Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи магістра

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

освітньо-наукова програма *Кібербезпека*
(назва освітньої програми)

на тему: «Модель ідентифікації інцидентів кібербезпеки»

Виконавець: студента ІІ курсу, групи КБм-21

(підпис) **Олег Шалатонов** (Ім'я, ПРІЗВИЩЕ)

	Ім'я, ПРІЗВИЩЕ	**Підпис**
Науковий керівник	Тетяна БАБЕНКО	
Нормоконтроль	Сергій ДАКОВ	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри
кібербезпеки
та захисту інформації
_____ Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

освітній ступень *магістр*

Здобувача(ки) КБм-21 (група) Шалатонова Олега Руслановича (прізвище ім'я по-батькові)

Тема дипломної роботи Модель ідентифікації інцидентів кібербезпеки

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Рішення засідання кафедри кібербезпеки та захисту інформації факультету інформаційних технологій протокол № 3 від 20.10.2022

2. МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень	Процес ідентифікації інцидентів кібербезпеки за допомогою інтелектуальних моделей.
Предмет досліджень	Моделі класифікації подій кібербезпеки на основі нейронних мереж.
Мета	Розробка моделі захисту від кібератак використанням нейронних мереж..
Вихідні дані для проведення роботи	Моделі класифікації подій кібербезпеки на основі нейронних мереж.

3. ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна Створені нетипові моделі ідентифікації загроз кібербезпеки, що відрізняються від сучасних аналогів способом аналізу даних, швидкодією та представленням.

Практична цінність Покращення достовірності ідентифікації загрози в сукупності з уже існуючими засобами захисту.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Робота виконана у повному обсязі відповідно до теми.

5. ЕТАПИ ВИКОНАННЯ РОБОТИ

Найменування етапів робіт	Строки виконання робіт (початок-кінець)
Розробка плану для досягнення мети роботи	24.10.2022 – 23.01.2023
Аналіз літературних джерел	24.01.2023 – 14.02.2023
Розробка моделі захисту від кібератак використанням нейронних мереж	15.02.2023 – 24.04.2023
Оформлення і друк пояснювальної записки	25.04.2023 – 19.05.2023

6. РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект Зниження збитків через швидке реагування на кіберінциденти.

Соціальний ефект Покращення технологій забезпечення захисту інформації корпоративного рівня.

7. ДОДАТКОВІ ВИМОГИ

Завдання видав _____ Тетяна БАБЕНКО
(підпис) (Ім'я, ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Олег ШАЛАТОНОВ
(підпис) (Ім'я, ПРІЗВИЩЕ)

Дата видачі завдання: 24.10.2022 р.
Термін подання кваліфікаційної роботи до ЕК 19.05.2023 р.

УДК. 004.432.16

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Модель ідентифікації інцидентів кібербезпеки»: 77 сторінка, 16 рисунків та 8 таблиць, 54 літературних джерела.

Об’єкт дослідження – процес ідентифікації інцидентів кібербезпеки за допомогою інтелектуальних моделей.

Мета роботи – розробка моделі захисту від кібератак використанням нейронних мереж.

Методи дослідження – імітаційне моделювання.

У роботі досліджено вітчизняну та зарубіжну літературу в сфері ідентифікації інцидентів кібербезпеки. Проведено аналіз наявних моделей ідентифікації інцидентів, класифікації подій кібернетичної безпеки. Розроблені моделі ідентифікації мережевих атак використання нейронних мереж з алгоритмом зворотного розповсюдження помилки на базі багатошарового перцептрону.

Наукова новизна: у роботі створені нетипові моделі ідентифікації загроз інформаційної безпеки, що відрізняються від сучасних аналогів способом аналізу даних, швидкодією та представленням.

Актуальність теми: використання штучного інтелекту та машинного навчання в сфері кібербезпеки зростає, так само і кількість кіберінцидентів кожен рік стає все більше, тактики та засоби зловмисників видозмінюються, проте патерни залишаються незмінними. Існуючі моделі потребують оновлених тренувальних датасетів та підходів до навчання. Тому запропонована модель ідентифікації інцидентів кібербезпеки створена на основі оновленої версії датасету, відрізняється способом аналізу даних, швидкодією та представленням.

Ключові слова: машинне навчання, нейронні мережі, інциденти кібербезпеки, перцептрон Розенблатта, зловмисний мережевий трафік, SQL ін’єкції.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API	–	Application Programming Interface
APT	–	Advanced Persistent Threat
BEC	–	Business Email Compromise
C&C / C2	–	Command and Control
DNS	–	Domain Name System
(D)DoS		(Distributed) Denial-of-Service attack
EDR	–	Endpoint Detection and Response
FTP		File Transfer Protocol
HTTP	–	HyperText Transfer Protocol
IDS	–	Intrusion Detection System
IOC	–	Indicators of compromise
IP	–	Internet Protocol
KNN		K-Nearest Neighbors algorithm
NDR	–	Network Detection and Response
SIEM	–	Security information and event management
SOC	–	Security Operations Center
SVM	–	Support Vector Machine
URL	–	Uniform Resource Locator
WAF	–	Web Application Firewall
МН	–	Машинне навчання
ПЗ	–	Програмне забезпечення
ШІ	–	Штучний інтелект

ЗМІСТ

ВСТУП

Сьогоднішній світ неможливо уявити без інформаційних технологій, які стали інтегральною частиною нашого життя. Завдяки розвитку інтернету та мережевої інфраструктури, люди та компанії спілкуються та обмінюються даними, а також отримують доступ до важливої інформації. Однак, це відкриває двері для кіберзлочинців, які намагаються скомпрометувати інформацію та завдати шкоди. З цієї причини, ідентифікація та класифікація кіберінцидентів є важливим аспектом кібербезпеки.

Моделі машинного навчання (МН) допомагають виявляти та класифікувати кіберінциденти, оскільки вони можуть автоматично вивчати та аналізувати великі обсяги даних про мережевий трафік, системні події та інші джерела інформації.

Обсяг ринку штучного інтелекту в галузі кібербезпеки оцінюють у 22,4 млрд доларів США у 2023 році і, за прогнозами, він становитиме 60,6 млрд доларів США до 2028 року; щорічний темп зростання становитиме 21,9% [1].

Такі фактори, як зростаюче впровадження IoT і збільшення кількості підключених пристроїв, збільшення кількості випадків кіберзагроз, зростаюча заклопотаність захистом даних стимулюють зростання галузі штучного інтелекту в кібербезпеці.

Моделі МН для класифікації кіберінцидентів можуть використовувати різні підходи, зокрема, контрольоване, неконтрольоване та напівконтрольоване навчання. Вони можуть використовувати різні методи, такі як логістична регресія, дерева рішень, опорні векторні машини (SVM), нейронні мережі та ансамблеві методи, для того, щоб навчитися визначати патерни, характерні для кібератак.

Процес класифікації кіберінцидентів за допомогою моделей МН включає кілька етапів. Спочатку збираються та обробляються дані з різних джерел, таких як мережеві журнали, системні події та інші індикатори компрометації. Далі, дані піддаються передопрацюванню та виділенню ознак, що можуть допомогти моделям виявляти закономірності та патерни, характерні для кіберінцидентів.

Після того, як дані підготовлено, вибираються відповідні моделі машинного навчання, які будуть навчені на основі цих даних. Різні моделі можуть мати свої переваги та недоліки, тому важливо обрати найкращу модель для конкретної задачі. Після навчання моделі, їх можна використовувати для класифікації нових кіберінцидентів у режимі реального часу.

Однак, використання моделей МН для класифікації кіберінцидентів має свої виклики. Моделі можуть потребувати великих обсягів даних для навчання, а також потрібно забезпечити якість та актуальність даних. Крім того, кіберзлочинці постійно розробляють нові атаки, тому моделі повинні бути гнучкими та здатними адаптуватися до нових загроз.

Актуальність обраної теми зумовлена тим, що використання штучного інтелекту та машинного навчання в сфері кібербезпеки зростає, так само і кількість кіберінцидентів кожен рік стає все більше, тактики та засоби зловмисників видозмінюються, проте патерни залишаються незмінними.

Метою роботи є розробка моделі захисту від кібератак використанням нейронних мереж.

Об’єктом дослідження є процес ідентифікації інцидентів кібербезпеки за допомогою інтелектуальних моделей.

Предметом дослідження є моделі класифікації подій кібербезпеки на основі нейронних мереж.

Методом дослідження виступає імітаційне моделювання.

Практична цінність отриманих результатів полягає в розробці інтелектульних моделей ідентифікації кіберінцидентів шляхом класифікації мережних подій, що можуть використовуватися в подальшому як підсистеми корпоративних систем захисту, таких як WAF (Web Application Firewall) та SIEM системи. Використання розроблених моделей може покращити достовірність ідентифікації загрози в сукупності з уже існуючими засобами захисту.

Наукова новизна дослідження. У роботі створені нетипові моделі ідентифікації загроз інформаційної безпеки, що відрізняються від сучасних аналогів способом аналізу даних, швидкодією та представленням.

РОЗДІЛ 1
СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз використання машинного навчання в кібербезпеці

Машинне навчання та штучний інтелект (ШІ) стали невід'ємними компонентами систем кібербезпеки, допомагаючи виявляти, запобігати та реагувати на загрози. Ось деякі ключові аспекти їх використання в кібербезпеці, а також відповідна статистика і прогнози:

1. Виявлення та аналіз загроз:

Алгоритми машинного навчання можуть аналізувати великі обсяги даних для виявлення закономірностей, аномалій та індикаторів потенційних кіберзагроз.

Згідно зі звітом MarketsandMarkets, очікується, що до 2028 року світовий ринок штучного інтелекту в кібербезпеці досягне 61 мільярда доларів США, зростаючи на 22% в середньорічному обчисленні протягом 2023-2028 років [1].

2. Виявлення та запобігання вторгненням:

Системи виявлення вторгнень (IDS) на основі штучного інтелекту можуть виявляти та реагувати на складні атаки, включаючи експлойти нульового дня.

Дослідження Стенфордського університету показало, що системи виявлення вторгнень на основі штучного інтелекту знизили кількість помилкових спрацьовувань на 70%, що дозволило аналітикам з безпеки зосередитися на справжніх загрозах.

3. Виявлення та запобігання шкідливим програмам:

Алгоритми машинного навчання можуть аналізувати характеристики файлів, поведінку та мережевий трафік, щоб виявити шкідливе програмне забезпечення та заблокувати його виконання.

Згідно зі звітом Statista, до 2027 року світовий ринок ШІ в кібербезпеці досягне $46,3 млрд, хоча в 2020 році це було лише 10 млрд доларів США [2].

4. Аналіз поведінки користувачів:

ШІ може встановлювати базові лінії нормальної поведінки користувачів і виявляти відхилення, які можуть свідчити про зловмисну діяльність, наприклад, інсайдерські загрози або захоплення облікових записів.

Опитування, проведене компанією ThoughtLab, показало, що 26% серед 1200 великих організацій використовують ШІ та машинне навчання для виявлення та реагування на внутрішні загрози [3].

5. Автоматизоване реагування на інциденти:

Системи безпеки на основі штучного інтелекту можуть автоматизувати процеси реагування на інциденти, дозволяючи швидше локалізувати та пом'якшити загрози.

6. Предиктивна аналітика та розвідка загроз:

Моделі машинного навчання можуть аналізувати історичні дані та інформацію в режимі реального часу, щоб генерувати прогностичні висновки, допомагаючи організаціям проактивно захищатися від нових загроз.

Відповідно до звіту IBM [4] серед 1000 опитаних компаній 64% використовують ШІ для задач інформаційної та кібербезпеки. І вони виділили 5 найголовніші задачі, у яких штучний інтелект їм допомагає. Результат зображений на рисунку 1.1

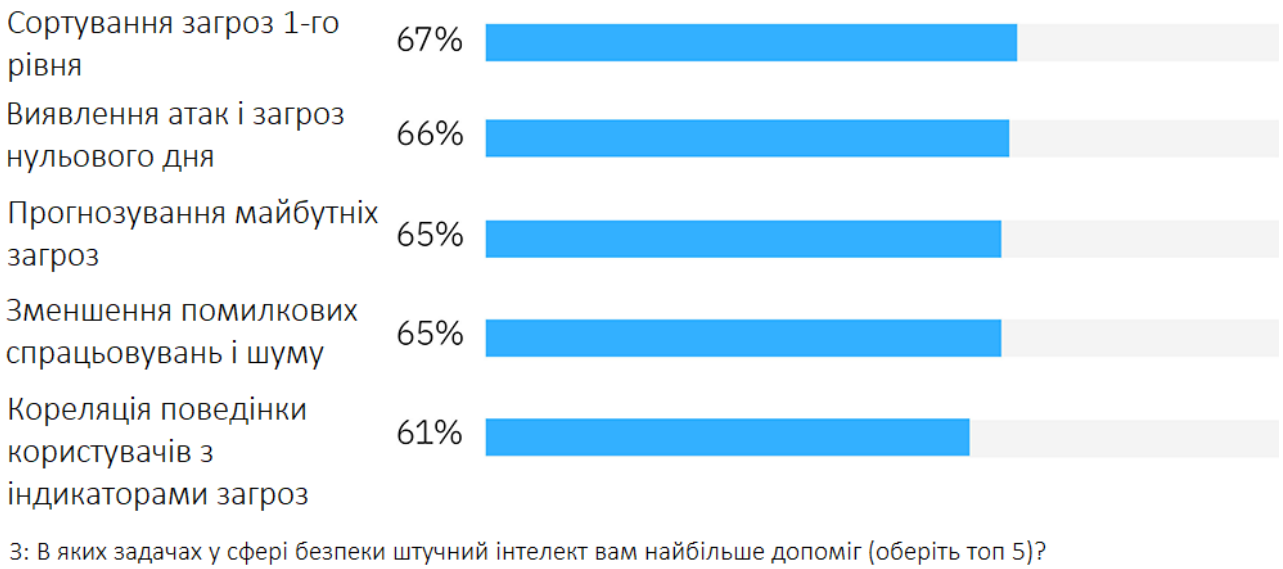


Рисунок 1.1 – Топ 5 задач, в яких допомагає штучний інтелект [4]

Хоча впровадження машинного навчання та штучного інтелекту в кібербезпеці пропонує значні переваги, важливо забезпечити надійність та етичність використання цих технологій. Постійні дослідження, співпраця та моніторинг необхідні для усунення потенційних вразливостей і забезпечення ефективності рішень у сфері кібербезпеки на основі ШІ.

1.2 Інциденти кібербезпеки та їх ідентифікатори

Згідно закону України «Про основні засади забезпечення кібербезпеки України» кіберінцидент - подія або ряд несприятливих подій ненавмисного характеру (природного, технічного, технологічного, помилкового, у тому числі внаслідок дії людського фактора) та/або таких, що мають ознаки можливої (потенційної) кібератаки, які становлять загрозу безпеці систем електронних комунікацій, систем управління технологічними процесами, створюють імовірність порушення штатного режиму функціонування таких систем (у тому числі зриву та/або блокування роботи системи, та/або несанкціонованого управління її ресурсами), ставлять під загрозу безпеку (захищеність) електронних інформаційних ресурсів [5].

1.2.1 Категорії кіберінцидентів

Державна служба спеціального зв'язку та захисту інформації України розробила перелік категорій кіберінцидентів, використовуючи рекомендації Європейської агенції з кібербезпеки (ENISA Reference Incident Classification Taxonomy, січень 2018 року) [7], а також спільний документ ENISA та Європейського центру боротьби з кіберзлочинністю Європолу (Common Taxonomy for Law Enforcement and The National Network of CSIRTs) [6, 8]. Не всі категорії інцидентів стосуються даної роботи, тому вони виключені зі списку, що представлений у таблиці 1.1.

Завжди існує ризик того, що загрози будуть пропущені серед потоків хибних спрацьовувань або просто прослизнуть крізь прогалини в видимості. Щоб побудувати дійсно комплексну систему кібербезпеки, аналітики Gartner пропонують застосувати

так званий підхід SOC Visibility Triad [9] (рисунок 1.2). Ця тріада видимості, що складається з SIEM, системи виявлення та реагування на події на кінцевих точках (EDR) та системи виявлення та реагування на події в мережі (NDR), дозволяє усунути слабкі місця та підвищити рівень кібербезпеки за допомогою співпраці.

Таблиця 1.1

Вибірка з переліку категорій кіберінцидентів

Код	Категорія інциденту	Назва інциденту
02	Шкідливий програмний код (Malicious Code)	Вірус
		Хробак
		Троян
		Шпигунське програмне забезпечення
		Діалер
		Руткіт
		Шкідливе програмне забезпечення
		Управління ботами
		Конфігурація шкідливого програмного забезпечення
		Програма-здирник
		Командно-контрольний центр
03	Збір інформації зловмисником (Information Gathering)	Сканування
		Перехоплення і аналіз мережевого трафіку
		Соціальна інженерія
06	Порушення доступності (Availability)	Атака на відмову в обслуговуванні
		Розподілена атака на відмову в обслуговуванні
		Саботаж, диверсія
		Збій без участі зловмисника

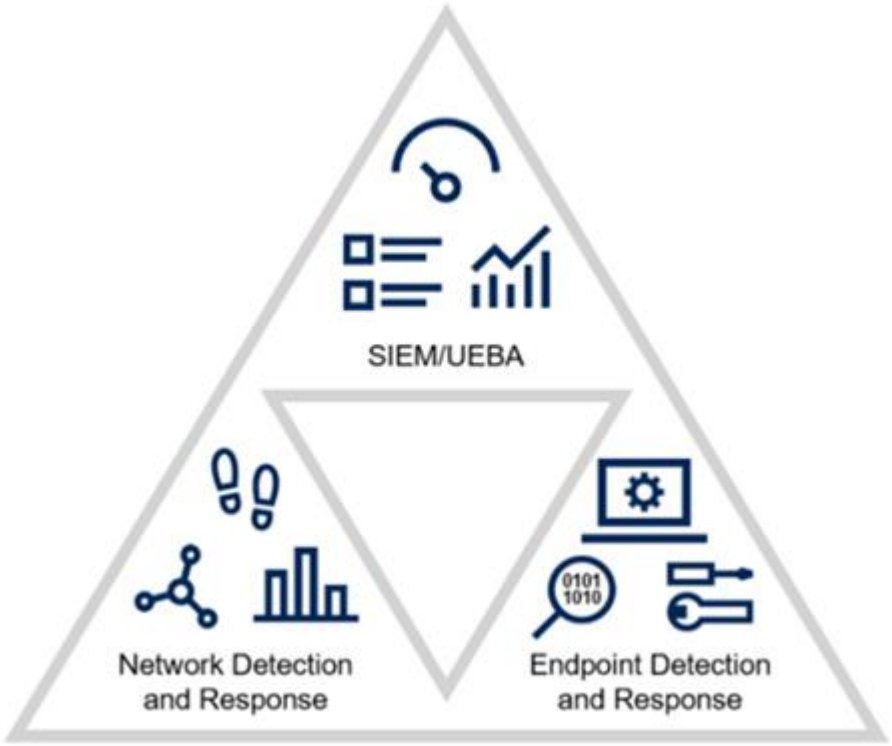


Рисунок 1.2 - SOC Visibility Triad за Gartner

У той час як EDR є лише вікном до кінцевих точок, системи SIEM обробляють все багатство і глибину журналів. NDR забезпечує цілісну мережеву перспективу. NDR дозволяє виявляти порушення на кожному етапі компрометації, оскільки кожен зловмисник залишає сліди в мережевому трафіку, і, таким чином, доповнює можливості SIEM та EDR.

Ця тріада запропонована ще у 2015 році, але під другим ім’ям. У 2019 році з’явилась її актуальна назва. І ось наразі 2023 рік – ще більшої популяризації піддалась течія захисту додатків, інформаційної безпеки у хмарному середовищі та технологія інформаційних пасток.

Можливо в найближчому майбутньому до тріади буде додано новий член – системи на основі пасток, які відіграватимуть роль попередження про можливі майбутні шкідливі дії на виробничих інформаційних системах. Але зараз тріада захоплює 2 проміжки часу – те, що відбувається зараз, та те, що відбулось.

За допомогою індикаторів компрометації (IOCs), котрі поширюються професійною спільнотою у напрямі кібербезпеки після аналізу кібернетичних

інцидентів, можливо бути однією ногою у майбутньому та захищати підприємство від нападу кіберзлочинців.

Оскільки переважна більшість зловмисних дій можна побачити в мережевому трафіку, включаючи шкідливе програмне забезпечення, ми можемо використовувати інструменти для проведення криміналістичного аналізу шаблонів трафіку та контенту. Сканування портів, зв'язок із серверами бот-мереж, C&C, передача великих обсягів даних, аномалії та зміни в поведінці хоста - це основні індикатори, які вказують на поточний злом.

Після того, як інструмент виявлення сповістив про підозрілу активність в мережі, спеціаліст повинен чітко визначити чи є ця подія інцидентом, чи є вона частиною групи інших підозрілих подій, що також можна віднести до інциденту. І при його розслідуванні зрозуміти, хто є автором події та ідентифікувати пристрій(и), відповідальний(і) за цю подію. Інструменти NDR дуже ефективні, оскільки вони можуть надати багато додаткової інформації, включаючи відповідний пристрій за IP-адресою, доменне ім'я пристрою, записане під час виявлення, фізичну адресу пристрою (MAC-адресу) та ідентифікатор користувача. У випадку з програмами-вимагачами можна витягти індикатори компрометації, такі як URL-адреси, хеші, IP-адреси тощо [7].

Моніторингу мереж для виявлення кібератак недостатньо, щоб захистити організацію від загроз. Коли технології кібербезпеки виявляють і блокують загрози, зловмисники розвивають свої стратегії, щоб їх уникнути. Покладатися на IOC для виявлення, захисту та запобігання не є ефективним.

Підприємства, безумовно, повинні бути знайомі з IOC для звичайних кібератак. Але кіберзлочинці та виконавці програм-вимагачів діють дуже витончено. Їхні стратегії атак і способи доставки корисного навантаження швидко еволюціонують. Зосередження уваги на IOCs попередньої атаки не підготує організацію до майбутньої атаки.

Особливо це стосується безпеки електронної пошти. Застарілі рішення безпеки зосереджені на відомих індикаторах компрометації, таких як підозрілі URL-адреси та шкідливі вкладення. Але сучасні атаки на електронну пошту, такі як компрометація

ділової електронної пошти (Business Email Compromise, BEC), покладаються на методи імітації та соціальної інженерії. Ці тактики не мають очевидних червоних прапорців, тому підходи до безпеки, будуть намагатися їх виявити [10].

1.2.2 Категорія “Шкідливий програмний код”

У другій категорії інцидентів вміщуються наступні:

- вірус;
- хробак;
- троян;
- шпигунське програмне забезпечення;
- діалер;
- руткіт;
- шкідливе програмне забезпечення;
- управління ботами;
- програма-здирник;
- конфігурація шкідливого програмного забезпечення;
- командно-контрольний центр.

Зазвичай поява та активізація шкідливого програмного коду відбувається в кінці або на середніх етапах атаки чи інциденту. Слід розумітись у загальноприйнятих етапах атаки, щоб адекватно аналізувати події під час інциденту.

Цей ланцюг допоможе зрозуміти, яка подія може означати, що в корпоративній мережі вже можливо щось відбулось. Наприклад, якщо спостерігається великий мережевий трафік, що схожий на вивантаження даних з локальної мережі, то мабуть вже відбувся інцидент зі сканування, компрометації привілейованого облікового запису, фішингу тощо.

Або якщо був помічений фішинг, то можливо вже десь відбувається активізація зловмисного програмного коду, отримання привілеїв адміністратора домену тощо.У 2017 році Пол Полс опублікував «The Unified Kill Chain» [11]. Цей ланцюг кібератаки

був розроблений на основі та вміщував в себе відомі інші ланцюги Lockheed Martin Cyber Kill Chain [12] та MITRE ATT&CK [13].

Таблиця 1.2

Фази ланцюга «The Unified Kill Chain» [11]

#	Фаза атаки	Опис
1	Розвідка	Дослідження, виявлення та вибір цілей за допомогою активної або пасивної розвідки.
2	Озброєння	Підготовчі заходи, спрямовані на створення інфраструктури, необхідної для здійснення атаки.
3	Доставка	Технології, результатом яких є передача озброєного об'єкта в цільове середовище.
4	Соціальна інженерія	Методи, спрямовані на маніпулювання людьми для виконання небезпечних дій.
5	Експлуатація	Методи використання вразливостей в системах, які можуть, серед іншого, призвести до виконання коду.
6	Стійкість	Будь-який доступ, дії або зміни в системі, які забезпечують постійну присутність зловмисника в системі.
7	Ухилення від оборони	Методи, які зловмисник може спеціально використовувати для ухилення від виявлення або обходу інших засобів захисту.
8	Командування та управління	Методи, які дозволяють зловмисникам взаємодіяти з контрольованими системами в цільовій мережі.
9	Поворот	Тунелювання трафіку через контрольовану систему до інших систем, до яких немає прямого доступу.
10	Виявлення	Методи, які дозволяють зловмиснику отримати знання про систему та її мережеве оточення.
11	Ескалація привілеїв	Результат застосування методів, які надають зловмиснику більш високі дозволи в системі або мережі.
12	Виконання	Методи, що призводять до виконання контрольованого зловмисником коду на локальній або віддаленій системі.
13	Доступ до облікових даних	Методи, що призводять до отримання доступу або контролю над обліковими даними системи, служби або домену.
14	Латеральне переміщення	Методи, які дозволяють зловмиснику отримати горизонтальний доступ до інших віддалених систем і контролювати їх.

Продовження таблиці 1.2

15	Збір	Методи, що використовуються для ідентифікації та збору даних з цільової мережі перед проникненням.
16	Витікання	Методи, які призводять або допомагають зловмиснику видалити дані з цільової мережі.
17	Вплив	Методи, спрямовані на маніпулювання, переривання або знищення цільової системи або даних.
18	Цілі	Соціально-технічні цілі атаки, спрямовані на досягнення стратегічної мети.

Автором було зібрано індикатори, які можуть свідчити про кіберінцидент, наявність шкідливого програмного забезпечення або кіберпорушника в інформаційно-комунікаційній системі (ІКС). Вони відображені у додатку А. Не було ціллю зібрати вичерпний деталізований список усіх ідентифікаторів, заглиблюючись у роботу різних протоколів та метрик трафіку, при створенні його. Наприклад, інциденти, що відносяться до спаму, фішингу, цільовий фішинг тощо, має свій набір індикаторів, який також було сформовано при дослідженні, проте не внесено до результатів роботи.

Було обрано декілька видів інцидентів для подальшого дослідження, вони описані далі у роботі.

1.2.3 Трояни і шпигунське програмне забезпечення

Трояни призначені для того, щоб обманом змусити користувача повірити, що вони є нешкідливим, легітимним програмним забезпеченням, щоб отримати доступ до системи користувача.

Трояни можуть бути приховані в легальному програмному забезпеченні або замасковані під нешкідливі файли, що ускладнює їхнє виявлення. Відомі хеш-суми зловмисних файлів можуть допомогти ідентифікувати їх, проте звісно не всі файли вдасться детектувати ними. Деякі рішення, як YARA [14], можуть знаходити шкідливі файли, аналізуючи їх нутрощі на наявність різних частин коду, рядків тощо.

Після встановлення трояна в системі жертви він створює чорний хід, який дозволяє хакерам отримати несанкціонований доступ до системи і віддалено виконувати команди.

Трояни несуть корисне навантаження, яке може варіюватися від крадіжки конфіденційної інформації, пошкодження або видалення файлів до захоплення комп'ютера жертви для використання в бот-мережі.

Усі команди надсилаються часто від командного серверу, який описаний нижче в роботі. Відслідкувати зловмисних трафік можна за IP та схожим патерном. Але знаходження інциденту за доменним ім’ям доволі складно, хоча й імовірно. Залежить від порушників: змінять вони домен чи ні.

Трояни зазвичай доставляються у вкладеннях до електронних листів, посиланнях на заражені веб-сайти або за допомогою тактики соціальної інженерії, яка заманює жертву завантажити або натиснути на шкідливий файл.

Щоб зменшити ризик троянів, організації повинні впровадити технічні засоби контролю, такі як рішення для захисту від шкідливого програмного забезпечення та брандмауери, а також навчити своїх співробітників розпізнавати фішинг-шахрайство та інші тактики соціальної інженерії. Регулярні оцінки та аудити безпеки також можуть допомогти виявити та зменшити вразливості в системі безпеки організації.

Шпигунське програмне забезпечення призначене для прихованої роботи без відома або згоди користувача з метою збору та передачі конфіденційної інформації.

Шпигунські програми в основному використовуються для збору конфіденційної інформації про жертву, такої як облікові дані для входу в систему, персональні та фінансові дані, а також звички перегляду веб-сторінок.

Шпигунське програмне забезпечення може бути доставлене різними способами, наприклад, у вигляді вкладень в електронній пошті, завантаження програмного забезпечення або за допомогою тактики соціальної інженерії, яка обманом змушує користувача встановити шпигунське програмне забезпечення.

Шпигунські програми розроблені так, щоб залишатися в системі жертви протягом тривалого періоду, продовжуючи збирати та передавати конфіденційну інформацію, доки їх не виявлять і не видалять.

Шпигунські програми можуть споживати значні системні ресурси, сповільнюючи роботу системи та впливаючи на її загальну продуктивність.

Щоб зменшити ризик шпигунських програм, організації повинні впроваджувати технічні засоби контролю, такі як рішення для захисту від шкідливого програмного забезпечення, системи виявлення та запобігання вторгненням, а також рішення для захисту кінцевих точок. Навчання співробітників також може допомогти знизити ризик зараження шпигунським програмним забезпеченням, навчивши користувачів розпізнавати та уникати поширених тактик соціальної інженерії. Регулярні оцінки та аудити безпеки також можуть допомогти виявити та зменшити вразливості в системі безпеки організації.

Серед останніх атак відмітився AgentTesla, який вважають і за троян, і за шпигунське ПЗ [15]. Він використовує XOR-шифрування для захисту свого зв'язку з сервером C&C, як і своїх конфігураційних файлів, від інструментів моніторингу мережевого трафіку.

Також нещодавно у лютому цього року урядова команда реагування на комп'ютерні надзвичайні події України CERT-UA повідомила про атаки з використанням трояна віддаленого доступу Remcos [16].

1.2.4 Програми-здирники та командно-контрольний центр

Програми-здирники призначені для вимагання грошей у жертв шляхом шифрування їхніх файлів або блокування доступу до їхніх систем до моменту сплати викупу.

Програми-вимагачі можуть доставлятися різними способами, наприклад, у вигляді вкладень в електронних листах, завантаження програмного забезпечення або за допомогою тактики соціальної інженерії, яка обманом змушує користувача встановити шкідливе програмне забезпечення.

Вони шифрують файли жертви, роблячи їх недоступними до тих пір, поки не буде сплачено викуп для отримання ключа розшифровки.

Програми-вимагачі вимагають оплату в обмін на ключ розшифрування, часто у формі криптовалюти, щоб уникнути виявлення та відстеження.

Після інсталяції програми-здирники можуть залишатися в системі жертви протягом тривалого періоду часу, часто витримуючи перезавантаження та оновлення програмного забезпечення, що ускладнює їх видалення.

Щоб знизити ризик появи програм-вимагачів, організації повинні впровадити технічні засоби контролю, такі як рішення для захисту від шкідливого програмного забезпечення, брандмауери, системи виявлення та запобігання вторгненням, а також рішення для захисту кінцевих точок доступу. Крім того, навчання співробітників може допомогти знизити ризик зараження, навчивши користувачів розпізнавати та уникати поширених тактик соціальної інженерії. Регулярні оцінки та аудити безпеки також можуть допомогти виявити та зменшити вразливості в системі безпеки організації. Важливо також постійно оновлювати операційні системи та програмне забезпечення найновішими патчами безпеки, щоб зменшити ризик експлуатації. Регулярне резервне копіювання критично важливих даних також може допомогти пом'якшити наслідки атаки зловмисників.

Усі ці види шкідливого програмного забезпечення звертаються до C&C - це центральний сервер або мережа, яка віддалено контролює та координує дії заражених шкідливим програмним забезпеченням пристроїв, таких як ботнети, програми-вимагачі та шпигунські програми. Тому на думку автора це є одним із найкращих місць для аналізу трафіку, щоб ідентифікувати інцидент.

Сервери C&C взаємодіють із зараженими пристроями за допомогою різних протоколів і каналів зв'язку, таких як HTTP, IRC і пірингові (P2P) мережі.

Сервери C&C можуть доставляти на заражені пристрої різноманітне корисне навантаження, наприклад, додаткове шкідливе програмне забезпечення, оновлення та інструкції для запуску атак або викрадення даних.

Сервери C&C можуть використовувати передові технології, щоб уникнути виявлення, такі як алгоритми генерації доменів (DGA) для генерації випадкових доменів або IP-адрес, або використання шифрування для заплутування зв'язку.

Сервери C&C можуть працювати протягом тривалого періоду часу, часто витримуючи спроби вивести їх з ладу, завдяки використанню резервних або дублюючих серверів.

Щоб зменшити ризик даного інциденту, організації повинні впроваджувати технічні засоби контролю, такі як рішення для захисту від шкідливого програмного забезпечення, брандмауери, системи виявлення та запобігання вторгненням, а також рішення для забезпечення мережевої безпеки. Крім того, регулярні оцінки та аудити безпеки можуть допомогти виявити та зменшити вразливості в системі безпеки організації. Важливо також постійно оновлювати операційні системи та програмне забезпечення найновішими патчами безпеки, щоб зменшити ризик експлуатації. Трафік до відомих C&C серверів можна заблокувати за допомогою різних методів, таких як блокування доменних імен, блокування IP-адрес або DNS-синкхолинг.

Серед останніх програм-здирників великої популярності набрав Conti через те, що у 2022 році відбувся витік його оригінального коду та навчальні матеріали по ньому від учасника групи хакерів. Після того почали створюватись нові варіації цього ПЗ і зловмисник угрупувань стало більше. Також активну діяльність веде група LockBit, у неї є декілька дочірніх груп, наприклад LockBit Green чи LockBit Black.

1.3 Дослідження у сфері класифікації подій кібербезпеки

Автори будь-якої моделі з ідентифікації кібератак та їх класифікації при її побудові проходили наступні основні етапи:

- збір даних, що стосуються конкретної проблеми, яку потрібно вирішити;
- підготовка даних до аналізу: очищення даних, обробка відсутніх або недійсних значень, перетворення даних у відповідний формат і поділ даних на навчальні, валідаційні та тестові набори;
- інженерія ознак, що передбачає вибір і перетворення найбільш релевантних ознак (критеріїв) з даних для використання в якості вхідних даних для моделі машинного навчання. Це може включати створення нових ознак на основі існуючих, масштабування або нормалізацію даних, а також зменшення розмірності даних;

● вибір алгоритму відповідно до задачі, яка постала для вирішення проблеми;

● навчання моделі на підготовлених даних, що також передбачає налаштування параметрів моделі для мінімізації помилки або функції втрат.

● оцінка моделі: після того, як модель навчена, потрібно оцінити її продуктивність на валідаційних і тестових наборах даних. Це передбачає вимірювання метрик точності (Ac), чутливості (Sn), специфічності (Sp) і коефіцієнта кореляції Метью (MCC), які базуються на TP (true positives), TN (true negatives), FP (false positives) і FN (false negatives), що позначають кількість істинно-позитивних, істинно-негативних, хибно-позитивних і хибно-негативних результатів відповідно [17].

Усі моделі та відповідні дослідження відрізняються одна від одного як мінімум на основі одного етапу: вибіркою даних (датасетом), підходом до попереднього підготовки даних, алгоритмом, обраними критеріями та звісно результатами.

У деяких роботах використовуються власні побудовані датасети на основі мережевого трафіку, системних викликів, дампу оперативної пам’яті, API викликів у операційній системі тощо. А інші автори використовуються вибірки даних, що є у вільному доступі в мережі Інтернет, серед яких:

- CSE-CIC-IDS2018 на AWS [18];
- набір даних для оцінки виявлення вторгнень CICIDS2017 [18];
- набір даних для оцінки виявлення вторгнень ISCXIDS2012 [19];
- набір даних NSL-KDD [20];
- набір даних для оцінки DDoS-атак CICDDoS2019 [21];
- датасет CIC DoS (2017) [22];
- Mal-API-2019 [23, 24];
- BODMAS [25];
- Ransomware датасет (2016) [26];
- BitcoinHeist Ransomware Dataset [27];
- RanSAP [28, 29] тощо;

У 2015 році Чжао та ін. [30] представили систему, яка використовує як зловмисні DNS, так і аналіз трафіку для виявлення заражень шкідливим програмним

забезпеченням APT. Система складається з двох основних компонентів: детектор шкідливого DNS, який витягує 14 ознак, що вказують на шкідливе ПЗ APT і домени C&C, і аналізатор мережевого трафіку, який поєднує в собі систему на основі сигнатур і систему на основі аномалій для виявлення інфекцій на основі точності правил з набору правил VRT Rule від Snort [31] і аномалій, що виникають на рівні протоколу і програми відповідно. Потім для класифікації загрози використовується дерево рішень J48.

Хеїр [32] описав системний підхід до побудови сигнатур виявлення на основі аномалій агентів користувача в HTTP-трафіку шкідливого програмного забезпечення. Підхід передбачав вилучення полів заголовків агентів користувача в HTTP-трафіку та їх кластеризацію в групи зі схожими шаблонами. На другому етапі кластеризації були згруповані схожі користувацькі агенти, які можна описати за допомогою спільного набору сигнатур. Інкрементна кластеризація K-середніх була використана для перегрупування агентів користувачів, які мають схожі послідовності шаблонів, в одні й ті ж кластери. Алгоритм токен-послідовності був використаний для вилучення цих спільних шаблонів і створення списків послідовностей токенів, які були перетворені в підписи, що можуть бути застосовані на мережевому або прикладному рівнях за допомогою веб-проксі-серверів.

Бекерман та ін. [33] у 2015 році розробили систему для виявлення шкідливого програмного забезпечення шляхом аналізу мережевого трафіку. Вони виділили 972 поведінкові ознаки (критерії) з мережевого трафіку на Інтернет-, транспортному та прикладному рівнях і вибрали підмножину ознак за допомогою алгоритму кореляційного вибору ознак. Три алгоритми класифікації були протестовані з використанням отриманих критеріїв: Наївний Байєс, дерево рішень (J48) та випадковий ліс.

Бохтута [34] запропонував систему виявлення та класифікації шкідливого програмного забезпечення, яка спирається на DPI та потокові заголовки. Шкідливе програмне забезпечення виконувалося в пісочниці для генерування репрезентативного шкідливого трафіку, а з нього виділялися ознаки двонаправленого потоку. В якості алгоритмів класифікації для визначення зловмисності трафіку

використовувалися алгоритми Boosted J48, J48, Naive Bayes, Boosted Naive Bayes і SVM. Приховані марківські моделі були використані для створення недетермінованих моделей, які профілювали сім'ї шкідливих програм за допомогою односпрямованих потоків, представлених у вигляді набору з 45 ознак.

Роберто Пердіскі та Венкі Лі [35] через рік запропонували метод поведінкової кластеризації шкідливого програмного забезпечення на основі HTTP-трафіку, отриманого в результаті моніторингу виконуваних файлів у контрольованому середовищі. Метод записував послідовності HTTP-запитів, які виконувало шкідливе програмне забезпечення, і використовував цю інформацію для кластеризації шкідливого програмного забезпечення з використанням грубозернистої кластеризації, дрібнозернистої кластеризації та алгоритмів злиття кластерів. Потім для кожного кластера були вилучені мережеві підписи, які використовувалися для ідентифікації заражених комп'ютерів.

Іман Шарафалдін з іншими [18] розробили датасет, використовуючи більше десятка віртуальних машин. Вони об’єднали різні атаки: DoS GoldenEye, Heartbleed, DoS Hulk, DoS Slowhttp, DoS slowloris, SSH-Patator, FTP-Patator, Web Attacks, Infiltration, Botnet attack, PortScan та DDoS. Вони відверто нескладні, адже більшість пов’язана з атаками на відмову в обслуговуванні, сканування та брутфорс. Якщо вони виконуються швидко, якщо пакети надходять без спеціально встановленої затримки, то за допомогою автоматизованих систем моніторингу мережі чи логів їх можна детектувати без використання машинного навчання. У своїй роботі вони описали основні кроки при побудові середовища для тестування та розробки, обрані алгоритми: KNN, RF, ID3, Adaboost, MLP, Naive-Bayes, QDA. І заміряли, скільки часу відводиться на тренування та тестування моделі. Гарним себе показав алгоритм RF.

Ранжит Панідраї з іншими [37] проаналізували CICIDS2017, помітили велику групу інстансів, які не мають всіх значень або мають значення нескінченності, та видалили їх із датасету, змінили класи та об’єднали їх. З 15 класів зробили 6. Таким чином вони досягли зниження відсотку хибних детектувань.

А в березні цього року Аніта Ширавані разом з іншими [38] зробил акцент на попередньому етапі обробки інформації чи датасету, на зменшенні критеріїв, за якими

потрібно буде класифікувати події автоматизованій системі. Вони розглянули декілька датасетів, що доступні у мережі Інтернет. Вони запропонували метод для покращення процесу вибору ознак на основі кореляції. Після застосування різних методів відбору ознак до набору даних, CFS (correlation feature selection) був обраний як найбільш ефективний метод відбору ознак серед наявних загальноприйнятих. І на основі нього зробили свій, який зміг прибрати хибні спрацювання та збільшити вірогідність вірних передбачень. Вони зменшили кількість критеріїв як мінімум у 6 разів, а для датасету CICIDS2017 у 27 разів.

І дійсно, якщо проаналізувати дослідження по CICIDS2017 [18], то можна побачити, що критерії датасету підібрані так, що вони охоплюють різні атаки. Таким чином, деякі з них не стосуються якоїсь категорії атак або мають маленький влпив, щоб розпізнати подію як інцидент серед усіх можливих. Найбільш популярними є Total Len F.Packets, Subflow F.Bytes, Flow Duration, B.Packet Len Std.

1.4 Постановка задачі

Об’єктом дослідження являється процес класифікації подій кібербезпеки за допомогою інтелектуальних моделей.

Оскільки дана задача включає в себе виконання широкого діапазону робіт (на основі попереднього аналізу) і орієнтована на постійне доопрацювання та удосконалення, то метою цієї дипломної роботи є побудова ефективної моделі виявлення та ідентифікації подій кібербезпеки відносно деяких мережевих загроз, розробка необхідного програмного забезпечення до основних етапів ідентифікації або ж використання існуючого, навчання нейронної мережі.

Згідно з ціллю роботи, поставлено наступні завдання:

- збір даних про існуючі мережеві атаки, збір матеріалу з мережевомим трафіком;
- переведення пакетів мережевого трафіку у формат потоків;
- розробка алгоритмів підготовки даних;
- розробка моделей нейронних мереж та їх подальше навчання;

- перевірка адекватності розроблених моделей.

Кожна з мережевих атак містить свої технологічні особливості та відмінності. Для проведення комплексного дослідження в роботі розглядалося декілька моделей нейронної мережі та аналізувалася доцільність їх використання.

Висновки за розділом 1

У першому розділі дипломної роботи розглянуто основні світові тенденції розвитку використання штучного інтелекту у кібербезпеці. Для забезпечення якості кібернетичної безпеки у засобах інформатизації все частіше використовуються засоби інтелектуального аналізу даних. Проаналізована зарубіжна література, досягнення інших науковцій в даній темі, створені датасети. Також на основі вітчизняної та зарубіжної літератури проаналізовано, що саме входить в поняття кіберінцидент та їх категорії. Широкий набір алгоритмів машинного навчання дає змогу реалізувати більш гнучкі засоби протидії різним кібератакам. У наступних розділах описано розробка декількох таких моделей.

РОЗДІЛ 2
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. РОЗРОБКА МОДЕЛІ ІДЕНТИФІКАЦІЇ ІНЦИДЕНТУ НА РІВНІ МЕРЕЖЕВОГО ТРАФІКУ

2.1 Теоретичні основи та поняття

2.1.1 Опис нейронних мереж

Штучні нейронні мережі (ШНМ) - це обчислювальні моделі, натхненні структурою та функціональністю людського мозку. Вони набули значної популярності в галузі штучного інтелекту та машинного навчання завдяки своїй здатності навчатися на основі даних, розпізнавати закономірності та робити прогнози або приймати рішення. ШНМ складаються з взаємопов'язаних вузлів, які називаються нейронами, що імітують поведінку біологічних нейронів [39].

Основне призначення штучних нейронних мереж - вирішувати складні проблеми, які важко вирішити за допомогою традиційних алгоритмічних підходів. ШНМ відмінно справляються з такими завданнями, як розпізнавання образів, класифікація, регресія та оптимізація. Вони успішно застосовуються в різних сферах, включаючи комп'ютерний зір, обробку природної мови, розпізнавання мови, робототехніку та фінансове прогнозування.

Фундаментальний принцип роботи штучних нейронних мереж полягає в імітації поведінки біологічних нейронних мереж. Кожен нейрон в ШНМ отримує вхідні сигнали від інших нейронів або зовнішніх джерел, обробляє їх і виробляє вихідний сигнал. Зв'язки між нейронами представлені вагами, які визначають силу та важливість сигналів.

Навчання - важливий етап у функціонуванні штучних нейронних мереж. Під час навчання мережа навчається на маркованому наборі даних, коригуючи свої внутрішні параметри (ваги та зсуви), щоб мінімізувати різницю між прогнозованими та бажаними результатами. Цей процес зазвичай виконується за допомогою

алгоритмів оптимізації, таких як градієнтний спуск, які ітеративно оновлюють ваги на основі розрахованої помилки.

Одним з ключових понять в ШНМ є функція активації. Вона визначає, чи слід активувати нейрон на основі зваженої суми його входів. Найпоширеніші функції активації включають сигмоїд, ReLU (випрямлена лінійна одиниця) і тангенс (гіперболічний тангенс). Функції активації вносять нелінійності в мережу, дозволяючи їй вивчати складні взаємозв'язки і фіксувати нелінійні закономірності в даних.

Штучні нейронні мережі можуть мати різну архітектуру, залежно від конкретної задачі та бажаної продуктивності. Найпоширенішою архітектурою є нейронна мережа прямого поширення, де інформація рухається в одному напрямку, від вхідних до вихідних шарів, без петель або циклів. Цей тип мереж часто використовується для таких завдань, як класифікація та регресія.

Інша архітектура - це рекурентна нейронна мережа (RNN), яка дозволяє зв'язкам між нейронами утворювати цикли, що дозволяє мережі зберігати інформацію з часом. ШНМ особливо добре підходять для завдань, пов'язаних з послідовними даними, таких як розпізнавання мови, моделювання мови та аналіз часових рядів.

2.1.2 Вибір архітектури нейронної мережі

Перцептрони, як перше покоління нейронних мереж, відіграли вирішальну роль у розумінні того, як мозок сприймає та обробляє інформацію. Розроблені Френком Розенблатом наприкінці 1950-х років, персептрони були спроектовані як прості моделі біологічних нейронів. Вони складалися з вхідних вузлів, ваг, функції підсумовування та функції активації, імітуючи структуру та поведінку реальних нейронів.

Перцептрони стали значним проривом у галузі штучного інтелекту та когнітивних наук, дозволивши дослідникам вивчити фундаментальні принципи, що лежать в основі обробки інформації мозком. Моделюючи сприйняття мозку, перцептрони створили основу для вивчення того, як нейронні мережі можуть

навчатися та адаптуватися до різних стимулів, прокладаючи шлях для майбутніх досягнень у машинному навчанні та когнітивних дослідженнях.

Роль перцептронів у моделюванні сприйняття мозку полягала в імітації механізмів, за допомогою яких нейрони інтегрують і передають сигнали. Завдяки взаємодії вхідних сигналів і регульованих ваг перцептрони могли навчитися розпізнавати патерни і приймати рішення на основі вхідних даних. Ця здатність вчитися на прикладах і налаштовувати ваги дозволила перцептронам виконувати такі завдання, як розпізнавання зображень, класифікація і прогнозування, що відображає аспекти людських когнітивних процесів.

Хоча персептрони продемонстрували потенціал нейронних мереж, їхні обмеження стали очевидними при вирішенні складних завдань, які вимагають нелінійних меж рішень. Алгоритм навчання персептронів, відомий як правило персептрона, міг збігатися лише для лінійно розділених наборів даних. Це обмеження призвело до розробки більш складних архітектур нейронних мереж і алгоритмів навчання, таких як багатошарові персептрони і зворотне поширення, які проклали шлях до значного прогресу в області штучного інтелекту і глибокого навчання. Тим не менш, внесок персептрона у створення основ досліджень нейронних мереж залишається неоціненним.

Властивості багатошарового перцептрону відображаються у конструкції конволюційних нейронних мереж (CNN) Згорткові нейронні мережі (CNN) - це тип глибоких нейронних мереж, які здійснили революцію в галузі комп'ютерного зору та обробки зображень. Вони були натхненні структурою та функціонуванням зорової кори головного мозку тварин, яка вміє розпізнавати зорові образи. ШНМ відмінно справляються з такими завданнями, як класифікація зображень, виявлення об'єктів і сегментація зображень. Ключова ідея CNN полягає у використанні згорткових шарів, які дозволяють мережі автоматично вивчати ієрархічні представлення візуальних ознак.

Архітектура ШНМ зазвичай складається з декількох шарів, включаючи згорткові шари, об'єднані шари та повністю з'єднані шари. Згорткові шари є будівельними блоками ШНМ і відповідають за виявлення локальних шаблонів та

особливостей у вхідному зображенні. Кожен згортковий шар складається з набору фільтрів або ядер, що навчаються, які згортаються над вхідним зображенням, застосовуючи операцію точкового добутку для виокремлення ознак. Цей процес дозволяє мережі вивчати просторові ієрархії об'єктів, фіксуючи низькорівневі деталі в ранніх шарах і високорівневі концепції в більш глибоких шарах.

Об'єднувальні шари, які часто вставляються між згортковими шарами, зменшують просторові розміри карт об'єктів, зберігаючи при цьому важливу інформацію. Вони допомагають зменшити обчислювальну складність і покращити здатність мережі до узагальнення, забезпечуючи просторову інваріантність. Найпоширенішою операцією об'єднання є максимальне об'єднання, яке обирає максимальне значення в межах локальної околиці. Цей процес зменшення вибірки ефективно зменшує роздільну здатність карт об'єктів, що призводить до більш компактного представлення.

Повністю з'єднані шари, розташовані в кінці мережі, беруть високорівневі ознаки, витягнуті попередніми шарами згортки та об'єднання, і зіставляють їх з потрібними вихідними класами. Ці шари з'єднують кожен нейрон попереднього шару з кожним нейроном наступного шару, що дозволяє створювати складні взаємозв'язки і приймати рішення. Як правило, останній повністю з'єднаний шар використовує функцію активації softmax для отримання ймовірностей класів.

ШНМ навчаються за допомогою процесу, який називається зворотним поширенням, що передбачає коригування ваг та упереджень мережі на основі різниці між прогнозованим виходом та істинними мітками. Великомасштабні набори мічених даних, такі як ImageNet, відіграли життєво важливу роль у навчанні ШНМ для досягнення найсучаснішої продуктивності в різних завданнях комп'ютерного зору. Крім того, такі методи, як навчання з перенесенням, коли попередньо навчені моделі ШНМ налаштовуються на конкретні завдання, виявилися ефективними, коли кількість мічених даних обмежена.

Сучасні архітектури CNN є ключовими для розвитку штучного інтелекту. До них відносяться такі моделі, як LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet

[40]. Кожна з цих архітектур має свої особливості і використовується для різних завдань у сфері штучного інтелекту.

Рекурентні нейронні мережі (RNN) - це клас нейронних мереж, спеціально розроблених для обробки послідовних і часових даних. На відміну від нейронних мереж прямого поширення, які обробляють вхідні дані незалежно, RNN мають петлю зворотного зв'язку, що дозволяє зберігати інформацію та обмінюватися нею на різних часових етапах. Цей повторюваний зв'язок дозволяє ШНМ моделювати та фіксувати часові залежності, присутні в послідовних даних, що робить їх придатними для таких завдань, як обробка природної мови, розпізнавання мовлення та аналіз часових рядів.

Фундаментальним будівельним блоком ШНМ є рекурентний блок, також відомий як комірка ШНМ. Найчастіше використовується комірка RNN з довгою короткочасною пам'яттю (LSTM), яка вирішує проблему зникаючого градієнта і дозволяє RNN вивчати довгострокові залежності. Іншим популярним варіантом є Gated Recurrent Unit (GRU), який спрощує архітектуру LSTM, зберігаючи при цьому ефективні можливості моделювання.

Ключовою ідеєю RNN є концепція прихованих станів. На кожному часовому кроці рекурентний блок отримує вхідний сигнал і комбінує його з попереднім прихованим станом, створюючи новий прихований стан на виході. Прихований стан можна розглядати як пам'ять або підсумок інформації, яку ШНМ обробив до цього часу. Підтримуючи та оновлюючи цей прихований стан протягом всієї послідовності, ШНМ здатні зберігати інформацію з попередніх вхідних даних і робити прогнози на основі накопиченого контексту.

Навчання ШНМ передбачає використання методу зворотного поширення в часі (BPTT), який є розширенням алгоритму зворотного поширення для рекурентних архітектур. BPTT поширює градієнт функції втрат через усі часові кроки, дозволяючи мережі навчатися на всій послідовності. Однак градієнт має тенденцію або зникати, або вибухати на довгих послідовностях, що може зробити навчання складним. Для пом'якшення цих проблем часто застосовують такі методи, як відсікання градієнта та регуляризація, наприклад, відсів або повторний відсів.

Одним з обмежень традиційних ШНМ є їхня складність у виявленні довготривалих залежностей. Коли послідовності дуже довгі, інформація з попередніх часових кроків може розмиватися або втрачатися. Для вирішення цієї проблеми були розроблені більш досконалі архітектури, такі як модель Transformer, яка включає в себе механізми самоуваги. Трансформатори привернули значну увагу в задачах обробки природної мови і досягли найсучасніших результатів у таких задачах, як машинний переклад і генерація мови.

Отже, рекурентні нейронні мережі (RNN) - це спеціалізовані нейронні мережі, призначені для обробки послідовних і часових даних. Використовуючи рекурентні зв'язки та приховані стани, RNN можуть фіксувати та моделювати залежності на різних часових етапах, що робить їх придатними для завдань, пов'язаних з послідовністю. Такі варіанти, як LSTM і GRU, вирішують проблему зникаючого градієнта і покращують здатність мережі вивчати довгострокові залежності.

Дійсно, довга короткочасна пам'ять (long short-term memory, LSTM) є важливою архітектурою рекурентних нейронних мереж (RNN), яка була запропонована в 1997 році Зеппом Хохрайтером та Юргеном Шмідгубером. LSTM-мережа вирішує проблему зникнення та затухання градієнта, що може виникати при навчанні традиційних RNN, і дозволяє зберігати та використовувати довготривалу залежність в послідовності даних.

Основна особливість LSTM-мережі полягає в наявності спеціальних блоків пам'яті, відомих як "клітинки", які можуть зберігати інформацію протягом довгого періоду часу. Клітинки мають механізми, що дозволяють контролювати забування інформації та додавання нової інформації до пам'яті. Це дозволяє LSTM-мережі ефективно працювати з послідовностями довільної довжини та обробляти довготривалі залежності між елементами послідовності.

LSTM-мережі мають широкий спектр застосувань. Вони ефективні для завдань класифікації, обробки та передбачення часових рядів, компресії тексту природною мовою, розпізнавання несегментованого рукописного тексту, автоматичного розпізнавання мовлення та інших задач, де важлива довготривала залежність між елементами послідовності.

Застосування LSTM-мереж знайшли великі компанії, такі як Google, Apple, Microsoft та Baidu, які використовують їх у своїх продуктах і розробках. LSTM-мережі показали вражаючі результати в багатьох завданнях обробки послідовностей, і їх використання продовжує розширюватись у багатьох галузях, де важлива робота з послідовними даними [41].

Мережа Гопфілда (Hopfield Network) є простою рекурентною нейронною мережею з повнозв'язними шароми та симетричною матрицею зв'язків. Вона була створена Джоном Гопфілдом в 1982 році і має лише один шар. Основними завданнями, для яких використовується ця мережа, є автоматична асоціація та оптимізація. У відміну від багатьох інших нейронних мереж, які працюють до отримання відповіді через певну кількість епох, мережа Гопфілда працює до досягнення рівноваги, коли наступний стан мережі дорівнює попередньому [42].

Машина Больцмана (Boltzmann machine) є особливим типом стохастичної рекурентної нейронної мережі, яка була винайдена Террі Сейновскі та Джеффрі Хінтоном в 1985 році і названа на честь Людвіга Больцмана, відомого австрійського вченого зі статистичної фізики. Машина Больцмана може бути розглянута як стохастична версія мережі Гопфілда і також відома у статистиці як випадкове марковське поле.

Робота такої мережі полягає у використанні імітаційних алгоритмів для навчання. Машина Больцмана може бути вважена першою нейронною мережею, яка здатна навчатися шляхом внутрішнього представлення та вирішувати складні комбінаторні завдання. Однак, через проблеми з необмеженою зв'язністю, машина Больцмана не є практичним інструментом для розв'язання конкретних задач. Але, якщо зв'язність обмежена, навчання може бути ефективним для практичних застосувань. Зокрема, каскад обмежених машин Больцмана використовується для побудови глибинних мереж переконань [43].

Іншим типом глибинної нейронної мережі в машинному навчанні є глибинна мережа переконань (deep belief network, DBN), Мережі глибоких переконань (Deep Belief Networks, DBN) - це тип генеративної нейромережевої моделі, яка поєднує в собі потужність неконтрольованого навчання з глибокими архітектурами.

Представлені Джеффрі Хінтоном та його колегами у 2006 році, DBN складаються з декількох шарів обмежених машин Больцмана (RBM) і продемонстрували великий потенціал у моделюванні складних розподілів даних та вилученні високорівневих репрезентацій. ШНМ успішно застосовуються в різних галузях, включаючи розпізнавання зображень, обробку мовлення та обробку природної мови.

Архітектура ШНМ складається з вхідного шару, декількох прихованих шарів RBM і вихідного шару. RBM - це імовірнісні моделі на основі енергії, які навчаються представляти спільний розподіл ймовірностей своїх вхідних даних. Кожен шар RBM отримує вхідні дані від попереднього шару і проектує свої власні активації на наступний шар. Цей пошаровий неконтрольований процес навчання дозволяє кожному RBM фіксувати і представляти основні характеристики даних. Вагові коефіцієнти, отримані на цьому некерованому етапі попереднього навчання, ініціалізують НМД для подальшого точного налаштування за допомогою керованих методів навчання, таких як зворотне розповсюдження.

Для ініціалізації ваг кожного RBM в DBN використовується підхід генеративного навчання, який називається "жадібне багаторівневе попереднє навчання". Під час попереднього навчання кожна РБМ навчається реконструювати свої входи на основі активацій попереднього шару. Цей процес дає змогу кожному RBM вивчити стиснене і розподілене представлення вхідних даних. Після попереднього навчання виконується етап точного налаштування на основі зворотного розповсюдження для коригування ваг всієї мережі з використанням маркованих даних. Точне налаштування дозволяє ШНМ вивчати дискримінантні ознаки та оптимізувати модель для конкретних завдань, таких як класифікація.

Однією з помітних переваг DBN є їхня здатність фіксувати ієрархічні представлення даних. Приховані шари в ДБН поступово вчаться представляти все більш абстрактні та високорівневі ознаки в міру того, як інформація протікає через мережу. Таке ієрархічне представлення дозволяє РБД фіксувати складні закономірності та залежності в даних, що робить їх ефективними в задачах, пов'язаних з багатовимірними і структурованими даними. Крім того, ШНМ можуть

генерувати нові зразки з вивченого розподілу, що робить їх генеративними моделями, які можуть створювати нові зразки даних.

Незважаючи на їхній успіх, навчання ШНМ може бути обчислювально дорогим і вимагає великих обсягів маркованих даних. Етап попереднього навчання, хоча й ефективний для ініціалізації, є трудомістким процесом. Однак нещодавні досягнення в галузі глибокого навчання, такі як впровадження більш ефективних алгоритмів навчання та наявність більших наборів даних, сприяли подоланню цих проблем і підвищенню продуктивності ШНМ.

Один з перших ефективних алгоритмів машинного навчання, який був розроблений на основі спостережень про можливості пошарового навчання DBN, був запропонований Yee-Whye Teh, учнем Джефрі Хінтона [44].

Автокодер - це тип архітектури нейронної мережі, який в основному використовується для неконтрольованого навчання та зменшення розмірності. Він складається з мережі-кодера, яка стискає вхідні дані до представлення з меншою розмірністю, що називається "латентним простором", і мережі-декодера, яка реконструює вихідні вхідні дані з латентного простору. Автокодер навчається мінімізувати різницю між вхідними і вихідними даними, ефективно навчаючись кодувати і декодувати дані.

Кодувальна мережа автокодера зазвичай складається з декількох прихованих шарів, які поступово зменшують розмірність вхідних даних. Кожен шар використовує нелінійні функції активації для вилучення та представлення важливих характеристик вхідних даних. Останнім шаром кодера є вузькопрофільний шар, який має найменшу кількість нейронів і представляє стиснене представлення вхідних даних.

Декодерна мережа по суті є дзеркальним відображенням кодера. Вона бере стиснене представлення з вузького шару і поступово розширює його до початкової розмірності. Кожен шар декодера застосовує нелінійні перетворення для реконструкції даних і в кінцевому підсумку видає реконструйовані дані, які в ідеалі повинні бути схожими на оригінальні вхідні дані.

Під час навчання автокодер порівнює вхідні дані з реконструйованим виходом і коригує ваги, використовуючи функцію втрат, наприклад, середньоквадратичну

похибку, щоб мінімізувати похибку реконструкції. Таким чином, автокодер вчиться ефективно кодувати і декодувати дані, фіксуючи найважливіші особливості вхідних даних.

Автокодери мають різні застосування в галузі машинного навчання. Одне з поширених застосувань - зменшення розмірності, коли автокодер можна навчити на даних високої розмірності і використовувати для отримання представлення в нижчій розмірності, яке фіксує найбільш суттєві особливості. Це може бути особливо корисно для візуалізації даних і зменшення обчислювальної складності наступних завдань машинного навчання.

Ще одне застосування - виявлення аномалій, коли автокодер навчається на певному типі даних, а потім використовується для реконструкції нових зразків даних. Якщо помилка реконструкції значно вища для певної вибірки, це може вказувати на аномалію або відхилення від вивченого розподілу даних.

Крім того, автокодери використовуються для зашумлення зображень, коли мережа навчається відновлювати чисті зображення із зашумлених вхідних даних. Змушуючи автокодер фіксувати основну структуру зображень, він може ефективно видаляти шум і підвищувати якість реконструйованих зображень [45].

Підсумовуючи вищевикладене, можна зробити висновок, що кожна архітектура нейромережі має свої концептуальні переваги, обумовлені теоретичним підходом до проектування та практичною значимістю. У сфері розв'язання різноманітних завдань людства використовуються інші архітектури нейромереж. Оскільки основною метою розробленої моделі є класифікація атак, мережа повинна бути класифікатором для певного набору даних. Для задач класифікації в галузі нейронних мереж найпростішою архітектурою є перцептрон, тому в даній роботі була використана одна з його концепцій.

Синтез нейромережевої моделі був виконаний з використанням багатошарового перцептрона Румельхарта, який є окремим випадком перцептрона Розенблатта. Цей перцептрон відрізняється від перцептрона Розенблатта тим, що коригування вагових коефіцієнтів нейронів здійснюється за допомогою алгоритму зворотного поширення похибки. Використання більше одного шару є характерною

особливістю багатошарового перцептрона, і зазвичай розглядаються два або три шари [39].

Багатошаровий персептрон (MLP) і персептрон Розенблата - це типи штучних нейронних мереж, але вони відрізняються за своєю архітектурою та алгоритмами навчання.

Персептрон Розенблата, запропонований Френком Розенблатом у 1957 році, є однією з найперших моделей нейронних мереж. Він складається з одного шару штучних нейронів, які називаються перцептронами. Кожен персептрон отримує кілька входів, застосовує до них ваги і пропускає зважену суму через функцію активації для отримання виходу. Функція активації в оригінальній моделі персептрона - це ступінчаста функція, яка видає або 0, або 1 на основі порогового значення. Перцептрон навчається за допомогою правила навчання перцептронів, яке оновлює ваги на основі похибки між прогнозованим і бажаним виходом. Однак персептрон Розенблата може вивчати лише лінійно відокремлювані шаблони і не може обробляти нелінійні зв'язки між входами.

З іншого боку, багатошаровий персептрон (MLP) є більш універсальною і потужною моделлю нейронної мережі. Він складається з декількох шарів нейронів, включаючи вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен нейрон в MLP подібний до персептрона, який застосовує ваги до своїх входів і пропускає зважену суму через функцію активації. Однак функція активації, що використовується в MLP, зазвичай є нелінійною функцією, наприклад, сигмоїдною або випрямленою лінійною одиницею (ReLU). Нелінійні функції активації дозволяють MLP вивчати складні нелінійні зв'язки між вхідними даними. ШНМ навчаються за допомогою зворотного поширення, яке є ітеративним процесом, що коригує ваги в мережі на основі градієнта функції втрат по відношенню до ваг. Це дозволяє ШНМ навчатися та апроксимувати будь-яку довільну функцію за умови наявності достатньої кількості прихованих нейронів та навчальних даних.

Підсумовуючи, можна виділити основні відмінності між багатошаровим персептроном (MLP) і персептроном Розенблата:

Архітектура: Персептрон Розенблата має один шар нейронів, тоді як MLP має кілька шарів, включаючи вхідний шар, один або кілька прихованих шарів і вихідний шар.

Функція активації: Персептрон Розенблата використовує ступінчасту функцію як функцію активації, тоді як MLP зазвичай використовує нелінійні функції активації, такі як сигмоїдальні або ReLU, що дозволяє йому вивчати складні нелінійні залежності.

Алгоритм навчання: Персептрон Розенблата використовує правило навчання персептрона, яке оновлює ваги на основі похибки між прогнозованими та бажаними результатами. MLP використовує зворотне поширення, яке коригує ваги на основі градієнта функції втрат, що дозволяє йому вивчати складні функції за допомогою ітеративного навчання.

Загалом, багатошаровий персептрон є більш гнучкою та потужною моделлю порівняно з персептроном Розенблата завдяки своїй багатошаровій архітектурі, нелінійним активаційним функціям та здатності вивчати складні нелінійні залежності за допомогою алгоритму зворотного поширення.

Функціональні переваги багатошарового перцептрона порівняно з перцептроном Розенблатта виявляться у випадку, коли його здатність генерувати ефективні реакції на стимули буде підвищена (адже багатошаровий перцептрон може передбачити реакцію кожного типу). Це в свою чергу сприятиме поліпшенню здатності до узагальнення, тобто до правильних реакцій на стимули, на які перцептрон не був навчений раніше. Проте наукова література та практика поки що не мають загальноприйнятих узагальнюючих теорем, існують лише дослідження різних стандартизованих тестів, які використовуються для порівняння різних архітектур.

Оскільки багатошаровий перцептрон може мати будь-яку кількість шарів, необхідно визначити оптимальну кількість шарів для нової моделі. У даній роботі використовувалися три шари: вхідний, вихідний і один внутрішній. Загальний вигляд запропонованої моделі можна представити у форматі, зображеному на рисунку 2.1.

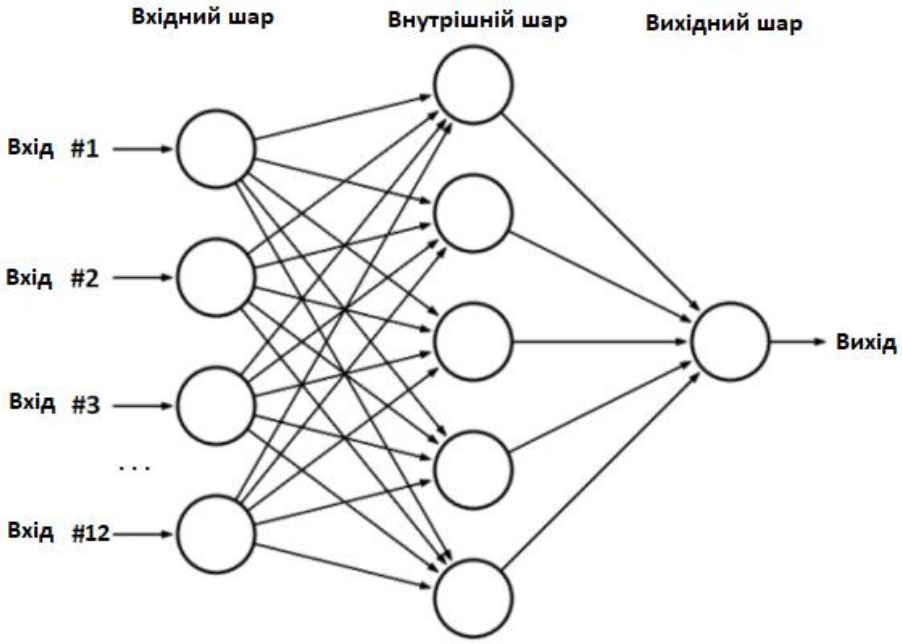


Рисунок 0.1 - Модель нейронної мережі

2.1.3 Опис архітектури мережі

Нейронна мережа зворотного розповсюдження є однією з найбільш широко використовуваних багатошарових нейронних мереж і відноситься до простих і загальних методів, що використовуються для контрольованого навчання багатошарових нейронних мереж. Вона працює шляхом наближення нелінійного зв'язку між входом і виходом шляхом регулювання значень ваг всередині мережі.

Звичайно, мережа зворотного розповсюдження має два основні етапи: навчання і тестування. Під час етапу навчання мережа "відповідає" вибірковим входам і правильним класифікаціям. Наприклад, вхід може представляти закодоване зображення обличчя, а вихід - код, який відповідає імені людини.

Нейронна мережа, так само як і більшість алгоритмів навчання, повинна кодувати входи і виходи відповідно до визначеної користувачем схеми. Схема визначає архітектуру мережі таким чином, що після навчання мережі неможливо змінити схему без створення абсолютно нової мережі.

Операції, що виконуються в нейронних мережах зворотного розповсюдження, можна розділити на два етапи: пряме розповсюдження та зворотне розповсюдження. Під час прямого розповсюдження вхідний шаблон подається на вхідний шар мережі, і сигнал поширюється шар за шаром через мережу до отримання виходу. Фактичний вихід порівнюється з очікуваним виходом, і для кожного вузла вираховується сигнал помилки. Оскільки всі приховані вузли в певній мірі призводять до помилок у вихідному шарі, сигнали помилок передаються назад від вихідного шару до кожного вузла в прихованому шарі, що впливає на вихідний шар. Цей процес повторюється шар за шаром, доки кожний вузол мережі не отримає сигнал про помилку, що відображає його внесок у загальну помилку.

Після визначення сигналу помилки для кожного вузла, ці дані використовуються для оновлення значень ваги на кожному з'єднання до тих пір, поки мережа не досягне стану, коли можна кодувати всі схеми тренувань. Алгоритм зворотного розповсюдження використовує метод, відомий як правило дельти або градієнтний спуск, для пошуку мінімального значення функції помилок у просторі ваги. Ваги, які мінімізують функцію помилок, вважаються розв'язком завдання навчання [46].

Поведінка нейронної мережі подібна до поведінки людини, якій надаються дані для класифікації в задані категорії. Так само, як людина, мережа формує "теорії" про те, як зразки впорядковано в категорії. Потім ці теорії перевіряються на правильність виходів, щоб оцінити точність передбачень мережі. Радикальні зміни в вагах позначаються значними змінами в останньому кроці, тоді як незначні зміни можуть розглядатися як дрібні корекції.

2.2 Синтез моделі

Для створення та аналізу моделі ідентифікації інцидентів кібербезпеки необхідно провести попередню підготовку вибірок навчальних, контрольних та тестових даних. Загальний процес обробки даних моделі представлено на рисунку 2.6.

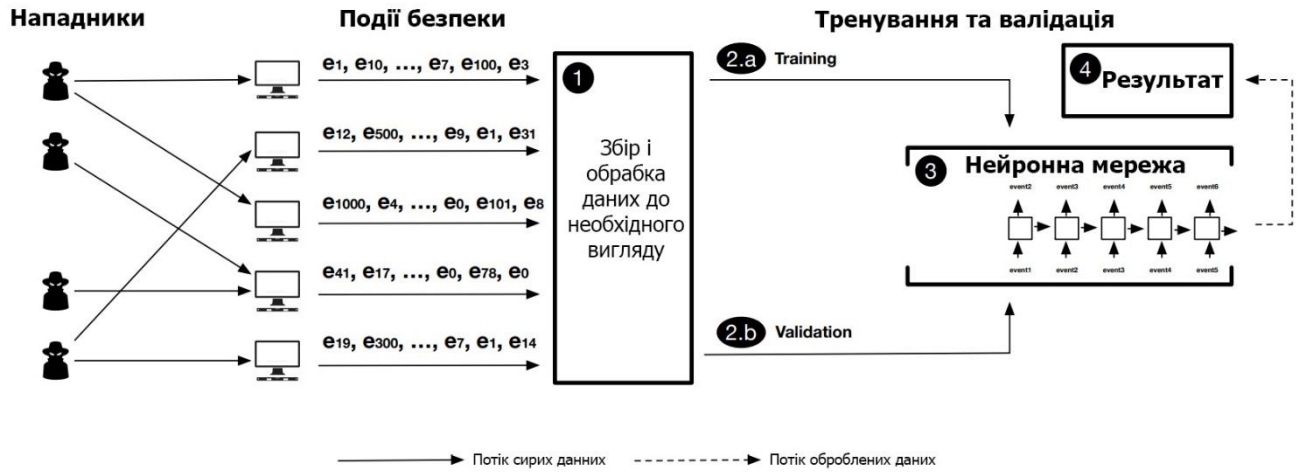


Рисунок 0.2 - Схема процесу обробки даних

Модель складається з чотирьох основних етапів, які включають:

1. Збір даних мережевої активності, системних логів та журналів подій.
2. Обробка попередніх даних і їх приведення до потрібного логічного формату для подальшої обробки.
3. Навчання та тестування нейронної мережі.
4. Аналіз отриманих результатів.

Ці етапи описують загальний підхід до проектування моделі ідентифікації атак, яка буде використовуватись у майбутньому. Кожна модель може мати свою власну інтерпретацію цих етапів і суттєво відрізнятися для виявлення різних типів атак. У даному розділі описується модель ідентифікації різних типів мережевих атак шляхом аналізу поведінки мережевого трафіку. На основі отриманих результатів буде оцінюватися можливість поліпшення моделей для виявлення конкретних атак.

2.3 Підготовка сирих даних та сценарії атак

Для ефективного навчання та тестування інтелектуальної моделі вимагається значна кількість даних. Одним із ідеальних способів збору даних про події безпеки є використання цільової системи або мережі, яка є об'єктом атаки, разом з мережею нападника. Це дозволяє створити спеціалізований набір даних (dataset), який містить інформацію про мережевий трафік, журнали подій, системні логи та інше.

У даній роботі використовувався dataset Канадського інституту кібербезпеки, який включає сценарії семи типів атак: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web атаки, проникнення в мережу зсередини. Мережева інфраструктура нападника включає 50 машин, а організація-жертва має 5 департаментів, 420 користувацьких робочих станцій та 30 серверів. Dataset містить дані про перехоплений мережевий трафік та системні логи для кожної з машин.

Під час аналізу набору даних використовуються два типи профілів: B-профіль (Benign) та M-профіль (Malicious).

B-профіль (Benign) використовує різні методи машинного навчання та статистичний аналіз, такі як K-Means, Random Forest, SVM і J48, для інкапсуляції поведінки користувача. В цьому профілі враховуються розподіли розмірів пакетів протоколу, кількість пакетів на потік, структури корисного навантаження, розмір корисного навантаження і розподіл запитів за часом по протоколу. Під час моделювання використовуються такі протоколи, як HTTPS, HTTP, SMTP, POP3, IMAP, SSH і FTP. На основі початкових спостережень встановлено, що більшість трафіку належить до протоколів HTTP і HTTPS.

M-профіль (Malicious) використовується для опису сценаріїв атаки. Його метою є однозначне описання сценарію атаки, що дозволяє людям і, в ідеальному випадку, автономним агентам та компіляторам інтерпретувати та виконувати ці сценарії. Для аналізу атак розглядаються шість різних сценаріїв.

Таким чином, використання профілів B-профілю та M-профілю дозволяє описати поведінку користувачів (безпечну) та сценарії атак (зловмисну) у наборі даних.

Проникнення в мережу зсередини: У цьому сценарії атаки, зловмисники надсилають жертві шкідливий файл через електронну пошту. Після успішної експлуатації, на комп'ютері жертви встановлюється бекдор, що дозволяє зловмисникам отримати контроль над системою. За допомогою комп'ютера жертви зловмисники сканують внутрішню мережу та шукають потенційно нові поштові скриньки.

Відмова в обслуговуванні HTTP: У цьому сценарії використовуються інструменти, такі як Slowloris і LOIC, для проведення атаки на веб-сервери, які спричиняють повну недоступність серверів. Slowloris атакує, встановлюючи повне з'єднання TCP з сервером і надсилаючи неповні HTTP-запити через певні проміжки часу, щоб уникнути закриття сокетів. Цей метод використовує обмежену здатність веб-серверів обслуговувати підключення, що призводить до блокування усіх доступних сокетів та неможливості встановити нові з'єднання. Ще одним відомим інструментом, який використовується для DoS-атак на веб-сайти, є HOIC.

Атака на веб-додатки: У цьому сценарії використовується "Damn Vulnerable Web App" (DVWA) - спеціально створений додаток для допомоги фахівцям з безпеки для перевірки своїх навичок в аналізі на вразливості. Спочатку веб-сайт сканується за допомогою сканера вразливостей веб-додатків, а потім на вразливому веб-сайті виконуються різні типи атак, такі як SQL-ін'єкція, виконання системних команд та необмежена можливість завантаження файлів.

Ці сценарії атак дозволяють моделювати та аналізувати поведінку зловмисників та їх методи в наборі даних, що містить інформацію про мережевий трафік та системні логи кожної з машин.

Атака грубої сили (Brute-force attack): Цей тип атаки спрямований на перебір різних комбінацій імен користувачів та паролів з метою отримання несанкціонованого доступу до облікових записів. Існує багато інструментів, які використовуються для проведення таких атак, зокрема Hydra, Medusa, Ncrack, Metasploit і Nmap NSE. Також існують інструменти, такі як hashcat і hashpump, які використовуються для злому хеш-паролів. Patator - один з потужних багатопотокових інструментів, написаний на Python, що дозволяє проводити атаки грубої сили, більш гнучкий із можливістю зберігання відповідей в окремих файлах журналу для подальшого аналізу. У даному сценарії використовується великий словник паролів, що містить 90 мільйонів слів.

Атаки останнього оновлення: Ці атаки базуються на використанні відомих вразливостей, які можуть бути експлуатовані протягом обмеженого періоду часу. Наприклад, вразливість Heartbleed, яка вплинула на мільйони серверів і може бути

використана для отримання конфіденційної інформації з пам'яті сервера. Для виконання атаки Heartbleed використовується інструмент Heartleech, який сканує системи з метою виявлення вразливостей і їх подальшої експлуатації.

Також на ресурсах Malware traffic analysis [47] та Hybrid Analysis [48] отримані pcap файли трафіку під час атак, а за допомогою CICFlowmeter пакети перероблені у потрібний набір даних.У таблиці 2.1 наведена інформація про досліджені атаки.

Таблиця 0.1

Атаки, виконані до цільових систем

Атака	**Використовувані засоби**
Bruteforce attack	FTP – Patator SSH – Patator
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest, Heartleech
Web attack	Damn Vulnerable Web App (DVWA) Selenium Framework (XSS and Bruteforce)
Infiltration attack	Nmap and portscan
Botnet attack	Ares (developed by Python): remote shell, file upload/download, capturing screenshots and key logging
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests
C&C	XLoader, Emotet, Bazar, IcedID, Gozi

2.4 Підготовка даних до навчання

Аналіз мережевих атак можна розглядати з двох аспектів:

- аналіз мережевої активності;
- аналіз вмісту пакетів даних.

Аналіз вмісту пакетів даних є більш складним процесом і буде розглянутий у подальшій роботі.

CICFlowMeter - це інструмент для аналізу мережевого трафіку, розроблений Канадським інститутом кібербезпеки (CIC) при Університеті Нью-Брансвік. Він призначений для моніторингу та аналізу мережевого трафіку на рівні пакетів, надаючи інформацію про різні аспекти поведінки та безпеки мережі. Інструмент

фокусується на вилученні особливостей на рівні потоку з даних мережевого трафіку, що дозволяє ефективно аналізувати та виявляти мережеві аномалії та загрози.

Основне призначення CICFlowMeter - збір і обробка даних про мережеві потоки. Мережеві потоки визначаються як послідовність пакетів, які мають спільні характеристики, такі як IP-адреси джерела і призначення, порти джерела і призначення та транспортний протокол. Захоплюючи і аналізуючи ці характеристики на рівні потоку, CICFlowMeter може надати цінну інформацію про шаблони мережевого трафіку, обсяг і потенційно зловмисну діяльність.

CICFlowMeter пропонує цілий ряд функціональних можливостей і функцій, в тому числі

Моніторинг потоку: Інструмент безперервно відстежує мережевий трафік і збирає дані про потік в режимі реального часу. Він збирає різні характеристики рівня потоку, такі як кількість пакетів, загальний обсяг байт, тривалість, IP-адреси джерела та призначення, а також транспортні протоколи.

Аналіз трафіку: CICFlowMeter дає уявлення про структуру трафіку, включаючи розподіл трафіку між різними протоколами, портами та IP-адресами. Це дозволяє ідентифікувати найбільш активних користувачів, найбільш популярні сервіси та схеми зв'язку в мережі.

Виявлення аномалій: Аналізуючи характеристики на рівні потоку, CICFlowMeter може виявляти аномалії та підозрілі дії в мережі. Він використовує методи машинного навчання для виявлення відхилень від нормальної поведінки трафіку, таких як об'ємні атаки, сканування портів або мережева розвідка.

Виявлення загроз безпеці: CICFlowMeter включає в себе різні алгоритми і моделі для виявлення відомих мережевих атак і загроз. Він може ідентифікувати поширені типи атак, такі як розподілена відмова в обслуговуванні (DDoS), активність бот-мереж, поширення шкідливого програмного забезпечення та мережеві вторгнення.

Візуалізація та звітність: CICFlowMeter забезпечує візуальне представлення даних мережевого потоку, що дозволяє користувачам ефективно аналізувати та інтерпретувати інформацію. Він пропонує інтуїтивно зрозумілий графічний

інтерфейс і генерує звіти, які підсумовують спостережувану поведінку мережі і виявлені інциденти безпеки.

CICFlowMeter широко використовується для моніторингу мережевої безпеки, виявлення вторгнень і дослідження аналізу трафіку. Він допомагає мережевим адміністраторам і аналітикам безпеки отримати уявлення про поведінку мережі, виявити потенційні ризики безпеки і оперативно реагувати на інциденти. Інструмент постійно оновлюється та вдосконалюється дослідницькою командою CIC, щоб йти в ногу з мережевими загрозами, що розвиваються, та підвищувати його точність і ефективність у виявленні зловмисних дій.

Після обробки даних CICFlowMeter генерує файл у форматі CSV з шістьма основними атрибутами: FlowID (ідентифікатор потоку), SourceIP (IP-адреса джерела), DestinationIP (IP-адреса призначення), SourcePort (вихідний порт), DestinationPort (порт призначення) і Protocol (використовуваний протокол). Крім цього, програма надає понад 80 статистичних атрибутів, які можуть бути включені за необхідності.

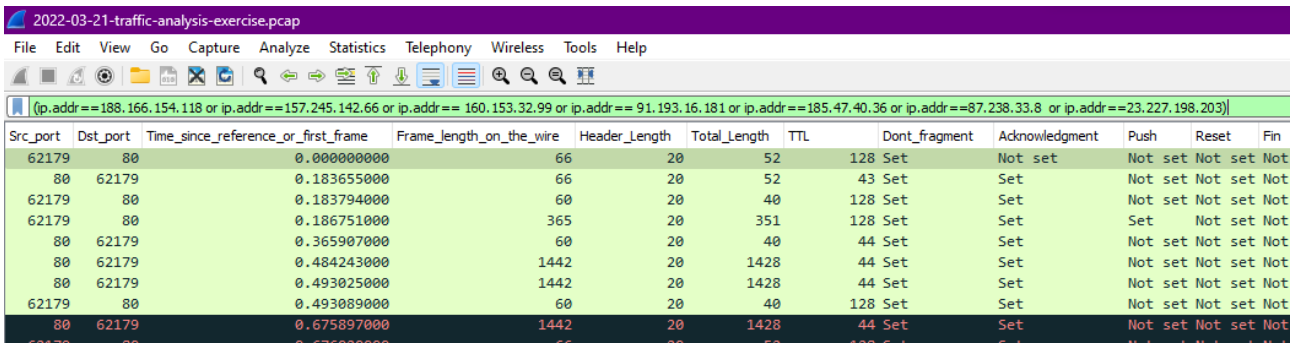
Щодо роботи програми, потоки TCP зазвичай завершуються після розриву з'єднання за допомогою пакету FIN, тоді як потоки UDP припиняються через тайм-аут потоку. Значення тайм-ауту потоку може бути встановлене залежно від власної схеми, наприклад, 600 секунд для TCP і UDP.

Після обробки даних за допомогою CICFlowMeter можна отримати різноманітні атрибути, які можуть бути вибрані та перейменовані відповідно до вимог моделювання.

З повним переліком параметрів потоку, який видає CICFlowMeter, можна ознайомитись у роботі її авторів [54].

Після формування набору даних, кожен із потоків позначили відповідно до атаки, що проходила на момент потоку, або як позитивний, якщо атака не проводилася (звичайне використання сервісів).

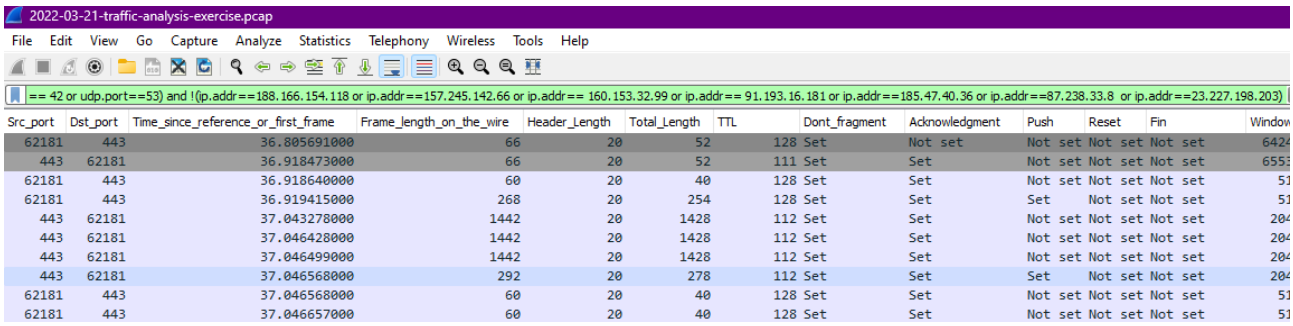
На прикладі одного pcap файлу можна простежити процес формування даних датасету. Знаходиться трафік, який стосується саме зловмисних ендпоінтів. Це показано на рисунку 2.7.



Src_port	Dst_port	Time_since_reference_or_first_frame	Frame_length_on_the_wire	Header_Length	Total_Length	TTL	Dont_fragment	Acknowledgment	Push	Reset	Fin
62179	80	0.000000000	66	20	52	128	Set	Not set	Not set	Not set	Not
80	62179	0.183655000	66	20	52	43	Set	Set	Not set	Not set	Not
62179	80	0.183794000	60	20	40	128	Set	Set	Not set	Not set	Not
62179	80	0.186751000	365	20	351	128	Set	Set	Set	Not set	Not
80	62179	0.365907000	60	20	40	44	Set	Set	Not set	Not set	Not
80	62179	0.484243000	1442	20	1428	44	Set	Set	Not set	Not set	Not
80	62179	0.493025000	1442	20	1428	44	Set	Set	Not set	Not set	Not
62179	80	0.493089000	60	20	40	128	Set	Set	Not set	Not set	Not
80	62179	0.675897000	1442	20	1428	44	Set	Set	Not set	Not set	Not

Рисунок 2.7 – Зловмисний трафік

Усі виокремлені пакети зберігаютсья як окремий pcap файл з позначкою infected. Потім усі інші пакети (рисунок 2.8) зберігаються в інший файл - benign.



Src_port	Dst_port	Time_since_reference_or_first_frame	Frame_length_on_the_wire	Header_Length	Total_Length	TTL	Dont_fragment	Acknowledgment	Push	Reset	Fin	Window
62181	443	36.805691000	66	20	52	128	Set	Not set	Not set	Not set	Not set	6424
443	62181	36.918473000	66	20	52	111	Set	Set	Not set	Not set	Not set	6553
62181	443	36.918640000	60	20	40	128	Set	Set	Not set	Not set	Not set	51
62181	443	36.919415000	268	20	254	128	Set	Set	Set	Not set	Not set	51
443	62181	37.043278000	1442	20	1428	112	Set	Set	Not set	Not set	Not set	204
443	62181	37.046428000	1442	20	1428	112	Set	Set	Not set	Not set	Not set	204
443	62181	37.046499000	1442	20	1428	112	Set	Set	Not set	Not set	Not set	204
443	62181	37.046568000	292	20	278	112	Set	Set	Set	Not set	Not set	204
62181	443	37.046568000	60	20	40	128	Set	Set	Not set	Not set	Not set	51
62181	443	37.046657000	60	20	40	128	Set	Set	Not set	Not set	Not set	51

Рисунок 2.8 – Незловмисний трафік

Навіть у середовищі Wireshark автором було обрано саме такі характеристики пакетів (рисунок 2.9), які найбільше відрізняються у пакетах, і які дійсно якось можуть вплинути на ідентифікацію зловмисного трафіку за заданим шаблоном. Тому як варіант можна аналізувати не потоки (netflow), у які будуть перероблюватись пакети потім, а одразу пакети. Але це буде складно, бо передачу різної кількості інформації можна розбити на дуже малі фрагменти.

✓	Src_port	Src port (unresolved)
	Destination	Dest addr (unresolved)
✓	Dst_port	Dest port (unresolved)
	Host	http.host or tls.handshake.ext
	Len	Packet length (bytes)
✓	Time_since_reference_or_first_frame	frame.time_relative
✓	Frame_length_on_the_wire	frame.len
✓	Header_Length	ip.hdr_len
✓	Total_Length	ip.len
✓	TTL	ip.ttl
✓	Dont_fragment	ip.flags.df
✓	Acknowledgment	tcp.flags.ack
✓	Push	tcp.flags.push
✓	Reset	tcp.flags.reset
✓	Fin	tcp.flags.fin
✓	Window	tcp.window_size_value
✓	Calculated_window_size	tcp.window_size
✓	Window_size_scaling_factor	tcp.window_size_scalefactor
✓	Bytes_sent_since_last_PSH_flag	tcp.analysis.push_bytes_sent
✓	Bytes_in_flight	tcp.analysis.bytes_in_flight
✓	Reassembled_PDU_in_frame	tcp.reassembled_in
	Conversation completeness	tcp.completeness
✓	Request_Method	http.request.method
	Connection	http.connection
✓	Time_since_first_frame_in_this_TCP_stream	tcp.time_relative
	Malformed Packet (Exception occurred)	_ws.malformed.expert
✓	Response_Phrase	http.response.phrase
✓	Status_Code	http.response.code
✓	Content-Type	http.content_type
✓	Content_length	http.content_length

Рисунок 2.9 - Обрані характеристики пакетів в Wireshark

Потім у CICFlowMeter обираються нові 2 файти та переробляються у частини датасетів. Робота у CICFlowMeter зображена на рисунку 2.10. Залишається до них додати клас і готово. Таким чином обробляються pcap файли.

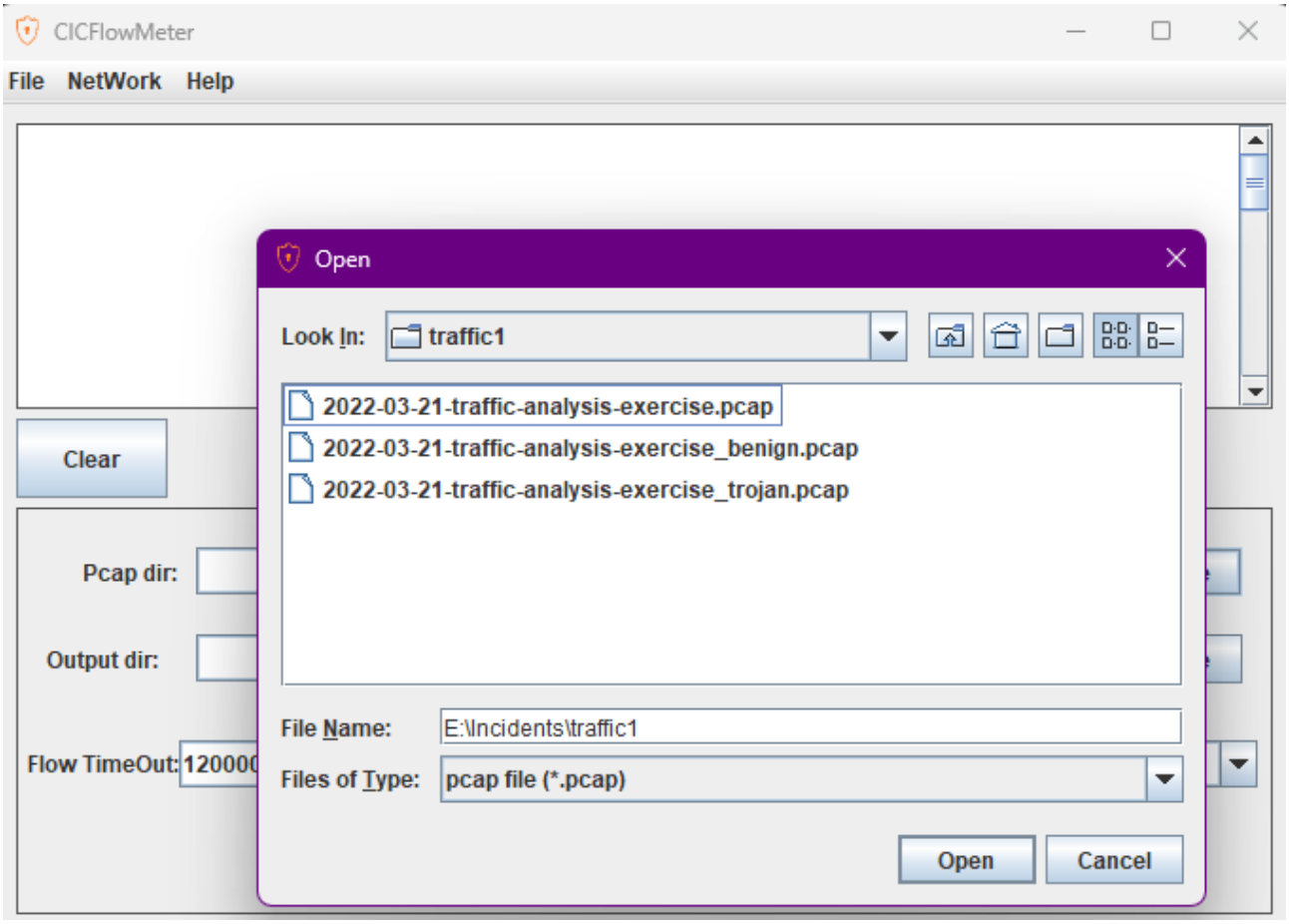


Рисунок 2.10 – Робота у CICFlowMeter для переробки мережевих пакетів

Після обробки трафіків групи зловмисного ПЗ усі результуючі файли об’єднуються в один для створення та навчання моделі машинного навчання.

Першим етапом були прибрані характеристики, які робили мінімальний вплив на кінцеве рішення. Це може допомогти у швидкодії прийняття рішень: зловмисний трафік чи ні. Прибрано наступні атрибути: Fwd PSH Flags, Fwd URG Flags, Bwd URG Flags, URG Flag Cnt, Fwd Byts/b Avg, Fwd Pkts/b Avg, Fwd Blk Rate Avg, Bwd Byts/b Avg, Bwd Pkts/b Avg, Bwd Blk Rate Avg, Init Fwd Win Byts, Fwd Seg Size Min. Після цього датасет пройшов нормалізацію деяких атрибутів. З 84 атрибутів залишились вже 72.

У фінальний набір даних було додано новий атрибут Label, що показує відповідність потоку певній атаці або позитивному трафіку. Атрибут Timestamp було видалено також, так як він не є інформативним.

Дані були помарковані відповідно до наступних значень: Benign, FTP-BruteForce, SSH-Bruteforce, DoS-GoldenEye, DoS-Slowloris, DoS-SlowHTTPTest, DoS-Hulk, DDoS attacks-LOIC-HTTP, DDoS-LOIC-UDP, DDOS-HOIC, Brute Force-Web, Brute Force-XSS, SQL Injection, Infiltration, Bot, C&C.

2.5 Підготовка моделі навчання

Багатошаровий перцептрон Румельхарта є модифікованою версією перцептрона Розенблатта, де використовується алгоритм зворотного поширення помилки для навчання всіх шарів мережі. Назва цього виду перцептрона не пов'язана з кількістю шарів, але враховується факт, що кілька шарів було використано і в перцептроні Розенблатта.

У багатошаровому перцептроні Румельхарта теоретично достатньо мати лише один шар прихованих нейронів для кодування вхідного сигналу і отримання лінійної карти для вихідного сигналу. Проте, використання декількох шарів може призвести до зменшення кількості нейронів у кожному шарі, що загалом зменшує загальну кількість нейронів у мережі. Це може поліпшити процес навчання і забезпечити кращу якість моделі.

Алгоритм зворотного поширення помилки використовується для навчання багатошарового перцептрона Румельхарта. Під час навчання, вихідна помилка порівнюється з очікуваним вихідним сигналом, і помилка поширюється назад по мережі, змінюючи ваги нейронів для мінімізації помилки. Цей процес повторюється на багатьох тренувальних прикладах, поки мережа не досягне заданої точності.

Отже, багатошаровий перцептрон Румельхарта є моделлю нейромережі, яка може бути навчена за допомогою алгоритму зворотного поширення помилки і має можливість використовувати декілька шарів для поліпшення якості навчання та зменшення загальної кількості нейронів у мережі.

Цей алгоритм реалізований у програмному продукті Weka. Weka - це відкрите програмне забезпечення, що використовується під ліцензією GNU General Public License, і включає набір алгоритмів машинного навчання для інтелектуального

аналізу даних. Воно містить інструменти для підготовки даних, класифікації, регресії, кластеризації, видобування асоціативних правил та візуалізації.

Weka також надає API (Application Programming Interface), розроблене на мові Java, що реалізує існуючі алгоритми для навчання моделей нейронних мереж. Завдання полягає у визначенні та налагодженні процесу моделювання та підготовки даних для навчання. Було створено спеціальне програмне забезпечення, що описує процес навчання та тестування даної моделі. Логіку моделі можна представити у вигляді схеми, яка наведена в таблиці 2.3.

Таблиця 0.2

Опис алгоритму для навчання нейронної мережі

Ініціалізація класової моделі
ModelGenerator mg = new ModelGenerator();
ModelGenerator - базовий клас генератора моделі. Він містить функції завантаження даних в оперативну пам’ять для постобробки.
Завантаження даних для навчання в оперативну пам’ять
Instances dataset = mg.loadDataset(DATASETPATH);
loadDataset - функція класу ModelGenerator, що реалізує інтерфейс завантаження даних з підготовленого файлу у форматі .arff.
Визначення номінальних значень з числових

Filter numToNominalFilter = new NumericToNominal();
String[] options = new String[2];
...
numToNominalFilter.setOptions(options);
numToNominalFilter.setInputFormat(dataset);
dataset = Filter.useFilter(dataset, numToNominalFilter);

Нейронна мережа представляє числові значення у певній залежності, що коригують ваги від значення величини. Під час процесу нормалізації даних залежності між певними атрибутами можуть невірно інтерпретуватись і збивати процес навчання.
Нормалізація даних
Filter filter = new Normalize(); filter.setInputFormat(dataset); Instances datasetnor = Filter.useFilter(dataset, filter);

Нормалізація даних у нейромережах – процес оптимізації значень набору даних для числових типів із великого діапазону в діапазон значень між 0 та 1 із збереженням пропорційності.
Розподіл навчальної та тестової вибірки

int trainSize = (int) Math.round(dataset.numInstances() * 0.9);
int testSize = dataset.numInstances() - trainSize;
 	
Instances traindataset = new Instances(datasetnor, 0, trainSize);
Instances testdataset = new Instances(datasetnor, trainSize, testSize);

Розбиття вибірки необхідне для первинної перевірки адекватності роботи моделі. У роботі навчальний сет складав 90 % від загальної кількості переданих даних для навчання.
Ініціалізація процесу навчання нейронної мережі
MultilayerPerceptron ann = (MultilayerPerceptron) mg.buildClassifier(traindataset);
MultilayerPerceptron – клас API Weka, що описує математичну архітектуру нейронної мережі на базі багатошарового перцептрону. buildClassifier – функція, описана в генераторі моделі, що ініціалізує екземпляр класу нейронної мережі з необхідними параметрами та виконує її навчання.
Первинне тестування моделі
String evalsummary = mg.evaluateModel(ann, traindataset, testdataset);
Функція evaluateModel використовує тестовий набір для можливості класифікації даних, та порівнює результуюче значення із шаблоном.

В додатку А дипломної роботи наведено весь код.

2.6 Навчання моделі

Для ідентифікації події з заданою точністю потрібно, щоб відносна похибка не перевищувала 4 %.

У зв’язку зі значними об’ємами даних для навчання, останні були поділені на блоки відповідно до типу атаки та прийнято рішення щодо синтезу окремої нейромережевої моделі для ідентифікації кожного типу атак.

На першому етапі досліджень було виконано синтез моделі ідентифікації FTP-SSH Brute Force атак на основі спеціалізованого набору даних, інформація по якому наведена на Рисунку 2.11.

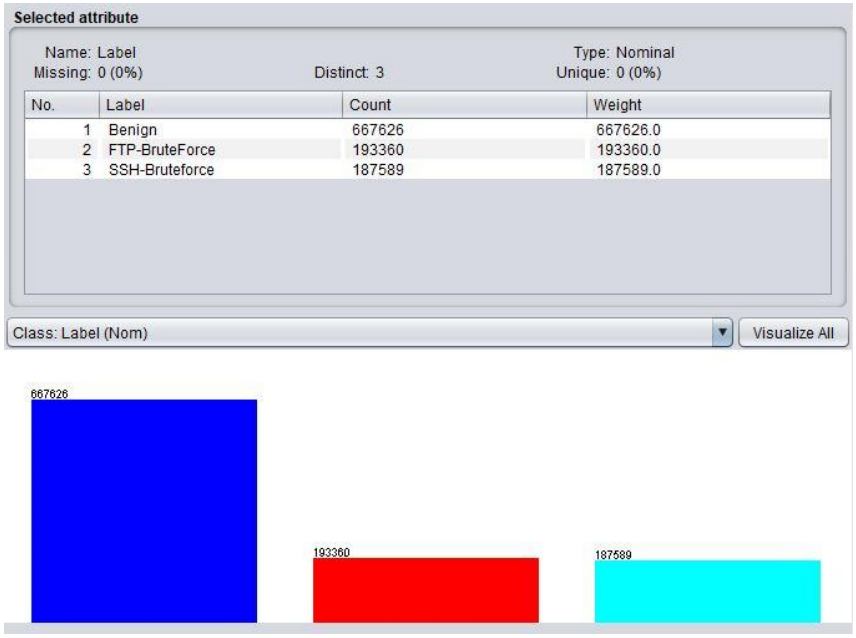


Рисунок 0.11 - Набір даних для навчання моделі виявлення FTP-SSH Brute Force атак

На прикладі наступного рисунку показано ініціалізацію алгоритму нейронної з параметрами:

```
public Classifier buildClassifier(Instances traindataset) {
    MultilayerPerceptron m = new MultilayerPerceptron();

    // m.setGUI(true);
    // m.setValidationSetSize(0);
    // m.setBatchSize("100");
    // m.setLearningRate(0.3);
    // m.setSeed(0);
    // m.setMomentum(0.2);
    m.setTrainingTime(50);// тривалість навчання (кількість епох)
    // m.setNormalizeAttributes(true);
  // m.setHiddenLayers("2,3,3") три приховані шари з 2 вузлами в першому шарі, 3 вузла в другому і 3 вузл

    /*...
    try {
        m.buildClassifier(traindataset);

    } catch (Exception ex) {
        Logger.getLogger(ModelGenerator.class.getName()).log(Level.SEVERE, null, ex);
    }
    return m;
}
```

Рисунок 0.12 - Представлення ініціалізації алгоритму та його параметрів

Learning Rate (швидкість або коефіцієнт навчання) повинна задаватися між 0 та 1, за замовчуванням = 0,3.

Momentum Rate повинен задаватися між 0 та 1, за замовчуванням = 0,2.

Кількість епох для тренування за замовчуванням = 500. Використовувалось 50 через надмірність даних.

Процентний розмір набору валідації, який використовується для припинення навчання, повинен бути між 0 та 100, за замовчуванням = 0.

Значення, яке використовується для генерації генератора випадкових чисел, повинно бути більше, або дорівнювати 0 і менше за long.

Кількість прихованих шарів повинна бути списком, розділеним комами натуральних чисел або літер 'a' = (атрибути + класи) / 2, 'i' = атрибути, 'o' = класи, 't' = атрибути + класи. За замовчуванням – 'a'.

Необхідний розмір пакету для прогнозування за замовчуванням = 100.

Для перевірки точності моделі вихідний набір даний поділився на менші: 100 000, 250 000, 500 000, 1 000 000 відповідно.

Результат навчання моделі з набором даних у 100 000 записів наведений на Рисунку 2.9.

```
Evaluation: Summary
Correctly Classified Instances       19893               99.465  %
Incorrectly Classified Instances       107                0.535  %
Kappa statistic                          0.9889
K&B Relative Info Score            1927505.1563 %
K&B Information Score                23721.0376 bits      1.1861 bits/instance
Class complexity | order 0           24377.6345 bits      1.2189 bits/instance
Class complexity | scheme             1035.0185 bits      0.0518 bits/instance
Complexity improvement     (Sf)      23342.616  bits      1.1671 bits/instance
Mean absolute error                      0.0118
Root mean squared error                  0.0606
Relative absolute error                  3.6326 %
Root relative squared error             15.0765 %
Total Number of Instances            20000

Detailed Accuracy By Class
                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1,000    0,003    0,999      1,000    0,999      0,998    0,998     0,996     SSH-Bruteforce
                 0,971    0,000    1,000      0,971    0,985      0,982    0,994     0,987     Benign
                 1,000    0,005    0,969      1,000    0,984      0,982    0,996     0,957     FTP-BruteForce
Weighted Avg.    0,995    0,002    0,995      0,995    0,995      0,993    0,997     0,989

Confusion Matrix
     a     b     c   <-- classified as
 13625     0     5 |     a = SSH-Bruteforce
    16  3425    86 |     b = Benign
     0     0  2843 |     c = FTP-BruteForce
```

Рисунок 0.13 - Результат навчання моделі та її перевірка на 20 % вихідного набору даних

Програмою був створений файл з розширенням .bin. Середній час створення моделі на цьому наборі даних складає 400 секунд. Цей файл буде використовуватися для класифікації нових даних.

Для класифікації атак за допомогою синтезованої моделі був створений спеціальний метод під назвою ClassifyInstance. Алгоритм його роботи можна знайти у таблиці 2.4.

Таблиця 0.3

Опис алгоритму використання нейронної мережі

Ініціалізація класової моделі
ModelGenerator mg = new ModelGenerator();
Ініціалізація нового екземпляру класу
Завантаження даних для навчання в оперативну пам’ять
Instances dataset = mg.loadDataset(VALIDATIONPATH);
Функція описана в таблиці 2.4.
Визначення номінальних значень з числових

Filter numToNominalFilter = new NumericToNominal();
String[] options = new String[2];
...
numToNominalFilter.setOptions(options);
numToNominalFilter.setInputFormat(dataset);
dataset = Filter.useFilter(dataset, numToNominalFilter);

Функціональність описана в таблиці 2.4.
Нормалізація даних
Filter filter = new Normalize(); filter.setInputFormat(dataset); Instances datasetnor = Filter.useFilter(dataset, filter);
Функціональність описана в таблиці 2.4.
Ініціалізація класифікатора
ModelClassifier cls = new ModelClassifier();
ModelClassifier – розроблений клас, що реалізує інтерфейс відповідності значення виходу нейронної мережі до типу ідентифікованої атаки.
Ініціалізація валідаторів
Validator[] validators = new Validator[] { new Validator("Benign"), new Validator("FTP-BruteForce"), new Validator("SSH-Bruteforce") };

Валідатори описують класи виходу мережі й використовуються для відображення значення класифікації користувачу.
Класифікація шаблону
String classname = cls.classifiy(dataset.get(i), MODElPATH);
Отримання відповіді роботи нейронної мережі
Визначення похибки роботи мережі
String defaultClass = dataset.get(i).stringValue(66); Validator.updateValidation(validators, defaultClass, classname);
Якщо дані, що передаються для перевірки класифікатора містять шаблони відповідностей, то можна вирахувати відносну похибку роботи мережі.

Для демонстрації роботи новоутвореної моделі виділили 15 записів: 5 – FTP Brute Force, 5 – Benign (звичайний трафік), 5 – SSH Brut Force потоків відповідно.

Усі дані спеціально помарковані як FTP Brute Force атака за замовчуванням і мають наступний вигляд:

```
@data
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,2,1,1,0,0,0,0,0,0,0,0,0,0,2,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,2,1,1,0,0,0,0,0,0,0,0,0,0,2,0,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
21,6,1,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,40,20,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
67,17,685,1,1,300,329,300,300,300,0,329,329,329,0,685,0,685,685,0,0,0,0,0,0,0,0,0,0,0,0,0,8,8,300,329,309
0,0,112640738,3,0,0,0,0,0,0,0,0,0,0,56320369,94.752309,56320436,56320302,112640738,56320369,94.752309,563
0,0,112640743,3,0,0,0,0,0,0,0,0,0,0,56320371.5,13.435029,56320381,56320362,112640743,56320371.5,13.435029
0,0,112640627,3,0,0,0,0,0,0,0,0,0,0,56320313.5,4401.739713,56323426,56317201,112640627,56320313.5,4401.73
0,0,112640733,3,0,0,0,0,0,0,0,0,0,0,56320366.5,51.618795,56320403,56320330,112640733,56320366.5,51.618795
22,6,555650,22,22,1928,2665,640,0,87.636364,137.780552,976,0,121.136364,258.64156,12922.093023,38554.860045
22,6,560585,22,22,1944,2665,640,0,88.363636,137.869001,976,0,121.136364,258.64156,13036.860465,48855.286144
22,6,530547,22,20,1928,2665,640,0,87.636364,137.780552,976,0,133.25,268.771254,12940.170732,45365.27566,271
22,6,533609,22,21,1960,2665,640,0,89.090909,137.953377,976,0,126.904762,263.574639,12704.97619,42668.192189
22,6,565469,22,21,1944,2665,640,0,88.363636,137.869001,976,0,126.904762,263.574639,13463.547619,45697.10508
```

Рисунок 0.4 - Дані для тестування моделі і класифікації атаки

Після запуску класифікатора отримані відповідності потоку даних до атаки, зображені на Рисунку 2.11.

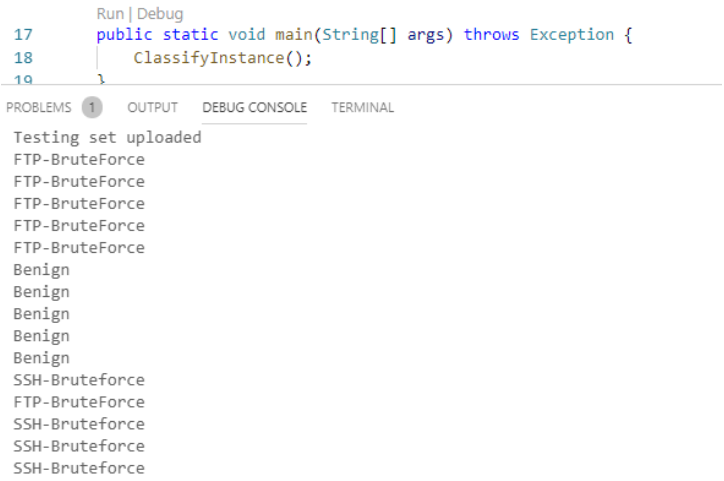


Рисунок 0.15 - Приклад класифікації атаки

Помилку показала лише одна атака, замість SSH-Bruteforce – FTP Bruteforce. Така демонстрація показує роботу мережі, але не відображає реальної доцільності. Для підтвердження адекватності її роботи, було протестовано на більших наборах даних та інших типах атак.

2.7 Аналіз адекватності роботи моделі

Для тестування моделі використовувались два додаткових набори даних для кожного з типів. Після запуску класифікатора отримана наступна інформація щодо SSH-, FTP-Bruteforce атак для сто тисячного набору, наведена на рисунку 2.16.

```
Benign: [90, 0, 0]
FTP-BruteForce: [0, 48340, 0]
SSH-Bruteforce: [0, 0, 46897]
Relative error: 0.0%
```

Рисунок 2.16 - Приклад тестування моделі

Детальна інформація по кожному з типів атак і результатах навчання наведена у таблиці 2.5.

Таблиця 0.4

Результат навчання моделей

Назва атаки	Сет для навчання	Сет для тестування 1	Сет для тестування 2	Відносна похибка, %
FTP-Bruteforce	96680	48340 / 48340	48200 / 48339	0.15
SSH-Bruteforce	93795	46897 / 46897	46801 / 46896	0.1
Dos attack GoldenEye	20753	198 / 10377	331 / 10377	97.5
Dos attack Slowloris	5495	2 / 2748	0 / 2748	98.96
Dos attack LOIC-UDP	865	0 / 433	0 / 432	100
Dos attack HOIC	343006	171503 / 171503	171503/171503	0
Dos attack HULK	230956	115478/115478	115478/115478	0
HTTP Benign	172011	18225 / 86006	15468 / 86005	80.4
Infilteration	45473	0 / 22736	0 / 22736	100
Botnet	140817	70407 / 70409	70405 / 70408	0.004
Inf-Bot Benign	469805	234902 / 234902	234902 / 234902	0
Trojan, C&C	4402	2179/2201	2154/2201	1.56

Висновки за розділом 2

У цьому розділі дипломної роботи розглянуто основні архітектури нейронних мереж, які з'явилися у період з XX до початку XXI століття. Кожна з цих архітектур вирізняється своєю концептуальною задачею, яку вона вирішує. Під час виконання дослідження було прийнято рішення використати архітектуру нейронної мережі, засновану на багатошаровому перцептроні Румельхарта, з основною метою класифікувати вхідний мережевий трафік на різні типи атак.

Для формування шаблонів даних використовувалися програмне забезпечення CICFlowMeter і розроблене додаткове програмне забезпечення на мові програмування Python. В якості джерел даних використовувалися доступні набори даних мережевих атак і pcap файли з трафіком, згенерованим зловмисним програмним забезпеченням.

Описано концепцію проектування нейромережі і представлені результати її роботи. З таблиці 2.5 можна зробити висновок, що дана модель найкраще підходить для виявлення атак типу Bruteforce на сервіси FTP і SSH, а також для виявлення Dos-атак з використанням HOIC, активності ботнетів HULK і трафіку, пов'язаного з C2-серверами.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛІ ІДЕНТИФІКАЦІЇ АТАК МЕРЕЖЕВИХ СЕРВІСІВ

3.1 Опис класифікації атак на веб-сервіси

Веб-сервіс є сервісом, який працює на комп'ютерному пристрої та використовує протокол HTTP (HyperText Transfer Protocol).

HTTP належить до протоколів моделі OSI (Open Systems Interconnection) на 7-му прикладному рівні і є одним з найпоширеніших протоколів в Інтернеті. Згідно з міжнародними даними, на 2020 рік кількість веб-сайтів, опублікованих в Інтернеті, перевищує 1,7 мільярдів [50]. Він слухає запити на певному порту через мережу і надає веб-документи (такі як HTML, JSON, XML, зображення), вирішуючи певні проблеми в Інтернеті.

За методологією OWASP (Open Web Application Security Project) можна виділити топ-10 кібернетичних атак на веб-сервіси. Найпоширенішою та найкритичнішою серед них є ін'єкція. Такі типи ін'єкцій, як SQL, NoSQL, ОС та LDAP, виникають, коли недовірчі дані передаються інтерпретатору як частина команди або запиту, і вони виконуються, надаючи доступ до критичних ресурсів або сервісів.

Broken Authentication - це ситуація, коли функції програми, пов'язані з аутентифікацією та керуванням сеансом користувача, реалізовані некоректно, що дозволяє зловмисникам скомпрометувати паролі, ключі, токени або інші ідентифікатори вже авторизованих користувачів.

Серед веб-програм і API поширені неефективні засоби захисту конфіденційних даних, що призводить до витоку конфіденційної інформації. Відсутність конфіденційності стосується відкрито переданих даних та помилок, допущених під час налаштування доступу до інформаційних ресурсів.

Зовнішні сутності в XML (XXE) становлять загрозу безпеці для багатьох препроцесорів XML, які застаріли або погано налаштовані. Зловмисники можуть

використовувати посилання на зовнішні ресурси в документах XML для виконання віддаленого коду, сканування внутрішніх портів, ініціювання атак на відмову в обслуговуванні, використання обробників файлів URI для спільного використання внутрішніх файлів тощо.

Часто засоби автоматизації не застосовують обмеження контролю доступу, накладені на автентифікованих користувачів, що призводить до порушення цих обмежень. Це надає зловмисникам можливість отримати доступ до важливих ресурсів, що належать користувачам і організаціям.

Використання стандартних конфігурацій, відображення помилок під час обробки запиту та потенційне включення конфіденційної інформації є поширеними проблемами, які виникають через неправильне налаштування інструментів безпеки. Це можна вважати головною проблемою, пов’язаною з роботою веб-додатків, що виникає через неадекватні або персоналізовані конфігурації та неправильно налаштовані заголовки HTTP.

Ненадійні дані, включені в новий веб-сайт без відповідної перевірки чи видалення, або використання API браузера для додавання створених користувачами даних до наявної сторінки можуть спричинити проблему міжсайтового сценарію XSS. Жертви XSS-атак можуть зіткнутися з виконанням сценаріїв у своїх браузерах, що може вплинути на сеанси користувачів, безпеку веб-сайтів або навіть перенаправити їх на зловмисні сайти.

Через віддалене виконання коду небезпечна десеріалізація може призвести до небезпечної неправильної обробки даних.

Використання вразливого компонента є проблемою, що призводить до втрати даних або зламу сервера. Ці компоненти, як-от бібліотеки, фреймворки та програмні модулі, мають однакові програмні привілеї. Коли програми та API використовують компоненти з відомими вразливими місцями, їх безпека значно знижується. Ось чому вкрай важливо уникати використання цих компонентів взагалі.

З поганим обліком і моніторингом, а також з неефективною автоматизацією реагування на IT-інциденти можуть статися порушення безпеки, і зловмисники можуть проникнути в системи без виявлення, маючи при цьому доступ до важливих

даних і маніпулювання ними. Процеси та інструменти внутрішнього моніторингу часто неефективні для виявлення цих інцидентів, а розслідування зовнішніми сторонами займає в середньому приблизно 200 днів. Аналізуючи даний контекст та результат попереднього розділу, було виявлено необхідність деталізувати модель для одного з типів атак на веб-сервіси, а саме SQL-ін’єкцію.

3.2 Розробка моделі ідентифікації SQL-атаки

3.2.1 Опис атаки

Використовуючи вразливості веб-додатків, виконується впровадження SQL. Конфіденційна інформація може бути змінена, знищена або отримана в результаті успішної атаки. Structured Query Language є основним методом атаки для модифікації запитів до бази даних.

Будь-яка веб-програма, яка використовує базу даних SQL, може бути уражена вразливістю SQL-ін’єкції. Серед потенційних баз даних – SQL Server, PostgreSQL, Oracle, MySQL, MSSQL та інші.

Згідно з методологією OWASP, атаки SQLi потрапили до десятки найпоширеніших атак.

Атаки SQL-ін’єкції можуть бути дуже серйозними, оскільки SQL, мова запитів для керування реляційними базами даних, зазвичай використовується веб-сайтами. Насправді бази даних SQL часто зберігають дані, які використовуються веб-сайтами. Що ще гірше, SQL можна навіть використовувати для виконання команд операційної системи.

Ключі до виконання атаки SQL-ін’єкцій знаходяться у веб-додатку, який робить себе вразливим до таких атак, і в параметрах, які користувач вводить безпосередньо в свій SQL-запит. Тоді зловмисник може маніпулювати вхідним вмістом (або корисним навантаженням) на свою користь. У результаті цього виконуються неочікувані команди після того, як зловмисник надсилає спеціально створений SQL-запит [51].

Форма SQL-запиту зазвичай містить:

select id, forename, surname from authors

Ідентифікаційні номери, імена та прізвища авторів можна отримати, вибравши ідентифікатор, ім’я та прізвище з їх бази даних.

Щоб отримати певну інформацію з таблиці «автори», скористайтеся цим запитом для отримання стовпців «id», «forename» і «surname». Відповідь з бази даних буде складатися з рядків таблиці. Щоб уточнити відповідь сервера, необхідно визначити запит.

select id, forename, surname from authors where forename = 'john' and surname = 'smith'

john smith є критеріями вибору ідентифікатора, імені та прізвища з таблиці авторів.

Параметри рядка, розділені одинарними лапками, містять важливу деталь.

Ці значення, розділені одинарними лапками, є вирішальними для параметрів рядка. Якщо припустити, що вони засновані на введенні користувачем з веб-форми, зловмисник має можливість змінити вихідний SQL-запит, ввівши ряд параметрів. Важливо пам’ятати про це.

Forename: jo'hn

Surname: smith

Результуючий запит буде виглядати так:

select id, forename, surname from authors where forename = 'jo'hn' and surname = 'smith'

Тому для авторів запиту виберіть ім’я, прізвище та ідентифікатор, але лише якщо ім’я – «Джон», а прізвище – «Сміт».

Подібно до наступної помилки, яка виникне, коли запит буде виконано базою даних:

Server: Msg 170, Level 15, State 1, Line 1

Line 1: Incorrect syntax near 'hn'.

Таблиця авторів буде стерта, якщо зловмисник введе "jo'; скиньте автори таблиці--" як значення для поля "ім'я", через те, що запит розділено встановленням

одинарних лапок. Це спонукає до виконання другого блоку, "hn", після того, як поле "forename" встановлено як "jo", і викликає помилку. Запит розбитий, що сприяє виникненню цієї помилки, що приведе до втрати даних. Причиною цього є наявність однорядкової послідовності символів коментаря в розширеному Transact-SQL, яка представлена «--». Його мета полягає в тому, щоб забезпечити роботу запиту без будь-яких помилок, оскільки інші частини основного запиту вважаються системою бази даних коментарем і, отже, не виконуються. Щоб відокремити запити, ";" використовується символ. Хоча існує багато способів маніпулювання базою, це може призвести до значної шкоди та збитків.

Атаки SQL-ін’єкцій можуть виконуватися різними методами: шляхом введення коду в поля введення або модифікації файлів cookie, через атаку між серверами, через команди SQL у заголовках HTTP або використання вразливостей у веб-додатках.

- помилки бази даних або команди UNION можна використовувати для виконання внутрішньосмугового SQLi.

Вилучення логічних операцій призводить до аналізу відповідей із бази даних у процесі, відомому як сліпий SQLi.

Використовуючи альтернативний підхід, позасмугове впровадження SQL відбувається, коли зловмисники розгортають окремі канали для отримання інформації. Результат такої атаки часто залежить від того, які функції ввімкнено на сервері бази даних [52].

DevOps має вжити низку запобіжних заходів, щоб забезпечити безпеку кінцевих користувачів і зменшити вплив цієї атаки на роботу бізнес-сервісів.

3.2.2 Побудова моделі

Основною метою роботи було створення логічного модуля, що зможе віднести вхідний HTTP-запит до SQLi-атаки.

Модель, що пропонується складається з трьох основних компонентів:

- генератор URL (Uniform Resource Locator) адрес;
- класифікатор URL адрес;

- модель на базі нейронної мережі.

Тестовий модуль, генератор URL-адрес, має вирішальне значення для створення початкового набору даних і складається з двох окремих компонентів. Перший компонент передбачає генерацію нормальних URL-адрес шляхом збору даних з популярних веб-сайтів, наприклад, аналізу файлів sitemap.xml. Другий компонент включає генерування шкідливих URL-адрес шляхом додавання типових параметрів URL-запиту для SQL-ін'єкції до URL-адрес, отриманих попереднім методом. Вибірка також може бути заповнена шкідливими параметрами з використанням наборів даних з відкритих джерел.

Ключові слова, написані на мові SQL, часто присутні в SQL-запитах на ін'єкцію. Ці ключові слова використовуються для виконання різних операцій над таблицями бази даних, які можуть бути застосовані як до всієї бази даних, так і до окремих таблиць. Наприклад, створення або видалення таблиць на відміну від редагування записів і уточнення пошуку застосовуються до різних рівнів деталізації.

Типовими параметрами можуть бути:

- що починаються на «'»;
- що закінчуються на «--», «/* */», «#»;
- містять UNION, SELECT і FROM, information_schema, exec, логічні оператори (OR, AND, =) та вирази (1=1);
- групові фрази ADMIN DROP, CREATE TABLE, DELETE FROM, INSERT INTO тощо.

Для аналізу та синтезу моделі ідентифікації атак автору необхідно зібрати числові дані за допомогою навчальних, контрольних та тестових наборів даних. Для цього було розроблено модуль класифікатора URL-адрес, який може конвертувати URL-адреси в двійковий формат, ідентифікуючи їх як атаку чи ні. Програмне забезпечення генерує вхідний вектор для кожної URL-адреси на основі проаналізованих основних ознак атак. Нейромережева модель вимагає лише числових даних у певному діапазоні, тому параметри URL-адрес були обрані евристично, щоб відповідати їй. Параметри, що аналізувались, представлено в табл. 3.1.

Таблиця 0.1

Призначені вектори шаблонам SQL параметрів

Номер параметра вхідного вектору	Параметр запиту
x_1	' (одинарні лапки)
x_2	CREATE
x_3	DELETE FROM
x_4	DROP TABLE
x_5	INSERT INTO
x_6	UNION
x_7	AND
x_8	OR
x_9	--
x_{10}	(пробіл)
x_{11}	FROM
x_{12}	EXEC

Таким чином, вхідний вектор можна представити у форматі:

$$X^{\mathrm{T}} = [x_1 x_2 \cdots x_n]$$

Для прикладу, при n=12 параметрів, URL, що містить вираз «CREATE TABLE … AND INSERT INTO…», буде представлена у вигляді: 010010100100.

Базуючись на такому підході, нейромережева модель матиме на вході 12 нейронів.

Кожний вхідний вектор характеризується параметром: Benign (0) – не відноситься до атаки, та Injection (1) – відноситься до атаки. Для цього достатньо на виході мати лише один нейрон, що розділяє вектори входу на два класи 0 і 1.

3.2.3 Навчання та аналіз моделі

Node JS використовувався разом із Synaptic, стороннім модулем, для створення програмного забезпечення, необхідного для навчання моделей.

Для тих, хто використовує node.js, існує бібліотека нейронних мереж JavaScript під назвою Synaptic, яка дозволяє розробляти та інструкції для широкого спектру налаштувань нейронних мереж першого або навіть другого порядку [52].

З архітектурою моделі, що складається з трьох нейронних шарів, прихований і вхідний рівні покладаються на функцію сигмоїдної активації, тоді як вихідний рівень використовує функцію лінійної активації. Модель розроблена з 12 нейронами для вхідного шару та шістьма для прихованого шару. Крім того, є лише один вихід. Коли мова заходить про вхідний вектор, він складається з 12 елементів X із значеннями в діапазоні від [0,1]. Для належної ініціалізації моделі використовувалися випадкові значення з інтервалу [-1 1].

Дотримуючись цих правил, було встановлено кількість нейронів у прихованому шарі:

«Кількість вузлів у вхідному шарі та вихідному шарі відіграє вирішальну роль у визначенні ідеального розміру прихованого шару, який зазвичай потрапляє в цей діапазон.

Під час визначення кількості нейронів для цього шару розрахунок виконується за допомогою середньої кількості нейронів, присутніх у вихідному та вхідному шарах.

Досягнення оптимальної конфігурації залежить від визначення різних параметрів, таких як значення помилки, швидкість навчання та інші, щоб визначити найкращий результат під час налаштування нейронної мережі.

В алгоритмах оптимізації, які використовуються в машинному навчанні та статистиці, існує параметр, відомий як швидкість навчання, який визначає розмір кожного кроку, зробленого під час ітерації для досягнення мінімуму функції помилки. Він позначається значенням від 0 до 1 і регулюється. [53]

Використання коефіцієнта навчання 0,2 послужило основою для висновків експерименту.

У цій конкретній спробі було зазначено, що для припинення навчання було використано не більше 200 ітерацій, якщо не було досягнуто мінімальної помилки.

Ця конкретна нейронна мережа мала граничну точку для навчання, також відому як мінімальна помилка. Значення похибки, що визначає межу, становило 0,005, як зазначено в дослідженні.

Тест був розділений на три частини: 15% контрольної, 15% тестової та 70% навчальної серії.

Для формування зразків було згенеровано 30 233 URL-адреси з максимальною різноманітністю.

У навчальному наборі з 20 182 записів 7 269 URL-адрес були зловмисними, а 12 913 — нормальними.

Шкідливі записи становили 1808 із 5025 записів у контрольній вибірці, а решта 3217 записів були нормальними.

Шкідливі файли становили меншу частку тестової вибірки порівняно зі звичайними. Із загальної кількості 5026 записів було 1809 файлів, які вважалися шкідливими, а решта 3217 були віднесені до категорії звичайних.

Синтезована модель після тривалого навчання продемонструвала вражаючий рівень точності до 95% при класифікації вхідних даних.

Проведений як на малих, так і на великих вибірках, навчальний графік моделі відображає зміну помилки навчання після кожної ітерації. Графік зображено нижче для довідки.

Рисунок 0.1 - Графік навчання на малій вибірці

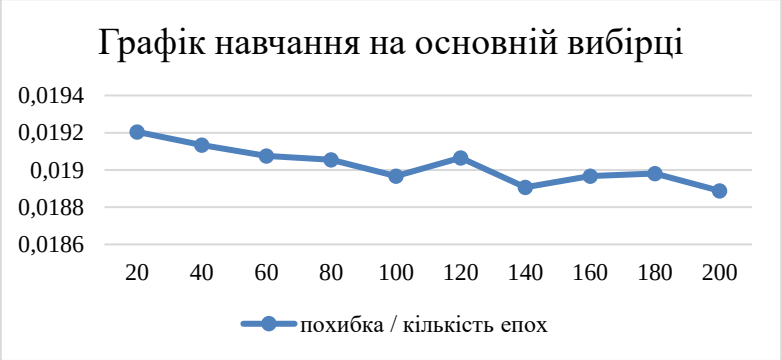


Рисунок 0.2 - Графік навчання на основній вибірці

На наведеному графіку похибка навчання для основної вибірки зображена у вигляді відносного значення 1,89%.

Можливі перезаписи:

- Під час роботи нейронної мережі може виникнути проблема, коли певні параметри впровадження SQL з'являються в URL-адресі у випадковій послідовності, фактично не становлячи загрози для веб-сервера чи бази даних (також відомий як «помилковий результат»). Однак, незважаючи на такий доброякісний характер, система позначає їх як атаку. Щоб вирішити цю проблему під час майбутніх розслідувань, ми можемо досліджувати конкретні шаблони впровадження SQL, а також брати до уваги коди відповідей HTTP веб-сервера.

- Іноді, коли нейронна мережа працює, вона стикається зі сценарієм, коли змінні SQL-ін'єкції, що вивчаються, не дотримуються передбачуваної послідовності в URL-адресі та фактично не завдають жодної шкоди веб-серверу або базі даних (це називається «хибно-позитивний результат»). Тим не менш, система все одно класифікує їх як спробу злому. Щоб уникнути цього ускладнення в наступних дослідженнях, ми могли б використовувати більш чіткі шаблони ін'єкцій SQL і включити коди статусу HTTP у відповіді веб-сервера.

- Під час роботи з нейронною мережею ми можемо зіткнутися з ситуацією, коли параметри SQL-ін'єкцій, які ми аналізуємо, проявляються у випадковому порядку в URL-адресі та не мають негативного впливу ні на веб-сервер, ні на базу даних (так званий " Помилковий результат"), але система позначає їх як вторгнення. Вирішення цієї складної ситуації в майбутній роботі може включати прийняття більш точних

моделей впровадження SQL і врахування статусів відповідей HTTP, які повертає веб-сервер.

З таблиці класифікації видно, що відносна похибка при використанні синтезованої моделі як на контрольних, так і на дослідних зразках не перевищує 5%. Підсумовуючи, це те, що було зазначено раніше в розділі 3.2.

Для розрізнення SQL-ін’єкцій аналіз змодельованих результатів дослідних даних показує, що запропонований нами метод може досить точно розпізнавати ці атаки на рівні 95,11%.

Таблиця 0.2

Результат навчання моделі

Вид трафіку	Навчальна	Контрольна (успішних/всього)	Тестова (успішних/всього)	Відносна похибка, %
Звичайний	12913	3196 / 3217	3190 / 3217	0,7 %
SQLi	7269	1719 / 1808	1721 / 1809	4,89 %

Висновки за розділом 3

Таким чином, аналіз результатів моделювання для тестової множини даних дозволяє зробити висновок, що запропонований підхід для ідентифікації SQL-ін’єкцій у цілому дозволяє виконати ідентифікацію атак даного типу з точністю 95,11 %. Підвищення точності ідентифікації, на нашу думку, можливо за рахунок навчання моделі значно на більших множинах даних.

ВИСНОВКИ

Розробка моделей, які використовують нейронні мережі для виявлення мережевих атак за допомогою алгоритму зворотного поширення помилок на основі багатошарового перцептрона, була головною темою роботи.

Під час виконання дослідницького завдання проводився аналіз даних та збір даних для моделювання. Були розроблені алгоритми попередньої обробки, синтезовані та навчені моделі нейронних мереж. Зрештою, за результатами дослідження було проаналізовано та визначено адекватність синтезованих моделей.

Зважаючи на результати дослідження, модель аналізу мережевого потоку виявляється корисною для виявлення атак Bruteforce на служби FTP і SSH, Dos-атак з використанням HOIC, активності ботнету HULK і підключень до серверів C2. Він також може відігравати роль модуля виявлення для наступних систем SIEM.

Брандмауер веб-застосунків може використовувати синтезовану модель для точної ідентифікації SQL-ін’єкцій зі швидкістю 95,11%.

Подальшою роботою в даній сфері є розробка інтелектуальних моделей для виявлення більшого спектру атак на мережеві сервіси, централізація моделей в єдиному програмному забезпечені та візуалізація їх роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MarketsandMarkets. “Artificial Intelligence in Cybersecurity Market by Security, Growth Drivers – 2028” [Електронний ресурс]. – Режим доступу: https://www.marketsandmarkets.com/Market-Reports/artificial-intelligence-security-market-220634996.html.

2. Statista. “AI in cyber security market size 2027” [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/1291380/ai-in-cyber-security-market-size/.

3. ThoughtLab. “Cybersecurity Solutions for a Riskier World” [Електронний ресурс]. – Режим доступу: https://thoughtlabgroup.com/cyber-solutions-riskier-world/.

4. IBM. “AI and automation for cybersecurity” [Електронний ресурс]. – Режим доступу: https://www.ibm.com/thought-leadership/institute-business-value/report/ai-cybersecurity.

5. Офіційний вебпортал парламенту України. “Про основні засади забезпечення кібербезпеки України” [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2163-19.

6. Державна служба спеціального зв'язку та захисту інформації України. “Перелік категорій кіберінцидентів” [Електронний ресурс]. – Режим доступу: https://www.cip.gov.ua/ua/news/perelik-kategorii-kiberincidentiv.

7. ENISA. “Reference Incident Classification Taxonomy” [Електронний ресурс]. – Режим доступу: https://www.enisa.europa.eu/publications/reference-incident-classification-taxonomy.

8. Europol. “Common Taxonomy for Law Enforcement and The National Network of CSIRTs” [Електронний ресурс]. – Режим доступу: https://www.europol.europa.eu/sites/default/files/documents/common_taxonomy_for_law_enforcement_and_csirts_v1.3.pdf.

9. Gartner. “Applying Network-Centric Approaches for Threat Detection and Response” [Електронний ресурс]. – Режим доступу: https://www.gartner.com/en/documents/3904768.

10. Abnormal. “Indicators of Compromise (IOCs): How They Work, How to Identify Them, and Why They Aren't Enough” [Електронний ресурс]. – Режим доступу: https://abnormalsecurity.com/glossary/indicators-of-compromise.

11. Paul Pols. “The Unified Kill Chain” [Електронний ресурс]. – Режим доступу: https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain.pdf.

12. Lockheed Martin. “Cyber Kill Chain” [Електронний ресурс]. – Режим доступу: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html.

13. MITRE ATT&CK. “MITRE ATT&CK” [Електронний ресурс]. – Режим доступу: https://attack.mitre.org/.

14. VirusTotal. “YARA - The pattern matching swiss knife for malware researchers” [Електронний ресурс]. – Режим доступу: https://virustotal.github.io/yara/.

15. BleepingComputer. “PureCrypter malware hits govt orgs with ransomware, info-stealers” [Електронний ресурс]. – Режим доступу: https://www.bleepingcomputer.com/news/security/purecrypter-malware-hits-govt-orgs-with-ransomware-info-stealers/.

16. CERT-UA. “Кібератака групи UAC-0050 (UAC-0096) з використанням програми Remcos (CERT-UA#6011)” [Електронний ресурс]. – Режим доступу: https://cert.gov.ua/article/3931296.

17. Medium. “How to Build a Machine Learning Model” [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/how-to-build-a-machine-learning-model-439ab8fb3fb1.

18. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018

19. Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection,

Computers & Security, Volume 31, Issue 3, May 2012, Pages 357-374, ISSN 0167-4048, 10.1016/j.cose.2011.12.012.

20. M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, “A Detailed Analysis of the KDD CUP 99 Data Set”, Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.

21. Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019.

22. Hossein Hadian Jazi, Hugo Gonzalez, Natalia Stakhanova, and Ali A. Ghorbani. "Detecting HTTP-based Application Layer DoS attacks on Web Servers in the presence of sampling." Computer Networks, 2017.

23. AF. Yazı, FÖ Çatak, E. Gül, Classification of Metamorphic Malware with Deep Learning (LSTM), IEEE Signal Processing and Applications Conference, 2019.

24. Catak, FÖ., Yazi, AF., A Benchmark API Call Dataset for Windows PE Malware Classification, 2019.

25. Yang, Limin and Ciptadi, Arridhana and Laziuk, Ihar and Ahmadzadeh, Ali and Wang, Gang, BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware, 4th Deep Learning and Security Workshop, 2021.

26. Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, Emil C. Lupu. "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection.", 2016.

27. Akcora, C.G., Li, Y., Gel, Y.R. and Kantarcioglu, M., 2019. BitcoinHeist: Topological Data Analysis for Ransomware Detection on the Bitcoin Blockchain. 2020.

28. M. Hirano, R. Hodota, R. Kobayashi, “RanSAP: An open dataset of ransomware storage access patterns for training machine learning models”, Forensic Science International: Digital Investigation, Volume 40, 2022, – Режим доступу: https://doi.org/10.1016/j.fsidi.2021.301314.

29. M. Hirano and R. Kobayashi, "Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained From Live-forensic Hypervisor," 2019 Sixth

International Conference on Internet of Things: Systems, Management and Security (IOTSMS), 2019, pp. 1-6, – Режим доступу: https://doi.org/10.1109/IOTSMS48152.2019.8939214.

30. G. Zhao, K. Xu, L. Xu and B. Wu, "Detecting APT Malware Infections Based on Malicious DNS and Traffic Analysis," in *IEEE Access*, vol. 3, pp. 1132-1142, 2015.

31. https://www.snort.org/

32. Nizar Kheir, Behavioral classification and detection of malware through HTTP user agent anomalies, Journal of Information Security and Applications, Volume 18, Issue 1, 2013, Pages 2-13, ISSN 2214-2126, – Режим доступу: https://doi.org/10.1016/j.jisa.2013.07.006.

33. D. Bekerman, B. Shapira, L. Rokach and A. Bar, "Unknown malware detection using network traffic classification," 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 2015, pp. 134-142,.

34. Boukhtouta, A., Mokhov, S.A., Lakhdari, NE. et al. Network malware classification comparison using DPI and flow packet headers. J Comput Virol Hack Tech 12, 69–100 (2016). – Режим доступу: https://doi.org/10.1007/s11416-015-0247-x

35. Roberto Perdisci, Wenke Lee, Ollmann, Method and system for network-based detecting of malware from behavioral clustering. 2016.

36. Kaggle “Trojan Detection” [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/datasets/subhajournal/trojan-detection.

37. Panigrahi, Ranjit & Borah, Samarjeet. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology. 7. 479-482.

38. (2023). Network intrusion detection using data dimensions reduction techniques. Journal of Big Data. 10. 10.1186/s40537-023-00697-5.

39. Хайкін С. Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation / Саймон Хайкін. – Москва: Вільямс, 2006. – 1104 с. – (2).

40. Sumit S. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс] – Режим доступу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

41. LSTM: A Search Space Odyssey / K.Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink., 2015. – 12 с.

42. Hopfield J. J. Neural networks and physical systems with emergent collective computational abilities / Hopfield. // Proceedings of National Academy of Sciences. – 1982. – №79. – С. 2554–2558.

43. Ackley D. H. A Learning Algorithm for Boltzmann Machines / D. H. Ackley, G. E. Hinton, T. J. Sejnowski. // Cognitive Science. – 1985. – №9. – С. 147–169.

44. Deep Belief Networks [Електронний ресурс] // DeepLearning – Режим доступу: http://deeplearning.net/tutorial/DBN.html.

45. Autoencoder for Words / C.Liou, C. Cheng, J. Liou, D. Liou. // Neurocomputing. – 2014. – №139. – С. 84–96.

46. Anthony J. P. Artificial Neural Networks Using Multilayer Perceptron [Електронний ресурс] / P. Anthony J., Dong Soo Kim – Режим доступу: https://www.cse.unsw.edu.au/~cs9417ml/MLP2/index.html.

47. Malware Traffic Analysis, [Електронний ресурс] – Режим доступу: https://www.malware-traffic-analysis.net/index.html.

48. Free Automated Malware Analysis Service - powered by Falcon Sandbox, [Електронний ресурс] – Режим доступу: https://www.hybrid-analysis.com/.

49. WEKA Manual for Version 3-9-3 / [R. Bouckaert, E. Frank, M. Hall та ін.]. – Hamilton: University of Waikato.

50. Total number of Websites [Електронний ресурс] // Internet Live Stats – Режим доступу до ресурсу: https://www.internetlivestats.com/total-number-of-websites/.

51. OWASP. SQL Injection [Електронний ресурс] / OWASP – Режим доступу до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection.

52. What is SQL Injection (SQLi) and How to Prevent It [Електронний ресурс] // Acunetix. – Режим доступу: https://www.acunetix.com/websitesecurity/sql-injection/.

53. Murphy K. P. Machine Learning: A Probabilistic Perspective / Kevin Murphy. – Cambridge: MIT Press, 2012. – 247 с.

54. Habibi Lashkari, Arash. (2018). CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. [Електронний ресурс] – Режим доступу: https://github.com/ISCX/CICFlowMeter.

ДОДАТОК А. Список ідентифікаторів кіберінцидентів

Реалізація нейромережевої моделі для ідентифікації SQL-ін’єкції

network.js

```
const synaptic = require("synaptic");
const csv = require("csv-parser");
const path = require("path");
const fs = require("fs");
const Architect = synaptic.Architect;
const Trainer = synaptic.Trainer;
const Classifier = require("./classifier");

var fuzzingSet = Classifier.getDataSet("injection.txt");
var benignSet = Classifier.getDataSet("clean.txt");

var network = new Architect.Perceptron(12, 6, 1);
var myTrainer = new Trainer(network);

// let finalSet = [];
// for (let i = 0; i < fuzzingSet.length; i++) {
//   if (i % 2 == 0)
//     finalSet.push({
//       input: fuzzingSet[i],
//       output: [1]
//     });
//   else
//     finalSet.push({
//       input: benignSet[i],
//       output: [0]
//     });
// }

function getInjectionDataFromCsv(filePath) {
  return new Promise(resolve => {
    let results = [];
    fs.createReadStream(filePath)
      .pipe(csv())
      .on("data", data => results.push(data))
      .on("end", () => {
        let sqliT = results
          .filter(el => el.attack_type == "sqli")
          .map(el => Classifier.binaryView(el.payload))
          .concat(fuzzingSet);
        let benignT = results
          .filter(el => el.attack_type == "norm")
          .map(el => Classifier.binaryView(el.payload))
```

```
          .concat(benignSet);

        var mySet = [
          ...benignT.map(content => {
            return {
              input: content,
              output: [0]
            };
          }),
          ...sqliT.map(content => {
            return {
              input: content,
              output: [1]
            };
          })
        ];
        let result = myTrainer.train(mySet, {
          rate: 0.02,
          iterations: 200,
          error: 0.005,
          shuffle: true,
          log: 20
        });

        console.log(result);

        console.log("Benign train set:", benignT.length);
        console.log("SQLi train set:", sqliT.length);
        resolve(result);
      });
  });
}

function controlTest() {
  console.log(
    network.activate(Classifier.binaryView("/hello/i'm benign/select from"))
  );
  console.log(
    network.activate(
      Classifier.binaryView(
        "/en/brand/k-m-by-l-a-n-g-e-en/?category=midi-
skirt exec master..xp_cmdshell 'ipconfig+/all'"
      )
    )
  );
  console.log(
    network.activate(
      Classifier.binaryView(
        "' union (select NULL, NULL, NULL, NULL,  NULL, (select @@version)) --"
      )
    )
```

```
  );
  console.log(network.activate([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]));

  let controlBenign = Classifier.getDataSet("control-b.txt");
  let controlInjection = Classifier.getDataSet("control-i.txt");

  for (let i = 0; i < controlBenign.length; i++) {
    let r = network.activate(controlBenign[i]);
    console.log("Benign:", r);
  }

  for (let i = 0; i < controlInjection.length; i++) {
    let r = network.activate(controlInjection[i]);
    console.log("Injection:", r);
  }

  console.log(
    "Контрольна вибірка:",
    controlBenign.length + controlInjection.length,
    controlBenign.length,
    controlInjection.length
  );
}

getInjectionDataFromCsv(
  path.join("HttpParamsDataset", "payload_train.csv")
).then(model => {
  let results = [];
  fs.createReadStream(path.join("HttpParamsDataset", "payload_test.csv"))
    .pipe(csv())
    .on("data", data => results.push(data))
    .on("end", () => {
      let sqliT = results
        .filter(el => el.attack_type == "sqli")
        .map(el => Classifier.binaryView(el.payload));

      let benignT = results
        .filter(el => el.attack_type == "norm")
        .map(el => Classifier.binaryView(el.payload));

      console.log(
        "Benign test set:",
        benignT.slice(0, Math.floor(benignT.length / 2)).length
      );
      console.log(
        "SQLi test set:",
        sqliT.slice(0, Math.floor(sqliT.length / 2)).length
      );
      let result = { success_b: 0, error_b: 0, success_i: 0, error_i: 0 };

      for (let record of sqliT.slice(0, Math.floor(sqliT.length / 2))) {
```

```
        let r = network.activate(record)[0];
        if (r < 0.5) result.error_i++;
        else result.success_i++;
      }
      for (let record of benignT.slice(0, Math.floor(benignT.length / 2))) {
        let r = network.activate(record)[0];
        if (r >= 0.5) result.error_b++;
        else result.success_b++;
      }
      console.log("Test result:", result);

      console.log(
        "Benign control set:",
        benignT.slice(Math.floor(benignT.length / 2), benignT.length).length
      );
      console.log(
        "SQLi control set:",
        sqliT.slice(Math.floor(sqliT.length / 2), sqliT.length).length
      );
      result = { success_b: 0, error_b: 0, success_i: 0, error_i: 0 };

      for (let record of sqliT.slice(
        Math.floor(sqliT.length / 2),
        sqliT.length
      )) {
        let r = network.activate(record)[0];
        if (r < 0.5) result.error_i++;
        else result.success_i++;
      }
      for (let record of benignT.slice(
        Math.floor(benignT.length / 2),
        benignT.length
      )) {
        let r = network.activate(record)[0];
        if (r >= 0.5) result.error_b++;
        else result.success_b++;
      }
      console.log("Test result:", result);
      // controlTest()
    });
});

var mySet = [
  ...benignSet.map(content => {
    return {
      input: content,
      output: [0]
    };
  }),
  ...fuzzingSet.map(content => {
    return {
```

```
      input: content,
      output: [1]
    };
  })
];
```

classifier.js

```
const fs = require("fs");
const path = require("path");

const classes = [
  "'",
  "CREATE",
  "DELETE FROM",
  "DROP TABLE",
  "INSERT INTO",
  "UNION",
  "AND",
  "OR",
  "--",
  " ",
  "FROM",
  "EXEC"
].map(el => el.toLowerCase());

function binaryView(str) {
  return classes.map(pattern => (str.toLowerCase().includes(pattern) ? 1 : 0));
}

function getDataSet(filename) {
  var data = fs.readFileSync(path.join(__dirname, filename), {
    encoding: "utf-8"
  });
  data = data.split("\n");
  for (let i = 0; i < data.length; i++) {
    data[i] = binaryView(data[i]);
  }
  return data;
}

module.exports = { getDataSet, binaryView };
```