Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(освітній рівень)

на тему: *Розробка програмного забезпечення для виявлення DDOS атак*

Виконав: студент (ка)

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Головенко Б. В.
підпис (прізвище та ініціали)

Керівник *Карпінський М.П.*
підпис *(прізвище та ініціали)*

Нормоконтроль *Лобур Т.Б.*
підпис (прізвище та ініціали)

Завідувач кафедри *Загородна Н.В.*
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м.Тернопіль – 2022

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека
(шифр і назва спеціальності)

Студенту Головенку Богдану Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи " Розробка програмного забезпечення для виявлення DDOS атак»

Керівник роботи Карпінський Микола Петрович, д.т.н., професор кафедри КБ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 23 » 03 2022 року № 4/7-178

2. Термін подання студентом завершеної роботи 17.06.2022

3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

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

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Пулька Ч.В., проф. кафедри МТ		

7. Дата видачі завдання 16.02.2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	16.02 – 19.02	*Виконано*
2.	Підбір джерел по методах виявлення вразливостей для веб-додатків	20.02 – 27.02	*Виконано*
3.	Опрацювання джерел в галузі дослідження	28.02 – 16.03	*Виконано*
4.	Вибір програмних засобів розробки	17.03 – 20.03	*Виконано*
5.	Реалізація моделі і методу для сформування коду автентичності і контролю цілісності	20.03-05.04	*Виконано*
6.	Оформлення розділу «Аналіз предметної області»	06.03 – 17.04	*Виконано*
7.	Оформлення розділу «Проєктування програмного додатку для визначення DDOS атак»	18.04 – 29.04	*Виконано*
8.	Оформлення розділу «Тестування програмного додатку для визначення DDOS атак»	30.04 – 13.05	*Виконано*
9.	Виконання завдання до підрозділу «Безпека життєдіяльності, основи хорони праці»	14.05 – 21.05	*Виконано*
10.	Оформлення кваліфікаційної роботи	22.05 – 05.06	*Виконано*
11.	Нормоконтроль	06.06 – 12.06	*Виконано*
12.	Перевірка на плагіат	10.06 – 15.06	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.06 – 19.06	*Виконано*
14.	Захист кваліфікаційної роботи	23.06.2022	

Студент _____ Головенко Б.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінський М.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення для виявлення DDOS атак // Кваліфікаційна робота ОР «Бакалавр» // Головенко Богдан Віталійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-42 // Тернопіль, 2022 // С. , рис. – , табл. – , кресл. – , додат. – .

Ключові слова: КІБЕРБЕЗПЕКА, МЕТОДИ ВИЯВЛЕННЯ DDOS-АТАК, НЕЙРОННА МЕРЕЖА.

Об'єктом дослідження є процес аналізу атак на інформаційні системи.

Предметом дослідження є проєктування програмного забезпечення для виявлення DDOS атак.

Метою роботи є виявлення DDOS атак на основі програмного застосунку який забезпечує своєчасне повідомлення за рахунок використання нейронної мережі.

Розглянуто можливості реалізації кіберзагроз, їх класифікація. Виявлені можливості різних методів аналізу виявлення аномалій або відхилення від нормальної роботи елементів інфраструктури інформаційних систем. Розглянуті можливості навчання нейронних мереж щодо виявлення кібератак на основі DDOS-атак.

Розроблений проєкт з використанням нейронної мережі дозволяє поєднати кращі підходи щодо виявлення DDOS-атак.

ANNOTATION

Software Development to Identify any DDOS Attacks // Qualification thesis of educational level "Bachelor" // Holovenko Bohdan Vitaliiovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, СБс-42 group // Ternopil, 2022 // P. , fig. - , table. - , chair. - , added. - .

Keywords: CYBER SECURITY, DDOS ATTACK DECLARATION METHODS, NEURAL NETWORK.

The object of research is the process of analyzing attacks on information systems.

The subject of the study is the design of software for detecting DDOS attacks.

The aim of the work is to identify DDOS attacks based on a software application that provides timely messages through the use of a neural network.

The possibilities of implementing cyber threats, their classification are considered. The possibilities of different methods of analysis for detecting anomalies or deviations from the normal operation of information systems infrastructure elements are revealed. The possibilities of training neural networks to detect cyber-attacks based on DDOS attacks are considered.

The developed project using a neural network allows you to combine the best approaches to detecting DDOS attacks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

LAN – Local Area Network (локальна обчислювальна мережа).

MTM – Matrix Traceability Maturity – таблиця, яка використовується для відстеження вимог при життєвого циклу розробці ПЗ.

SQL Injection – спосіб зламу сайтів та програм, що працюють з базами даних, заснований на впровадженні в запит довільного SQL-коду.

XSS – Cross-Site Scripting (тип атаки на вебсистеми, що полягає у впровадженні у сторінку шкідливого коду, що видається вебсистемою).

WLAN – Wireless Local Area Network (локальна обчислювальна мережа на основі бездротового зв’язку).

ЖЗ – життєвий цикл.

МГОА – метод групового обліку аргументів.

ПЗ – програмні застосунки.

СВВ – системи виявлення вторгнень.

ВСТУП

Нині інформаційні технології проникли майже всі сфери життя сучасного суспільства, неодмінною частиною інформаційних технологій є мережа Інтернет. Причиною такого інтенсивного розвитку інформаційних технологій є зростальна потреба у швидкій та якісній обробці інформації, моментальної передачі у різні куточки світу. У зв'язку з цим, одним із головних завдань є забезпечення безпеки інформації, що передається або обробляється в мережі, захист від мережевих атак.

На цей час все більшого значення набувають комплексні системи захисту інформації. Як компоненти такої системи виступають системи антивірусного захисту, системи контролю цілісності, міжмережеві екрани, засоби аналізу вразливостей, системи виявлення та запобігання вторгненням тощо. Системи виявлення та запобігання вторгненням, або, як їх ще називають, засоби виявлення атак, є саме тим механізмом захисту мережі, на який покладено функції захисту від мережевих атак.

Існує велика кількість методів для визначення мережевих атак, але оскільки атаки постійно змінюються спеціальні бази даних правил або сигнатур для виявлення атак вимагають безперервного адміністрування, виникає необхідність додавати нові правила. Одним зі шляхів усунення даної проблеми є використання нейронних мереж як механізм для виявлення мережевих атак. Даний тип нейронних мереж проводить аналіз інформації на основі атак, які вона навчена розпізнавати і також надає інформацію про атаки, на відміну від сигнатурного підходу. Крім цього, нейронні мережі мають перевагу – вони здатні адаптуватися до раніше невідомих атак та виявляти їх. Саме тому розробка програмного забезпечення на основі нейронних мереж є актуальною.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд атак на інформаційну систему

Система виявлення вторгнень – це програма, яка використовується для моніторингу та захисту мережі від зловмисників [1–3]. Зі швидким прогресом технологій на основі Інтернету з'явилися нові сфери застосування комп'ютерних мереж [4–10]. У таких областях, як бізнес, фінанси, промисловість, у сфері безпеки та охорони здоров'я прогресували програми LAN та WAN. Усе це зробило мережу привабливою метою зловмисників [5]. Хакери використовують внутрішні системи організації для збору інформації та викликають уразливості, такі як помилки у програмному забезпеченні, помилки в адмініструванні, виявлення систем із налаштуваннями за умовчанням [6–8]. У міру появи Інтернету в суспільстві з'являються нові загрози, такі як віруси та хробаки. Зловмисники використовують різні методи, такі як зламування пароля, виявлення незашифрованого тексту для створення вразливостей у системі.

Використання веб-застосунків у сучасних інформаційних системах дозволяє використовувати критичні точці у веб-застосунках щодо зламу та/або проникнення до конфіденційної інформації. Тому Безпека веб-застосунків – одне з найгостріших питань у контексті інформаційної безпеки. Як правило, більшість веб-сайтів, доступних в Інтернеті, мають різного роду вразливість і постійно піддаються атакам. На рис. 1.1 наведені найпоширеніші проблеми безпеки веб-застосунків.

Існує багато типів недоліків ін'єкції: SQL Injection, XML-ін'єкція, Ін'єкція HTML, введення команд ОС, Ін'єкція LDAP [1–10].

Недоліки ін'єкції є результатом класичної помилки фільтрації ненадійного введення. Це може статися, коли ви передаєте нефільтровані дані на сервер SQL (ін'єкція SQL) приклад на рис. 1.2, у браузер (XSS), на сервер LDAP (ін'єкція LDAP) або де завгодно. Проблема полягає в тому, що зловмисник може вводити команди в ці об'єкти, що призводить до втрати даних і викрадення браузерів клієнтів.

Все, що ваша програма отримує з ненадійних джерел, має бути відфільтровано, бажано відповідно до білого списку. Ви майже ніколи не

повинні використовувати чорний список, оскільки отримати це право дуже важко і зазвичай легко обійти. Антивірусні програмні продукти зазвичай дають чудові приклади невдалих чорних списків. Зіставлення шаблону не працює.

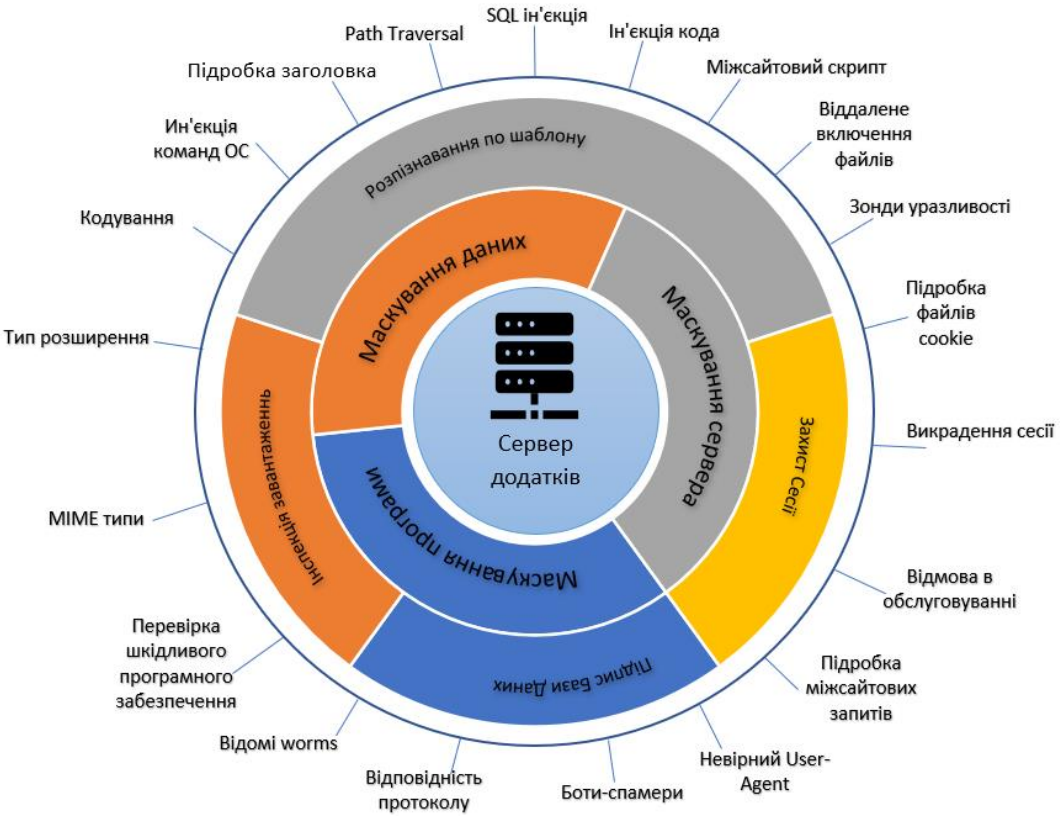

Рисунок 1.1 – Види веб-загроз

Запобігання полягає в тому, що захист від ін’єкції – це “просто” питання правильної фільтрації введених даних і роздумів про те, чи можна довіряти введеним даних. Але погана новина полягає в тому, що всі вхідні дані повинні бути належним чином відфільтровані, за винятком випадків, коли йому безперечно можна довіряти (але тут на думку спадає вислів “ніколи не кажи ніколи”.

Оскільки фільтрацію досить важко зробити правильно (наприклад, криптовалюту), треба покладатися на функції фільтрації framework. Тому потрібно ретельно підбирати фреймворк для розробки веб додатків. Таким чином використовуючи Angular framework захищаємо наш веб додаток від XSS

атак. Angular перед тим як вставити данні до шаблону фільтрує їх при правильному проєктуванні Angular додатку. Також для запобігання ін'єкція до БД у веб додатку використовуємо параметризовані запроси, які допомагають відфільтрувати вхідні данні.

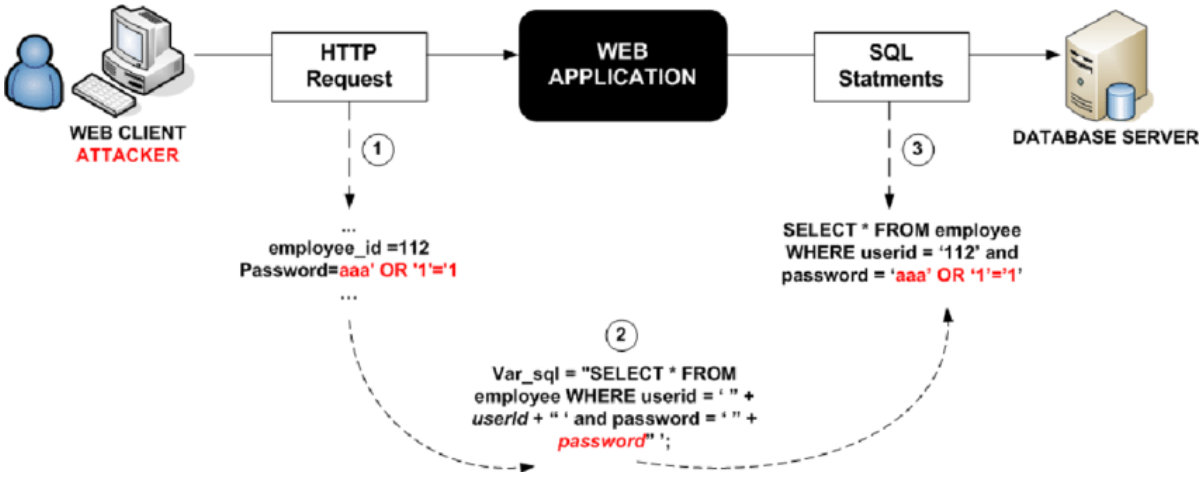

Рисунок 1.2 – Приклад SQL ін'єкції

Отже, необхідно підвищення безпеки мережі для захисту конфіденційної інформації в системі від зловмисників. Брандмауер є одним із найпопулярніших методів захисту, і вони використовуються для захисту приватної внутрішньої мережі компанії від несанкціонованого проникнення з публічної мережі (Інтернет). СОВ використовуються контролю роботи ЛОМ підприємств, медичних додатків, страхових агентствах тощо. [8].

Метою виявлення вторгнень є моніторинг мережевих ресурсів виявлення аномальної поведінки й неправильного використання мережі. Концепція виявлення вторгнень була введена на початку 1980-х років після розвитку інтернету, і пов'язана зі спостереженням та моніторингом загроз. З того часу технології СВВ удосконалили методи виявлення вторгнень. У роботі [9] описаний підхід, згідно з яким контрольні журнали містять важливу інформацію, яка може бути корисною для відстеження неправомірного використання та розуміння поведінки користувачів.

1.2 Аналіз проблем захисту інформації в інформаційних системах

Щоб узагальнити погляд на класифікацію загроз інформаційній безпеці, виділяють наступні характеристики: за джерелом - природне джерело, техногенне джерело, техногенне джерело; за передбачуваним ступенем пошкодження - загроза та небезпека; за повторюваністю діяти - повторювати і продовжувати; за походженням - екзогенні та ендогенні; можлива реалізація - можлива, неможлива, випадкова; за рівнем визначеності - регулярна і випадкова; за значенням - прийнятна і неприйнятна; за структурою впливу - систематична, структурна та елементарна; за характер реалізації - реальні, потенційні, досконалі, уявні; пов'язані з ними - об'єктивні та суб'єктивні; за об'єктами впливу - окремі особи, суспільства, нації. Але збільшення ознак не гарантує практичного розв’язання проблем підвищення інформаційної безпеки організацій, тому що не дозволяє синтезувати системну модель порушника інформаційної безпеки та локалізувати місця доступу до інформаційних ресурсів (ІР). А це, своєю чергою, визначити інформаційні ризики для здійснення оптимізації управління ними.

1.3 Опис структури та функції систем виявлення вторгнень

Системи аналізу захищеності досліджують налаштування елементів захисту операційних систем робочих станцій та серверів, аналізують топології мережі, шукають незахищені мережеві з’єднання, досліджують налаштування міжмережевих екранів. Дані системи дозволяють значно знизити ризик виявлених загроз у системі захисту локальних мереж.

До сучасних засобів моніторингу комп’ютерних атак відносяться аналізатори трафіку та системи виявлення атак проникнення.

Істотним недоліком даних систем є те, що аналіз трафіку адміністратором безпеки здійснюється практично вручну із застосуванням найпростіших засобів автоматизації, таких як аналіз протоколів. Розв’язання цієї проблеми є застосування засобів моніторингу, здатних аналізувати трафік великого обсягу у режимі реального часу. До таких засобів моніторингу належать системи виявлення вторгнень.

Системи виявлення вторгнень (СВВ) є окремим класом програмних за стосунків (ПЗ). Повна назва СВВ – це системи виявлення та запобігання атакам

вторгнення, тому що у можливості автоматизованої протидії атакам полягає одна з основних переваг таких систем, порівняно, наприклад, із засобами, заснованими на людському факторі.

Використання СВВ дає можливість вирішити багато завдань для досягнення цілей інформаційної безпеки.

СВВ використовує два підходи виявлення атаки вторгнення: зіставлення з шаблоном і евристичний аналіз. Кожен із цих підходів має свої переваги та недоліки. Фактичні сучасні СВВ часто поєднують два підходи.

В ідеалі, СВВ має бути швидкою, простою та точною. СВВ має виявляти всі атаки з невеликим зниженням продуктивності. СВВ може використовувати деякі з таких підходів:

- фільтр із заголовків пакетів;
- фільтр за вмістом пакета;
- підтримка стану з'єднання;
- використання складних, мультипакетних підписів;
- використання мінімальної кількості підписів з максимальним ефектом;
- фільтр у реальному часі, онлайн;
- приховування присутності.

Під виявленням атак розуміється процес оцінки подій IP та його інформаційних потоків. Ці потоки можуть бути реалізовані за допомогою аналізу реєстраційних журналів операційних систем (ОС) та додатків чи мережевого трафіку. Реалізація більшості мережевих атак здійснюється у три етапи.

На першому, підготовчому етапі здійснюється пошук ознак відомих атак. Під таким пошуком розуміється пошук вразливостей, які теоретично можуть призвести до реалізації тієї чи іншої атаки.

Другим етапом є безпосередньо реалізація атаки за виявленою вразливістю.

Під час третього етапу відбувається завершення атаки. Іноді першої та другі етапи об'єднуються в один. Наприклад, якщо хакер проводить сканування мережі з використанням сканерів безпеки, то це вже можна вважати атакою.

Технології виявлення атак постійно розвиваються та вдосконалюються, і ця область постійно приваблює нових виробників та розробників. Всупереч

нестачі теоретичних основ технології виявлення атак, є досить ефективні методи, які використовуються сьогодні.

Існує кілька способів класифікації систем виявлення атак, кожен із яких грунтується на різних характеристиках. Тип слід визначати за такими характеристиками:

Контроль системи. За методами контролю над системою діляться на network-based, host-based.

Аналізатор. Цей компонент СВВ відповідає за аналіз подій, що відбулися в мережі. На основі аналізу СВВ приймає рішення щодо можливості атаки проникнення. Для аналізу мережевих подій використовуються два способи:

- виявлення аномальних подій (anomaly detection);
- виявлення зловживань (misuse detection).

Затримки. Між отриманням даних із мережі, їх аналізом та прийняттям рішення може відбуватися якийсь час. Тому СВВ можна розділити на системи знаходження атак interval-based (пакетного режиму) і real-time системи.

Виявлення атак вимагає виконання однієї з двох умов: або знання всіх можливих атак та їх модифікацій, або розуміння очікуваної поведінки контрольованого об’єкту системи. Усі технології сьогодення, які виявляють мережеві атаки, можна розділити на два типи: методи, що базуються на основі сигнатур (зразків та правил); методи з урахуванням аномалій.

Важливим фактом знаходження атак на основі сигнатур є процес опису атаки у вигляді шаблону або сигнатури. Далі система виконує пошук та порівняння даних із контрольованого простору (наприклад, мережевого трафіку або журналу реєстрації) з сигнатурами в базі даних СВВ. Дана СВВ має можливість знаходити лише відомі атаки, при цьому така СВВ не здатна виявляти нові, ще невідомі атаки.

Під час розробки СВВ, формуються дві поширені проблеми. По-перше, це створення спеціального механізму, який дає можливість описувати шаблони (сигнатури) атак. По-друге, це опис усіх можливих модифікацій відомої атаки у вигляді сигнатур. Схема технології виявлення атак з урахуванням сигнатур наведено на рис. 1.3.

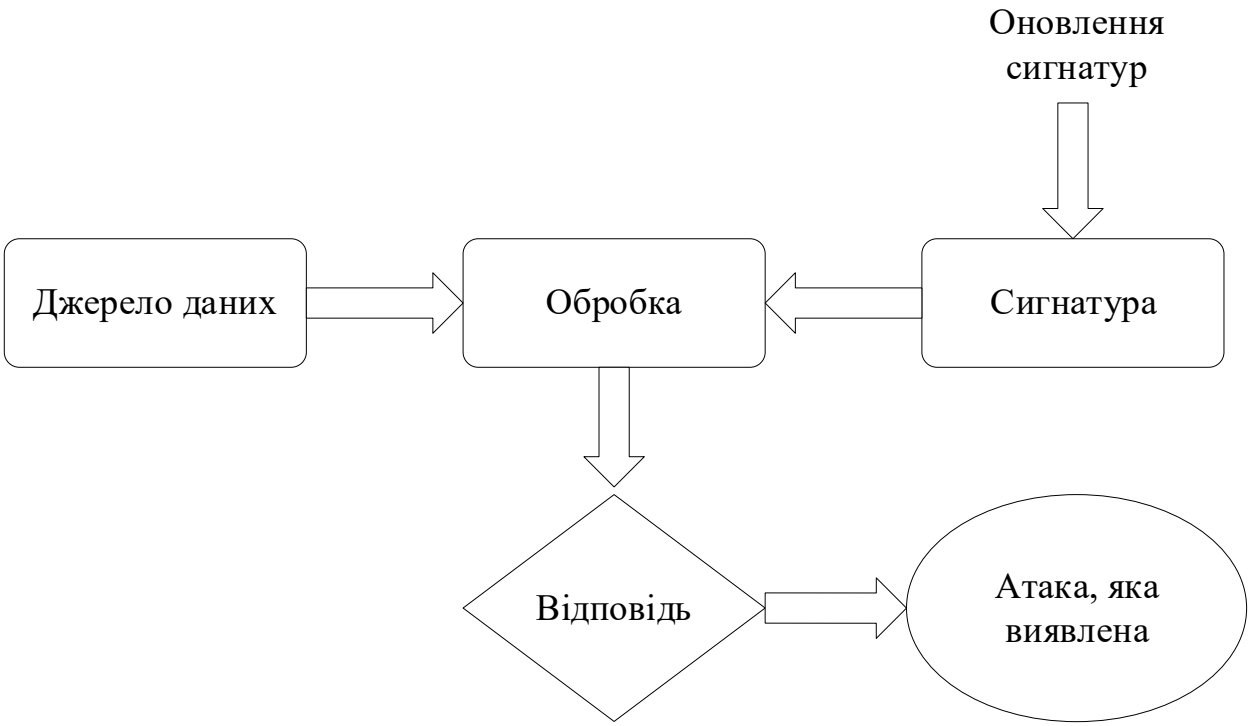

Рисунок 1.3 – Схема виявлення атак на основі сигнатур

Переваги:

– детектори зловживань мають високу ефективність визначення атаки і ймовірність створення помилкових повідомлень є дуже низькою;

– детектори зловживань дають змогу швидко та надійно діагностувати використання конкретного інструментального засобу чи технології атаки. Це дає змогу адміністратору відкорегувати заходи задля забезпечення безпеки;

– швидкість аналізу.

Недоліки:

– оскільки детектори зловживань виявляють лише відомі їм атаки, бази даних потребує постійного оновлення для отримання нових сигнатур атак;

– більшість детекторів зловживань розроблено так, що вони використовують обмежену кількість сигнатур і не дають змогу виявити різноманітні варіанти атак.

Знаходження на основі аномальних подій зазвичай має працювати зі статистично значущою кількістю пакетів, тому що будь-який пакет є лише аномалією у порівнянні з деяким базовим рівнем. Ця потреба у базовій лінії становить кілька труднощів. Наприклад, виявлення на основі аномалій не зможе виявити атаки, які можуть бути виконані декількома або навіть одним пакетом. Ці атаки, такі як ping of death, ще існують і набагато краще підходять

для виявлення на основі сигнатур. Додаткові проблеми виникають через те, що мережевий трафік зрештою залежить від поведінки людини. Хоча поведінка людини дещо передбачувана, вона, як правило, досить мінлива, щоб викликати проблеми з виявленням аномалій.

Зазвичай системи виявлення аномальної активності використовують як джерело даних журнали реєстрації та поточну діяльність користувача, хоча є приклади системи виявлення аномалій в мережевому трафіку.

Традиційне використання цієї технології полягає не в чіткому виявленні атак, а у визначенні підозрілої активності, що відрізняється від нормальної. Основна проблема методу у тому, щоб визначити критерій нормальної активності. Необхідно також встановити припустимі відхилення від нормального трафіку, які ще не можна вважати атакою.

При використанні даної технології виявлення атак можливі два варіанти неправильного виявлення атаки:

– виявлення дії, що не є атакою, та віднесення її до класу атак;

– пропуск атаки, яка не підпадає під сигнатури атак. Цей випадок небезпечніший, ніж помилкове віднесення дозволеної дії до класу атак. Підкатегорією такого методу є аналіз на основі профілів, коли нормальна поведінка визначається для окремих суб’єктів (користувачів/систем).

Іноді елементи такого аналізу зустрічаються і в інших методах, скажімо, в розшифровці протоколу, коли виявлений елемент не належить до певного протоколу або порушує правила використання протоколів.

Схема типової системи виявлення аномалій наведено на рис. 1.4.

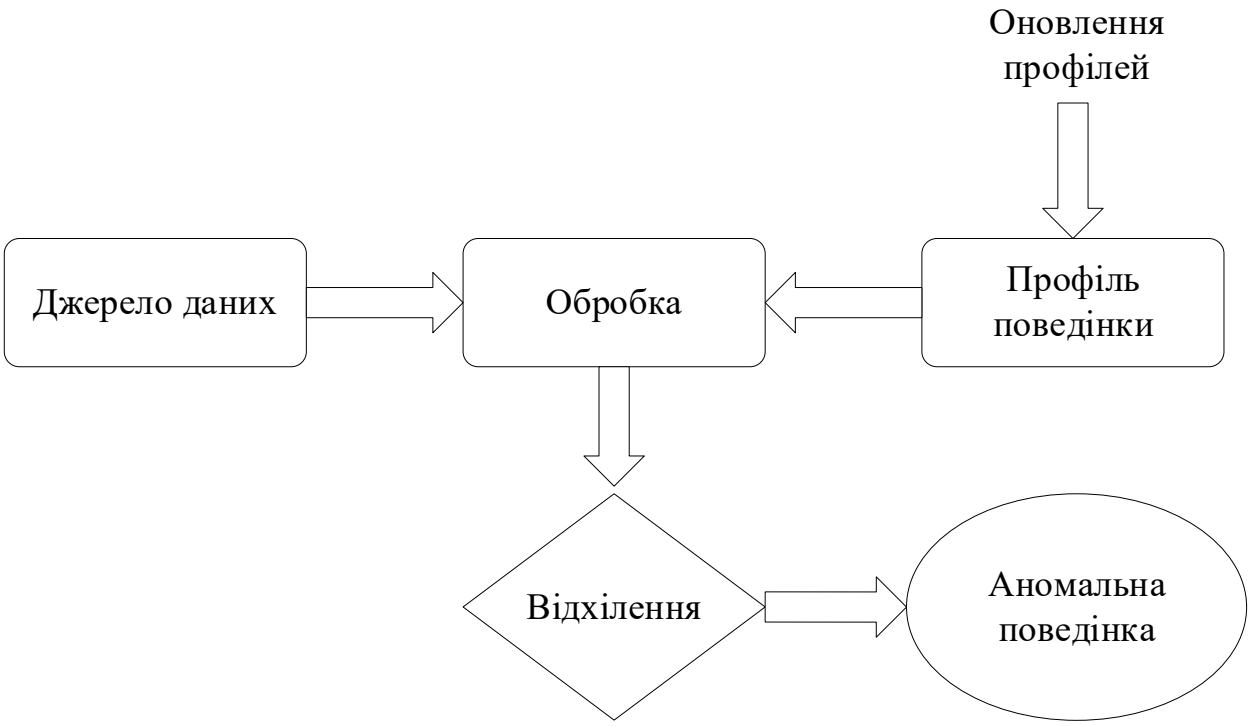

Рисунок 1.4 – Схема типової системи виявлення аномалій

Якщо описати профіль нормальної поведінки суб’єкта, будь-яке відхилення від нього можна схарактеризувати як аномальна поведінка.

Переваги:

– СВВ, виявлення аномалій, фіксуючи незвичну поведінку системи, отримують можливість визначати характер дії атак, не маючи конкретних відомостей про атаку;

– детектори аномалій здатні збирати інформацію, яка надалі передається до детекторів зловживань для визначення сигнатур.

Недоліки:

– під час виявлення аномалій, існує ймовірність створення помилкових сповіщень у разі непередбаченої поведінки користувачів та мережевої активності;

– цей метод часто потребує навчання системи на базі минулих спрацювань, під час якого визначаються особливості нормальної поведінки. Проходження цього етапу навчання суттєво впливає на подальшу ефективність СВВ;

– не можна реалізувати опис атаки за елементами. Повідомляється, що відбувається щось підозріле;

– дана технологія значно залежить від середовища функціонування як визначального фактора анормальної поведінки;

– відносна низька швидкість аналізу;

– трудомісткі завдання побудови профілів суб'єктів ІС.

Кожен засіб захисту адресований конкретній загрозі у системі. Щобільше, кожен засіб захисту має слабкі та сильні сторони. Тільки комбінуючи їх, можна захиститись від максимально великого спектра атак.

Системи виявлення атак стають необхідним доповненням безпеки інфраструктури в кожній організації. Технології виявлення проникнень не гарантують абсолютну безпеку системи. Але практична, велика користь від даної системи систем виявлення атак існує і не маленька, що доведено експертним методом оцінювання в таблиці 1.1.

Таблиця 1.1 – Порівняння методів СВВ

Характеристика	Сигнатурні методи	Методи аномалій
множина виявлених атак	обмежується відомими видами атак	обмежується можливостями налаштування та методами аналізу СВВ
ймовірність пропуску атаки	середня	низька
ймовірність помилкового спрацьовування	дуже низька	висока
вимоги до обчислювальних ресурсів ІС	середні	високі

Важливий елемент у системах знаходження атак - це швидкість реакції, що відбувається через певні проміжки часу, тобто пакетно.

1.4 Дослідження чинних методів виявлення вторгнень

Системи, що виявляють вторгнення є одним з механізмів аналізу поведінки комп'ютерної мережі та виступають як важливе доповнення безпеки мережі. СВВ служать механізмами моніторингу та спостереження підозрілої активності, проводять аналіз ресурсів мережі, а також самостійні дії щодо

ідентифікації аномальних подій у мережі – реальних порушень та спроб порушень [1–10].

Існує значний обсяг публікацій та досліджень, присвячених аналізу сучасних СВВ та методів виявлення несанкціонованих вторгнень. У роботі проводиться порівняльний аналіз систем, наводяться їх переваги та недоліки, загальна характеристика їх структури.

У роботі [3] наведена структура сучасних СВВ, яка включає наступні підсистеми:

– підсистема збору інформації про систему, що підлягає захисту;

– підсистема аналізу для пошуку атак та вторгнень у систему;

– підсистема представлення даних для контролю системи у режимі реального часу.

Підсистема збору інформації отримує дані від автономних модулів, датчиків програмного забезпечення (ПЗ) системи, датчиків хоста, міжмережевих та мережевих датчиків, скомпонованих залежно від завдань структури мережі та типу інформації, що підлягає аналізу.

Ієрархічно підсистема аналізу як вхідні дані використовує інформацію з попередньої підсистеми та включає набір аналізаторів, скомпонованих за завданнями виявлення вторгнень заданого типу. Ефективність виявлення вторгнень залежить від параметрів аналізаторів та їх кількості.

Підсистема представлення даних орієнтована різні групи користувачів, які контролюють певні підсистеми мережі. Тому таких СВВ використовують розмежування доступу, групові політики, повноваження та інших.

Залежно від наборів параметрів оцінки стану системи, сучасні СВВ використовують дві групи методів. У разі фіксованого набору параметрів оцінки та фіксованого часу навчання використовуються методи контрольованого тобто “навчання з вчителем”. У разі коли множина параметрів оцінки може змінюватися протягом заданого часу дослідження, а процес навчання відбувається весь час, використовуються методи неконтрольованого навчання (“навчання без вчителя”). У табл. 1.2–1.4 наведені характеристики методів навчання [4].

В таблиці 1.2 наведені характеристики методів виявлення аномалій методами «навчання з вчителем»

Таблиця 1.2 – Виявлення аномалії методами “навчання з вчителем”

Метод виявлення	Системи	Характеристика методу
Моделювання правил	W&S	Під час навчання СВВ будує набір правил, які описують нормальну поведінку системи. СВВ аналізує поведінку користувачів системи, порівнюючи їх із набором правил. Якщо виявляється розбіжність, то СВВ подає сигнал про виявлення атаки вторгнення.
Описова статистика	IDES, NIDES, EMERLAND, JiNao, HayStack	Система навчається з урахуванням зібраної описової статистики безлічі показників системи. Для визначення атаки вторгнення обчислюють “відстань” між векторами показників – поточних та збережених. Якщо відстань між векторами перевищує певне значення, такий стан вважається атакою
Нейромережа	Hyperview	Використовуються різні нейромережеві структури. Для навчання застосовуються дані про нормальний стан системи. Далі нейромережа використовується виявлення атак вторгнення, які від нормального поведінки системи.

В таблиці 1.3 наведені характеристики методів виявлення аномалій методами «навчання без вчителя».

Таблиця 1.3 – Виявлення аномалії за методами “навчання без вчителя”

Метод виявлення	Системи	Характеристика методу
Моделювання безлічі станів	DPEM, JANUS, Bro	Нормальний стан системи описується спеціальним набором станів та переходів між цими станами. Стан системи – це особливий вектор з певним набором значень параметрів системи
Описова статистика	MIDAS, NADIR, Haystack, NSM	Використання методів навчання з вчителем

В таблиці 1.4 наведені методи зловживань «навчання з вчителем»

Таблиця 1.4 – Виявлення зловживань методами “навчання з вчителем”

Метод виявлення	Системи	Характеристика методу
Моделювання станів	USTAT, IDIOT	Для визначення атаки вторгнення використається набір певних станів. Під станом розуміють набір визначених значень параметрів, що оцінюють систему визначених чином.
Експертні системи	NIDES, EMERLAND, MIDAS, DIDS	Для виявлення атак використовуються різні набори правил
Моделювання правил	NADIR, HayStack, JiNao, ASAX, Bro	Спрощений варіант експертних систем
Синтаксичний аналіз	NSM	СВВ проводить синтаксичний аналіз, який дає змогу знайти певні комбінації символів, які можна розпізнати як атаку.

Основною ідеєю виявлення нестандартної поведінки мережі, що підлягає захисту є формування профілю або образу мережі. Тому основними методами, у яких базується реалізація СВВ, є методи розпізнавання образів. У цьому образ нормальної поведінки формується з урахуванням аналізу параметрів оцінки мережі.

Висновки про аномальну поведінку формуються на основі відхилень від значень оцінок параметрів від профілю мережі. Розмір і характер відхилень, як правило, в режимі реального часу дозволяють проводити ідентифікацію аномалії - технічний збій, припустимо відхилення пов’язане з дією зовнішнього середовища, атака на мережу.

Формування профілю або образу СВВ проводиться за допомогою наступних методів:

– статистичні методи аналізу параметрів оцінки;

– методи множинного опису подій та формальної логіки;

– методи нечіткої логіки та нейронних мереж.

Виходячи зі способів формування профілю мережі та методів виявлення аномалій, можна визначити два класи задач:

– вибір оптимальної множини параметрів оцінки;

– визначення загального показника аномальності.

Питання визначення інтегральної оцінки аномальності поведінки мережі на сьогодні практично не вирішене завдяки неоднозначному розв’язання задачі формування множини оцінок параметрів мережі, що підлягає захисту. Крім того, виникає ще три питання:

– формування оптимальної множини параметрів;

– зміна множини, що склалася в часі;

– оцінка адекватності множини, що склалася, на заданому інтервалі часу і при зміні типів вторгнень у мережу.

Вирішення завдань формування множини оцінок параметрів та їх динаміки можливе при використанні методу групового обліку аргументів (МГУОА), еволюційних та генетичних алгоритмів [5]. Тоді визначення інтегральної оцінки може бути проведено за допомогою ймовірнісних методів на основі баєсівської статистики [6], аналізу коваріацій параметрів оцінок [6], методів кореляційного аналізу [7], автокореляційних моделей [7].

РОЗДІЛ 2. ПРОЄКТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ ВИЗНАЧЕННЯ DDOS АТАК

2.1 Список функціональних вимог

Функціональні вимоги відповідають питанням “Що має робити програмне забезпечення?”. Функціональні вимоги – це всі основні функції, які мають бути реалізовані у системі [8].

Розглянемо функціональні вимоги відповідно до поставленого завдання.

Користувач повинен мати можливість:

1) Навчання нейромережі:
   – Створення файлу для навчання нейромережі.
   – Вибір файлу для навчання нейромережі.
2) Тестування нейромережі:
   – Створення файлів для тестування нейромережі.
   – Вибір файлів для тестування нейромережі.
3) Перегляд результатів у вигляді графіків.
4) Редагування таблиць БД:
   – Додавання даних.
   – Редагування даних.
   – Видалення даних.

Вимоги до ПЗ зручно подати за допомогою діаграми варіантів використання (рисунок 2.1).

Діаграми варіантів використання описують взаємовідносини та залежності між групами варіантів використання та дійовими особами, що беруть участь у процесі. Варіант використання визначає, з погляду дійової особи, групу дій у системі, що призводять до конкретного результату. Варіанти використання мають можливість взаємодіяти з іншими варіантами використання.

Існує 3 типи взаємодії між варіантами використання:

– Варіант використання, який вбудовано в інший варіант використання являється включеним.

– Розширення вказує, що у певних ситуаціях варіант використання буде розширено іншим.

– Варіанти використання мають можливість взаємодіяти з іншими варіантами використання.

– Узагальнення вказує, на те, що варіант, який використовується, успадковує характеристики “батьківського” варіанта використання і має змогу самостійно перевизначити деякі з них або додати нові.

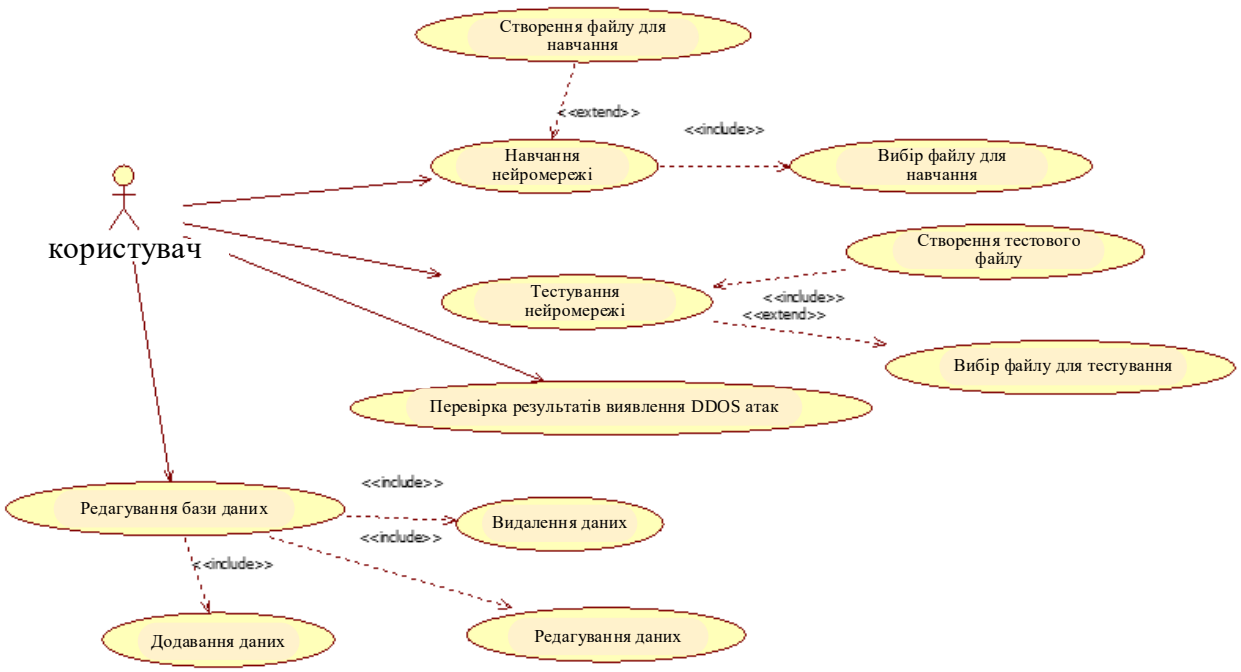

Рисунок 2.1 – Use case діаграма вимог до ПЗ

2.2 Список нефункціональних вимог

Нефункціональні вимоги – це неформальні вимоги, які описують, як саме має працювати дана система або програмний продукт, і які властивості чи характеристики вона повинна мати [9]. Для програмного забезпечення були зазначені такі вимоги:

– Надійність. Дублювання важливих даних.

– Доступність. Цілодобова робота системи.

– Інтуїтивно зрозумілий інтерфейс.

– Супроводжуваність. Наявність коментарів у коді.

– Можливість інтеграції з аналізаторами трафіку.

Обмеження, пов’язані з обладнанням комп’ютерів: мінімальна частота процесора (як на сервері, так і на ПК-клієнтах) – 2.0 ГГц, мінімальний обсяг

оперативної пам'яті на сервері 4 Гб, на клієнтах – 2 Гб, мінімальний об’єм дискового простору на сервері – 80 Гб на клієнтах 40 Гб.

Мережа з підтримкою технології Fast Ethernet зі швидкістю щонайменше 10 Мбіт/с, рекомендована швидкість 100 Мбіт/с.

2.3 Вибір нейронної мережі визначення DDOS атак

Багатошаровим персептроном є нейронна мережа прямого поширення. Вхідний сигнал у таких мережах даного типу поширюється в наплямку прямо від шару до шару відповідно. Багатошаровий персептрон у загальному поданні складається з наступних елементів [1–7, 10–31]:

– масив вхідних вузлів, які і створюють вхідний шар;

– обчислювальних нейронів, що являють собою один або кілька прихованих шарів;

– одного вихідного шару нейронів.

Багатошаровий персептрон є узагальнення одношарового персептрона Розенблата. Прикладом багатошарового персептрона являється така модель нейронної мережі (рис. 2.2).

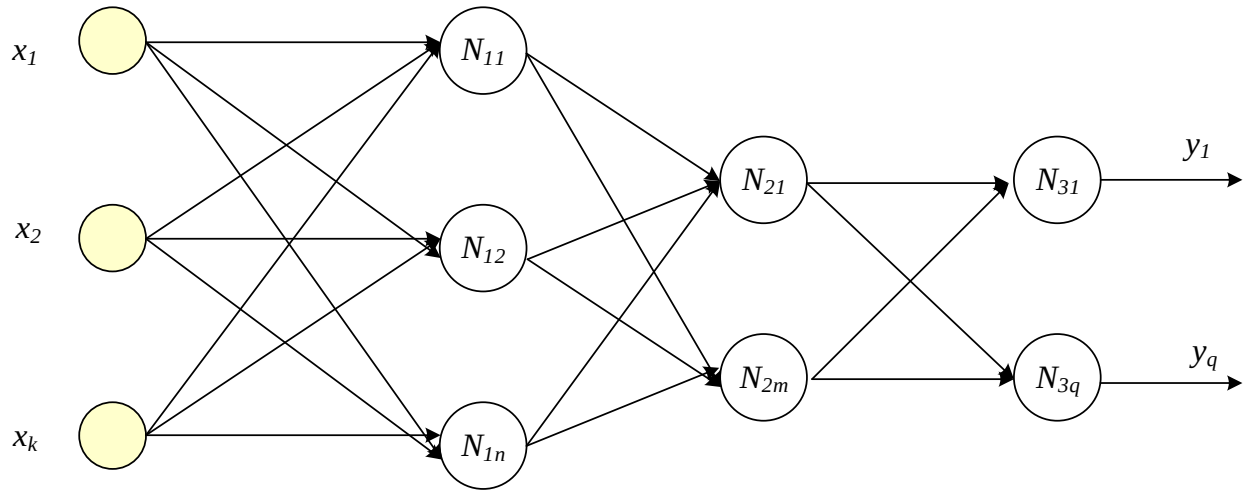

Рисунок 2.2 - Багатошаровий персептрон

Перед побудовою нейронної мережі на тренувальному наборі даних було виключено параметри, які мали однакові значення на всій вибірці. Це було

зроблено для прискорення отримання результату. Перша нейронна мережа була побудована за 28 параметрами.

Вона складалася з вхідного, одного прихованого та вихідного шарів. Вхідний та прихований шари мали по 28 нейронів кожен, вихідний складався з одного нейрона, що містить висновок про атаку (1 – атака 0 – нормальний трафік). Ця нейронна мережа представлена рис. 2.3.

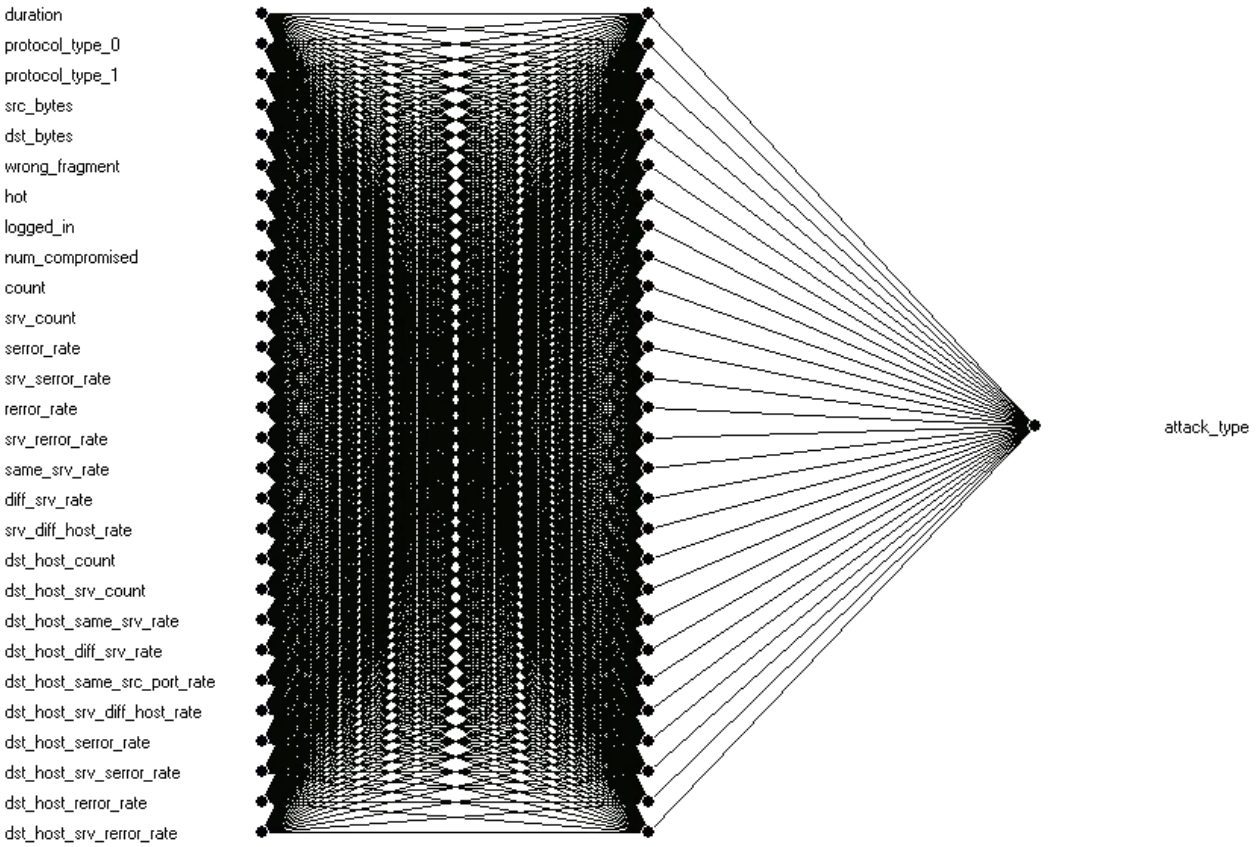

Рисунок 2.3 – Нейронна мережа 28×28×1

Після побудови нейромережі було проведено 3 тести для оцінки якості її роботи з виявлення атак. Перший тест був проведений для атак з відомими для нейромережі типами (back, neptune, smurf, teardrop). Були отримані такі результати (рис. 2.4).

Рисунок 2.4 – Тестування нейромережі на відомих типах атак

Як видно з рисунку нейромережа майже 100% точністю розпізнала атаки відомих типів.

Далі проводився тест для нормального трафіку. Результати обчислень представлені рисунку 2.5.

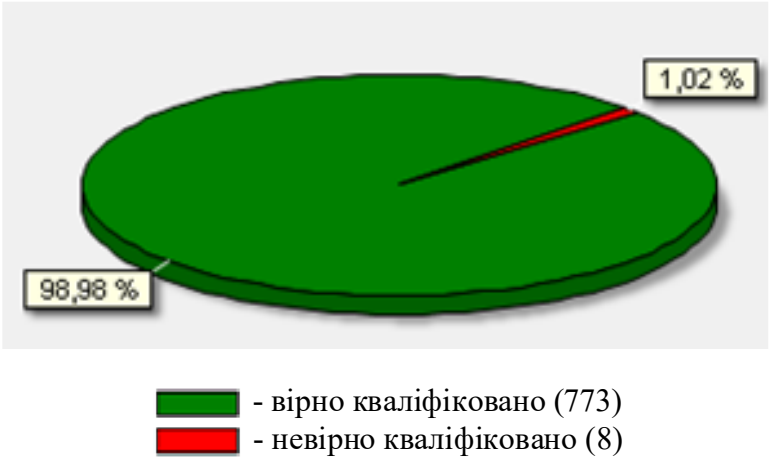

Рисунок 2.5 – Тестування нейромережі на нормальному трафіку

В цьому випадку результати були навіть кращими, ніж у випадку з певними типами атак.

І останній тест виконувався з невідомими типами атак на нейронну мережу, а саме з атаками типу land і pod. Результат цього тесту зображено рисунку 2.6.

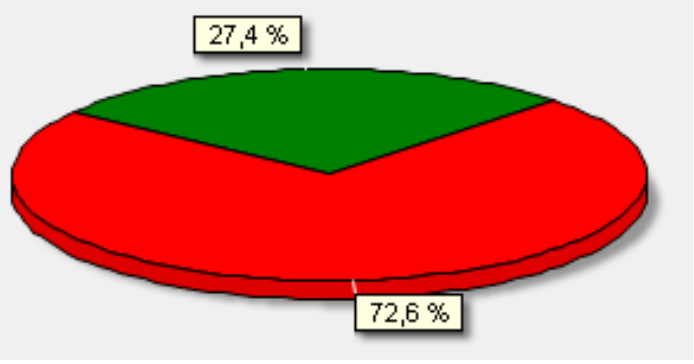

Рисунок 2.6 – Тестування нейромережі на невідомих типах атак

На відміну від попередніх тестів, результат сильно відрізняється. Тобто в цьому випадку можна сказати, що кожна 4-атака буде виявлена. Але слід зазначити, що оскільки ці типи атак не були присутні в тренувальному наборі, то можна сказати, що це гарний результат. А також знання того, що такі методи як статистичний аналіз та метод сигнатурного аналізу, з відсутністю інформації про дані атаки взагалі промаркувала б їх як нормальний трафік говорить про те, що використання нейромереж для виявлення вторгнень є виправданим, оскільки вони мають здатність адаптуватися до невідомих атак.

Оскільки були отримані задовільні результати, було прийнято рішення про побудову нейронних мереж з різними параметрами для визначення оптимальної конфігурації для виявлення максимальної кількості атак. Зміни проводилися у кількості вхідних параметрів, зміні функції активації та її крутості, і кількості прихованих шарів.

Наступні нейронні мережі мають загальну конфігурацію: 15 вхідних нейронів, 16 нейронів на прихованому шарі та 1 вихідний нейрон (рис. 2.7).

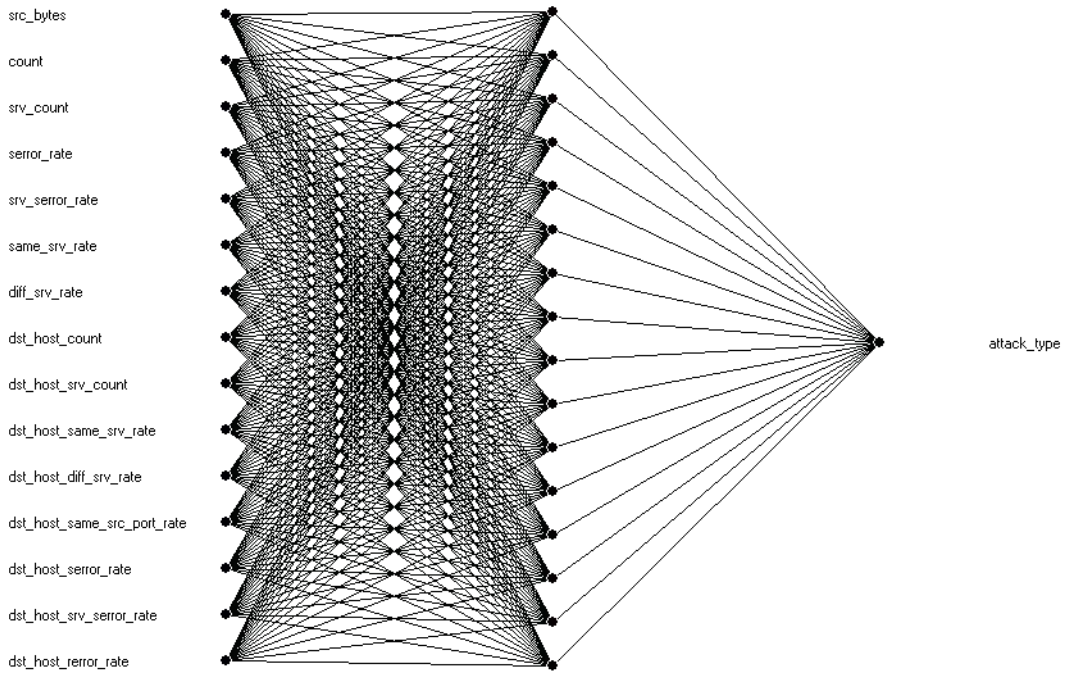

Рисунок 2.7 – Нейронна мережа 15×16×1

Всі нейромережі 15×16×1 мають однаковий вигляд, різниця між ними полягає лише у різних функціях активації та значення параметра крутості (табл. 2.1).

Таблиця 2.1 – Тестові нейронні мережі

№	Розмір	Функція активації	Крутизна функції
1	15×16×1	Сігмоїда	1
2	15×16×1	Сігмоїда	1,5
3	15×16×1	Гіпертангенс	1
4	15×16×1	Арктангенс	1

Для кожної з побудованих мереж проводилися раніше описані тести, а саме виявлення атак із відомими типами, нормального трафіку та атак із невідомими типами. Результати, отримані під час використання даних нейромереж представлені у таблиці 2.2.

Таблиця 2.2 – Результати дослідження нейромереж 15×16×1

№	Виявлення відомих атак, %	Виявлення нормального трафіку, %	Виявлення невідомих атак, %
1	99,76	97,18	34,25
2	99,88	95,13	33,79
3	100	0	100
4	99,18	60,69	57,08

Виходячи з результатів можна сказати, що найкраще показала себе функція активації сигмоїду. Арктангенс і гипертангенс видали незадовільні результати, хоча розпізнавання атак з невідомим типом значно зросла, сильно постраждала якість визначення нормального трафіку. Тому в такому випадку можна зробити висновки, що для цього завдання краще підходить функція активації сигмоїду. Щодо коефіцієнта крутості можна сказати, що коефіцієнт 1,5 не надав поліпшення результатів. Наступні дослідження проводилися для сигмоїд та коефіцієнтом 1, оскільки в цій конфігурації були отримані найкращі результати. Подальші зміни стосуються лише кількості нейронів та кількості прихованих шарів. Далі були побудовані нейромережі з 21, 26 та 31 нейронами у прихованому шарі (рис. 2.8 – 2.10).

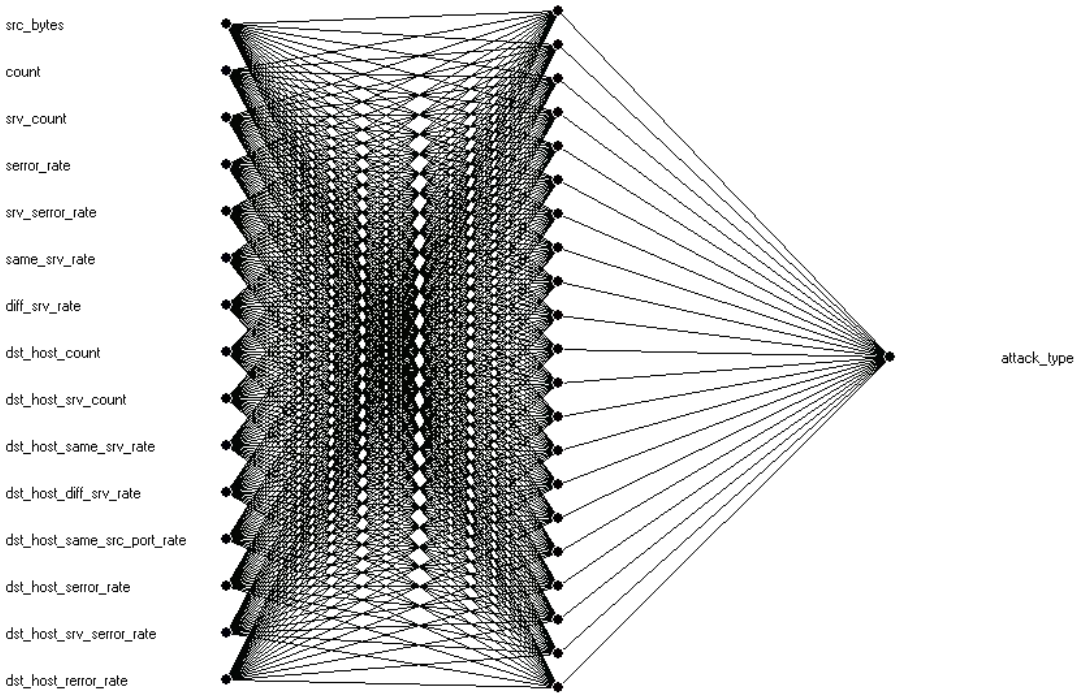

Рисунок 2.8 – Нейронна мережа 15×21×1

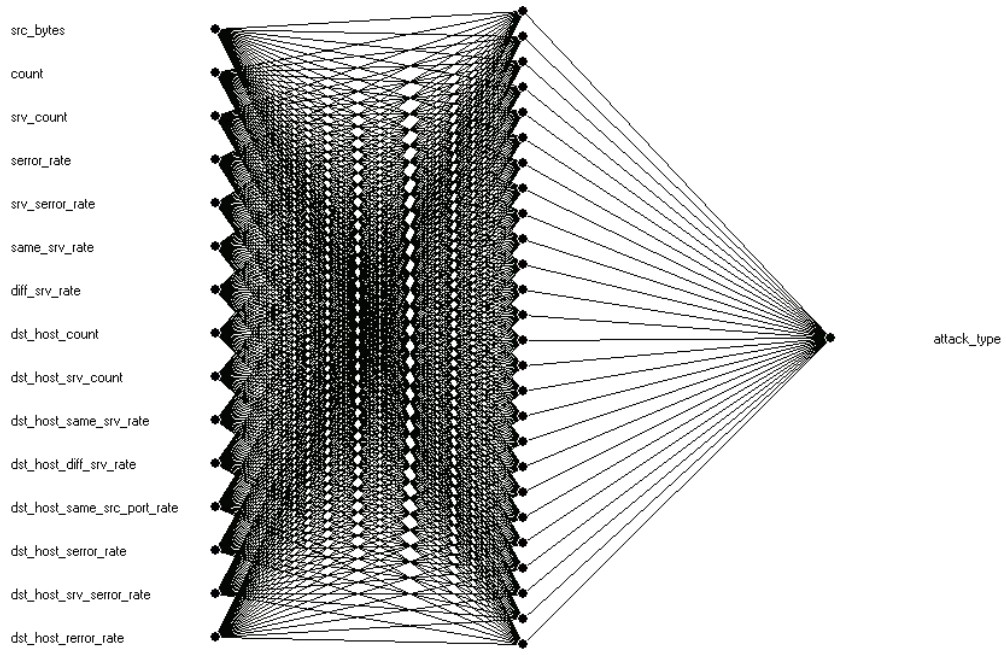

Рисунок 2.9 – Нейронна мережа 15×26×1

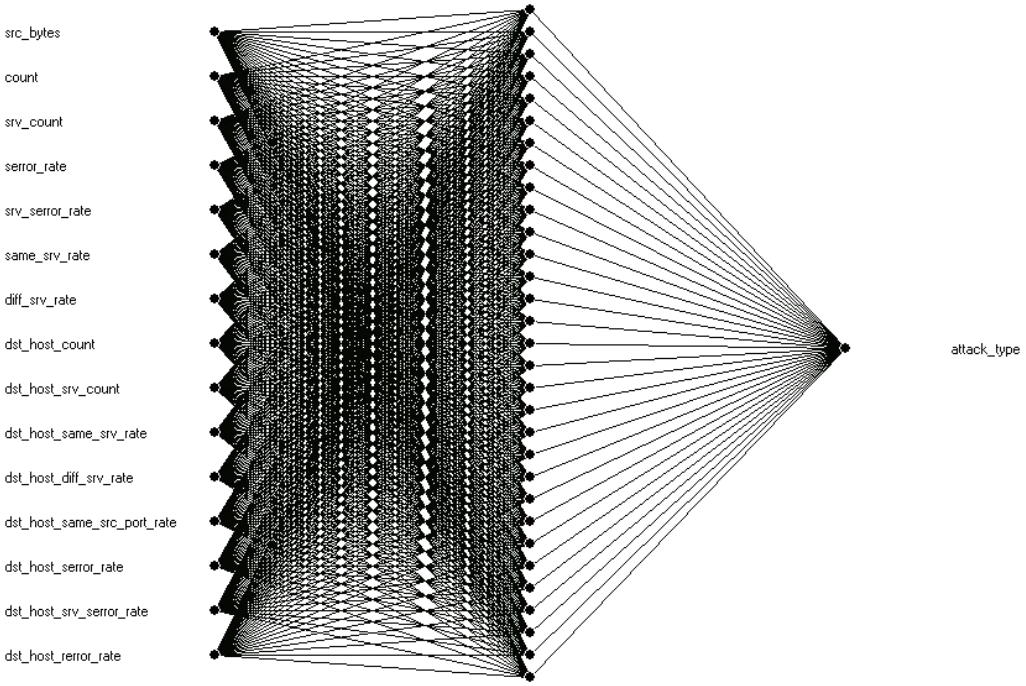

Рисунок 2.10 – Нейронна мережа 15×31×1

У табл. 2.3 наведені результати тестування мереж.

Таблиця 2.3 – Результати тестування

Мережа	Виявлення відомих атак, %	Виявлення нормального трафіку, %	Виявлення невідомих атак, %
15×21×1	99,68	95,77	33,79
15×26×1	99,78	96,03	33,79
15×31×1	99,7	96,8	34,7

З отриманих результатів можна побачити, що результати не сильно відрізняються, але в такому випадку нейромережа з великою кількістю нейронів є більш точною. Проте однозначний висновок у тому, що зростання кількості нейронів прихованого шару веде до зростання точності зробити важко.

Останні два експерименти проводилися з нейромережею з двома прихованими шарами (рис. 2.11) та нейромережею з меншою кількістю вхідних нейронів (рис. 2.12), а саме з 10.

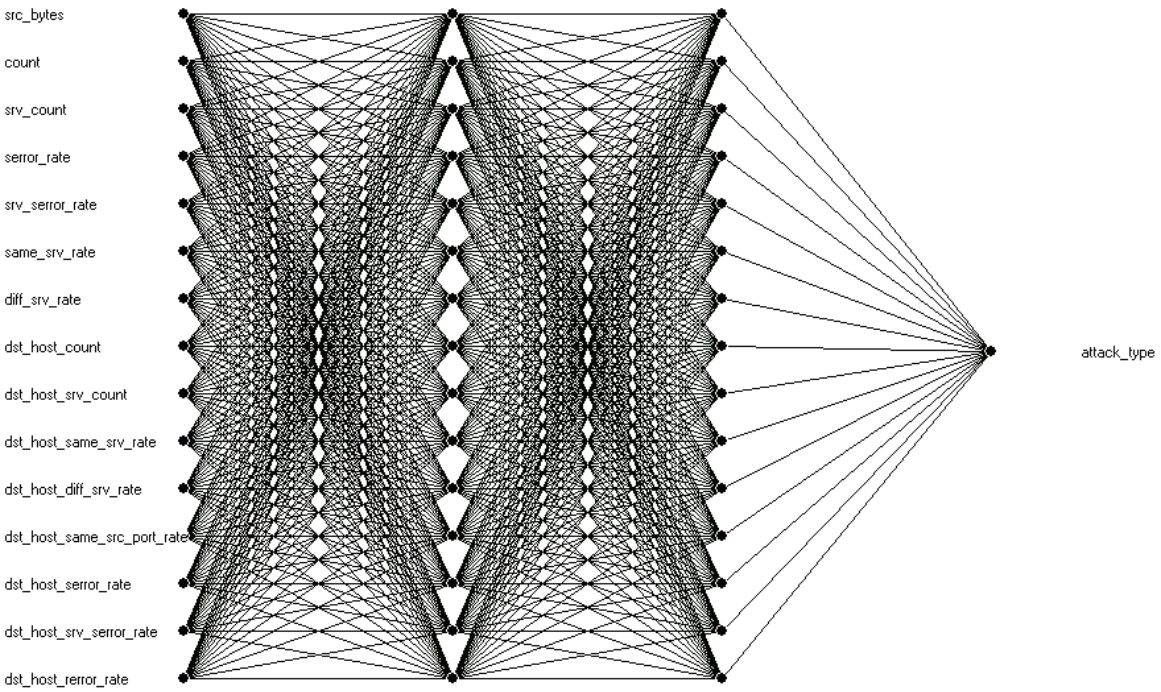

Рисунок 2.11 – Нейромережа з двома прихованими шарами

Дана нейромережа має 2 приховані шари, кожен з яких містить 15 нейронів.

У наступному випадку для експерименту було знижено кількість параметрів, що подаються на вхід нейромережі. Метою було дізнатися, наскільки зміниться точність обчислень і в який бік, при обмеженій кількості відомої інформації.

Як вхідні нейрони були обрані 10 параметрів, на всій вибірці частіше за інших отримували різні значення, число нейронів на прихованому шарі було збільшено до 22. Результати можна побачити в табл. 2.4.

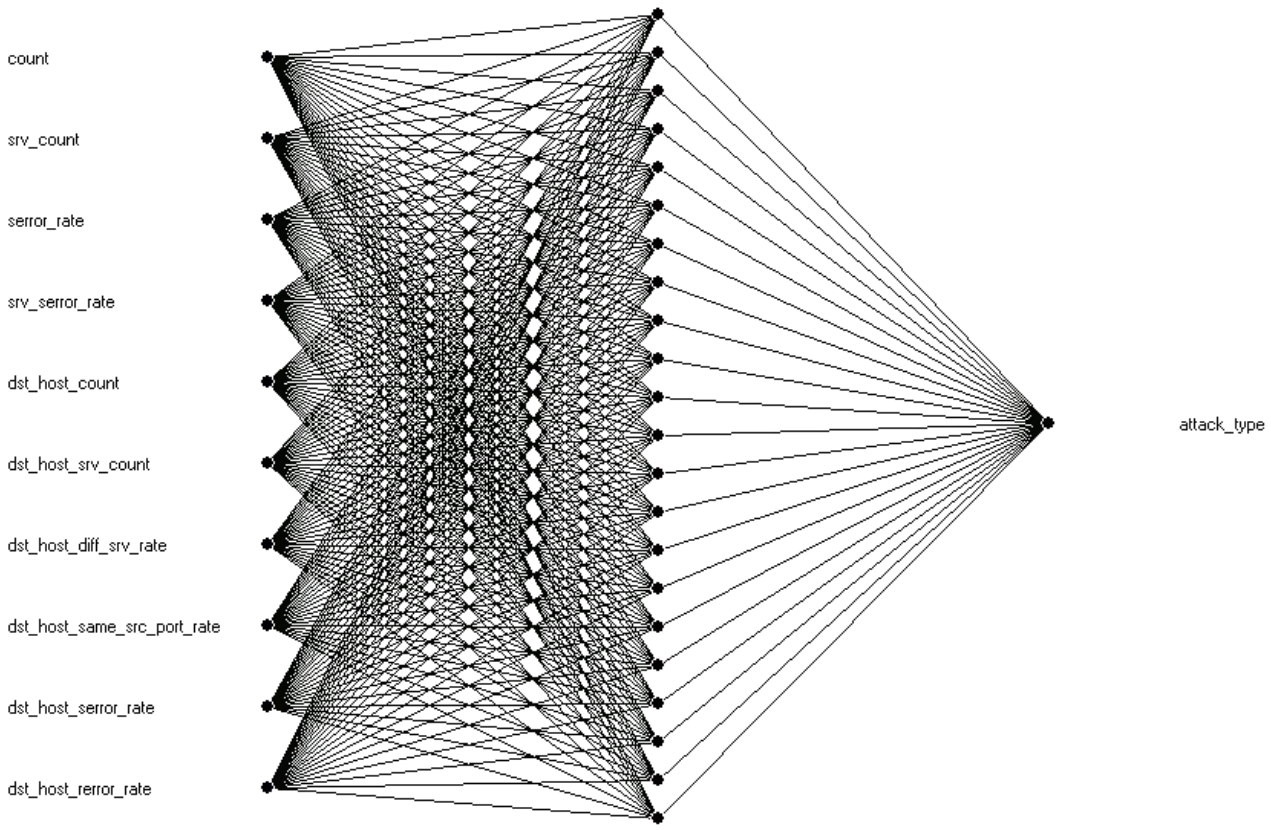

Рисунок 2.12 – Нейромережа з 10 вхідними нейронами

Таблиця 2.4 - Результати нейромереж 15×15×15×1 та 10×22×1

Мережа	Виявлення відомих атак, %	Виявлення нормального трафіку, %	Виявлення невідомих атак, %
15×15×15×1	99,8	95,01	34,25
10×22×1	99,96	93,34	31,51

З результатів досліджень видно, що нейромережа з 10 вхідними нейронами має найгірші результати, ніж нейромережі з великою кількістю вхідних параметрів. Отже, сильне скорочення числа вхідних параметрів негативно впливає на результат.

Що стосується нейромережі з двома прихованими шарами, то вона має приблизно такі ж результати, як і нейромережі 15×16×1 та 15×31×1. Якщо узагальнити значення (підсумувати відсотки знаходження та розділити на кількість експериментів) для мереж з кращим результатом, а саме для 15×16×1, 15×31×1 та 15×15×15×1, то можна побачити, яка нейромережа в результаті краще впоралася із завданням (табл. 2.5).

Таблиця 2.5 – Результати нейромереж з найкращими результатами

Мережа	Виявлення відомих атак, %	Виявлення нормального трафіку, %	Виявлення невідомих атак, %	Мережа
15×15×15×1	99,8	95,01	34,25	76,35
15×31×1	99,7	96,8	34,7	77,07
15×16×1	99,76	97,18	34,25	77,06

Серед розглянутих у таблиці 2.5 нейромереж краще із завданням виявлення атак впоралася нейромережа з 31 нейроном на прихованому шарі.

Отже, як можна побачити в порівнянні з першим експериментом, де% знаходження невідомих атак був 27,4% вдалося отримати збільшення до 34%, тобто, кожен третій невідома атака буде виявлено.

Таким чином, можна зробити висновок, що хоча відсоток знаходження не дуже великий, але є задовільним, оскільки це набагато краще, ніж пропускати атаки як нормальний трафік. Можна сказати, що використання багатошарового персептрона для цього є виправданим.

2.4 Обґрунтування вибору програмних та технічних засобів

Як СУБД обрано MS SQL Server. Microsoft SQL Server 2016 (рис. 2.13), який є найновішою та найпотужнішою системою управління базами даних.

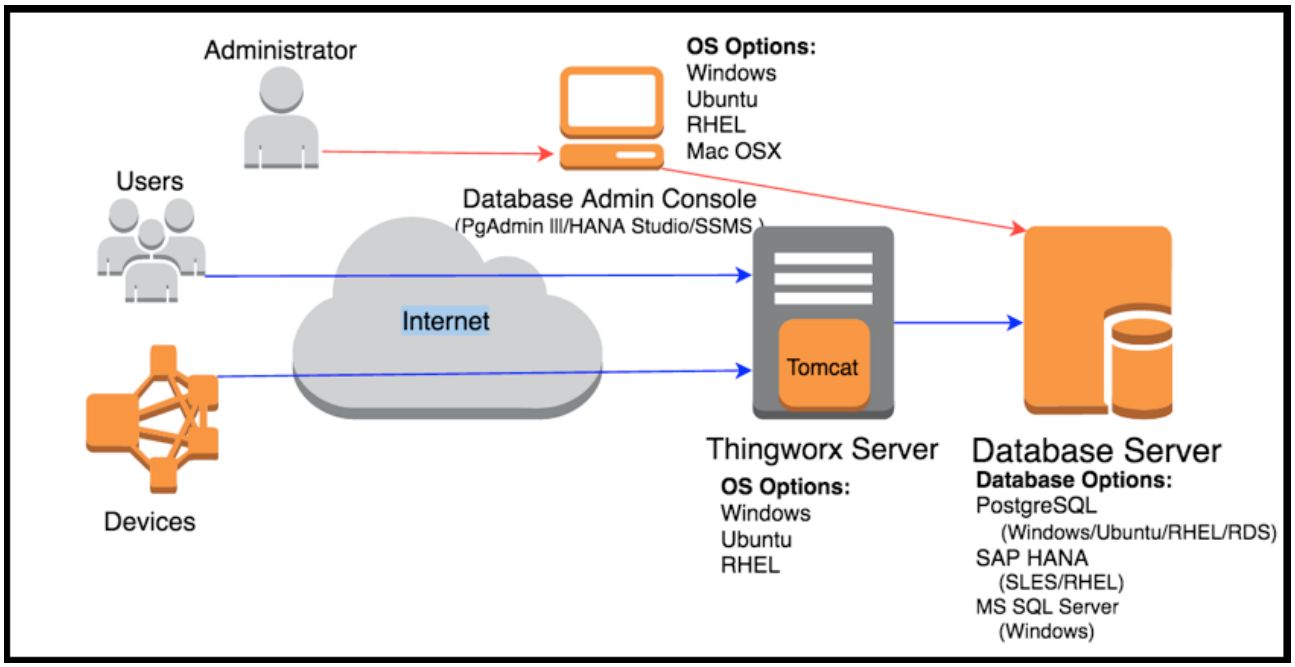

Рисунок 2.13 – Платформа даних SQL Server

Остання версія SQL Server 2016 була випущена у червні 2016 року. Вона надає безліч нових та покращених можливостей, у тому числі:

Нова функція реляційної бази даних, яка може використовуватися для зберігання деяких даних на місці та надсилання даних, що рідко використовуються, в хмару Microsoft Azure. Програми можуть отримати доступ до всіх даних, незалежно від того, де вони зберігаються [32–35].

Функція Always Encrypted дозволяє шифрувати дані на рівні стовпців.

Polybase, яка інтегрує SQL Server з розподіленою файловою системою Hadoop та дозволяє запитувати дані Hadoop за допомогою SQL та об'єднувати їх із власними реляційними даними.

Розширена аналітика в базі даних дозволяє включати мову програмування R до SQL Server, включаючи його в збережені процедури.

динамічне маскування даних може використовуватися для приховування даних, захищаючи фактичні значення даних від неуповноваженого персоналу з метою регулювання та відповідності.

Підтримка тимчасових даних дозволяє автоматично відстежувати історичні зміни даних з часом.

Інші функції включають вбудовану підтримку нотації об'єктів JavaScript, а також покращення можливостей вбудованої пам'яті Hekaton від Microsoft. Ця остання версія також надає новий формат зберігання, безпеку на рівні рядків та покращення Transact-SQL для оптимізованих для пам'яті таблиць.

Впровадження Microsoft SQL Server із віддзеркалюванням баз даних (рис. 2.14). Дзеркало – забезпечує високу доступність бази даних Microsoft SQL. Він працює, підтримуючи дві копії однієї бази даних, які знаходяться у різних примірниках SQL Server.

Один із двох примірників обслуговує запити від програми, тоді як інший діє як сервер "гарячого" резервування, готовий розпочати роботу будь-якої миті.

Запуск цих баз даних у режимі високої безпеки гарантує, що дані завжди однакові в обох базах даних, при цьому транзакції записуються в обидві бази даних одночасно.

Віддзеркалювання також підтримує аварійне перемикання без втрати даних через сервер, що слідує. Це окремий сервер Microsoft SQL, який контролює основний екземпляр та забезпечує плавне перемикання у разі збою.

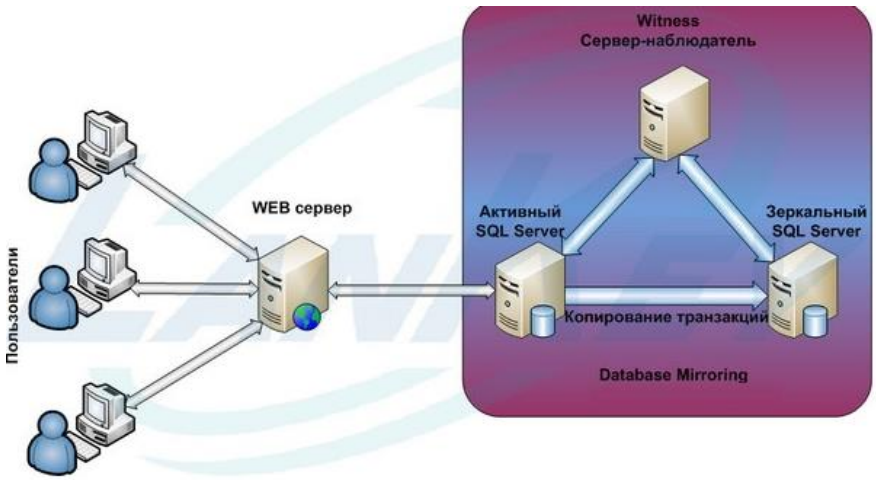

Рисунок 2.14 – Database Mirroring в SQL Server

У SQL Server дзеркало дають рішення для забезпечення високої доступності та аварійного відновлення.

Для переміщення транзакцій бази даних з однієї бази даних SQL Server (основна база даних) до іншої бази даних SQL Server (база даних віддзеркалювання) в іншому примірнику використовується дзеркальне зображення бази даних. У разі збою основного сервера дзеркальний сервер автоматично стає новим основним сервером і відновлює основну базу даних, використовуючи сервер, що стежить, в режимі високої доступності. Це суміш реплікації та доставлення журналів. Дзеркало працює тільки з повною моделлю відновлення [32–35].

Кластери високої доступності (High Availability Cluster) (рис.2.15).

Починаючи з MS SQL Server 2008R2, відмовостійка кластеризація забезпечує підтримку високої доступності для всього екземпляра SQL Server. Відмовостійкий кластер є комбінацією одного або декількох вузлів або серверів з двома або більше загальними дисками. Мінімум 2 вузла є обов'язковою умовою. Усі програми встановлені в кластерну групу Microsoft Cluster Service (MSCS), відому як групу ресурсів. В ідеалі ресурс Ім'я групи визначається для групування всіх ресурсів, які використовує кластер SQL. Будь-коли кожна група ресурсів належить лише одному вузлу в кластері. Служба програми має віртуальне ім'я, яке не залежить від імен вузлів, і згадується як ім'я примірника кластеру відмови. Додаток може підключитися до кластеру відмови від посилання на ім'я екземпляра кластеру відмови. Додаток не повинен знати, на якому вузлу знаходиться екземпляр відмовостійкого кластера.

Примірник кластеру відмови від SQL Server відображається в мережі як один комп'ютер, але має функціональність, що забезпечує перемикання з одного вузла на інший, якщо поточний вузол стає недоступний. Наприклад, під час збою обладнання, не пов’язаного з диском, збою операційної системи або запланованого оновлення операційної системи, можна налаштувати екземпляр SQL Server на одному вузлу відпрацювання відмови кластера для перемикання на будь-який інший вузол у групі дисків.

Відмовостійкий кластер не захищає від збою диска. Можна використовувати стійку кластеризацію, щоб зменшити час простою системи і забезпечуючи більш високу доступність додатків. Відмовостійка кластеризація підтримується в SQL Server Enterprise і SQL Server Developer, і, з деякими обмеженнями, в SQL Server Standard [32–35].

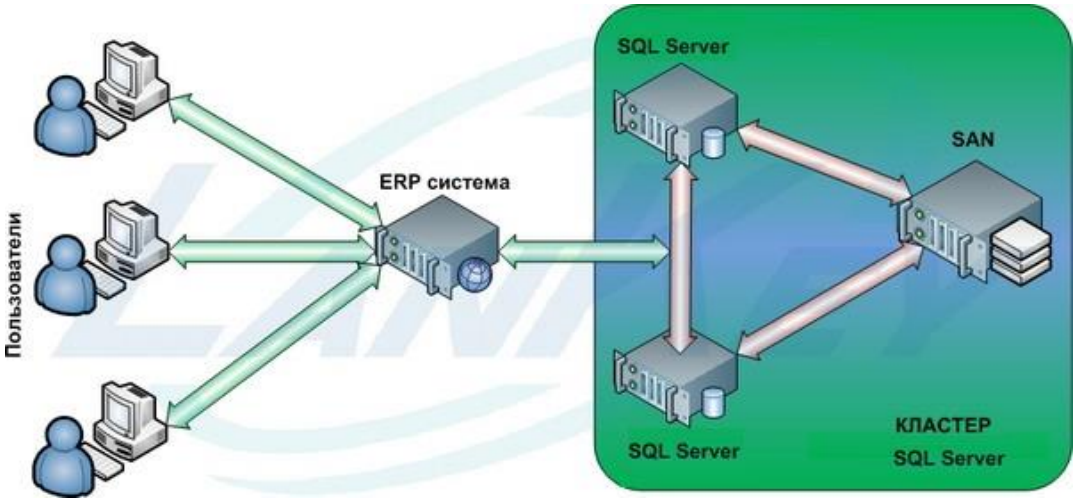

Рисунок 2.15 – Двовузловий кластер на базі SQL Server

На сьогоднішній день мова програмування C# одна з найпотужніших швидко розвиваються та популярний мов у IT-галузі. На цей час на C# здійснюється розробка різноманітних програм, такі: як невеликі десктопні програми та великі вебпортали та вебсервіси, що обслуговують щодня мільйо ни користувачів.

У порівнянні з іншими мовами C # досить молодий, але в той же час він просунувся далеко вперед. Перша версія мови була випущена в лютому 2002 року з випуском Visual Studio .NET . Поточна версія мови - версія 7.0 C#, випущена разом зі Visual Studio 2017 7 березня 2017 року. C# - це мова зі

синтаксисом, подібним до C, близьким до C++ та Java у цьому відношенні. Тож якщо ви знайомі з однією з цих мов, вам буде легше освоїти C#. C# є об’єктно-орієнтованим і багато що почерпнув з Java та C++ [32–35].

Наприклад, C# підтримує поліморфізм, успадкування, завантаження операторів, статичну типізацію. Об’єктно-орієнтований підхід дозволяє вирішити проблему створення великих, але гнучких, масштабованих і розширюваних додатків. І C# все ще розвивається, з кожною новою версією додається все більше цікавих функцій, таких як лямбда, динамічне зв’язування, асинхронні методи тощо.

2.5 Розробка інтерфейсу користувача

Діалог між користувачем ЕОМ визначається функціями, які виконує розроблений додаток. Доступ до функціонала інформаційної системи здійснюється за допомогою екранних форм. Дерево функція для інформаційної системи відділу продаж представлено рисунку 2.16. Усі функції поділяються на два підмножини:

– службові функції;

– основні функції.

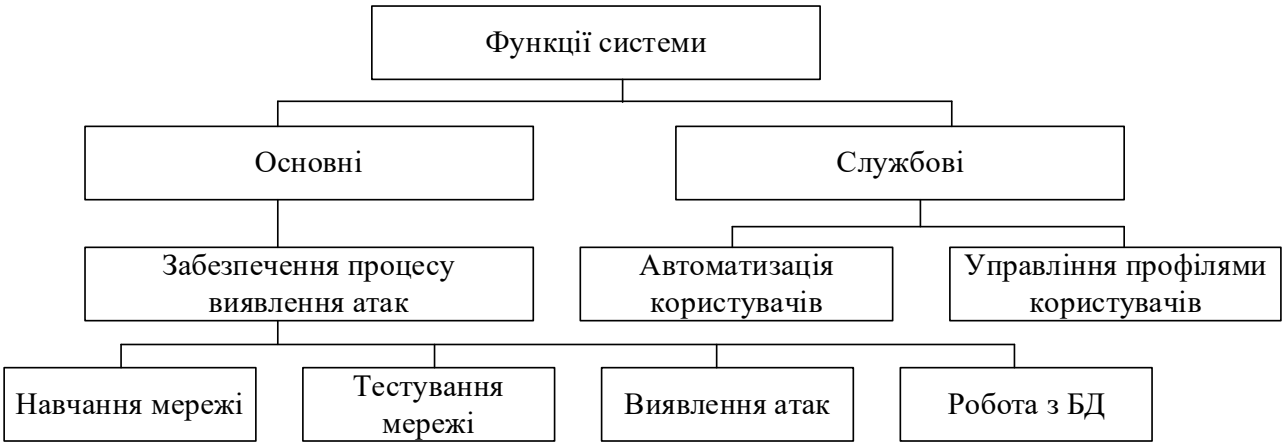

Рисунок 2.16 – Древо функцій

Виділені функції, їх ієрархія дозволяють розробити сценарій діалогу, який наведено на рисунку 2.17.

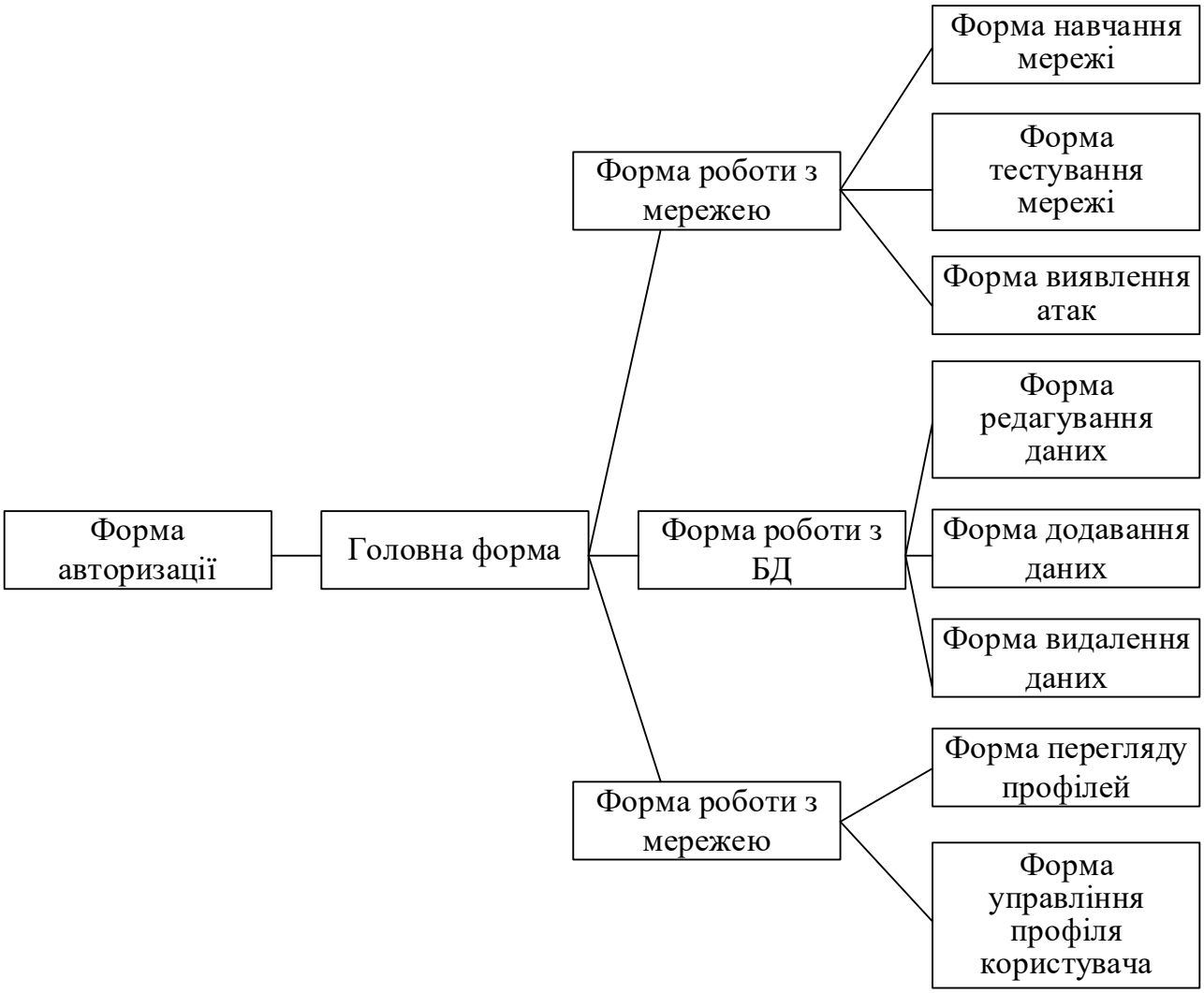

Рисунок 2.17 – Сценарій діалогу

Запропонований підхід на рис. 2.16–2.17 дозволяє забезпечити формування програмного застосунку на основі використання можливостей СВВ.

2.6 Проєктування баз даних. Нормалізація таблиць бази даних

Інформацію про базу даних можна подати у вигляді концептуальної, логічної та фізичної моделі даних.

Модель даних являє собою ER-модель (Entity-relationship model – модель “сутність–зв’язок”), що описує на кількох рівнях набір взаємопов’язаних сутностей, які згруповані за функціональними областями та зображати потреби бізнесу в аналітичному аналізі та звітності. Фізична модель даних предметної області представлена рисунку 2.18.

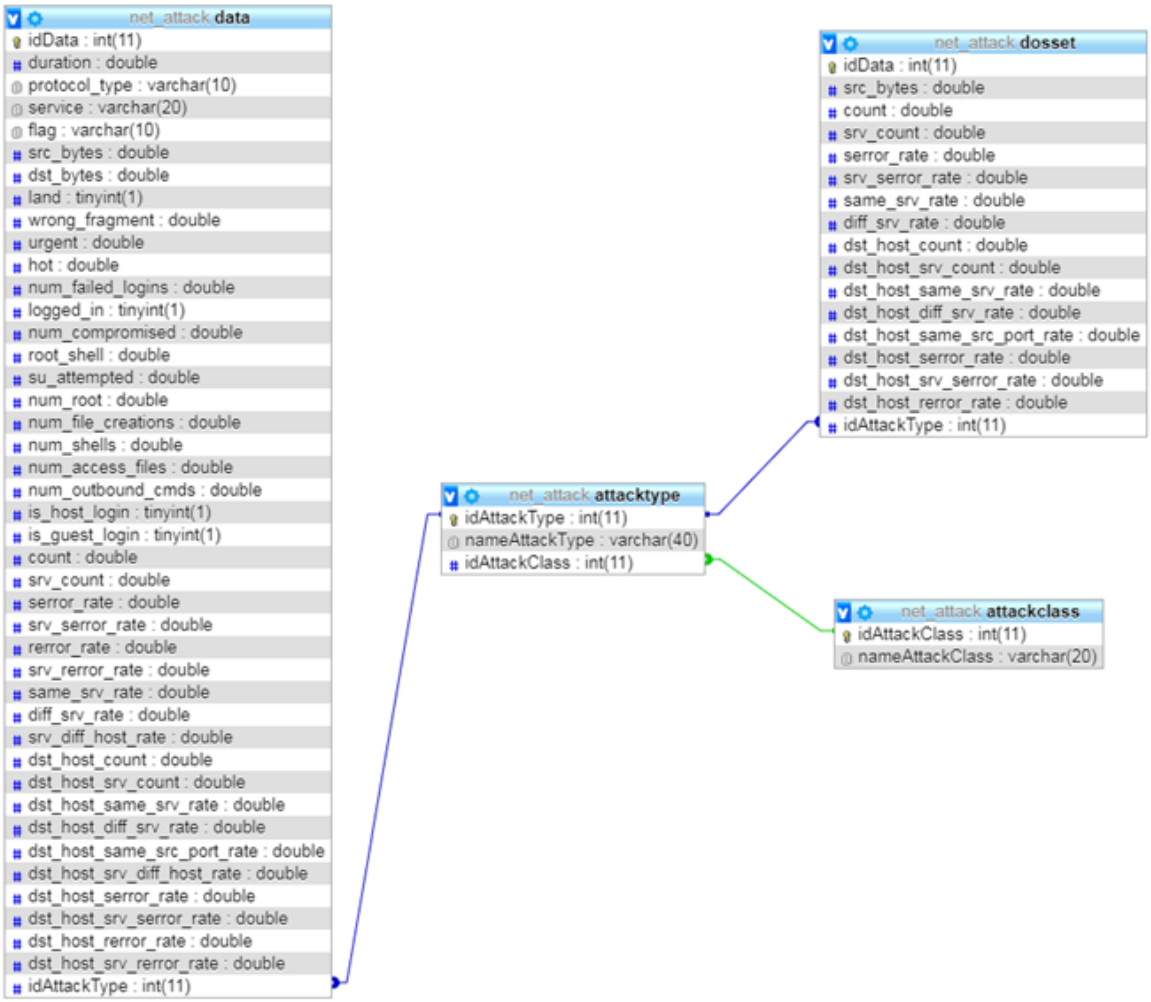

Рисунок 2.18 – Модель даних

Концептуальна модель даних є описом головних (основних) сутностей і відносин між ними. Концептуальна модель є відбивання предметних областей, у яких планується побудова сховища даних.

Логічна модель розширює концептуальну шляхом визначення для сутностей їх атрибутів, описів та обмежень, уточнює склад сутностей та взаємозв’язку між ними.

Фізична модель даних визначає реалізацію об'єктів логічної моделі лише на рівні об'єктів конкретної бази даних.

Таблиця data призначена для зберігання повного набору даних про нормальний трафік та атаки.

Складається з наступних полів:

– idData – ідентифікатор даних;

– duration – тривалість з’єднання;

– protocol_type – Протокол транспортного рівня (TCP, UDP, ICMP);

– service – сервіс прикладного рівня (http, ftp, smtp, telnet);

– flag – Статус з’єднання. Можливі статуси:

SF, S0, S1, S2, S3, OTH, REJ, RSTO, RSTOS0, SH, RSTRH, SHR;

– src_bytes – вхідний потік байт;

– dst_bytes – вихідний потік байт;

– land – якщо адреси джерела та одержувача збігаються, то дорівнює 1, інакше 0;

– wrong_fragment – кількість неправильних фрагментів;

– urgent – кількість строкових пакетів (пакети з активованим терміновим бітом);

– hot – сума гарячих дій у поєднанні, наприклад: введення системного каталогу, створення програм та виконання програм;

– num_failed_logins – число невдалих спроб входу;

– su_attempted якщо команда su була використана тоді 1 або 0;

– num_root – кількість спроб доступу з правами адміністратора;

– num_file_creations – кількість операцій створення файлу;

– num_shells – кількість спроб використання командного рядка;

– num_access_files – кількість операцій з файлами контролю доступу;

– num_outbound_cmds – кількість вихідних команд у ftp сеансі;

– is_host_login – якщо користувач має права адміністратора 1, інакше 0;

– is_guest_login – якщо користувач має права гостя 1, інакше 0;

– count – кількість з’єднань з тією ж IP-адресою;

– srv_count – кількість з’єднань з тим самим портом;

– serror_rate – відсоток підключень, що активували flag s0, s1, s2 або s3, серед з’єднань, агрегованих у count;

– srv_serror_rate – відсоток підключень, які активували flag s0, s1, s2 або s3, серед з’єднань, агрегованих у srv_count;

– rerror_rate – відсоток підключень, що активували flag REJ, серед з’єднань, агрегованих у count;

– srv_rerror_rate – відсоток підключень, що активували flag REJ, серед з’єднань, агрегованих у srv_count;

– same_srv_rate – відсоток з’єднань тієї послуги серед з’єднань, агрегованих у count;

– diff_srv_rate – відсоток підключень до різних послуг серед з’єднань, агрегованих у count;

– srv_diff_host_rate – відсоток підключень до різних цільових машин серед з’єднань, агрегованих у srv_count;

– dst_host_count – сума підключень до тієї ж IP-адреси;

– dst_host_srv_count - сума підключень до того ж порту;

– dst_host_same_srv_rate – відсоток підключень до тієї послуги серед з’єднання, які об’єднані в dst_host_count;

– logged_in – успішний вхід;

– num_compromised – кількість скомпрометованих станів;

– root_shell - використання адміністратором командного рядка 1, інакше 0;

– dst_host_diff_srv_rate – відсоток підключень до різних послуг серед з’єднання, які об’єднані в dst_host_count;

– dst_host_same_src_port_rate – відсоток підключень до одного і того ж порту джерела, серед з’єднань, агрегованих в dst_host_srv_count;

– dst_host_srv_diff_host_rate – відсоток з’єднань, що були для різних цільових машин, серед з’єднань, агрегованих у dst_host_srv_count;

– dst_host_serror_rate – відсоток підключень, які активували flag s0, s1, s2 або s3, серед з’єднань, агрегованих у dst_host_count;

– dst_host_srv_serror_rate – відсоток підключень, які активували flag s0, s1, s2 або s3, серед з’єднань, агрегованих в dst_host_srv_count;

– dst_host_rerror_rate – відсоток підключень, що активували flag REJ, серед з’єднань, агрегованих у dst_host_count;

– dst_host_srv_rerror_rate – відсоток підключень, які активували flag REJ, серед з’єднань, агрегованих в dst_host_srv_count;

– idAttackType – ідентифікатор типу атаки.

Таблиця attacktype складається з 3 полів і містить 22 типи атак і тип нормального трафіку.

Складається з наступних полів:

– idAttackType – ідентифікатор типу атаки;

– nameAttackType – назва типу атаки (back, land, pod)

– idAttackClass – ідентифікатор класу атаки.

Таблиця attackclass призначена для зберігання даних за класами атак та класами нормального трафіку.

Складається з наступних полів:

– idAttackClass – ідентифікатор класу атаки.

– nameAttackClass – назва класу атаки (U2R, R2L, DOS, Probe, normal).

Таблиця dosset призначена для зберігання даних про dos-атаки. Вона містить у собі дані про dos-атаку та нормальний трафік з таблиці data. Складається з 17 полів, яких достатньо для виявлення атак даного класу, а саме полів idData, src_bytes, count, srv_count, serror_rate, srv_se.

РОЗДІЛ 3. ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ ВИЗНАЧЕННЯ DDOS АТАК

3.1 Розробка тестових сценаріїв

Тестування програмного забезпечення – перевірка відповідності між фактичною та очікуваною поведінкою програми, це робиться на кінцевому наборі тестів, обраному певним чином. Також тестування є одним із методів контролю якості, що містить такі дії, як керування тестуванням, проєктування тестів, виконання тесту та аналіз тесту.

Тестування – один із найважливіших етапів створення програмного продукту. Без проведення комплексного та ретельного тестування неможливо випустити якісний продукт. Крім того, відсутність тестування, особливо на ранніх етапах використання, може обернутися розробнику втратою значної кількості тимчасових та фінансових ресурсів. Відповідно, займатися тестуванням необхідно з перших стадій розробки.

У світі неможливо виділити якийсь один метод тестування, який охоплював відразу весь програмний продукт. Необхідно проводити якнайбільше видів різного тестування, не зупиняючись на якомусь одному; комбінація ручного та автоматичного тестування також допоможе виявити приховані помилки, які важко виявити.

Вибір моделі розробки ПЗ серйозно впливає на процес тестування та його місце у життєвому циклі розробки програмного забезпечення, визначаючи вибір стратегії, розклад, необхідні ресурси тощо. Моделей розробки програмного забезпечення багато, але в загальному випадку класичними можна вважати водоспадну, v-подібну, ітераційну інкрементальну, спіральну та гнучку. Не одна модель не є догмою чи універсальним рішенням. Нема ідеальної моделі. Є та, яка гірша чи найкраще підходить для конкретного проєкту, конкретної команди, конкретних умов.

Недоліки, переваги та місце тестування у життєвому циклі розробки програмного забезпечення у різних моделях розробки ПЗ розглянуті в табл. 3.1.

Таблиця 3.1 – Місце тестування у життєвому циклі розробки ПЗ

Модель	Переваги	Недоліки	Тестування
Водоспадна	• Кожна стадія має чіткий результат. • Кожного часу команда виконує один вид роботи. • Добре працює для невеликих завдань.	• Повна нездатність адаптувати проєкт до змін у вимогах. • Вкрай пізніше створення працюючого продукту.	з середини проєкту
V-образна	• Кожна стадія має чіткий результат. • Увага тестування приділяється з першої стадії. • Добре працює для проєктів із стабільними вимогами.	• Недостатня гнучкість та адаптованість. • Відсутнє раннє прототипування. • Складність усунення проблем, пропущених на ранніх стадіях розвитку проєкту.	На переходах між стадіями
Ітераційна, інкрементальна	• Досить раннє прототипування. • Простота керування ітераціями. • Декомпозиція проєкту на керовані ітерації.	• Недостатня гнучкість усередині ітерацій. • Складність усунення проблем, пропущених ранніх стадіях розвитку проєкту.	• У певні моменти ітерацій • Повторне тестування (після доопрацювання) вже перевіреного раніше
Спіральна	• Глибокий аналіз ризиків. • Підходить для великих проєктів. • Досить раннє прототипування.	• Високі витрати. • Складність застосування для невеликих проєктів.	
Гнучка	• Максимальне залучення замовника. • Багато роботи із вимогами. • Тісна інтеграція тестування та розробки.	• Складність реалізації великих проєктів. • Складність побудови стабільних процесів.	У певні моменти ітерацій та у будь-який необхідний момент

Тестування програмного забезпечення – це оцінка ПЗ відповідно до вимог, отриманих від користувачів та специфікацій системи. Тестування відбувається під час фази розробки ПЗ у його життєвому циклі або ж на модульному рівні у коді програми. Тестування ПЗ складається з валідації та верифікації [30–35]. Валідація – це процес перевірки відповідності програмного забезпечення вимогам користувача. Якщо програмне забезпечення відповідає вимогам, для яких було зроблено, воно перевіряється. Валідація гарантує, що продукт, що розробляється, відповідає вимогам користувача. Валідація наголошує на вимогах користувача.

Верифікація (перевірка) – це процес підтвердження того, що програмне забезпечення відповідає бізнес-вимогам, і воно розроблене відповідно до належних специфікацій та методологій. Перевірка гарантує, що продукт, що розробляється, відповідає проєктним специфікаціям. Верифікація відповідає питанням: “Чи розвиваємо цей продукт, суворо дотримуючись всіх проєктних специфікацій?” Перевірка концентрується на дизайні та технічних характеристиках системи.

Метою тесту є:

– Помилки – це реальні помилки кодування, зроблені розробниками. Крім того, якщо є різниця в результаті роботи програмного забезпечення та бажаного результату, то це теж вважається помилкою.

– Несправність – за наявності помилки виникає несправність. Несправність є наслідком помилки, яка може призвести до збою системи.

– Відмова - під відмовою розуміється нездатність системи виконати бажане завдання. Збій відбувається, коли у системі існує несправність.

Оцінка може бути виконана вручну або за допомогою автоматизованого інструменту тестування:

Вручну – це тестування виконується за допомогою інструментів автоматичного тестування. Тестувальник програмного забезпечення готує тестові набори для різних розділів та рівнів коду, виконує тести та повідомляє результат менеджеру. Ручне тестування вимагає багато часу та ресурсів. Тестувальник повинен удостовіритись, що суб’єкт тестування відповіда. Відповідним вимогам до продукту, який тестується. Більша частина тестування здійснюється вручну.

Автоматизоване тестування – це тестування, яке є процесом тестування, виконаним за допомогою інструментів автоматичного тестування. Обмеження при ручному тестуванні можуть бути подолані за допомогою інструментів автоматичного тестування. Тест повинен перевірити, чи можливо відкпити вебсторінку в браузері. Це можна з легкістю зробити за допомогою ручного тестування. Але щоб перевірити, чи вебсервер може витримати навантаження в 100тис користувачів, тестування вручну займе багато часу, що є нераціональним.

Існують програмні та апаратні засоби, які допомагають тестувальнику проводити навантажувальне тестування, стрес-тестування, регресійне тестування.

Підходи до тестування [32–34]:

Тести можуть проводитись на основі таких підходів:

– контекстно-орієнтоване тестування;

– дослідне тестування;

– сесійне попереднє тестування;

– функціональне тестування;

– тестування реалізації;

Під час тестування функціональності без урахування фактичної реалізації, це називається тестуванням чорної скриньки. Інакший спосіб тестування ПЗ є спосіб білої скриньки, де тестується не лише функціональність, а й спосіб її реалізації.

Вичерпні тести – вважаються найкращим методом для бездоганного тестування. В діапазоні вхідних та вихідних значень перевіряється кожне можливе значення. Неможливо перевірити кожне значення у сценарії реального часу, якщо діапазон значень занадто великий.

Тестування чорної скриньки – проводиться з метою перевірки працездатності програми. Також називається “поведінковим” тестуванням. У цьому випадку тестувальник має набір вхідних значень та відповідних очікуваних результатів. Надаються вхідні дані, і якщо вихідні дані збігаються з бажаними результатами, програма вважається “гаразд”, інакше виникає проблема.

При даному методі тестування дизайн і структура коду не відомі тестувальнику, і інженери з тестування і кінцеві користувачі проводять цей тест на рівні ПЗ.

Методи тестування чорної скриньки:

- Клас еквівалентності – при проходженні одного елементу класу можна вважати, що весь клас пройдено. Вхід ділиться на аналогічні класи.
- Граничні значення – при проходженні верхніх та нижніх кінцевих значень, передбачається, що значення, які знаходяться між ними також можуть пройти. Вхід ділиться на верхні і нижні кінцеві значення відповідно.

- Причинно-наслідковий графік – на відміну від попередніх методів, де перевіряється лише одне вхідне значення за раз, цей метод має можливість систематичного тестування комбінації вхідних значень.
- Парне тестування. Поведінка ПЗ залежить від декількох параметрів. Множинні параметри тестуються попарно відповідно їх різних значень при попарному тестуванні.
- Засноване на стані тестування – система змінює стан при наданні введення. Ці системи тестуються на основі їх станів та вхідних даних.

Тестування білої скриньки - проводиться для тестування програми та її реалізації з метою підвищення ефективності чи структури коду. Також відоме як “структурне” тестування. У цьому методі тестування дизайн та структура коду відомі тестувальнику.

Методи тестування білої скриньки:

- Тестування потоку керування. Мета тестування потоку управління для налаштування тестових випадків, що охоплюють усі оператори та умови розгалуження. Умови розгалуження перевіряються як на істинність, так і на хибність, щоб можна було охопити всі оператори.
- Тестування потоку даних – цей метод тестування акцентує на всіх змінних даних, включених до програми. Він перевіряє, де змінні були оголошені та визначені та де вони були використані чи змінені.

Саме тестування може бути визначено різних рівнів ЖЦ. Процес тестування відбувається паралельно з розробкою програмного забезпечення. Перед тим, як перейти до наступного етапу, етап перевіряється та підтверджується.

Тестування окремо проводиться тільки для того, щоб переконатися, що програмне забезпечення не залишило прихованих помилок або проблем.

Програмне забезпечення тестується на різних рівнях:

- Модульне тестування. Під час кодування програміст виконує деякі тести на модулях програми, щоб дізнатися, чи вони не містять помилок. Тестування проводиться за принципом білої скриньки. Модульне тестування допомагає розробникам вирішити, що окремі модулі програми працюють відповідно до вимог та не містять помилок.
- Інтеграційне тестування. Навіть якщо одиниці програмного забезпечення працюють окремо, необхідно з’ясувати, чи будуть одиниці,

об’єднані разом, також працювати без помилок. Наприклад, передання аргументів, оновлення даних тощо.

- Системне тестування. Програмне забезпечення компілюється як продукт, та був тестується загалом. Це може бути виконано з використанням одного або кількох таких тестів:

- Перевірка функціональності. Перевіряє всі можливості програмного забезпечення на відповідність вимогам.

- Тестування продуктивності – цей тест показує, наскільки ефективно програмне забезпечення. Він перевіряє ефективність та середній час, необхідний програмі для виконання бажаного завдання. Тестування продуктивності виконується за допомогою навантажувального тестування та стрес-тестування, коли програмне забезпечення піддається високому навантаженню користувача та даних у різних умовах середовища.

- Безпека та переносність. Ці тести проводяться, коли програмне забезпечення призначене для роботи на різних платформах та доступне для кількох людей.

- Приймальне тестування. Коли програмне забезпечення готове для передачі клієнту, воно має пройти останній етап тестування, де воно перевіряється на взаємодію з користувачем та реагування. Це важливо, тому що навіть якщо програмне забезпечення відповідає всім вимогам користувача і якщо користувач не подобається, як воно виглядає або працює, воно може бути відхиллено.

- Альфа-тестування. Команда розробників самостійно виконує альфа-тестування, використовуючи систему, якби вона існувала в готовому вигляді. Розробники намагаються визначити, як система повинна реагувати на вихідні дані та як кінцевий користувач реагуватиме на деякі дії в ПЗ.

- Бета-тестування – після внутрішнього тестування ПЗ, з ціллю остаточного тестування ПЗ, з ціллю остаточного тестування воно передається кінцевим користувачам для використання у виробничому середовищі. Це не є кінцевим продуктом. Розробники очікують, що на цьому етапі користувачі зможуть виявити проблеми, які не передбачені технічним завданням і були пропущені на попередніх етапах тестування.

- Регресійне тестування. Щоразу, коли в кінцевому продукті оновлюється програмний код або ж додається новий фукнціонал, його ретельно перевіряють, щоб визначити, чи є якийсь негативний вплив доданого коду.

Тестові документи готуються на різних етапах.

Тестування починається зі створення тестових випадків.

Наступні документи необхідні для підготовки тестової документації:

- Специфікація – документ про функціональні вимоги.
- Документ про політику тестування – описує, наскільки далеко має пройти тестування перед випуском продукту.
- Документ щодо стратегії тестування – докладно описуються аспекти команди тестування, матриця відповідальності та права/відповідальність менеджера з тестування та інженера з тестування.
- Документ матриці відстеження – це документ ЖЦ, який пов’язаний із процесом збору вимог. У міру появи нових вимог вони додаються до цієї матриці. Ці матриці допомагають тестувальникам знати джерело вимог. Їх можна простежити вперед та назад.

Наступні документи можуть знадобитися, коли тестування розпочато та виконується: Документ з описом тестових наборів (Test Case document) – цей документ містить перелік тестів, які потрібно провести. Він включає план модульних випробувань, план інтеграційних випробувань, план системних випробувань і план приймальних випробувань.

Опис тесту – цей документ є докладним описом всіх тестових випадків і процедур для їх виконання. Звіт про тестовий приклад (Test case report) – цей документ містить звіт про тестовий набір у результаті тесту. Журнали тестів (Test Logs). Цей документ містить журнали тестів для кожного звіту про тестовий приклад.

Наступні документи можуть бути створені після тестування:

- Тестовий звіт (Test summary) – це зведення тесту – є зведеним аналізом всіх звітів і журналів випробувань. Він підсумовує та робить висновок, чи готове програмне забезпечення для запуску.
- Програмне забезпечення випущено під системою контролю версій, якщо готове до запуску.
- Життєвий цикл дефекту чи життєвий цикл помилки – це особливий набір станів, якими помилка проходить від виявлення до виправлення.

Існує певна кількість станів через, які проходить дефект від проєкту до поєкту. Нижче наведено повний список статусів дефектів, що охоплює всі можливі стани Новий (New): Статус НОВИЙ присвоюється дефекту, який реєструється та публікується вперше. Призначено (Asssigned): як тільки

тестувальник створює публікацію повідомлення про помилку, виявлений дефект призначається розробнику для ліквідації. Відкрити (Open): розробник аналізує дефект та починає роботу з його виправлення. Виправлено (Fixed): після внесення необхідних змін до коду розробником, він перевіряє цю зміну, якщо це виправило помилку, він може змінити статус помилки на “Виправлено”.

Очікує повторне тестування: після виправлення дефекту, необхідно задати код для повторного тестування.

Повторне тестування. Під час цього етапу тестувальник повторно тестує код, відбувається повірка на помилки і на їх усунення. В результані отримується статус на “повторне тестування”.

Перевірено: тестер повторно тестує помилку після того, розробник виправить її. Якщо в програмному забезпеченні не виявлено помилок, помилки виправляються і статус “перевірено”.

Повторне відкриття: якщо відбувається збереження помилки після її виправлення, то статус стає “Ново виявлено”. Знову ж таки, помилка виникає протягом усього життєвого циклу.

Закрито: якщо помилку виправлено, тестер надасть статус “закрито”.

Duplicate: статус стає Duplicate, якщо відбувається повторення дефекту або дефект відповідає тій самій концепції помилки.

Відхилено: замініть дефект на “відхилено”, при умові, що дефект реальний. Відкладено: якщо поточна помилка не має пріоритету і очікується, що вона буде виправлена в наступному випуску, цьому типу помилки надається статус “Відкладено”. Не помилка: якщо це не впливає на функціональність програми, видається статус помилки – “не помилка”.

3.2. Робоча текстова документація

3.2.1. Чек-лист процесу тестування

Мають місце різного роду документів, що потрібні тестувальникам під час тестування. Найпоширенішими текстовими документами є [20–35]:

Специфікація вимого до програмного забезпечення є основою фрейворку, який буде реалізовано. Загалом вимоги описують список побажань клієнта та те, що повинен робити продукт. Деякі досі використовуються для отримання відгуків від клієнтів. Технічне завдання (ТЗ) – дозволяє повідомити про

характер контенту, який має створити ваша команда. Допомагає зрозуміти, яку функціональність повинен мати продукт, а іноді вказує на використану технологію та як вона була реалізована. Технічне завдання допомагає краще зрозуміти програму. Непорозуміння, у свою чергою, може призвести до помилок у реалізації продукту або нерозумінь щодо програмного продукту аудиторії.

Технічне завдання має бути поширене, щоб усі співробітники могли їх переглядати. У цьому відношенні технічні завдання є вигідними, оскільки нових співробітників легко залучити. Наприклад, якщо тестувальник виявить помилки чи проблеми, він чи вона зможе негайно повідомити про це. Початківець тестувальник не захоче перевіряти все, що вам потрібно і не потрібно. Тому що перше, що потрібно перевірити, це те, що має працювати на транспортному засобі.

Технічна документація – повинна включати повний опис логіки конкретної частини продукту, який формується, а також варіанти, сценарії для використання користувачем теми розробки. Така документація разом із ТЗ є хорошим джерелом знань, які допоможуть новим співробітникам зрозуміти великі проєкти. Навчання деяких систем займає тижні. Якщо є документація, співробітники можуть легко може знайти необхідну інформацію, щоб негайно почати роботу, і це є частиною вихідного коду.

Проєктна документація, підготовлена тестувальники:

Тестова документація має записувати весь процес тестування програмного забезпечення, тому що це допомагає оцінити вимого до тестування та контролювати охоплення тестами.

План тестування – це дуже важливий документ, адже містить все, що пов’язане з тестуванням. Опишіть стратегію, яка буде використовуватися для тестування програми, ресурси, які будуть використовуватися, тестове середовище, в якому проводитися тест, а також межі тесту та плану тестування. План тестування є важливою частиною будь-якого добре організованого процесу тестування. Як правило, керівник групи із забезпечення якості відповідає за написання плану тестування тп розробку дизайну тексту.

План тестування має включати наступне: вступ, обґрунтування вимог до тестування, контрольний список, перелік функцій, які підлягають тестуванню, опис методів, що використовуються при тестуванні програмного забезпечення,

перелік результатів, що підлягають тестуванню, включаючи ризики, пов'язані з тестуванням, завдання та етапи планування.

Контрольний список є частиною плану тестування, конкретного списку, який потрібно перевірити. Корисно для планування планування майбутніх і поточних дат завершення. У ньому можна відзначати, скільки часу необхідно для перевірки та скільки було витрачено. Відбиває ступінь готовності товару. Контрольний список наочно показує теперішній стан продукту, який розробляється. Зберігає історію раніше проведених тестів. Запам’ятовує, які тести треба виконати першочергово, які у другу, які у третю тощо.

3.2.2. Тест кейс

Тестовий сценарій – оповіщає про те, яку ділянку та послідовність перевірки у програмі буде здійснено [29–30]. Тестові сценарії використовуються, щоб ефективно протестувати функціонал програмного забезпечення. В залежності від об’єму та складності програми кількість тестових сценаріїв може варіюватись від одного до кількох сотень. Терміни “тестовий сценарій” та “тест кейс” можуть іноді взаємозамінятись, але тестовий сценарій має декілька етапів, в той час як тест кейс має лише один крок. З цієї точки зору, тестові сценарії – це набір тестів кейсів, які мають визначену кількість тестів та послідовність їх виконання.

Тест кейси – містять набір послідовних дій, передумов та вхідних умов, які можуть бути використані при виконанні тест кейсів. Основною умовою є забезпечення стабілної роботи програми в плані функціональності та інших аспектів. Існує велика кількість типів тест кейсів, для прикладу: функціональні, логічні помилки, негативні, багі, логічні тест кейси, фізичні тести на навантаження або проникнення в систему, кейси тестування UX/UI інтерфейсу і т.д. Також, є тест кейси для відстеження тестового покриття програмного забезпечення. Як правило, немає офіційних шаблонів, які можуть бути використані під час написання тестового випадку.

Однак наступні компоненти потрібно включати у перевірку кожного тесту кейсу: модуль продукту; версія продукту; ідентифікатор тесту кейсу; історія редакції, очікуваний результат; кроки; фактичний результат; постумови.

Тест-кейси зручно показувати як таблиці. Популярні спеціалізовані інструменти для ведення тест-кейсів: Zephyr, Jira (з інтегрованими інструментами), Redmine, Meliora, TestRail, Confluence, TFS та багато інших

Test management system tools. Матриця відповідності вимог також відома як Matrix Traceability Maturity (MTM) – це таблиця, яка використовується для відстеження вимог при життєвого циклу розробці програмного забезпечення. Вона може бути використана для прямого відстеження (тобто від вимог до дизайну або кодування) або навпаки (тобто від кодування з вимогами). Існує багато власних шаблонів для MTM (рис. 3.1).

Requirement Identifiers	Reqs Tested	REQ1 UC 1.1	REQ1 UC 1.2	REQ1 UC 1.3	REQ1 UC 2.1	REQ1 UC 2.2	REQ1 UC 2.3.1	REQ1 UC 2.3.2	REQ1 UC 2.3.3	REQ1 UC 2.4	REQ1 UC 3.1	REQ1 UC 3.2	REQ1 TECH 1.1	REQ1 TECH 1.2	REQ1 TECH 1.3
Test Cases	321	3	2	3	1	1	1	1	1	1	2	3	1	1	1
Tested Implicitly	77														
1.1.1	1	x													
1.1.2	2		x	x											
1.1.3	2	x											x		
1.1.4	1			x											
1.1.5	2	x												x	
1.1.6	1		x												
1.1.7	1			x											
1.2.1	2				x		x								
1.2.2	2					x		x							
1.2.3	2								x	x					
1.3.1	1										x				
1.3.2	1										x				
1.3.3	1											x			
1.3.4	1											x			
1.3.5	1											x			
etc....															
5.6.2	1														x

Рисунок 3.1 – Матриця відповідності вимогам

3.2.3. Bug report

Звіт про результати тестування (Bug Report) – це звіт в цифровому або письмовому вигляді про виконану роботу та її результати. Регулярно зафіксовує інформацію. До неї завжди можна буде повернутись і побачити, що саме було виконано і що саме отримали у результаті.

Для формування Bug Report також використовується ті ж Jira, Redmine, Mantis (рис. 3.2), що регулярно використовується у застосовуванні тест-кейсів.

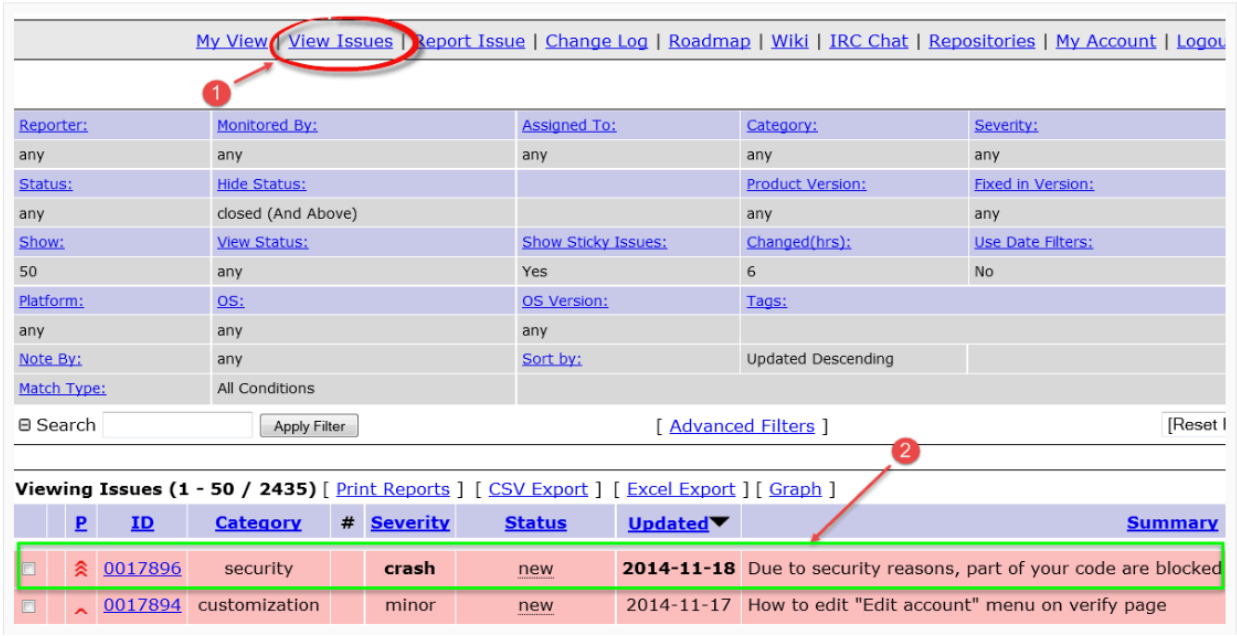

Рисунок 3.2 – Система Mantis Bug Tracker

Bug Tracking System – система автоматичного інформування про результати всіх, кому потрібно знати про дані результати. Наприклад, для розробників повідомляється про критичнв проблеми, а для співробітників відділу підтримки – надається інформація про вихід нової версії програми, що розробляється і пому подібне.

Звіт про тестування відіграє важливу роль щодо подальших дій.

3.3 Тестування програмної програми

Тестовий випадок складається з 3 частин:

PreConditions (Передумови) – це перелік кроків, які наводять систему, яка тестується в придатний для тестування стан, або перевірка умов в переліку того, що система перебуває в потрібному стані.

Test Case Description (Опис тестового випадку) – це перелік дій, за допомогою яких проводиться основна перевірка функціональності (після якої і звіряється фактичний результат з очікуваним).

PostConditions (Пустослів’я) - це передік дій, що повернуть систему у початковий стан [20–30].

Для прикладу тестування ПЗ виконаємо навчання та тестування нейронної мережі. Опис тестових випадків надано у табл. 3.2–3.4.

Таблиця 3.2 – Тест кейс навчання нейромережі

Дія	Очікуваний результат	Результат тесту
Передумови		
Відкрити сторінку Neural net	Сторінка Neural net відкрита та доступна	
Шаги тесту		
Створити файл для навчання	Повідомлення про створення файлу	Пройдено
Вибрати файл для навчання	Відображення імені файлу	Пройдено
Виконати навчання нейромережі	Виконати навчання нейромережі	Пройдено

Тепер докладно розглянемо описаний процес. Для навчання нейромережі потрібно перейти на сторінку Neural net та створити файл на навчання. Коли вже файл буде готовий, з'явиться відповідне повідомлення (рис. 3.3).

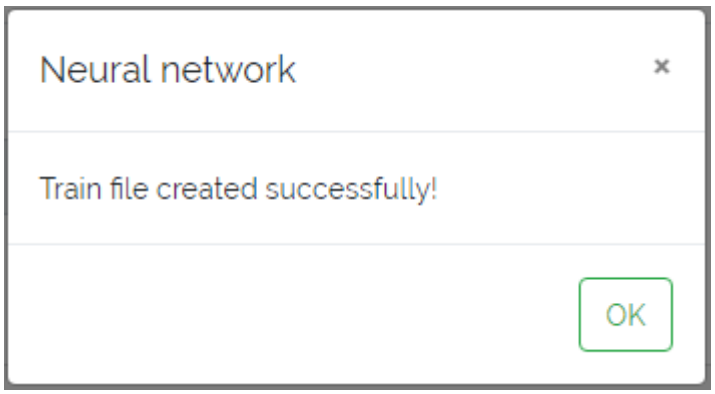

Рисунок 3.3 – Повідомлення про формування початково навчального файлу

Наступний крок – це натискання кнопки Choose Train File і вибрати створений файл. після чого він буде відображено у полі (рис. 3.4).

Шлях до файлу показуватися таким чином, тому що у браузері є політики безпеки, що забороняють JavaScript знати повний локальний шлях до файлу. Це необхідно, щоб файлова структура клієнта була недоступна серверу.

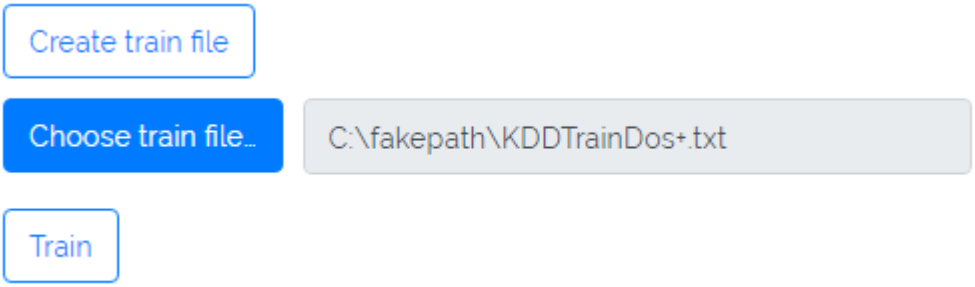

Рисунок 3.4 – Вибір файлу для навчання

Коли файл обрано, нажимається кнопка Train і після завершення навчання нейромережі з’явиться відповідне повідомлення (рис. 3.5).

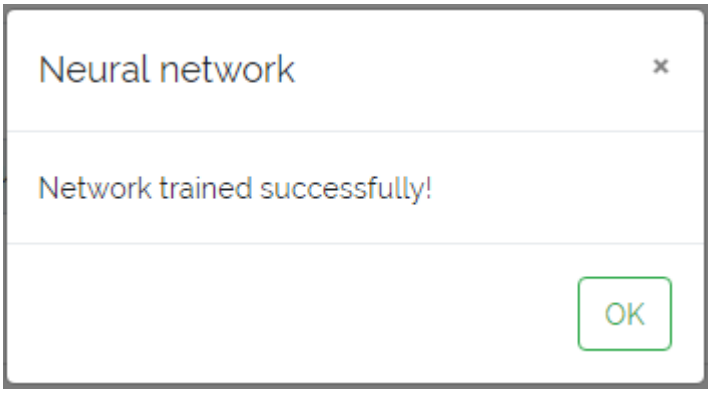

Рисунок 3.5 – Повідомлення про завершення навчання

Після навчання нейронної мережі можна перевірити якість її роботи. Опис тест кейсів для тестування нейромережі представлений у табл. 3.3–3.4.
У табл. 3.3 наведено тестовий приклад, у якому були обрані всі файли для тестування, тому тест не пройдено.

Таблиця 3.3 – Тест кейс тестування нейромережі провалено

Дія	Очікуваний результат	Результат тесту
Передумови		
Відкрити сторінку Neural net	Перейти на вкладку Test neural network. Сторінка Neural net відкрита та доступна	
Шаги тесту		
Створити файл для тестування	Повідомлення про створення файлу	Пройдено
Вибрати файл для тестування	Відображення імені файлу	Провалений
Виконати тестування нейромережі	Повідомлення про завершення тестування	Провалений
Постуслів’я		
Відкрити сторінку Home	Сторінка Home відкрита	

У табл. 3.4 було виконано всі описані кроки тесту, тому тест пройдено. Після надання тест кейсів для тестування виконаємо всі описані кроки.

Таблиця 3.4 – Тест кейс тестування нейромережі пройдено

Дія	Очікуваний результат	Результат тесту
Передумови		
Відкрити сторінку Neural net. Перейти на вкладку Test neural network	Сторінка Neural net відкрита та доступна	
Шаги тесту		
Створити файл для тестування	Повідомлення про створення файлу	Пройдено
Вибрати файл для тестування	Відображення імені файлу	Пройдено
Виконати тестування нейромережі	Повідомлення про завершення тестування	Пройдено
Постуслів'я		
Відкрити сторінку Home	Сторінка Home відкрита	

Для тестування нейромережі необхідно перейти на вкладку Test Neural Network та виконати аналогічні навчання дії. Треба створити тестові файли, натиснувши кнопку Create Test Files, після чого з'явиться повідомлення про створення файлів (рис. 3.6).

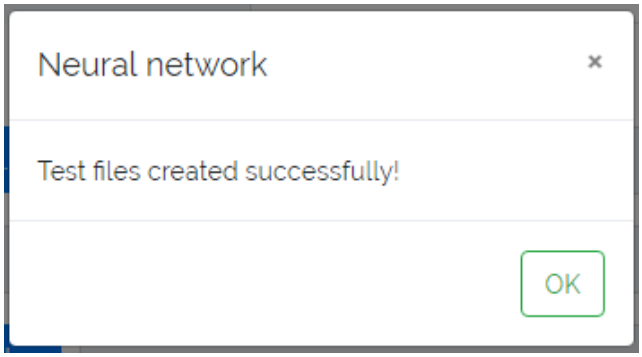

Рисунок 3.6 – Повідомлення про створення файлів для тестування

Далі необхідно вибрати тестові файли для відомих, невідомих типів атак та нормального трафіку. Якщо не будуть вибрані всі файли, відображається відповідне повідомлення (рис. 3.7).

Якщо всі файли вибрані, жодних попереджень не виникне і можна почати тестування нейронної мережі. Коли тестування закінчиться, з'явиться відповідне повідомлення (рис. 3.8).

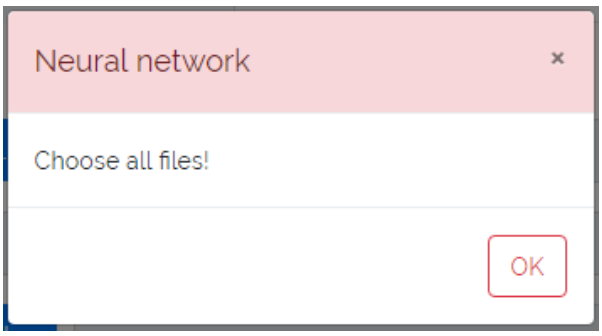

Рисунок 3.7 – Повідомлення про необхідність вибрати усі файли

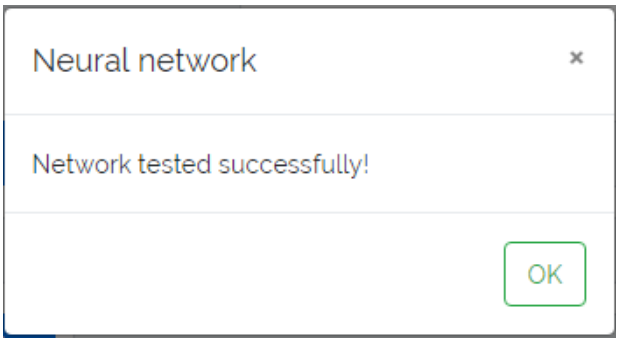

Рисунок 3.8 – Повідомлення про успішне тестування

Після цього можна перейти на сторінку Charts і переглянути результат тестування (рис. 3.9).

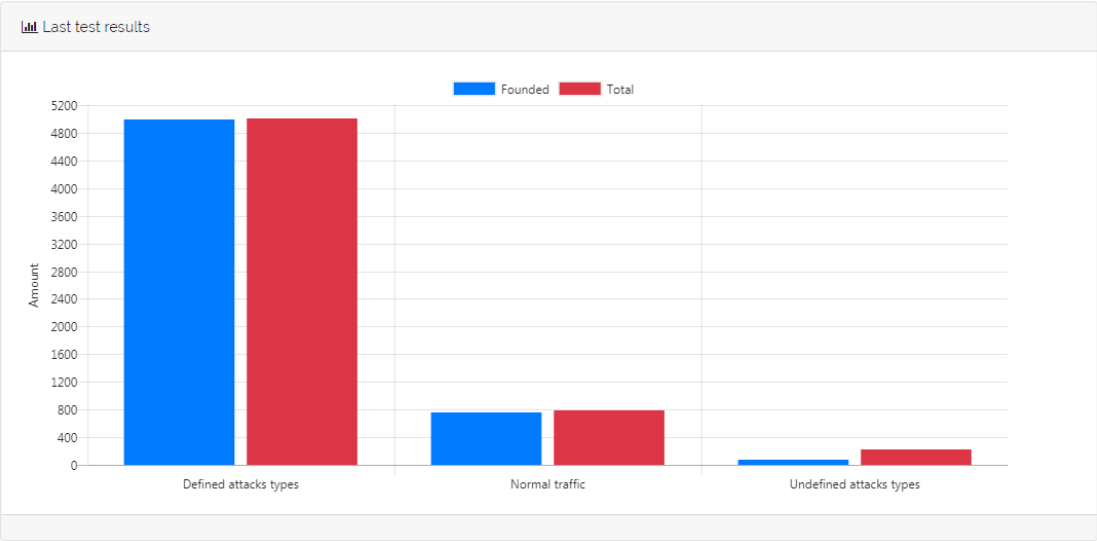

Рисунок 3.9 – Результат тестування

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Інструктажі з охорони праці на підприємстві, в організаціях

За характером і часом проведення інструктажі з питань охорони праці поділяються на вступний, первинний, повторний, позаплановий та цільовий.

Вступний інструктаж проводиться:

– з усіма працівниками, яких приймають на постійну або тимчасову роботу, незалежно від освіти, стажу роботи та посади;

– з працівниками інших організацій, які прибули на підприємство і беруть безпосередню участь у виробничому процесі або виконують інші роботи для підприємства;

– з учнями та студентами, які прибули на підприємство для проходження виробничої практики;

– у разі екскурсії на підприємство;

– з усіма вихованцями, учнями, студентами та іншими особами, які навчаються в середніх, позашкільних, професійно-технічних, вищих закладах освіти при оформленні або зарахуванні до закладу освіти.

Первинний інструктаж проводиться до початку роботи безпосередньо на робочому місці з працівником:

– новоприйнятим (постійно чи тимчасово) на підприємство;

– який переводиться з одного цеху виробництва до іншого;

– який буде виконувати нову для нього роботу;

– з відрядженим працівником, який бере безпосередню участь у виробничому процесі на підприємстві.

Проводиться з вихованцями, учнями та студентами середніх, позашкільних, професійно-технічних, вищих закладів освіти:

– на початку занять у кожному кабінеті, лабораторії, де навчальний процес пов'язаний з небезпечними або шкідливими хімічними, фізичними, біологічними факторами, у гуртках, перед уроками трудового навчання,

фізкультури, перед спортивними змаганнями, вправами на спортивних снарядах, при проведенні заходів за межами території закладів освіти;

– перед виконанням кожного навчального завдання, пов'язаного з використанням різних механізмів, інструментів, матеріалів;

– на початку вивчення кожного нового предмета (розділу, теми) навчального плану (програми) – із загальних вимог безпеки, пов'язаних з тематикою і особливостями проведення цих занять.

Повторний інструктаж проводиться з працівниками на робочому місці б терміни, визначені відповідними чинними галузевими нормативними актами або керівником підприємства з урахуванням конкретних умов праці, але не рідше:

– на роботах з підвищеною небезпекою – 1 раз на три місяці;

– для решти робіт – 1 раз на шість місяців.

Позаплановий інструктаж проводиться:

– при введенні в дію нових або переглянутих нормативних актів про охорону праці, а також при внесенні змін та доповнень до них;

– при зміні технологічного процесу, заміні або модернізації устаткування, приладів та інструментів, вихідної сировини, матеріалів та інших факторів, що впливають на стан охорони праці;

– при порушеннях працівниками вимог нормативних актів про охорону праці, що можуть призвести або призвели до травм, аварій, пожеж;

– при виявленні особами, які здійснюють державний нагляд і контроль за охороною праці, незнання вимог безпеки стосовно робіт, що виконуються працівником.

Всі вище наведені інструктажі проводяться згідно з документом «Типове положення про порядок проведення навчання та перевірки знань з питань охорони праці, від 26.01.2015 р., №15 (Зм. Від 16.11.2011 р.,№273)»

4.2 Стомлення, його причини та психофізіологічні механізми

Проблема стомлення є вельми складним науково-практичним питанням, яке досліджують представники різних наук – фізіологи, психологи.

Втома – сукупність тимчасових змін у фізіологічному і психічному стані людини, які з'являються внаслідок напруженої чи тривалої діяльності і призводять до погіршення її кількісних та якісних показників. Стан втоми залежить від звички людини до фізичного та розумового напруження. Якщо таких звичок немає, то втома може настати на самому початку роботи. Суб'єктивне відчуття втоми називається зумореністю (стомленістю).

Стомлення проявляється в різних сферах. Тому розрізняють техніко-економічні, фізіологічні, психологічні й медичні ознаки стомлення.

До числа техніко-економічних ознак утоми входять зниження виробітку, зростання браку. До фізіологічних ознак – зменшення витривалості, тремтіння у пальцях, подовження часу зорово-моторної реакції, зростання температури шкіри голови і рук, інші показники. Психологічні ознаки втоми – це відчуття зумореності, загальмованість психічних процесів, інші ознаки. Медичними показниками стомлення є травматизм і виробничо обумовлені захворювання.

Стомлення за своєю біологічною суттю є нормальним фізіологічним процесом, який супроводжується певними змінами функціонального стану і виконує захисну роль в організмі, оберігаючи його від надмірного перенапруження і можливого, у зв'язку з цим, ураження і виснаження.

У разі відсутності належного відпочинку між робочими днями розвивається перевтома, або хронічна втома. Перевтома характеризується змінами стану основних фізіологічних систем, порушенням оптимуму їх взаємовідносин, загальним падінням продуктивності праці, зниженням резистентності, творчої активності і розумової працездатності, підвищенням артеріального тиску.

Характерними ознаками перевтоми є невротичні симптоми: підвищена дратівливість, швидка стомлюваність, відсутність бажання займатися улюбленою роботою, головні болі, порушення сну.

Головні болі, як одна із основних суб'єктивних ознак перевтоми, пов'язані, як свідчать клінічні дослідження, з підвищенням внутрішньо-черепного тиску, змінами порогів збудливості механорецепторів судин головного мозку.

Виникнення та прояви перевтоми залежать від психофізіологічних особливостей людини. За наявності тих чи інших відхилень у психіці, набутих від народження чи сформованих у процесі життя, ймовірність можливості розвитку невротичного вибуху підвищується.

Хворобливі стани можуть мати місце в особистостей, які не враховують своїх сил і можливостей при плануванні фізичних і розумових завдань, тобто коли виникає конфлікт між потребою і реальною можливістю її досягнення. Типи вищої нервової системи – важливий фактор, що визначає величину працездатності конкретної людини. Слабкий тип нервової системи має порівняно невелику працездатність. Сильні типи, навпаки, характеризуються значною працездатністю.

На рівень працездатності, а відтак і на швидкість формування втоми та перевтоми, у процесі трудової діяльності суттєвий вплив має мотивація. Йдеться про сукупність матеріальних і моральних стимулів, на основі яких людина у праці ставить перед собою конкретні цілі.

Окрім мотивацій, на ступінь працездатності та на розвиток втоми і перевтоми впливає вік працівника; співвідношення праці і відпочинку; наявність в даний момент іншої домінуючої діяльності, несумісної з трудовою діяльністю людського організму.

Виділяють такі види перевтом – початковий, легкий, виражений і тяжкий, кожен з яких вимагає відповідної профілактики. Так, для зняття початкової перевтоми досить регламентувати режим праці і відпочинку. При легкому ступені перевтоми потрібно зробити нетривалий перерив у праці й

ефективно використати його для відновлення працездатності. При вираженій перевтомі слід терміново здійснити організований відпочинок, а при тяжкому ступені перевтоми – лікування.

ВИСНОВКИ

Сучасний розвиток обчислювальної техніки, формування та використання елементів інтерне-речей, дозволяє формувати кіберфізичні мережі, які, в свою чергу, можуть бути підтверджені кіберфізичним атакам. Однією з розновідностей є DDOS-атака, яка забезпечує блокування окремих програмних застосунків у захищеному домені. Розглянуто можливості реалізації кіберзагроз, їх класифікація. Проведений аналіз показав, що для своєчасного виявлення кібератак використовуються різні підходи, які дозволяють визначити або відхилення від нормальної роботи, або аномалію у роботі елементів інформаційної мережі.

Проведено аналіз нейромереж для завдання виявлення вторгнень, який дозволяє визначити основні вимоги щодо їх використання з урахуванням проєктування та визначення параметрів, які дозволяють використовувати три кращих підходу щодо виявлення DDOS-атак. Розглянуті можливості навчання нейронних мереж щодо виявлення кібератак на основі DDOS-атак.

Запропонований проєкт дозволяє забезпечити використання трьох кращих методів виявлення DDOS-атак, що дозволяє своєчасно приймати превентивні заходи щодо протидії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kyaw K.L.L. Hybrid Honeypot System for Network Security / K.L.L. Kyaw, P. Gyi //World Academy of Science, Engineering and Technology 48 2008. – Mandalay: Mandalay Technological University, 2008.

2. Balas E. Honeynet data analysis: A technique for correlating sebek and network data / E. Balas // Workshop on Information Assurance and Security United States Military Academy, West Point, NY. – IEEE., 2004.

3. Hope P. Mastering FreeBSD and OpenBSD Security / P. Hope, Y. Korff, B. Potter. – Ca.: O'Reily Media, 2005. – P.464.

4. Cox K., Gerg C. Managing Security with Snort and IDS Tools / K. Cox, C. Gerg. – Ca.: O'Reily Media, 2004. – p.288.

5. Honeytrap – A Dynamic Meta-Honeypot Daemon [Електронний ресурс]: (honeytrap) – Режим доступу: http://honeytrap.carnivore.it/

6. Nebula – Generating Syntactical Network Intrusion Signatures / Werner T., Fuchs C., Gerhards-Padilla E., Martini P. // Lecture Notes in Computer Science. – B.: Springer Berlin, 2005. – Vol.3245. – P.105-113.

7. Kreibich C. Honeycomb – creating intrusion detection signatures using honey-pots / C. Kreibich, J. Crowcroft // Second Workshop on Hot Topics in Net-works (Hotnets II). – Boston, 2003.

8. Newsome J. Polygraph: Automatically generating signatures for polymorphic worms / J. Newsome, B. Karp, D. Song // Proceedings of the 2005 IEEE Sym-posium on Security and Privacy. – Washington: IEEE Computer Society, 2005.

9. Automated worm fingerprinting / Singh S., Estan C., Varghese G., Savage S. // OSDI. – San Diego.: University of California, 2004. – P.45–60.

10. Improving Web Application Security: Threats and Countermeasures 1st Edition, [eBook] / Microsoft Corporation – Microsoft Press; 1st edition (September 5, 2003), English, 960 pages

11. Web Application Vulnerabilities And Prevention, [eBook] / Ms. Amrita Mitra – Independently published (August 19, 2019), 142 pages

12. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws 2nd Edition, Kindle Edition, [eBook] / by Dafydd Stuttard, Marcus Pinto – Wiley; 2nd edition (August 31, 2011), 912 pages

13. Hacking: The Art of Exploitation, 2nd Edition 2nd Edition, Kindle Edition, [book] / by Jon Erickson No – Starch Press; 2nd edition (February 1, 2008), 681 pages

14. Hands-On Python with 162 Exercises, 3 Projects, 3 Assignments & Final Exam: BEGINNER, [eBook] / by Musa Arda – October 15, 2021, 1011 pages

15. Introducing Python: Modern Computing in Simple Packages 2nd Edition, [book] / by Bill Lubanovic – O'Reilly Media; 2nd edition (December 3, 2019), 630 pages

16. CompTIA Security+ Study Guide: Exam SY0-601 8th Edition, Kindle Edition, [eBook] / by Mike Chapple, David Seidl – Sybex; 8th edition (January 5, 2021), 642 pages

17. Mike Meyers' CompTIA Security Certification Guide, Third Edition (Exam SY0-601) 3rd Edition, Kindle Edition, [eBook] / by Mike Meyers, Scott Jernigan – McGraw-Hill Education; 3rd edition (May 7, 2021), 830 pages

18. Learning Angular: A no-nonsense beginner's guide to building web applications with Angular 10 and TypeScript, 3rd Edition 3rd Edition, Kindle Edition, [eBook] / by Aristeidis Bampakos, Pablo Deeleman – Packt Publishing; 3rd edition (September 7, 2020), 430 pages

19. Pro Angular 9: Build Powerful and Dynamic Web Apps 4th Edition, Kindle Edition, [eBook] / by Adam Freeman – Apress; 4th edition (June 12, 2020), 1367 pages

20. Angular Development with TypeScript 2nd Edition, Kindle Edition, [eBook] / by Anton Moiseev, Yakov Fain – Manning; 2nd edition (December 5, 2018), 560 pages

21. Yegneswaran V. An architecture for generating semantic-aware signatures / V. Yegneswaran, J.T. Giffin, S.J. Paul Barford // Proceedings of the 14th USENIX Security Symposium, 2005. – P.97–112.

22. Kim H.A. Autograph: Toward automated, distributed worm signature detec-tion / H.A. Kim, B. Karp // Proceedings of the USENIX Security Symposium, 2004. – P.271–286.

23. Tang Y. Defending against internet worms: A signature-based approach / Y. Tang, S. Chen // Proceedings of IEEE INFOCOM’05, 2005.

24. Wang S.S.K. Anomalous payload-based network intrusion detection / S.S.K. Wang //Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection. – Sophia Antipolis, 2004.

25. Liang Z. Fast and automated generation of attack signatures: A basis for building selfprotecting servers / Z. Liang, R. Sekar // Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS). – Alex-andria, 2005.

26. Smirnov C.T. Dira: Automatic detection, identification, and repair of control-hijacking attacks. / C.T. Smirnov // Proceedings of NDSS05: Network and Dis-tributed System Security Symposium Conference Proceedings. – San Diego, 2005.

27. Detection of injected, dynamically generated, and obfuscated malicious code / Rabek J.C., Khazan R.I., Lewandowski S.M., Cunningham R.K. // Proceedings of the 2003 ACM workshop on Rapid Malcode. – New York, 2003. – P.76–82.

28. Crandall J.R. Experiences using minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities / J.R. Crandall, S.F. Wu, F.T. Chong // Proceedings of DIMVA'05, 2005. – P.32–50.

29. Lam L., Chiueh T. Automatic extraction of accurate application-specific sandboxing policy / L. Lam, T. Chiueh // Lecture Notes in Computer Science. – B.: Springer Berlin, 2004. – Vol.3224. – P.1– 20.

30. Security for structured peer-to-peer overlay networks / Castro M., Druschel P., Ganesh A., Rowstron A., and Wallach D.S.// Proceedings of the 5th Sympo-sium on Operating Systems Design and Implementation (OSDI'02). – Boston, 2002.

31. Honeystat: Local worm detection using honeypots / Dagon D., Qin X., Gu G., Lee W., Grizzard J.B., Levine J.G., and Owen H.L. // Lecture Notes in Computer Science. – B.: Springer Berlin, 2004. – Vol.3224. – P. 39–58.

32. Database Mirroring в SQL Server. URL: https://docs.microsoft.com/en-us/sql/database-engine/database-mirroring/prepare-a-mirror-database-for-mirroring-sql-server?view=sql-server-ver15

33. Web Application Architecture: Principles, Protocols and Practices 2nd Edition, Kindle Edition [eBook] / by Leon Shklar, Rich Rosen - Wiley; 2nd edition (May 19, 2009), 781 pages

34.Web Application Security, A Beginner's Guide 1st Edition, Kindle Edition [eBook] / by Bryan Sullivan, Vincent Liu - McGraw-Hill Education; 1st edition (December 6, 2011), 353 pages

35.Cyber and Penetration Testing for Web Applications [eBook] / by Roman Zaikin - April 29, 2019, 325 pages

ДОДАТКИ

```
def create_model(lyrs=[10], act='linear', opt='Adam', dr=0.0):

    # set random seed for reproducibility
    seed(42)
    tensorflow.random.set_seed(42)

    model = Sequential()

    # create first hidden layer
    model.add(Dense(lyrs[0], input_dim=X_train.shape[1], activation=act))

    # create additional hidden layers
    for i in range(1,len(lyrs)):
        model.add(Dense(lyrs[i], activation=act))

    # add dropout, default is none
    model.add(Dropout(dr))

    # create output layer
    model.add(Dense(1, activation='sigmoid'))  # output layer

    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['accuracy'])

    return model
```

Рисунок А.1 – Програмний код для побудови моделі нейронної мережі

```
# train model on full train set, with 80/20 CV split
training = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2, verbose=0)
val_acc = np.mean(training.history['val_acc'])
print("\n%s: %.2f%%" % ('val_acc', val_acc*100))
```

Рисунок А.2 – Програмний код для навчання нейронної мережі