

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ
МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ
SOFTWARE IMPLEMENTATION OF THE NETWORK ATTACK
DETECTION SYSTEM BASED ON FUZZY INFERENCE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-111сп

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДОРОШЕНКО О.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЕОЩЕНКО С.Д.

(ПРИЗВИЩЕ та ініціали)

Рецензент СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. СУББОТІН
« » 2024 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДОРОШЕНКА Олександра Вадимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація системи виявлення мережесих атак на основі нечіткого виведення. Software Implementation of the Network Attack Detection System based on Fuzzy Inference

керівник проєкту (роботи) д-р філософ., Леощенко Сергій Дмитрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «24» квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 07 червня 2024 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., ст.викл.		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання «24» квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

(підпис) Олександр ДОРОШЕНКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Сергій ЛЕОЩЕНКО
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
96 с., 5 табл., 25 рис., 4 дод., 22 джерела.

ЗАХИСТ, МЕРЕЖА, ФІЛЬТР, НЕЧІТКА ЛОГІКА, РОЗРОБКА, МОВА
ПРОГРАМУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес програмної реалізації системи виявлення мережових атак на основі нечіткого виведення.

Предмет дослідження – програмні засоби для програмної реалізації системи виявлення мережових атак на основі нечіткого виведення.

Метою роботи є програмна реалізація системи виявлення мережових атак на основі нечіткого виведення.

Матеріали, методи та технічні засоби: мова програмування Java та засоби моделювання UML.

Результати. Розроблено систему виявлення мережових атак на основі нечіткого виведення.

Практична цінність роботи полягає у програмній реалізації системи виявлення мережових атак на основі нечіткого виведення.

Висновки. Реалізовано систему виявлення мережових атак на основі нечіткого виведення.

Галузь використання – користувачі Інтернету.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 96 pages, 5 tables, 25 figures, 4 appendixes, 22 sources.

SECURITY, NETWORK, FILTER, FUZZY LOGIC, DEVELOPMENT, PROGRAMMING LANGUAGE, SOFTWARE.

The object of research is the process of software implementation of a network attack detection system based on fuzzy output.

The subject of the research is software tools for software implementation of a network attack detection system based on fuzzy output.

The aim of this paper is to provide a software implementation of a network attack detection system based on fuzzy output.

Materials, methods, and technical tools: the Java programming language and UML modeling tools.

Results. A network attack detection system based on fuzzy output has been developed.

The practical value of the work lies in the software implementation of a network attack detection system based on fuzzy output.

Conclusions. Implemented a network attack detection system based on fuzzy output.

Scope of use – Internet users.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Аналіз предметної області	15
1.1 Кіберзахист від мережеских атак	15
1.2 Використання спеціалізованого ПЗ для захисту від мережеских атак	19
1.3 Виявлення мережеских вторгнень	23
1.3.1 Класифікація систем виявлення вторгнень	24
1.3.2 Методи виявлення вторгнень	25
1.4 Порівняння існуючих аналогів	26
1.5 Постановка завдання	34
1.5.1 Функціональність системи	35
1.5.2 Вимоги до інтерфейсу	36
1.5.3 Вимоги до безпеки	36
1.5.4 Не функціональні вимоги до продукту	37
1.6 Висновки за розділом 1	37
2 Матеріали і методи	38
2.1 Вибір мови програмування	38
2.2 Вибір середовища розробки	45
2.3 Використання нечіткої логіки	50
2.4 Структурна схема розробки	53
3 Основні рішення щодо реалізації компонентів системи	55
3.1 Проектування моделей системи	55
3.2 Класифікація навчальних даних	59
3.3 Стратегія формування нечітких правил	60
4 Експлуатація, тестування та експериментальне дослідження програми	61
4.1 Призначення й умови застосування програми	61
4.2 Вимоги до інтерфейсу	61
4.3 Умови виконання програми	62

4.4 Методика та результати тестування	62
4.5 Результати роботи алгоритму.....	64
4.6 Висновки за отриманими результатами	67
Висновки	68
Перелік джерел посилань	69
Додаток А Технічне завдання	71
Додаток Б Опис програми	76
Додаток В Текст програми	81
Додаток Д Слайди презентації	90

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

JVM – Java Virtual Machine;

UML – Unified Modeling Language;

АПКО – аналіз поведінки користувачів та організацій;

МН – машинне навчання;

ОС – операційна система;

ПЗ – програмне забезпечення;

СВВ – системи виявлення вторгнень;

ШІ – штучний інтелект.

ВСТУП

Штучний інтелект (ШІ) та машинне навчання (МН) спонукають революцію в інформаційній безпеці. Їх застосування варіюється від виявлення загроз і реагування на них до автоматизації завдань безпеки [1]–[3].

ШІ – це моделювання людського інтелекту в машинах, які запрограмовані мислити і навчатися подібно людям. МН – різновид штучного інтелекту, передбачає використання алгоритмів і статистичних моделей, що дозволяють системам покращувати свою продуктивність при виконанні завдань з плином часу і набуттям досвіду [1]–[3].

У контексті кібербезпеки ШІ та МН використовуються для виявлення, запобігання загрозам та реагування на них ефективніше та точніше, ніж традиційні методи. Вони дозволяють аналізувати величезні обсяги даних для виявлення закономірностей і аномалій, які можуть вказувати на проблеми з безпекою [1]–[3].

ШІ та МН можуть аналізувати дані з різних джерел для виявлення потенційних загроз. Це включає мережевий трафік, поведінку користувачів та системні журнали. Вивчаючи, що є нормальною поведінкою, ці системи можуть виявляти аномалії, які можуть вказувати на шкідливу активність [1]–[3].

Виявлення аномалій: методи МН можуть визначати базовий рівень нормальної поведінки та відзначати відхилення від цього базового рівня. Це особливо корисно при виявленні невідомих загроз, таких як атаки нульового дня [1]–[3].

Виявлення на основі підписів: ШІ може вдосконалити традиційні методи, засновані на підписах, шляхом швидкого оновлення та створення нових підписів для відомих загроз [1]–[3].

Системи, керовані ШІ, можуть допомогти в процесі реагування на інциденти, швидко аналізуючи дані для визначення масштабів і наслідків

інциденту безпеки. Вони також можуть рекомендувати або автоматизувати дії щодо стримування та усунення загрози [1]–[3].

Автоматизоване реагування: системи ІШ можуть бути налаштовані на виконання заздалегідь визначених дій при виконанні певних умов, таких як ізоляція скомпрометованого пристрою або блокування шкідливих IP-адрес.

Криміналістика та аналіз: моделі МН можуть аналізувати величезні обсяги даних журналу, щоб точно визначити першопричину інциденту, що необхідно для ефективного усунення несправностей та запобігання їх у майбутньому [1]–[3].

ІШ може аналізувати моделі поведінки користувачів для виявлення внутрішніх загроз і скомпрометованих облікових записів. Розуміючи типову поведінку користувача, системи штучного інтелекту можуть відзначати незвичайні дії, які можуть вказувати на проблему з безпекою [1]–[3].

Аналіз поведінки користувачів та організацій (АПКО) – включає моніторинг користувачів та пристроїв для виявлення відхилень від нормальної поведінки. Системи АПКО можуть виявляти скомпрометовані облікові дані, внутрішні загрози та шкідливе програмне забезпечення, аналізуючи взаємодію користувачів із системами [1]–[3].

ІШ та МН можуть бути використані для виявлення спроб фішингу в електронних листах, вебсайтах та повідомленнях шляхом аналізу їх вмісту, контексту та інформації про відправника.

Аналіз вмісту – ІШ може аналізувати вміст електронної пошти, включаючи текст, посилання та вкладення, щоб виявити ознаки фішингу. Перевірка відправника: моделі МН дозволяють перевірити справжність відправника, порівнявши його з відомими профілями та шаблонами відправників [1]–[3].

ІШ може допомогти виявити вразливості в системах шляхом аналізу програмного коду, конфігурацій і мережевих налаштувань. Цей проактивний підхід дозволяє організаціям вирішувати слабкі місця, перш ніж вони можуть бути використані [1]–[3].

Статичний та динамічний аналіз: інструменти штучного інтелекту можуть виконувати як статичний аналіз коду, так і динамічне тестування програм для пошуку вразливостей [1]–[3].

Управління виправленнями: МН може визначати пріоритетність вразливостей залежно від ступеня ризику, який вони представляють, допомагаючи організаціям зосередитися на найважливіших виправленнях.

ШІ та МН широко використовуються в таких галузях, як банківська справа та електронна комерція, для виявлення та запобігання шахрайським діям.

Моніторинг транзакцій: моделі МН можуть аналізувати схеми транзакцій для виявлення потенційно шахрайських дій.

Аналіз у режимі реального часу: системи штучного інтелекту можуть здійснювати моніторинг у режимі реального часу та виявляти підозрілі транзакції для подальшого розслідування [1]–[3].

Навчання з вчителем передбачає відпрацювання моделі МН на позначених даних. У сфері кібербезпеки це може бути використано для класифікації електронних листів як спаму або не спаму або для ідентифікації відомих типів шкідливих програм.

Дерева рішень: використовуються для задач класифікації та регресії, а також можуть застосовуватися для виявлення вторгнень шляхом вивчення історичних даних про атаки.

Методи опорних векторів використовуються для класифікаційних завдань і можуть застосовуватися для виявлення аномалій у мережевому трафіку [1]–[3].

Методи навчання без вчителя працюють з немаркованими даними для пошуку прихованих закономірностей. Це корисно для виявлення аномалій та виявлення нових типів загроз.

Кластеризація: такі методи, як кластеризація за методом k-середніх, дозволяють групувати подібні точки даних разом, допомагаючи виявляти незвичайні закономірності, які можуть вказувати на загрозу безпеці.

Аналіз головних компонент – він використовується для зменшення розмірності і може допомогти у візуалізації та розумінні даних, що полегшує виявлення аномалій [1]–[3].

Навчання з підкріпленням передбачає навчання моделей методом спроб і помилок, щоб максимізувати функцію винагороди. Це може бути застосовано в таких сценаріях, як автоматичне виявлення вторгнень і реагування на них. Q-навчання – цей метод може бути використаний для розробки політик автоматичного реагування на кіберзагрози шляхом вивчення найкращих дій в різних станах середовища [1]–[3].

Глибоке навчання – підрозділ МН, що включає багат шарові нейронні мережі. Воно особливо ефективно в таких завданнях, як розпізнавання зображень і мови, які можуть бути застосовані в сфері кібербезпеки для розширеного виявлення загроз. Згорткові нейронні мережі можна використовувати для аналізу структури мережевого трафіку та виявлення шкідливих дій. Рекурентні нейронні мережі корисні для аналізу часових рядів даних, їх можна застосовувати для аналізу послідовності подій у журналах безпеки для виявлення аномалій [1]–[3].

Системи ШІ можуть обробляти величезні обсяги даних і масштабуватися відповідно до потреб великих організацій. Це має вирішальне значення в епоху, коли обсяг даних і кількість підключених пристроїв постійно зростають [1]–[3].

ШІ може аналізувати дані та реагувати на загрози в режимі реального часу, що значно скорочує час, необхідний для виявлення та усунення інцидентів безпеки [1]–[3].

Моделі МН вдосконалюються з часом, оскільки вони отримують доступ до більшої кількості даних. Це забезпечує більш точне виявлення загроз і меншу кількість помилкових спрацьовувань, дозволяючи фахівцям з безпеки зосередитися на реальних загрозах [1]–[3].

ІІІ забезпечує проактивний підхід до кібербезпеки, прогножуючи потенційні загрози та виявляючи вразливості, перш ніж вони можуть бути використані.

Проте, для ефективності моделей ІІІ та МН потрібна велика кількість високоякісних даних. Неповні, застарілі або упереджені дані можуть призвести до неточних моделей та неправильного прийняття рішень [1]–[3].

Кібератаки можуть використовувати змагальні методи для обману моделей ІІІ. Це включає в себе використання в моделях зловмисно створених вхідних даних, призначених для неправильної класифікації або ухилення від виявлення [1]–[3].

Багато моделей ІІІ, особливо моделі глибокого навчання, часто розглядаються як чорні ящики через їх складність. Відсутність можливості інтерпретації може ускладнити розуміння того, як приймаються рішення, і довіру до результатів.

Інтеграція рішень ІІІ з існуючою інфраструктурою кібербезпеки може бути складною і вимагати значних ресурсів. Забезпечення сумісності та безперебійної роботи є серйозною проблемою [1]–[3].

Розробка та обслуговування систем ІІІ та МН можуть бути дорогими та вимагати спеціальних знань та ресурсів.

Тривають дослідження з розробки методів захисту від ворожих атак на моделі ІІІ. Це включає методи підвищення надійності моделей та визначення того, коли вони стають мішенню [1]–[3].

Прикладаються зусилля з розробки систем ІІІ, які були б більш прозорими і піддавалися б інтерпретації. Зрозумілий ІІІ може допомогти зміцнити довіру і полегшити розуміння фахівцями в області безпеки інформації, отриманої за допомогою ІІІ, і вжиття заходів відповідно до неї.

ІІІ може підвищити ефективність аналізу загроз, поєднуючи та аналізуючи дані з різних джерел, щоб отримати повне уявлення про загрози [1]–[3].

ШІ може сприяти співпраці між організаціями, обмінюючись інформацією про загрози та аналітичними даними, що призводить до більш скоординованого та ефективного захисту від кіберзагроз [1]–[3].

Вкрай важливо забезпечити, щоб системи ШІ розроблялися і використовувалися з дотриманням етичних норм. Це включає вирішення таких проблем, як упередженість, справедливість та конфіденційність у рішеннях з кібербезпеки, заснованих на ШІ [1]–[3].

ШІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ШІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація системи виявлення мережевих атак на основі нечіткого виведення.

Для виконання такої задачі необхідно виконати наступні завдання:

- дослідити проблемну область кіберзахисту від мережевих атак;
- дослідити існуючі рішення для проблемної області кіберзахисту від мережевих атак;
- спираючись на обґрунтований вибір технічних засобів виконати програмну реалізацію системи виявлення мережевих атак на основі нечіткого виведення;
- провести тестування розробки;
- сформулювати висновки на основі виконаної роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Засоби виявлення вторгнень все частіше стають ключовим елементом систем захисту для виявлення аномальної активності в інформаційній системі. В основному, традиційні системи виявлення вторгнень базуються на знаннях експертів, зокрема на їх знаннях, щодо захищається системи. Для того щоб зменшувати цю залежність, застосовуються різні методи аналізу даних, а так само методи машинного навчання [1]–[3].

Одним із методів виявлення вторгнень є використання системи, заснованої на нечіткій логіці, яка дозволяє ефективно виявляти факти проникнення.

Дана дипломна робота спрямована на розробку системи на основі нечіткої логіки, яка дозволить виявляти вторгнення за рахунок використання правил з нечітким висновком. Експерименти та оцінки системи виявлення пропонується проводити на основі наборів даних KDD Cup 99 [1]–[3].

Метою роботи є виявлення мережевих атак із застосуванням нечіткого виведення.

Завдання:

- визначення значущих атрибутів;
- формування бази нечітких правил;
- розробка прототипу програмного комплексу;
- тестування прототипу системи.

1.1 Кіберзахист від мережевих атак

Кіберзахист від мережевих атак включає в себе ряд стратегій, технологій і процесів, призначених для захисту мереж, пристроїв і даних від несанкціонованого доступу, збоїв в роботі або знищення. Оскільки кіберзагрози продовжують ускладнюватися і частішати, вирішальне значення набувають надійні механізми мережевого захисту [4]–[6].

Мережеві атаки – це шкідливі дії, спрямовані на мережеву інфраструктуру з метою порушення роботи служб, крадіжки інформації або отримання несанкціонованого доступу. Ці атаки можуть приймати різні форми, включаючи [4]–[6].

DoS-атаки – перевантажують мережу надлишковим трафіком, роблячи її недоступною. DDoS-атаки використовують кілька скомпрометованих систем для переповнення цільової мережі трафіком, посилюючи вплив атаки [4]–[6].

Атаки – людина посередині. Зловмисники перехоплюють і змінюють комунікації між двома сторонами без їх відома, часто з метою крадіжки конфіденційної інформації або впровадження шкідливого контенту [4]–[6].

Фішинг та соціальна інженерія. Методи, що використовуються для обману людей з метою розголошення конфіденційної інформації або завантаження шкідливого програмного забезпечення (ПЗ).

Зараження шкідливим ПЗ. Включає віруси, черв'яків, троянські програми, програми-вимагачі та шпигунські програми, призначені для пошкодження, дезорганізації або отримання несанкціонованого доступу до мереж [4]–[6].

Впровадження SQL. Зловмисники використовують уразливості у вебзастосунках для виконання шкідливих SQL-запитів, що дозволяє їм отримувати доступ до баз даних або маніпулювати ними.

Експлойти нульового дня. Атаки, що використовують невідомі або не усунені уразливості в програмному або апаратному забезпеченні.

Ефективна кіберзахист від мережевих атак вимагає багаторівневого підходу, що поєднує різні технології, процеси і політики. Ключовими компонентами є:

- політики мережевої безпеки – розробка комплексних політик для управління мережевими ресурсами та їх захисту, включаючи прийнятне використання, контроль доступу та протоколи реагування на інциденти [4]–[6];

- брандмауер – апаратні або програмні рішення, які відстежують і керують вхідним і вихідним мережевим трафіком на основі заздалегідь визначених правил безпеки;
- системи виявлення та запобігання вторгнень – системи, які відстежують мережевий трафік на предмет підозрілих дій та вживають заходів для запобігання потенційним загрозам [4]–[6];
- захист кінцевих точок – рішення безпеки, розгорнуті на пристроях, підключених до мережі, забезпечують захист від шкідливих програм та інших загроз;
- шифрування – захист переданих і збережених даних шляхом перетворення їх в безпечний формат, доступний для читання тільки авторизованими сторонами [4]–[6];
- сегментація мережі – поділ мережі на більш дрібні сегменти для обмеження поширення атак і локалізації потенційних порушень;
- управління виправленнями – регулярне оновлення програмного та апаратного забезпечення для усунення вразливостей та підвищення безпеки;
- управління інформацією про безпеку та події – інструменти, які збирають та аналізують дані з різних джерел, щоб отримати повне уявлення про стан безпеки мережі;
- план реагування на інциденти – чітко визначена стратегія виявлення інцидентів безпеки, реагування на них і відновлення після них.

Технології та методи, що використовуються в кіберзахисті. Брандмауери – це перша лінія захисту в області мережевої безпеки. Вони фільтрують трафік на основі зумовлених правил, блокуючи несанкціонований доступ і забезпечуючи можливість законного обміну даними [4]–[6].

Брандмауери з фільтрацією пакетів – перевіряють окремі пакети даних і блокують їх, якщо вони не відповідають заздалегідь визначеним правилам безпеки.

Брандмауери, що відстежують стан – відстежують стан активних з'єднань та приймають рішення на основі контексту трафіку [4]–[6].

Брандмауери нового покоління – поєднують традиційні функції брандмауера з додатковими функціями, такими як інформованість про додатки, запобігання вторгнень і аналіз загроз [4]–[6].

Системи виявлення та запобігання вторгнень – мають вирішальне значення для виявлення та припинення шкідливих дій у мережі:

- виявлення на основі підписів – порівнює мережевий трафік з базою даних, що містить відомі шаблони атак або підписи;
- виявлення на основі аномалій – встановлює базовий рівень нормальної поведінки мережі та виявляє відхилення, які можуть вказувати на загрозу;
- можливості запобігання: крім виявлення, деякі системи можуть автоматично блокувати шкідливий трафік або поміщати скомпрометовані пристрої в карантин [4]–[6].

Платформи Endpoint Protection (EPP) і Endpoint Detection and Response (EDR). Ці рішення захищають окремі пристрої від шкідливих програм та інших загроз. EPP – надає функції антивірусу, захисту від шкідливих програм та брандмауера для запобігання зараженню. EDR — пропонує розширені можливості виявлення загроз, розслідування та реагування, включаючи моніторинг та аналіз дій кінцевих точок у режимі реального часу [4]–[6].

Шифрування. Шифрування гарантує, що дані не будуть прочитані несанкціонованими користувачами. Передача даних: протоколи Secure Sockets Layer (SSL)/Transport Layer Security (TLS) шифрують дані при їх передачі по мережі [4]–[6].

Зберігання даних: такі методи, як повне шифрування диска та шифрування на рівні файлів, захищають збережені дані.

Сегментація мережі. Сегментація мережі передбачає поділ мережі на ізольовані сегменти для запобігання зломів і обмеження пересування злоумисників.

Віртуальні локальні обчислювальні мережі – розділення пристроїв на різні логічні мережі в межах однієї фізичної мережі [4]–[6].

Мікросегментація – використовує програмно визначені мережі для створення детальної політики безпеки на рівні робочого навантаження.

Системи інформації про безпеку та управління подіями об'єднують та аналізують дані про безпеку з різних джерел для виявлення та реагування на загрози [4]–[6].

Збір і аналіз логів: збирайте логи з брандмауерів, IP-адрес, серверів та інших пристроїв для виявлення закономірностей, що вказують на атаку. Кореляція та попередження: зіставляйте події з різних джерел, щоб виявити складні схеми атак та генерувати попередження.

ШІ і МН покращують захист мережі, автоматизуючи виявлення загроз і реагування на них.

Виявлення аномалій: моделі ML виявляють відхилення від нормальної поведінки, які можуть вказувати на потенційну загрозу.

Аналіз загроз: системи ШІ аналізують дані про загрози з різних джерел, щоб отримати уявлення про виникаючі загрози [4]–[6].

1.2 Використання спеціалізованого ПЗ для захисту від мережевих атак

Існує безліч застосунків і програм для кіберзахисту від мережевих атак. Ці інструменти варіюються від комплексних наборів засобів захисту до спеціалізованого ПЗ, орієнтованого на конкретні аспекти мережевої безпеки. Нижче наведено список деяких відомих додатків і програм для різних категорій кіберзахисту.

Перша група застосунків – брандмауери. Cisco ASA – це популярне рішення брандмауера, яке забезпечує надійний захист, підтримку VPN та розширений захист від загроз.

Palo Alto Networks – брандмауер нового покоління, що пропонує розширені функції безпеки, включаючи обізнаність про застосунки, ідентифікацію користувачів та можливості запобігання загрозам [4]–[6].

Fortinet FortiGate – забезпечує високопродуктивний захист і пропонує інтегровану аналітику загроз для захисту від широкого спектру атак.

Другим рівнем виділяють – системи виявлення та запобігання вторгнень.

Snort – система виявлення та запобігання вторгнень з відкритим кодом, здатна аналізувати трафік у режимі реального часу та реєструвати пакети.

Suricata – ще один механізм виявлення мережових загроз з відкритим вихідним кодом з можливостями виявлення вторгнень, запобігання та моніторингу безпеки [4]–[6].

Cisco Firepower – комплексна система захисту від загроз, яка інтегрується з екосистемою безпеки Cisco для розширеного виявлення загроз і реагування на них.

Далі слід розглянути платформи захисту кінцевих точок та виявлення та реагування на кінцеві точки.

Symantec Endpoint Protection – добре відоме рішення для забезпечення безпеки кінцевих точок, що надає антивірус, захист від шкідливих програм та інші важливі функції безпеки.

CrowdStrike Falcon – удосконалена платформа EDR, що забезпечує аналіз загроз, моніторинг в режимі реального часу і автоматизоване реагування.

Carbon Black (VMware) – надає комплексні можливості EDR, включаючи безперервний моніторинг кінцевих точок, пошук загроз і реагування на інциденти [4]–[6].

Управління інформацією про безпеку та події. Splunk Enterprise Security – потужна платформа, яка забезпечує розширене виявлення загроз, реагування на інциденти і звітність про відповідність вимогам.

IBM QRadar – інтегроване рішення для управління і журналами, яке забезпечує виявлення загроз в режимі реального часу і управління відповідністю вимогам.

ArcSight (Micro Focus) – надійне рішення, що пропонує широкі можливості моніторингу, аналізу та реагування на інциденти [4]–[6].

Управління вразливістю. Tenable.io – комплексна платформа управління вразливістю, яка допомагає виявляти, визначати пріоритети та усувати вразливості.

QualysGuard – забезпечує безперервний моніторинг і оцінку мережевих вразливостей, а також управління відповідністю вимогам.

Nessus – широко використовуваний сканер вразливостей, який допомагає виявляти потенційні вразливості в мережевих пристроях, операційних системах і додатках [4]–[6].

Моніторинг та аналіз мережі. Wireshark – аналізатор мережевих протоколів з відкритим кодом, який дозволяє користувачам фіксувати та аналізувати мережевий трафік у режимі реального часу.

SolarWinds Network Performance Monitor – комплексний інструмент моніторингу мережі, який забезпечує моніторинг продуктивності мережі, структури трафіку та потенційних проблем.

Nagios – система моніторингу мережі з відкритим кодом, яка відстежує мережеву інфраструктуру на предмет проблем та попереджає адміністраторів [4]–[6].

Захист від DDoS-атак. Cloudflare – популярна служба веббезпеки, яка пропонує захист від DDoS-атак, послуги мережі доставки вмісту та брандмауер вебзастосунків.

Akamai Kona Site Defender – забезпечує надійний захист від DDoS-атак поряд з WAF та іншими службами безпеки для захисту вебзастосунків.

Imperva Incapsula – пропонує послуги з запобігання DDoS-атак, а також набір служб безпеки застосунків, WAF і CDN [4]–[6].

Шифрування та захист даних. VeraCrypt – програмне забезпечення для шифрування дисків з відкритим вихідним кодом, що забезпечує надійне шифрування даних в стані спокою.

BitLocker (Microsoft) – функція повного шифрування диска, доступна в Windows, що забезпечує надійне шифрування і захист даних.

OpenSSL – широко використовуваний інструментарій для реалізації шифрування SSL/TLS для захисту переданих даних.

Управління паролями. LastPass – популярний менеджер паролів, який надійно зберігає та керує паролями, забезпечуючи надійне шифрування та багатофакторну автентифікацію [4]–[6].

Dashlane – забезпечує управління паролями, безпечне зберігання і моніторинг темної павутини для захисту облікових даних користувачів.

1Password – забезпечує безпечне управління паролями за допомогою таких функцій, як генерація, зберігання та обмін паролями.

Розширений захист від загроз. Microsoft Defender ATP – інтегрована платформа для захисту кінцевих точок, виявлення загроз і реагування на інциденти безпеки [4]–[6].

FireEye Endpoint Security – пропонує розширені можливості виявлення загроз і реагування на них, включаючи захист від шкідливих програм і криміналістичний аналіз [4]–[6].

Хмарна безпека. AWS Security Hub – надає повний огляд попереджень про безпеку та стан відповідності в облікових записах AWS з інтегрованим виявленням загроз та реагуванням на них [4]–[6].

Центр безпеки Microsoft Azure – забезпечує єдине управління безпекою і розширений захист від загроз в гібридних хмарних середовищах [4]–[6].

Командний центр хмарної безпеки Google – платформа безпеки та управління ризиками, яка допомагає виявляти та вирішувати загрози в хмарних середовищах Google [4]–[6].

Управління ідентифікацією та доступом. Okta – провідне рішення для управління ідентифікацією та доступом, що забезпечує безпечне управління ідентифікаційними даними та можливості єдиного входу [4]–[6].

Microsoft Azure Active Directory – служба управління ідентифікацією, що пропонує єдиний вхід, багатофакторну аутентифікацію і умовний доступ для хмарних і локальних додатків [4]–[6].

Перевірка справжності Ping – надає передові рішення для захисту особистих даних, включаючи єдиний вхід, багатофакторну аутентифікацію і адаптивну аутентифікацію [4]–[6].

Область кіберзахисту обширна і постійно розвивається. Перераховані вище інструменти та програми є одними з найкращих і найбільш широко використовуваних рішень у галузі, кожне з яких пропонує унікальні функції та можливості для захисту від мережових атак. Впровадження комбінації цих інструментів, адаптованих до конкретних потреб організації та профілю ризиків, має важливе значення для побудови надійної стратегії кіберзахисту. Регулярні оновлення, постійний моніторинг та навчання співробітників також є важливими компонентами підтримки безпечного мережевого середовища [4]–[6].

1.3 Виявлення мережових вторгнень

Частота проникнення в комп'ютерні системи збільшуються через комерціалізації Інтернету і локальних мереж. Комп'ютерні системи, виявляються все більш і більш уразливі до атак, через розширення можливостей мережевого підключення. Основна мета мережових атак полягає в тому, щоб порушити стандартні процеси безпеки і виконати дії понад повноваження атакуючого. Ці дії можуть включати в себе читання секретних або конфіденційних даних або просто призводити до порушення цілісності системи або призначених для користувача файлів [1], [3], [6].

Адміністратор безпеки системи може виявляти шкідливу поведінку в системі, за рахунок використання спеціальних інструментів, які дозволяють постійно контролювати стан системи і повідомляти про «нестандартні активності». Системи виявлення вторгнень відіграють важливу роль у

забезпеченні захисту мережі. Система виявлення вторгнень (IDS) контролює мережеві пристрої та здійснює пошук аномальних або шкідливих дій у потоці даних. У хорошій системі виявлення вторгнень повинна існувати можливість встановити відмінності між стандартною та аномальною поведінкою користувачів. Сукупність подій, станів, переданого контенту, а так само поведінки користувачів буде розцінюватися, як аномальне на основі зумовлених критеріїв [1], [3], [6].

1.3.1 Класифікація систем виявлення вторгнень

Системи виявлення вторгнень (СВВ) можуть бути класифіковані різними способами. Ця класифікація може ґрунтуватися на джерелі даних, поведінці системи, архітектурі, на тому, як система захищена, а також як виявляються проникнення (рис. 1.1).

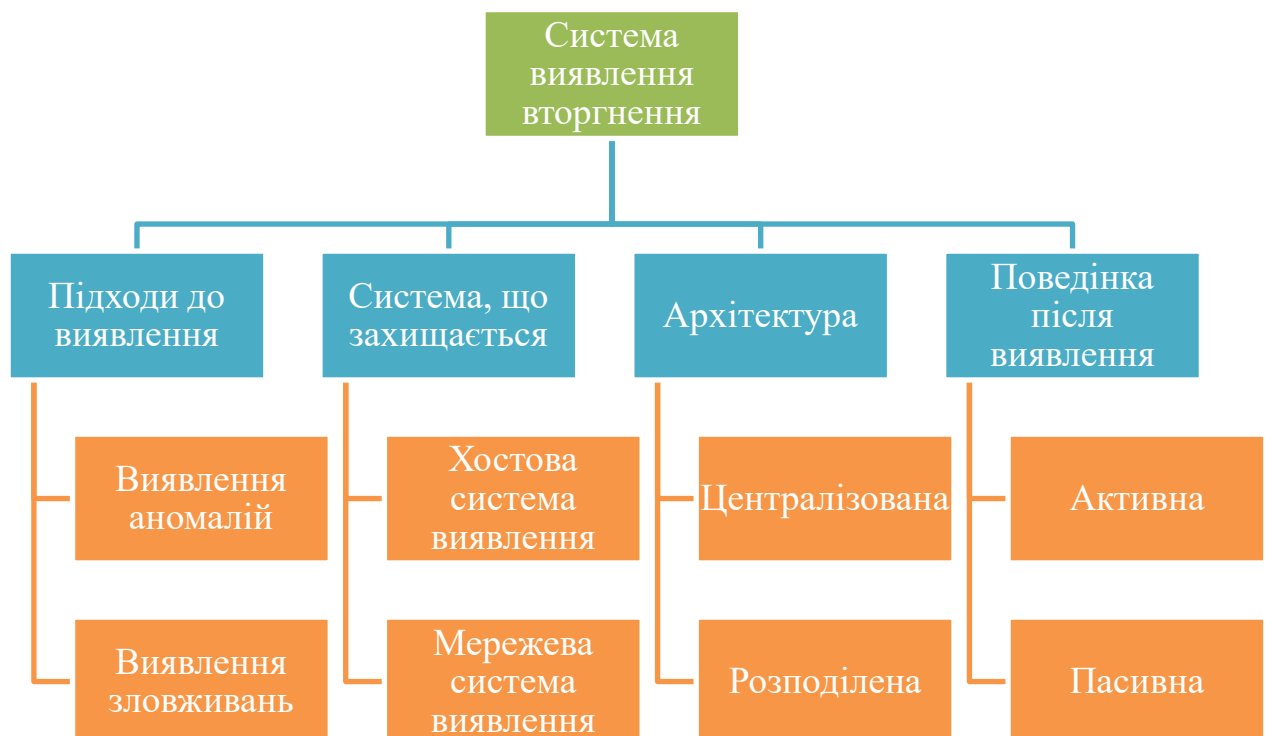


Рисунок 1.1 – Класифікація систем виявлення вторгнень

Відповідно до стратегії виявлення. СВВ, головним чином, можуть бути класифіковані, як системи виявлення аномалій і зловживань. Виявлення вторгнень на основі аномалій, також відоме як поведінковий аналіз. Метод базується на виявленні трафіку, який сильно відрізняється від нормальної поведінки системи. Застосування даного підходу дозволяє виявляти раніше невідомі атаки. Другий підходи-це виявлення зловживань, також відоме як аналіз підпису. Основою методу є пошук атак за заздалегідь складеними шаблонами, таким чином, система в змозі виявити тільки відомі атаки. Кожен з цих методів має свої переваги н недоліки [1], [3], [6].

Тип СВВ. заснований на способі захисту, може бути класифікований відповідно до джерела даних, від якого отримана інформація. Хостова СОВ аналізує дані, отримані від єдиного вузла нлн комп'ютерної системи. З іншого боку, мережеві системи виявлення вторгнень тримають під наглядом кожен вузол мережі. Однак, системи, доступні на ринку, в більшості своїй є гібридом NIDS і HIDS.

Система виявлення вторгнення може бути як розподіленою, так і централізованою. У розподілених системах в мережі присутні кілька СВВ. пов'язаних один з одним, нлн підключених до централізованого сервера. Також слід розуміти, що IDS може бути автономною системою [1], [3], [6].

Відповідно до поведінки, СВВ можна розділити на активні та пасивні. Активні системи дозволяють не тільки виявляти, але і негайно реагувати на можливі атаки. З іншого боку, пасивні СВВ лише виявляють проникнення. Отже, активні системи також відомі, як системи запобігання вторгнень (СЗВ).

1.3.2Методи виявлення вторгнень

Виявлення вторгнень є значущою областю досліджень, так як неможливо створити системи без вразливостей. Однією з основних складнощів у виявленні вторгнень є необхідність виділення прихованих атак ПЗ величезного обсягу трафіку. Існують такі методи МН, наприклад, нейронні

мережі, метод опорних векторів, генетичні алгоритми, нечітка логіка та методи аналізу даних, які використовуються для виявлення аномальних активностей у великому обсязі складних і динамічних наборів даних.

Генерація правил є дуже важливим аспектом для IDS, оскільки системі необхідно відрізняти стандартне поведінку від нестандартного. Для виявлення раніше зустрічалися вторгнень проведено безліч різних досліджень. Метод аналізу даних для виявлення взаємозв'язків між подіями та виявлення прихованих моделей даних, які в іншому випадку пройшли б непоміченими, відомий як Data Mining [1], [3], [6].

Відповідно до стратегії виявлення, системи Data Mining можна розділити на дві основні категорії. Це виявлення зловживань, де для ідентифікації проникнень, використовуються шаблони відомих вторгнень або вразливостей системи, а також виявлення аномалій, яке дозволяє виявити відхилення від стандартної поведінки [1], [3], [6].

1.4 Порівняння існуючих аналогів

Для виявлення DDoS-атак, Probes та нормального трафіку наступні три програми є одними з найпопулярніших і широко використовуваних у галузі. Кожна з цих програм пропонує комплексні функції для моніторингу мережевого трафіку, виявлення аномалій і захисту від різних типів мережевих загроз [7], [8].

Wireshark – це широко використовуваний аналізатор мережевих протоколів з відкритим кодом. Це дозволяє користувачам відстежувати та інтерактивно переглядати трафік, що передається через комп'ютерну мережу. Wireshark забезпечує глибоку перевірку сотень протоколів, а також можливості оперативного відстеження та автономного аналізу [7], [8].

Wireshark фіксує мережеві пакети і відображає їх якомога детальніше, забезпечуючи всебічне уявлення про мережеву активність. Користувачі

можуть аналізувати поточні мережеві дані або відкривати раніше збережені файли відстеження [7], [8].

Розширена підтримка протоколів – підтримує широкий спектр мережевих протоколів, що робить його універсальним для різних мережевих середовищ.

Інтерфейс програмного забезпечення Wireshark наведено на рис. 1.2.

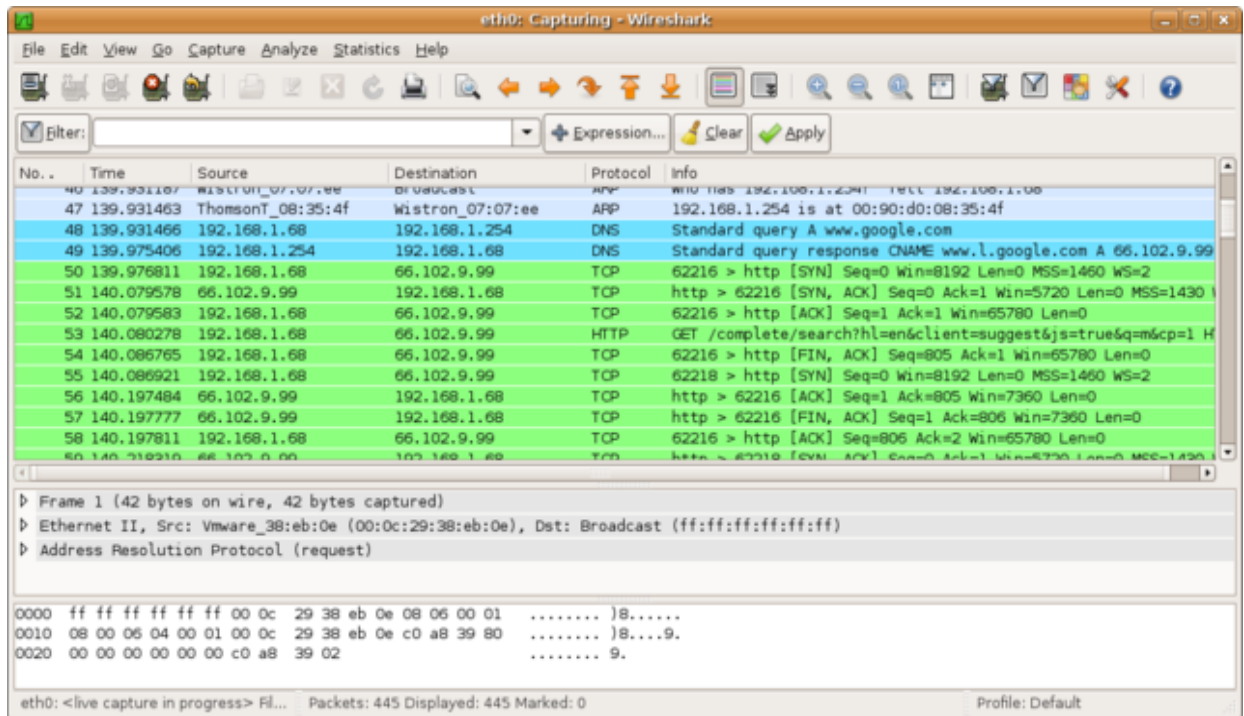


Рисунок 1.2 – Wireshark [8]

Настроювані звіти – користувачі можуть створювати власні звіти та фільтри, щоб зосередитись на конкретних мережевих проблемах [7], [8].

Сумісність між платформами – доступний у кількох операційних системах, включаючи Windows, macOS та Linux.

Варіанти використання:

– виявлення DDoS-атак – аналізуючи структуру трафіку та виявляючи аномалії, Wireshark може допомогти виявити наявність DDoS-атаки;

– ідентифікація мережевих зондів – мережеві зонди, які сканують уразливості, можуть бути виявлені шляхом виявлення незвичайних потоків трафіку і дій по скануванню;

– моніторинг звичайного трафіку – Wireshark допомагає адміністраторам мережі зрозуміти типові схеми трафіку, що полегшує виявлення аномалій.

Snort (рис. 1.3) – це система виявлення та запобігання вторгненням з відкритим кодом (IDPS), здатна виконувати аналіз трафіку в режимі реального часу та реєстрацію пакетів. Вона може виявляти широкий спектр атак і зондувань, що робить її потужним інструментом мережевої безпеки [9], [10].

The screenshot displays the Snort Alerts management interface. At the top, there's a breadcrumb trail: Services / Snort / Alerts. Below this is a navigation bar with tabs: Snort Interfaces, Global Settings, Updates, Alerts (active), Blocked, Pass Lists, Suppress, IP Lists, SID Mgmt, Log Mgmt, and Sync. A link 'Clear all interface log files' is visible. The 'Alert Log View Settings' section includes a dropdown for 'Interface to Inspect' (set to WAN), an 'Auto-refresh view' checkbox, a text input for 'Alert lines to display' (set to 1000), and a 'Save' button. Below this are 'Alert Log Actions' with 'Download' and 'Clear' buttons. The 'Alert Log View Filter' section has a plus icon. The main area is titled 'Last 1000 Alert Log Entries' and contains a table with the following data:

Date	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	SID	Description
2017-07-23 20:49:52	1	UDP	A Network Trojan was Detected	66.240.205.34	1066		16464	1:31136	MALWARE-CNC Win.Trojan.ZeroAccess inbound connection
2017-07-22 06:15:49	2	UDP	Potentially Bad Traffic	163.172.17.76	54465		5060	140:26	(spp_sip) Method is unknown
2017-07-21 09:26:30	2	UDP	Potentially Bad Traffic	163.172.22.169	52428		5060	140:26	(spp_sip) Method is unknown
2017-07-21 01:03:28	2	UDP	Potentially Bad Traffic	163.172.17.76	46834		5060	140:26	(spp_sip) Method is unknown
2017-07-20 20:36:37	2	UDP	Potentially Bad Traffic	163.172.22.169	54788		5060	140:26	(spp_sip) Method is unknown
2017-07-20 08:31:30	2	UDP	Potentially Bad Traffic	163.172.17.76	59571		5060	140:26	(spp_sip) Method is unknown

Рисунок 1.3 – Snort [10]

Основні можливості:

– аналіз трафіку в режимі реального часу – Snort відстежує мережевий трафік у режимі реального часу для виявлення потенційних загроз;

- виявлення на основі підписів – використовує повний набір заздалегідь визначених правил для виявлення відомих загроз та моделей атак;
- аналіз протоколів – аналізує мережеві протоколи для виявлення шкідливої активності;
- гнучкість – може бути налаштований для роботи в різних режимах, в тому числі в якості реєстратора пакетів, мережевого аналізатора або повноцінної системи запобігання вторгнень [9], [10];
- регулярні оновлення – постійно поповнюється новими правилами і сигнатурами, щоб бути в курсі виникаючих загроз.

Варіанти використання:

- виявлення DDoS-атак – Snort може виявляти шаблони трафіку, характерні для DDoS-атак, такі як незвично високий трафік з декількох джерел [9], [10];
- ідентифікація мережевих зондів – виявляє мережеві зонди та дії сканування, розпізнаючи підписи відомих інструментів та методів сканування;
- моніторинг звичайного трафіку – допомагає встановити базовий рівень для нормального мережевого трафіку, що полегшує виявлення відхилень, які можуть вказувати на проблему безпеки [9], [10].

SolarWinds Network Performance Monitor – це потужний інструмент моніторингу мережі, призначений для забезпечення всебічного контролю продуктивності мережі та трафіку. Це допомагає виявити, діагностувати та усунути проблеми з мережею, включаючи загрози безпеці [11], [12].

Основні можливості:

- моніторинг мережі в режимі реального часу – постійний моніторинг продуктивності мережі та трафіку в режимі реального часу [11], [12];
- налаштованість оповіщення – генерує оповіщення на основі заздалегідь визначених порогових значень і умов, допомагаючи адміністраторам швидко реагувати на потенційні проблеми;

- аналізатор трафіку NetFlow – надає детальну інформацію про структуру мережевого трафіку та використання пропускної здатності;
- комплексні інформаційні панелі – пропонує інтуїтивно зрозумілі інформаційні панелі, що відображають найважливіші мережеві показники і продуктивність [11], [12];
- масштабованість – підходить для мереж будь-якого розміру, від малого бізнесу до великих підприємств.

Варіанти використання [12], [13]:

- виявлення DDoS-атак – аналізатор трафіку NetFlow може виявляти раптові сплески трафіку, які можуть вказувати на DDoS-атаку;
- ідентифікація пробників – відстежує мережевий трафік на предмет незвичайних моделей, які можуть вказувати на дії з перевірки, такі як повторні спроби підключення до декількох портів;
- моніторинг звичайного трафіку – допомагає встановити та візуалізувати звичайні моделі трафіку, що полегшує виявлення та розслідування аномалій [12], [13].

Wireshark, Snort і Solarwinds network Performance Monitor – три найбільш популярних і ефективних інструменту для виявлення DDoS-атак, мережевих зондів і моніторингу звичайного трафіку. Кожен із цих інструментів має унікальні функції та можливості, які роблять їх придатними для різних аспектів мережевої безпеки. Використовуючи ці програми, адміністратори мережі можуть отримувати повну інформацію про свій мережевий трафік, ефективніше виявляти та реагувати на загрози та підтримувати безпечну та стійку мережеву інфраструктуру [12], [13].

Спробуємо провести порівняння цих рішень за наступними критеріями:

- призначення – основна функція або роль інструменту. Допомагає визначити, який інструмент підходить для вирішення конкретних завдань, таких як аналіз пакетів, виявлення вторгнень або моніторинг продуктивності мережі;

- основні характеристики – основні функціональні можливості інструменту. Виділення ключових функцій допомагає визначити, що може робити кожен інструмент і як його можна ефективно використовувати для забезпечення мережевої безпеки;
- варіанти використання – сценарії або ситуації, в яких зазвичай використовується цей інструмент;
- переваги – сильні сторони та переваги використання цього інструменту. Розуміння переваг допомагає оцінити позитивні аспекти, які можуть зробити один інструмент кращим за інший у певних ситуаціях;
- обмеження – недоліки або труднощі, пов'язані з використанням інструменту. Знання обмежень допомагає оцінити потенційні проблеми або стримуючі фактори, які можуть вплинути на рішення про використання конкретного інструменту;
- розгортання – як інструмент встановлюється та використовується в мережевому середовищі. Міркування, пов'язані з впровадженням, впливають на простоту використання, інтеграцію з існуючими системами та загальний процес налаштування;
- ціна – витрати, пов'язані з придбанням та використанням інструменту. Ціноутворення впливає на прийняття бюджетних рішень та аналіз витрат і вигод, особливо при порівнянні безкоштовних рішень з відкритим кодом з платними варіантами;
- простота використання – наскільки зручний інструмент для користувача, включаючи процес навчання та зручність використання. Простота використання важлива для забезпечення ефективного використання інструменту передбачуваними користувачами, будь то новачки або просунуті професіонали;
- спільнота та підтримка – доступність ресурсів спільноти, документації та професійної підтримки. Сильна спільнота та підтримка можуть покращити зручність використання інструменту, допомогти усунути неполадки та забезпечити постійне вдосконалення за допомогою оновлень.

Ці функції та аспекти були обрані, оскільки вони відображають основні міркування, необхідні для оцінки інструментів безпеки мережі. Вони дають повне уявлення про те, що пропонує кожен інструмент, його практичне застосування, переваги, проблеми, вимоги до розгортання, фінансові наслідки, зручність користування та доступну підтримку. Це допомагає користувачам приймати обґрунтовані рішення на основі їх конкретних потреб, бюджету та технічних знань.

Аналіз програмних засобів виявлення мережових атак наведено у табл. 1.1.

Таблиця 1.1 – Програмні засоби виявлення мережових атак

Критерій порівняння	Wireshark	Snort	SolarWinds Network Performance Monitor
1	2	3	4
Мета	Аналізатор мережових протоколів	Система виявлення та запобігання вторгнень	Моніторинг та управління продуктивністю мережі
Ключові функції	глибока перевірка пакетів, аналіз в режимі реального часу і в автономному режимі, підтримка широкого спектру протоколів, налаштовані звіти, кросплатформна сумісність	аналіз трафіку в режимі реального часу; виявлення на основі сигнатур; аналіз протоколів; гнучкі моделі розгортання; регулярні оновлення	моніторинг мережі в режимі реального часу; налаштовані оповіщення; аналізатор трафіку NetFlow; комплексні інформаційні панелі; масштабованість
Варіанти використання	виявлення DDoS-атак; ідентифікація зондів; моніторинг звичайного трафіку	виявлення DDoS-атак; ідентифікація зондів; моніторинг звичайного трафіку	виявлення DDoS-атак; ідентифікація зондів; моніторинг звичайного трафіку

Продовження таблиці 1.1

1	2	3	4
Перевага	детальний аналіз на рівні пакетів; універсальна підтримка протоколів; з відкритим вихідним кодом	виявлення загроз в режимі реального часу; повний набір правил; легко настраюється	цілісна видимість мережі; простота у використанні; широкі можливості створення звітів
Обмеження	може бути складним для новачків; вимагає великих ресурсів для великих знімків; в першу чергу це інструмент діагностики, а не профілактики	вимагає регулярного оновлення сигнатур; може генерувати помилкові спрацьовування; вимагає регулярного налаштування та обслуговування	вища вартість порівняно з інструментами з відкритим кодом; може знадобитися спеціальне обладнання; може бути складно налаштувати додаткові функції
Розгортання	Desktop застосунок	ПЗ розгорнуто в мережі	ПЗ для виділеного серверу або машини
Вартість	Безкоштовне (з відкритим кодом)	Безкоштовне (з відкритим кодом)	Платно (залежить від ліцензії та функцій)
Простота використання	Від середнього до просунутого рівня	Від середнього до просунутого рівня	Помірна
Спільнота та підтримка	Потужна підтримка спільноти, доступна документація	Сильна спільнота, велика документація	Доступна професійна підтримка, вичерпні ресурси

Підводячи підсумки порівняння, можна відзначити, що Wireshark ідеально підходить для детального аналізу на рівні пакетів, корисний для діагностики мережеских проблем і розуміння поведінки протоколу. Він безкоштовний і з відкритим кодом, але може бути складним для початківців і ресурсоємним при великих обсягах перехоплення.

Snort надає надійні можливості виявлення та запобігання вторгненням, включаючи аналіз трафіку в режимі реального часу та виявлення на основі підписів.

Він також безкоштовний і з відкритим кодом, але вимагає регулярних оновлень і налаштування для підтримки ефективності.

Solarwinds network Performance Monitor (NPM) пропонує комплексний моніторинг продуктивності мережі з інтуїтивно зрозумілими панелями моніторингу, налаштованим оповіщеннями і великою звітністю.

Він забезпечує цілісне уявлення про стан мережі, але вимагає більш високих витрат і може зажадати спеціального обладнання.

Проте жоден інструмент не використовує принципи штучного інтелекту або ж машинного навчання для покращення результатів. Саме тому актуальною задачею є програмна реалізація системи виявлення мережових атак на основі нечіткого виведення.

1.5 Постановка завдання

ІІІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ІІІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація програмної системи для виявлення мережових атак на основі нечіткого виведення.

1.5.1 Функціональність системи

Система має реалізовувати наступні функціональні вимоги [13]–[15]:

- збір та реєстрація даних (моніторинг мережевого трафіку – система повинна постійно відстежувати та збирати дані про мережевий трафік у режимі реального часу з кількох джерел (наприклад, маршрутизаторів, комутаторів, брандмауерів); управління журналами – збір журналів з різних мережевих пристроїв і додатків для забезпечення всебічного контролю за мережевою активністю);
- попереднє оброблення (очищення даних – видалення шуму та непотрібної інформації із зібраних даних для забезпечення високоякісного введення для аналізу; нормалізація даних – приведення даних до стандартної шкали для полегшення точних порівнянь та аналізу);
- витяг характеристик (витяг характеристик трафіку – витягніть відповідні характеристики з мережевого трафіку, такі як розмір пакета, тип протоколу, IP-адреси джерела та призначення та номери портів; витяг поведінкових характеристик – витягніть поведінкові характеристики, такі як тривалість з'єднання, частота підключень і незвичайні схеми трафіку);
- розробка системи нечіткого виведення (розробка бази правил – розробка повного набору нечітких правил, заснованого на експертних знаннях, для визначення нормальної та ненормальної поведінки мережі; функції приналежності – визначення нечітких функцій приналежності для кожної вхідної ознаки, щоб впоратися з невизначеністю і неточностями в даних; механізм логічного висновку – впровадити механізм логічного висновку, щоб застосувати нечіткі правила до вхідних даних та визначити ймовірність атаки);
- виявлення атаки (виявлення аномалій – визначати відхилення від нормальної поведінки за допомогою нечіткої логіки для виявлення потенційних мережевих атак; виявлення на основі підписів – порівняння мережевої активності із відомими сигнатурами атак, використовуючи нечіткий логічний висновок для ідентифікації конкретних типів атак; гібридне

виявлення – комбінування методів виявлення на основі аномалій та підписів для комплексного виявлення атак);

– генерація попереджень та створення звітів (генерація попереджень – генерація попередження при виявленні потенційних атак, класифікуючи їх за ступенем серйозності (наприклад, низька, середня, висока); визначення пріоритетності сповіщень – визначення пріоритетності сповіщень в залежності від серйозності і потенційного впливу виявленої атаки; звітність – створення детальних звітів з коротким описом виявлених атак, включаючи мітки часу, інформацію про джерело/одержувача та тип атаки).

1.5.2 Вимоги до інтерфейсу

Наявність інформаційної панелі забезпечує зручну подачу для моніторингу активності мережі, перегляду сповіщень та доступу до звітів.

Впровадження інструментів візуалізації для графічного представлення мережевого трафіку та виявлених атак, що допомагає швидко аналізувати та приймати рішення [13]–[15].

Управління сповіщеннями: пропонує інтерфейс для управління сповіщеннями, включаючи перегляд детальної інформації, підтвердження та закриття сповіщень.

1.5.3 Вимоги до безпеки

Обов'язкове шифрування даних для захисту конфіденційної інформації.

Управління доступом – впровадження управління доступом на основі ролей, щоб обмежити доступ до системних функцій та даних на основі ролей користувачів [13]–[15].

1.5.4 Не функціональні вимоги до продукту

Апаратні вимоги:

- IBM-сумісний персональний комп'ютер;
- центральний процесор з тактовою частотою від 2 GHz;
- оперативна пам'ять формату DDR4, об'ємом від 3 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 1 Гб;
- підключення до мережі Інтернет.

Програмні вимоги:

- розгорнута операційна система (ОС), наприклад: Windows, Linux або macOS;
- наявність веббраузера актуальної версії.

1.6 Висновки за розділом 1

Перший розділ пояснювальної записки до роботи було присвячено ознайомленню із проблемною областю. Після аналітичної частини було наведено перелік функціональних та нефункціональних вимог до ПЗ, що буде розроблятися під час роботи.

2 МАТЕРІАЛИ І МЕТОДИ

Другий розділ роботи над пояснювальною запискою буде присвячено дослідженню та обґрунтуванню вибору технічних засобів розробки. Буде наведено пояснення та представлено матеріали, що засвідчують вибір тих чи інших рішень..

2.1 Вибір мови програмування

Java може бути кращим вибором для реалізації системи виявлення мережових атак, заснованої на нечіткому логічному висновку, з кількох вагомих причин. Нижче наведено ключові фактори, які роблять Java кращою мовою програмування для такої системи [16], [17].

Незалежність від платформи – незалежність Java від платформи означає, що один і той же код може виконуватися на будь-якому пристрої, оснащеному віртуальною машиною Java (JVM). Ця функція має вирішальне значення для розробки системи виявлення мережових атак, яка повинна працювати на різних платформах і середовища [16], [17].

Комплексні бібліотеки – Java пропонує багатий набір бібліотек та API, які полегшують мережеве програмування, багатопотоковість, структури даних тощо. Ці бібліотеки можуть полегшити реалізацію мережевого моніторингу, обробки даних та нечітких механізмів виведення.

Мережеві API – Java надає широку підтримку мереж за допомогою класів у пакеті `java.net`, що спрощує розробку функціональних можливостей, пов'язаних з мережею [16], [17].

Ефективне збирання сміття – автоматичне збирання сміття в Java допомагає ефективно керувати пам'яттю, зменшуючи ризик витоків пам'яті, що може погіршити продуктивність системи [16], [17].

Можливості багатопоточності – вбудовані в Java можливості багатопоточності дозволяють створювати швидко реагують і

високопродуктивні системи виявлення мережових атак, які можуть виконувати кілька завдань одночасно.

Функції безпеки – Java включає в себе надійні функції безпеки, такі як Java Security Manager, який забезпечує детальний контроль над системними ресурсами. Це важливо для NADS, яким потрібно безпечно працювати в потенційно небезпечному середовищі [16], [17].

Ізольоване середовище – Java-застосунки можуть запускатися в середовищі з обмеженим доступом (sandbox), що підвищує безпеку за рахунок обмеження можливостей програми [16], [17].

Об'єктно-орієнтоване програмування – об'єктно-орієнтований характер Java спрощує управління складними системами, розбиваючи їх на багаторазово використовувані об'єкти і класи. Модульність допомагає підтримувати та розширювати систему.

Читабельний синтаксис – синтаксис Java розроблений таким чином, щоб його було легко читати та розуміти, що полегшує співпрацю та обслуговування великими командами розробників.

Велика спільнота розробників – Java має велику спільноту розробників, що означає наявність багатьох ресурсів, бібліотек, фреймворків та інструментів. Підтримка спільноти може виявитися неоціненною для усунення несправностей та оптимізації NAD [16], [17].

Корпоративна підтримка – Java широко використовується в корпоративних середовищах, що забезпечує довгострокову підтримку та регулярні оновлення від таких постачальників, як Oracle та спільнота OpenJDK.

Інтеграція з іншими технологіями – Java може легко інтегруватися з іншими технологіями та системами, такими як бази даних (через JDBC), вебсервіси (через RESTful API) та черги повідомлень (через JMS). Це полегшує інтеграцію NADS у більш широкую інфраструктуру безпеки [16], [17].

Сумісність – можливість Java викликати машинний код через Java Native Interface (JNI) дозволяє здійснювати інтеграцію з існуючими системами і бібліотеками, які можуть бути написані на інших мовах, таких як C або C++.

Наявність бібліотек нечіткої логіки – для реалізації нечіткої логіки доступно кілька бібліотек Java, таких як JFuzzyLogic та Fuzzy4J. ці бібліотеки можуть полегшити розробку нечіткої вихідної системи, надаючи готові до використання компоненти [16], [17].

JIT-компіляція – JIT-компілятор Java оптимізує байт-код до машинного під час виконання, підвищуючи продуктивність програми. Така динамічна оптимізація корисна для систем, яким потрібно обробляти мережеві дані в режимі реального часу [16], [17].

Незалежність Java від платформи, надійна стандартна бібліотека, можливості продуктивності, функції безпеки, простота використання, потужна підтримка спільноти, можливості інтеграції та доступність інструментів розробки роблять його відповідним вибором для розробки системи виявлення мережевих атак на основі нечіткого виводу. Його повний набір функцій та інструментів може допомогти розробникам створити масштабовану, безпечну та підтримувану систему, яка ефективно виявляє та реагує на мережеві атаки [16], [17].

Вибір мови програмування представимо виконавши порівняння за наступними критеріями:

- кросплатформність – здатність програм на мові програмування запускатися в різних операційних системах без змін. Забезпечує можливість розгортання системи в різних середовищах, підвищуючи гнучкість і доступність;

- стандартні бібліотеки – набір готових класів і функцій, що надаються мовою. Багата стандартна бібліотека спрощує розробку, надаючи необхідні інструменти і функціональні можливості з коробки;

- продуктивність – швидкість і ефективність, з якими виконується програма. Висока продуктивність має вирішальне значення для виявлення мережових атак у режимі реального часу та обробки великих обсягів даних;
- Управління пам'яттю – як мова програмування обробляє виділення, звільнення і збірку сміття в пам'яті. Ефективне управління пам'яттю має важливе значення для підтримки стабільності та продуктивності системи, особливо в ресурсномістких додатках;
- багатопоточність – можливість одночасного виконання декількох потоків. Багатопоточність важлива для вирішення завдань одночасної обробки даних, підвищення швидкодії і продуктивності;
- функції безпеки – вбудовані механізми, що забезпечують безпечне виконання коду та обробку даних. Безпека має першорядне значення в системі виявлення мережових атак для захисту від вразливостей і забезпечення цілісності даних;
- простота використання – швидкість вивчення та зручність використання мови. Простіші мови скорочують час розробки та знижують бар'єр для входу нових розробників;
- швидкість розробки – час, необхідний для написання, тестування та розгортання коду. Більш висока швидкість розробки забезпечує більш швидку ітерацію і впровадження оновлень і функцій;
- спільнота та підтримка – доступність ресурсів, документації та допомоги колег. Потужна підтримка спільноти допомагає вирішувати проблеми та отримувати доступ до багатства загальних знань та інструментів;
- можливості інтеграції – простота, з якою мова може взаємодіяти з іншими системами та технологіями. Можливості інтеграції важливі для підключення NADS до існуючої інфраструктури та інших засобів забезпечення безпеки;
- інструменти розробки – доступність середовища розробки, налагоджувачів та інших допоміжних засобів розробки. Хороші інструменти розробки підвищують продуктивність та якість коду;

- бібліотеки нечіткої логіки – доступність бібліотек для реалізації нечіткого логічного висновку. Вже існуючі бібліотеки полегшують реалізацію нечіткої логіки, прискорюючи розвиток;
- процес навчання – наскільки важко вивчити мову та оволодіти нею досконало. Більш високий рівень навчання може вплинути на час розробки та можливість залучення нових членів команди;
- вартість розробки – загальні витрати, пов'язані з розробкою, розгортанням та обслуговуванням системи. Облік витрат має вирішальне значення для складання бюджету та розподілу ресурсів;
- масштабованість – здатність справлятися зі зростаючими обсягами роботи або розширюватися. Масштабованість дозволяє системі рости разом з мережею і обробляти великі обсяги даних;
- технічне обслуговування – простота оновлення системи і підтримки її в робочому стані. Простота обслуговування знижує довгострокові експлуатаційні витрати і трудовитрати;
- обробка в режимі реального часу – можливість обробляти дані і реагувати на події в режимі реального часу. Важливо для оперативного виявлення мережових атак і реагування на них;
- спільнота та документація – доступність офіційної документації та ресурсів, наданих спільнотою. Доступ до всеосяжних ресурсів допомагає розробникам долати труднощі та вдосконалювати свої знання;
- обробка помилок – механізми управління помилками під час виконання і реагування на них. Надійна обробка помилок підвищує надійність системи і зручність роботи з користувачем;
- читаність коду – наскільки легко читати та розуміти код. Читабельний код полегшує співпрацю, налагодження та внесення змін у майбутньому;
- сумісність – можливість роботи з іншими мовами і системами. Сумісність гарантує, що NADS можуть легко інтегруватися з різними компонентами в мережевому середовищі.

Ці критерії були обрані тому, що вони відповідають найважливішим вимогам і завданням при розробці системи виявлення мережових атак на основі нечіткого логічного висновку. Вони дають повне уявлення про те, як кожна мова програмування може відповідати вимогам такої системи, включаючи продуктивність, безпеку, простоту використання, інтеграцію та зручність обслуговування. Оцінюючи ці аспекти, розробники можуть прийняти обґрунтоване рішення щодо найбільш підходящої мови для їх конкретних потреб.

Порівняння мов програмування наведено у табл. 2.1

Таблиця 2.1 – Порівняння мов програмування

Критерій	Java	C++	Python
1	2	3	4
Кросплатформність	Високий рівень	Середній рівень	Високий рівень
Стандартні бібліотека	Надійні бібліотеки для мережової взаємодії, паралелізму і т. д.	Великий, але може вимагати використання бібліотек сторонніх виробників	Велика стандартна бібліотека, безліч модулів сторонніх виробників
Продуктивність	Високий рівень	Високий рівень	Помірний рівень
Управління пам'яттю	Автоматичне збирання сміття	Ручне управління пам'яттю, можливі витіки пам'яті	Автоматичне керування пам'яттю за допомогою збирання сміття
Багатопоточність	Потужна вбудована підтримка	Потужна підтримка, але вимагає ретельного управління	GIL обмежує багатопотоковість; підходить для завдань, пов'язаних з введенням / виведенням
Функції безпеки	Вбудований менеджер безпеки, можливості ізолюваного середовища	Потрібні додаткові бібліотеки та ретельне кодування	Хороші бібліотеки, але менш безпечні за замовчуванням

Продовження таблиці 2.1

1	2	3	4
Швидкість розробки	Помірна	Повільна	Висока
Простота використання	Синтаксис досить детальний, але зрозумілий	Складний-вимагає детального розуміння низькорівневих операцій	Високий рівень-простий і читабельний синтаксис
Спільнота підтримка та	Великі, великі ресурси та бібліотеки	Великий обсяг, особливо в системному програмуванні	Великі та великі ресурси та бібліотеки
Можливості інтеграції	Висока	Висока	Висока
Бібліотеки нечіткої логіки	Доступні	Обмежений набір	Розширений набір
Процес навчання	Помірний рівень	Складний рівень	Низький рівень
Вартість розробки	Помірна	Висока	Низька
Масштабованість	Високий рівень	Високий рівень	Високий рівень
Підтримка	Помірний рівень	Складний рівень	Легкий рівень
Обробка в режимі реального часу	Хороша	Найкраща	Хороша
Спільнота та документація	Велика документація та активна спільнота	Великий, але орієнтований на системне програмування	Велика документація та активна спільнота
Обробка помилок	Надійний механізм обробки винятків	Обробка помилок вручну	Надійний механізм обробки винятків
Читаність коду	Помірний рівень	Високий рівень	Високий рівень
Сумісність	Високопродуктивний інтерфейс Java для виклику машинного коду	Високий ступінь взаємодії з апаратним забезпеченням і низькорівневим кодом	High-дозволяє легко викликати код на C / C++

Недоліки Java пов'язані зі складним синтаксисом, помірною швидкістю розвитку сучасних нотацій мови та помірно складного рівня вивчення навчання.

Враховуючі чисельні переваги: кросплатформність розробок, надійність стандартних бібліотек, хорошу продуктивність, надійні функції безпеки, потужна підтримка спільноти, чудові можливості інтеграції – вибір був зроблений саме на користь Java.

2.2 Вибір середовища розробки

Найпопулярніші та найрозповсюдженіші середовища розробки (IDE) для мови програмування Java на сьогоднішній день: Eclipse, NetBeans та IntelliJ IDEA [18], [19].

Важливо відібрати критерії за якими порівняти ці середовища для аргументованого вибору. Так для порівняння були обрані критерії на основі їх відповідності ключовим факторам, які розробники враховують при виборі IDE для розробки Java. Ці фактори охоплюють широкий спектр функціональних можливостей і якостей, які впливають на продуктивність, результативність і загальний досвід роботи з користувачем. Розглянемо їх детальніше [18], [19]:

- підтримка платформи – операційні системи (ОС), на яких може працювати IDE. Забезпечує можливість використання IDE в різних середовищах розробки (Windows, macOS, Linux);
- продуктивність – швидкість і чуйність IDE, особливо при великих робочих навантаженнях. Впливає на продуктивність розробників і зручність роботи з користувачами, особливо при роботі над великими проєктами;
- простота використання – наскільки інтуїтивно зрозумілим та зручним є IDE як для початківців, так і для досвідчених розробників. Визначає, наскільки швидко розробники зможуть освоїти та ефективно використовувати IDE;

- інтерфейс користувача – дизайн, компонування та можливості налаштування графічного інтерфейсу IDE. Впливає на зручність використання та наскільки легко розробники можуть орієнтуватися та використовувати функції IDE [18], [19];

- завершення коду – здатність IDE надавати пропозиції та автоматично заповнювати код. Підвищує швидкість кодування і зменшує кількість помилок, допомагаючи розробникам писати код більш ефективно;

- інструменти рефакторингу – інструменти, які допомагають розробникам реструктуризувати існуючий код без зміни його зовнішньої поведінки. Важливо для підтримки та покращення якості коду та керованості;

- налагодження – можливості і простота використання засобів налагодження IDE. Важливо для виявлення та виправлення помилок у коді;

- інтеграція з системою контролю версій – підтримка систем контролю версій, таких як Git, SVN і т.д. Важлива для управління змінами коду, співпраці з членами команди та ведення історії коду;

- плагіни та розширення – доступність та різноманітність плагінів та розширень для розширення функціональності IDE. Дозволяє налаштувати та розширити IDE для задоволення конкретних потреб розробки;

- вбудовані інструменти – попередньо включені інструменти для різних завдань розробки (наприклад, інструменти побудови, профілювання, управління базами даних). Зменшується потреба в зовнішніх інструментах і спрощується робочий процес розробки;

- налаштованість – можливість адаптувати налаштування і зовнішній вигляд IDE до індивідуальних переваг. Підвищення комфорту та ефективності роботи розробника шляхом створення персоналізованого робочого середовища [18], [19];

- спільнота та підтримка – розмір і активність спільноти користувачів, а також наявність офіційної підтримки. Забезпечує доступ до ресурсів, довідки та оновлень, що є життєво важливим для довгострокового використання;

- процес навчання – наскільки швидко розробники можуть освоїти IDE. Впливає на час, необхідний для того, щоб увійти в курс справи і почати продуктивно працювати з IDE;
- управління проектом – функції, які допомагають в управлінні та організації проектів з розробки коду. Важливо для ефективної роботи з великими та складними проектами;
- вбудований профілювальник – інструменти для аналізу та оптимізації продуктивності коду. Допомагає виявити вузькі місця в продуктивності та підвищити ефективність застосунків;
- використання пам'яті – обсяг системних ресурсів, споживаних IDE. Впливає на продуктивність системи, особливо при роботі на менш потужних комп'ютерах;
- підтримка Java EE – підтримка розробки на Java Enterprise Edition. Необхідна розробникам, які працюють над додатками корпоративного рівня;
- мобільна розробка – підтримка розробки мобільних застосунків. Важливо для розробників, яким необхідно створювати мобільні додатки поряд з традиційними Java-застосунками;
- ліцензія – тип ліцензії, за якою поширюється IDE (безкоштовна, з відкритим вихідним кодом, платна). Впливає на вартість та доступність для окремих розробників та організацій;
- інтеграція із засобами збірки – сумісність із засобами автоматизації збірки, такими як Maven і Gradle. Спрощує процес складання та добре інтегрується з сучасними методами розробки;
- корпоративні функції – функції, спеціально призначені для розробки на рівні підприємства. Необхідні великим організаціям зі складними потребами в розробці та розгортанні;
- підтримка інших мов – можливість розробки на мовах, відмінних від Java. Забезпечує гнучкість для багатомовних проектів та розробників поліглотів;

– оновлення та обслуговування – частота і надійність оновлень і обслуговування, що надаються IDE. Забезпечує відповідність IDE новим технологіям і підтримує стабільність і безпеку.

Ці критерії були обрані, оскільки вони охоплюють основні функціональні можливості та якості, які впливають на ефективність IDE для розробки Java. Вони охоплюють різні аспекти, такі як продуктивність, зручність використання, набір інструментів, підтримка спільноти та розширюваність, надаючи всебічне уявлення про сильні та слабкі сторони кожної IDE. Оцінюючи ці аспекти, розробники можуть прийняти обґрунтоване рішення щодо вибору IDE, яка найкраще відповідає їхнім конкретним потребам та робочим процесам.

Порівняння середовищ розробки наведено у табл. 2.2.

Таблиця 2.2 – Порівняння середовищ розробки

Критерій	Eclipse	NetBeans	IntelliJ IDEA
1	2	3	4
Підтримка платформи	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux
Продуктивність	Помірна	Помірна	Висока
Простота використання	Помірна	Висока	Висока
Інтерфейс користувача	Настроюваний	Зрозумілий	Інтуїтивно зрозумілий
Завершення коду	Хороший рівень	Хороший рівень	Хороший рівень
Інструменти рефакторингу	Помірний рівень	Хороший рівень	Хороший рівень
Налагодження	Широкі можливості	Широкі можливості	Широкі можливості
Інтеграція системою контролю версій ³	Підтримує Git, SVN, CVS і т. д.	Підтримує Git, SVN, Mercurial і т. д.	Підтримує з Git, SVN і т. д.
Плагіни та розширення	Велика екосистема плагінів	Хороший вибір плагінів	Велика екосистема плагінів
Вбудовані інструменти	Включає в себе безліч інструментів	Включає кілька вбудованих інструментів	Включає повний набір вбудованих інструментів

Продовження таблиці 2.2

1	2	3	4
Процес навчання	Простий, але вимагає практики	Легкий для початківців	Легко підібрати і освоїти
Управління проєктом	Потужні функції управління проєктами	Ефективне управління проєктами	Просунуте управління проєктами
Настройка	Легко налаштовується	Обмежені налаштування	Легко налаштовується
Спільнота та підтримка	Велика спільнота та документація	Хороша спільнота та офіційна підтримка	Велика активна спільнота та чудова підтримка
Вбудований профілювальник	Потрібні додаткові плагіни	Вбудований профайлер	Вбудований профайлер
Використання пам'яті	Може бути ресурсоємним	Помірний – загалом ефективний	Помірно ефективний
Підтримка Java EE	Сильна підтримка плагінів	Сильна підтримка плагінів	Всебічна підтримка
Розробка мобільних додатків	Потрібні плагіни	Вбудована підтримка мобільних платформ	Чудово-інтегрована підтримка Android
Ліцензія	Безкоштовний	Безкоштовна	Безкоштовна / платна версії
Інтеграція із засобами збірки	Підтримує Maven, Gradle і т. д.	Підтримує Maven, Gradle і т. д.	Підтримує Maven, Gradle і т. д.
Корпоративні функції	Хороші корпоративні можливості	Базові корпоративні можливості	Великі корпоративні можливості
Підтримка інших мов	Хороший – підтримує багато мов за допомогою плагінів	В основному орієнтовано на Java і пов'язані з нею програми	Підтримка багатьох мов
Оновлення та технічне обслуговування	Регулярні оновлення та нові версії	Регулярні оновлення та нові версії	Регулярні оновлення та нові версії

IntelliJ IDEA має низку переваг: відмінна продуктивність, інтелектуальне завершення коду, розширений рефакторинг, інтуїтивно зрозумілий користувальницький інтерфейс, комплексні вбудовані інструменти, потужні корпоративні функції.

Загалом, IntelliJ IDEA пропонує відточений, високопродуктивний інтерфейс з розширеними функціями, придатними для професійного та корпоративного розвитку.

2.3 Використання нечіткої логіки

Нечітка логіка в МН – це форма логіки, яка стосується міркувань, які є приблизними, а не фіксованими та точними. На відміну від традиційної бінарної логіки, яка стверджує, що твердження має бути або істинним, або хибним, нечітка логіка допускає різні ступені істинності. Це особливо корисно при роботі з реальними сценаріями, де дані часто є невизначеними або неточними.

Нечіткі множини – у нечіткій логіці змінні можуть мати діапазон значень, кожне з яких має певний ступінь приналежності. Наприклад, замість того, щоб називати людину просто «молодою» або «старою», нечітка логіка допускає ступені «молодості» або «старості», представлені значеннями від 0 до 1 [20], [21].

Функції приналежності – ці функції визначають, як кожній точці у вхідному просторі зіставляється значення приналежності від 0 до 1. Загальні форми функцій приналежності включають трикутні, трапецієподібні і гауссові криві.

Нечіткі правила – це твердження типу «Якщо – то», які формують основу для прийняття рішень в системі нечіткої логіки. Наприклад, «якщо температура висока, а вологість низька, то швидкість обертання вентилятора висока».

Система нечіткого виводу – це процес формування відображення із заданих вхідних даних у вихідні з використанням нечіткої логіки. Він включає в себе [20], [21]:

- фаззифікація – перетворення чітких вхідних даних в нечіткі значення з використанням функцій приналежності;
- оцінка правил – застосування нечітких правил до нечітких вхідних даних для генерації нечітких вихідних даних;
- агрегування – об'єднання нечітких вихідних даних усіх правил;
- дефазифікація – перетворення агрегованих нечітких вихідних даних назад у чітке значення.

Робота з невизначеністю та неточностями – нечітка логіка особливо ефективна при роботі з галасливими або неповними даними. Вона дозволяє моделям машинного навчання приймати рішення, навіть якщо вхідні дані неточні [20], [21].

Покращена інтерпретація – нечіткі логічні системи часто легше зрозуміти та інтерпретувати порівняно з традиційними моделями машинного навчання, оскільки вони використовують мовні змінні та правила, що імітують людське мислення [20], [21].

Системи управління – нечітка логіка широко використовується в системах управління, наприклад, для регулювання температури в системі кондиціонування повітря або швидкості автомобіля. Нечіткі правила можуть бути легко змінені для налаштування поведінки системи.

Класифікація – нечітка логіка може бути використана для класифікації даних за категоріями, які не є взаємовиключними. Наприклад, об'єкт може бути класифікований як «важкий» і «великий» в різному ступені.

Нечітка логіка може бути інтегрована з МН різними способами:

- нечіткі дерева рішень – традиційні дерева рішень можуть бути доповнені нечіткою логікою для більш ефективної обробки безперервних даних за рахунок використання нечітких множин і правил замість жорстких порогових значень;

- нечітка кластеризація – такі алгоритми, як Fuzzy C-Means, дозволяють відносити точки даних до декількох кластерів з різним ступенем приналежності, забезпечуючи більш детальну кластеризацію, ніж традиційні методи [20], [21];

- нечіткі нейронні мережі – нейронні мережі можуть використовувати нечітку логіку, щоб краще впоратися з невизначеністю. Цього можна досягти шляхом інтеграції нечіткої логіки в нейрони або шари мережі;

- системи, засновані на нечітких правилах – методи МН можуть використовуватися для автоматичної генерації або оптимізації нечітких правил і функцій приналежності, поєднуючи гнучкість нечіткої логіки з можливостями машинного навчання [20], [21].

Переваги нечіткої логіки в МН:

- гнучкість – системи нечіткої логіки можна легко модифікувати, змінюючи правила і функції приналежності;

- надійність – вони стійкі до перешкод і можуть ефективно обробляти неточні вхідні дані;

- інтерпретація – використання лінгвістичних змінних та правил робить систему більш зрозумілою;

- взаємодоповнюваність – нечітка логіка може доповнювати інші методи машинного навчання, надаючи додаткові інструменти для вирішення невизначеності та неточностей.

Складнощі нечіткої логіки в МН [20], [21]:

- складність формулювання правил – створення відповідного набору нечітких правил може бути складним і трудомістким;

- обчислювальна потужність – системи нечіткої логіки, особливо з великою кількістю правил і функцій приналежності, можуть вимагати великих обчислювальних витрат;

– масштабованість – вирішення масштабних проблем за допомогою нечіткої логіки може бути складним завданням через експоненціальне зростання комбінацій правил [20], [21].

Нечітка логіка забезпечує потужний інструмент для вирішення невизначеностей та неточностей, властивих реальним даним світу. При інтеграції з машинним навчанням вона підвищує здатність моделі приймати деталізовані рішення, обробляти зашумлені дані і надавати інтерпретовані результати. Це робить його особливо цінним у сферах, де потрібна здатність міркувати та приймати рішення, як людина [20], [21].

2.4 Структурна схема розробки

Загальна схема системи за обраною темою на рис. 2.1.

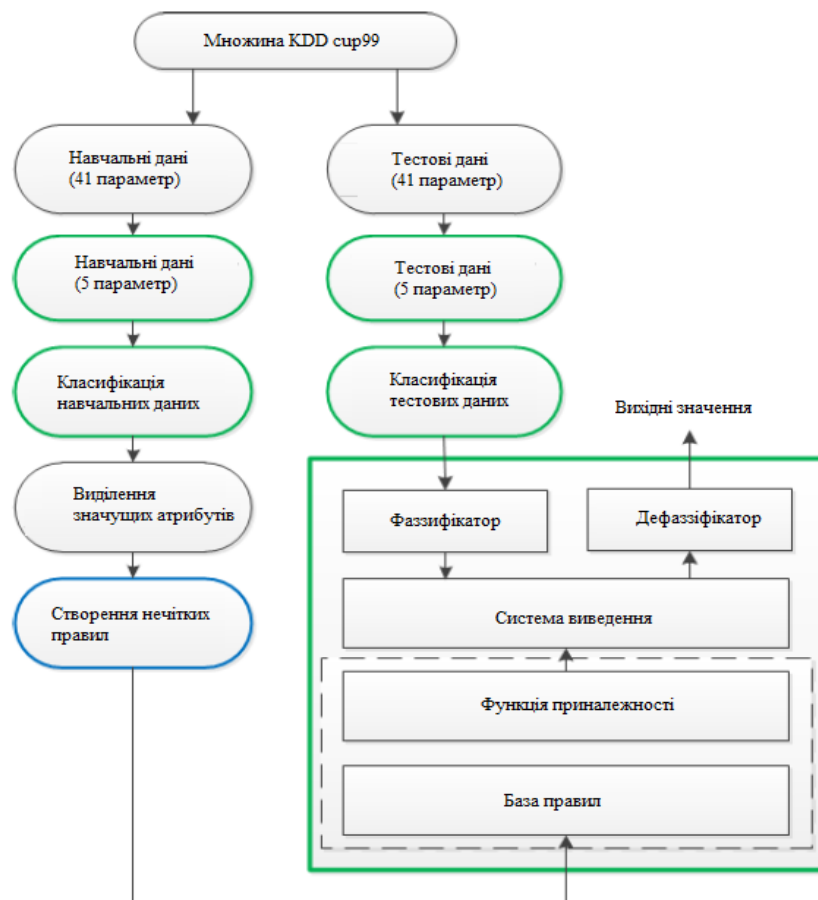


Рисунок 2.1 – Загальна схема системи за обраною темою

Пропонована система виявлення вторгнення використовує нечітку логіку. Вхідними даними є безліч KDD Сир 99, яке розділене на два підмножини: Навчальний набір даних і тестовий набір даних. Спочатку, навчальна вибірка даних класифікується в п'ять підмножин так, щоб виділити чотири типи атак: (DoS (відмова в обслуговуванні). R2L (віддалений до локального). U2R (користувач, з правами адміністратора). Probe), і нормальні дані. Далі з використанням ПЗ Weka виявляються найбільш важливі атрибути, які згодом використовуються для складання певних і невизначених правил. Потім, відповідно до певного правила фаззифікації, генеруються нечіткі правила, таким чином, щоб отримати ряд продукційних правил виду «якщо ... то», які дозволять визначити, до якого класу належить запис – нормальний дані або ж це аномалія. Для ефективного навчання нечіткої системи, сформовані правила перераховані в базі нечітких продукційних правил. На етапі тестування, тестові дані зіставляються з нечіткими правилами з бази для того, щоб визначити, чи є дані нормальними або аномальними.

Побудова систем з нечітким висновком передбачає виконання наступних етапів:

- визначення входів і виходів створюваної системи;
- завдання для кожної з вхідних і вихідних змінних функції приналежності;
- розробка бази нечітких правил;
- вибір і реалізація алгоритму нечіткого логічного висновку;
- аналіз процесу управління створеної системи.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Проєктування моделей системи

Моделювання уніфікованої мови моделювання (UML) є критичною практикою при розробці додатків з кількох причин. UML забезпечує стандартизований спосіб візуалізації дизайну системи, що дозволяє покращити взаємодію, розуміння та документування протягом усього життєвого циклу розробки програмного забезпечення [22]. Ось кілька ключових причин, чому моделювання UML настільки важливе:

UML надає універсальну мову для розробників, дизайнерів та зацікавлених сторін, гарантуючи, що всі матимуть загальне уявлення про архітектуру та поведінку системи.

Візуальні схеми полегшують розуміння та обговорення складних систем, сприяючи покращенню спілкування між членами команди та зацікавленими сторонами, які не мають технічного значення.

Візуалізація системи – діаграми UML допомагають візуалізувати загальну структуру та компоненти системи, що допомагає виявити потенційні проблеми та розробити більш надійні архітектури [22].

Раннє виявлення проблем – шляхом моделювання системи на ранній стадії процесу розробки можна виявити і усунути потенційні недоліки конструкції, невідповідності і області, що вимагають поліпшення, до початку впровадження.

Детальна документація – UML-діаграми служать всеосяжною документацією по системі, яка має неоціненне значення для подальшого обслуговування і адаптації нових членів команди.

Історична довідка – вони надають історичну довідку про дизайн та еволюцію системи, на яку можна посилається при розробці майбутніх проєктів або вдосконалень [22].

Фіксація вимог: діаграми прицедентів та інші моделі UML допомагають фіксувати та уточнювати системні вимоги, гарантуючи, що всі функціональні та нефункціональні вимоги зрозумілі та узгоджені.

Узгодження із зацікавленими сторонами: вони допомагають привести очікування зацікавлених сторін у відповідність з реальними можливостями системи, надаючи чітке уявлення про функціональні можливості і взаємодії системи.

Вибір дизайну – моделі UML допомагають оцінити різні варіанти дизайну та їх наслідки, що дозволяє приймати більш обґрунтовані рішення.

Архітектурне планування – вони являють собою схему архітектури системи, допомагаючи ефективно планувати і організовувати завдання розробки [22].

Автоматизація – деякі інструменти UML підтримують автоматичну генерацію коду на основі UML-діаграм, що може прискорити процес розробки і зменшити кількість помилок при ручному кодуванні.

Зворотна інженерія – UML можна використовувати для створення моделей з існуючого коду, допомагаючи розуміти та документувати застарілі системи.

Діаграми UML можуть допомогти розбити проєкт на керовані завдання та компоненти, допомагаючи в плануванні та управлінні проєктом. Вони забезпечують візуальне представлення прогресу системи та поточного стану розвитку, що може бути корисним для відстеження та звітування [22].

Моделювання на мові UML відіграє життєво важливу роль у успішній розробці додатків, забезпечуючи чіткий, стандартизований та всеосяжний спосіб візуалізації, проєктування, документування та передачі різних аспектів системи. Впроваджуючи UML в процес розробки, команди можуть забезпечити кращу узгодженість, більш високу якість і більш структурований підхід до створення складних програмних систем [22].

На рис. 3.1 наведено діаграму класів для системи, що розробляється.

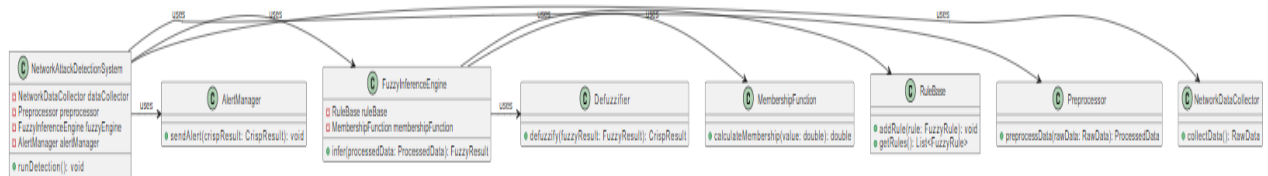


Рисунок 3.1 – Діаграма класів системи

Опис класів:

- **networkattackdetectionsystem**: це центральний клас, який координує весь процес виявлення. Він використовує інші компоненти, такі як Збирач даних, препроцесор, нечіткий вихідний механізм та менеджер сповіщень, для виявлення та реагування на мережеві атаки;
- **networkdatacollector**: відповідає за збір необроблених даних про мережевий трафік, які потім передаються в препроцесор;
- **препроцесор**: отримує необроблені дані з **NetworkDataCollector** і обробляє їх у форматі, придатному для нечіткого виводу. Це може включати в себе нормалізацію даних, Витяг функцій і т.д.;
- **fuzzyinferenceengine**: основний компонент, який застосовує нечітку логіку до оброблених даних для виявлення потенційних атак. Він використовує базу правил і функції приналежності для виконання логічного висновку;
- **база правил**: зберігає нечіткі правила, які визначають, як система повинна визначати потенційні атаки на основі вхідних даних;
- **Функція членства**: визначає, як обчислюється ступінь членства для різних нечітких множин. Це має вирішальне значення для процесу нечіткого виведення;
- **дефазифікатор**: перетворює нечіткі результати, що генеруються механізмом логічного висновку, у чіткі результати, які можна використовувати для прийняття рішень або запуску сповіщень;

– alertmanager – відповідає за надсилання сповіщень або сповіщень при виявленні потенційної атаки. Це може включати реєстрацію сповіщення, надсилання електронного листа або запуск іншого механізму реагування.

На цій діаграмі представлений загальний огляд архітектури та основних компонентів системи виявлення мережових атак на основі нечіткого логічного висновку. Кожен клас і взаємозв'язок можуть бути додатково деталізовані в залежності від конкретних вимог і деталей реалізації.

Діаграма прецедентів наведена на рис. 3.2.

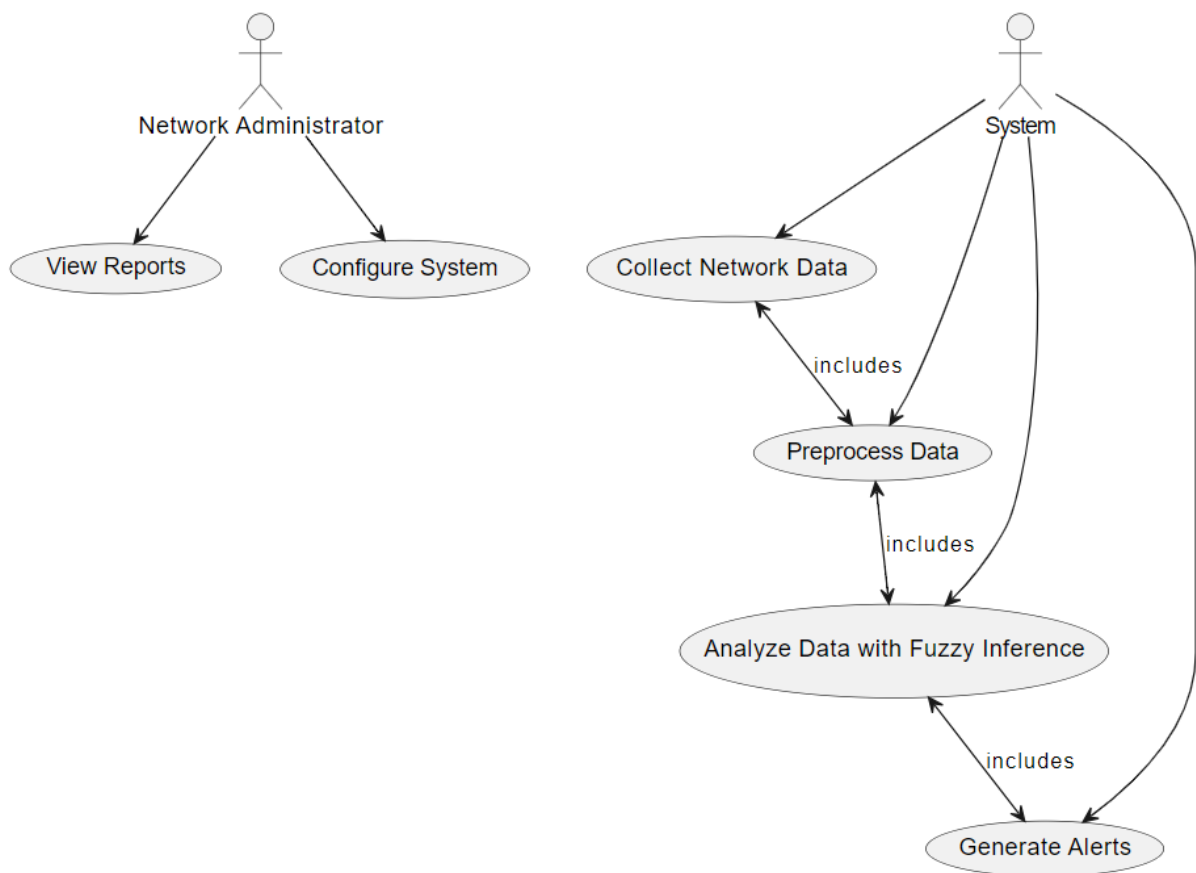


Рисунок 3.2 – Діаграма прецедентів розробленої системи

На вхід алгоритму подаються кількісні значення, на виході формуються також Числові значення, на проміжних етапах застосовується апарат нечіткої логіки. Це і є основна перевага нечітких систем-можливість роботи зі звичними числовими даними і при цьому використовувати можливості, що надаються системами нечіткого виведення.

На рис.3.3 представлені послідовно виконуються етапи алгоритму.



Рисунок 3.3 – Основні етапи нечіткого виведення

3.2 Класифікація навчальних даних

Перший компонент запропонованої системи відповідає за розділення вхідних даних на кілька класів, беручи до уваги різні атаки, включені до набору вихідних даних для виявлення вторгнень. Набором даних для аналізу поведінки системи виявлення вторгнень є безліч KDD-Cup 99. Дані KDD-Cup 99 складаються з чотирьох різних типів атак і нормальних даних, кожен із записів має 41 параметр, які можуть бути представлені, як безперервними, так

і символічними значеннями. Потім, набір даних (D) ділиться на дві підмножини на основі мітки класу, встановленої у вихідній множині $D = \{D_i; 1 \leq i \leq 2\}$. Мітка класу описує приналежність запису до однієї з чотирьох основних атак (відмова в обслуговуванні, R2L, U2R і Probe) або до множини нормальних даних. У розробленій системі навчальні дані розбираються на два класи: *normal* і *anomaly*. Потім ці два підмножини даних використовуються для генерації дерева виводу, яке згодом буде використано в якості основи нечітких правил, так щоб нечітка система могла ефективно застосовувати правила.

При тестуванні системи використовуються різні варіанти класифікації тестових даних: також розбиття вихідного безлічі на два підмножини *normal* і *anomaly*, а інший варіант - це розбиття вхідних даних на п'ять підмножин, відповідно до міткою класу атаки. Такий підхід дозволить проаналізувати ефективність поставлених нечітких продукційних правил як для кожного з типів атак, так і для більш загального випадку.

3.3 Стратегія формування нечітких правил

Зазвичай, нечіткі правила, що використовуються в нечіткій системі, складаються вручну або із залученням експертів, які формулюють їх на основі емпіричних даних з аналізу вторгнень или власних знань. У випадку з ТСП-трафіком дуже трудомістко складати нечіткі правила вручну через те, що обсяг вхідних даних величезний і необхідно брати до уваги багато параметрів, а також НХ всілякі комбінації. На жаль, зараз доступно небагато досліджень з прикладами автоматичної генерацією нечітких продукційних правил. Керуючись цим фактом, необхідно визначити метод вибору найкращого набору правил.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

ІІІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ІІІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація системи виявлення мережових атак на основі нечіткого виведення.

Система має реалізовувати наступні функціональні вимоги:

- збір та реєстрація даних;
- попереднє оброблення;
- витяг характеристик;
- розробка системи нечіткого виведення;
- виявлення атаки;
- генерація попереджень та створення звітів.

4.2 Вимоги до інтерфейсу

Наявність інформаційної панелі забезпечує зручну подачу для моніторингу активності мережі, перегляду сповіщень та доступу до звітів.

Впровадження інструментів візуалізації для графічного представлення мережевого трафіку та виявлених атак, що допомагає швидко аналізувати та приймати рішення.

Управління сповіщеннями: пропонує інтерфейс для управління сповіщеннями, включаючи.

4.3 Умови виконання програми

Апаратні вимоги:

- IBM-сумісний персональний комп'ютер;
- центральний процесор з тактовою частотою від 2 GHz;
- оперативна пам'ять формату DDR4, об'ємом від 3 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 1 Гб;
- підключення до мережі Інтернет.

Програмні вимоги:

- розгорнута операційна система (ОС), наприклад: Windows, Linux або macOS;
- наявність веббраузера актуальної версії.

4.4 Методика та результати тестування

В даному підрозділі описується система нечіткої логіки (рис. 4.1) для знаходження підходящої мітки класу тестового набору даних. Лотфі Заде наприкінці 1960-х [15] ввів поняття нечіткої логіки.

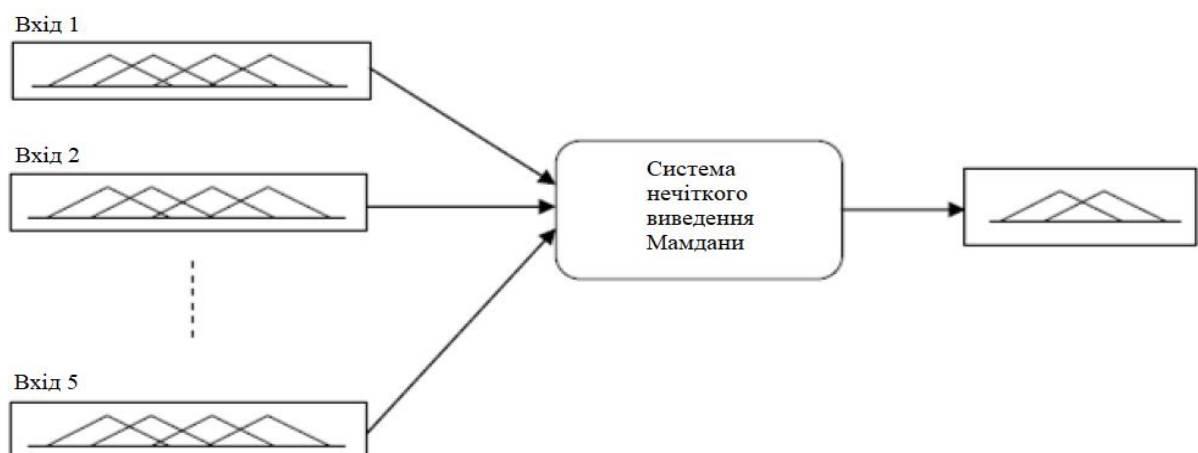
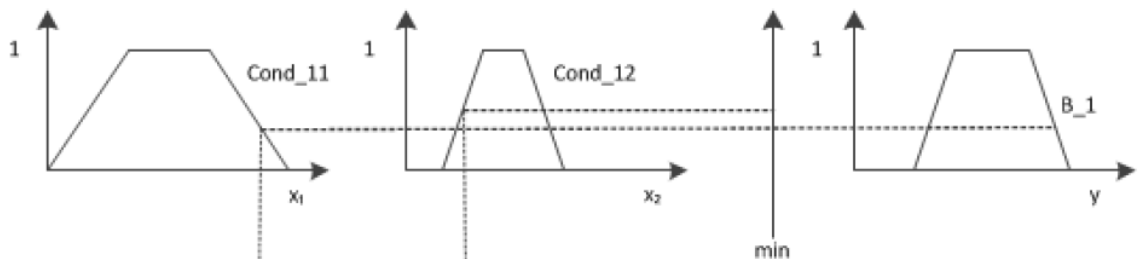


Рисунок 4.1 – Система нечіткого виведення

Нечітка система, зображена на рис. 4.1 містить п'ять входів і один вихід, де входи пов'язані з п'ятьма атрибутами, а вихід, пов'язаний з міткою класу (атака нлн нормальні дані). Застосовується система нечіткого виведення Мамдані з дефазіфікації за методом – центру ваги (рис. 4.2). Тут, кожна вхідна нечітка множина, визначена в нечіткій системі, включає кілька функцій належності (залежно від різноманітності даних), а вихідна нечітка множина містить дві функції належності.

Rule 1:



Rule 2:

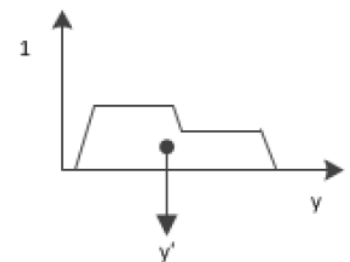
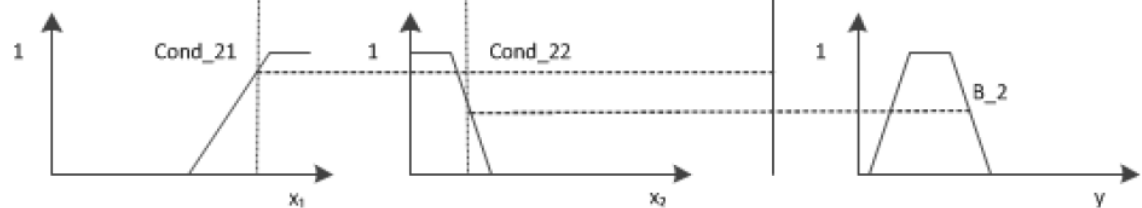


Рисунок 4.2 – Приклад нечіткого висновку Мамдані

Для фаззифікації (розмивання) даних в системі застосовуються функції приналежності різних видів: трикутні, трапецієподібні і лінійні (вибір форми функції залежить від типу і специфіки даних, а також від діапазону значень, який необхідно покрити). Нечіткі правила, отримані на етапі,

описаному в підрозділі 3.2 використовуються в якості бази нечітких продукційних правил.

4.5 Результати роботи алгоритму

Для тестування, на вхід системі нечіткої логіки, описаної в розділі 4.4, подаються тестові дані з безлічі KDD-cup 99. По-перше, тестові вхідні дані подаються на вхід фаззифікатору, який перетворює 5 атрибутів (Числові змінні) в лінгвістичні змінні, використовуючи відповідну йому функцію приналежності. Вихідні дані від фаззифікатора подаються на вхід механізму логічного виведення, який, в свою чергу порівнює цей конкретний вхід з базою правил. Виходом логічного висновку є одне зі значень лінгвістичної змінної з наступної множини {Low and High}, потім, дефаззифікатор перетворюється його до чіткого значення. Чітке значення, отримане від механізму нечіткого виведення, змінюється в діапазоні від 0 до 1. де «0» означає, що дані є абсолютно нормальними, а «1» вказує на те, що це однозначно атака. Класи атак User to Root і Remote to Local при оцінці результатів роботи системи не розглядалися, тому що кількість записів для них в тестовому безлічі дуже мало (близько 30-100 шт). В силу малої кількості записів для даних класів рекомендується створювати окремі підмножини правил. Тестування системи проводилося на випадкових наборах записів з кроком в 300 штук (від 300 до 1500).

У таблиці 4.1 представлені узагальнені дані для кожного з класів.

Таблиця 4.1 – Зведена таблиця результатів тестування системи

Клас	Точність, %	Хибні відпрацювання, %
Normal	93,47	2,83
DoS	92,09	7,56
Probes	96,2	0,23

На рис. 4.3 діаграма ілюструє кількість вірно класифікованих записів.

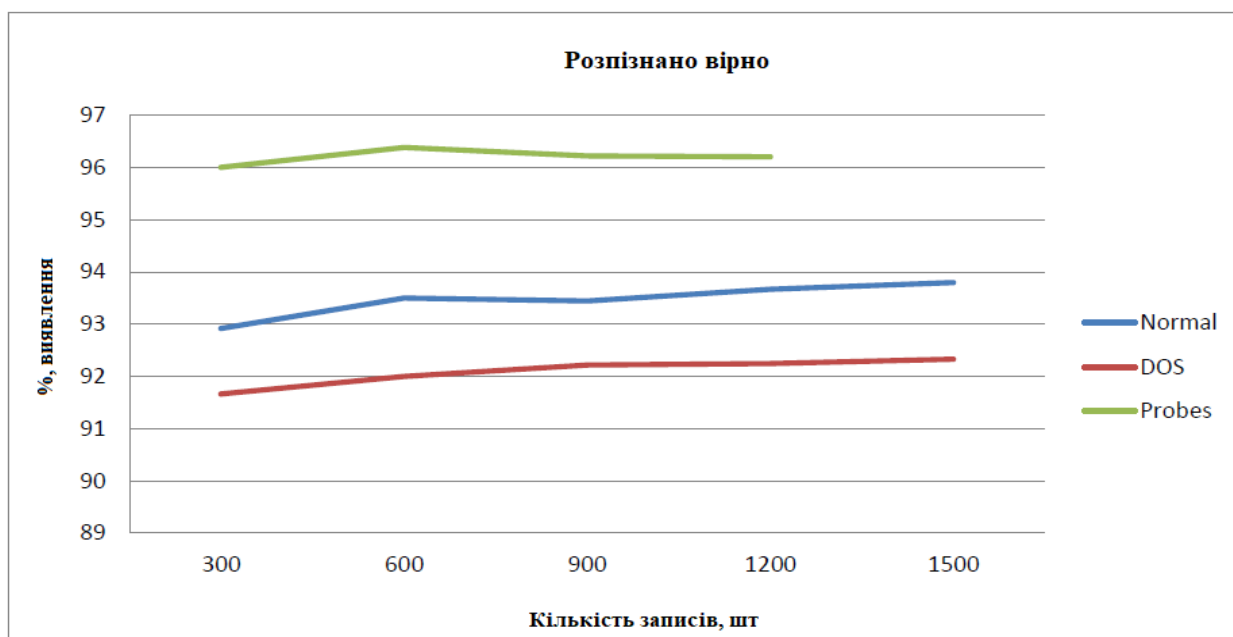


Рисунок 4.3 – Діаграма розпізнавання різних класів записів на тестовій вибірці

Рис. 4.4 ілюструє відсоток помилкових спрацьовувань для кожного з класів.

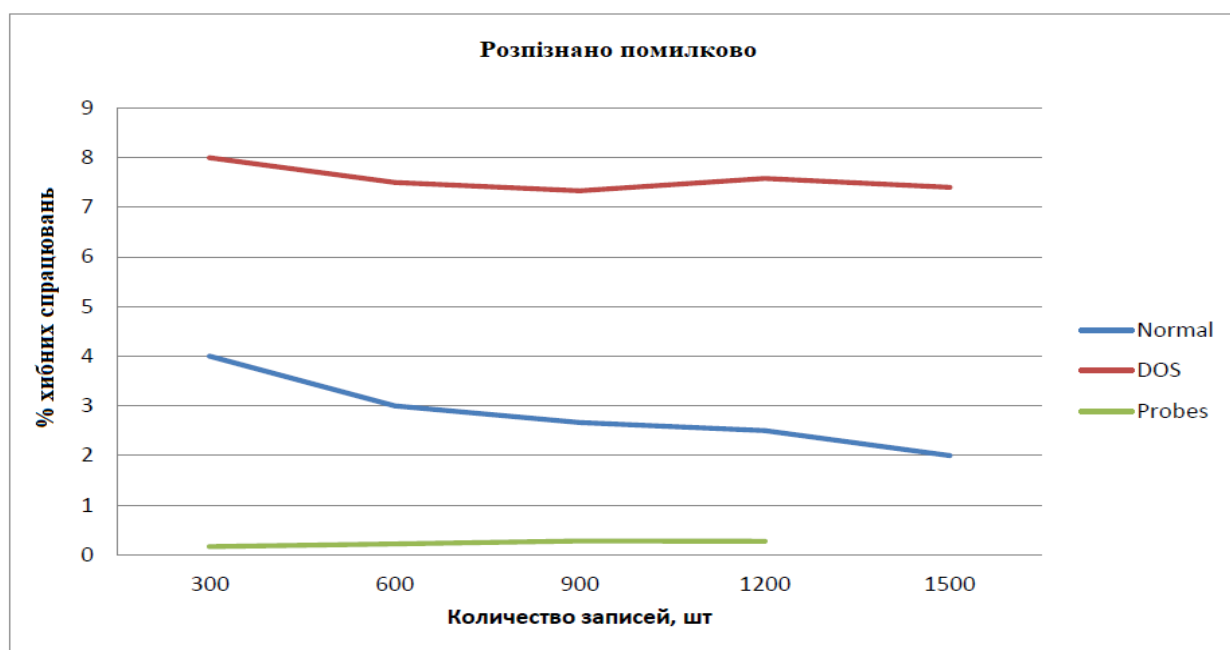


Рисунок 4.4 – Діаграма помилково класифікованих записів для кожного з класів на тестовій вибірці

На рис. 4.5 представлено процентне співвідношення числа записів, для яких в базі нечітких продукційних правил не було знайдено жодного відповідного. Відповідно такі записи не були віднесені до жодного з двох класів атак або до нормального трафіку.

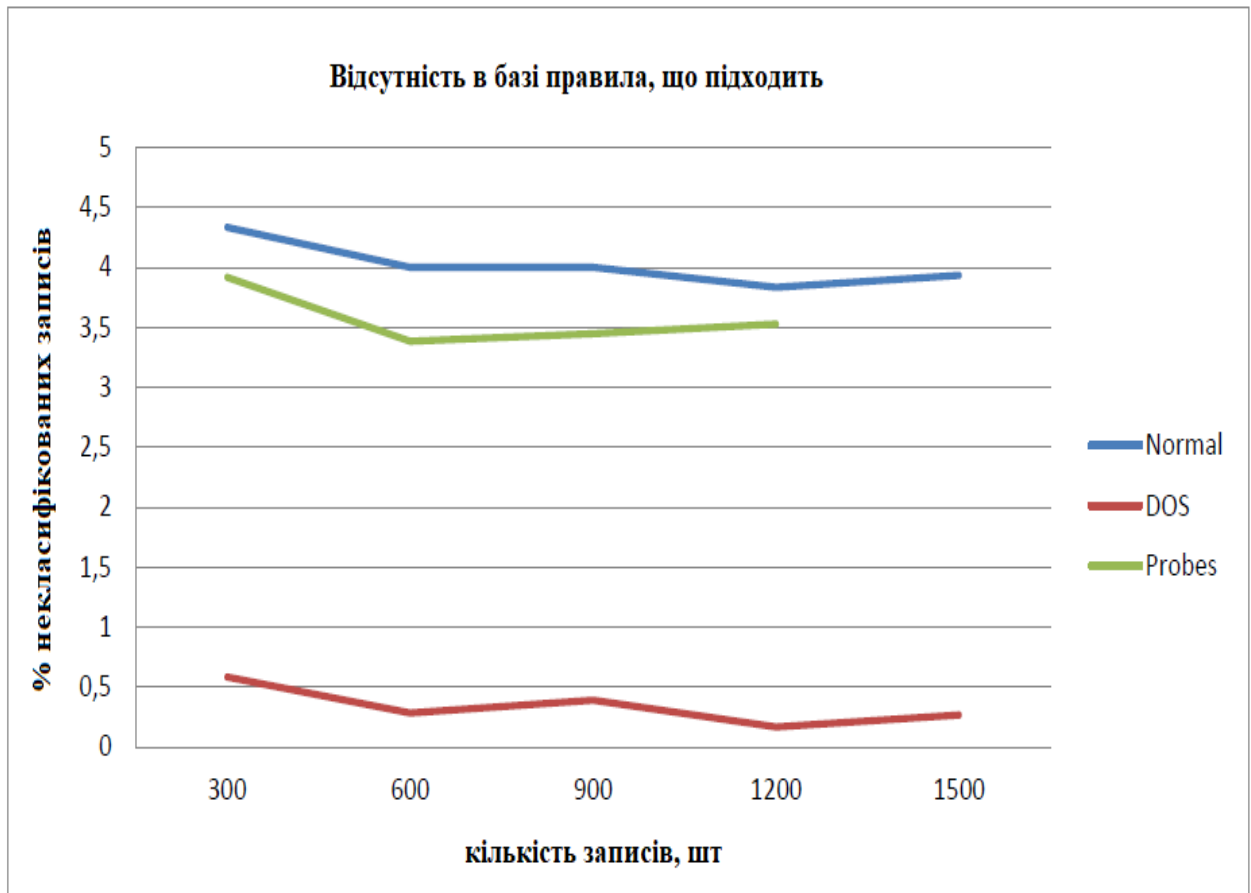


Рисунок 4.5 – Діаграма відображає відсоток записів тестової вибірки не потрапляють ні під одне правило з бази правил

Для оцінки результатів роботи прототипу були проаналізовані результати інших дослідників, які також використовували безліч KDD-cup 99 для навчання і тестування СВВ. У роботах [2] також розглядається можливість застосування нечіткої логіки для виявлення мережових атак. Для складання бази нечітких правил в [2] використовувався генетичний алгоритм.

Порівняння результатів наведено у табл. 4.2.

Таблиця 4.2 – Порівняння результатів

Клас	Розроблений прототип	Мережа з генетичним методом	Мережа з генетичним алгоритмом	Мережа з опорними векторами
Normal	93.47 %	92,78 %	69.5 %	90,3 %
DoS	92.09 %	98,91 %	99.4 %	94,92 %
Probes	96.2 %	50,35 %	71.1 %	90,9 %

4.6 Висновки за отриманими результатами

На основі даних, представлених в розділі можна зробити висновок про те, що застосування нечіткої логіки в системах виявлення мережевих атак досить ефективно. Отримані в результаті експериментів дані доводять це твердження. Розроблений прототип показує результати не гірше, а для більшості класів атак - краще, ніж існуючі аналоги.

З рисунків 4.3–4.5 видно, що зі збільшенням числа записів у вибірці зростає відсоток вірно розпізнаних класів, зменшується кількість помилкових спрацьовувань, а також зменшується, або ж, в решті решт, не зростає відсоток записів, для яких в базі правил не існує жодного правила, на основі якого можна було б віднести запис до того чи іншого класу. Однак в деяких випадках спостерігаються зміни в характері графіка, що можна інтерпретувати неоднорідністю розподілу прикладів в тестовому безлічі. Цей факт може бути обумовлений двома причинами: нерівномірний розподіл тестових записів по класах і ситуаціями, в яких не було активовано жодне правило з бази.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було виконано програмну реалізацію системи виявлення мережових атак на основі нечіткого виведення. Для цього було виконано ряд завдань:

- досліджено проблемну область кіберзахисту від мережових атак;
- досліджено існуючі рішення для проблемної області кіберзахисту від мережових атак;
- спираючись на обґрунтований вибір технічних засобів виконано програмну реалізацію системи виявлення мережових атак на основі нечіткого виведення;
- проведено тестування розробки;
- сформулювати висновки на основі виконаної роботи.

Розроблений застосунок був реалізований із використанням: мови програмування Java та вибірки KDD-cup 99 , засоби моделювання UML.

Застосунок було протестовано відповідно до сценарію його використання. Тестування засвідчило коректність роботи.

Застосунок реалізує такі функціональні вимоги:

- збір та реєстрація даних;
- попереднє оброблення;
- витяг характеристик;
- розробка системи нечіткого виведення;
- виявлення атаки;
- генерація попереджень та створення звітів.

В результаті розробки реалізовано систему, що відповідає всім вимогам визначеним на початку розробки у вигляді функціональних вимог. В майбутньому можливо вдосконалити систему додавши реалізацію еквалайзера для налаштувань відтворення композицій. Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. What is an Intrusion Detection System [Electronic resource]. – Access mode: <https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids>.
2. What is Attack Detection System [Electronic resource]. – Access mode: <https://www.igi-global.com/dictionary/attack-detection-system/84478>.
3. Network Based Intrusion Detection System [Electronic resource]. – Access mode: <https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system>.
4. Top 6 Free Network Intrusion Detection Systems (NIDS) Software [Electronic resource]. – Access mode: <https://www.upguard.com/blog/top-free-network-based-intrusion-detection-systems-ids-for-the-enterprise>.
5. Detecting and preventing attacks using network intrusion detection systems [Electronic resource]. – Access mode: https://www.researchgate.net/publication/41845843_Detecting_and_preventing_attacks_using_network_intrusion_detection_systems.
6. AI-Driven Bot Protection. Block Bots Effortlessly. Learning Center [Electronic resource]. – Access mode: <https://netacea.com/learn/>.
7. The world's most popular network protocol analyzer [Electronic resource]. – Access mode: <https://www.wireshark.org/>.
8. Wireshark [Electronic resource]. – Access mode: <https://dou.ua/forums/topic/44274/>.
9. Snort [Electronic resource]. – Access mode: <https://www.snort.org/>.
10. SNORT—Network Intrusion Detection and Prevention System [Electronic resource]. – Access mode: <https://www.fortinet.com/resources/cyberglossary/snort>.
11. SolarWinds Network Performance Monitor [Electronic resource]. – Access mode: <https://www.solarwinds.com/network-performance-monitor>.

12. SolarWinds Network Performance Monitor (NPM) [Electronic resource]. – Access mode: <https://softprom.com/vendor/solarwinds/product/solarwinds-network-performance-monitor-npm>.

13. How To Develop an App in 15 Easy Steps @ 2024 Guide [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/how-develop-app-15-easy-steps-2024-guide-ankit-sachan-wpx6c>.

14. How To Develop an App in 12 Easy Steps (2024 Guide) [Electronic resource]. – Access mode: <https://www.upwork.com/resources/how-to-develop-an-app>.

15. How to create an app in 2024 [Electronic resource]. – Access mode: <https://www.goodbarber.com/blog/how-to-make-an-app/>.

16. Java? [Electronic resource]. – Access mode: <https://aws.amazon.com/ru/what-is/java/>.

17. Learn Java Programming [Electronic resource]. – Access mode: <https://www.programiz.com/java-programming>.

18. IDE for Java [Electronic resource]. – Access mode: <https://foxminded.ua/ide-dlja-java/>.

19. Eclipse, NetBeans, IntelliJ IDEA [Electronic resource]. – Access mode: <https://javarush.com/ua/groups/posts/4027-eclipse-netbeans-ili-intellij-idea-vihbirae-m-ide-dlja-java-razrabotki>.

20. Fuzzy Logic For Beginners [Electronic resource]. – Access mode: <https://pzs.dstu.dp.ua/logic/bibl/masao.pdf>.

21. Fuzzy Logic: Definition, Meaning, Examples, and History [Electronic resource]. – Access mode: <https://www.investopedia.com/terms/f/fuzzy-logic.asp>.

22. UML [Electronic resource]. – Access mode: <https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-basics.html>.

ДОДАТОК А
Технічне завдання

Вступ

Даний застосунок присвячений для впорядкування домашньої музичної колекції (бібліотеки) – фонотеки.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему Програмна реалізація системи виявлення мережових атак на основі нечіткого виведення, затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

A.2 Основні вимоги до програми, що розробляється

ІІІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ІІІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація системи виявлення мережових атак на основі нечіткого виведення.

A.2.1 Вимоги до функціональних характеристик

Система має реалізовувати наступні функціональні вимоги:

- збір та реєстрація даних (моніторинг мережевого трафіку – система повинна постійно відстежувати та збирати дані про мережевий трафік

у режимі реального часу з кількох джерел (наприклад, маршрутизаторів, комутаторів, брандмауерів); управління журналами – збір журналів з різних мережевих пристроїв і додатків для забезпечення всебічного контролю за мережевою активністю);

- попереднє оброблення (очищення даних – видалення шуму та непотрібної інформації із зібраних даних для забезпечення високоякісного введення для аналізу; нормалізація даних – приведення даних до стандартної шкали для полегшення точних порівнянь та аналізу);

- витяг характеристик (витяг характеристик трафіку – витягніть відповідні характеристики з мережевого трафіку, такі як розмір пакета, тип протоколу, IP-адреси джерела та призначення та номери портів; витяг поведінкових характеристик – витягніть поведінкові характеристики, такі як тривалість з'єднання, частота підключень і незвичайні схеми трафіку);

- розробка системи нечіткого виведення (розробка бази правил – розробка повного набору нечітких правил, заснованого на експертних знаннях, для визначення нормальної та ненормальної поведінки мережі; функції приналежності – визначення нечітких функцій приналежності для кожної вхідної ознаки, щоб впоратися з невизначеністю і неточностями в даних; механізм логічного висновку – впровадити механізм логічного висновку, щоб застосувати нечіткі правила до вхідних даних та визначити ймовірність атаки);

- виявлення атаки (виявлення аномалій – визначати відхилення від нормальної поведінки за допомогою нечіткої логіки для виявлення потенційних мережевих атак; виявлення на основі підписів – порівняння мережевої активності із відомими сигнатурами атак, використовуючи нечіткий логічний висновок для ідентифікації конкретних типів атак; гібридне виявлення – комбінування методів виявлення на основі аномалій та підписів для комплексного виявлення атак);

- генерація попереджень та створення звітів (генерація попереджень – генерація попередження при виявленні потенційних атак, класифікуючи їх за ступенем серйозності (наприклад, низька, середня, висока); визначення

пріоритетності сповіщень – визначення пріоритетності сповіщень в залежності від серйозності і потенційного впливу виявленої атаки; звітність – створення детальних звітів з коротким описом виявлених атак, включаючи мітки часу, інформацію про джерело/одержувача та тип атаки).

A.2.2 Вимоги до інтерфейсу програми

Наявність інформаційної панелі забезпечує зручну подачу для моніторингу активності мережі, перегляду сповіщень та доступу до звітів.

Впровадження інструментів візуалізації для графічного представлення мережевого трафіку та виявлених атак, що допомагає швидко аналізувати та приймати рішення.

Управління сповіщеннями: пропонує інтерфейс для управління сповіщеннями, включаючи перегляд детальної інформації, підтвердження та закриття сповіщень.

A.2.3 Вимоги до надійності

Обов'язкове шифрування даних для захисту конфіденційної інформації.

Управління доступом – впровадження управління доступом на основі ролей, щоб обмежити доступ до системних функцій та даних на основі ролей користувачів.

A.2.4 Вимоги до складу та параметрів технічних засобів

Апаратні вимоги:

- IBM-сумісний персональний комп'ютер;
- центральний процесор з тактовою частотою від 2 GHz;

- оперативна пам'ять формату DDR4, об'ємом від 3 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 1 Гб;
- підключення до мережі Інтернет.

Програмні вимоги:

- розгорнута операційна система (ОС), наприклад: Windows, Linux або macOS;
- наявність веббраузера актуальної версії.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

ІІІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ІІІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація системи виявлення мережових атак на основі нечіткого виведення.

Б.2 Функціональне призначення

Система має реалізовувати наступні функціональні вимоги [13]–[15]:

- збір та реєстрація даних (моніторинг мережевого трафіку – система повинна постійно відстежувати та збирати дані про мережевий трафік у режимі реального часу з кількох джерел (наприклад, маршрутизаторів, комутаторів, брандмауерів); управління журналами – збір журналів з різних мережових пристроїв і додатків для забезпечення всебічного контролю за мережевою активністю);
- попереднє оброблення (очищення даних – видалення шуму та непотрібної інформації із зібраних даних для забезпечення високоякісного введення для аналізу; нормалізація даних – приведення даних до стандартної шкали для полегшення точних порівнянь та аналізу);
- витяг характеристик (витяг характеристик трафіку – витягніть відповідні характеристики з мережевого трафіку, такі як розмір пакета, тип протоколу, IP-адреси джерела та призначення та номери портів; витяг

поведінкових характеристик – витягніть поведінкові характеристики, такі як тривалість з'єднання, частота підключень і незвичайні схеми трафіку);

- розробка системи нечіткого виведення (розробка бази правил – розробка повного набору нечітких правил, заснованого на експертних знаннях, для визначення нормальної та ненормальної поведінки мережі; функції приналежності – визначення нечітких функцій приналежності для кожної вхідної ознаки, щоб впоратися з невизначеністю і неточностями в даних; механізм логічного висновку – впровадити механізм логічного висновку, щоб застосувати нечіткі правила до вхідних даних та визначити ймовірність атаки);

- виявлення атаки (виявлення аномалій – визначати відхилення від нормальної поведінки за допомогою нечіткої логіки для виявлення потенційних мережових атак; виявлення на основі підписів – порівняння мережової активності із відомими сигнатурами атак, використовуючи нечіткий логічний висновок для ідентифікації конкретних типів атак; гібридне виявлення – комбінування методів виявлення на основі аномалій та підписів для комплексного виявлення атак);

- генерація попереджень та створення звітів (генерація попереджень – генерація попередження при виявленні потенційних атак, класифікуючи їх за ступенем серйозності (наприклад, низька, середня, висока); визначення пріоритетності сповіщень – визначення пріоритетності сповіщень в залежності від серйозності і потенційного впливу виявленої атаки; звітність – створення детальних звітів з коротким описом виявлених атак, включаючи мітки часу, інформацію про джерело/одержувача та тип атаки).

Б.3 Опис логічної структури

Загальна схема системи за обраною темою на рис. Б.1.

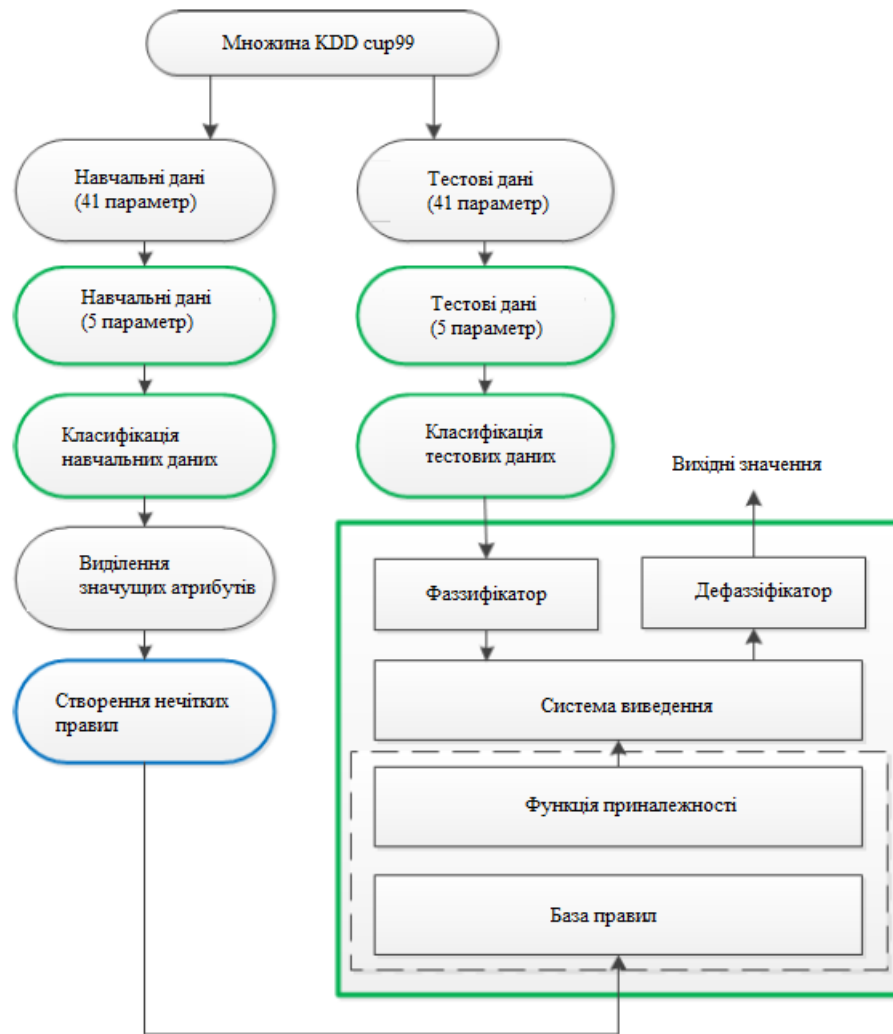


Рисунок Б.1 – Загальна схема системи за обраною темою

Пропонована система виявлення вторгнення використовує нечітку логіку. Вхідними даними є безліч KDD Cup 99, яке розділене на два підмножини: Навчальний набір даних і тестовий набір даних. Спочатку, навчальна вибірка даних класифікується в п'ять підмножин так, щоб виділити чотири типи атак: (DoS (відмова в обслуговуванні). R2L (віддалений до локального). U2R (користувач, з правами адміністратора). Probe), і нормальні дані. Далі з використанням ПЗ Weka виявляються найбільш важливі атрибути, які згодом використовуються для складання певних і невизначених правил. Потім, відповідно до певного правила фазифікації, генеруються нечіткі правила, таким чином, щоб отримати ряд продукційних правил виду «якщо ... то», які дозволяють визначити, до якого класу належить запис – нормальний

дані або ж це аномалія. Для ефективного навчання нечіткої системи, сформовані правила перераховані в базі нечітких продукційних правил. На етапі тестування, тестові дані зіставляються з нечіткими правилами з бази для того, щоб визначити, чи є дані нормальними або аномальними.

Побудова систем з нечітким висновком передбачає виконання наступних етапів:

- визначення входів і виходів створюваної системи;
- завдання для кожної з вхідних і вихідних змінних функції приналежності;
- розробка бази нечітких правил;
- вибір і реалізація алгоритму нечіткого логічного висновку;
- аналіз процесу управління створеної системи.

Б.4 Використані технічні засоби

Апаратні вимоги:

- IBM-сумісний персональний комп'ютер;
- центральний процесор з тактовою частотою від 2 GHz;
- оперативна пам'ять формату DDR4, об'ємом від 3 Гб;
- накопичувач (формату HDD або SSD) з вільним простором від 1 Гб;
- підключення до мережі Інтернет.

Програмні вимоги:

- розгорнута ОС, наприклад: Windows, Linux або macOS;
- наявність веббраузера актуальної версії.

ДОДАТОК В
Текст програми

```
package spendreport;

import java.util.Objects;

@SuppressWarnings("unused")
public final class Alarm
{
    public enum Level

    {

        private final int Val;

        private Level(int Val)
        {
            this.Val = Val;
        }

        public int getVal()
        {
            return Val;
        }
    }

    public Alarm(long id, String tstamp, String
attk, String comDant, Level level)
    {
        this.honeypotId = id;
        this.TMstat = tstamp;
        this.attk = attk;
    }
}
```

```
        this.command = comDant;
        this.level = level;
    }

    public long getHoneypotId()
    {
        return honeypotId;
    }

    public void setHoneypotId(long honeypotId)
    {
        this.honeypotId = honeypotId;
    }

    public String getTMstat()
    {
        return TMstat;
    }

    public void setTMstat(String TMstat)
    {
        this.TMstat = TMstat;
    }

    public String getattk()
    {
        return attk;
    }

    public void setattk(String attk)
```

```
{
    this.atk = atk;
}

public String getCommand()
{
    return command;
}

public void setCommand(String command)
{
    this.command = command;
}

public Level getLevel()
{
    return level;
}

public int getLevelVal()
{
    return level.getVal();
}

public String getLevelString()
{
    return level.name();
}

public void setLevel(Level level)
```

```

    {
        this.level = level;
    }

    @Override
    public boolean equals(Object o)
    {
        if (this == o) {
            return true;
        } else if (o == null || getClass() !=
o.getClass()) {
            return false;
        }
        Alarm that = (Alarm) o;
        return hnyPotInt == that.hnyPotInt
            && TMstat.equals(that.TMstat)
            && attk.equals(that.attk)
            && command.equals(that.command)
            && level == that.level;
    }

    @Override
    public int hashCode()
    {
        return Objects.hash(honeypotId, TMstat,
attk, command, level);
    }

    @Override
    public String toString()

```

```

        {
            return "Alarm{"
                + "hnyPotInt=" + hnyPotInt
                + ", TMstat=" + TMstat
                + ", attk=" + attk
                + ", command=" + command
                + ", level=" + level
                + '}';
        }
    }
}

package spendreport;

public final class IDSEngine
{
    private long hnyPotInt;
    private String log;
    private String tmShf d h;
    private String thife;
    private String cmRow;
    private Alert.Level level;

    public IDSEngine(long honeypotId, String log)
    {
        this.honeypotId = honeypotId;
        this.log = log;
        this.timestamp = null;
        this.attacker = null;
        this.cmRow = null;
        this.level = null;
    }
}

```

```

    public Alert processLog()
    {
        if (this.log.indexOf("] Command found: ")
!= -1)
            processCmd();
        else if (this.log.indexOf("] New
connection: ") != -1)
            processSSHConnect();
        else if (this.log.indexOf("] login attempt
[b'root'/b'kali'] succeeded") != -1)
            processRootLogin();
        else if (this.log.indexOf("] Connection
lost after") != -1)
            processSSHDisconnect();
        else
            return null;

        if (this.timestamp == null || this.thife ==
null || this.cmd == null || this.level == null)
            return null;

        Alert alert = new Alert(this.honeypotId,
this.timestamp, this.attacker, this.cmd, this.level);
        return alert;
    }

    private void processCmd()
    {

```

```

//                                     "2024-06-12T16:09:46+0000
[HoneyPotSSHTransport,9,172.18.0.5] Command found: ls"

        String pattern1 = new String("] Command
found: ");

        int idx1 = this.log.indexOf(pattern1);
        this.cmd = this.log.substring(idx1 +
pattern1.length());

        int idx2 = findAttacker(idx1, "
[HoneyPotSSHTransport,");
        if (idx2 == -1)
            return;

        findTimestamp(idx2);

        //set appropriate level of alert
        CommandsMap cmdMap = new CommandsMap();
        this.level = cmdMap.getCmdLevel(this.cmd);
    }

    private void processSSHConnect()
    {
        //                                     "2024-06-12T16:09:44+0000
[cowrie.ssh.factory.CowrieSSHFactory] New connection:
172.18.0.5:43516 (172.21.0.2:2222) [session:
1576158385b1]"

        String pattern1 = new String("] New
connection: ");

```

```

        String pattern2 = new String("
[session:");

        int IdX1 = this.log.indexOf(pattern1);
        int idx2 = this.log.indexOf(pattern2);
        if (idx2 < idx1)
            return;

        String substr = this.log.substring(idx1 +
pattern1.length(), idx2); //172.18.0.5:43516
(172.21.0.2:2222
        pattern2 = new String(" (");
        idx2 = substr.indexOf(pattern2);
        String targetHost = substr.substring(idx2 +
pattern2.length());
        this.cmd = "SSH connect to " + targetHost;
        this.attacker = substr.substring(0, idx2);

        pattern2 = new String("
[cowrie.ssh.factory.CowrieSSHFactory]");
        idx2 = this.log.indexOf(pattern2);

        findTimestamp(idx2);

        this.level = Alert.Level.HIGH;
    }

```

ДОДАТОК Д
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмна реалізація системи виявлення мережевих атак на основі нечіткого виведення

Виконав: Олександр ДОРОШЕНКО
ст. групи КНТ-111сп

Керівник: Сергій ЛЕОЩЕНКО
д-р філософ., ст. викладач

Рисунок Д.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- ОБ'ЄКТ ДОСЛІДЖЕННЯ – ПРОЦЕС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ.
- ПРЕДМЕТ ДОСЛІДЖЕННЯ – ПРОГРАМНІ ЗАСОБИ ДЛЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ.
- МЕТОЮ РОБОТИ Є ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ.
- .

Рисунок Д.2 – Слайд 2

ПОРІВНЯННЯ ІСНУЮЧИХ АНАЛОГІВ

	Wireshark	Snort	SolarWinds Network Performance Monitor
Мета	Аналізатор мережевих протоколів	Система виявлення та запобігання вторгнень	Моніторинг та управління продуктивністю мережі
Ключові функції	– глибока перевірка пакетів; – аналіз в режимі реального часу і в автономному режимі; – підтримка широкого спектру протоколів; – налаштовані звіти; – кроссплатформна сумісність	– аналіз трафіку в режимі реального часу; – виявлення на основі сигнатур; – аналіз протоколів; – гнучкі моделі розгортання; – регулярні оновлення	– моніторинг мережі в режимі реального часу; – налаштовані оповіщення; – аналізатор трафіку NetFlow; – комплексні інформаційні панелі; – масштабованість
Варіанти використання	– виявлення DDoS-атак; – ідентифікація зондів; – моніторинг звичайного трафіку	– виявлення DDoS-атак; – ідентифікація зондів; – моніторинг звичайного трафіку	– виявлення DDoS-атак; – ідентифікація зондів; – моніторинг звичайного трафіку

3

Рисунок Д.3 – Слайд 3

ПОСТАНОВКА ЗАВДАННЯ

ШІ та МН трансформують інформаційну безпеку та захищеність від загроз, розширюючи можливості щодо виявлення, запобігання та реагування на загрози. Хоча вони дають значні переваги, вони також створюють проблеми, які потрібно вирішити. Майбутнє кібербезпеки все більше залежатиме від ШІ та МН, що вимагатиме постійних досліджень, співпраці та дотримання етичних норм. Використовуючи ці передові технології, організації зможуть краще захищати свої дані та системи в умовах постійно мінливого ландшафту загроз.

Отже актуальною задачею залишається програмна реалізація системи виявлення мережевих атак на основі нечіткого виведення

4

Рисунок Д.4 – Слайд 4

ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ

	Java	C++	Python
Кросплатформність	Високий рівень	Середній рівень	Високий рівень
Стандартні бібліотека	Надійні бібліотеки для мережевої взаємодії, паралелізму і т. д.	Великий, але може вимагати використання бібліотек сторонніх виробників	Велика стандартна бібліотека, безліч модулів сторонніх виробників
Продуктивність	Високий рівень	Високий рівень	Помірний рівень
Управління пам'яттю	Автоматичне збирання сміття	Ручне управління пам'яттю, можливі витoki пам'яті	Автоматичне керування пам'яттю за допомогою збирання сміття
Багатопоточність	Потужна вбудована підтримка	Потужна підтримка, але вимагає ретельного управління	GIL обмежує багатопотоковість, підходить для завдань, пов'язаних з введенням / виведенням
Функції безпеки	Вбудований менеджер безпеки, можливості ізоляованого середовища	Потрібні додаткові бібліотеки та ретельне кодування	Хороші бібліотеки, але менш безпечні за замовчуванням
Простота використання	Синтаксис досить детальний, але зрозумілий	Складний-вимагає детального розуміння низькорівневих операцій	Високий рівень-простий і читабельний синтаксис

5

Рисунок Д.5 – Слайд 5

ПОРІВНЯННЯ СЕРЕДОВИЩ РОЗРОБКИ

	Eclipse	NetBeans	IntelliJ IDEA
Підтримка платформи	Windows, macOS, Linux	Windows, macOS, Linux	Windows, macOS, Linux
Продуктивність	Помірна	Помірна	Висока
Простота використання	Помірна	Висока	Висока
Інтерфейс користувача	Настроюваний	Зрозумілий	Інтуїтивно зрозумілий
Завершення коду	Хороший рівень	Хороший рівень	Хороший рівень
Інструменти рефакторингу	Помірний рівень	Хороший рівень	Хороший рівень
Налагодження	Широкі можливості	Широкі можливості	Широкі можливості
Інтеграція з системою контролю версій	Підтримує Git, SVN, CVS і т. д.	Підтримує Git, SVN, Mercurial і т. д.	Підтримує з Git, SVN і т. д.
Плагіни та розширення	Велика екосистема плагінів	Хороший вибір плагінів	Велика екосистема плагінів
Вбудовані інструменти	Включає в себе безліч інструментів і уявлень	Включає кілька вбудованих інструментів	Включає повний набір вбудованих інструментів
Настройка	Легко налаштовується	Обмежені налаштування	Легко налаштовується
Спільнота та підтримка	Велика спільнота та документація	Хороша спільнота та офіційна підтримка	Велика активна спільнота та чудова підтримка

6

Рисунок Д.6 – Слайд 6

ЗАГАЛЬНА СХЕМА ЗАСТОСУНКУ

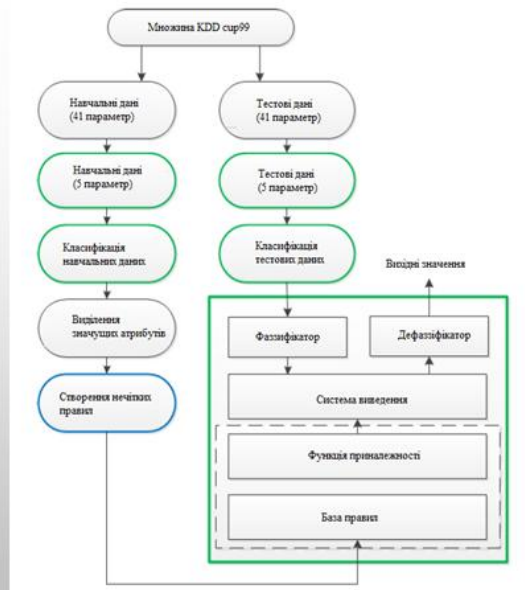


Рисунок Д.7 – Слайд 7

ДІАГРАМА КЛАСІВ

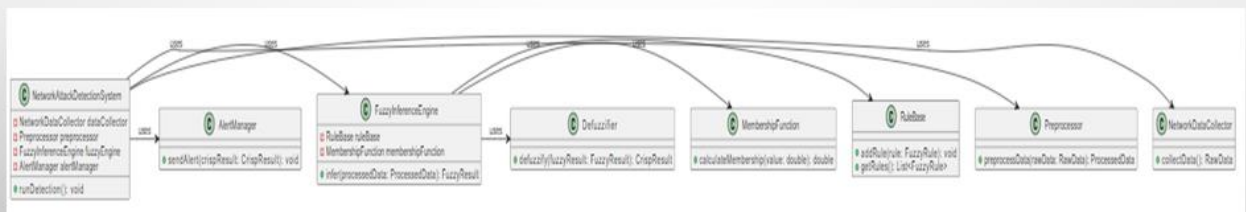


Рисунок Д.8 – Слайд 8

ДІАГРАМА ПРЕЦЕДЕНТІВ

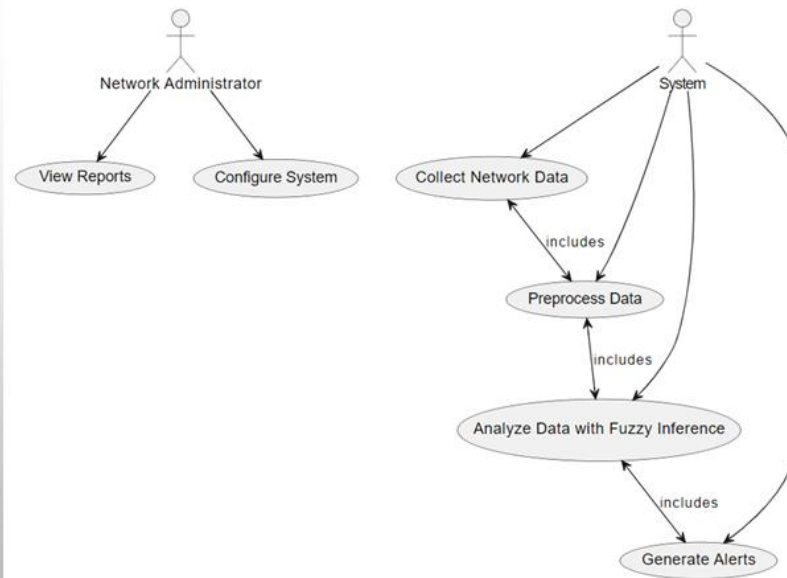


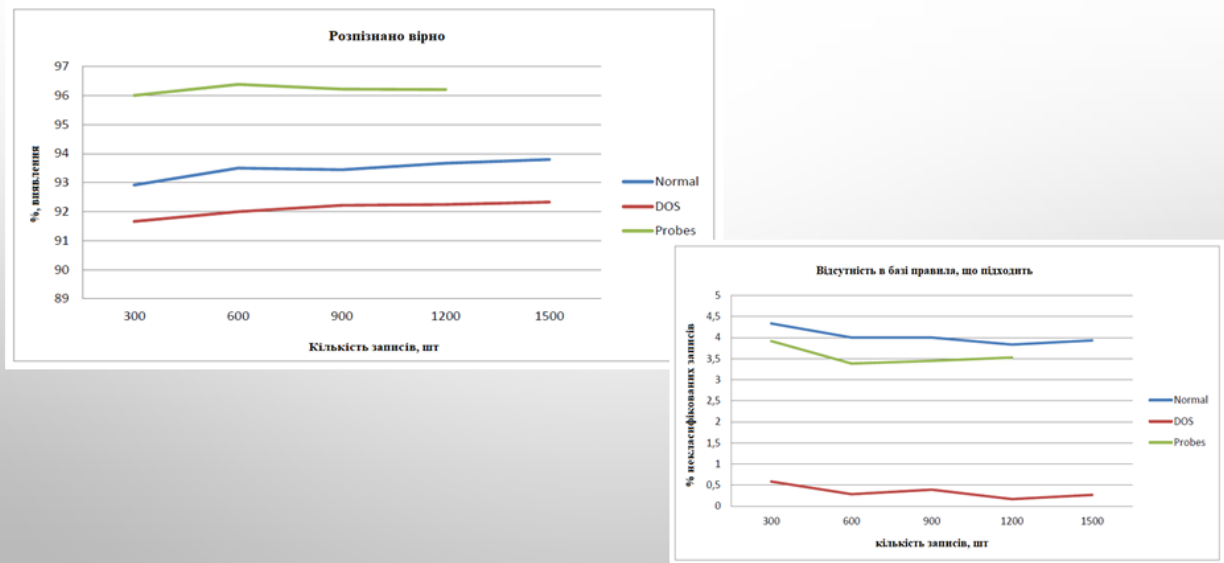
Рисунок Д.9 – Слайд 9

ОСНОВНІ ЕТАПИ НЕЧІТКОГО ВИВЕЛЕННЯ



Рисунок Д.10 – Слайд 10

РЕЗУЛЬТАТИ ТЕСТУВАННЯ



11

Рисунок Д.11 – Слайд 11

ВИСНОВКИ

В ХОДІ ВИКОНАННЯ ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА БУЛО ВИКОНАНО ПРОГРАМНУ РЕАЛІЗАЦІЮ СИСТЕМИ ВІЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ. ДЛЯ ЦЬОГО БУЛО ВИКОНАНО РЯД ЗАВДАНЬ:

- ДОСЛІДЖЕНО ПРОБЛЕМНУ ОБЛАСТЬ КІБЕРЗАХИСТУ ВІД МЕРЕЖЕВИХ АТАК;
- ДОСЛІДЖЕНО ІСНУЮЧІ РІШЕННЯ ДЛЯ ПРОБЛЕМНОЇ ОБЛАСТІ КІБЕРЗАХИСТУ ВІД МЕРЕЖЕВИХ АТАК;
- СПИРАЮЧИСЬ НА ОБҐРУНТОВАНИЙ ВИБІР ТЕХНІЧНИХ ЗАСОБІВ ВИКОНАНО ПРОГРАМНУ РЕАЛІЗАЦІЮ СИСТЕМИ ВІЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЧІТКОГО ВИВЕДЕННЯ;
- ПРОВЕДЕНО ТЕСТУВАННЯ РОЗРОБКИ;
- СФОРМУВАТИ ВИСНОВКИ НА ОСНОВІ ВИКОНАНОЇ РОБОТИ.

В РЕЗУЛЬТАТІ ВИКОНАННЯ ЗАВДАННЯ БУЛО ОТРИМАНО СИСТЕМУ, ЩО ВІДПОВІДАЄ ВСІМ ВИМОГАМ ВИЗНАЧЕНИМ НА ПОЧАТКУ РОЗРОБКИ ТА ЗАКРІПЛЕНИМ В ТЕХНІЧНОМУ ЗАВДАННІ. РОЗРОБЛЕНА СИСТЕМА Є ЗРУЧНОЮ ДЛЯ ВИКОРИСТАННЯ ТА ВИКОНУЄ ВСІ НЕОБХІДНІ ФУНКЦІЇ.

УСІ ЗАВДАННЯ ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА ПОВНІСТЮ ВИКОНАНО.

12

Рисунок Д.12 – Слайд 12