Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту

Спеціальність 125 «Кібербезпека та захист інформації»

Освітня програма «Безпека інформаційних і комунікаційних систем»

«Допущено до захисту»

В.о. зав. кафедрою БІСТ

Марина ЄСІНА

_____

« » 2024 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Аналіз можливостей AI при реалізації Static і Dynamic Honeypot та моделювання механізмів реалізації сценарної хмари поведінкових реакцій»

оцінка « »

Голова ЕК

Олександр ЛЕМЕШКО _____

Керівник: к.т.н., доцент каф. КБІСМіТ

Малахов С. В.

Рецензент: PhD, доцент кафедри інтелектуальних програмних систем і технологій

Родінко М. Ю.

Виконавець: студент групи КБ-61

Михайленко Д. Д.

Харків – 2024

РЕФЕРАТ

Пояснювальна записка містить: 66 сторінок, 25 рисунків, 3 таблиці, 3 додатка, 59 використаних джерел.

Метою роботи є покращення параметрів захисту сучасних інформаційних систем за рахунок інтеграції можливостей штучного інтелекту і алгоритмів активного (Dynamic) HoneyPot та покращення властивостей сценарної хмари поведінкових реакцій HoneyPot.

Об’єкт дослідження – методи мережевого моніторингу та активного захисту інформаційних ресурсів сучасних інформаційних систем (ІС).

Предмет дослідження – процедури моніторингу мережевої активності та синтезу сценарної хмари зворотних реакцій активних HoneyPot.

Основними методами дослідження є аналіз та узагальнення джерел інформації, комп’ютерне моделювання, синтез програмної моделі прототипу сценарної хмари поведінкових реакцій активного вузлового HoneyPot та узагальнення отриманих результатів.

У роботі досліджено питання сучасних ключових викликів сфери інформаційної безпеки корпоративних мереж. Проаналізовано статистику використання штучного інтелекту зловмисниками та адміністраторами безпеки. Розглянуто сучасні стратегії реагування на кіберінциденти. Визначено основні методи класифікації HoneyPot. Наведено сучасні системи мережевих пасток та проаналізовано тенденції використання цих систем. Обґрунтовано недосконаліть технології HoneyPot. Оглянуто доцільність залучення штучного інтелекту (AI) у процеси функціонування HoneyPot. Оглянуто види нейромереж та їх вплив на системи HoneyPot. Розглянуто сучасні моделі мережевих пасток, що використовують AI. Запропоновано схему взаємодії HoneyPot з AI. Змодельовано експериментальну модель HoneyPot з використанням AI та описано середовище функціонування даної моделі. Описано загальний алгоритм навчання нейромережі та описано декілька функцій навчання нейронної мережі.

Проведено тестування на коректність виявлення аномалій мережевого трафіку та описано алгоритм відповіді системи. Розглянуто переваги та недоліки даної моделі.

Результати роботи можуть бути використані в процесах синтезу поведінкових сценарних хмар сучасних HoneyPot. Крім того, робота може бути застосована у задачах дослідження мережевого трафіку на предмет виявлення та блокування підозрілої активності.

Ключові слова: ІНФОРМАЦІЙНА БЕЗПЕКА, МЕРЕЖЕВІ АТАКИ, АКТИВИНИЙ ЗАХИСТ, HONEYPOT, ШТУЧНИЙ ІНТЕЛЕКТ.

ABSTRACT

The explanatory note contains: 66 pages, 25 figures, 3 tables, 3 appendices, 59 references.

The purpose of the work is to improve the protection parameters of modern information systems by integrating the capabilities of artificial intelligence and algorithms of active (Dynamic) HoneyPot and improving the properties of the scenario cloud of behavioral reactions of HoneyPot.

Object of research - methods of network monitoring and active protection of information resources of modern information systems (IS).

The subject of the study is the procedures for monitoring network activity and synthesizing a scenario cloud of feedback reactions of active HoneyPot.

The main research methods are the analysis and generalization of information sources, computer modeling, synthesis of a software model of a prototype scenario cloud of behavioral reactions of an active HoneyPot node and generalization of the results obtained.

The paper investigates the issues of current key challenges in the field of information security of corporate networks. The statistics on the use of artificial intelligence by attackers and security administrators are analyzed. Modern strategies for responding to cyber incidents are considered. The main methods of HoneyPot classification are defined. Modern network trap systems are presented and trends in the use of these systems are analyzed. The imperfection of HoneyPot technology is substantiated. The expediency of involving artificial intelligence (AI) in the processes of HoneyPot functioning is considered. The types of neural networks and their impact on HoneyPot systems are reviewed. Modern models of network traps using AI are considered. The scheme of interaction between HoneyPot and AI is proposed. An experimental model of HoneyPot using AI is modeled and the environment of this model is described. A general algorithm for training a neural network is described and several neural network training functions are described. Testing for the correctness of

detecting network traffic anomalies is carried out and the system response algorithm is described. The advantages and disadvantages of this model are discussed.

The results of the work can be used in the processes of synthesizing behavioral scenario clouds of modern HoneyPots. In addition, the work can be applied to the tasks of studying network traffic to detect and block suspicious activity.

Keywords: INFORMATION SECURITY, NETWORK ATTACKS, ACTIVE DEFENSE, HONEYPOT, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

ІБ	–	Інформаційна безпека
ІКС	-	Інформаційно-комунікаційна система
ІС	–	Інформаційна система
ОС	–	Операційна система
ПЕОМ	–	Персональна електронно обчислювальна машина
ПЗ	–	Програмне забезпечення
AI	–	Artificial Intelligence, штучний інтелект
API	–	Application Programming Interface, прикладний програмний інтерфейс
CNN	–	Convolutional Neural Networks
DDoS	–	Distributed Denial of Service
DL	–	Deep Learning, глибоке навчання
DNS	–	Domain Name Service
EDR	–	Endpoint Detection and Response
FID	–	Fréchet Inception Distance
FNN	–	Feedforward Neural Networks
FTP	–	File Transfer Protocol
FW	–	Firewall, брандмауер
GAN	–	Generative Adversarial Networks
HTTP	–	Hypertext Transfer Protocol
IDS	–	Intrusion Detection System
IoT	–	Internet of Things, інтернет речей
IP	–	Internet Protocol
IPS	–	Intrusion Prevention System
JSON	-	JavaScript Object Notation
LLM	–	Large Language Model

MAC	–	Medium Access Control;
MTU	–	Maximum Transmission Unit;
OSINT	–	Open source intelligence, Розвідка на основі відкритих джерел
RBM	–	Restricted Boltzmann Machines
RNN	–	Recurrent Neural Networks
SIEM	–	Security Information and Event Management
SMB	–	Server Message Block;
SOAR	–	Security Orchestration, Automation, and Response, Оркестрація, автоматизація й реагування
SSH	–	Secure Shell
XDR	–	eXtended ednpoint Detection and Response

ВСТУП

Питання кібербезпеки набуває все більшого значення через стрімкий розвиток цифрових технологій та зростаючу залежність суспільства від інформаційних систем. Така ситуація зумовлює необхідність розробки нових методів та підходів, здатних протистояти новим типам атак.

Серед таких технологій - системи Honeypot. Їх використання дозволяє не лише виявляти атаки, але й збирати унікальні дані про методи та інструменти, які використовують зловмисники. Однак класичні підходи до побудови Honeypot наділені рядом викликів, що не дає ефективно застосовувати дані методи захисту у системах захисту інформації. У динамічному середовищі такі системи потребують інтелектуальних механізмів адаптації до мінливих моделей атак, задля забезпечення ефективного функціонування. Удосконалення систем Honeypot відкриває абсолютно нові горизонти завдяки методам штучного інтелекту та машинного навчання.

Ця робота присвячена вивченню потенціалу використання штучного інтелекту (*AI*) для реалізації, як статичних, так і динамічних систем захисту на базі технології Honeypot. Проведені дослідження і імітаційне моделювання процесів *«атака-захист»* з використанням парадигми *AI HoneyPot,* продемонстрували роботу синтетичних (тобто, отриманих с залученням AI) сценарних хмар при реалізації різних поведінкових реакцій емульованої системи активного мережевого захисту. Узагальнення існуючих проблем у сфері створення й застосування засобів мережевої безпеки, дозволяє окреслити актуальні виклики, але також формує основу для подальшої реалізації запропонованих/отриманих рішень. Проведенні дослідження мають гарні перспективи їх подальшого продовження, оскільки вдало реалізують новітні підходи до проблематики систем активного мережевого захисту, де існують певні труднощі і проблеми.

1. СУЧАСНИЙ СТАН ПРОБЛЕМАТИКИ ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ ТА РЕАКЦІЙ НА ІНЦИДЕНТИ

1.1 Огляд сучасного стану та ключових викликів у сфері ІБ

ІБ перебуває в стані постійного розвитку та адаптації до нових викликів. За 2024 рік кіберінциденти(інциденти) стали більш складними та небезпечними, оскільки зловмисники використовують все більш витончені тактики, техніки та інструменти задля досягнення своєї мети. Компанія Microsoft відстежує понад 78 трильйонів сигналів небезпеки на день [1], що дозволяє робити висновки про постійну активність хакерів при атаці на сучасні ІС.

Компанія Google звітує про продовження тенденції постійної еволюції ландшафту кіберзагроз [2]. Ключовим занепокоєнням є зростаюче використання вразливостей «нульового дня», коли зловмисники експлуатують недоліки ПЗ до того, як розробникам ПЗ стає відомо про наявність вразливостей. Ця тенденція в поєднанні зі зростаючою витонченістю методів атак становить значний ризик для організацій будь-якого розміру.

Ще однією серйозною загрозою, про яку йдеться у звіті, є націленість на периферійні пристрої, такі як пристрої та датчики IoT. Ці пристрої, часто зі слабкими заходами безпеки, можуть слугувати точками входу для зловмисників, які отримують доступ до мереж.

Однією з ключових проблемою забезпечення якісного впровадження механізмів безпеки є брак досвіду інших компаній у процесах реагування на інциденти та спричинених наслідків. Із точки зору компаній, що стали ціллю для хакерів, розголошення інформації про деталі атаки та її наслідки може завдати шкоди репутації компанії через недостатній рівень захищеності. Це може відвернути потенційних клієнтів, розкрити деталі внутрішнього функціонування цільової ІКС (інформаційно-комунікаційної системи), а отже і стимулювання конкурентів до переймання практик компанії. Комплекс таких проблем

призводить до значної втрати прибутку. У таких умовах компаніям легше мінімізувати втрати шляхом приховування факту зламу, грошовою виплатою хакерам.

Нові загрози з'являються постійно, і все більше зловмисників використовують штучний інтелект для проведення складних атак. Кількість запитів «AI phishing» за останні 4 роки зображена на рисунку 1.1(за даними Google Trends [3]). Зловмисники використовують штучний інтелект для створення більш персоналізованих та переконливих підробок(Deep Fake). Такі інструменти генерують результат, що дуже схожий на реальні ресурси мережі, а тому є важкими для виявлення.

Рисунок 1.1 – Статистика кількості запитів AI phishing

Також, зловмисники активно розробляють AI інструменти для аналізу даних, що отримані у ході розвідки своєї потенційної цілі. Такі аналізатори можуть у напівавтоматичному режимі збирати дані про ціль з мережі(OSINT), проводити сканування портів і одразу виявляти слабкі місця у інфраструктурі компанії, виявляти інструменти захисту.

Для боротьби з такими загрозами, компанії впроваджують AI у існуючі інструменти:

- NGFW- Next-generation firewalls [4] – Інтеграція AI у FW;
- Sentinel AI [5] – інстремент виявлення Deep Fakes;
- Vectra AI [6] – XDR платформа, що використовує AI.

Також, активно досліджується впровадження AI у системи IDS/IPS [7-9]. Процес впровадження AI у інструменти виявлення та запобігання вторгнень є доволі зрозумілим, оскільки дозволяє покращити результати евристичного аналізу. Такі інструменти покращують процес виявлення нових типів атак, та автоматизації протидії вже існуючим. Така тенденція має ряд переваг, серед яких:

- Автоматизація процесів. Використання AI дозволяє автоматизувати аналіз журналів, моніторинг трафіку та виявлення загроз, що знижує навантаження на спеціалістів з безпеки [10];
- Прогнозування загроз. AI дозволяє прогнозувати можливі загрози, аналізуючи великі обсяги даних та виявляючи нові патерни поведінки [11];
- Швидкість реагування. Використання AI значно скорочує час реагування на інциденти, дозволяючи автоматизувати процеси реагування, що особливо важливо у випадках складних атак. Також швидкість реагування важлива для атак, що видозмінюються у процесі вторгнення.

Попри очевидні переваги AI, є низка викликів, з якими стикаються розробники та дослідники [12-13]:

- Висока складність моделей. Створення та навчання AI-моделей, особливо у сфері DL, вимагає великих обсягів даних і обчислювальних ресурсів. Також потрібно враховувати, що для ефективного навчання моделей потрібні не тільки дані, але й експертні знання для правильної інтерпретації результатів;
- Помилкові спрацьовування. Навіть найкращі AI-моделі можуть іноді давати помилкові тривоги, що може спричинити хибну поведінку, а отже і некоректні дані для аналізу. Це створює необхідність постійно перевіряти коректність результатів;
- Проблеми конфіденційності. Використання великих обсягів даних для навчання AI-платформ може створювати ризики для

конфіденційності. У дослідженні [14] підкреслюється, що важливо знаходити баланс між ефективністю систем безпеки та забезпеченням конфіденційності даних користувачів;

- Якість навчальних даних є критичним фактором успіху AI-моделей. AI-моделі повинні мати доступ до великої кількості якісних даних для ефективного навчання. Моделі, навчені на неякісних або застарілих даних, можуть бути менш ефективними та вразливими до нових типів атак.

1.2 Узагальнення стратегій реакції на інциденти безпеки

Ще одна проблема зростання кількості та збитків від інцидентів пов`язана із недосконалістю стратегії реактивного реагування на атаки, а не інструментів для захисту інформації. Така стратегія передбачає, що будь-які дії для забезпечення захисту впроваджуються лише після атаки на ІС, а не під час атаки. Окрім того, невдалі спроби зловмисника, сприймаються як вдала реакція на дії, спрямовані на нанесення шкоди ІС, проте такі дії можуть бути лише тестуванням нових методів та інструментів зламу. Якщо такі інструменти не дають очікуваного результату, хакери модифікують їх, а система ІБ залишається статичною. Таким чином, із плином часу, інструменти та методики будуть модифіковані до такого рівня, що хакери винайдуть новий спосіб зламу систему, що ставить під загрозу, як конкретну цільову ІС, де вони умовно кажучи «практикувались», так і інші ІС, що використовують подібні механізми захисту. Задля подолання цих викликів, було створено проактивну стратегію реагування на інциденти ІБ, впровадженням якої активно займаються провідні фахівці й компанії світу [15-17].

Проактивна стратегія ІБ є підходом, орієнтованим на запобігання інцидентів до того, як вони відбудуться, а не на реагування на загрози, що вже виникли. На відміну від реактивної стратегії, проактивний підхід передбачає постійний моніторинг, аналіз і поліпшення системи безпеки з метою мінімізації ризиків. Це охоплює передбачення потенційних загроз, їх запобігання та побудову більш захищеної інфраструктури.

Основними елементами проактивної стратегії є предиктивний аналіз і керування ризиками. Системи предиктивного аналізу використовуються для ідентифікації потенційних вразливостей і загроз. Це дає змогу заздалегідь виявляти можливі вразливості, що допомагає запобігти атакам ще на стадії їх підготовки. Управління ризиками передбачає проведення регулярних аудитів безпеки, тестування на проникнення і моделювання можливих сценаріїв атак, що дає змогу ефективно розподіляти ресурси та усувати слабкі місця до того, як вони будуть використані зловмисниками.

Проактивна стратегія також передбачає впровадження та використання інноваційних технологій, як-от фреймворки виявлення аномалій і *SOAR(англ. Security Orchestration, Automation, and Response, Оркестрація, автоматизація й реагування)*. Ці інструменти допомагають активно збирати інформацію про їхні методи і тактики. Застосування засобів глибокої аналітики та моніторингу дає змогу оперативно виявляти загрози, які можуть залишатися непоміченими традиційними методами виявлення.

Ще одним важливим аспектом проактивної стратегії є постійне навчання та адаптація. Системи безпеки повинні регулярно оновлюватися й адаптуватися до мінливого середовища загроз, щоб підтримувати свою ефективність. Це охоплює навчання співробітників правилам інформаційної гігієни, проведення тренінгів з реагування на інциденти та використання AI для автоматичного оновлення профілів загроз.

Висновки за розділом:

1) Аналіз поточного ландшафту інформаційних загроз свідчить про активне використання AI як зловмисниками, так і фахівцями ІБ;
2) Одною з ключових проблем ІБ є проблема недосконалості реактивної стратегії реагування на інциденти;
3) Впровадження проактивної системи дозволяє своєчасно реагувати на інциденти, збирати якісну аналітику, надає додаткові можливості протидії спробам зламу.

2. ТЕХНОЛОГІЯ МЕРЕЖЕВИХ ПАСТОК ЯК КЛЮЧОВА ТЕХНОЛОГІЯ ПРОАКТИВНОЇ СТРАТЕГІЇ ЗАХИСТУ

2.1 Основні пластивості та переваги технології мережевих пасток

Одна з ключових технологій для впровадження проактивного підходу є технологія мережевих пасток – HoneyPot/HoneyNet, що вперше була висунута як концепція наприкінці 90-х років для залучення зловмисників і збору даних про їхні методи [18]. В основі цієї технології лежить створення спеціальних пасток, які імітують вразливі ресурси, тим самим привертаючи увагу атакуючих. Це дозволяє не лише захистити реальні системи, а й отримати цінну інформацію про атаки для вдосконалення заходів кібербезпеки. Існує кілька типів і реалізацій Honeypot (табл. 2.1-2.2):

Таблиця 2.1 – Типи HoneyPot за рівнем взаємодії

Тип HoneyPot	**Опис**	**Приклади**
Low-interaction Honeypot	Імітують базові служби та протоколи, надаючи мінімальний рівень взаємодії із зловмисником. Вони легкі в налаштуванні та управлінні, не потребують значних ресурсів, але мають обмежені можливості щодо збору даних.	*Honeyd*
Medium-interaction Honeypot	Забезпечують більший рівень взаємодії, дозволяючи зловмисникам виконувати обмежені дії в середовищі, яке частково імітує справжні системи. Це допомагає збирати більше інформації про атаки, не наражаючи на небезпеку основну інфраструктуру	*Kippo*
High-interaction Honeypot	Це повноцінні системи або віртуальні машини, що повністю імітують справжнє середовище, дозволяючи зловмисникам виконувати будь-які дії. Вони забезпечують максимальну кількість інформації про методи атак, але вимагають значних ресурсів і ретельного контролю	*Cowrie*

Конкретні рішення та їх особливості

- Honeyd [19] — легкий у використанні Honeypot, що дозволяє імітувати різні операційні системи та мережеві сервіси. Основна його

особливість — можливість створення великої кількості віртуальних хостів з різними IP-адресами. Це дає змогу виявляти спроби сканування та відстежувати активність зловмисників.

Основні переваги: простота, гнучкість конфігурації, мінімальні витрати.

Недоліки: обмежений функціонал для виявлення складних атак.

- Kippo [20] — SSH Honeypot, що імітує слабкозахищений сервер Linux. Його особливістю є детальний логінг команд, які вводяться зловмисниками, що дозволяє вивчати їхні дії. Kippo також зберігає введені паролі та аналізує способи зламу.

  Перевага: висока деталізація зібраних даних.

  Недоліки: обмежена можливість взаємодії з сучасними атаками на SSH.

- Cowrie [21] — розширений варіант Kippo, що підтримує більше команд та сценаріїв. Він дозволяє імітувати повноцінну командну оболонку і навіть роботу з файлами, що значно підвищує реалістичність симуляції. Особливість Cowrie — інтеграція з іншими інструментами аналізу загроз, такими як ELK Stack.

  Переваги: гнучкість, підтримка інтеграцій із іншими застосунками.

  Недоліки: потребує більше ресурсів на налаштування та підтримку.

- Dionaea [22] — Honeypot, розроблений для виявлення та аналізу шкідливого ПЗ. Він підтримує широкий спектр протоколів, таких як SMB(Server Message Block), HTTP(Hypertext Transfer Protocol), FTP(File Transfer Protocol), і може зберігати зразки шкідливих програм для подальшого аналізу.

  Переваги: ефективний збір зразків шкідливого ПЗ, простота розгортання.

  Недоліки: обмежені можливості для складних сценаріїв атаки.

Таблиця 2.2 - Типи HoneyPot за типом середовища

Тип HoneyPot	Опис	Приклади
Програмні Honeypot	Використовують спеціальні програми, що імітують роботу мережевих протоколів та сервісів. Вони зручні для тестування та експериментів, але можуть бути обмежені в реалістичності симуляції.	*Dionaea, Honeypot*
Апаратні Honeypot	Застосовуються як фізичні пристрої, що працюють у реальних мережах. Вони забезпечують високий рівень реалістичності, але потребують більших витрат на впровадження та обслуговування	фізичний сервер або спеціалізований пристрій

Концепція мережевих пасток, має ряд переваг, серед яких:

- Можливість отримати детальну інформацію про методи атак, які використовуються зловмисниками;
- Відволікання уваги атакуючих від реальних ресурсів;
- Аналіз зібраних даних для вдосконалення заходів безпеки та прогнозування нових загроз;
- Використання Honeypot для тестування та навчання фахівців з кібербезпеки.

Але, у той же час, така технологія має декілька обмежень:

- Ризик того, що досвідчені зловмисники зможуть розпізнати Honeypot і уникнути пастки;
- Необхідність постійного моніторингу та оновлення для запобігання використанню Honeypot хакером;
- Значні ресурси, потрібні для розгортання та підтримки високого рівня інтерактивності;
- Обмежена здатність імітувати реальні ресурси в разі недостатньо гнучких налаштувань.

2.2 Особливості концепції Cyber Deception

Така ідея отримала розвиток у виді розширення концепції HoneyPot до організації окремих кластерів таких серверів(HoneyNet), спеціально створених та розголошених даних(HoneyToken), окремих портів у вузлах IC(HoneyPort). Таким чином, майже будь-який функціональний(сервер, порт, інтерфейс) або організаційний(токен доступу, логін та пароль, файл, директорія або шлях) компонент можуть бути використані як приманка для хакера. Загальна концепція введення хакера в оману отримала назву Cyber Deception [23-28].

Така концепція активно використовується останні роки, підтвердженням цього слугують дані Drupal. Де наведено статистику(Рис. 2.1) використання модуля HoneyPot з 2012 року [29]

Рисунок 2.1 - Статистика використання модуля HoneyPot

Наведені дані чітко демонструють збільшення попиту на технології мережевих пасток в більше, ніж 100 разів за останні 12 років. Такий попит обумовлюється вищезгаданими перевагами технології HoneyPot. Проте, незважаючи на лінію тренду, бачимо, що за останні 3 роки попит на HoneyPot не

збільшується. У той же час, кількість Drupal проектів залишається високою(понад 700 тисяч проектів, що використовують Drupal Core [30]). Цей факт свідчить про неготовність розробників застосовувати технологію мережевих пасток у нових проектах.

2.3 Визначення недосконалості діючих реалізацій технології HoneyPot.

Збільшення проектів, де використовуються технології CyberDeception, зокрема HoneyPot, безумовно привернуло увагу хакерів. Це є негативним наслідком активного використання мережевих пасток, оскільки все більше зловмисників знають про наявність і властивості такої технології і все частіше замислюються про можливість її застосування у цільових ІКС, та можливі контрміри для протидії. Відносна простота концепції технологій Cyber Deception дозволяє навчитися відокремлювати виставлені пастки від реальних вузлів, що є доцільними для атак зловмисників. Також, на погіршення наслідків використання технологій CyberDeception впливає тривале застосування цих технологій без суттєвих змін у діючої парадигмі її поведінкових алгоритмів. В цьому випадку, із точки зору хакера це легкодоступний сервер, набір даних, порт, котрий ніяк або майже ніяк не задіяні у процесах функціонування цільової ІКС.

Тобто сервер, на який не потрапити звичайним шляхом, порт, на який не надходять пакети даних, токени доступу, які не використовуються тощо. Такі характерні ознаки виявляються на етапі пасивної та активної рекогносцировки цільової системи (Рис. 2.2) із залученням інструментів для перехоплення мережевого трафіку на кшталт WireShark, аналізаторів портів, прикладами яких є ZenMap, Nmap тощо [31].

Недостатня або взагалі відсутня залученість окремих вузлів або компонентів у процесах функціонування ІС є характерним маркером технологій Cyber Deception. У свою чергу наявність таких маркерів слугує підказкою для хакерів, як оминати мережеві пастки. На сьогодні існує ряд аналізаторів, що можуть відрізнити HoneyPot [32-34]. Такі аналізатори можуть чітко розпізнати HoneyPot за наведеними вище ознаками, а також наявністю критичних вразливостей, що також є частиною концепеції технологій Cyber Deception.

Відповідно, що у таких умовах, використання проактивного підходу реагування на інциденти не дає очікуваних результатів.

Висновки за розділом:

1) Технологія HoneyPot здатна збирати важливу інформацію про спроби вторгнення, надавати відповіді на дії зловмисника, але може бути скомпрометована у разі некоректної конфігурації та/або підтримки;
2) Загальна концепція введення в оману під час спроби зламу системи отримала назву Cyber Deception. Ця концепція є розширенням розуміння HoneyPot;
3) Останні роки спостерігається тенденція збільшення зацікавленності розробників до концепції Cyber Deception;
4) Останні 3 роки, попит на HoneyPot залишається на одному рівні, що свідчить про обмежену зацікавленість нових проектів у концепції мережевих пасток через недосконалість HoneyPot.

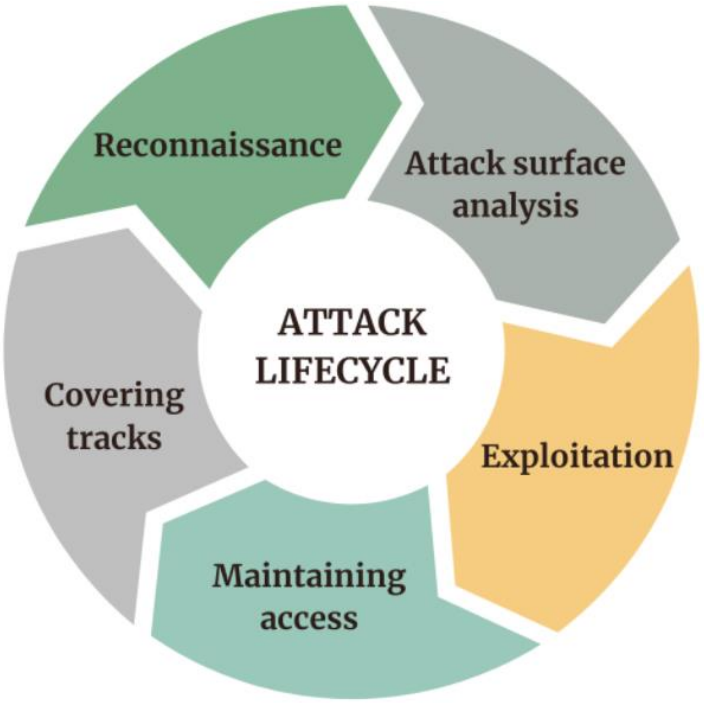

Рисунок 2.2 - Життєвий цикл проведення атаки [34]

3. НАПРЯМИ ПОКРАЩЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ HONEYPOT ШЛЯХОМ ЗАЛУЧЕННЯ AI

3.1 Обґрунтування доцільеості залучення можливостей AI у HoneyPot

Подальшим розвитком можливостей технологій Cyber Deception та вирішенням вищезгаданих недосконалостей HoneyPot (п.п.2.3), є залучення можливостей штучного інтелекту у процеси функціонування мережевих пасток. Дане питання активно розглядається у ряді наукових робіт [35-38], у висновках яких дослідники одностайно схиляються до доцільності використання AI у HoneyPot. При реалізації вичерпної стратегії реагування системи на рівнях планування та впровадження, такий підхід дозволяє швидко адаптувати конфігурацію сервера-пастки, до дій зловмисника [39]. Як наслідок – впровадження більш якісної моделі вузла-пастки, що виглядає як реальний вузол ІС та може збирати корисну аналітику мережевої активності. Також, така система може самостійно аналізувати шляхи проникнення та методи протидії та може надавати більш якісні реакції на дію зловмисника. Ще одним наслідком впровадження технологій AI є синтез профілів «нормальної» та «підозрілої» активності [40-41] для конкретної ІС без залучення адміністраторів безпеки системи, що будуть синтезовані самою штучною мережею у процесі роботи. Такі профілі можуть слугувати вихідним «матеріалом» для формування організаційних і технічних заходів захисту для кожної окремої ІКС, оскільки адміністратори з ІБ матимуть чітке уявлення, які дії варто обмежувати.

Базуючись на отриманих даних, HoneyPot може використати механізми евристичного аналізу задля прогнозування подальших ймовірних атак. Ця можливість є ключовою для вивчення нових загроз та дозволяє вивести здатність HoneyPot аналізувати трафік на принципово новий рівень [42]. У той же час, ця можливість була б недосяжної без залучення технологій AI для збору та аналізу

інформації. Таке прогнозування може бути використано іншими системами захисту, накшталт IDS/IPS, FW тощо.

HoneyPot може використати AI для синтезу відповідної реакції двома принципово різними шляхами:

- Експертна система(expert system). Штучний інтелект виступає у такій системі у ролі фахівця, що синтезує рішення про реактивні дії HoneyPot, базуючись на можливостях конкретного HoneyPot. Для коректної відповіді такій системі потрібна якісно спроектована та навчена модель нейромережі, що враховує нюанси проведення атаки;
- Прецедентна система(case-based system). Штучний інтелект зіставляє конкретну атаку із попередніми сценаріями, для яких вже є відповідь у підготовленій базі даних прецедентів. У разі відсутності відповідного прецеденту, система вносить випадок у базу даних, а рішення має внести адміністратор безпеки. Для коректної роботи такої системи, необхідна якісна база даних. Розмір і конкретність цієї бази є ключовим фактором успішності для всієї системи.

Ці 2 підходи відрізняються складністю нейромережі, яку треба впровадити у систему, а отже і ресурсами, що потрібні на розробку і підтримку. У той час, як експертна система, теоретично, є більш гнучкою і самостійною у прийнятті рішень, прецедентна система є більш економною.

3.2 Типи нейромереж, що можуть бути використані для HoneyPot

Ключову роль у ефективній взаємодії AI та HoneyPot відіграє коректний вибір типу нейромережі, що буде використано. Розуміння сутності існуючих різновидів нейромереж дозволяє чітко визначити переваги та недоліки кожного з їх типів, розробити детальний план впровадження та експлуатації системи мережевих пасток, визначення клювих метрик оцінювання системи.

Feedforward Neural Networks, нейронні мережі прямого поширення [43] є найпростішим типом нейромереж, де дані передаються лише вперед, від вхідного шару до вихідного. Дана архітектура є найпростішою, а її застосування

досить обмежено. У HoneyPot може застосовуватись для класифікації трафіку, вибору найбільш ефективної реакції на дії зловмисника. Така нейромережа обмежена у можливостях синтезу нового рішення, або модифікації вже існуючих, тобто таку мережу варто застосовувати лише як засіб реалізації прецедентної системи.

Convolutional Neural Networks [44]*, Згорткові нейронні мережі* - призначені для обробки даних із сітковою топологією, таких як зображення. Такі нейромережі здатні визначити важливі ознаки із вхідних даних, що дає змогу виявляти складні шаблони у зображеннях. CNN використовуються в інструментах, таких як TensorFlow і PyTorch. CNN можуть аналізувати метадані трафіку і виявляти аномальні патерни, характерні для атак, шляхом перетворення мережевого трафіку на графічні репрезентації. Таким чином, нейромережа може сформувати «зображення» нормального трафіку і вказувати на аномалії, базуючись на порівнянні «зображень» поточного трафіку та нормального. Таким чином, CNN може бути використано для аналізу мережевого трафіку та виявлення шкідливого програмного забезпечення. Ще одним завданням CNN може стати класифікація аномалій, що дає змогу визначити тип атаки, а також спрогнозувати можливі реакції HoneyPot, спираючись на «зображення» нормального трафіку у системі. CNN можуть бути імплементовані у Honeypot через модуль аналізу трафіку. Дані трафіку перетворюються на візуальну форму, а далі проходять через нейромережу для класифікації загрози і прогнозування реакції.

Restricted Boltzmann Machines (обмежена машина Больцмана або обмежена модель Шеррінгтона-Кіркпатріка із зовнішнім полем або обмежена стохастична модель Ізинга-Ленца-Літтла) [45] широко застосовуються для зменшення розмірності даних, виконання класифікаційних задач, розробки систем спільної фільтрації, виявлення прихованих особливостей, тематичного аналізу текстів, а також в галузях імунології та квантової фізики для моделювання складних багатокомпонентних систем. У контексті тематики Honeypot їх можна застосовувати для пошуку унікальних шаблонів, що вказують

на присутність аномалій або загроз, а також для ефективного стиснення даних мережевої інформації для подальшого аналізу.

Recurrent Neural Networks, Рекурентні нейронні мережі [46] були запропоновані в 1980-х роках Девідом Румелхартом і його колегами. Ця мережа обробляє послідовні дані, такі як часові ряди, використовуючи внутрішні стани для запам'ятовування попередніх даних. RNN часто використовуються в таких інструментах, як Keras і PyTorch. Ця мережа здатна передбачити атаку, зіставляючи поточну послідовність трафіку з збереженими даними. Маючи дані про атаку, система теоритично здатна ініціювати автоматичні заходи відповіді. Таким чином, RNN може виявляти аномалії у поведінці користувача та прогнозувати подальші дії, що є ключовими елементами для впровадження прецедентної системи. RNN будуть найкращим рішенням у разі потреби прогнозування атак і вжиття превентивних заходів.

Transformers, трансформ ери [47] була запропонована компанією Google у 2017 році. Дана модель є основою сучасних моделей, таких як BERT і GPT, і орієнтовані на обробку великих обсягів послідовностей даних. У Honeypot трансформери можна застосовувати для семантичного аналізу команд, введених зловмисником, або тексту запитів у взаємодії із Honeypot та синтезом найбільш оптимальної стратегії реагування. Дана модель здатна до визначення довготривалих та складних шаблонів дій зловмисника, що може бути використано для поліпшення процесу оптимізації процесу оптимізації поточних параметрів захисту сучасних ІС.

Generative Adversarial Networks [48] були запропоновані Іеном Гудфеллоу і його колегами в 2014 році. Вони складаються з генератора і дискримінатора, які спільно створюють дані, що наближені до тих, які були закладені у їх набір даних для навчання. У HoneyPot це може бути використано 2 принципово різними шляхами:

1) Нейромережа здатна формувати наближений до нормального трафік, задля створення більш переконливої імітації через неможливість відрізнити

пастку за ознакою недостатньої залученості у систему, що стає додатковою перепоною для хакера у процесі виявлення.

2) GAN симулює різні сценарії атак і реакції системи для тестування систем безпеки і визначення найбільш ефективні заходи захисту, навчаючи системи адаптуватися до нових видів загроз.

3.3 Вдосконалення сценарної хмари взаємодії Dynamic AI Honeypot

Важливим аспектом роботи Honeypot систем є її реакція на дії хакера, що спрямована на введення його в оману та збір даних про його поведінку. При цьому, навіть без використання AI, пастки здатні демонструвати широкий спектр можливих взаємодій. Взаємодії HoneyPot без участі штучного інтелекту зображено на рисунку 3.1 [49]:

1) Обмеження швидкості передачі трафіку
   a) Обмеження MTU
   b) Розрив з’єднання
   c) Обмеження кількості пакетів на одиницю часу
   d) Затримка часу відповіді
2) Маскування вузлів мережі
   a) Зміна MAC-адреси
   b) Зміна IP-адреси
3) Обмеження на завантаження файлів
   a) Обмеження розміру завантажувальних файлів
   b) Обмеження прав доступу до файлів або директорій

Зазначений перелік не є вичерпним, оскільки реакції HoneyPot залежать від реалізації системи. На етапі конфігурації HoneyPot, адміністратор безпеки визначає сценарну хмару взаємодії, що може бути описана наступним чином:

- Дія 1 - файл сценарію 1;
- Дія 2 - файл сценарію 2;

  …
- Дія N - файл сценарію N.

Тут, сценарна хмара взаємодії представлена у вигляді Map об’єкту, в якому Дія - є конкретною дією зловмисника, відповіддю на яку має бути реакція, що описана у відповідному файлі сценарію. Таким чином, ефективність функціонування HoneyPot залежить від кількості дій, що було визначено адміністратором безпеки, кількості файлів сценарію, що можуть бути задіяні, якістю відповідних реакцій, аби бути доцільними у конкретній ситуації.

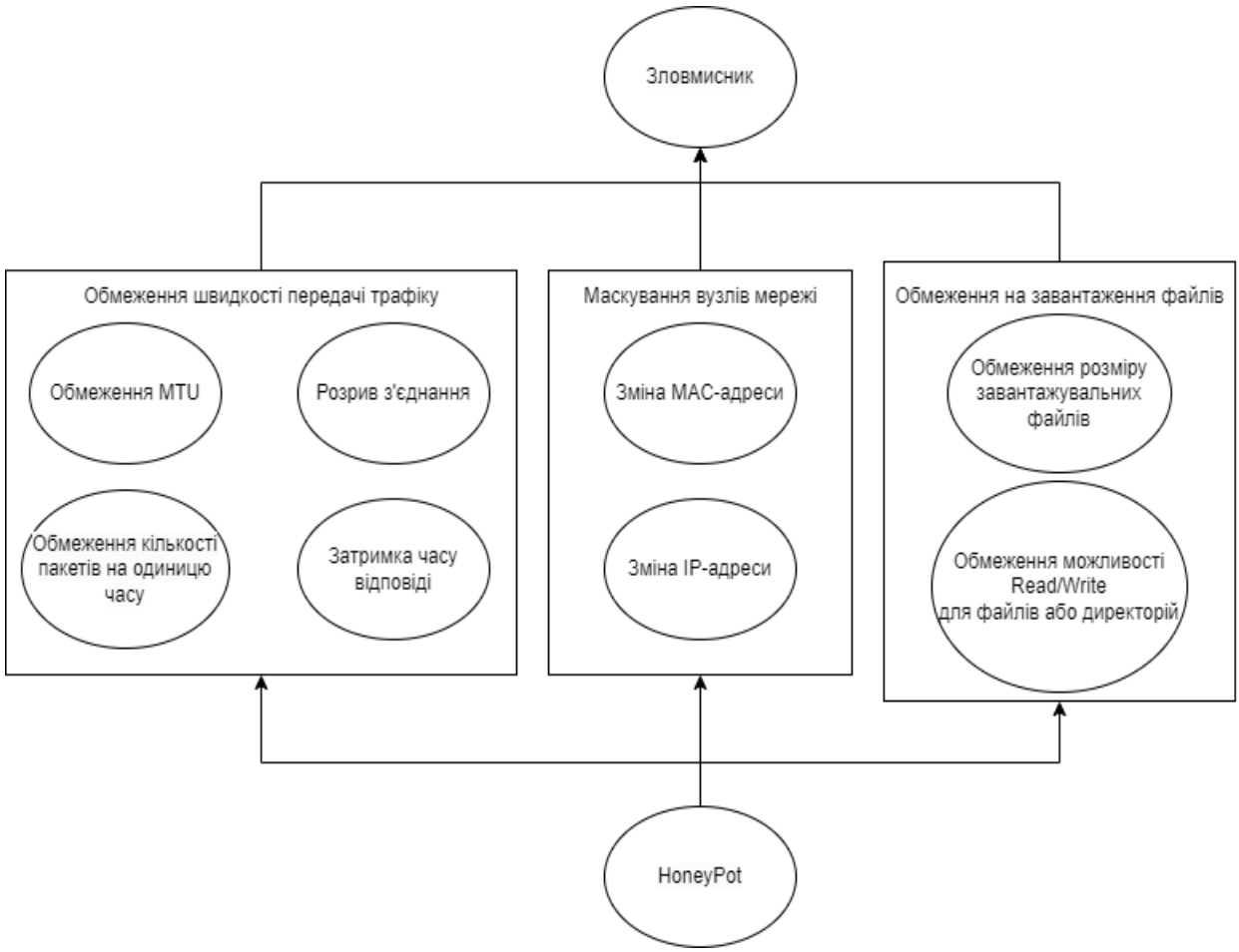

Рисунок 3.1 - Сценарна хмара зворотніх дій HoneyPot без залучення AI

Суттєвим недоліком такого підходу є недостатня швидкість реагування на нові інциденти безпеки, тобто коли зловмисник виконує дію, що не може бути класифікована HoneyPot(модифікація методу зламу), або не визначена адміністратором безпеки(раніше невідомий метод зламу), система або не надасть реакції, або буде відтворювати заданий шаблон для невідомих ситуацій.

Залучення AI у процес проектування, покращення сценарноъ хмари надає можливість визначати основні характеристики, на які спирається метод зламу, а

отже підбирати коректну реакцію системи на вторгнення. Окрім покращення вищезазначених протидій, сценарна хмара може бути розширена додатковими реакціями. Насамперед це створення фальшивих файлів і даних, що повинні виглядати досить переконливо, щоб привернути увагу зловмисника. Наприклад, можна використовувати назви файлів, які натякають на важливу інформацію, такі як "Q4_Financial_Report_2023.xlsx" або "Admin_Credentials.txt". Розміщення таких файлів повинно бути продуманим: вони можуть бути розташовані як у легкодоступних каталогах, так і глибше в структурі директорій, створюючи враження справжніх, але прихованих документів.

Крім цього, важливим елементом є створення фальшивих облікових записів, які можуть імітувати як привілейованих користувачів, так і звичайних співробітників. Облікові записи з іменами на кшталт "admin_user" або "sys_admin" можуть бути використані для залучення уваги хакерів, оскільки такі облікові дані можуть бути використані для підвищення привілеїв у системі. Ці облікові дані можна розміщувати у внутрішніх системах та на корпоративних порталах, де хакер може спробувати скористатися цими акаунтами для отримання доступу до захищених ресурсів.

Таким чином можна створювати окрему мережу HoneyToken, що створені і поширені без участі адміністратора безпеки. Використання такої протидії дає можливість відслідкувати слабкі місця у системі, що потенційно є місцем витоку даних.

Важливою функцією таких Honeypot систем є можливість тренування і тестування систем захисту інформації. За допомогою AI Honeypot можна відтворювати сценарії атак, що були використані зловмисником у попередніх спроб злому. Такі симуляції має сенс використовувати для навчання та перевірки ефективності поточного рівня захисту (пентестингу) цільових ІС. Реалізація концепції ретроспективного аналізу стану захищеності ІС є критично важливою при впровадженні проактивної стратегії реагування, оскільки дає регулярну звітність про вразливих місць в поточних заходах безпеки, а отже своєчасне винайдення ефективних протидій.

3.4 Узагальнення існуючих AI HoneyPot

Комерційні HoneyPot систем, що використовують AI, розробляються в умовах значного зростання комерційного інтересу до технологій кібербезпеки через збільшення масштабів та варіативності кібератак. Компанії прагнуть знайти рішення, яке мінімізує ризики втрат.

Одним з головних стимулів для розробки комерційних AI Honeypot є їх здатність ефективно взаємодіяти з іншими елементами інфраструктури безпеки. Вони можуть об'єднуватися з IDS, FW та SIEM-платформами, створюючи єдиний центр управління кіберзахистом. Це дозволяє компаніям отримувати детальні аналітичні звіти, проводити повний аналіз інцидентів та адаптивно реагувати на потенційні загрози у режимі реального часу. Для розробників HoneyPot, що використовують AI, найважливішою перевагою інтеграції багатьох систем у єдину платформу є можливість активно продавати інші свої продукти, що можуть вільно взаємодіяти між собою шляхом використання одного API. У свою чергу, для клієнтів, така можливість спрощує процес розгортання та підтримки системи захисту.

Огляд існуючих AI-HoneyPot [50] дозволяє оцінити поточний стан розвитку технологій кібербезпеки та зрозуміти їхні сильні та слабкі сторони. Це допомагає визначити, які інструменти вже довели свою ефективність у протидії сучасним загрозам, а які технології потребують подальшого вдосконалення. Аналіз наявних рішень також дозволяє сформувати базу для порівняння, що є важливим для подальшої розробки нових концепцій та їхнього тестування в умовах реального застосування. Прикладами таких HoneyPot-ів є:

NeuralPot [51] – це AI-Honeypot, що використовує два типи нейронних мереж для генерації мережевого трафіку, який призначений для навчання систем кіберзахисту. GAN відповідає за створення трафіку, а Autoencoder дозволяє адаптувати його, щоб він максимально нагадував реальний, спираючись на дані з існуючих мереж. Якість згенерованого трафіку оцінюється за допомогою показника FID, який визначає ступінь подібності до реальних зразків. Автори системи планують випробувати NeuralPot у реальних промислових мережах для

перевірки його ефективності під час реальних атак. Вони також планують розширити підтримку системи на інші промислові протоколи, що зробить її більш універсальною та придатною для захисту різних видів інфраструктури.

SPHINX AI Honeypot [52] – це експериментальна система, створена для виявлення аномалій у кібератаках за допомогою алгоритмів машинного навчання. У її складі є самонавчальна нейронна мережа, яка аналізує дії зловмисників у honeypot і автоматично ідентифікує підозрілу активність. Система використовує спеціальний механізм для відстеження шаблонів поведінки атакувальників, що дозволяє розрізняти звичайну активність і потенційно небезпечні дії. SPHINX AI Honeypot здатний адаптуватися до нових загроз у реальному часі, представляючи собою приклад автоматизованої honeypot системи, що має можливість самостійно налаштовувати свої параметри захисту.

Ghost [53] – це honeypot-система, розроблена для захисту пристроїв IoT із застосуванням технологій штучного інтелекту. Вона спеціалізується на виявленні та блокуванні кібератак, спрямованих на IoT-пристрої, які часто мають обмежені можливості для захисту. Ghost створює приманки для зловмисників, перенаправляючи їхні атаки на пастки, а не на реальні пристрої, що дозволяє запобігти потенційним загрозам.

Ключова особливість Ghost – використання алгоритмів машинного навчання для аналізу мережевого трафіку. Завдяки навчанню на даних, отриманих із реальних IoT-пристроїв, система здатна виявляти аномальні дії та відхилення від стандартної поведінки в режимі реального часу. Це робить її ефективною для виявлення складних атак, таких як ботнети, DDoS та інші загрози, характерні для IoT-середовища. Ghost також має адаптивні можливості, що дозволяють системі підлаштовуватися під зміни в мережі. Завдяки постійному навчанню на нових даних, система здатна оновлювати свої алгоритми, забезпечуючи оперативну реакцію на нові загрози, що робить Ghost надійним інструментом захисту для динамічного IoT-середовища.

HoneyAgents [54] – це система для захисту веб-додатків, яка використовує автономних AI «агентів» для виявлення та нейтралізації кіберзагроз. Основу HoneyAgents становить фреймворк AutoGen, який дозволяє створювати складні додатки з LLM через взаємодію кількох агентів. Ці агенти здатні вести «діалоги», координувати складні завдання та за потреби залучати персонал з ІБ для прийняття рішень, що забезпечує автономну роботу такої системи.

HoneyAgents аналізує журнали подій для виявлення підозрілої активності, автоматично оновлює чорні списки для блокування загроз та створює детальні звіти про активність зловмисників. Інтелектуальні агенти в режимі реального часу збирають дані про шаблони атак і автоматично налаштовують конфігурації проксі-серверів для захисту мережі. Багато HoneyPot-систем, здатних взаємодіяти зі зловмисниками, є пропрієтарними, що обмежує можливості аудиту та налаштування для адміністраторів інформаційної безпеки. Відкриті рішення з відкритим кодом, на відміну від них, пропонують гнучкість у налаштуванні, підвищуючи надійність захисту, особливо в умовах появи нових, невідомих загроз.

Наведені вище моделі пропонують принципово різні підходи, демонструючи спектр можливостей, що надає AI при використанні у комплексі з HoneyPot-ами. Проте, всі, наведені вище, рішення є:

1) Експерементальними моделями, а їх використання у комерційних цілях є недоцільним, оскільки відсутня відповідна юр. особа, що є відповідальною за реалізацію продукта, надає технічну підтримку у процесі реалізації та впровадження системи у ІС потенційних клієнтів, займається збором та аналізом отриманих відомостей. Натомість, збір та аналіз інформації про поточний ландшафт загроз надається окремим компаніям, де якість аналізу у повній мірі залежить від кваліфікації їх адміністраторів безпеки.
2) Обмеженим або неможливим для конфігурації із боку адміністраторів безпеки, що планують впроваджувати її у свою ІС. За конфігурацію відповідних реакцій, налаштування та навчання AI відповідальні розробники системи. У таких системах, тільки розробник може вносити

зміни у модель AI, що використовується, формувати сценарну хмару реакцій системи на спробу вторгнення, що обмежує гнучкість системи для конкретних зразків ІС.

3.5 Опис експериментальної моделі AI HoneyPot з гнучкою конфігурацією

Вирішенням існуючих проблем є впровадження комплексної моделі HoneyPot, що може бути повністю сконфігурована для потреб конкретної ІС. Загальна схема функціонування зображена на рисунку 3.2. Така схема передбачає розділення системи на незалежні компоненти, що можуть бути як із відкритим програмним кодом, так і закритим. Взаємодія таких компонентів можлива за допомогою використання спільного API. Вибір конкретних компонентів покладається на адміністраторів безпеки конкретної ІС [55].

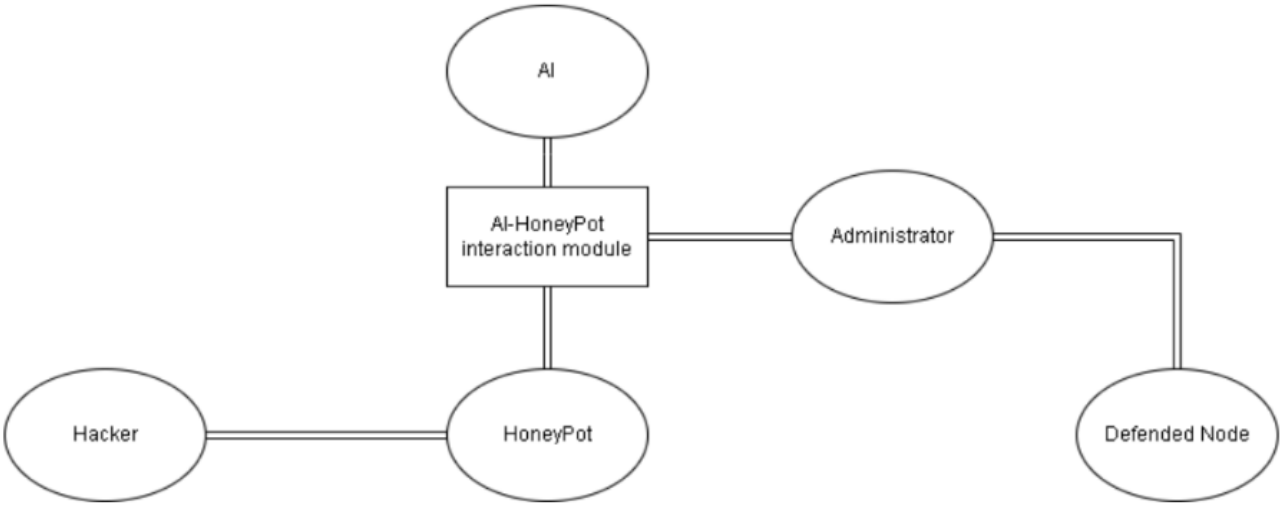

Рисунок 3.2 - Структурна модель AI HoneyPot із гнучкою конфігурацією

Розділення системи на окремі незалежні компоненти надає ряд переваг:

- Можливість покращення показників масштабованості;
- Впровадження єдиної політики реагування для мережі HoneyNet, оскільки один модуль AI може надавати інструкцію декільком HoneyPot-ам;
- Впровадження декількох модулів AI для одного HoneyPot-а з метою впровадження відказостійкості HoneyPot або для тестування різних моделей нейромереж у однакових умовах.

Ця модель передбачає роботу системи у напівавтоматичному режимі. Задля зменшення ризиків виникнення помилкових реакцій, часткового або повного виходу з ладу системи AI HoneyPot, передбачена необхідність участі адміністратора безпеки у процесах функціонування системи захисту. Таке гібридне управління водночас спрощує процес реагування на атаки і зменшує можливі ризики використання AI.

Висновки з розділом:

1) Використання AI у технології HoneyPot здатно надати нові та покращити вже реалізовані можливості системи, відкриваючи нові шляхи впровадження проактивної стретегії реагування;
2) Збір якісної аналітики системою HoneyPot може бути використано у інших системах захисту;
3) Залучення AI у процеси функціонування HoneyPot можливе 2 принципово різними шляхами - Expert system та case based system. Кожен із цих методів має свої переваги та недоліки;
4) Коректний вибір класу нейромереж має ключову роль для визначення цілей HoneyPot, метрик оцінювання;
5) Використання AI здатно значно розширити сценарну хмару взаємодії HoneyPot шляхом додавання нових реакцій, наприклад створення мереж HoneyToken;
6) Наразі, моделі AI HoneyPot є обмеженими або неможливими для використання у комерційних системах через відсутність відповідальної особи або недостатню гнучкість налаштування системи;
7) Якісно новий рівень зручності здатна забезпечити комплексна модель AI HoneyPot, що склададається з окремих модулів. Взаємодія між модулями можлива через API.

4 СТВОРЕННЯ ТА ДОСЛІДЖЕННЯ ЕКСПЕРИМЕНТАЛЬНОЇ МОДЕЛІ AI HONEYPOT

4.1 Опис середовища функціонування системи

Задля доведення ефективності моделі AI HoneyPot була створена експериментальна модель, що використовує нейромережу для виявлення та реагування на інциденти у HoneyPot. За основу було взято платформу T-Pot [56]. Дана платформа налічує понад двадцяти HoneyPot, що працюють із найпоширенішими протоколами, що дає змогу якісного збору, аналізу та реагування на принципово різні види атак.

Дана система підтримує можливість розподіленого встановлення. Систему може бути поділено на 2 основних компоненти: T-Pot Sensors(сенсори), T-Pot Management(вузол керування). Схема розподіленого розташування зображена на рисунку 4.1.

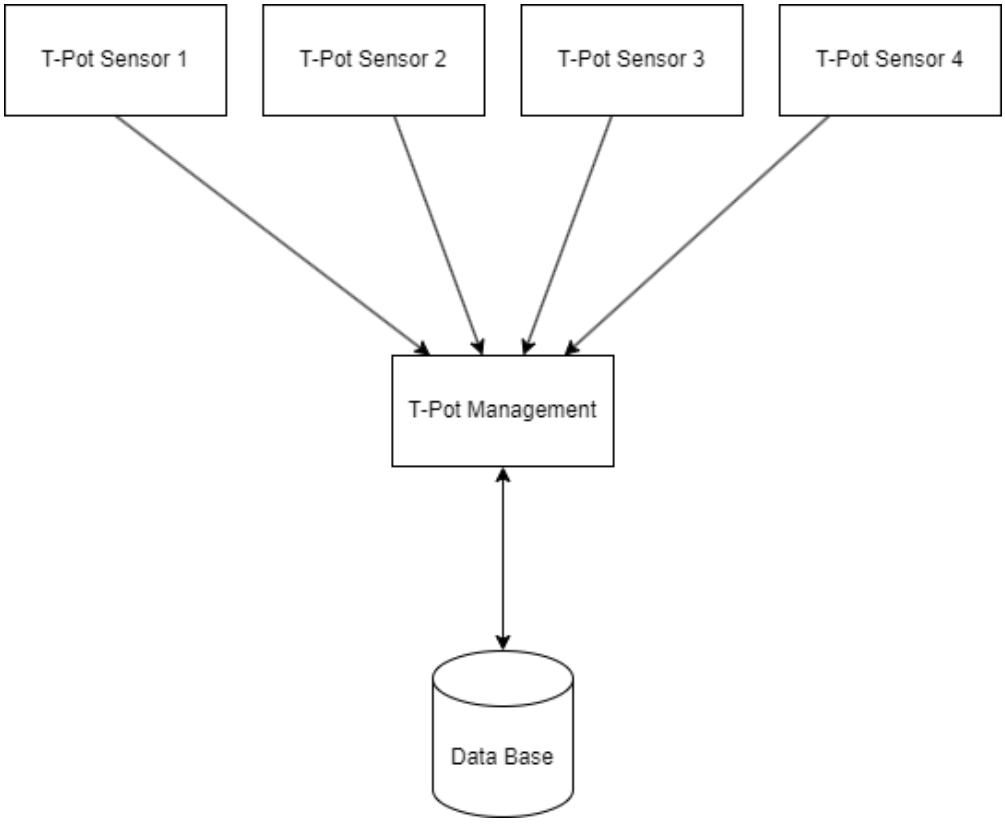

Рисунок 4.1 – Схема розподіленого розташування системи T-Pot

Сенсори є HoneyPot-ами, що можуть бути розгорнуті на будь-якому вузлі ІС. У базовій реалізації даний компонент лише збирає відомості про атаку, та передає до вузла керування. У свою чергу, вузол керування, у базовій конфігурації, збирає аналітику, та зберігає файли логів. Розподілення системи дозволяє значно скоротити вимоги до серверу, де буде розміщено T-Pot, проте потребує додаткового налаштування. У даній роботі буде використано інший метод розгортання системи – Hive. Даний метод передбачає, що сенсори та вузол керування об'єднані і розміщенні на одному вузлі.

Зручним інструментом аналізу є наявність Elasticsearch browser, що дозволяє візуалізувати процеси, що відбуваються в усіх HoneyPot. Одною з важливих функцій є побудова мапи атак на основі DNS аналізу IP-адрес. Схема роботи зображена на рисунку 4.2.

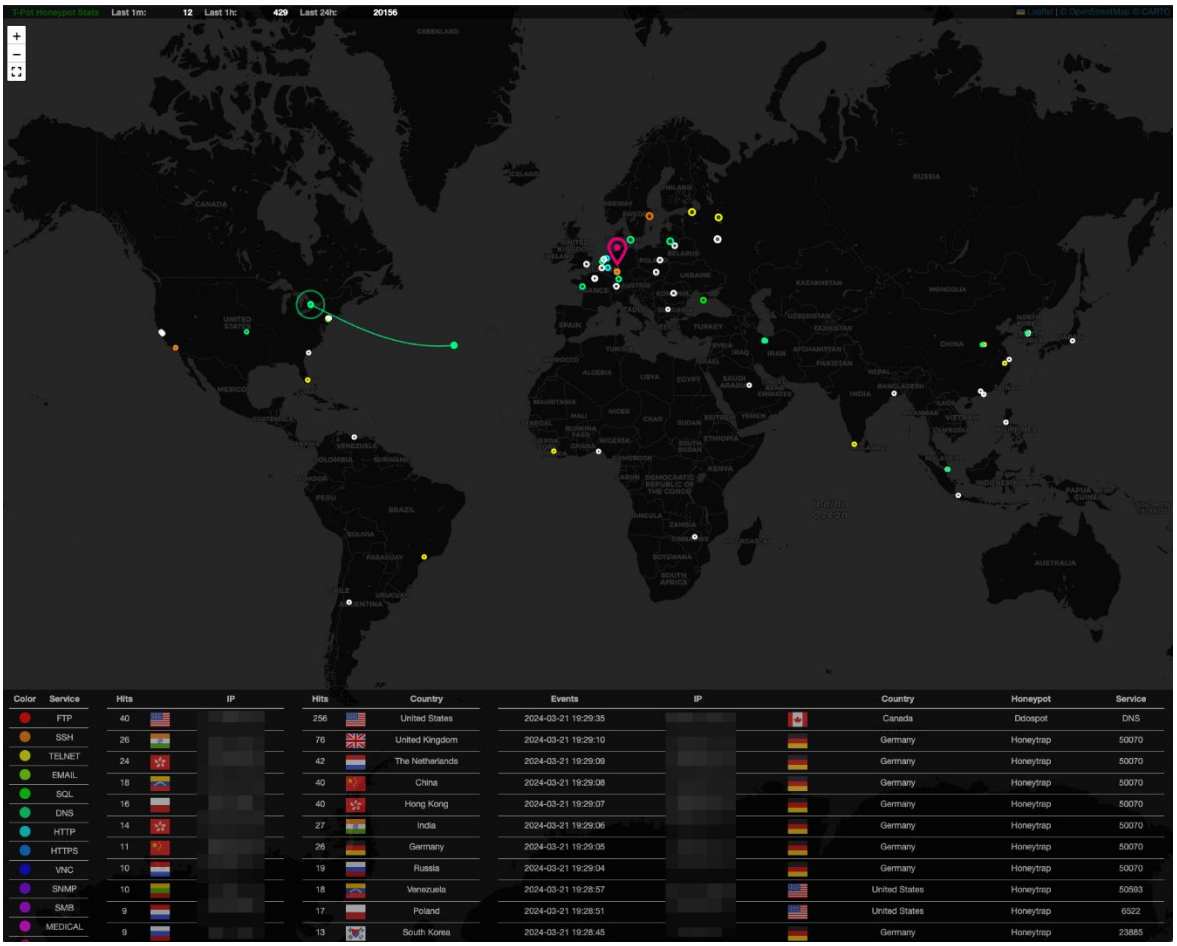

Рисунок 4.2 – Приклад мапи мережевого трафіку, що було відправлено на сервер T-Pot

Проте, важливим недоліком цієї мапи є ігнорування внутрішніх адрес, тому цю карту не може бути використано у процесі тестування на локальному сервері.

У процесі тестування у локальній мережі, більш ефективним буде використання інструменту Kibana, загальний вигляд якої представлено на рисунку 4.3.

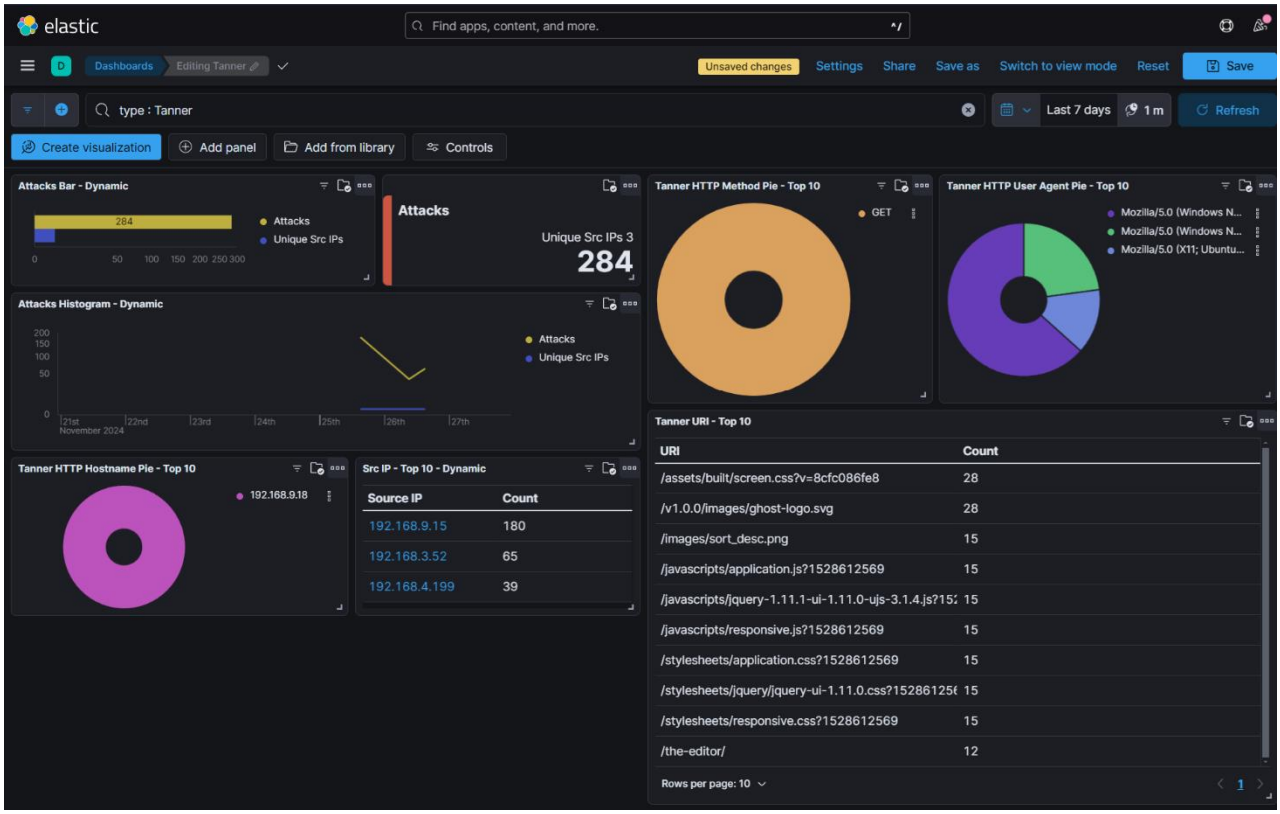

Рисунок 4.3 – Діаграми Kibana

Даний інструмент дозволяє відстежувати основні ключові параметри атаки, а саме:

- 10 найпоширеніших HTTP API методів;
- Агенти, з яких йшов запит;
- IP-адреси, з яких було надіслано запити на систему;
- URI, до яких звертались.

До складу T-Pot входять декілька інструментів для обробки даних, а саме:

- Cyberchef – інструмент для аналізу та декодування даних. Інструмент організовано як веб-застосунок, що розгорнуто на окремому сервері.

- SpiderFoot [57] – це інструмент автоматизації розвідки з відкритим вихідним кодом (OSINT). Він інтегрується практично з усіма доступними джерелами даних і використовує цілий ряд методів для аналізу даних, що полегшує навігацію в цих даних. SpiderFoot має вбудований веб-сервер для забезпечення чистого та інтуїтивно зрозумілого веб-інтерфейсу, але також може використовуватися повністю через командний рядок. Він написаний на Python 3 і має ліцензію MIT. Зовнішній вигляд запуску сканування зображено на рисунку 4.4.

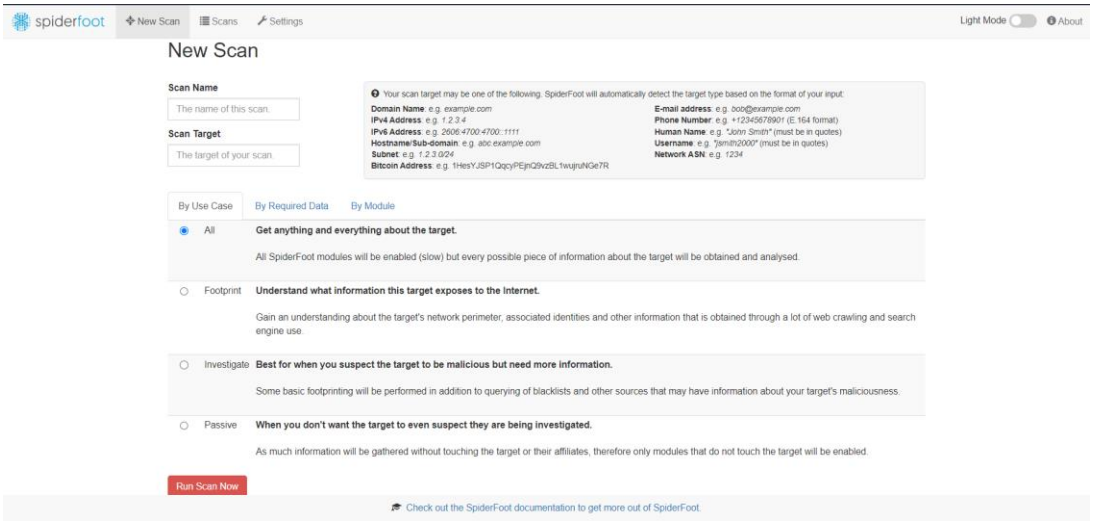

Рисунок 4.4 – Зовнішній вигляд інструменту SpiderFoot

Головною особливістю є використання 26-ти HoneyPot, різного ступню взаємодії, цільового призначення та можливих реакцій на дії хакера. При цьому кожен HoneyPot працює незалежно, реалізуючи взаємодію із хакером на окремих портах. Повний перелік портів, що використовуються у T-Pot надано у таблиці 4.1.

Таблиця 4.1 – Лист портів, які використано у системі T-Pot

Порт, що використано	**Protocol**	**Description**
64295	tcp	T-Pot: SSH доступ
64297	tcp	T-Pot Доступ до зворотнього проксі(NGINX)
5555	tcp	HoneyPot: ADBHoney
5000	udp	Honeypot: CiscoASA
8443	tcp	Honeypot: CiscoASA
443	tcp	Honeypot: CitrixHoneypot

Продовження таблиці 4.1

Порт, що використано	**Protocol**	**Description**
80, 102, 502, 1025, 2404, 10001, 44818, 47808, 50100	tcp	Honeypot: Conpot
161, 623	udp	Honeypot: Conpot
22, 23	tcp	Honeypot: Cowrie
19, 53, 123, 1900	udp	Honeypot: Ddospot
11112	tcp	Honeypot: Dicompot
21, 42, 135, 443, 445, 1433, 1723, 1883, 3306, 8081	tcp	Honeypot: Dionaea
69	udp	Honeypot: Dionaea
9200	tcp	Honeypot: Elasticpot
22	tcp	Honeypot: Endlessh
21, 22, 23, 25, 80, 110, 143, 443, 993, 995, 1080, 5432, 5900	tcp	Honeypot: Heralding
53, 123, 161, 5060	udp	Honeypot: qHoneypots
631	tcp	Honeypot: IPPHoney
80, 443, 8080, 9200, 25565	tcp	Honeypot: Log4Pot
25	tcp	Honeypot: Mailoney
2575	tcp	Honeypot: Medpot
6379	tcp	Honeypot: Redishoneypot
5060	tcp/udp	Honeypot: SentryPeer
80	tcp	Honeypot: Snare (Tanner)
8090	tcp	Honeypot: Wordpot

Ключовою особливістю є те, що кожен HoneyPot розгорнуто у окремому віртуальному середовищі, за допомогою технології контейнерів. У даному сервісі використовується *Docker*, а саме *docker-compose*, що спрощує процес розгортання самої мережі пасток, дозволяє налаштувати внутрішню мережу взаємодії пасток між собою та вузлом управління.

Не менш важливою перевагою застосування контейнерів для розгортання пасток є їх ізоляція від основної системи, що дає змогу змінювати конфігурацію, вимикати, перезапускати та підміняти конкретні контейнери без необхідності зупинення всієї системи. Також, зберігання пасток у окремих контейнерах дає додатковий механізм безпеки у разі зламу пастки зловмисником. У такому випадку він отримає доступ до віртуальної середи виконання сервісу пастки, а не

вузла, де розгорнуто систему T-Pot. У випадку своєчасного реагування, шкода від зламу HoneyPot є мінімальною.

У даній роботі буде використано дані, що збережено HoneyPot-ом *Snare(Tanner)*. Даний HoneyPot дозволяє імітувати веб-сервер, що прослуховує і відповідає на порт 80. Приклад веб сторінки зображено на рисунку 4.5.

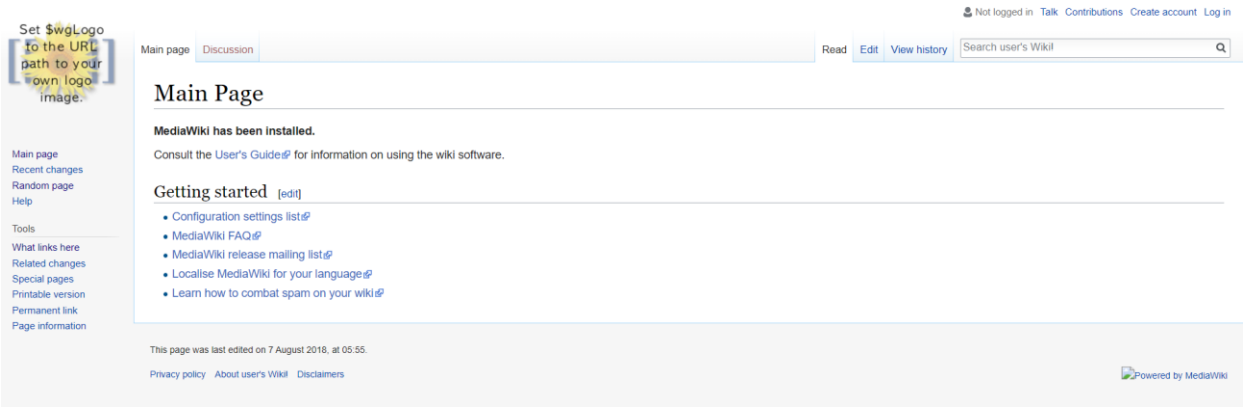

Рисунок 4.5 – Веб-сервер, що імітовано пасткою *Tanner*

Усі запити, що поступили на цей сервіс, занесені у log-файл, що знаходиться за шляхом *"tpotce/data/tanner/log/tanner_report.json"*. Отримання останнього занесеного запиту зображено на рисунку 4.6.

Рисунок 4.6 – Останній запит, що занесено до файлу логів tanner_report.json

Даний HoneyPot збирає інформацію про конкретний запит у вигляді JSON об'єктів, що мають наступні поля:

- HTTP API метод;
- Шлях, до якого відбулося звернення;
- IP-адреса отримувача та відправника;
- Браузер, що застосував відправник;
- Мови, що приймає відправник;

- Порт, з якого надіслано запит;
- Статус відповіді сервера;
- Мітка часу.

Через велику кількість одночасно працюючих віртуальних оточень для кожного HoneyPot, T-Pot надає необхідні вимоги до апаратного устаткування серверу, а саме:

- наявність 16 ГБ RAM;
- 256 ГБ доступної пам’яті на жорсткому диску.

Дані вимоги наведено для конфігурації Hive, тобто поєднання сенсорів та вузлу керування на одному фізичному сервері.

Впровадження AI у систему може накладати додаткові вимоги. Насамперед це збільшення RAM до 32 ГБ задля забезпечення безперебійного функціонування системи. Задля покращення швидкості навчання нейронної мережі, рекомендується використовувати CUDA-сумісну відеокарту.

4.2 Підготовка мережевого оточення

Для розгортання експериментальної моделі було використано сервер із наступними характеристиками:

- Процессор: Intel Core i7 5500U, 3 GHz;
- RAM: 8 GB DDR3;
- Обсяг доступної пам’яті на жорсткому диску: 256 GB;
- Мережева адреса: 192.168.9.18.

Для інсталяції було обрано ОС Ubuntu Server 24.04.01. Дана UNIX подібна операційна система повністю сумісна з системою HoneyPot. Вибір серверної комплектації пов’язано із уникненням конфліктів портів, що використовуються графічною оболонкою GNOME. Для коректної роботи T-Pot, ОС має бути встановлена у мінімальній конфігурації з встановленням OpenSSH Server.

Після інсталяції системи, у домашню директорію користувача dmykhailenko було завантажено віддалений Git-репозиторій, що містить систему

T-Pot. Команда для завантаження та результат виконання зображено на рисунку 4.7.

```
dmykhailenko@laptop-100030:/home/dmykhai1enko$ sudo git clone https://github.com/telekom-security/tpotce
Cloning into 'tpotce'...
remote: Enumerating objects: 16779, done.
remote: Counting objects: 100% (311/311), done.
remote: Compressing objects: 100% (175/175), done.
remote: Total 16779 (delta 145), reused 282 (delta 131), pack-reused 16468 (from 1)
Receiving objects: 100% (16779/16779), 296.72 MiB | 9.53 MiB/s, done.
Resolving deltas: 100% (9345/9345), done.
dmykhailenko@laptop-100030:/home/dmykhai1enko$ ls -la
drwxr-xr-x - root 28 Nov 00:20 tpotce
dmykhailenko@laptop-100030:/home/dmykhai1enko$
```

Рисунок 4.7 – Домашня директорія користувача після завантаження Git репозиторію

Задля встановлення системи, розробниками передбачено скрипт install.sh. Важливою умовою коректної інсталяції є запуск файлу без прав root користувача. Виконання цього скрипту, інсталяції буде завантажено необхідні пакети, а саме: ansible, cracklib-runtime, wget. Після встановлення, користувачу буде запропоновано обрати конфігурацію, у якій буде встановлено T-Pot, що зображено на рис. 4.8.

```
### Playbook was successful.


### Choose your T-Pot type:
### (H)ive   - T-Pot Standard / HIVE installation.
###            Includes also everything you need for a distributed setup with sensors.
### (S)ensor - T-Pot Sensor installation.
###            Optimized for a distributed installation, without WebUI, Elasticsearch and Kibana.
### (M)obile - T-Pot Mobile installation.
###            Includes everything to run T-Pot Mobile (available separately).
### Install Type? (h/s/m)
```

Рисунок 4.8 – Вибір конфігурації T-Pot

Для можливості взаємодії із WEB інтерфейсом керування T-Pot система вимагає створення нового користувача, дані якого використовуються для входу по адресі: https://<T-Pot IP>:64297. Процес створення веб-користувача зображено на рисунку 4.9.

```
### T-Pot User Configuration ...

### Enter your web user name: dmytro_mykhailenko
### Your username is: dmytro_mykhailenko
### Is this correct? (y/n) y_
```

Рисунок 4.9 – Створення нового користувача

Задля уникнення можливих конфліктів, логін веб-користувача має відрізнятись від логіну системного користувача. Для веб-користувача обрано логін: dmytro_mykhailenko. У разі втрати даних для веб-користувача, або задля реалізації адміністрування системи декількома адміністраторами, передбачена процедура додавання нового користувача. Відповідний файл genuser.sh знаходиться у кореневому каталозі tpotce.

Після цього налаштування, система почне завантажувати Docker образи HoneyPot-ів. Останньою дією має бути перезапуск ОС. Після цього, доступ до системи здійснюється за допомогою SSH з використанням порту 64295. Процес входу до серверу через SSH продемонстровано на рис. 4.10.

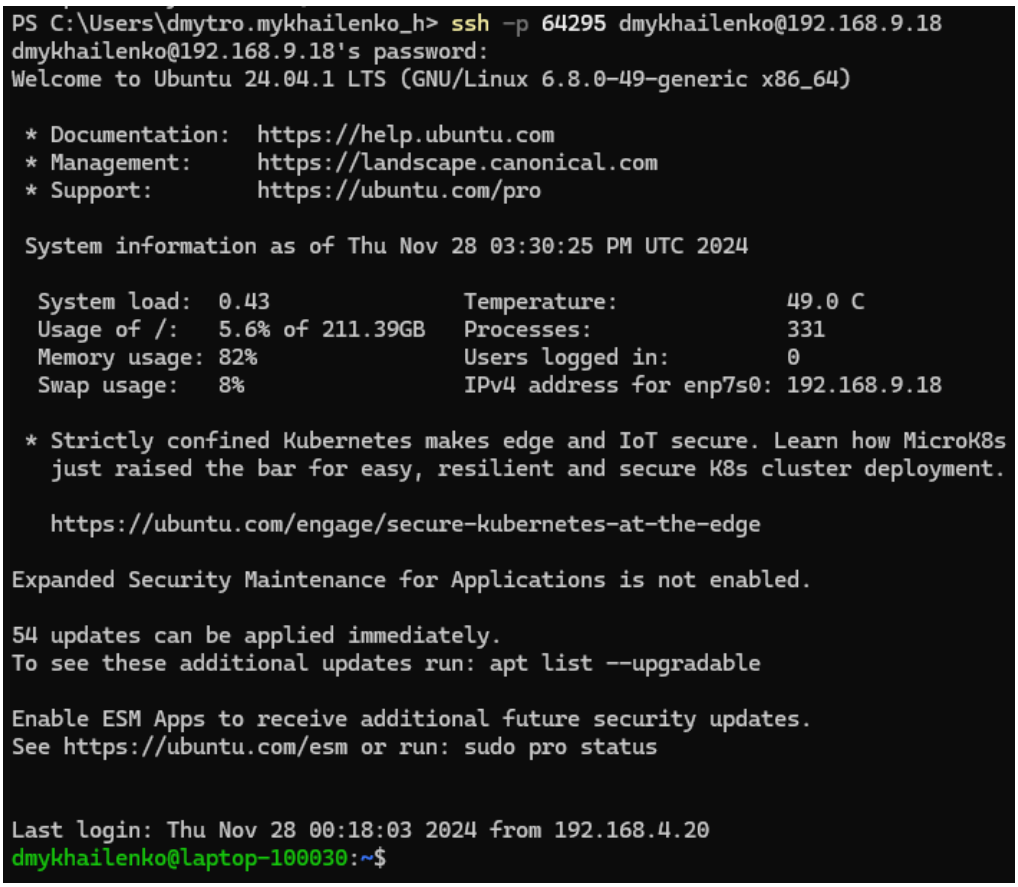

Рисунок 4.10 - Вхід до системи за допомогою SSH

Інструменти візуалізації, прикладні інструменти для тестування доступні у вигляді веб-сторінки за адресою https://<T-Pot IP>:64297. Веб інтерфейс зображено на рисунку 4.11.

Рисунок 4.11 – Веб інтерфейс T-Pot

4.3. Реалізація експериментальної моделі фільтрації DDoS трафіку

Задля доведення практичної можливості імплементації AI у дану систему, було розроблено нейромережу, що здатна визначити DDoS трафік, аналізуючи запити, що занесено у файл логів у форматі JSON. Варто зауважити, що у даній роботі для навчання нейронної мережі, DDoS трафік визначено як аномально високу кількість запитів з одного вузла. У разі залучення до атаки великої кількості IP-адрес, що надсилатимуть пакети із прийнятною, для звичайного користувача, швидкістю, дана модель не зможе визначити DDoS трафік.

Аналіз трафіку на наявність DDoS атак відбувається за допомогою вирахування часу міжпакетного інтервалу, тобто час, що пройшов між запитами від одного клієнта. Згідно даних, що занесені log-файл HoneyPot-у Tanner, мінімальний міжпакетний інтервал має значення $1 * 10^{-2}$ с. Це значення буде використане як порогова межа для створення набору даних для навчання нейромережі. Задля визначення часового міжпакетного інтервалу (ΔT) використано формулу:$\Delta T = |T_2 - T_1|$.

Ще одним важливим показником є IP-адреса, з якої було надіслано запит. У процесі навчання, нейромережа не враховує IP-адресу, оскільки даний вхід значно ускладнює вимоги до навчального набору. Також, використання IP-адреси теоретично дозволяє створення переліку «надійних» та «ненадійних» IP-адрес, а отже і спотворення результатів експерименту. Але, IP-адреса

використовується під час обчислення міжпакетного інтервалу, тобто посередньо приймає участь у процесі виявлення DDoS трафіку.

У процесі аналізу не буде враховано показники «HTTP API метод», «UUID», “User Agent”. Дані показники можуть надати більше інформації нейромережі задля класифікації трафіку, проте потребують більш складної архітектури.

У якості навчального набору було взято log-файл «нормального» трафіка, що подано у якості набору JSON об’єктів. На момент формування, log-файл налічував приблизно 300 записів. Трафік, що має бути класифіковано як DDoS атака, було синтезовано на основі «нормального» трафіку шляхом модифікації часової мітки, емулюючи ситуацію одночасного потрапляння на сервер великої кількості пакетів. Для розпізнавання, до синтезованого трафіку було додано поле is_ddos із значенням «1». Це значення використовується як очікуване у процесі навчання нейромережі. При створенні навчального набору було використано *«правило 80/20»*, що пропонує розділення даних на набори. Перший складає 80% від всього набору даних та слугує для навчання нейромережі. Другий набор слугує для перевірки коректності навченої мережі та становить 20% від всього набору.

Для оптимальності навчання нейромережі, кількість синтезованих DDoS пакетів приблизно рівна кількості «нормальних» пакетів. Ітогова кількість кількість трафіку сягає приблизно 600 пакетів. Задля забезпечення кращого процесу навчання трафік було перемішано випадковим чином, аби нейромережа із рівною вірогідністю отримувала як «нормальний», так і DDoS трафік.

Для цього, було використано скрипт перемішування трафіку, вихідний код якого представлено у додатку А. Загальна схема роботи представлена на рисунку 4.12.

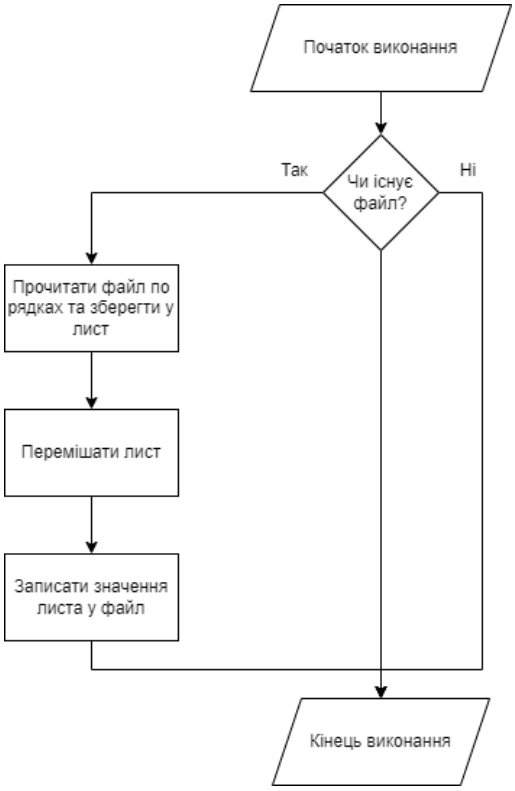

Рисунок 4.12 – Блок-схема алгоритму перемішування трафіку

Нейромережа, що аналізує трафік є реалізацією Feedforward Neural Network. Система має 1 вхід, а саме: різниця часу надсилання запитів. Час реєстрації запиту клієнта на сервері занесено у полі timestamp, що має наступний формат:

"timestamp": "2024-11-25T14:26:44.951962", де:

"timestamp" - назва поля JSON;

2024 - рік звернення;

11 - місяць звернення у числовому форматі;

25 - день звернення;

14 - година звернення;

26 - хвилина звернення;

44 - секунда звернення;

951962 - мікросекунда звернення.

Задля спрощення процесу розгортання, алгоритм було розділено на 2 модуля, вихідний код яких у додатку Б та додатку В відповідно:

Модуль ddos_learning.py реалізує механізм створення та навчання нейромережі. Результатом його роботи є навчена модель, що збережена у файл ddos_model.h5. Даний модуль складається з наступних функцій:

1) preprocess_training_data(training_log_file), аргументом якої шлях до логу із трафіком, що має бути використано як набір даних для навчання. Даний модуль зчитує log-файл по рядках та розбиває дані на 2 листи. В одному зберігається дані, що подаються на вхід нейромережі, в другому листі зберігається очікуване значення виходу. Перед подачею, дані проходять процедури масштабування(center scaling) та нормалізації.

   Нормалізація даних відбувається за наступною формулою:

$$x` = \frac{x - x_{min}}{x_{max} - x_{min}},$$

   де $x`$ −нормалізоване значення.

   Для Масштабування(center scaling) даних, потрібно визначити середнє значення даних та значення стандартної девіації. Середнє значення даних визначається як:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

   де $\mu -$ середнє значення даних;

   n −кількість даних(довжина масиву даних);

   $x_i -$ поточне значення даних.

   У свою чергу, середнє значення девіації визначається як:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2},$$

   де $\sigma -$ значення стандартної девіації.

   Таким чином, значення масштабованих даних, визначається за формулою:

$$z = \frac{x - \mu}{\sigma},$$

   де $z -$ масштабовані дані; $\mu -$ середнє значення змінної.

   Для впровадження процесів нормалізації та масштабування(center scaling) було застосовано бібліотеку scikit-learning, а саме StandardScaler;

2) extract_features(log_entry, client_timestamps), аргументами якого є log_entry – JSON об’єкт запиту до сервера. client-timestamps словник(dictionary, dict) що містить час останнього запиту від клієнта. Дана функція створена задля зчитування та обробки даних з JSON об’єкту. У даній функції відбувається обчислення міжпакетного інтервалу. У разі, якщо запит від клієнту відбувається вперше, дані про випадок не заносяться до набору навчальних даних, оскільки визначення міжпакетного інтервалу неможливе. З точки зору адміністратора інформаційної безпеки, такий трафік не може бути класифіковано як DDoS за недостатньою кількістю наявних даних. Натомість, створюється новий запис у словнику client_timestamps;
3) determine_label(log_entry) має 1 аргумент, що є записом JSON об’єекту. Функція створена для зчитування очікуваного значення, до якого має прийти нейромережа. У разі наявності поля “is_ddos”, значення якого встановлено як «1» для всіх пакетів, що було синтезовано як DDoS трафік, функція повертає значення цього поля. У іншому випадку буде повернуто «0», що є очікуваним значенням для нормального трафіку.
4) build_model(input_dim) приймає 1 аргумент, а саме – розмірність масиву вхідних даних. У даному модулі, для обчислення розмірності масиву, використано функцію shape з модуля numpy. Функція build_model() описує структуру моделі нейромережі. Для даної мережі було обрано тип Sequential, що має наступні особливості:
   - Нейрони розділено на шари(layers);
   - Кожен шар має 1 вхід та 1 вихід;
   - Підтримує лише лінійну топологію(Розгалуження шарів неможливе).

   Нейромережа має 9 шарів(layers), з яких 8 є прихованими(внутрішніми). Кількість нейроннів у кожному попередньому шарі збільшується у 2 рази тобто кількість нейронів конкретного шару можна вирахувати за такою формулою:

$$N = 2^i,$$

   де N −кількість нейронів; i −порядковий номер шару.

Функцією активації нейронів у вхідному та прихованих шарах є ReLu(Rectified Linear unit, зрізаний лінійний вузол), що математично може бути описано як максимальне значення між x та 0

$$y = \max(0, x)$$

Тобто функція приймає значення у межах [0, x_{max}], де x_{max} ‘ максимальне нормоване значення набору даних. У контексті даної нейромережі максимальне значення «10», тобто ReLu приймає значення у межах [0,10], а значення можу бути інтерпретовано як вірогідність пакету бути DDoS трафіком. Функція активації була обрана через свою простоту обчислення, оскільки при додатніх значення x, функція може бути описана як $y = x$. Не менш важливою характеристикою ReLU є уникнення проблеми загасання градієнта під час зворотного поширення помилки, оскільки похідна даної функції має значення «1» упродовж усіх додатніх значень x, що є актуальною проблемою у випадку використання сигмоїдної(sigmoid) функції активації. Графік функції ReLu представлено на рисунку 4.13. [58]

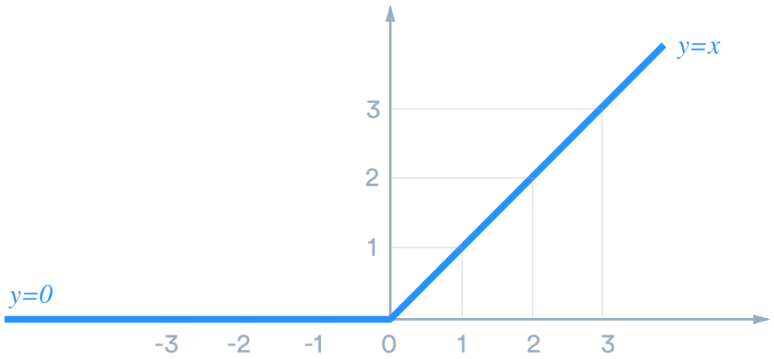

Рисунок 4.13 – Графік ReLu

Для вихідного нейрону використовується сигмоїдна(sigmoid) функція активації. Дана функція приймає значення у діапазоні$(-\infty, +\infty)$, а вихідні значення лежать у діапазоні [0, 1]. Математичну функцію може бути описано як:

$$y = \frac{1}{1 + e^{-x}}$$

Графічно сигмоїдна функція активації має вигляд монотонно зростаючої S-подібної кривої. Графік даної функції представлено на рисунку 4.14.

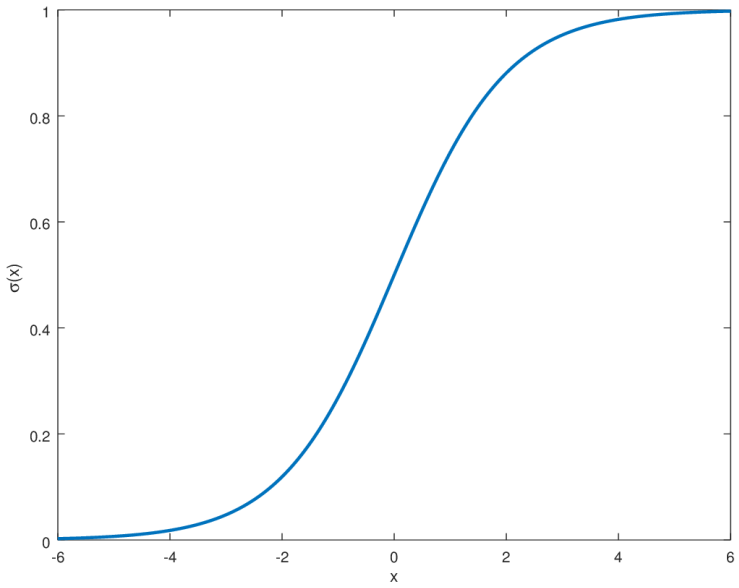

Рисунок 4.14 – Графік sigmoid [59]

Дана функція часто застосовується у задачах бінарної класифікації даних, тобто приналежності даних до одного з визначених класів. У такій задачі, якщо визначити питання «чи належить об’єкт до класу 1», то вихідне значення сигмоїдної функції визначатиме імовірність належності даних до класу 1. Задача пошуку DDoS трафіку у log-файлах може бути зведена до завдання бінарної класифікації, а питання, на яке відповідає мережа: «чи належить даний пакет до DDoS трафіку». Відповідь, що синтезована нейромережею визначатиме вірогідність наявності DDoS поведінки у проаналізованому трафіку. Функцією активації для вихідного шару було обрано сигмоїдну через нелійність вихідного значення, з оглядом на загальну практику використання даної функції для задач бінарної класифікації.

5) train_and_save_module(training_log_file) як аргумент приймає шлях з log-файлом. Мета даної функції – запуск процесу навчання нейромережі. Необхідна кількість епох(epoch) процесу навчання була визначена емпіричним шляхом та дорівнює «800». Менша кількість ітерацій не дає нейромережі достатньо часу для навчання, зменшуючи коректність

відповіді. У той же час, після сотої ітерації, помилка не змінюється, лише коливаючись у певному діапазоні. Крім того, така велика кількість епох призводить до втрати нейромережею узагальнюючої функції, тобто помилка збільшується.

Розмір батча(batch_size, chunk) було обрано 64 пакети. Даний параметр впливає на те, скільки пакетів одночасно було завантажено у пам'ять та подано нейромережі перед запуском алгоритму самонавчання(back propagation). Кількість даних впливає на швидкість навчання, оскільки загальна кількість ітерацій навчання може бути визначена за такою формулою:

$$N_I = E * \frac{S_{DS}}{S_B},$$

де $N_I -$ загальна кількість ітерацій навчання;

$E -$ кількість епох;

$S_{DS} -$ розмір навчальних даних;

$S_B -$ розмір одного батча.

Занадто великий розмір батчу може призвести до задіяння завеликих апаратних ресурсів, перш за все RAM, тому даний параметр слід підбирати, виходячи із конфігурації ПЕОМ, де буде здійснено процес навчання. У той же час, зміна розміру батчу, має бути супроводжена зміною кількості епох задля запобігання недостатнього або надлишкового навчання мережі. Як метод визначення необхідної кількості епох пропонується: задати велику кількість епох(3-10 тисяч), запустити процес навчання і відслідковувати момент, у якому градієнт зміни помилки наближується до 10^{-4}, а саме значення похибки періодично зростає та спадає. Епоха, де було помічено таку поведінку, є точкою, на якій процес навчання слід зупинити. Останнім показником, що впливає на швидкість навчання є коефіцієнт навчання(learning rate). Даний коефіцієнт визначає швидкість перебалансування ваг між нейронами під час алгоритму самонавчання(back propagation). Для даної мережі встановлено це значення

дорівнює 10^{-4}. Для оцінки якості навчання було задіяно 2 метрики оцінювання: коректність нейромережі та похибка. Обидві метрики задані у відсотках. Метрика коректності математично може бути виражена як:

$$\mathrm{C} = \frac{N_C}{N_T} * 100\%,$$

де C −коректність нейромережі;

N_C −кількість правильних відповідей;

N_T −загальна кількість відповідей.

Метрика похибки оцінюється за формулою визначення бінарної кроссентропії:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)],$$

де N −кількість прикладів у вибірці;

y_i −очікуване значення для і-го приклада;

$\hat{y}_i$ −надане значення для і-го приклада.

Для створення модулю використовувались наступні інструменти:

- JSON для читання та обробки даних у форматі JSON;
- NumPy для роботи з масивами даних, що використані для навчання нейромережі;
- Keras як інструмент побудови нейромережі;
- SciKit-learning як інструмент для нормалізації та масштабування даних.

Візуальне представлення нейронної мережі зображено на рисунку 4.15.

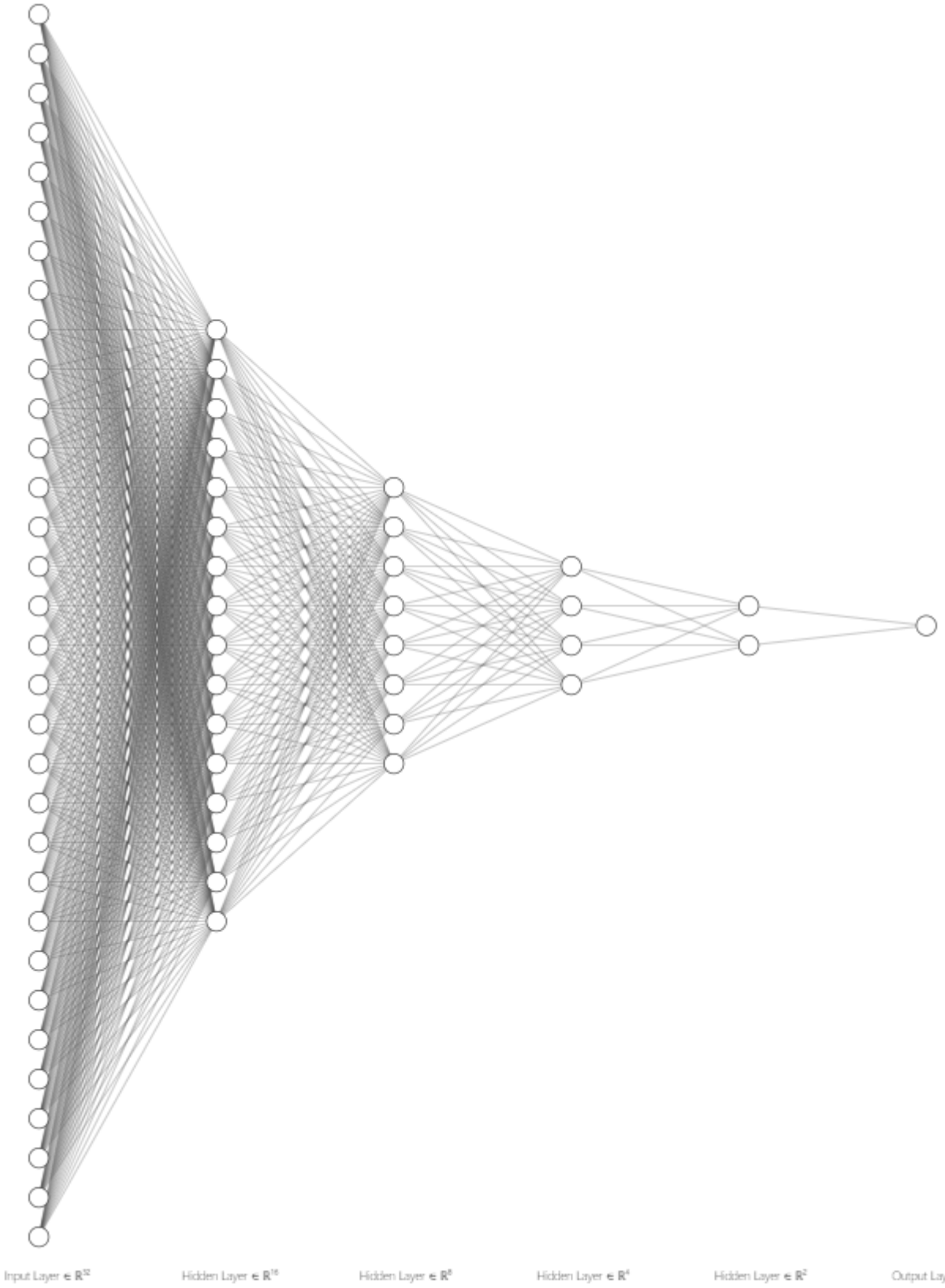

Рисунок 4.15 – Візаульне предствалення нейромережі фільтрації DDoS трафіку.

Для запуску було використано механізм контейнирезації Docker. Скрипт для збору контейнера наведено нижче:

```
FROM python:3.11-slim

RUN apt-get update && apt-get install -y \
    build-essential \
    && rm -rf /var/lib/apt/lists/*

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install --no-cache-dir -r requirements.txt

CMD ["python", "ddos_learning.py"]
```

Тут, за основу взято образ python 3.11. Першою дією скрипта є оновлення менджеру пакетів apt, та встановлення пакету build-essentail, що є компонентом, що забезпечує сумісність з дистрибутовим Debian, який використовується для встановлення бібліотек python. Робочою дирктрією обрано /app. У цю директорію буде завантажено файл навченої моделі нейромережі. Далі відбувається копіювання файлу, що містить необхідні бібліотеки(залежності, dependencies) модуля ddos_learning. Останнім кроком підготовки є встановлення залежностей, що описані у зкопійованому файлі. Точкою входу у віртуальне середовище встановлено модуль ddos_learning.

Запуск алгоритму відбувається за допомогою команди: docker run -v .\:/app/ docker_analyzer, де -v це флаг, що дозволяє монтувати директорії у контейнер у форматі “source:destination”. Після запуску команди відбувається процес навчання, що показано на рисунку 4.16.

Після завершення, модель буде збережено у файлі ddos_model.h5 у кореневій директорії виконання відповідного модулю ddos_learning. Монтування директорії у віртуальне середовище дозволить зберегти файл на ПЕОМ, з якої було запущено процес навчання.

Рисунок 4.16 – Перші епохи процесу навчання

Створення файлу, що містить навчену нейромережу показано на рисунку 4.17. Графік втрат та коректності відповідей на рисунку 4.18.

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_mo
del(model)`. This file format is considered legacy. We recommend using instead the native Keras fo
rmat, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       293
           1       1.00      1.00      1.00       310

    accuracy                           1.00       603
   macro avg       1.00      1.00      1.00       603
weighted avg       1.00      1.00      1.00       603

[[293   0]
 [  0 310]]
Model successfully saved to 'ddos_model.h5'.
```

Рисунок 4.17 – Збереження моделі у файл

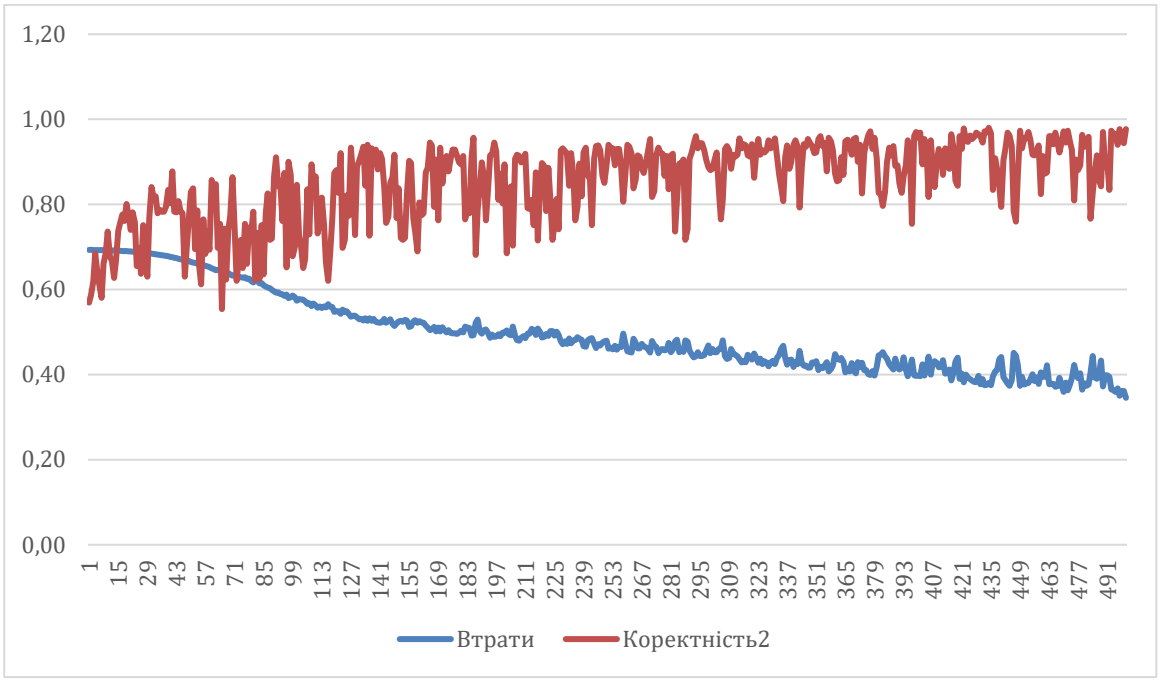

Рисунок 4.18 – Діаграма процесу навчання нейромережі

На діаграмі на осі абсцис вказано епоху навчання нейромережі, на осі ординат задано значення що було на поточну епоху. За графіком втрат, впродовж перших 30 епох йде стабільне значення бінарної кросентропії на рівні «0.69». На даному етапі, нейромережа формує базову порогову функцію, що розділяє вхідні дані на 2 класи. Після формування даної функції, значення втрат починає спад, оскільки нейромережа вже здатна визначити клас, до якого належать дані. Упродовж останніх епох, значення втрат починає коливання, тобто нейромережа остаточно сформувала граничну функцію, та надалі процес навчання полягатиме

у коригуванні граничної функції задля повної відповідності даним. У той же час, графік коректності показує, що нейромережа періодично зустрічає дані, що не може класифікувати коректно через недосконаліть граничної функції. Це є показником ефективності створеного набору навчальних даних.

За розгортання нейронної мережі і зчитування трафіку відповідає модуль ddos_detection. Його мета: перевіряти наявність файлів у директорії “/app/traffic/”. Дані файли мають містити JSON об’єкти запитів. Даний модуль складається з наступних функцій:

1) preprocess_data(log_entry, client_timestamps). Аргументами даної функції є log_entry – JSON об’єкт запиту до сервера та client-timestamps – словник(dictionary, dict) що містить час останнього запиту від клієнта. Функція зчитує дані, що містяться у файлі логів, вираховує міжпакетний інтервал, та нормалізує дані;
2) write_ddos_ip(ip_address, file_path). Аргументами є ip_address – IP-адреса, з якого було надіслано DDoS трафік та file_path – шлях файлу, де мають міститись файл із IP-адресами, які належать зловмиснику. Дана функція створена для запису у файл IP-адрес, з яких надходить DDoS трафік.;
3) classify_traffic(log_entry, client_timestamps, packet_number). Аргумент packet_number – це порядковий номер пакета, що аналізується. Функція подає дані до нейромережі, що аналізу трафік і подає вірогідність пакету, бути DDoS трафіком. Якщо дана вірогідність вища за 0.5(50%), трафік визнається зловмисним, а адреса, з якого надійшов запит, заноситься у відповідний файл;
4) monitor_directory(directory_to_watch). Функція перевіряє директорію на наявність файлів із логами, що необхідно проаналізувати. Коли такий файл знайдено, запускає процес аналізу пакетів.

Запуск модуля здійснюється за допомогою наступного Docker скрипта:

```
FROM python:3.11-slim

RUN apt-get update && apt-get install -y \
    build-essential \
```

```
    && rm -rf /var/lib/apt/lists/*

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip install --no-cache-dir -r requirements.txt
CMD ["python", "ddos_detection.py"]
```

Запустивши модуль ddos_detection, виконання запускається, а модуль стає демоном(*daemon*), що періодично перевіряє директорію на наявність файлу з трафіком. Якщо таку директорію буде знайдено, модель виведе на екран результати сканування, що показано на рисунку 4.19.

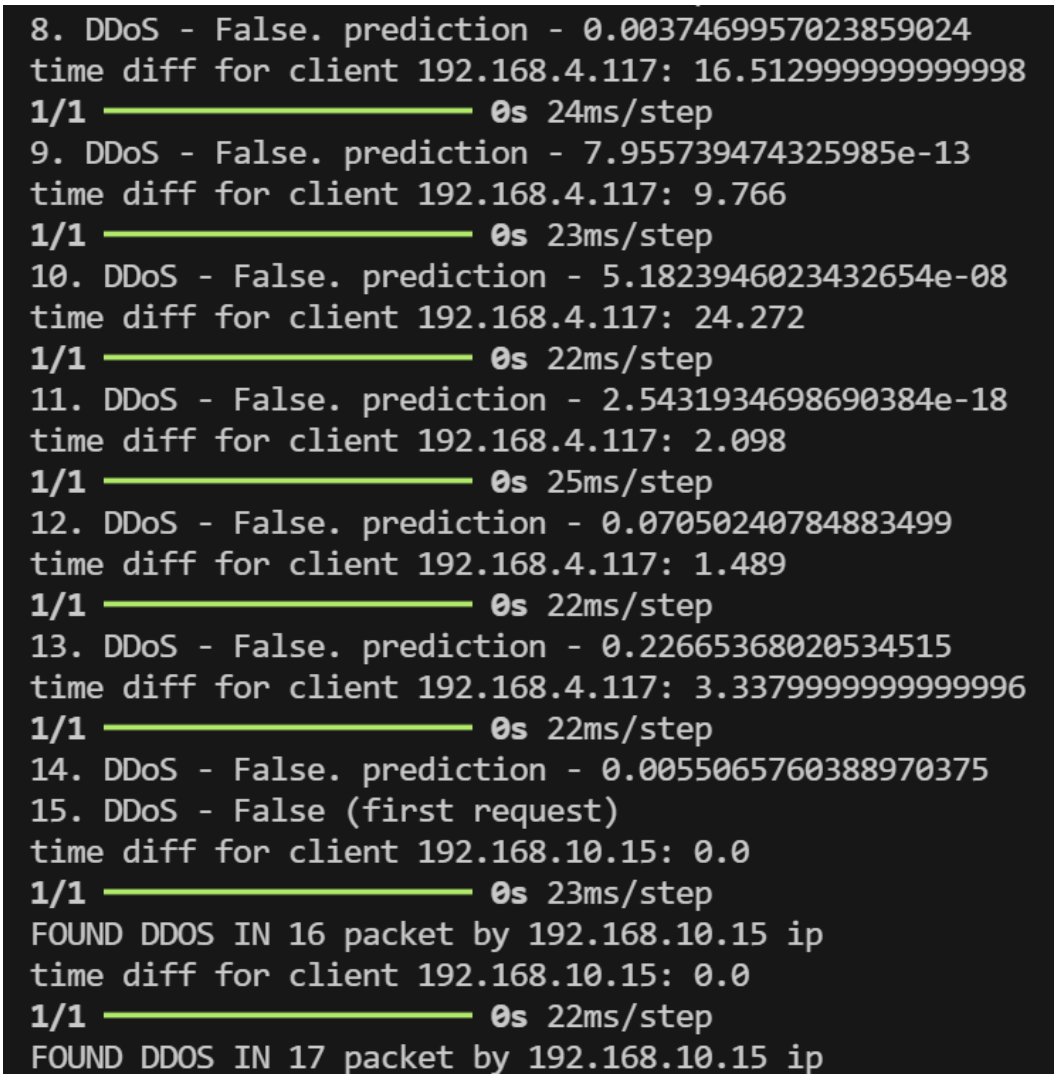

Рисунок 4.19 – Фрагмент результату сканування трафіку

Результати сканування показують, що система успішно справляється з демонстраційним набором даних. Обсяг демонстраційного набору сягає 120 рядків, що є 20% від обсягу навчального набору.

4.4 Впровадження механізму автоматичної реакції системи

У даній роботі буде впроваджено реакцію системи на спробу DDoS трафіку, а саме блокування IP-адреси, з якої було помічено DDoS активність. Для цього необхідно додати відповідний запис до брандмауеру. У даній роботі, це буде методом внесення запису у iptables. Команда блокування трафіку:

iptables -I INPUT -s <ip_adress> -j DROP,

-I – флаг, що вказує на тип трафіку для блокування;

-s – флаг, що вказує IP_адресу для блокування;

-j – Дія, що відбувається з трафіком.

Даний механізм було впроваджено у систему, як файл, що містить bash-скрипт, код якого представлено нижче:

```
#!/bin/bash

IP_FILE="/home/dmykhailenko/AIprj/traffic/ddos_ips.txt"

INTERVAL=10

block_ips() {
    UNIQUE_IPS=$(sort "$IP_FILE" | uniq)

    for IP in $UNIQUE_IPS; do
        if ! sudo iptables -L INPUT -v -n | grep -q "$IP"; then
            echo "Blocking IP: $IP"
            sudo iptables -A INPUT -s "$IP" -j DROP
        fi
    done
}

while true; do
    if [ -f "$IP_FILE" ]; then
        block_ips
    fi
    rm -rf "IP_FILE"
    sleep "$INTERVAL"
done
```

Даний алгоритм перевіряє наявність файлу ddos_ips.txt, повний шлях до якої задано у змінній IP_FILE, кожні 10 секунд. Якщо файл знайдено,

запускається функція block_ips(). У цій функції, з файлу вибирається лист UNIQUE_IPS, що містить унікальні IP-адреси задля виключення можливості повторення. Кожна з адрес, перевіряється по списки iptables. Якщо знайдено запис, що вже регулює правила обміну трафіком з данним вузлом, скрипт не вносить запис до таблиці правил. У іншому випадку до таблиці вноситься запис на заборону вхідного трафіку з цієї IP-адреси, а у stdout передається відповідне повідомлення. Після завершення усіх дій з IP-адресами, файл видаляється задля уникнення проблеми повторів IP-адрес та розростання обсягів цього файлу.

Команда для його запуску:

sudo bash ./blocking_daemon.sh

Під час виконання, скрипт буде оповіщувати яку адресу було заблоковано. Результат виконання можна побачити, виконавши команду iptables -L з правами адміністратора, що показано на рисунку 4.20.

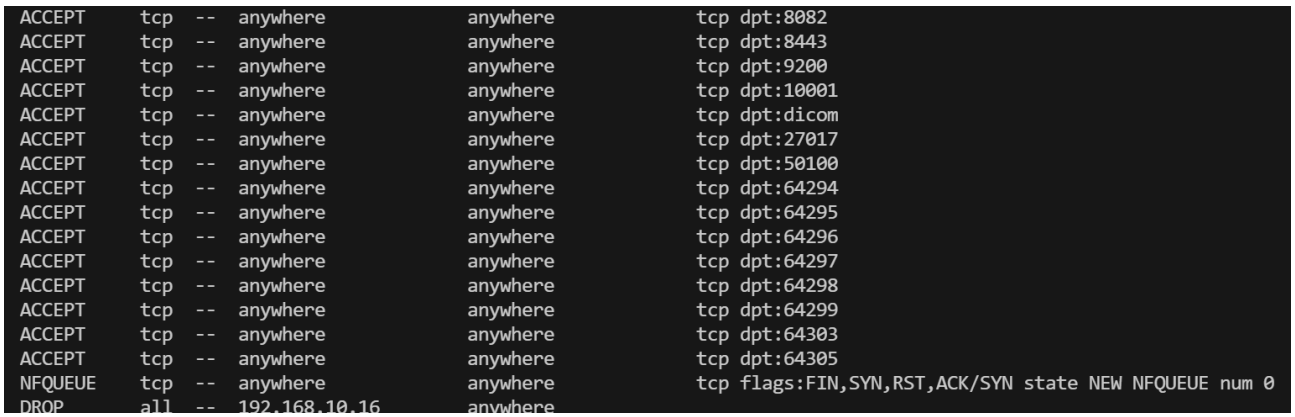

Рисунок 4.20 – Фрагмент записів iptables

4.5 Оцінка результатів роботи дослідної (тестової) системи

Синтезована система здатна до автоматичної реакції на спробу DDoS атаки. На контрольному наборі даних, система захисту виявила усі пакети, що мають ознаку DDoS атаки. Дана нейромережа здатна виявляти зловмисний трафік незалежно від IP адреси, що робить її теоретично здатною виявити спробу реалізації атаки «відмова в обслуговуванні» із задіянням багатьох мережевих пристроїв. Також, ця мережа виявляє трафік незалежно від методу, тобто здатна

виявити зловмисний трафік у разі використання нестандартних методів. Розроблена мережа здатна до аналізу будь-якого трафіку, що записано у форматі JSON, що робить можливим взаємодію з іншими HoneyPot, що заносять дані про отримані запити у форматі JSON.

Проте, діюча версія тестової системи не здатна до виявлення атаки у разі збільшення міжпакетного інтервалу DDoS трафіку, оскільки за часовою оцінкою, такий трафік характеризується як легітимний. Також, система може визнати зловмисним трафік, що зайшов одночасно. Таке може статись при нестабільній роботі мережі, коли клієнт надіслав набір пакетів, але сервер не отримав їх через втрату зв'язку. Під час відновлення зв'язку, клієнт одночасно надсилає усі пакети, на які не отримав відповідь.

Емульована система захисту має ряд принципових недоліків, що можуть призвести до виходу системи з ладу. Найголовніша проблема – роздування log-файлу. Оскільки HoneyPot-и фіксують усі події, що відбуваються, файли із записами подій можуть швидко переповнити увесь доступний простір. Дана проблема частково вирішена самою платформою T-Pot, шляхом обмеження часу зберігання log-файлів. Час зберігання, що встановлено за замовчуванням – 30 днів. Проте цей параметр може бути змінено, а значення має визначатись на основі кількості мережевого трафіку, що поступає на HoneyPot. Ще одним варіантом рішення може бути виділення окремого місця зберігання log-файлів. Таким місцем може виступати окремий вузол мережі або окремий жорсткий диск. Задля більш ефективного використання пам'яті, можливо імплементувати механізм архівації.

Наразі на log-файли не накладається жодного обмеження на багатопоточне читання/запис. Це може стати проблемою при роботі компонентів системи реагування в асинхронному режимі. Під час читання файлу з IP-адресами демоном блокування, модуль аналізу може вносити додаткові IP-адреси до списку, які не будуть заблоковані. Дана проблема може бути вирішена шляхом блокування файлів під час роботи одного з модулів через реалізацію механізму семафорів.

Не менш важливою є проблема швидкості реагування нейромережі на DDoS атаку. Наразі, нейромережа перевіряє наявність файлу із log-записами через заданий інтервал у 20 секунд. У разі спроби зловмисника почати DDoS атаку, нейромережа може не встигнути своєчасно відреагувати на спробу виведення з ладу. Не менш важлива проблема швидкості реагування, оскільки демон(daemon), що блокує трафік, перевіряє наявність файлу з IP-адресами зловмисника у інтервалі 10 секунд. Таким чином, реакція системи може сягати до 30 секунд. Подальшим шляхом покращення є задіяння механізмів прослуховування подій, що активуватимуть механізми аналізу та реакції під час створення або зміни log-файлу. При реалізації такого підходу слід враховувати можливість вичерпання апаратних ресурсів сервера через велику швидкість створення або модифікації log-файлів, а як наслідок, часту кількість запусків процесу аналізу та реакції.

Основною проблемою реакції тестової системи на спробу реалізації атаки типу *«Відмова в обслуговуванні»* є збільшення кількості правил у таблиці брандмауера. При виявленні кожного нового зловмисника, система блокує IP-адресу, з якої відправлено зловмисний трафік. У разі задіяння ботнету(botnet), що складається з 200 пристроїв, до відповідною таблиці буде занесено 200 додаткових правил. Відповідність даним правилам перевіряються для усього трафіку, що надходить у систему, тобто потенційно впливає на легітимний трафік. Таким чином може виникнути ситуація занадто повільної передачі пакетів від клієнтів. Рішенням проблеми може стати оптимізація даної таблиці шляхом об’єднання адрес у CIDR мережі. Ще одним вирішенням є введення часу, впродовж якого діятиме правило на блокування. Таким чином, застарілі правила можуть бути видалені, оскільки інцидент не повторювався, а отже і відповідь системи є неактуальною. Таке правило можливо впровадити, шляхом інтеграції системи із SIEM. Не менш важливим кроком покращення є винесення найбільш часто застосованих правил на початок списку. Таким чином, трафік, що має бути заблоковано, буде заблоковано одразу, без необхідності перевірки інших умов.

ВИСНОВКИ

Аналіз поточного стану загроз ІБ свідчить про зростаючу тенденцію на активне використання штучного інтелекту як засобу скоєння атак. У той же час, *AI* активно імплементується у системи захисту інформаційних систем через можливість покращення та введення автоматизації у процес аналізу та реагування на підозрілі мережеві аномалії. Проте використання нових методів та технологій захисту є недостатнім у разі використання реактивної стратегії реагування на інциденти безпеки. Вдалою альтернативою є використання проактивної стратегії реагування, яка передбачає оперативні поведінкові реакції безпосередньо під час скоєння атаки, а не на етапі ретроспективного аналізу(аудиту подій). Використання даної стратегії дає змогу зібрати більш якісну та актуальну інформацію про методи та цілі зловмисника, надає додаткові можливості протидії.

Ключовим елементом реалізації проактивної стратегії реагування є технологія HoneyPot, що виступає як сервер-пастка для зловмисника. Дана технологія дозволяє у реальному часі детектувати та реєструвати події, які відбуваються з системою, що може бути оперативно використано для виявлення нових методів зламу, збору інформації про сучасні інструменти, якими скористався зловмисник. Також, ця технологія дозволяє будувати різні тактики(сценарії) відбивання нападу, шляхом задіяння певних реакцій системи на дії хакера. Такі дії можуть включати в себе маніпуляції з часом пересилання трафіку, обмеженням трафіку, що передаються у мережі, частковою зміною архітектури мережі. Проте дана технологія може бути використана самим зловмисником, у разі цілеспрямованої атаки на HoneyPot.

Розглянута в роботі концепція введення хакера в оману отримала назву *CyberDeception*, що є еволюційним розширенням розуміння технології HoneyPot. За останні роки спостерігається зріст попиту на дану концепецію, через її ефективність та відносну простоту імплементації в структуру діючих ІС. Проте,

через активне зростання попиту, ця технології введення в оману, зокрема технологія HoneyPot отримала увагу з боку зловмисників, а проста природа даної технології робить її легкою для виявлення.

Методом покращення HoneyPot є залучення штучного інтелекту, що дозволяє створити нові та збільшити ефективність створених раніше можливостей системи, насамперед реакцій системи на спробу зламу. Також залучення штучного інтелекту дозволяє покращити процес збору аналітики та автоматизувати процес обробки результатів. Такий HoneyPot може бути інтегровано у єдину систему реагування на кіберінциденти, як компонент збору та обробки даних.

Залучення системи може відбуватись 2 принципово різними шляхами. Перший – використання штучного інтелекту як модуля прийняття рішення про необхідну дію системи на спробу зламу. У даному випадку, використовується можливість штучного інтелекту до синтезу комплексу дій, що будуть найкраще підходити у конкретній ситуації. Інший шлях – використання штучного інтелекту як модуля зіставлення інцидентів з базою даних випадків. Такий підхід спирається на можливість штучного інтелекту виявляти складні шаблони поведінки та розрізняти контекст конкретного інциденту, а як наслідок, обирати найкращу стратегію реагування з можливих. Визначну роль у коректній реалізації обраного шляху імплементації штучного інтелекту у HoneyPot є коректне визначення типу нейронної мережі, що буде застосовано у системі.

Існуючі наразі HoneyPot-и, що використовують штучний інтелект є обмеженими у конфігурації або неможливими для використання у комерційних системах через відсутність відповідальної особи або недостатню гнучкість налаштування системи. За результатами проведеного моделювання стає очевидним, що якісно новий рівень зручності здатна забезпечити комплексна модель AI HoneyPot, що складається з окремих модулів. Взаємодія між модулями можлива через використання спільного інтерфейсу.

Для демонстрації можливості впровадження AI HoneyPot, була розроблена експериментальна модель фільтрації тестового DDoS трафіку. Дана модель

здатна розрізняти трафік, за принципом віднесення пакетів до 2 класів “нормальний/типовий трафік”, “DDoS трафік”, використовуючи час, що пройшов між запитами від одного клієнта. Для тестування була використана схема 80/20, де 80% тестового набору – дані для навчання нейромережі. 20% складали демонстраційні дані, які застосовуються для перевірки ефективності роботи нейромережі. При цьому додатково впроваджено механізм автоматичної реакції блокування IP-адрес, з яких надійшов тестовий DDoS трафік.

У ході тестування, штучний інтелект виявив усі пакети, що мають ознаку DDoS трафіку, у той же час, правильно визначають легітимний трафік.

За результатами проведеного циклу випробувань можна стверджувати, що, створена тестова система захисту здатна автоматично блокувати типові DDoS атаки. У той же час, система має ряд специфічних недоліків, котрі мають бути виправлені перед імплементацією отриманих напрацювань у реальні інформаційні системи, після відповідних корегувань програмного коду діючої версії прототипу AI HoneyPot.

В цілому, розробка даної системи доводить практичну можливість інтеграції штучного інтелекту у системи класу HoneyPot будь-якої складності. Отримані результати тестування свідчать про ефективність даної технології як дієвого інструменту для суттєвого покращення часових та параметричних зворотних реакцій системи захисту на спробу вторгнення(атаку) в умовну ІКС.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft. Microsoft Digital Defense Report 2024 [Електронний ресурс]: Режим доступу: https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/final/en-us/microsoft-brand/documents/Microsoft%20Digital%20Defense%20Report%202024%20%281%29.pdf
2. Google. Cybersecurity Forecast 2024. Insights for future planning [Електронний ресурс]: Режим доступу: https://services.google.com/fh/files/misc/google-cloud-cybersecurity-forecast-2024.pdf
3. AI phishing [Електронний ресурс] // Google Trends. – Режим доступу: https://trends.google.ru/trends/explore?date=today%205-y&q=AI%20phishing&hl=ru (дата звернення: 01.12.2024)
4. Ahmadi S. Next Generation AI-Based Firewalls / Sina Ahmadi // A Comparative Study. International Journal of Computer (IJC). – 2023. – Vol. 49. – P. 245–262.
5. Sentinel. Artificial Intelligence [Електронний ресурс]: Режим доступу: https://www.sentinel.com/AI
6. Vectra AI Platform - Powered by Attack Signal Intelligence [Електронний ресурс] // Stop Advanced Cyberattacks with Vectra AI. – Mode of access: https://www.vectra.ai/platform (дата звернення: 23.11.2024)
7. Chan A. IDs for AI Systems //arXiv preprint arXiv:2406.12137. – 2024.
8. Kim A. AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection [Electronic resource] / Aechan Kim, Mohyun Park, Dong Hoon Lee // IEEE Access. – 2020. – Vol. 8. – P. 70245–70261. – Mode of access: https://doi.org/10.1109/access.2020.2986882 (дата звернення: 23.11.2024)

9. AI Threat Detection: Leverage AI to Detect Security Threats [Electronic resource] // SentinelOne. – Mode of access: https://www.sentinelone.com/cybersecurity-101/data-and-ai/ai-threat-detection/ (дата звернення: 23.11.2024)
10. What Is the Role of AI in Threat Detection? [Електронний ресурс] // Palo Alto Networks. – Режим доступу: https://www.paloaltonetworks.com/cyberpedia/ai-in-threat-detection (дата звернення: 23.11.2024)
11. Roshanaei M. Enhancing Cybersecurity through AI and ML: Strategies, Challenges, and Future Directions [Електронний ресурс] / Maryam Roshanaei, Mahir R. Khan, Natalie N. Sylvester // Journal of Information Security. – 2024. – Т. 15, № 03. – С. 320–339. – Режим доступу: https://doi.org/10.4236/jis.2024.153019 (дата звернення: 23.11.2024)
12. Artificial intelligence in cyber security: research advances, challenges, and opportunities [Електронний ресурс] / Zhimin Zhang [та ін.] // Artificial Intelligence Review. – 2021. – Режим доступу: https://doi.org/10.1007/s10462-021-09976-0 (дата звернення: 23.11.2024)
13. The Ethical Dilemmas of AI in Cybersecurity [Електронний ресурс] // Cybersecurity Certifications and Continuing Education | ISC2. – Режим доступу: https://www.isc2.org/Insights/2024/01/The-Ethical-Dilemmas-of-AI-in-Cybersecurity (дата звернення: 23.11.2024)
14. Rethinking Active Defense: A Comparative Analysis of Proactive Cybersecurity Policymaking [Електронний ресурс] / Scott J. Shackelford [та ін.] // SSRN Electronic Journal. – 2018. – Режим доступу: https://doi.org/10.2139/ssrn.3303407 (дата звернення: 23.11.2024)
15. Craig A. N. Proactive Cybersecurity: A Comparative Industry and Regulatory Analysis [Електронний ресурс] / Amanda N. Craig, Scott J. Shackelford, Janine S. Hiller // American Business Law Journal. – 2015. – Т. 52, № 4. –

С. 721–787. – Режим доступу: https://doi.org/10.1111/ablj.12055 (дата звернення: 23.11.2024)

16. Smith J. Cybersecurity in the Age of AI: A Proactive Defense Approach / Jane Smith, Patrick Thomas. – [Б. м.] : EasyChair, 2024. – 10 с. – (Препринт / 13306).
17. Stoll C. The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage [Електронний ресурс] / Clifford Stoll, John W. D. Connolly // Physics Today. – 1990. – Т. 43, № 8. – С. 75–76. – Режим доступу: https://doi.org/10.1063/1.2810663 (дата звернення: 23.11.2024)
18. Developments of the Honeyd Virtual Honeypot | Honeyd [Електронний ресурс] // TailBliss. – Режим доступу: https://www.honeyd.org (дата звернення: 23.11.2024)
19. GitHub - desaster/kippo: Kippo - SSH Honeypot [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/desaster/kippo (дата звернення: 23.11.2024)
20. GitHub - cowrie/cowrie: Cowrie SSH/Telnet Honeypot https://cowrie.readthedocs.io [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/cowrie/cowrie (дата звернення: 23.11.2024)
21. GitHub - DinoTools/dionaea: Home of the dionaea honeypot [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/DinoTools/dionaea (дата звернення: 23.11.2024)
22. Amanda Berlin. A Guide To Cybersecurity Deception Techniques – 2021 – Вилучено з: https://www.blumira.com/cybersecurity-deception-techniques/
23. Jeffrey Pawlick and Quanyan Zhu. Deception by Design: Evidence-Based Signaling Games for Network Defense. Вилучено з: https://arxiv.org/pdf/1503.05458v1.pdf
24. Kimberly J. Ferguson-Walter, Maxine M. Major, Chelsea K. Johnson, Daniel H. Muhleman. Examining the Efficacy of Decoy-based and Psychological Cyber Deception. Вилучено з: https://www.usenix.org/system/files/sec21-ferguson-walter.pdf

25. MITRE. Shields Up: A Good Cyber Defense Is an Active Defense – 2020 – Вилучено з: https://www.mitre.org/news-insights/impact-story/shields-good-cyber-defense-active-defense
26. Citrix Ready Marketplace. Attivo Networks ThreatDefend Platform. Вилучено з: https://citrixready.citrix.com/attivo-networks/threatdefend-platform.html
27. CounterCraft. Beyond Threat Detection and Response | One Step Ahead of Attackers. Вилучено з: https://www.countercraftsec.com
28. Usage statistics for Honeypot[Електронний ресурс] // Drupal. – Режим доступу: https://www.drupal.org/project/usage/honeypot (дата звернення: 23.11.2024)
29. Usage statistics for Drupal core[Електронний ресурс] // Drupal. – Режим доступу: https://www.drupal.org/project/usage/drupal (дата звернення: 23.11.2024)
30. Михайленко Д. Д. МОЖЛИВОСТІ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В АЛГОРИТМАХ РОБОТИ МЕРЕЖЕВИХ ПАСТОК ТА СПОСОБИ ЙОГО ІМПЛЕМЕНТАЦІЇ В СУЧАСНІ HONEYPOT / Дмитро Денисович Михайленко, В. Чудновський, С. Малахов // Розвиток суспільства та науки в умовах цифрової трансформації матеріали VII Міжнародноїстудентської наукової конференції,м.Тернопіль, 15 листопада, 2024рік / ГО «Молодіжна наукова ліга».— Вінниця: ТОВ «УКРЛОГОСГруп», 202 4. — 508с. – 2024
31. Honeypot Or Not? [Електронний ресурс] // Shodan. – Режим доступу: https://honeyscore.shodan.io (дата звернення: 23.11.2024)
32. GitHub - honeynet/checkpot: Checkpot Honeypot Checker [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/honeynet/checkpot (дата звернення: 23.11.2024)
33. GitHub - UnaPibaGeek/honeypots-detection: Nuclei templates for honeypots detection. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/UnaPibaGeek/honeypots-detection (дата звернення: 23.11.2024)

34. How OSINT is used in cybersecurity? - Part Two | ioSENTRIX [Електронний ресурс] // Cybersecurity Solutions | Penetration Testing, DevSecOps, AppSec Services | ioSENTRIX. – Режим доступу: https://www.iosentrix.com/blog/how-osint-is-used-in-cybersecurity-part-2 (дата звернення: 23.11.2024)
35. Adamov A. Искусственный интеллект в кибербезопасности [Електронний ресурс] / Alexander Adamov // dou.ua. – Режим доступу: https://dou.ua/lenta/articles/ai-in-cybersecurity/ (дата звернення: 23.11.2024)
36. Zanoramy Ansiry Zakaria W. A review on artificial intelligence techniques for developing intelligent honeypot, / W. Z. Ansiry Zakaria, M. L. M. Kiah // 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT), Seoul, 26 квіт. 2012 р. – [Б. м.]. – С. 696–701.
37. Zanoramy Ansiry Zakaria W. A review of dynamic and intelligent honeypots [Електронний ресурс] / Wira Zanoramy Ansiry Zakaria, Miss Laiha Mat Kiah // ScienceAsia. – 2013. – Т. 39S, № 1. – С. 1. – Режим доступу: https://doi.org/10.2306/scienceasia1513-1874.2013.39s.001 (дата звернення: 23.11.2024)
38. Dowling S. New framework for adaptive and agile honeypots [Електронний ресурс] / Seamus Dowling, Michael Schukat, Enda Barrett // ETRI Journal. – 2020. – Т. 42, № 6. – С. 965–975. – Режим доступу: https://doi.org/10.4218/etrij.2019-0155 (дата звернення: 23.11.2024)
39. AI-Driven Honeypots: The Future of Cyber Defense [Електронний ресурс] // RedTeam Cybersecurity Labs. – Режим доступу: https://theredteamlabs.com/cybersecurity-with-ai-driven-honeypots/ (дата звернення: 23.11.2024)
40. Anomaly Detection in Network Traffic using Deep Learning / K. Sharma [та ін.] // International Conference on Recent Advances in Science and Engineering Technology (ICRASET), B G NAGARA, 23.11. 2023 p

41. A Highly Interactive Honeypot-Based Approach to Network Threat Management / X. Yang [et al.] // Future Internet, Beijing, 28 March 2023.
42. Intelligent Threat Detection-AI-Driven Analysis of Honeypot Data to Counter Cyber Threats. / Lanka P, Gupta K, Varol C // Electronics – 2024. https://doi.org/10.3390/electronics13132465
43. Bebis G. Feed-forward neural networks / G. Bebis, M. Georgiopoulos // IEEE Potentials. – 1994. – Т. 13, № 4. – С. 27–31.
44. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects / Z. Li [та ін.] // IEEE Transactions on Neural Networks and Learning Systems. – 2022. – Т. 33, № 12. – С. 6999–7019.
45. Fischer A. An Introduction to Restricted Boltzmann Machines / A. Fischer, C. Igel // Springer, Berlin, 8 листоп. 2012 р
46. Recurrent Neural Networks [Електронний ресурс] / ред.: L. Medsker, L. C. Jain. – [Б. м.] : CRC Press, 1999. – Режим доступу: https://doi.org/10.1201/9781420049176 (дата звернення: 23.11.2024)
47. A survey of transformers [Електронний ресурс] / Tianyang Lin [та ін.] // AI Open. – 2022. – Режим доступу: https://doi.org/10.1016/j.aiopen.2022.10.001 (дата звернення: 23.11.2024)
48. Generative adversarial networks [Електронний ресурс] / Ian Goodfellow [та ін.] // Communications of the ACM. – 2020. – Т. 63, № 11. – С. 139–144. – Режим доступу: https://doi.org/10.1145/3422622 (дата звернення: 23.11.2024)
49. Кохановська Т., Нарежний О., Дьяченко О. (2020). Дослідження можливостей технології Honeypot. https://doi.org/10.26565/2519-2310-2020-1-03
50. Mykhailenko D. Specifics of AI implementation in modern Honeypots / Dmytro Mykhailenko, I. Haltseva, S. Malakhov // Proceedings of the XII International Scientific and Practical Conference, Rotterdam, 19–22 листоп. 2024 р. – [Б. м.]. – С. 368–372. URL: https://isg-konf.com/prospective-directions-of-

modern-science-and-education-in-the-world/ Available at: DOI: 10.46299/ISG.2024.2.12

51. NeuralPot: An Industrial Honeypot Implementation Based On Deep Neural Networks [Електронний ресурс] / Ilias Siniosoglou [та ін.] // 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 лип. 2020 р. – [Б. м.], 2020. – Режим доступу: https://doi.org/10.1109/iscc50000.2020.9219712 (дата звернення: 23.11.2024)
52. SPHINX Toolkit Components Development - Artificial Intelligence (AI) Honeypot [Електронний ресурс] // Cyberwatching. – Режим доступу: https://www.cyberwatching.eu/projects/2109/sphinx/videos/sphinx-toolkit-components-development-artificial-intelligence-ai-honeypot (дата звернення: 23.11.2024)
53. Ghost Security- Platform [Електронний ресурс] // Ghost | Supernatural Application Security. – Режим доступу: https://ghostsecurity.com/platform (дата звернення: 23.11.2024)
54. GitHub - mrwadams/honeyagents: HoneyAgents is a PoC demo of an AI-driven system that combines honeypots with autonomous AI agents to detect and mitigate cyber threats. Features include intelligent threat analysis, automated deny list updates, and detailed natural language threat reports. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/mrwadams/honeyagents (дата звернення: 23.11.2024)
55. Михайленко Д. ВИКОРИСТАННЯ МОЖЛИВОСТЕЙ AI ПРИ РЕАЛІЗАЦІЇ STATIC ТА DYNAMIC HONEYPOT ДЛЯ ПОКРАЩЕННЯ ПАРАМЕТРІВ ЗАХИСТУ ІНФОРМАЦІЙНИХ РЕСУРСІВ. Михайленко Д., Чорна Т. Технології, інструменти та стратегії реалізації наукових досліджень: матеріали IV Міжнародної наукової конференції, м. Суми, 7 жовтня, 2022 р. / Міжнародний центр наукових досліджень. — Вінниця:

Європейська наукова платформа, 2022. — 142 с. DOI10.36074/mcnd-07.10.2022

56. GitHub - telekom-security/tpotce: T-Pot - The All In One Multi Honeypot Platform [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/telekom-security/tpotce (дата звернення: 01.12.2024).
57. GitHub - smicallef/spiderfoot: SpiderFoot automates OSINT for threat intelligence and mapping your attack surface. [Електронний ресурс] // GitHub. – Режим доступу: https://github.com/smicallef/spiderfoot (дата звернення: 01.12.2024).
58. Liu D. A Practical Guide to ReLU [Електронний ресурс] / Danqing Liu // Medium. – Режим доступу: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7 (дата звернення: 01.12.2024)
59. The sigmoid function (a.k.a. the logistic function) and its derivative [Електронний ресурс] // Is That All There Is?. – Режим доступу: https://hvidberrrg.github.io/deep_learning/activation_functions/sigmoid_function_and_derivative.html (дата звернення: 01.12.2024)

ДОДАТОК А

Лістинг коду процедури/процесу перемішування трафіку для створення набору даних *(власна розробка)*

```
import random
import json

def read_json_lines(file_path):
    with open(file_path, 'r') as f:
        return [json.loads(line) for line in f]

def write_json_lines(file_path, json_lines):
    with open(file_path, 'w') as f:
        for line in json_lines:
            f.write(json.dumps(line) + '\n')

def insert_json_objects_randomly(main_file, insert_file, output_file):
    # Read JSON objects from both files
    main_json_lines = read_json_lines(main_file)
    insert_json_lines = read_json_lines(insert_file)

    # Shuffle the JSON objects to be inserted
    random.shuffle(insert_json_lines)

    # Insert each JSON object from the insert file into random positions
in the main file
    for insert_json in insert_json_lines:
        insert_position = random.randint(0, len(main_json_lines))
        main_json_lines.insert(insert_position, insert_json)

    # Write the combined JSON objects to the output file
    write_json_lines(output_file, main_json_lines)

if __name__ == "__main__":
    main_file =
"C:\\Users\\dmytro.mykhailenko_h\\AI\\ddos_analyzer\\log\\tanner_report.j
son"
    insert_file =
"C:\\Users\\dmytro.mykhailenko_h\\AI\\ddos_analyzer\\log\\DDOS_report.jso
n"
    output_file =
"C:\\Users\\dmytro.mykhailenko_h\\AI\\ddos_analyzer\\log\\combined_output
.json"
    insert_json_objects_randomly(main_file, insert_file, output_file)
    print(f"JSON objects from '{insert_file}' have been inserted into
'{main_file}' and saved to '{output_file}'.")
```

ДОДАТОК Б

Лістинг коду процесу навчання нейронної мережі *(власна розробка)*

```
import json
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix
from datetime import datetime
from keras.callbacks import Callback

class MetricsLogger(Callback):
    def __init__(self, loss_file='loss.txt',
accuracy_file='accuracy.txt'):
        super().__init__()
        self.loss_file = loss_file
        self.accuracy_file = accuracy_file

    def on_epoch_end(self, epoch, logs=None):
        logs = logs or {}
        loss = logs.get('loss')
        accuracy = logs.get('accuracy')
        if loss is not None:
            with open(self.loss_file, 'a') as lf:
                lf.write(f"{loss}\n")
        if accuracy is not None:
            with open(self.accuracy_file, 'a') as af:
                af.write(f"{accuracy}\n")

def preprocess_training_data(training_log_file):
    # Load and preprocess the data
    with open(training_log_file, 'r') as f:
        data = [json.loads(line) for line in f]

    X = []
    y = []
    client_timestamps = {}  # Dictionary to store the last timestamp for
each client
    for entry in data:
        features = extract_features(entry, client_timestamps)
        if features is not None:
            X.append(features)
            y.append(determine_label(entry))

    X = np.array(X)
    y = np.array(y)

    # Normalize data
    scaler = StandardScaler()
```

```
    X = scaler.fit_transform(X)

    return X, y, scaler

def extract_features(log_entry, client_timestamps):
    client_ip = log_entry['peer']['ip']

    # Calculate time difference between requests by the same client
    current_timestamp = datetime.strptime(log_entry['timestamp'], '%Y-%m-
%dT%H:%M:%S.%f')
    time_diff = 0
    if client_ip in client_timestamps:
        previous_timestamp = client_timestamps[client_ip]
        time_diff = abs((current_timestamp -
previous_timestamp).total_seconds())  # Use abs to ensure non-negative
values
        print(f"Timediff for client {client_ip}: {time_diff}")
    else:
        # If it is the first request, return None to indicate no time
difference
        client_timestamps[client_ip] = current_timestamp
        return None

    # Update the last timestamp for the client
    client_timestamps[client_ip] = current_timestamp

    feature_vector = [
        time_diff
    ]
    return feature_vector

def determine_label(log_entry):
    # Use the is_ddos field to determine the label, default to 0 if not
present
    return log_entry.get("is_ddos", 0)

def build_model(input_dim):
    model = Sequential([
        Dense(256, input_dim=input_dim, activation='relu'),
        Dense(128, activation='relu'),
        Dense(64, activation='relu'),
        Dense(32, activation='relu'),
        Dense(16, activation='relu'),
        Dense(8, activation='relu'),
        Dense(4, activation='relu'),
        Dense(2, activation='relu'),
        Dense(1, activation='sigmoid')  # For binary classification
    ])
    optimizer = Adam(learning_rate=0.0001)
    model.compile(optimizer=optimizer, loss='binary_crossentropy',
metrics=['accuracy'])
```

```
    return model

def train_and_save_model(training_log_file):
    X, y, scaler = preprocess_training_data(training_log_file)
    model = build_model(input_dim=X.shape[1])

    # Initialize the metrics logger callback
    metrics_logger = MetricsLogger()

    model.fit(X, y, epochs=500, batch_size=64,
callbacks=[metrics_logger])

    # Evaluate the model
    y_pred = (model.predict(X) > 0.5).astype("int32")
    print(classification_report(y, y_pred))
    print(confusion_matrix(y, y_pred))

    model.save('ddos_model.h5')
    with open('scaler.json', 'w') as f:
        json.dump({'mean_': scaler.mean_.tolist(), 'scale_':
scaler.scale_.tolist()}, f)

    print("Model successfully saved to 'ddos_model.h5'.")

if __name__ == "__main__":
    training_log_file = "./log/tanner_report.json"
    train_and_save_model(training_log_file)
```

ДОДАТОК В

Лістинг коду процедури аналізу трафіку на наявність DDoS атаки *(власна розробка)*

```
import os
import time
import json
import numpy as np
from keras.models import load_model
from sklearn.preprocessing import StandardScaler
from datetime import datetime

# Load the trained model and preprocessors
model = load_model('ddos_model.h5')

with open('scaler.json', 'r') as f:
    scaler_data = json.load(f)
scaler = StandardScaler()
scaler.mean_ = np.array(scaler_data['mean_'])
scaler.scale_ = np.array(scaler_data['scale_'])

# Function to preprocess data
def preprocess_data(log_entry, client_timestamps):
    client_ip = log_entry['peer']['ip']

    # Calculate time difference between requests by the same client
    current_timestamp = datetime.strptime(log_entry['timestamp'], '%Y-%m-
%dT%H:%M:%S.%f')
    time_diff = 0
    if client_ip in client_timestamps:
        previous_timestamp = client_timestamps[client_ip]
        time_diff = abs((current_timestamp -
previous_timestamp).total_seconds()*1000)  # Use abs to ensure non-
negative values
        print(f"time diff for client {client_ip}: {time_diff}")
    else:
        # If it is the first request, return None to indicate no time
difference
        client_timestamps[client_ip] = current_timestamp
        return None

    # Update the last timestamp for the client
    client_timestamps[client_ip] = current_timestamp

    features = np.array([[time_diff]])
    features_scaled = scaler.transform(features)
    return features_scaled

def write_ddos_ip(ip_address, file_path="./traffic/ddos_ips.txt"):
    with open(file_path, 'a') as f:
```

```
        f.write(f"{ip_address}\n")

# Function to classify traffic
def classify_traffic(log_entry, client_timestamps, packet_number):
    features = preprocess_data(log_entry, client_timestamps)
    if features is None:
        # If it is the first request, return "DDoS - False" without using
the neural network
        print(f"{packet_number}. DDoS - False (first request)")
        return log_entry
    prediction = model.predict(features)
    is_ddos = prediction[0][0] > 0.5  # Threshold for DDoS classification
    if is_ddos:
        client_ip = log_entry['peer']['ip']
        print(f"FOUND DDOS IN {packet_number} packet by
{log_entry['peer']['ip']} ip")
        write_ddos_ip(client_ip)  # Write the IP address to the file
    else:
        print(f"{packet_number}. DDoS - False. prediction -
{prediction[0][0]}")
    return log_entry

# Function to monitor directory and process files
def monitor_directory(directory_to_watch):
    client_timestamps = {}
    packet_number = 0
    while True:
        for filename in os.listdir(directory_to_watch):
            file_path = os.path.join(directory_to_watch, filename)
            if os.path.isfile(file_path):
                with open(file_path, 'r') as f:
                    for line in f:
                        log_entry = json.loads(line)
                        packet_number += 1
                        classify_traffic(log_entry, client_timestamps,
packet_number)
                os.remove(file_path)
                client_timestamps = {}
                packet_number = 0
        time.sleep(20)

if __name__ == "__main__":
    directory_to_watch = "./traffic"
    monitor_directory(directory_to_watch)
```