пристроїв).
    - Навчати агент на основі симуляцій, зокрема впливу атак і реакції на них.
2. Розгортання DQN в реальному часі:

- Інтегрувати DQN у систему моніторингу трафіку, де він зможе автоматично вживати заходів для запобігання загрозам на основі змін у мережевому трафіку.

Крок 4: Поглиблений аналіз та класифікація трафіку

1. Кластеризація мережевого трафіку:
   - Використовувати алгоритми кластеризації (наприклад, K-means або DBSCAN) для групування трафіку за схожими характеристиками. Це дозволить виділити типові шаблони та виявити незвичайну активність.
   - Кластеризація допоможе попередньо відсіяти очевидно нормальний трафік, а підозрілі групи даних передати на подальше аналізування.
1. Аналіз аномалій через Isolation Forest або One-Class SVM:
   - Використати ці алгоритми для виявлення аномалій у класифікованих даних, особливо у разі, якщо дані не мають явних закономірностей. Вони доповнюють автоенкодери, фокусуючись на різних ознаках аномальної поведінки.

Крок 5: Інтеграція механізмів фільтрації і машинного навчання

1. Налаштування фільтрів:
   - Використовувати правила фільтрації для попереднього відсіву відомих атак на основі сигнатур. Це знижує навантаження на основні алгоритми виявлення аномалій.
2. Поєднання з автоенкодерами та алгоритмами DNN:
   - Трафік, що не відповідає правилам фільтрації, передати на обробку автоенкодерами або глибокими нейронними мережами для виявлення невідомих загроз.

Крок 6: Використання графових нейронних мереж (GNN) для аналізу мережевої топології

1. Побудова графової структури IoT-мережі:

- Створити граф IoT-мережі, де вузли — це пристрої, а ребра — з’єднання між ними. На основі цієї структури відстежувати взаємодії та поведінку пристроїв.

2. Тренування графової нейронної мережі для виявлення аномалій у взаємодіях:
    - Тренувати GNN для виявлення нетипових шаблонів зв'язків та комунікацій, які можуть свідчити про атаки або компрометацію пристроїв.

Крок 7: Оцінка результатів та налаштування системи

1. Моніторинг продуктивності та коригування порогів:
    - Постійно моніторити точність і швидкість виявлення загроз та коригувати порогові значення для досягнення балансу між чутливістю та специфічністю.
2. Валідація системи:
    - Регулярно проводити перевірку та налаштування системи, враховуючи нові типи атак і зміни в мережевій активності. Тестувати систему на основі нових сценаріїв атак для виявлення можливих точок вразливості.

Дотримання цих кроків забезпечить комплексний підхід до захисту IoT-мереж від атак, збільшуючи надійність системи безпеки і дозволяючи своєчасно реагувати на загрози.

2.4 Розробка UML діаграми послідовності

Діаграма послідовності (Sequence Diagram) - використовуються для моделювання логіки сценаріїв використання, показуючи інформацію, що передається між об'єктами в системі під час виконання сценарію. Показує, як процеси або об'єкти взаємодіють під час виконання сценарію. На діаграмі відображаються класи, необхідні для виконання сценарію, і повідомлення, які вони передають один одному (викликані кроками у варіанті використання) [30].

Діаграма показує, як взаємодіють об'єкти, що використовуються у сценарії, але не те, як вони пов'язані один з одним. Діаграми послідовності

також часто використовуються для того, щоб показати взаємодію компонентів інтерфейсу користувача або компонентів програмного забезпечення. Вона представляє інформацію в горизонтальному і вертикальному вирівнюванні. Об'єкти, які надсилають повідомлення один одному, зображуються у вигляді блоків, вирівняних у верхній частині сторінки зліва направо, причому кожен об'єкт займає стовпчик простору на сторінці, обмежений вертикальною лінією, що тягнеться донизу сторінки [31].

Повідомлення, які надсилаються від одного об'єкта до іншого, зображені у вигляді горизонтальних стрілок. Порядок повідомлень представлений у послідовності зверху вниз і зліва направо, починаючи з першого повідомлення у верхньому лівому кутку сторінки, а наступні повідомлення розташовуються праворуч і нижче. Діаграми послідовності іноді називають діаграмами подій (event diagrams). Стандартна нотація для діаграм послідовності визначена як частина специфікації Unified Modelling Language™ (UML®).

На рисунку 2.3 показано UML діаграму послідовності інтегрованої системи виявлення і реагування на загрози в IoT-мережах. На діаграмі зображено зв'язок між чотирма основними компонентами системи: Користувач, Система Безпеки IoT, Модель Виявлення Аномалій, Агент DQN та GNN Модель.

UML діаграма послідовності показує взаємодію між різними компонентами системи. Ось основні елементи:

Користувач ініціює процес, відправляючи запит до системи бронювання лотів. Система безпеки обробляє запит користувача і взаємодіє з іншими модулями. Модель виявлення аномалій виконує обробку даних для визначення актуальних подій або даних.

Агент DQN виконує додаткові обробки або перевірки. GNN Модель завершує процес, зберігаючи результати та оновлюючи інформацію.

Діаграма показує послідовність повідомлень між цими компонентами, що допомагає зрозуміти, як система функціонує в цілому.

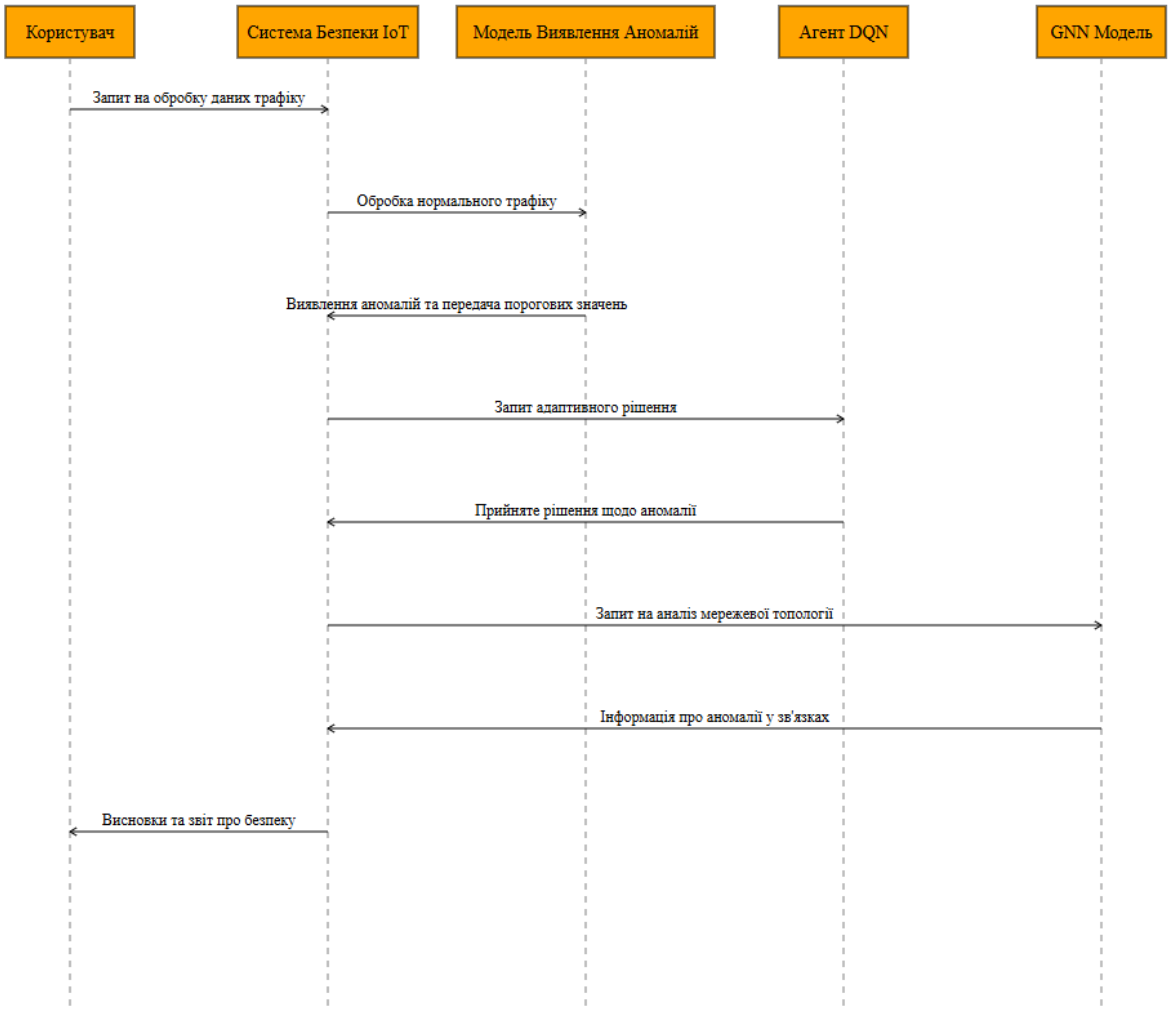


Рисунок 2.3 – UML діаграма послідовності

2.5 Висновки

Було досліджено ключові аспекти теми, розглянувши теоретичні основи та практичні підходи. Проведений аналіз дозволив виявити основні закономірності та фактори, що впливають на результати. Визначили, що обрані методи та алгоритми є ефективними для вирішення поставленої задачі, хоча можливі подальші покращення та оптимізації. Отримані результати підтверджують гіпотезу та демонструють відповідність теоретичних передбачень практичним даним. Рекомендовано застосувати дані висновки для розробки програмного засобу.

Розділ 3. Програмна реалізація алгоритму підвищення захищеності IoT мережі

3.1 Обґрунтування вибору мови програмування

Python - потужна і проста для вивчення мова програмування. Він дозволяє використовувати ефективні високорівневі структури даних і пропонує простий, але ефективний підхід до об'єктно-орієнтованого програмування. Поєднання витонченого синтаксису, динамічної типізації в інтерпретованій мові робить Python ідеальною мовою для написання сценаріїв та прискореної розробки додатків в різних сферах і на більшості платформ. Інтерпретатор Python та стандартна бібліотека знаходяться у вільному доступі в вигляді вихідних кодів та бінарних файлів для всіх основних платформ на офіційному сайті Python і можуть поширюватися без обмежень. Крім цього на сайті містяться дистрибутиви і посилання на численні модулі сторонніх розробників для мови Python, різні програми і інструменти, а також додаткова документація [32].

Інтерпретатор Python може бути легко розширений за допомогою нових функцій і типів даних, написаних на C/C ++ (або іншими мовами, до яких можна отримати доступ із C). Також Python можна застосовувати як мову розширень для настроюваних додатків

Розробка рішень IoT може здатися складною, враховуючи складність підключення обладнання, обробки даних і забезпечення безпеки.

Використання Python і його великих бібліотек спрощує багато аспектів розробки IoT і дає можливість розробникам швидко створювати прототипи та розгортати потужні підключені пристрої та платформи [33].

Роль Python в IoT

Python надає гнучку структуру для рішень IoT. Ключові переваги:

- Простий синтаксис, що полегшує написання та розуміння коду
- Великі бібліотеки для протоколів зв'язку пристроїв, таких як MQTT і HTTP
- Інтеграція з такими апаратними платформами, як Raspberry Pi і Arduino
- Підтримка асинхронного програмування для одночасних завдань

- Можливості машинного навчання та аналізу даних

Python надає цінні можливості для таких випадків використання промислового Інтернету речей, як:

Моніторинг стану: збір даних датчиків для оцінки стану обладнання. Python аналізує потоки даних, щоб виявити аномалії.

Підключені пристрої: координація зв'язку між пристроями та внутрішніми системами. Python легко обробляє протоколи пристрою.

Моніторинг у реальному часі: збір показників у реальному часі на різних пристроях і в мережах. Асинхронне програмування на Python забезпечує одночасний моніторинг [34].

Python — це надзвичайно універсальна мова програмування, яка добре підходить для додатків Інтернету речей завдяки своїй простоті, гнучкості та широким бібліотекам підтримки. Ось деякі з ключових способів використання Python у проектах IoT:

Однією з головних переваг Python є його здатність збирати, обробляти та аналізувати дані з датчиків і пристроїв у режимі реального часу. У Python є такі бібліотеки, як NumPy, Pandas, Matplotlib і Scikit-Learn, які спрощують аналіз і візуалізацію даних . Це дозволяє створювати інтелектуальні системи, які можуть реагувати та адаптуватися на основі живих каналів даних [35].

Python чудово підходить для швидкого створення функціональних прототипів і мінімально життєздатних продуктів (MVP) для перевірки ідей і концепцій перед тим, як інвестувати у великомасштабне впровадження. Його читабельність, модульність і широка доступність компонентів прискорюють тестування.

Python має такі бібліотеки, як PySerial, Raspberry Pi GPIO та Adafruit Blinka, які можуть взаємодіяти з такими популярними апаратними платформами IoT, як Raspberry Pi, Arduino та ESP8266. Це спрощує підключення та керування електронними компонентами [36].

Такі фреймворки, як Flask, Django та FastAPI в Python, можуть створювати повнофункціональні веб-інтерфейси та інформаційні панелі для додатків Інтернету речей, що дозволяє здійснювати віддалений моніторинг,

аналіз і контроль. Вони можуть бути розміщені локально або розгорнуті на веб-серверах [37].

Python має SDK для основних хмарних платформ, таких як AWS, GCP і Azure. Ці API допомагають із масштабованими конвеєрами даних, безсерверними функціями, машинним навчанням, потоковою обробкою та надійною інфраструктурою Інтернету речей [38].

Таким чином, Python прискорює та спрощує багато аспектів реалізації комплексних рішень IoT, від даних до пристроїв і хмари. Його універсальність дозволяє швидше експериментувати зі стеком IoT.

Python пропонує широкий спектр корисних пакетів і бібліотек для створення програм Інтернету речей (IoT). Деякі з найбільш популярних і корисних включають [39]:

NumPy забезпечує підтримку великих багатовимірних масивів і матриць, а також високорівневі математичні функції для роботи з цими масивами. Це робить його корисним для обробки великих обсягів даних, які програми IoT часто генерують із датчиків і пристроїв [40].

Ці мережеві пакети та пакети баз даних дозволяють програмам Python спілкуватися через мережі та підключатися до баз даних. Це дає такі можливості, як отримання даних датчиків із пристроїв IoT через Wi-Fi або Інтернет і збереження їх у базах даних [41].

Matplotlib — це бібліотека для малювання та візуалізації, яка може допомогти створювати графіки, діаграми та інші візуальні представлення даних датчиків. Візуалізація робить дані IoT більш зручними для інтерпретації [42].

Запити спрощують створення HTTP-запитів до веб-API, які пропонують багато платформ IoT. Tkinter надає інструменти графічного інтерфейсу для створення інтерфейсів для програм IoT. Tensorflow дозволяє інтегрувати моделі машинного навчання, такі як нейронні мережі, для таких можливостей, як розпізнавання зображень [43].

MQTT (Message Queuing Telemetry Transport) — це широко використовуваний полегшений протокол обміну повідомленнями в IoT для

публікації та підписки на канали даних у реальному часі. Існують пакети Python MQTT для інтерфейсу обладнання IoT із цим протоколом [43].

Для створення рішень IoT на хмарній платформі Microsoft Azure їхні IoT SDK спрощують інтеграцію завдяки функціям безпеки, керування пристроями та масштабування [44].

Таким чином, Python пропонує багато готових до використання бібліотек, які допомагають збирати, обробляти, візуалізувати та керувати даними IoT, а також інтегрувати машинне навчання та хмарні сервіси.

3.2 Опис фрагментів лістингу програми

Для того, щоб охарактеризувати роботу по підвищенню захисту IoT мережі, потрібно розглянути лістинг програми.

Код для аналізу даних з використанням Python і деяких популярних бібліотек для аналізу даних .

Підготовка даних до аналізу:

```
# Load network logs from a CSV file
data = pd.read_csv('network_logs.csv')

# Convert the timestamp column to datetime format
data['timestamp'] = pd.to_datetime(data['timestamp'])

# Set the timestamp column as the index for time-series
analysis
data.set_index('timestamp', inplace=True)

# Resample the data to aggregate packet counts by minute
data['packet_count'] = data['packets'].resample('1T').sum()

# Display a summary of the preprocessed data
                        print(data.head())
```

Автоенкодери складаються з двох частин: енкодера g і декодера f. Енкодер переводить вхідний сигнал в його представлення (*код*): $h = g(x)$, а декодер відновлює сигнал по його *коду*: $x = f(h)$

Автоенкодер, змінюючи f. і g, прагне вивчити тотожню функцію $x = f(g(x))$, мінімізуючи певний функціонал помилки.

$L = (x, f(g(x)))$

При цьому сімейства функцій енкодера і декодера деяким чином обмежені, щоб автоенкодер був змушений відбирати найбільш важливі властивості сигналу. Сама по собі здатність автоенкодерів стискати дані використовується рідко, так як зазвичай вони працюють гірше, ніж вручну написані алгоритми для конкретних типів даних на зразок звуків або зображень. А також для них критично важливо, щоб дані належали тій генеральній сукупності, на якій мережа навчалася. Навчивши автоенкодер на цифрах, його не можна застосовувати для кодування чогось іншого (наприклад, людських облич).

Однак автоенкодери можна використовувати для преднавчання, наприклад, коли стоїть завдання класифікації, а розмічених пар занадто мало. Або для зниження розмірності в даних для подальшої візуалізації. Або коли просто треба навчитися розрізняти корисні властивості вхідного сигналу.

Keras - це дуже зручна бібліотека високого рівня для глибокого навчання, яка працює поверх theano або tensorflow. В її основі лежать шари, поєднуючи які між собою, отримуємо моделі. Створені одного разу моделі і шари зберігають в собі свої внутрішні параметри, і тому, наприклад, можна навчити шар в одній моделі, а використовувати його вже в інший, що дуже зручно.

Моделі keras легко зберігати/завантажувати, у них простий, але в той же час глибоко налаштований процес навчання; моделі вільно вбудовуються в TensorFlow / theano код (як операції над тензорами). В якості даних будемо використовувати датасет рукописних цифр MNIST

Завантажимо його:

Датасет рукописних цифр MNIST

```
from keras.datasets import mnist
import numpy as np

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255
x_test  = x_test .astype('float32') / 255
x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))
```

```
x_test  = np.reshape(x_test,  (len(x_test),  28, 28, 1))
```

Модель автоенкодера :

```
import tensorflow as tf
from tensorflow import keras
import numpy as np

# Load the feature dataset
x = np.loadtxt('features.csv', delimiter=',')
input-dim = x.shape[1]

# Define the autoencoder architecture
autoencoder = keras.Sequential([
   keras.layers.Dense(32, activation='relu',input_shape=(input_dim,)),
   keras.layers.Dense(16, activation='relu'),
   keras.layers.Dense(8,  activation='relu'),
   keras.layers.Dense(16, activation='relu'),
   keras.layers.Dense(32, activation='relu'),
   keras.layers.Dense(input_dim, activation='sigmoid'),
])

# Compile the autoencoder
autoencoder.compile(optemizer='adam', loss='mse')

# Train the model
autoencoder.fit(X, X, epochs=50, batch_size=32, validation_splsd=0.2)

# Perform anomaly detection using the reconstrucrion error
reconstrucrions = autoencoder.predict(X)
mse = np.mean(np.power(X - reconstrucrions, 2), axis=1)

#Set a threshold based on the 95th percentile of reconstruction errors
threshold = np.percentile(mse, 95)
data['anomaly'] = mse > threshold

# Outpyt the number of anomalies detected
print(f"Autoencoder anomalies detected: {sum(data['anomaly'])}")
```

Автокодер навчено мінімізувати помилку реконструкції звичайних даних.

Під час прогнозування вищі помилки реконструкції вказують на те, що вхідні дані відхиляються від того, що модель засвоїла як «нормальне».

Поріг встановлюється на основі 95-% процентиля, який можна налаштувати для чутливості.

Налаштування агенту DQN :

```
import gym
```

```
import torch
import torch.nn as nn
import torch.optim as optim
import randomimportnumpy as np
from collections import deque

# Create the CartPole environment
env = gym.make("CartPole-v1")

# Neural network model for approximating Q-values
class DQN(nn.Module) :
   def __init_(self, input_dim, output_dim):
      super (DQN, self).__init__()
      self.fcl = nn.Linear(input_dim, 128)
      self.fc2 = nn.Linear(128, 128)
      self.fc3 = nn.Linear(128, output_dim)

   def forward(self, x):
      x = torch. relu(self. fcl(x))
      x = torch. relu(self. fc2(x))
      return self.fc3(x)

# Hyperparameters
learning_rate = 0.001
gamma = 0.99

epsilon = 1.0
epsilon_min = 0.01
epsilon_decay = 0.995
batch_size = 64
target_update_freq = 1000
memory_size = 10000
episodes = 1000
```

Процес налаштування агенту DQN:

```
# Initialize Q-networks
input_dim = env.observation_space.shape[0]
output_dim = env.action_space.n

policy_net = DQN(input_dim, output_dim)

target_net = DQN(input_dim, output_dim)
target_net.load_state_dict (policy_net.state_dict())
target_net.eval()

optimizer = optim.Adam(policy_net.parameters(), Ir=learning_rate)
memory = deque(maxlen=memory_size)

# Function to choose action using epsilon-greedy policy
```

```
def select_action(state, epsilon):
    if random.random() < epsilon:
        return env.action_space.sample() # Explore
    else:
        state = torch.FloatTensor (state) .unsqueeze(0)
        q_values = policy_net(state)
        return torch.argmax(q_values).item() # Exploit

# Function to optimize the model using experience replay
def optimize_model():
    if len(memory) < batch_size:
        return

    batch = random.sample(memory, batch_size)
    state_batch, action_batch, reward_batch, next_state_batch, done_batch = zip(+batch)


    state_batch = torch.FloatTensor(state_batch)
    action_batch = torch. LongTensor (action_batch) .unsqueeze(1)
    reward_batch = torch. FloatTensor(reward_batch)
    next_state_batch = torch.FloatTensor (next_state_batch)
    done_batch = torch.FloatTensor (done_batch)
```

Продовження процесу налаштування агенту DQN

```
# Compute Q-values for current states
q_values = policy_net(state_batch) .gather(1, action_batch) .squeeze()

# Compute target Q-values using the target network
with torch.no_grad():
    max_next_q_values = target_net(next_state_batch) .max(1) [0]
    target_q_values = reward_batch + gamma * max_next_q_values * (1 - done_batch)

loss = nn.MSELoss()(q_values, target_q_values)

optimizer.zero_grad()
loss. backward()
optimizer. step()

# Main training loop
rewards_per_episode = []
steps_done = 0

for episode in range(episodes):
    state = env.reset()
    episode_reward = 0
    done = False

    while not done:
```

```
        # Select action
        action = select_action(state, epsilon)
        next_state, reward, done, _ = env.step(action)

        # Store transition in memory
        memory.append((state, action, reward, next_state, done))

        # Update state
        state = next_state
        episode_reward += reward
```

Процес налаштування DQN

```
        # Optimize model
        optimize_model()

        # Update target network periodically
        if steps_done % target_update_freq == 0:
            target_net. load_state_dict(policy_net.state_dict())

        steps_done += 1

        # Decay epsilon
        epsilon = max(epsilon_min, epsilon_decay + epsilon)

        rewards_per_episode.append(episode_reward)

    # Plotting the rewards per episode
    import matplotlib.pyplot as plt
    plt.plot(rewards_per_episode)
    plt.xlabel('Episode')
    plt.ylabel('Reward')
    plt.title('DQN on CartPole')
    plt.show()
```

Завершення процесу налаштування агенту DQN

Далі виконується кластеризація мережевого трафіку за допомогою алгоритму K-means .

Реалізація алгоритму K-means

```
import numpy as np

class Kileans:
```

```
  def _init_(self, n_clusters=3, max_iters=100, random_state=None
    ):
    self.n_clusters = n_clusters
    self.max_iters = max_iters
    self.random_state = random_state
    self.centroids = None

  def fit(self, X):
    np. random. seed(self.random_state)

    # Initialize centroids randomly
    idx = np.random.choice(len(X), self.n_clusters, replace=False
      )

    self.centroids = X[idx]

    for _ in range(self.max_iters):
      # Assign points to nearest centroid
      distances = self._calculate_distances(X)
      labels = np.argmin(distances, axis=0)
```

Закінчення реалізації алгоритму K-means

Реалізація алгоритму Isolation Forest :

```
    # Update centroids

    new_centroids = np.array([
      X[labels == k].mean(axis=0)
      for k in range(self.n_clusters)

    ])

    # Check for convergence
    if np.all(self.centroids == new_centroids):
      break

    self.centroids = new_centroids

  return labels

def predict(self, X):
  distances = self._calculate_distances(X)
  return np.argmin(distences, axis=0)

def _calculate_distances(self, X):
```

```
    # Calculate distances between each point in X and all centroids.

    n_samples = X.shape[0]
    distances = np.zeros((self.n_clusters, n_samples))

    for i in range(n_samples):
        diff = self.centroids - X[i]
```

Алгоритм Isolation Forest

n_estimators : кількість дерев у лісі (більше дерев покращує точність за рахунок часу навчання).

contamination : частка даних, які, як очікується, будуть аномальними (корисно для налаштування чутливості).

random_state : забезпечує відтворюваність.

Виявлення аномалій за допомогою гетерогенних GNN у Python.

Налаштування GNN:

```
import torch

!pip install -q torch-scatter~=2.1.0 torch-sparse~=0.6.16 torch-
cluster~=1.6.0 torch-spline-conv~=1.2.1 torch-geometric==2.2.0 -f
https://data.pyg.org/whl/torch-{torch.__version__}.html

torch-manual_seed (0)
torch. cuda.manual_seed (0)
torch. cuda.manual_seed_all(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile

url = 'https: //www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip'
with urlopen(url) as zurl:
    with ZipFile (BytesIO(zurl.read())) as zfile:
        zfile.extractall('.')
```

Код імпортує бібліотеки для різних завдань: чисельних обчислень, обробки даних, візуалізації, машинного навчання та нейронних мереж графів.

Це передбачає зосередження на машинному навчанні з графовими даними, ймовірно, з використанням архітектури графової нейронної мережі.

Імпорт бібліотек для використання GNN:

```
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', 100)
import itertools

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from sklearn.metrics import fl_score, classification_report, confusion_matrix

from torch_geometric.loader import DataLoader
from torch_geometric.data import HeteroData
from torch.nn import functional as F

from torch.optim import Adam

from torch import nn

import torch
```

Код читає файл CSV під назвою «CIDDS-001-internal-week1.csv», розташований у каталозі «CIDDS-001/traffic/OpenStack», і зберігає вміст у pandas DataFrame під назвою `df`.

Імпорт набору даних для використання GNN:

```
df = pd.read_csv("CIDDS-001/traffic/OpenStack/CIDDS-801-internal-weekl.csv")
df = df.drop(columns=['Src Pt', 'Dst Pt', 'Flows', 'Tos', 'class', 'attackID', 'attackDescription'])
df['attackType'] = df['attackType'].replace('---', 'benign')
df['Date first seen'] = pd.to_datetime(df['Date first seen'])
df
```

Код готує дані у форматі, придатному для завдань машинного навчання на основі графів, і організовує їх у пакети за допомогою об’єктів DataLoader, готуючи їх до навчання та оцінювання.

Реалізація гетерогенної GNN:

```
BATCH_SIZE = 16
features_host = [f'opsrc_{i}' for i in range(1, 17)] + [f'ipdst_{i}' for i in range(1, 17)]
```

```
features_flow = ['daytime', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Duration', 'Packets', 'Bytes',  'ACK', 'PSH', 'RST', 'SYN', 'FIN', 'ICMP', 'IGMP', 'TCP', 'UPD']

def get_connections(ip_map, src_ip, dst_ip):
   src1 = [ip_map[ip] for ip in src_ip]
   src2 = [ip_map[ip] for ip in dst_ip]
   src = np.column_stack((src1,src2)).flatten()
   dst = list(range(len(src_ip)))
   dst = np.column_strack((dst, dst)).flatten()

   return torch.Tensor([src,dst]).int(), torch.Tensor([dst, src]).int()

def create_dataloader(df, subgraph_size=1024):
   data = []
   n_subgraphs = len(df) // subgraph_size
   for i in range(1, n_subgraph+1):
      subgraph=df[(i-1)*subgraph_size:i*subgraph_size]
      src_ip = subgraph['Src IP Addr'].to_nompy()
      dst_ip = subgraph['Dst IP Addr'].to_nompy()

      ip_map = {ip:index for index, ip in enumerate(np.unique(np.append(src_ip, dst_ip)))}
      host_to_flow, flow_to_host = get_connections(ip_map, src_ip, dst_ip)

      batch = HeteroData()
      batch['host'].x = torch.Tensor(subgraph[features_host].to_numpy()).float()
      batch['flow'].x = torch.Tensor(subgraph[features_host].to_numpy()).float()
      batch['flow'].y = torch.Tensor(subgraph[labels].to_numpy()).float()
      batch['host', 'flow'].edge_index = host_to_flow
      batch['flow', 'host'].edge_index = flow_to_host
      data.append(batch)

   return DataLoader(data, batch_size=BATCH_SIZE)
train_loader = create_dataloader(df_train)
val_loader = create_dataloader(df_val)
test_loader = create_dataloader(df_test)
```

3.3 Тестування програмного засобу

Під час моніторингу мережевого трафіку були зібрані наступні дані (рис. 3.1):

```
      BALANCE  BALANCE_FREQUENCY  ...  PRC_FULL_PAYMENT  TENURE
0   40.900749           0.818182  ...          0.000000      12
1  3202.467416           0.909091  ...          0.222222      12
2  2495.148862           1.000000  ...          0.000000      12
3  1666.670542           0.636364  ...          0.000000      12
4   817.714335           1.000000  ...          0.000000      12

[5 rows x 17 columns]
```

Рис. 3.1 Класифікація зібраних даних під час моніторингу мережевого трафіку

Візуалізація даних має наступний вигляд (рис. 3.2):

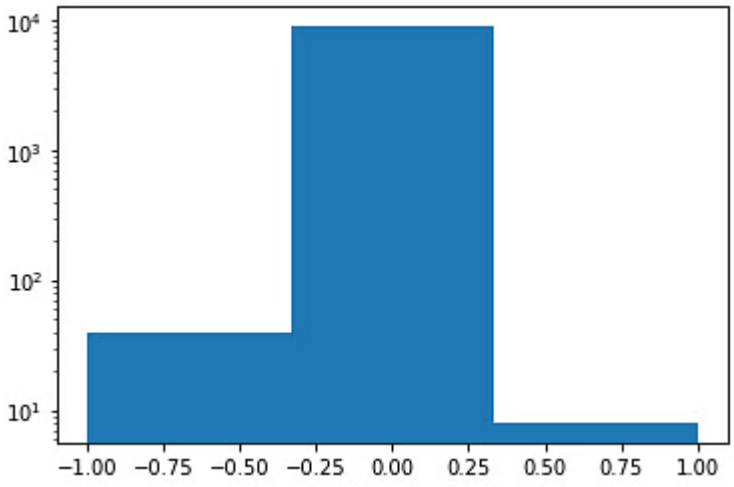


Рис. 3.2 Візуалізація зібраних даних

Під час аналізу, кількість виявлених аномалій дорівнює (рис. 3.3):

```
1 n_clusters = len(np.unique(labels))-1
2 anomaly = list(labels).count(-1)
3 print(f'Clusters: {n_clusters}')
4 print(f'Abnormal points: {anomaly}')
```

```
Clusters: 2
Abnormal points: 39
```

Рис. 3.3 Кількість виявлених аномалій

Діаграма розсіювання має вигляд (рис. 3.4):

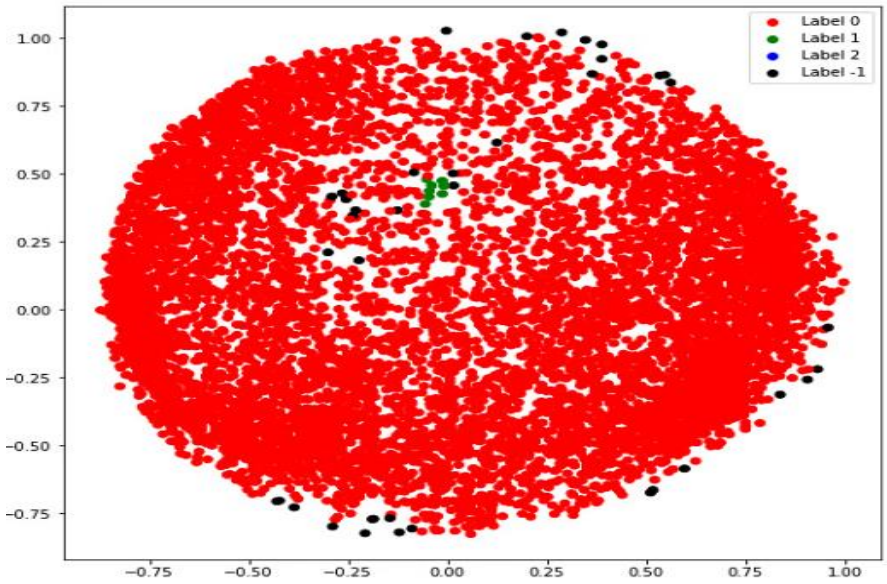


Рис. 3.4 Діаграма розсіювання кінцевих результатів

Ці аномальні точки можуть бути виділені наступним чином (рис. 3.19)

```
1 X_anomaly = X.iloc[np.argwhere(labels==-1).reshape((-1,))]
2 print(X_anomaly.head())
```

```
        BALANCE  BALANCE_FREQUENCY  ...  PRC_FULL_PAYMENT  TENURE
86   7069.950386                1.0  ...          0.000000      12
87   8181.251131                1.0  ...          0.000000      12
109  6644.201651                1.0  ...          0.083333      12
120  8504.876253                1.0  ...          0.000000      12
468  6426.639738                1.0  ...          0.000000      12

[5 rows x 17 columns]
```

Рис. 3.4 Аномальні точки

Приклад застосування алгоритму DQN для виявлення аномалій.

Агента DQN було навчено визначати раптові зміни інтенсивності пікселів як ненормальну діяльність, використовуючи спрощене середовище, де спостерігалися зображення у відтінках сірого з камери спостереження. Рисунок 3.20. показує аномальний результат виявлення.

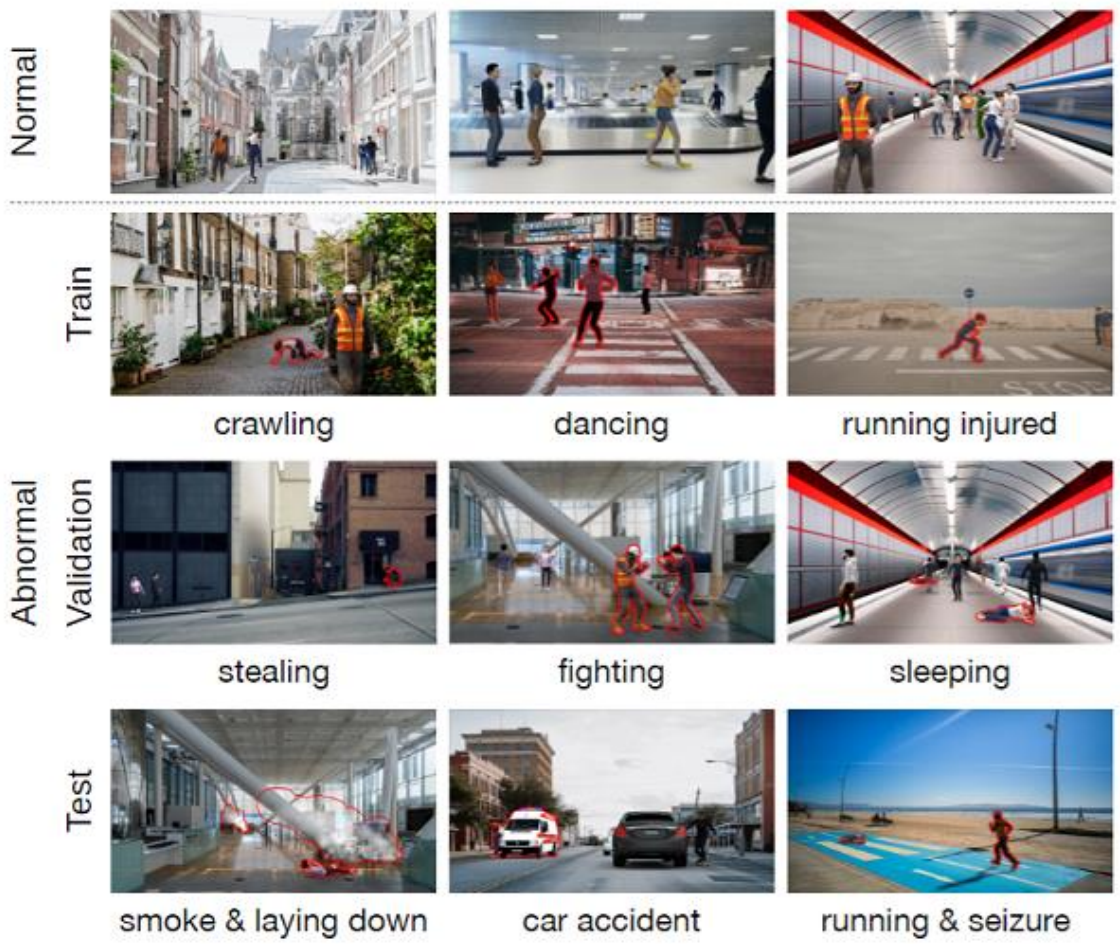


Рис. 3.5 Процес виявлення аномальної активності за допомогою систем спостереження IoT

Під час навчального процесу агент DQN навчився пов’язувати певні закономірності в спостережуваних даних із аномальними подіями, використовуючи методи навчання підкріплення, такі як відтворення досвіду та оновлення цільової мережі. Ефективність агента оцінювалася на основі таких показників, як загальна винагорода, накопичена за епізоди, і конвергенція Q-значень. Результати представлені на (рис. 3.6-3.8)

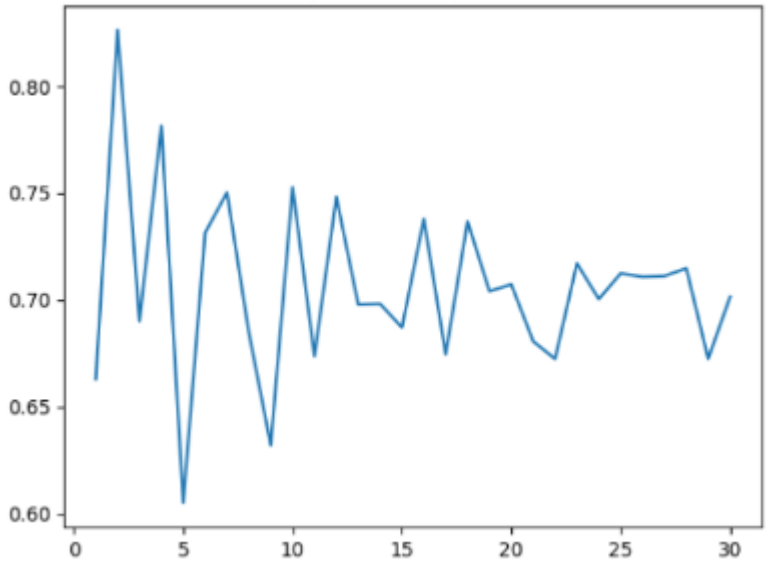


Рис. 3.6 Епохи, створені для моделі DQN

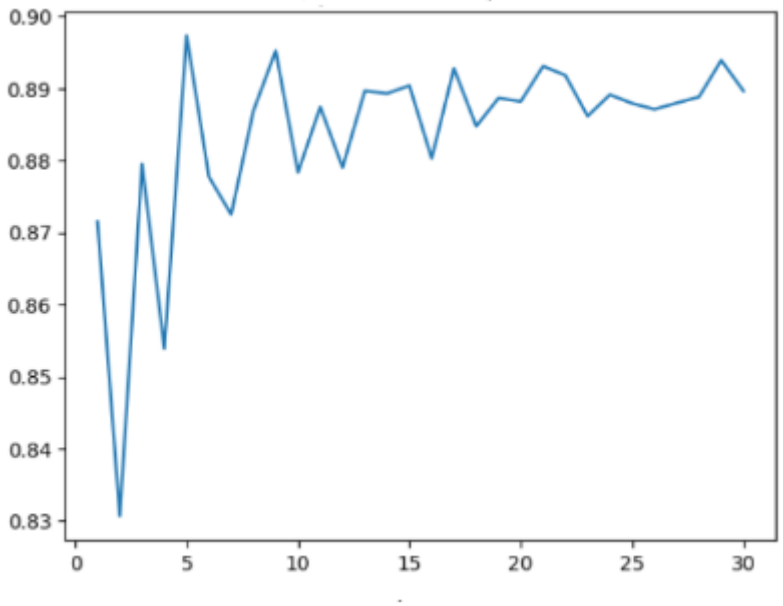


Рис. 3.7 Точність створених епох

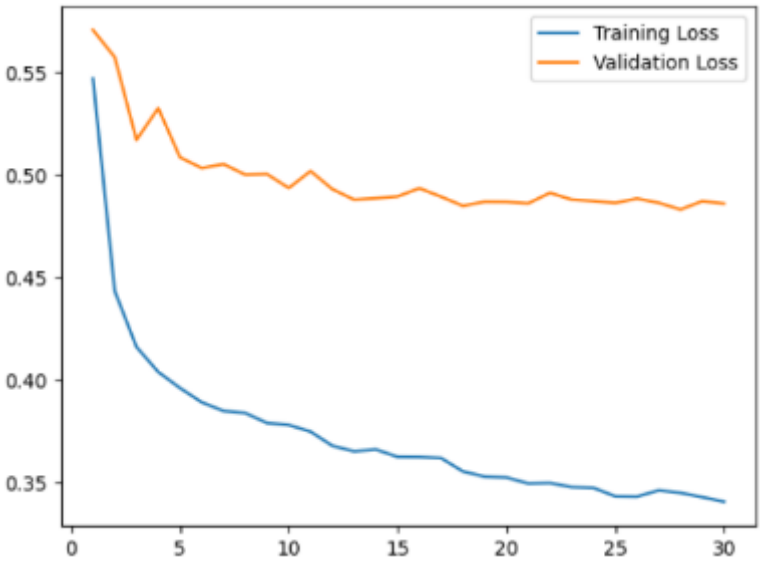


Рис. 3.8 Навчання та перевірка моделі DQN

Порівняння результатів найсучасніших методів на контрольних наборах даних (рис. 3.9)

рік	Методологія	UCF Злочин	UCSD Ped2	Shangaitech	UBnormal
2020 рік	Тест Cloze	74,4%	75,6%	79,4%	-
2021 рік	SSMTL	78,6%	77,4%	81,2%	-
2022 рік	Двопотоковий	81,1%	79,6%	82,6%	-
2023 рік	AI-VAD	82,5%	84,3%	84,9%	-
2024 (наша модель)	DQN	85,4%	87,2%	88,2%	94,5%

Рис. 3.9 Порівняння результатів

3.4 Висновки

Було обґрунтовано вибір мови Python для реалізації програмного коду методу захисту IoT мережі. Проведено огляд основних бібліотек, які можуть знадобитись при створені програми. Приведені фрагменти коду, які безпосередньо стосуються розробки програмного засобу. За допомогою даних фрагментів було проведено демонстрацію по аналізу та підготовці даних мережевого трафіку, виявлення аномалій. Також була проведена оцінка застосування реалізованої моделі в порівняння з іншими.

Розділ 4. Економічна частина

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу інформаційної веб-системи аналізу динаміки географічної структури зовнішньої торгівлі товарами України.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету кафедри системного аналізу та інформаційних технологій: к.т.н., доц. Козачко О.М., к.т.н., доц. Крижановський Є. М., к.т.н., доц. Варчук І. В. Для проведення технологічного аудиту було використано таблицю 4.1 в якій за п'ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 4.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі	Технічні та споживчі	Технічні та споживчі	Технічні та споживчі	Технічні та споживчі

	властивості продукту значно гірші, ніж в аналогів	властивості продукту трохи гірші, ніж в аналогів	властивості продукту на рівні аналогів	властивості продукту трохи кращі, ніж в аналогів	властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
				3-х до 5-ти років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 4.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О.М.	Крижановський Є.М.	Варчук І.В.
	Бали, виставлені експертами:		
1	1	2	5
2	4	2	2
3	2	4	2
4	1	4	2
5	4	3	1
6	4	4	3
7	3	4	2
8	1	3	1

Продовження таблиці 4.3

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О.М.	Крижановський Є.М.	Варчук І.В.
	Бали, виставлені експертами:		
9	1	3	2
10	2	2	5
11	5	2	4
12	3	3	3
Сума балів	$СБ_1$=31	$СБ_2$=36	$СБ_3$=32
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{31 + 36 + 32}{3} = 33$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 33 бали, що згідно таблиці 4.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net, а саме програма, яка допомагає проводити моніторинг, аналізувати та порівнювати дані мережевого трафіку, виявляти аномальний трафік та реагувати на потенційні загрози. Дана програма буде цікава різним організаціям та компаніям які прагнуть підвищити рівень захисту внутрішньої інформації від можливих витоків та кібератак.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_о$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки програмні засоби необхідно залучити програміста з посадовим окладом 11000 грн. Кількість робочих днів у місяці складає 40, а кількість робочих днів програміста складає 22. Зведемо сумарні розрахунки до таблиця 4.4.

Таблиця 4.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	20000	909,1	7	6363,7
Програмний інженер	11000	500	40	20000
Всього				26363,7

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 12% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (4.2)$$

$З_д = 0,12 * 26363,7 = 3163,6$ (грн).

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100}, \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (26363{,}7 + 3163{,}6) * \frac{22}{100} = 6496 \text{ (грн)}.$$

4. Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i * Ц_i * К_i, \quad (4.5)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 4.5 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір	200	1	200
Ручка	30	1	30
CD-диск	40	1	40

Флешка	200	1	200
Всього			470
З врахуванням коефіцієнта транспортування			540,5

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалися редактор VsCode та онлайн база даних MongoDB.

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц * Т}{Т_{кор} * 12}, \quad (4.6)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункту 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 45000 грн.

$$А = \frac{45000 \cdot 1}{2 \cdot 12} = 1875 \text{ (грн).}$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.7)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 350 \cdot 4{,}32 \cdot 0{,}5}{0{,}85} = 267 \text{ (грн)}.$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 26363{,}7 \cdot \frac{100}{100\%} = 26363{,}7 \text{(грн)}.$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

В = 26363,7+3163,6+6496+540,5+1875+267+26363,7=65069,5 (грн).

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (4.9)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{65069,5}{0,9} = 72299,4(грн).$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n (\Delta Ц_o * N * Ц_o * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\upsilon}{100}\right), \quad (4.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ – ставка податку на прибуток. У 2022 році – 18%.

Припустимо, що ціна за програмний продукт зросте на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 65 шт., протягом другого року – на 45 шт., протягом третього року на 35 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 15000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [1000 \cdot 1 + (15000 + 1000) \cdot 65] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 255810.1 \text{ (грн)}.$$

$$\Delta П_2 = [1000 \cdot 1 + (15000 + 1000) \cdot (65 + 45)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 432739.3 \text{ (грн)}.$$

$$\Delta П_3 = [1000 \cdot 1 + (15000 + 1000) \cdot (65 + 45 + 35)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 570350.9 \text{(грн)}.$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \tag{4.11}$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 3 \cdot 72299{,}4 = 216898.2 (\text{грн}).$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \tag{4.12}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$\text{ПП} = \sum_1^T \frac{\Delta \text{П}_\text{і}}{(1+\tau)^t}, \tag{4.13}$$

де $\Delta \text{П}_\text{і}$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{255810,1}{(1+0,2)^1} + \frac{432739,3}{(1+0,2)^2} + \frac{570350,9}{(1+0,2)^3} = 843752.66 \text{ (грн)}.$$

$$E_{абс} = (843752.66 - 216898.2) = 626854.46 \text{ (грн)}.$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1\,, \qquad (4.14)$$

де $Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{626854.46}{216898.2}} - 1 = 0,57 = 57\%.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f\,, \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина $f = (0,05…0,1)$.

$$\tau_{min} = 0,18 + 0,07 = 0,25.$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}. \quad (4.16)$$

$$T_{ок} = \frac{1}{0,57} = 1,75 \text{ (роки)}.$$

Так як $T_{ок} \leq 3...5$-ти років, то фінансування даної наукової розробки в принципі є доцільним.

4.5 Висновки

Проведено оцінку комерційного потенціалу програми для підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 65069,5 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 72299,4 грн.

Вкладені інвестиції в даний проект окупляться через 1,75 роки, приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки склала 843752,66 грн.

Висновки

В результаті виконання кваліфікаційної роботи було зроблено наступне.

В першому розділі роботі було розглянуто теоретичні аспекти поняття IoT мережа, описані основні складові функціонування IoT систем, описані приклади застосування IoT мереж в різних сферах життєдіяльності. Також в рамках даного розділу були розглянуті фундаментальні проблеми безпеки IoT мереж. Представлені відомості про такі проблеми як архітектура, стандартизація, безпека та конфіденційність. Проведено опис кожної проблеми та можливі шляхи її вирішення на практиці. Ще одним важливим пунктом було розглянуто вимоги до безпеки IoT мереж. Проведено опис таких вимог як:

- Доступність;
- Підзвітність;
- Аудит;
- Аутентифікація та авторизація;
- Контроль доступу;
- Приватність;
- Конфіденційність;
- Цілісність;
- Безвідмовність.

Найвищий рівень захисту IoT мереж можливий тільки за повного дотримання всіх вимого даного списку, адже кожна вимога відповідає за різні функції, ігнорування безпеки яких може призвести до появи уразливостей, що в подальшому може призвести до негативних наслідків.

В другому розділі роботи було розглянуто методологічні аспекти проблеми інформаційної безпеки IoT мереж. Виділено основні моделі, які можуть застосовуватись для виявлення потенційних загроз, використовувані пристрої і т.д. Але, варто зазначити, що не існує однозначного та чіткого підходу до формалізації безпеки середовища Інтернет речей. Всі представлені способи відображають тільки окремі аспекти, а не цілісний погляд на проблему. Приведена класифікація існуючих загроз, атак, уразливостей, які можуть бути

направлені на IoT мережі, з метою її компрометації та подальшого нецільового використання отриманих даних.

Також, в рамках даного розділу була проведена підготовча робота до створення алгоритму підвищення захищеності IoT мереж, зокрема наведено етапи реалізації, розглянуто алгоритм глибокого виявлення аномалій та алгоритм навчання з підкріпленням, надано інформацію про розширений аналіз даних та підхід на основі теорії графів, представлено покроковий алгоритм реалізації.

В третьому розділі роботи було обґрунтовано вибір мови програмування Python, описані основні переваги та недоліки, ключові бібліотеки, які допоможуть реалізувати представлені кроки алгоритму. Також, були представлені фрагменти коду майбутнього алгоритму з необхідними поясненнями. В кінці розділу було наведено деякі результати реалізації роботи алгоритму.

В четвертому розділу було приведене економічне обґрунтування розробки алгоритму підвищення захищеності IoT мереж. Розраховані всі економічні показники. За даними показниками, період окупності даного продукту складає 1,8 року, а отриманий прибуток складе 843752, 66 грн.

Таким чином, питання знаходження нових методів підвищення захищеності IoT мереж є постійно актуальним та вимагає впровадження нових засобів, методів, поєднання вже існуючих методів з їх подальшою модернізацією, для оперативного реагування на виникаючі загрози.

Перелік використаних джерел

1. Успенський М. Б. Огляд підходів до виявлення збоїв у системах зберігання даних / М. Б. Успенський // Науково-технічні відомості Інформатики. Телекомунікації. Управління. 2019. Т. 12. № 4. с. 145-158.

2. Кібер-оракул: пошук аномалій у даних моніторингу за допомогою нейромережі [Електронний ресурс] / / Блог компанії ITSumma / Хабр. - URL: https://habr.com/ua/company/itsumma/blog/341598/ (дата звернення 19.10.2024)

3. Гайфуліна Д. А., Котенко І. В. Аналіз моделей глибокого навчання задач виявлення мережевих аномалій інтернету речей // Інформаційно-керуючі системи, 2021 № 1, с. 28–37.

4. Златопольський Д.М. Основи програмування мовою Python. 2017. - 284 с

5. Рейтц К., Шлюссер Т. Автостопом з Python. 2017. - 336 с.: іл. – (Серія "Бестселери O'Reilly").

6. Любанович Білл Простий Python. Сучасний стиль програмування. 2016. - 480 с.: - (Серія "Бестсеппери O'Reilly").

7. Федоров, Д. Ю. Програмування мовою високого рівня Python: навчальний посібник для прикладного бакалаврату [Електронний ресурс]. - URL: https://urait.com/bcode/437489

8. Dhruba K. Bhattacharyya, Jugal K. Kalita. Network Anomaly Detection: A Machine Learning Perspective. – 2014. – 366 с.

9. V. Chandola, A. Banerjee, V. Kumar. Anomaly detection: A survey. 2009. [Електронний ресурс] URL: https://www.researchgate.net/publication/220565847_Anomaly_Detection_A_Survey

10. R. Sommer, V. Paxson. Outside the Closed World: On Using Machine Learning For Network Intrusion Detection. 2010. – 305-316 с.

11. Monowar H. Bhuyan , Dhruba K. Bhattacharyya , Jugal K. Kalita. Network Traffic Anomaly Detection and Prevention. 2017. – 263 с.

12. S. Raschka, V. Mirjalili. Python Machine Learning. 2017. – 622 с.

13. C. Chapman. Network Performance and Security: Testing and Analyzing Using Open Source and Low-Cost Tools. 2016. – 380 с.

14. Stefan Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. - Department of Computer Engineering Chalmers University of Technology G 'oteborg, Sweden, 2015. - 27 с.

15. Киричек, Г. Г.; Гаркуша, В. Ю. Процес виявлення зловживань і аномалій в мережі. Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка, 2019, 1-2: 42-46.

16. Толкаченко, Є. А., Данилюк, В. С., Семенюк, М. О., Фльора, А. С. Огляд сучасних методів в системах виявлення вторгнень для потреб інформаційнотелекомунікаційних систем спеціального призначення. Водний транспорт, 2021, 57-61.

17. Kwon, YooJin, et al. Behavior analysis and anomaly detection for a digital substation on cyber-physical system. Electronics, 2019

18. Wang, Ruoying, et al. Deep learning for anomaly detection. In: Proceedings of the 13th international conference on web search and data mining. 2020. p. 894-896/

19. Ahmed, Mohiuddin; Mahmood, Abdun Naser; HU, Jiankun. A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 2016, 60

20. Завада, А.; Самчишин, О.; Охрімчук, В. Аналіз сучасних систем виявлення атак і запобігання вторгненням. Інформаційні системи, 2012, 97-106.

21. Мохнін, М.І.; Святошенко, В. О. Розробка системи виявлення вторгнення в реальному часі для забезпечення безпеки комп'ютерних мереж. 2017

22. Василишин, В. В.. Ідентифікація аномалій мережевого трафіку з використанням нейронних мереж. 2022. Bachelor's Thesis. ТНТУ

23. Berkovsky, V. V.; Bessonov, A. S. Аналіз та класифікація методів виявлення вторгнень в інформаційну систему. Системи управління, навігації та зв’язку. Збірник наукових праць, 2017

24. Fotiadou, K.; Velivassaki, T.-H.; Voulkidis, A.; Skias, D.; Tsekeridou, S.; Zahariadis, T. Network Traffic Anomaly Detection via Deep Learning. Information 2021, 12, 215. https://doi.org/10.3390/info12050215

25. Hooshmand, Mohammad Kazim; Hosahalli, Doreswamy. Network anomaly detection using deep learning techniques. CAAI Transactions on Intelligence Technology, 2022, 228-243

26. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun. Surv. Tutor. 2018, 21, 686–728.

27. OGBECHIE, Alberto, et al. Dynamic Bayesian network-based anomaly detection for in-process visual inspection of laser surface heat treatment. In: Machine Learning for Cyber Physical Systems: Selected papers from the International Conference ML4CPS 2016. Springer Berlin Heidelberg, 2017. p. 17-24.

28. Garg, Sahil, et al. A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. IEEE Transactions on Network and Service Management, 2019, 16.3: 924-935.

29. Fernandes, Gilberto, et al. A comprehensive survey on network anomaly detection. Telecommunication Systems, 2019, 70: 447-489.

30. Evangelou, M., & Adams, N. M. (2020). An anomaly detection framework for cybersecurity data. Computers & Security, 97, 101941. https : / / doi . org / https : //doi.org/10.1016/j.cose.2020.101941.

31. Smiti, A. (2020). A critical overview of outlier detection methods. Computer Science Review, 38, 100306. https://doi.org/https://doi.org/10.1016/j.cosrev.2020. 100306.

32. Liu, Hongyu; Lang, Bo. Machine learning and deep learning methods for intrusion detection systems: A survey. applied sciences, 2019, 9.20: 4396.

33. K. Park, Y. Song, and Y. G. Cheong, “Classification of attack types for intrusion detection systems using a machine learning algorithm,” in Proceedings of the 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 282–286, Bamberg, Germany, March 2018.

34. Xiang, Ling, et al. Condition monitoring and anomaly detection of wind turbine based on cascaded and bidirectional deep learning networks. Applied Energy, 2022, 305: 117925.

35. Гер В.М. Система моніторингу мережі в IoT інфраструктурі. MS thesis. КПІ ім. Ігоря Сікорського, 2022.

36. Колтаков О. А. "Розробка та дослідження системи моніторингу мережі." (2022).

37. Барабаш О.В. ,"Забезпечення функціональної стійкості інформаційних мереж на основі розробки методу протидії DDoS-атакам." (2018)

38. Оборін О. О. "Методи виявлення аномального трафіку в IoT." (2022).

39. Каганюк О. К., Бортник К. Я., Свиридюк В. В. "Аналіз аномальних станів трафіка комп'ютерної мережі на базі нейромереж." Комп'ютерноінтегровані технології: освіта, наука, виробництво 18 (2015): 83-86.

40. Кльоц Ю.П., Петляк Н.С. "Виявлення аномального трафіку у загальнодоступних комп'ютерних мережах." measuring and computing devices in technological processes 3 (2022): 79-86.

41. Крилов С. О. "Розробка системи аналізу мережевого трафіку." (2019).

42. Zeeshan A. "Network intrusion detection system: A systematic study of machine learning and deep learning approaches." Transactions on Emerging Telecommunications Technologies 32.1 (2021): e4150.

43. Бабкін А. А., and О. В. Кудін. "Огляд нейромережевих моделей систем виявлення вторгнень." Вчені записки Таврійського національного університету імені ВІ Вернадського Серія: Технічні науки 31.70 (2020): 77-82.

44. Довбешко С. В., Толюпа С. В., and Я. В. Шестак. "Застосування методів інтелектуального аналізу даних для побудови систем виявлення атак." Сучасний захист інформації 1.37 (2019): 6-15.

45. Козак, Є. Б. "Принципи впровадження моделей машинного навчання у сфері інтелектуального обслуговування промислового обладнання." Таврійський науковий вісник. Серія: Технічні науки 3 (2021): 19-28.

46. Кравченко, С. М., Гришкун Є. О., Власенко О. В. "Методи класифікації машинного навчання з використанням бібліотеки scikit-learn." Вчені записки Таврійського національного університету імені ВІ Вернадського. Серія: Технічні науки (2020).

47. Коберникова Т., Федчук Т.. "Інформаційна технологія безпечного доступу до ресурсів DNS на базі тренованих моделей ідентифікаії тарфіку." SWorldJournal 21-01 (2023): 80-91.

48. Gavrylenko, S., V. Zozulia. " Дослідження методів виявлення аномалій на етапі попередньої обробки данних." Системи управління, навігації та зв'язку. Збірник наукових праць 1.67 (2022): 52-56.

49. Боднар О.Б. "Виявлення небезпечних входжень у комп'ютерну мережу за допомогою систем виявлення вторгнень та забезпечення захисту такої мережі." (2021).

50. Khadafi, Shah, Yuni Dian Pratiwi, and Enggar Alfianto. "Keamanan Ftp Server Berbasiskan Ids Dan Ips Menggunakan Sistem Operasi Linux Ubuntu." Network Engineering Research Operation 6.1 (2021): 11-24.

Додатки

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор

_____ Юрій ЯРЕМЧУК
"20" вересня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

"Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Network (DQN)"

08-72.МКР.013.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.н., доцент Грицак А.В.

_____

Вінниця – 2024 р.

1. Найменування та область застосування

Програмний засіб для підвищення захисту IoT мереж від атаки, загроз, уразливостей. Область застосування: захист інформаційних ресурсів від несанкціонованого доступу у системах безпеки.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №310 від 17. 09. 2024 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного методу підвищення захисту IoT мереж.

3.2 Призначення: розроблений програмний засіб покращує захист на основі аналізу аномального трафіку.

4. Джерела розробки

4.1. Гайфуліна Д. А., Котенко І. В. Аналіз моделей глибокого навчання задач виявлення мережевих аномалій інтернету речей // Інформаційно-керуючі системи, 2021 № 1, с. 28–37.

4.2. Berkovsky, V. V.; Bessonov, A. S. Аналіз та класифікація методів виявлення вторгнень в інформаційну систему. Системи управління, навігації та зв’язку. Збірник наукових праць, 2017

4.3. Бабкін А. А., and О. В. Кудін. "Огляд нейромережевих моделей систем виявлення вторгнень." Вчені записки Таврійського національного університету імені ВІ Вернадського Серія: Технічні науки 31.70 (2020): 77-82

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

5.1.1 Програмний засіб повинен мати зручний, легкий у використанні інтерфейс користувача;

5.1.2 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.1.3 Програмний засіб повинен виконувати процес автентифікації користувачів у системі.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Бази даних повинні бути налаштовані на автоматичне створення резервних копій;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– процесор – Pentium 1500 МГц і подібні до них;

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

6. Вимоги до програмної документації

6.1 Обов’язкова поетапна інструкція для майбутніх користувачів, наведена у пункті 3.4

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Неможливість отримання доступу незареєстрованих користувачів до інформаційних ресурсів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	20.09.2024	23.09.2024
2	Аналіз предметної області обраної теми	23.09.2024	27.09.2024
3	Апробація отриманих результатів	28.09.2024	01.10.2024
4	Розробка алгоритму роботи	02.10.2024	15.10.2024
5	Написання магістерської роботи на основі розробленої теми	16.10.2024	22.11.2024
6	Розробка економічної частини	23.11.2024	02.12.2024
7	Передзахист магістерської кваліфікаційної роботи	03.12.2024	03.12.2024
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	04.12.2024	09.12.2024
9	Захист магістерської кваліфікаційної Роботи	10.12.2024	10.12.2024

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Савшак А.С.

Додаток Б Лістинг програми Підготовка даних до аналізу

```
# Load network logs from a CSV file
data = pd.read_csv('network_logs.csv')

# Convert the timestamp column to datetime format
data['timestamp'] = pd.to_datetime(data['timestamp'])

# Set the timestamp column as the index for time-series analysis
data.set_index('timestamp', inplace=True)

# Resample the data to aggregate packet counts by minute
data['packet_count'] = data['packets'].resample('1T').sum()

# Display a summary of the preprocessed data
print(data.head())
```

Датасет рукописних цифр MNIST

```
from keras.datasets import mnist
import numpy as np

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.astype('float32') / 255
x_test  = x_test .astype('float32') / 255
x_train = np.reshape(x_train, (len(x_train), 28, 28, 1))
x_test  = np.reshape(x_test,  (len(x_test),  28, 28, 1))
```

Модель автоенкодера

```
import tensorflow as tf
```

```
from tensorflow import keras
import numpy as np

# Load the feature dataset
x = np.loadtxt('features.csv', delimiter=',')
input-dim = x.shape[1]

# Define the autoencoder architecture
autoencoder = keras.Sequential([
   keras.layers.Dense(32, activation='relu',input_shape=(input_dim,)),
   keras.layers.Dense(16, activation='relu'),
   keras.layers.Dense(8,  activation='relu'),
   keras.layers.Dense(16, activation='relu'),
   keras.layers.Dense(32, activation='relu'),
   keras.layers.Dense(input_dim, activation='sigmoid'),
])

# Compile the autoencoder
autoencoder.compile(optemizer='adam', loss='mse')

# Train the model
autoencoder.fit(X, X, epochs=50, batch_size=32, validation_splsd=0.2)

# Perform anomaly detection using the reconstrucrion error
reconstrucrions = autoencoder.predict(X)
mse = np.mean(np.power(X - reconstrucrions, 2), axis=1)

#Set a threshold based on the 95th percentile of reconstruction errors
threshold = np.percentile(mse, 95)
data['anomaly'] = mse > threshold

# Outpyt the number of anomalies detected
print(f"Autoencoder anomalies detected: {sum(data['anomaly'])}")
```

Налаштування агенту DQN

```
import gym
import torch
import torch.nn as nn
import torch.optim as optim
import randomimportnumpy as np
from collections import deque

# Create the CartPole environment
env = gym.make("CartPole-v1")

# Neural network model for approximating Q-values
class DQN(nn.Module) :
   def __init_(self, input_dim, output_dim):
      super (DQN, self).__init__()
      self.fcl = nn.Linear(input_dim, 128)
      self.fc2 = nn.Linear(128, 128)
      self.fc3 = nn.Linear(128, output_dim)

   def forward(self, x):
      x = torch. relu(self. fcl(x))
      x = torch. relu(self. fc2(x))
      return self.fc3(x)

# Hyperparameters
learning_rate = 0.001
gamma = 0.99

epsilon = 1.0
epsilon_min = 0.01
epsilon_decay = 0.995
```

```
batch_size = 64
target_update_freq = 1000
memory_size = 10000
episodes = 1000
# Initialize Q-networks
input_dim = env.observation_space.shape[0]
output_dim = env.action_space.n

policy_net = DQN(input_dim, output_dim)

target_net = DQN(input_dim, output_dim)
target_net.load_state_dict (policy_net.state_dict())
target_net.eval()

optimizer = optim.Adam(policy_net.parameters(), Ir=learning_rate)
memory = deque(maxlen=memory_size)

# Function to choose action using epsilon-greedy policy
def select_action(state, epsilon):
    if random.random() < epsilon:
        return env.action_space.sample() # Explore
    else:
        state = torch.FloatTensor (state) .unsqueeze(0)
        q_values = policy_net(state)
        return torch.argmax(q_values).item() # Exploit

# Function to optimize the model using experience replay
def optimize_model():
    if len(memory) < batch_size:
        return

    batch = random.sample(memory, batch_size)
    state_batch, action_batch, reward_batch, next_state_batch, done_batch = zip(+batch)
    state_batch = torch.FloatTensor(state_batch)
```

```
        action_batch = torch. LongTensor (action_batch) .unsqueeze(1)
        reward_batch = torch. FloatTensor(reward_batch)
        next_state_batch = torch.FloatTensor (next_state_batch)
        done_batch = torch.FloatTensor (done_batch)
```

Процес налаштування DQN

```
# Compute Q-values for current states
q_values = policy_net(state_batch) .gather(1, action_batch) .squeeze()

# Compute target Q-values using the target network
with torch.no_grad():
   max_next_q_values = target_net(next_state_batch) .max(1) [0]
   target_q_values = reward_batch + gamma * max_next_q_values * (1 - done_batch)

loss = nn.MSELoss()(q_values, target_q_values)

optimizer.zero_grad()
loss. backward()
optimizer. step()

# Main training loop
rewards_per_episode = []
steps_done = 0

for episode in range(episodes):
   state = env.reset()
   episode_reward = 0
   done = False

   while not done:
      # Select action
      action = select_action(state, epsilon)
      next_state, reward, done, _ = env.step(action)
```

```
            # Store transition in memory
            memory.append((state, action, reward, next_state, done))

            # Update state
            state = next_state
            episode_reward += reward

        # Optimize model
        optimize_model()

        # Update target network periodically
        if steps_done % target_update_freq == 0:
            target_net. load_state_dict(policy_net.state_dict())

        steps_done += 1

        # Decay epsilon
        epsilon = max(epsilon_min, epsilon_decay + epsilon)

        rewards_per_episode.append(episode_reward)

# Plotting the rewards per episode
import matplotlib.pyplot as plt
plt.plot(rewards_per_episode)
plt.xlabel('Episode')
plt.ylabel('Reward')
plt.title('DQN on CartPole')
plt.show()
```

Реалізація алгоритму K-means

```
import numpy as np
```

```
class Kileans:
    def _init_(self, n_clusters=3, max_iters=100, random_state=None
        ):
        self.n_clusters = n_clusters
        self.max_iters = max_iters
        self.random_state = random_state
        self.centroids = None

    def fit(self, X):
        np. random. seed(self.random_state)

        # Initialize centroids randomly
        idx = np.random.choice(len(X), self.n_clusters, replace=False
            )

        self.centroids = X[idx]

        for _ in range(self.max_iters):
            # Assign points to nearest centroid
            distances = self._calculate_distances(X)
            labels = np.argmin(distances, axis=0)
        # Update centroids

        new_centroids = np.array([
            X[labels == k].mean(axis=0)
            for k in range(self.n_clusters)

        ])

        # Check for convergence
        if np.all(self.centroids == new_centroids):
            break

        self.centroids = new_centroids
```

```
    return labels

def predict(self, X):
    distances = self._calculate_distances(X)
    return np.argmin(distences, axis=0)

def _calculate_distances(self, X):

    # Calculate distances between each point in X and all centroids.

    n_samples = X.shape[0]
    distances = np.zeros((self.n_clusters, n_samples))

    for i in range(n_samples):
        diff = self.centroids - X[i]
```

Алгоритм Isolation Forest

```
from sklearn.ensemble import IsolationForest
import numpy as np

# Load the preprocessed features from a CSV file
X = np.loadtxt('features.csv', delimiter=',')

# Initialize and train the Isolation Forest model
model = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
model. Fit (X)

# Predict anomalies (-1 indicates an anomaly, 1 indicates normal behavior)
predictions = model.predict(x)
anomalies = np.sum(predictions == -1)

print(f"Total anomalies detected: {anomalies}")
```

```
# Add predictions to the original dataset for analysis

data['anomaly'] = predictions
print (data[data[' anomaly'] == -1] .head())
```

Налаштування GNN

```
import torch

!pip install -q torch-scatter~=2.1.0 torch-sparse~=0.6.16 torch-
cluster~=1.6.0 torch-spline-conv~=1.2.1 torch-geometric==2.2.0 -f
https://data.pyg.org/whl/torch-{torch.__version__}.html

torch-manual_seed (0)
torch. cuda.manual_seed (0)
torch. cuda.manual_seed_all(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile

url = 'https: //www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip'
with urlopen(url) as zurl:
   with ZipFile (BytesIO(zurl.read())) as zfile:
      zfile.extractall('.')
```

Імпорт бібліотек для використання GNN

```
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', 100)
import itertools

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from sklearn.metrics import fl_score, classification_report, confusion_matrix
```

```
from torch_geometric.loader import DataLoader
from torch_geometric.data import HeteroData
from torch.nn import functional as F

from torch.optim import Adam

from torch import nn

import torch
```

Імпорт набору даних для використання GNN

```
d.read_csv("CIDDS-001/traffic/OpenStack/CIDDS-801-internal-weekl.csv")
df = df.drop(columns=['Src Pt', 'Dst Pt', 'Flows', 'Tos', 'class', 'attackID', 'attackDescription'])
df['attackType'] = df['attackType'].replace('---', 'benign')
df['Date first seen'] = pd.to_datetime(df['Date first seen'])
df
```

Реалізація гетерогенної GNN

```
BATCH_SIZE = 16

features_host = [f'opsrc_{i}' for i in range(1, 17)] + [f'ipdst_{i}' for i in range(1, 17)]

features_flow = ['daytime', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Duration',

'Packets', 'Bytes',  'ACK', 'PSH', 'RST', 'SYN', 'FIN', 'ICMP', 'IGMP', 'TCP', 'UPD']


def get_connections(ip_map, src_ip, dst_ip):
    src1 = [ip_map[ip] for ip in src_ip]
    src2 = [ip_map[ip] for ip in dst_ip]
    src = np.column_stack((src1,src2)).flatten()
    dst = list(range(len(src_ip)))
    dst = np.column_strack((dst, dst)).flatten()


    return torch.Tensor([src,dst]).int(), torch.Tensor([dst, src]).int()


def create_dataloader(df, subgraph_size=1024):
    data = []
    n_subgraphs = len(df) // subgraph_size
    for i in range(1, n_subgraph+1):
        subgraph=df[(i-1)*subgraph_size:i*subgraph_size]
```

```
        src_ip = subgraph['Src IP Addr'].to_nompy()
        dst_ip = subgraph['Dst IP Addr'].to_nompy()

        ip_map = {ip:index for index, ip in enumerate(np.unique(np.append(src_ip, dst_ip)))}
        host_to_flow, flow_to_host = get_connections(ip_map, src_ip, dst_ip)

        batch = HeteroData()
        batch['host'].x = torch.Tensor(subgraph[features_host].to_numpy()).float()
        batch['flow'].x = torch.Tensor(subgraph[features_host].to_numpy()).float()
        batch['flow'].y = torch.Tensor(subgraph[labels].to_numpy()).float()
        batch['host', 'flow'].edge_index = host_to_flow
        batch['flow', 'host'].edge_index = flow_to_host
        data.append(batch)

    return DataLoader(data, batch_size=BATCH_SIZE)
train_loader = create_dataloader(df_train)
val_loader = create_dataloader(df_val)
test_loader = create_dataloader(df_test)
```

Додаток В. Презентація до дипломної магістерської кваліфікаційної роботи

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кваліфікаційна робота освітнього ступеня магістр
125 «Кібербезпека»
125 «Кібербезпека інформаційних технологій та систем»
на тему:
«Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Network (DQN»

Виконав: ст. гр. КІТС-23м
Савшак А.С.
Керівник: к.т.н., доц. Грицак А.В.

м. Вінниця, 2024

АКТУАЛЬНІСТЬ ТЕМИ

У взаємопов'язаному ландшафті Інтернету речей (IoT) забезпечення надійної безпеки мережі має першочергове значення для захисту конфіденційних даних і підтримки цілісності систем. Розповсюдження пристроїв Інтернету речей відкрило безліч точок входу для кібератак, що змусило організації впровадити комплексні заходи безпеки.

МЕТА РОБОТИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Метою кваліфікаційної роботи є розробка заходів підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net.

Об'єктом кваліфікаційної роботи є засоби підвищення захисту IoT мереж на сучасному етапі розвитку.

Предметом кваліфікаційної роботи є впровадження методів підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Net.

ОСНОВНІ ЗАДАЧІ ДЛЯ ДОСЯГНЕННЯ МЕТИ ДОСЛІДЖЕНЬ :

–Розглянути поняття IoT мережі та проблем безпеки, пов'язаних з даним видом мереж;

–Виділити основні вимоги безпеки, які висуваються IoT мережам;

–Провести огляд методологічних аспектів проблем інформаційної безпеки IoT мереж;

–Навести класифікацію загроз, атак, розголошень, вразливостей, які можуть бути застосовані до IoT мереж;

–Провести розробку алгоритму підвищення захищеності IoT мереж;

–Провести тестування розробленого алгоритму та навести результати його тестування;

–Оцінити економічну доцільність реалізації даного проекту.

ВИМОГИ ДО БЕЗПЕКИ IOT МЕРЕЖ

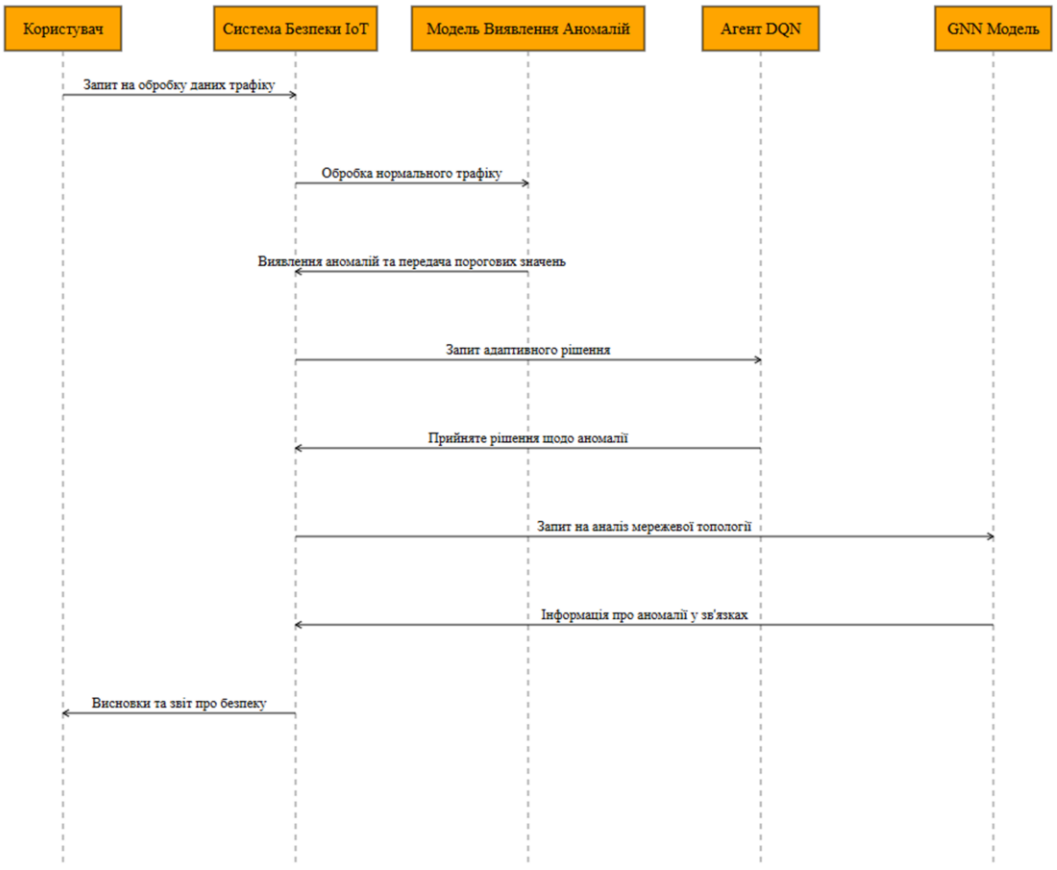
Доступність	Підзвітність	Аудит
Аудит	Аутентифікація	Контроль доступу
Конфіденційність	Цілісність	Безвідмовність

КЛАСИФІКАЦІЯ ЗАГРОЗ ДЛЯ IOT МЕРЕЖ

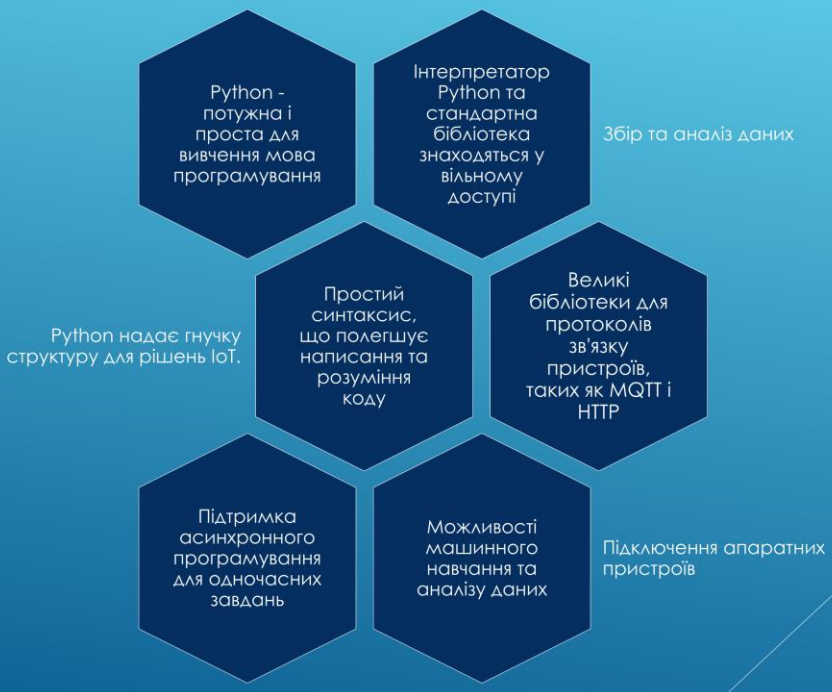
- Атаки на фізичній основі
- Атаки на основі імітації
- Атаки на основі даних
- Атаки на основі спуфінгу
- Атаки на основі доступу
- Атаки на основі приватності
- Сигнальні атаки
- Атаки на основі бічних каналів
- Атаки відмови в обслуговуванні

ПОКРОКОВИЙ АЛГОРИТМ ІНТЕГРОВАНОЇ СИСТЕМИ ВИЯВЛЕННЯ І РЕАГУВАННЯ НА ЗАГРОЗИ В ІОТ-МЕРЕЖАХ

1. Підготовка Даних
2. Модель Виявлення Аномалій (Автоенкодер)
3. Впровадження Адаптивного Порогу
4. Адаптивний Агент на Основі Навчання з Підкріпленням (DQN)
5. Поглиблений Аналіз Трафіку (Кластеризація, Isolation Forest)
6. Інтеграція Фільтрів і Машинного Навчання
7. Графові Нейронні Мережі для Аналізу Топології
8. Оцінка та Оптимізація Системи



АРГУМЕНТУВАННЯ ВИБОРУ МОВИ ПРОГРАМУВАННЯ



РОЗРОБКА АЛГОРИТМУ ПІДВИЩЕННЯ ЗАХИЩЕНОСТІ IOT МЕРЕЖ

- Підготовка даних

```
# Load network logs from a CSV file
data = pd.read_csv('network_logs.csv')

# Convert the timestamp column to datetime format
data['timestamp'] = pd.to_datetime(data['timestamp'])

# Set the timestamp column as the index for time-series analysis
data.set_index('timestamp', inplace=True)

# Resample the data to aggregate packet counts by minute
data['packet_count'] = data['packets'].resample('1T').sum()

# Display a summary of the preprocessed data
print(data.head())
```

МОДЕЛЬ АВТОЕНКОДЕРА

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np

# Load the feature dataset
X = np.loadtxt('features.csv', delimiter=',')
input_dim = X.shape[1]

# Define the autoencoder architecture
autoencoder = keras.Sequential([
    keras.layers.Dense(32, activation='relu', input_shape=(input_dim,)),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(8, activation='relu'),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(input_dim, activation='sigmoid')
])

# Compile the autoencoder
autoencoder.compile(optimizer='adam', loss='mse')

# Train the model
autoencoder.fit(X, X, epochs=50, batch_size=32, validation_split=0.2)

# Perform anomaly detection using the reconstruction error
reconstructions = autoencoder.predict(X)
mse = np.mean(np.power(X - reconstructions, 2), axis=1)

# Set a threshold based on the 95th percentile of reconstruction errors
threshold = np.percentile(mse, 95)
data['anomaly'] = mse > threshold

# Output the number of anomalies detected
print(f"Autoencoder anomalies detected: {sum(data['anomaly'])}")
```

НАЛАШТУВАННЯ АГЕНТУ DQN

```python
import gym
import torch
import torch.nn as nn
import torch.optim as optim
import random
import numpy as np
from collections import deque

# Create the CartPole environment
env = gym.make("CartPole-v1")

# Neural network model for approximating Q-values
class DQN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(DQN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, output_dim)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.fc3(x)

# Hyperparameters
learning_rate = 0.001
gamma = 0.99
epsilon = 1.0
epsilon_min = 0.01
epsilon_decay = 0.995
batch_size = 64
target_update_freq = 1000
memory_size = 10000
episodes = 1000
```

```python
# Initialize Q-networks
input_dim = env.observation_space.shape[0]
output_dim = env.action_space.n
policy_net = DQN(input_dim, output_dim)
target_net = DQN(input_dim, output_dim)
target_net.load_state_dict(policy_net.state_dict())
target_net.eval()

optimizer = optim.Adam(policy_net.parameters(), lr=learning_rate)
memory = deque(maxlen=memory_size)

# Function to choose action using epsilon-greedy policy
def select_action(state, epsilon):
    if random.random() < epsilon:
        return env.action_space.sample()  # Explore
    else:
        state = torch.FloatTensor(state).unsqueeze(0)
        q_values = policy_net(state)
        return torch.argmax(q_values).item()  # Exploit

# Function to optimize the model using experience replay
def optimize_model():
    if len(memory) < batch_size:
        return

    batch = random.sample(memory, batch_size)
    state_batch, action_batch, reward_batch, next_state_batch, done_batch = zip(*batch)

    state_batch = torch.FloatTensor(state_batch)
    action_batch = torch.LongTensor(action_batch).unsqueeze(1)
    reward_batch = torch.FloatTensor(reward_batch)
    next_state_batch = torch.FloatTensor(next_state_batch)
    done_batch = torch.FloatTensor(done_batch)
```

```python
    # Compute Q-values for current states
    q_values = policy_net(state_batch).gather(1, action_batch).squeeze()

    # Compute target Q-values using the target network
    with torch.no_grad():
        max_next_q_values = target_net(next_state_batch).max(1)[0]
        target_q_values = reward_batch + gamma * max_next_q_values * (1 - done_batch)

    loss = nn.MSELoss()(q_values, target_q_values)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

# Main training loop
rewards_per_episode = []
steps_done = 0

for episode in range(episodes):
    state = env.reset()
    episode_reward = 0
    done = False

    while not done:
        # Select action
        action = select_action(state, epsilon)
        next_state, reward, done, _ = env.step(action)

        # Store transition in memory
        memory.append((state, action, reward, next_state, done))

        # Update state
        state = next_state
        episode_reward += reward
```

РЕАЛІЗАЦІЯ АЛГОРИТМУ K-MEANS

```python
import numpy as np

class KMeans:
    def __init__(self, n_clusters=3, max_iters=100, random_state=None
        ):
        self.n_clusters = n_clusters
        self.max_iters = max_iters
        self.random_state = random_state
        self.centroids = None

    def fit(self, X):
        np.random.seed(self.random_state)

        # Initialize centroids randomly
        idx = np.random.choice(len(X), self.n_clusters, replace=False
            )
        self.centroids = X[idx]

        for _ in range(self.max_iters):
            # Assign points to nearest centroid
            distances = self._calculate_distances(X)
            labels = np.argmin(distances, axis=0)
```

```python
            # Update centroids
            new_centroids = np.array([
                X[labels == k].mean(axis=0)
                for k in range(self.n_clusters)
            ])

            # Check for convergence
            if np.all(self.centroids == new_centroids):
                break

            self.centroids = new_centroids

        return labels

    def predict(self, X):
        distances = self._calculate_distances(X)
        return np.argmin(distances, axis=0)

    def _calculate_distances(self, X):
        """
        Calculate distances between each point in X and all centroids.
        """
        n_samples = X.shape[0]
        distances = np.zeros((self.n_clusters, n_samples))

        for i in range(n_samples):
            diff = self.centroids - X[i]
```

АЛГОРИТМ ISOLATION FOREST

```python
from sklearn.ensemble import IsolationForest
import numpy as np

# Load the preprocessed features from a CSV file
X = np.loadtxt('features.csv', delimiter=',')

# Initialize and train the Isolation Forest model
model = IsolationForest(n_estimators=100, contamination=0.05, random_state=42)
model.fit(X)

# Predict anomalies (-1 indicates an anomaly, 1 indicates normal behavior)
predictions = model.predict(X)
anomalies = np.sum(predictions == -1)

print(f"Total anomalies detected: {anomalies}")

# Add predictions to the original dataset for analysis
data['anomaly'] = predictions
print(data[data['anomaly'] == -1].head())
```

НАЛАШТУВАННЯ GNN

```
import torch
!pip install -q torch-scatter~=2.1.0 torch-sparse~=0.6.16 torch-
cluster~=1.6.0 torch-spline-conv~=1.2.1 torch-geometric==2.2.0 -f
https://data.pyg.org/whl/torch-{torch.__version__}.html

torch.manual_seed(0)
torch.cuda.manual_seed(0)
torch.cuda.manual_seed_all(0)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
from io import BytesIO
from urllib.request import urlopen
from zipfile import ZipFile

url = 'https://www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip'
with urlopen(url) as zurl:
    with ZipFile(BytesIO(zurl.read())) as zfile:
        zfile.extractall('.')
```

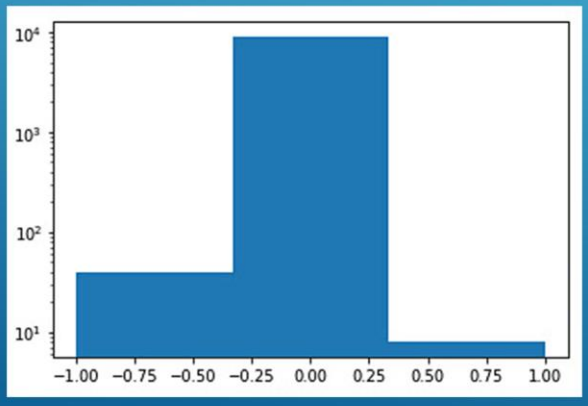
РЕЗУЛЬТАТИ ТЕСТУВАННЯ

- Зібрані дані

	BALANCE	BALANCE_FREQUENCY	...	PRC_FULL_PAYMENT	TENURE
0	40.900749	0.818182	...	0.000000	12
1	3202.467416	0.909091	...	0.222222	12
2	2495.148862	1.000000	...	0.000000	12
3	1666.670542	0.636364	...	0.000000	12
4	817.714335	1.000000	...	0.000000	12

[5 rows x 17 columns]

- Візуалізація даних під час моніторингу мережевого трафіку



- Кількість виявлених аномалій

```
1 n_clusters = len(np.unique(labels))-1
2 anomaly = list(labels).count(-1)
3 print(f'Clusters: {n_clusters}')
4 print(f'Abnormal points: {anomaly}')
```

```
Clusters: 2
Abnormal points: 39
```

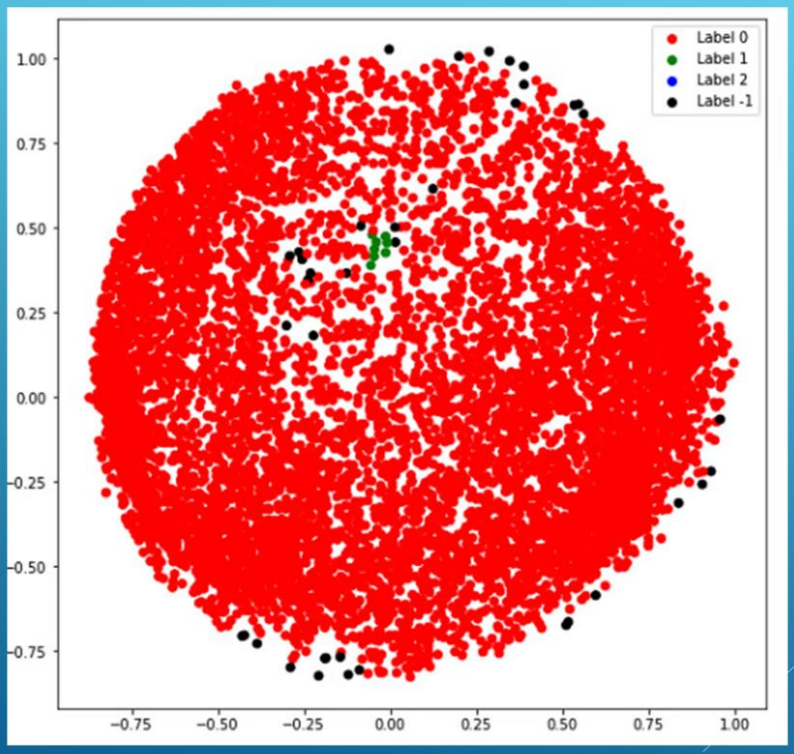
- Аномальні точки

```
1 X_anomaly = X.iloc[np.argwhere(labels==-1).reshape((-1,))]
2 print(X_anomaly.head())
```

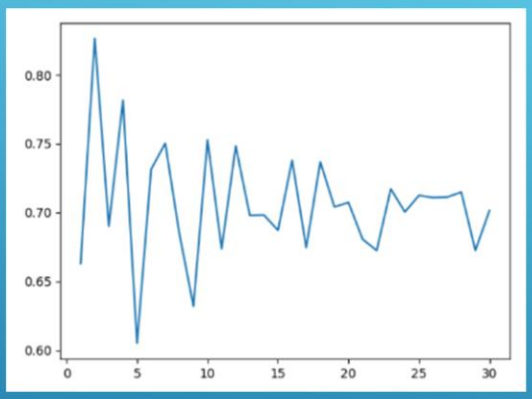
	BALANCE	BALANCE_FREQUENCY	...	PRC_FULL_PAYMENT	TENURE
86	7069.950386	1.0	...	0.000000	12
87	8181.251131	1.0	...	0.000000	12
109	6644.201651	1.0	...	0.083333	12
120	8504.876253	1.0	...	0.000000	12
468	6426.639738	1.0	...	0.000000	12

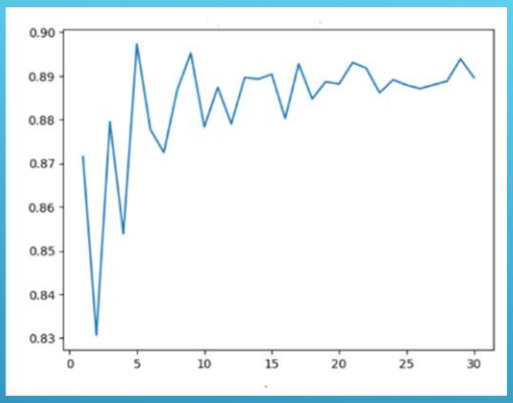
[5 rows x 17 columns]

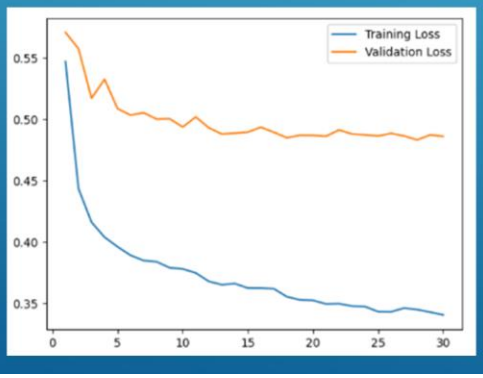
- Діаграма розсіювання кінцевих результатів



ДОСЛІДЖЕННЯ ЕПОХ DQN







ВИСНОВКИ

- Потреба в покращенні захищеності IoT мереж
- Проблеми та вимоги до безпеки IoT
- Створення алгоритму на основі використання DQN
- Покращення захисту за допомогою GNN
- Аналіз отриманих результатів

Додаток Г. Протокол перевірки на антиплагіат

Назва роботи:

Підвищення захищеності IoT мереж на основі аналізу мережевого трафіку і машинного навчання з використанням вдосконаленого алгоритму глибокого виявлення аномалій із застосуванням адаптивного порогу виявлення та алгоритму навчання з підкріпленням Deep Q-Network (DQN)

Тип роботи: магістерська кваліфікаційна робота
Підрозділ Кафедра менеджменту на безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
Гр. КІТС-23м
Керівник доцент Грицак А.В.

Показники звіту подібності

Turnitin	
Оригінальність	78 %
Загальна схожість	22 %

Аналіз звіту подібності (відмітити потрібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Савшак А.С.
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

Особа, відповідальна за перевірку _____ Коваль Н.П.
(підпис) (прізвище, ініціали)

Експерт _____
(за потреби) (підпис) (прізвище, ініціали, посада)