Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Підвищення стійкості ідентифікації вторгнень у комп'ютерну систему на основі нейронних мереж»

Виконала: ст. 2-го курсу, групи УБ-21м
спеціальності 125– Кібербезпека
Освітня програма – Управління інформаційною безпекою
(шифр і назва напряму підготовки, спеціальності)

Гайдей В. О.
(прізвище та ініціали)

Керівник: д.ф., ст. викл. каф. МБІС
Салієва О. В.
(прізвище та ініціали)

«15» грудня 2022 р.

Опонент: к.т.н., доц., доцент каф. ОТ
Войцеховська О.В
(прізвище та ініціали)

«15» грудня 2022 р.

Допущено до захисту
Голова секції УБ кафедри МБІС

Юрій ЯРЕМЧУК
"15" грудня 2022 р.

Вінниця ВНТУ - 2022 рік

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

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Управління інформаційною безпекою

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
_____ **Юрій ЯРЕМЧУК**
“15” вересня 2022 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Гайдей Владлені Олександрівні
(прізвище, ім`я, по-батькові)

1. Тема роботи: «Підвищення стійкості ідентифікації вторгнень в комп’ютерну систему на основі нейронних мереж»
Керівник роботи: Салієва Ольга Володимирівна, к.т.н., доцент
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «14» вересня 2022 року № 203

2. Строк подання студентом роботи: до 10 грудня 2022

3. Вихідні дані до роботи: матеріали та напрацювання, здійснені під час переддипломної практики, вимоги законодавства в сфері захисту інформації, технічне завдання 08-72.МКР.003.00.000.ТЗ

4. Зміст текстової частини: вступ, основна частина (розділ 1: аналіз способів застосування технологій нейронних мереж в системах виявлення вторгнень, розділ 2: підвищення стійкості ідентифікації вторгнень в комп’ютерну систему за рахунок глибинної нейромережі , розділ 3: програмна реалізація системи вторгнень, розділ 4: економічна доцільність розробки ПЗ, висновки, список використаних джерел, додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень): рисунки, таблиці, презентаційний матеріал у вигляді слайдів.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Отримання завдання	15.09.2022	
2.	Пошук та аналіз науково-технічної літератури, наукових публікацій, інших достовірних джерел інформації	15-20.09.2022	
3.	Збір та підготовка матеріалів під час переддипломної практики на підприємстві, ознайомлення із спеціальною та технічною документацію, експериментальні дослідження, практична реалізація обраних рішень	01-30.09.2022	
4.	Готовність основної частини	до 24.11.2022	
5.	Попередній захист на кафедрі	24-25.11.2022	
6.	Перевірка магістерської кваліфікаційної роботи на наявність ознак академічного плагіату	28-30.11.2022	
7.	Подача роботи опоненту, отримання відгуку	05-09.12.2022	
8.	Перевірка керівником, отримання відгуку	05-09.12.2022	
9.	Фінальна перевірка	до 10.12.2022	
10.	Захист	19, 21.12.2022	

Студент _____ **В.О. Гайдей**
(підпис)

Керівник роботи _____ **О.В. Салієва**
(підпис)

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень): рисунки, таблиці, презентаційний матеріал у вигляді слайдів.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Салієва О.В., доктор філософії, старший викладач кафедри МБІС		
Економічна частина	Лесько О.Й., завідувач кафедри економіки підприємства і виробничого менеджменту ФМІБ ВНТУ		

7. Дата видачі завдання 15 вересня 2022 р.

АНОТАЦІЯ

Магістерська кваліфікаційна робота присвячена вирішенню науково-практичного завдання щодо підвищення стійкості ідентифікації вторгнень у комп'ютерну систему на основі нейронних мереж.

Сьогодні системи виявлення вторгнень відіграють важливу роль у безпеці системи в цілому. В ідеалі вони здатні виявляти кібератаки в реальному часі. Системи виявлення вторгнень можуть бути реалізовані за допомогою різних підходів до виявлення вторгнень з їх перевагами та обмеженнями. В магістерській кваліфікаційній роботі наведено огляд сильних сторін і обмежень різних підходів до виявлення атак, у тому числі статистичних аномалій, зіставлення шаблонів, інтелектуального аналізу даних і машинного навчання.

Результати показують, що машинне навчання є найбільш ефективним підходом для впровадження систем виявлення вторгнень, завдяки тому, що воно являє собою автоматизований процес, який майже не потребує втручання людини. Тому в роботі подані різні методи машинного навчання та зроблено висновок, що найкращою технологією для підвищення стійкості ідентифікації вторгнень у комп'ютерну мережу є застосування нейромереж. За допомогою цієї технології можна розробити та впровадити системи виявлення вторгнень, які майже не потребують людського втручання.

Ключові слова: несанкціоновані дії, глибинна нейромережа, захист системи, аналіз вторгнення.

ABSTRACT

The master's thesis is devoted to the solution of practical scientific and technical tasks - increasing the reliability of identification of unauthorized actions and attacks in computer networks.

Today, intrusion detection systems play an important role in the security of the system as a whole. Ideally, intrusion detection systems are capable of detecting cyber attacks in real time. Intrusion detection systems can be implemented using different approaches to intrusion detection, with their advantages and limitations. The master's thesis provides an overview of the strengths and limitations of various approaches to attack detection, including statistical anomalies, pattern matching, data mining, and machine learning.

The results show that machine learning is the most suitable approach for the implementation of intrusion detection systems due to its ability to operate as an automated process that requires almost no human intervention. Therefore, the paper presents various methods of machine learning and concludes that the best technology for increasing the stability of identification of computer network intrusions is the use of neural networks. With this technology, it is possible to design and implement intrusion detection systems that require almost no human intervention.

Keywords: unauthorized actions, deep neural network, system protection, intrusion analysis.

ЗМІСТ

ВСТУП

Актуальність теми.

Комп’ютерні системи вразливі до вторгнень. Вторгнення - це отримання несанкціонованого доступу до системи з метою порушення її безпеки. Мета кібератаки – отримати несанкціонований доступ до системи задля отримання конфіденційної інформації. Зловмисники можуть спробувати скомпрометувати доступність, цілісність і конфіденційність інформації в системі.

Вважається, що поведінка зловмисників відрізняється від поведінки авторизованого користувача. Різниця у поведінці між авторизованим користувачем та зловмисником дозволяє виявляти зловмисників за допомогою різних методів. Виявлення вторгнень - це акт виявлення дій і поведінки, які намагаються порушити цілісність, конфіденційність або доступність комп’ютерного ресурсу. Виявлення вторгнень здійснюється за допомогою системи виявлення вторгнень, яка є системою безпеки або програмним забезпеченням, що виявляє дії та поведінку, які відрізняються від «нормальної» поведінки, яка зазвичай відбувається в системі.

Система визначається як «система безпеки, яка відстежує комп’ютерні системи та мережевий трафік та аналізує цей трафік на наявність можливих кібератак. Метою системи є виявлення вторгнень у реальному часі та реагування на них відповідним чином до того, як зловмисник отримає конфіденційну інформацію або завдасть будь-якої шкоди системі. Характеристики системи включають: мінімальний нагляд із боку людини, здатність оновлюватися за допомогою автоматизованого процесу, високу точність виявлення атак, здатність виявляти всі атаки та мати можливість швидко реагувати.

Існує кілька підходів до виявлення вторгнень, які можна використовувати для реалізації системи. Ці підходи включають статистичні аномалії, зіставлення шаблонів, інтелектуальний аналіз даних і застосування підходу машинного навчання. Зокрема, варто зосередити увагу на підвищенні стійкості ідентифікації вторгнень в комп’ютерну систему.

Вирішенням даного питання може бути використання апарату штучних нейронних мереж. Адже завдяки навченим штучним нейронним мережам легше аналізувати вторгнення в комп'ютерну мережу.

Метою і задачею роботи є підвищення стійкості ідентифікації вторгнень в комп'ютерну систему за рахунок штучної нейронної мережі.

Для досягнення мети дослідження були поставлені та вирішені наступні основні завдання:

1. Проаналізувати проблеми безпеки мережі.

2. Проаналізувати існуючі методи виявлення вторгнень в мережу і вибрати прототип.

3. Вдосконалити методологію за рахунок впровадження процесу донавчання.

4. Провести комплексне моделювання та аналітичну оцінку запропонованого рішення.

Об'єкт дослідження – процес застосування глибокого навчання для підвищення стійкості ідентифікації вторгнень в комп'ютерну систему.

Предмет дослідження – процес застосування глибинної нейромережі для виявлення вторгнень в комп'ютерну систему.

Методи дослідження: методи матаналізу для створення математичної моделі нейроподібної мережі, апарат штучних нейронних мереж для створення структури нейроподібної мережі, методи контрольованого та неконтрольованого навчання штучних нейронних мереж для розробки підконтрольного навчання для мережі Кохонена.

Наукова новизна. Підвищення стійкості ідентифікації вторгнень в комп'ютерну систему за рахунок використання розробленої архітектури глибинної нейронної мережі.

Практична цінність одержаних результатів.

Розроблено програмний продукт для реалізації системи автоматизованого тестування стійкості ідентифікації вторгнень у комп'ютерну систему на основі глибинної нейромережі.

1. АНАЛІЗ СПОСОБІВ ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ НЕЙРОННИХ МЕРЕЖ В СИСТЕМАХ ВИЯВЛЕННЯ ВТОРГНЕНЬ

1.1 Огляд існуючих технологій нейромереж для виявлення вторгнень в комп'ютерну систему

На сучасному етапі найчастіше пропонують системи виявлення вторгнень у комп'ютерну мережу, що побудовані на основі таких штучних нейронних мереж як багатошарового персептрона, мережі Кохонена або самоорганізованої карти та їх комбінацій [1].

Завдання виявлення вторгнень, зазвичай, вирішуються за допомогою: експертних систем, штучної нейронної мережі, штучної імунної системи, нечітких систем та генетичних алгоритмів.

Розглянемо детальніше інструменти штучного інтелекту для виявлення вторгнень.

Експертні системи найчастіше використовуються в системах виявлення вторгнень. Цей факт можна пояснити тим, що сигнатурний метод аналізу мережевого трафіку є найшвидшим і не вимагає великої обчислювальної потужності. Найбільш відомими представниками систем виявлення вторгнень на основі експертних систем є: Snort, Tripwire, IBM ISS, McAfee.

Найбільшим недоліком експертних систем як систем виявлення вторгнень є те, що вони в принципі не можуть виявити нові види атак. Крім того, відомо багато методів обходу експертних систем виявлення вторгнень, таких як поліморфний код оболонки, вставка, виключення тощо.

Штучна нейронна мережа – це математична модель та її програмна або апаратна реалізація, заснована на принципах організації та функціонування біологічної нейронної мережі [2]. Обмежене використання нейронних мереж у системах виявлення вторгнень пояснюється великою обчислювальною потужністю, необхідною для роботи як NIDS (системи виявлення вторгнення в мережу) у великих корпоративних мережах, і нездатністю швидко аналізувати великі обсяги даних.

Нейронні мережі дозволяють вирішувати практичні завдання, пов'язані з розпізнаванням та класифікацією образів. Нейронна мережа складається з взаємозалежних нейронів, що утворюють вхідний, проміжні (приховані) та вихідний шари. Навчання відбувається шляхом коригування значень ваги нейронів для мінімізації помилки класифікації.

Перевагами нейронних мереж є їхня здатність автоматично набувати знань у процесі навчання, а також здатність до узагальнення, основний недолік – чутливість до шуму у вхідних даних.

На сучасному етапі найчастіше пропонують системи виявлення мережевих атак, що побудовані на основі таких нейронних мереж (НМ): багатошарового персептрона [1, 3-6], мережі Кохонена або самоорганізованої карти [7, 10] та їх комбінацій [2-4, 7].

Багатошаровий персептрон (Multi Layer Perceptron, MLP). А. В. Крижановський за допомогою програми Statistic Neural Networks створив НМ, що допускає середню квадратичну помилку під час виявлення атаки 0,006826 % [2]. А. Г. Мустафаев за допомогою нейропакета Neural Network Toolbox програми MatLAB створив НМ, що в 93 % випадків правильно класифікує атаки [2]. І. В. Жуковицький і В. М. Пахомова за допомогою програми Fann Explorer створили НМ конфігурації 19–1–25–5 (19 – кількість початкових нейронів; 1 – кількість прихованих шарів; 25 – кількість прихованих нейронів; 5 – кількість результуючих нейронів), що дозволить у реальному часі виявити загрози Back, Buffer_overflow, Quess_password, Ipsweep, Neptune на комп’ютерну мережу [9].

Самоорганізована карта (Self Organizing Maps, SOM). Gunes Kayachik, Nur Zincir–Hejvud і Malkolm Hejvud [10] під час використання бази даних KDD99 з’ясували, що система виявляє атаки U2R і R2L в 90,4 % випадків, але при цьому помилкове виявлення атаки складає 1,38 %. Palomo та Esteban [6] провели дослідження на основі зростаючої ієрархічної самоорганізованої карти із застосуванням бази KDD99 [14]. У результаті визначено майже максимальний рівень виявлення атаки, що складає 99,99 %, але при цьому є суттєвий недолік, який полягає у великій імовірності помилки у 5,44 % порівняно із іншими НМ.

Ortis Andres [7] також проводила дослідження із застосуванням зростаючої ієрархічної самоорганізованої карти. У результаті була створена НМ із підвищенням швидкості виявлення атак, яка для навчання використовувала метод маркування ймовірностей. Для цієї НМ використано базу NSL–KDD. Досягнуто найвищу частоту виявлення атак 99,68 % із найменшим помилковим спрацьовуванням 0,02 %. Отже, краща загальна продуктивність, але рівень виявлення U2R–атак найгірший серед усіх інших атак. Згідно з [9], наявні системи виявлення вторгнень на основі самоорганізованих карт мають труднощі, які пов'язані з тривалим часом обчислень та низькою частотою виявлення атак U2R і R2L. Щоб подолати ці труднощі, кафедра обчислювальної техніки інституту інженерних технологій у запропонованій моделі використала підхід зі зростаючою ієрархічною самоорганізованою картою з метою збільшення часу обчислень і відповідних функцій для атак U2R і R2L та підвищення продуктивності. У результаті на такій моделі приблизно на 75 % збільшується частота виявлення атак U2R і R2L порівнянно з самоорганізованими картами.

Комбінований підхід. Для виявлення DDoS-атак у роботі [3] застосовано SOM і MLP. За допомогою самоорганізованої карти відбувається кластеризація 50-символьних подій у вузли матриці, у яких згруповані події аналогічних числових символів. Фактично окремі вузли будуть являти собою певні сценарії атак. Після цього дані заголовків пакетів та інформація про групування подаються на вхід багатошарового персептрона, навченого розпізнавати аномальний трафік, але вже з урахуванням інформації про подію, тобто належності пакета до тієї чи іншої групи. У результаті тестування НМ отримані результати: помилка першого роду (помилкове спрацьовування) склала 3,16 % , а помилка другого роду (пропуск атаки) – 1,23 %. Технологічний університет імені короля Монгкута [5] провів подібний експеримент для виявлення таких типів атак: Neptune, Port Sweep і Satan. У ході експерименту поділено 121820 навчальних шаблонів даних порівну на 8 наборів. Кожен набір об'єднується в мережу самоорганізованої карти, далі у тришаровий персептрон, що складається з 70 нейронів у першому шарі, 12 нейронів у другому (прихованому) шарі та 4

нейронів у результуючому шарі. У результаті НМ в 90 % випадків виявляє атаку і має менше 5 % помилкових спрацьовувань.

Хафізов А. Ф. запропонував використовувати гібридні нейронні мережі для аналізу атакованих пітограм [11]. На першому етапі гібридної штучної нейронної мережі шар Кохонена навчається на наборі вхідних векторів. У результаті нейрони в цьому шарі самоорганізуються таким чином, що їхні вагові вектори найкраще представляють розподіл цих тренувальних векторів. Далі вагові коефіцієнти фіксуються і навчальні зразки подаються на вхід, після чого відбувається остаточне налаштування вагових коефіцієнтів нейронів. На другому етапі відбувається навчання мережі перцептрона. Навчання відбувається разом з учителем. Для даної мережі навчальний сигнал складається з вихідного сигналу шару Кохонена та вектора очікуваного значення.

Результатом цієї роботи нейронної мережі є призначення вхідних даних або класу атаки, або класу нормальної взаємодії. Ефективність системи полягає в розробці методу виявлення мережевих вторгнень, який на 15% перевищує існуючі рішення.

Генетичні алгоритми (GA) - це евристична модель пошуку для імітації процесів природного відбору. Цей евристичний підхід часто використовується для виявлення атак. GA - це свого роду еволюційний алгоритм (EA), який використовує природні методи еволюції для виявлення атак.

Під час виявлення вторгнень, зважаючи на велику кількість характеристик вихідних даних, GA може шукати підмножину необроблених функцій за допомогою Support Vector Machine (SVM), нейронних мереж (NN) або інших класифікаторів як функцій оцінювання вторгнення або його імітації. Перевага цього підходу полягає в тому, що він має гнучку та потужну можливість глобального пошуку, яка збігається з багатьох напрямків без урахування попередніх знань про поведінку системи. Основним недоліком є високе споживання обчислювальних ресурсів.

Дерево рішень є потужною моделлю (Shah, Hayat, Awan 2015), яка використовується для проблем класифікації (Singh, Nene 2013). (Rai, Devi,

Guleria 2016) описує Дерева рішень як деревоподібний граф, що складається з внутрішніх вузлів, які представляють перевірку атрибута, гілок, які є результатом перевірки, і листкових вузлів, які є міткою класу. Оскільки дерева рішень є потужними для класифікації, вони використовувалися для реалізації IDS, які можуть класифікувати вторгнення та мати можливість їх виявляти.

На ребрах «гілках» дерева прийняття рішень записані атрибути, яких залежить цільова функція, в «листях» записані значення цільової функції, а інших вузлах - атрибути, якими відрізняються об'єкти. Щоб класифікувати новий об'єкт, треба спуститися по дереву від кореня до «листка» й одержати відповідний клас, тобто. шлях від кореня до листа виступає правилами класифікації з урахуванням значень атрибутів об'єкта.

Переваги дерев ухвалення рішень - простий принцип їх побудови, хороша інтерпретованість результатів, недолік - невисока точність класифікації.

Фреймворк машинного навчання адаптований до симетрії - це нова система штучного інтелекту, яка передбачає пошук інформації для виявлення прихованих кореляцій. Він може отримувати та аналізувати інформацію зі схем ідентифікації в Інтернеті, що також виявляє наявність невидимих і нових мережевих загроз для боротьби з викликами та загрозами безпеці мережі. Різні алгоритми машинного навчання, як-от аналіз кластерів, методи асоціації та алгоритми класифікації, можуть допомогти екстраполювати та знайти спроби вторгнення для проблем безпеки технологій та інформації, таких як зловмисне програмне забезпечення, витік даних, зловмисне програмне забезпечення, APT та об’єднання даних.

Технологія опорних векторних машин (SVM) - це керований тип машинного навчання, який вирішує проблеми продуктивності класифікації атак за допомогою різних класифікаторів.

Точність системи виявлення вторгнень визначає її продуктивність. В останніх дослідженнях використовувалися різні методи для підвищення продуктивності. Основним завданням системи виявлення вторгнень є аналіз великих обсягів даних мережевого трафіку. Щоб вирішити цю проблему

знадобиться добре організована методологія класифікації. Технологія опорних векторів (SVM) і наївний Байєс - це два використовувані методи машинного навчання. Набір даних виявлення знань NSL KDD використовується для оцінювання систем виявлення вторгнень. Для виконання порівняльного аналізу використовуються опорні векторні машини та нативний Байєс, а також обчислюється їх точність і рівень помилкової класифікації.

У [12, 13] досліджується спосіб виявлення атак на основі методу опорних векторів. Метод використовувався для побудови моделі, що класифікує, з даних навчальної вибірки. Модель випробувана на атаках типу переповнення буфера, руткіт та SYN-повінь та показала актуальність застосування методу опорних векторів як основу системи виявлення атак.

1.2 Моделі глибинних нейромереж та алгоритми їхнього навчання

Вплив глибокого навчання НМ почався на початку 2000-х років, коли вже обробляли приблизно від 10% до 20% усіх чеків, виписаних у США, за словами Янна Лекуна [18] Промислове застосування глибокого навчання для широкомасштабного розпізнавання мовлення почалося приблизно у 2010 році. Алгоритми глибокого навчання працюють майже з будь-якими типами даних і потребують великої кількості обчислювальної потужності та інформації для вирішення складних завдань. Розглянемо найкращі алгоритми глибокого навчання.

Моделі глибокого навчання використовують кілька алгоритмів. Хоча жодна мережа не вважається ідеальною, деякі алгоритми краще підходять для виконання конкретних завдань. Щоб вибрати правильні алгоритми, добре мати чітке розуміння всіх основних алгоритмів.

Перший загальний, робочий алгоритм навчання для керованих, глибоких, прямих, багатошарових перцептронів був опублікований Олексієм Івахненком і Лапою у 1967 році [19] У статті 1971 року описується глибока мережа з восьми рівнів, навчених груповим методом обробки даних [20]. Інші робочі архітектури глибокого навчання, зокрема створені для комп’ютерного бачення, почалися із

Neocognitron, представленого Куніхіко Фукусімою у 1980 році.

Мережі глибокого навчання (рис. 1.1) виконують автоматичне виділення функцій без втручання людини, на відміну від більшості традиційних алгоритмів машинного навчання. Враховуючи те, що вилучення функцій - це завдання, на виконання якого команди спеціалістів із обробки даних можуть витратити роки, глибоке навчання - це спосіб обійти проблеми обмеженості експертів. Це збільшує повноваження невеликих наукових груп даних, які за своєю природою не масштабуються.

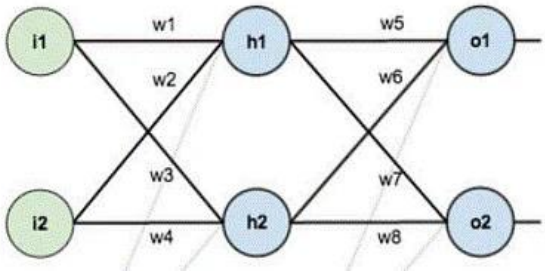


Рисунок 1.1 - Модель глибокого навчання

За допомогою класифікації глибоке навчання здатне встановити кореляції між пікселями на зображенні та іменем людини. Це можна назвати статичним прогнозом. Таким же чином, за допомогою достатньої кількості потрібних даних глибоке навчання здатне встановлювати кореляції між поточними та майбутніми подіями. Воно може здійснювати регресію між минулим і майбутнім. Майбутня подія в певному сенсі схожа на мітку. Глибоке навчання не обов'язково використовує час або той факт, що щось ще не відбулося. Враховуючи часовий ряд, глибоке навчання може прочитати рядок чисел і передбачити число, яке, швидше за все, з'явиться наступним.

Модель глибокого навчання розроблена для постійного аналізу даних із логічною структурою, подібною до того, як людина робить висновки. Щоб завершити цей аналіз, програми глибокого навчання використовують багатошарову структуру алгоритмів, яка називається штучною нейронною мережею. Прототипом для створення штучної нейронної мережі стала біологічна

мережа нейронів у людському мозку, що призводить до створення системи навчання, яка набагато ефективніша, ніж у стандартних моделей машинного навчання.

Нейронні мережі допомагають кластеризувати та класифікувати. Їх можна розглядати як рівень кластеризації та класифікації. Вони допомагають групувати немарковані дані відповідно до подібності серед прикладів вхідних даних і класифікують дані, коли мають позначений набір даних для навчання. Нейронні мережі також можуть витягувати функції, які передаються іншим алгоритмам для кластеризації та класифікації; тому можна розглядати глибокі нейронні мережі як компоненти більших програм машинного навчання, що включають алгоритми навчання з підкріпленням, класифікації та регресії. Метою використання нейронної мережі є досягнення точки найменшої помилки.

Нейронні мережі бувають різних типів і структур, включаючи рекурентні нейронні мережі, згорткові нейронні мережі, штучні нейронні мережі та нейронні мережі прямого зв’язку, і кожна з них має переваги для конкретних випадків використання. Однак усі вони функціонують однаково — надаючи дані та дозволяючи моделі самостійно визначити, чи правильно вона інтерпретувала чи прийняла рішення щодо даного елемента даних.

Глибока нейронна мережа (DNN) — це штучна нейронна мережа (ANM) із кількома рівнями між вхідним і вихідним рівнями [21-25]. Існують різні типи нейронних мереж, але вони завжди складаються з одних і тих же компонентів: нейронів, синапсів, вагових коефіцієнтів, зміщень і функцій [26]. Ці компоненти в цілому функціонують подібно до людського мозку, і їх можна навчити, як і будь-який інший алгоритм.

Штучна нейронна мережа базується на сукупності з’єднаних одиниць, які називаються штучними нейронами (аналогічно біологічним нейронам у біологічному мозку). Кожен зв'язок (синапс) між нейронами може передавати сигнал іншому нейрону. Приймаючий (постсинаптичний) нейрон може обробляти сигнал, а потім сигналізувати підключеним до нього нейронам нижче. Нейрони можуть мати стан, зазвичай представлений дійсними числами, як

правило, від 0 до 1. Нейрони та синапси також можуть мати вагу, яка змінюється в міру навчання, що може збільшити або зменшити силу сигналу, який вони надсилають на нижчі рівні [22].

Кожен прихований прошарок отримує певні ваги (випадково призначені значення). Комбінація ваг і вхідних даних надходить до функції активації, яка передається на наступний рівень для визначення виходу. Якщо не досягається очікуваний результат, обраховуються нові вагові коефіцієнти. Це ітераційний процес, доки не буде отримано прогнозований результат (метод проб і помилок). Це важливо для навчання моделі глибокого навчання, оскільки вагові коефіцієнти впливають на результат навчання.

Моделі глибокого навчання повинні бути завантажені в оперативну пам’ять і потребують значного обчислювального часу на GPU та/або CPU. На відміну від хмарних служб, обчислення на пристрої мають відбуватися під час спільного використання цих системних ресурсів з іншими запущеними програмами [21, 23]. Нарешті, обчислення мають бути достатньо ефективними, щоб обробити велику кількість вхідної інформації за досить короткий проміжок часу [26].

У багаторівневій глибокій нейронній мережі останній прошарок відіграє особливу роль. Маючи справу з позначеним входом, вихідний рівень класифікує кожне мережеве з’єднання, застосовуючи найімовірнішу мітку. Кожен вузол на вихідному рівні представляє одну мітку, і цей вузол вмикається або вимикається відповідно до потужності сигналу, який він отримує від вхідних даних і параметрів попереднього рівня.

Автокодери - це нейронні мережі, які мають здатність реконструювати вхідні дані (рис. 1.2). Вони кодують зразки даних у, як правило, менші версії самих себе, а потім декодують їх, реконструюючи зразки до їх оригінальної форми. Таким чином, якщо автокодерів навчають лише з нормальними даними, вони навчаються, як реконструювати звичайні дані так, щоб помилка реконструкції, тобто різниця між оригінальною вибіркою та її реконструкцією, була невеликою. Однак, якщо такі автокодери намагатимуться реконструювати шкідливі зразки, вони створюватимуть великі помилки реконструкції, оскільки

вони не були навчені такому типу даних. Таким чином, зразки даних оцінюються шляхом вимірювання відхилення між ними та їх реконструкції за допомогою автокодувальника, щоб великі відхилення вказували на високу ймовірність того, що шаблон даних представляє зловмисну діяльність [7, 24].

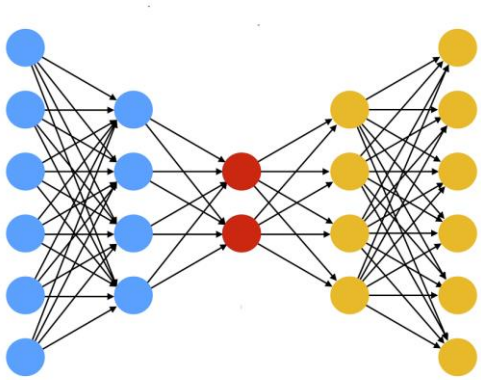


Рисунок 1.2 – Структура автокодування

Штучна нейронна мережа (ART) складається з набору елементарних елементів, які взаємопов'язані один з одним і перетворюють набір вхідних даних до набору бажаних вихідних даних. Результат перетворення визначається характеристиками елементів та вагами, що відповідають взаємозв'язкам між ними [28]. Шляхом зміни з'єднань між вузлами мережі можна адаптуватися до бажаних вихідних результатів.

Для збору мережного трафіку використовується бібліотека PCap (Packet Capture) [29]. Застосування бібліотеки PCap дозволяє значно знизити час отримання та реєстрації трафіку. У роботі [30] наведено приклад використання драйвера PCap та його модифікації WinPCap. Для подальшого аналізу в роботі пропонується виділяти 41 параметр трафіку згідно з описом, складеним на конференції International Knowledge Discovery and Data Mining Tools Competition [31].

Загальні положення АРТ-2 висунуті З. Гроссбергом і докладно викладено у його роботах [30–31]. Подальше застосування мереж АРТ викладено у роботах

[32]. Основна ідея полягає в тому, що розпізнавання векторів даних, що описують різні образи, є результатом часткової або повної відповідності стану ваг одного з навчених нейронів, що розпізнають, вхідному нормалізованому вектору, тобто. входження в резонанс сенсорного та розпізнаючого прошарків мережі. Резонанс, що виник, оцінюється керуючими нейронами. Якщо він достатній, тобто, перевищує заздалегідь певний поріг, вважається, що відповідність між вектором вхідних даних і даних з пам'яті мережі встановлено. Інакше керуючий прошарок заморожує резонуючий нейрон прошарку, що розпізнається, і процедура розпізнавання повторюється. Якщо наприкінці все нейрони розпізнаючого прошарку виявляються замороженими, то в цей прошарок додається новий нейрон та його ваги навчаються в такий спосіб, щоб він з достатньою мірою відповідав вектору вхідних даних, тобто. відбувається донавчання мережі.

На рисунку 1.3 зображено схему нейронної мережі АРТ-2. Ця мережа приймає на вхід вектора дійсних чисел.

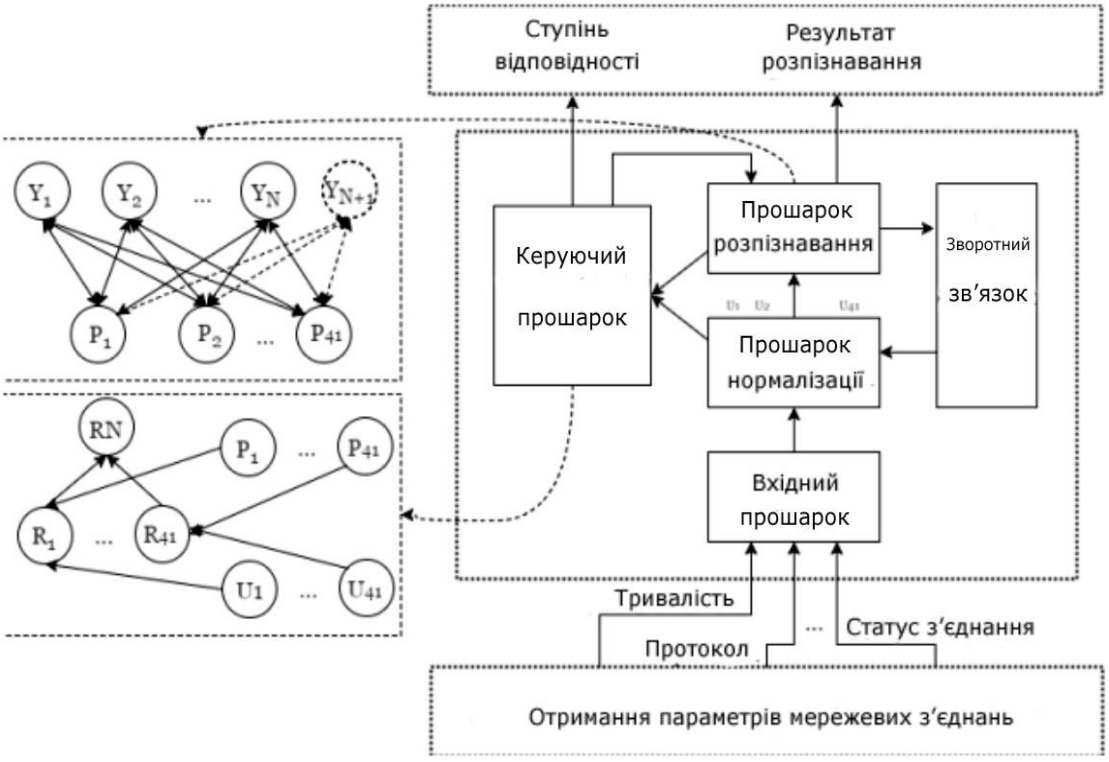


Рисунок 1.3 - Схема мережі АРТ-2 виявлення мережевих атак

Кожен вхідний вектор містить 41 параметр мережевого з’єднання, виділений з мережевого трафіку. Далі у блоці нормалізації відбувається нормування вхідних даних. Отриманий вектор поєднується з інформацією із зворотного зв'язку з прошарком, що розпізнає.

Після нормалізації дані потрапляють до групи нейронів P прошарку, що розпізнає. Далі обчислюється значення нейронів прошарку Y на підставі відповідних ваг зв'язків від P до Y:

$$Y_j = P_i x z_{ij} \tag{1.1}$$

де zij - вага зв'язку від Pi до Yj;

j = (1..N), i = (1..41);

N - кількість нейронів прошарку Y.

Серед нейронів прошарку Y визначається найбільше значення. Потім коригуються нейрони P прошарку із використанням ваги зв'язків від обраного максимального нейрона Y прошарку:

$$m = max(Y) \tag{1.2}$$

$$p_i = U_i + dxz_{mi} \tag{1.3}$$

де max() - функція, що повертає індекс максимального нейрона з прошарку Y;

i = (1..41);

U i – вихідний нейрон прошарку нормалізації;

d – константа, прийнята 0,9.

Отриманий результат оцінюється за допомогою обчислення вихідних значень групи нейронів Ri керуючого прошарку із значень вихідних векторів, що нормалізує та розпізнає прошарки:

$$R_i = U_i + cxP_i \tag{1.4}$$

де i = (1..41), c = 0,1.

Далі обчислюється норма RN вектора R і її відповідність пороговому значенню. Якщо R не задовольняє граничному значенню, то вибраний нейрон Y прошарку заморожується і процедура розпізнавання повторюється ще раз без його участі, поки не будуть заморожені всі нейрони або поки не буде знайдений такий результат розпізнавання, який задовольняв би граничне значення. Якщо в результаті повторного виконання процедури розпізнавання всі нейрони Y прошарку, що розпізнає, виявляються заморожені, то до нейронів Y прошарку додається новий нейрон і відбувається навчання ваг його зв'язків з нейронами прошарку P.

Для вирішення задачі аналізу трафіку та виявлення мережевих атак мережі АРТ-2 мають такі ключові особливості:

- можливість створення нового класу векторів, що розпізнаються, у разі невідповідності вхідного вектора жодному з існуючих класів;

- відсутність необхідності повного перенавчання мережі для додавання нової інформації;

– у вагах кожного нейрона, прошарку, що розпізнає, зберігається лише одне мережеве з’єднання, отримане в результаті виділення загальних властивостей мережевих з’єднань навчальної вибірки.

Нечіткий класифікатор. Для вирішення завдання аналізу також використовують апарат нечіткої логіки та нейронних мереж [2, 3]. Для формалізації знань експертів про кібератаку створено п'ять лінгвістичних змінних, кожна з яких характеризує один із компонентів вектора параметрів. Нагадаємо, що лінгвістичною змінною називають об'єкт виду (B, T, X, G, M) , де B - ім'я лінгвістичної змінної, T - множина найменувань нечітких змінних, що становлять лінгвістичну змінну, G - синтаксична процедура, що дозволяє оперувати елементами множини T, M - семантична процедура, що дозволяє переводити значення лінгвістичної змінної в нечітку змінну (рис. 1.4).

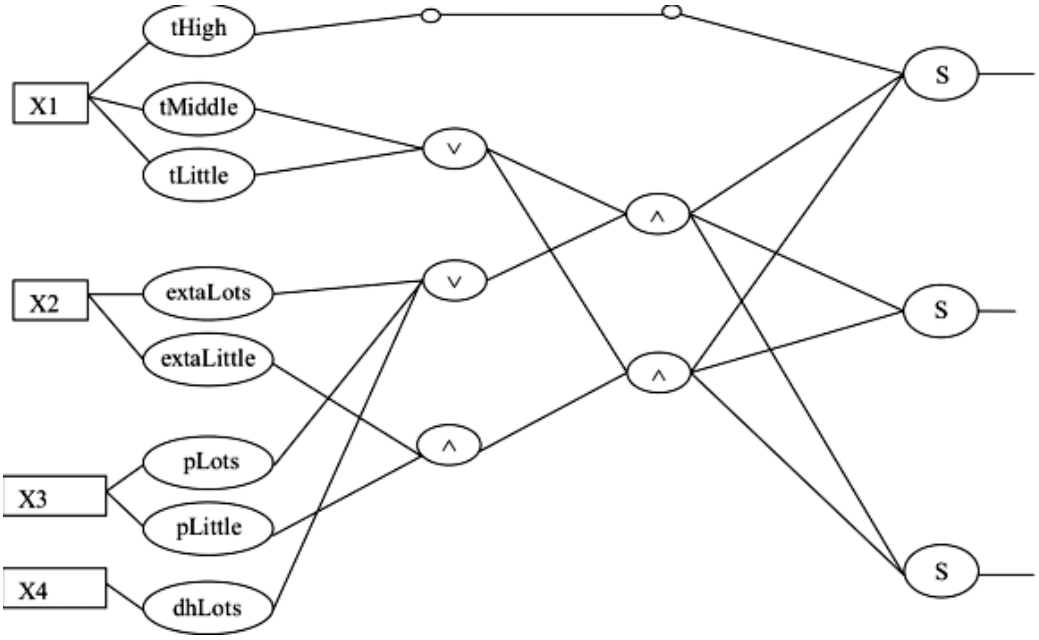


Рисунок 1.4 - Схема нечіткого класифікатора для виявлення SYN Flood атак

На схемі символом ∨ позначений нечіткий або-нейрон, символом ∧ - нечіткий І-нейрон, символом S - класичний нейрон, X1, X2, X3, X4 - відповідні входи, а tLittle, tMiddle, tHigh, extraLittle, extraLots, pLittle, pLots, dhLots - функції активації для нечітких змінних.

Таким чином, побудований нечіткий класифікатор, який повинен за даними, поданими на вхід, визначати ступінь впевненості в атаці.

Ця нечітка система може бути представлена у формі багатошарової нейронної мережі з прямим поширенням сигналу. Зазначене подання нечіткого класифікатора дає можливість використовувати такі переваги нейронних мереж, як навчання і адаптація під конкретну ситуацію. Це може бути корисним у тому випадку, коли конкретні параметри лінгвістичних змінних невідомі та підбирати їх доводиться в процесі експлуатації системи виявлення атак. Нейронна мережа, що реалізує нечіткий класифікатор, матиме структуру, ідентичну представленій на рис. 1.3. У цій мережі присутні нейрони наступних видів: І-нейрони, що обчислюють значення функції нечіткої кон'юнкції, АБО-нейрони, що обчислюють значення нечіткої диз'юнкції, нейрони, що обчислюють значення функцій належності нечітким множинам, і S-нейрони, що обчислюють вихід

нечіткого.

Для навчання цієї нейронної мережі можна використовувати будь-який градієнтний метод мінімізації цільової функції.

Оскільки вихідні дані DNN і CNN враховують лише вплив поточного введення без урахування інформації із попереднього та майбутнього часу, вони можуть досягти значної продуктивності в задачах класифікації або розпізнавання без змінних у часі характеристик. Використовуючи дані, що залежать від часу, RNN пропонується як особлива категорія структур нейронної мережі, яка розроблена з функцією «пам'яті» для збереження попередніх даних. Насправді така конструкція збігається з ідеєю, що «людське пізнання ґрунтується на минулому досвіді та пам'яті». Таким чином, RNN добре справляється з інформацією часових рядів. Проте все ще існують деякі проблеми в структурному проектуванні RNN, такі як зникнення градієнта або вибух градієнта, що призводить до нездатності запам'ятовувати або моделювати тривалу залежність.

Як правило, глибокі мережі будуються на основі мережі прямого поширення. Однак останні дослідження показали, як методи глибокого навчання можуть бути застосовані до рекурентних нейронних мереж. Згорткові нейронні мережі використовуються в області машинного зору, і цей підхід довів свою ефективність. Згорткові нейронні мережі також застосовуються для розпізнавання мови.

Навчання глибоких нейронних мереж можна здійснити за допомогою звичайного алгоритму зворотного поширення, використовуючи кілька правил обчислення вагових коефіцієнтів. Наприклад, ваговий коефіцієнт $\omega_{\mathrm{ij}}(t)$ вивчається за допомогою алгоритму стохастичного градієнтного спуску:

$$\omega_{\mathrm{ij}}(t+1) = \omega_{\mathrm{ij}}(t) + \eta \frac{\partial C}{\partial \omega_{\mathrm{ij}}} \tag{1.5}$$

де η — константа для налаштування поточного розміру кроку, а C — функція втрат. Вибір функції втрат може визначатися класом завдання

машинного навчання (з учителем, без учителя, з підкріпленням) і функцією активації.

Дві основні проблеми з глибокими нейронними мережами включають ті самі проблеми, що виникають під час навчання звичайних нейронних мереж: час навчання та перенавчання.

У мережах із глибоким навчанням кожен рівень вузлів тренується на окремому наборі функцій на основі результатів попереднього рівня.

Існує декілька основних парадигм навчання: навчання з учителем, навчання без учителя, навчання з підкріпленням.

Навчання з учителем передбачає, що набір даних містить мітки, тобто для кожного набору вхідних параметрів правильна відповідь, яку має надати нейронна мережа, відома заздалегідь. Але сам процес навчання зводиться до того, що навчальні дані по черзі надходять на вхід, а на виході мережі беруться значення і виявляється помилка, яка потім використовується для навчання мережі, з метою зменшення цієї помилки.

Навчання з нуля (без учителя). Цей метод вимагає, щоб розробник зібрав великий набір даних із мітками та налаштував мережеву архітектуру, яка може вивчати функції та модель. Ця техніка особливо корисна для нових програм, а також програм із великою кількістю категорій виводу. Однак, загалом, це менш поширений підхід, оскільки він вимагає надмірної кількості даних, через що навчання займає дні або тижні.

Навчання з підкріпленням використовується, щоб допомогти машинам опанувати складні завдання, які постачаються з масивними наборами даних, наприклад водіння автомобіля. Шляхом спроб і помилок програма вчиться приймати низку рішень, необхідних для багатьох багатоетапних процесів.

1.3 Переваги та недоліки системи виявлення вторгнення на основі нейромереж

На відміну від експертних систем (ЕС), що визначають відповідність конкретним характеристикам, закладеним у базі даних правил, нейромережа

(НМ) проводить аналіз інформації та надає можливість оцінити, що дані узгоджуються з характеристиками, які вона навчена розпізнавати. У той час як ступінь відповідності нейромережевого подання може досягати 100%, достовірність вибору залежить від якості системи в аналізі прикладів поставленого завдання (так зване навчання). Спочатку НМ навчається шляхом правильного виявлення попередньо відібраних даних предметної області. Реакція НМ аналізується та налаштовується в системі таким чином, щоб досягти задовільних результатів. На додаток до початкового періоду навчання, НМ також набирається досвіду з часом у міру того, як вона проводить аналіз даних, пов'язаних з предметною областю.

Системи виявлення вторгнень на основі НМ здатні вирішити деякі проблеми, що є в системах на основі правил. Основні переваги застосування НМ:

- гнучкість – можливість виявлення зловживань за неповними чи спотвореними даними;

- здатність виявляти маловідомі атаки, а також можливість проведення атаки розподіленої в часі;

- висока швидкість аналізу даних.

До недоліків слід віднести:

- необхідність навчання НМ, що обумовлює застосування різних методів навчання та підготовки навчальних даних (вибірки) для найкращого аналізу та виявлення зловмисної діяльності;

- суть процесів, що відбуваються всередині НМ, прихована, і якість аналізу залежить безпосередньо від навчання.

Є кілька поширених варіантів реалізації нейромереж у системах виявлення атак (СВА). Перший включає поєднання їх із існуючими експертними системами (рис. 1.5). Дане рішення використовує НМ для фільтрації вхідних даних, які можуть вказувати на зловживання, та передачі цих подій до експертної системи. Ця конфігурація має покращити ефективність системи виявлення за рахунок зниження числа помилкових спрацьовувань, властивих

ЕС. Оскільки НМ визначатиме ймовірність того, що конкретна подія вказує на атаку, порогова величина може бути встановлена там, де подія перенаправляється до ЕС для додаткового аналізу.

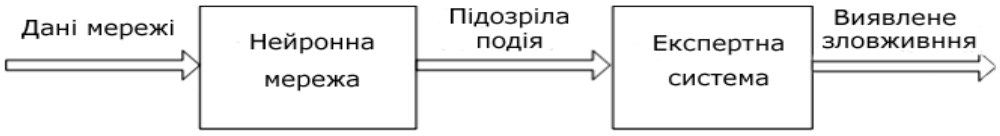


Рисунок 1.5 - Схема спільного використання НМ та експертної системи

Основний недолік даного підходу полягає у необхідності підтримання актуальності баз даних ЕС відповідно до рівня навчання НМ. Якщо експертну систему не було оновлено, то нові атаки, що виявляються НМ, будуть значною мірою пропускатися ЕС, тому що її власні правила не здатні розпізнати нову загрозу.

Другий підхід полягає у реалізації НМ як окремої системи виявлення атак (рис. 1.6). У цій конфігурації мережа отримує весь потік даних та аналізує інформацію на наявність у них зловживань. Будь-які випадки, що ідентифікуються із вказівкою на атаку, перенаправляються до адміністратора безпеки або використовуються системою автоматичного реагування на атаки.

Основні переваги цього підходу:

- висока швидкість виявлення атак у порівнянні з попереднім підходом, оскільки існує лише один рівень аналізу;

- підвищення ефективності виявлення атак з часом через навчання НМ. На відміну першого підходу, ця концепція не обмежується аналітичною здатністю ЕС.

Рисунок 1.6 - Схема застосування НМ для виявлення зловживань

НМ отримали практичне застосування у багатьох дослідженнях систем виявлення атак. Так, у роботах Дж. Райана (J.Ryan) [33] описується автономна система виявлення аномалій (Off-line anomaly detection system), в якій застосовується багатошарова НМ, що навчається за алгоритмом зворотного поширення помилки. Ця мережа навчалася на профілі користувача, що працює на комп'ютері, виявляючи у командах користувача можливі відхилення (аномалії). Для НМ була обрана 3-шарова структура з двома прихованими прошарками. НМ дозволила виявити аномалії у 22 випадках з 24.

У роботі Джеймса Кеннеді (James Cannady) [35] 3-шарова НМ застосовувалася для автономної класифікації записів мережевих з'єднань за класами нормальної та підозрілої діяльності. У роботі використовувалася вибірка з 10000 записів мережевих з'єднань, з яких 1000 записів були імітованими атаками. У процесі навчання використовувалася вибірка із 30 % записів. У результаті отримана система дозволила правильно класифікувати підозрілу діяльність у 89-91% випадках. В інших дослідженнях [34] автори застосували 3-шарові та 4-шарові НМ, отримавши результати визначення підозрілої діяльності у 99,25% випадках. Різні групи дослідників використовували у своїй роботі карти, що самоорганізуються (Self-Organized Maps) для виявлення атак [36].

У зв'язку з особливостями застосування НМ реалізація систем виявлення зловживань реального часу, заснована виключно на даному підході, у практичному плані є не простою, що не виключає застосування НМ в автономних (off-line) системах. Необхідність навчання, а також природа «чорної

скриньки» НМ обумовлює обов'язкову наявність навчальної бази даних зловживань, а також тимчасових витрат та коригування навчального процесу (вибору архітектури мережі, алгоритму навчання тощо).

На відміну від традиційних систем виявлення вторгнень, які отримують усі пакети мережі, бездротові системи роблять вибірку мережного трафіку. Стандарти сімейства 802.11 використовують два основних діапазони частот: 2,4 та 5 ГГц, які, у свою чергу, діляться на канали. WIDS забезпечують почергове сканування каналів щодо наявності активних атак.

Система працює у двох режимах:

- режим конфігурування, коли як вхідні дані в блок побудови класифікуючої моделі завантажується набір сигнатур, що представляють пару {вектор параметрів трафіку | тип атаки};

- режим нормальної роботи, коли значення параметрів трафіку подаються як вхідні дані на підсистему сенсорів. Далі підсистема прийняття рішень за допомогою побудованої на попередньому етапі класифікуючої моделі визначає, чи відповідають показання сенсорів нормальному стану або тій чи іншій атаці, і, залежно від результату, спрацьовує підсистема сигналізації.

Основою виявлення атак є база знань, побудова якої на етапі початкового конфігурування системи забезпечує блок побудови класифікуючої моделі. Класифікуюча модель будується на основі сигнатур навчальної вибірки і потім

1.4 Висновки до Розділу 1 та постановка задачі

У розділ 1 проведено аналіз методів виявлення вторгнень у комп’ютерну мережу. Розглянуто використання різноманітних методів виявлення вторгнень у мережі. Проаналізовано переваги та недоліки використання глибинної нейромережі.

Виконавши аналіз, сформулюємо мету роботи: підвищити стійкість ідентифікації вторгнень у комп’ютерну систему за рахунок штучної нейронної мережі та розробити відповідний програмний засіб.

Для досягнення мети необхідно розв’язати такі задачі:

- розробити модель несанкціонованих дій в комп'ютерних мережах на основі симптомів вторгнення;

- удосконалити діагностику вторгнень у комп'ютерні системи за допомогою глибинної нейромережі;

- здійснити програмну реалізацію системи виявлення вторгнення на основі глибинної нейронної мережі.

2. ПІДВИЩЕННЯ СТІЙКОСТІ ІДЕНТИФІКАЦІЇ ВТОРГНЕНЬ У КОМП'ЮТЕРНУ СИСТЕМУ ЗА РАХУНОК ГЛИБИННОЇ НЕЙРОМЕРЕЖІ

2.1 Аналіз можливості підвищення стійкості ідентифікації входу в комп'ютерну систему

Однією з основних тенденцій останніх років у сфері комп'ютерних злочинів є зростання кількості та складності атак на доступність інформації (ресурсів автоматизованої системи), як один із трьох основних критеріїв, поряд із конфіденційністю та цілісністю інформаційної безпеки об'єкта. Дані атаки утворюють клас атак типу «відмова у обслуговуванні»[37].

У цілому, такі атаки націлені як на мережі в цілому і серверні кластери, так і на кінцеві хости, їх завданням є максимальне споживання ресурсів з метою значного погіршення або припинення надання сервісу нормальним користувачам. Зазвичай атакованими ресурсами є: ширина каналу, процесорний час серверів та роутерів та конкретні реалізації протоколів. Як приклади можна навести SYN-атаку, націлену на переповнення стека TCP операційної системи; спрямовані широкомовні ICMP – атакованому відправляються подібні пакети, відповіді від нього знижують пропускну спроможність мережі; DNS-флуд-атаки, що використовують певну слабкість протоколу DNS та спрямовані на суттєве збільшення трафіку до атакованого.

З метою мінімізації наслідків атак, їх виявлення та класифікація є вкрай важливим та водночас складним завданням. Основний спосіб розпізнавання атаки полягає у виявленні аномалій у структурі трафіку.

Традиційні механізми забезпечення безпеки – міжмережеві екрани та системи виявлення вторгнень - не є ефективними засобами для виявлення атак та захисту від них, особливо атак трафіком великого обсягу [38].

Фундаментальною передумовою виявлення атак є побудова контрольних характеристик трафіку під час роботи мережі у штатних умовах із подальшим пошуком аномалій у структурі трафіку (відхилення від контрольних характеристик) [39]. Аномалія мережного трафіку - це подія чи умова мережі, характеризується статистичним відхиленням від стандартної структури трафіку, отриманої з урахуванням раніше зібраних профілів і контрольних характеристик. Будь-яка відмінність у структурі трафіку, що перевищує певне граничне значення, викликає спрацювання сигналу тривоги.

Натомість існуючі методи виявлення атак, що дозволяють ефективно розпізнавати атаки транспортного рівня (SYN-флуд, UDP-флуд та інші), малоефективні для виявлення низько інтенсивних кібератак прикладного рівня [40]. Даний клас атак виник порівняно недавно і на сьогоднішній день становить основну загрозу доступності інформації у розподілених комп'ютерних мережах [41]. Відрізнити трафік, що генерується в ході даних атак, від легального HTTP-трафіку досить складно, крім того, канали передачі практично не перевантажуються. Дані атаки призводять до втрат запитів та відповідей, тобто. фактичної відмови веб-серверів на основі Microsoft IIS, Apache та інших систем. Крім того, атака може бути адаптована для впливу на SMTP і навіть DNS-сервери. У варіанті низько інтенсивної атаки особливу небезпеку представляє використання техніки DNS-amplification, для лавиноподібного посилення потужності мережевого ботнета.

Ці факти зумовлюють актуальність розробки нових механізмів виявлення низько інтенсивних розподілених атак прикладного рівня в комп'ютерних мережах за допомогою методів штучного інтелекту.

2.1.1 Аналіз механізму Low-Rate DDoS

Загальним фактором, необхідним для здійснення Low-Rate DDoS-атак, є наявність великої кількості компрометованих або хостів, що добровільно беруть участь, і грубе "завалювання" пакетами атакований вузол. Саме «грубість» у реалізації даних атак може звести нанівець весь ефект у разі виявлення великих обсягів аномального трафіку мережевими екранами [39, 42].

Low-Rate DDoS-атаки є періодичним трафіком малого обсягу. У момент, коли відкрита сесія підключення повинна закритися по тайм-ауту, надсилається новий пакет для підтримки даної сесії у відкритому стані. Поступово буфер маршрутизатора або сервера переповнюватиметься, що призведе до відмови в обробці легітимного трафіку. При такому підході не потрібна велика пропускна здатність і обчислювальна потужність у атакуючої сторони.

Показовим прикладом є DDoS-атаки, спрямовані на зниження смуги пропускання TCP потоків трафіку, що здійснюються з низькою інтенсивністю, щоб уникнути виявлення. Використовуючи вразливість у механізмі тайм-ауту повторної передачі TCP-стека, можна досягти нульової пропускної здатності шляхом змішування з основним трафіком спеціально підібраних шаблонів DDoS-трафіку.

Управління смугою пропускання TCP здійснюється на 2-х тимчасових шкалах. На малій часовій шкалі позначки часу проходження пакетів каналом зв'язку до адресата і назад (RTT), зазвичай від 10 до 100 мілісекунд, TCP-стек використовує адитивно-мультиплікативе (additive-increase multiplicative-decrease) управління (AIMD) для передачі кожного потоку трафіку на однакових швидкостях через найвужче місце. Коли канал зв'язку починає «забиватися» і виникає велика кількість втрат, TCP-стек починає працювати за 2-ою більшою тимчасовою шкалою з відмітками тайм-аутів повторної передачі пакетів (RTO, рекомендоване мінімальне значення 1 секунда). Щоб уникнути «забиття» каналу, потік трафіку зменшується до одного пакета, і після часу RTO пакет пересилається заново. При подальших втратах час RTO подвоюється з кожним наступним тайм-аутом. У разі успішного отримання пакета, TCP-стек починає використовувати AIMD-управління [43].

Для проведення Low-Rate DDoS-атаки необхідно розглядати потоки трафіку у вигляді імпульсів і фіксувати періодичні імпульсні атаки, що складаються з коротких піків зі спеціально підібраною тривалістю, що повторюються з певною, спеціально обраною частотою за повільною часовою шкалою. Якщо для першого потоку TCP-трафіку загальний трафік (DDoS-атаки і звичайний) протягом піку достатній, щоб відбулися втрати пакетів, цей потік "відвалиться" по тайм-ауту і буде спроба відправити новий пакет після RTO. У випадку, якщо періодичність посилки DDoS-трафіку збігається (навіть приблизно) з RTO нормального трафіку, звичайний трафік постійно отримуватиме тайм-аут, як наслідок, втрати наближатимуться до 100 % і пропускна здатність наблизиться до нуля. Крім того, якщо період DDoS-посилок приблизно однаковий, але лежить поза діапазоном RTO, то спостерігатиметься суттєве (але не повне) зниження смуги пропускання. Більш детально даний механізм розглянутий у роботі [44], а механізм тайм-ауту TCP-стека в [45].

2.1.2 Опис системи дослідження атак типу «відмова у обслуговуванні»

Оскільки системи віртуалізації дозволяють створювати ізольовані віртуальні машини із набором виділених ресурсів, для кожної віртуальної машини виділяється процесорний час і оперативна пам'ять, при вичерпанні яких не відбуватиметься втручання в ресурси інших віртуальних машин. Також системи віртуалізації дозволяють змінювати пропускну здатність віртуальних мереж та регулювати втрати при передачі даних у цих мережах, тим самим можна створити модель, що максимально близько наближена до реальних умов у глобальних мережах.

Як апаратна частина використовувався персональний комп'ютер на базі процесора Intel Core i7-3770 з тактовою частотою 3.4 ГГц, об'ємом оперативної пам'яті в розмірі 16 Гб і твердотільний високошвидкісний жорсткий диск об'ємом 120 Гб.

На комп'ютер було встановлено «Citrix XenServer 6.1» (є продуктом Citrix, що безкоштовно розповсюджується) з активованим параметром у BIOS – апаратна віртуалізація.

Для проведення експериментів використовувався стенд у складі:

1. Атакований сервер - Debian Wheezy, web-сервер Apache, Php5 (libapache2-mod-php5), MySQL-server.

2. Атакуючий сервер - Debian Wheezy, Perl, Slowloris ddos script.

3. Сервер збору даних поведінки трафіку у мережі – Windows 7 Pro x64, WinPcap, прототип системи виявлення атак.

Для дослідження різних видів атак типу «відмова в обслуговуванні» було створено середовище, в якому для передачі, зберігання та обробки інформації використовуються мережеві протоколи різних рівнів моделі OSI.

В рамках дослідження було змодельовано атаку slowloris. Атака базується на вразливості протоколу HTTP. Slow HTTP POST атака працює наступним чином: зловмисник відправляє POST заголовок з легітимним полем Content-Length, яке дозволяє web-серверу зрозуміти, який обсяг даних до нього надходить. Як тільки заголовок відправлений, тіло POST-повідомлення починає передаватися з дуже повільною швидкістю, що дозволяє використовувати ресурси сервера набагато довше, ніж це необхідно, і, як наслідок, перешкодити обробці інших запитів. Декілька тисяч таких з'єднань можуть вивести web-сервер з ладу на кілька хвилин.

2.1.3 Виявлення атак із використанням гібридної нейронної мережі

Опишемо метод та архітектуру прототипу систем виявлення атак (СВА). Існує думка, що спеціальні засоби для виявлення атак не потрібні, оскільки факт атаки неможливо не помітити. У багатьох випадках це справді так. Проте досить часто відзначалися успішні атаки, помічені жертвами лише через 2–3 доби. Бувало, що негативні наслідки атаки (типу флуд) полягали у зайвих витратах на оплату трафіку, що з'ясовувалося лише при отриманні рахунку.

Більшість сучасних СВА здійснюють виявлення атак шляхом контролю профілів поведінки чи пошуку специфічних рядкових сигнатур. Використовуючи ці методи, практично неможливо створити повну базу даних, яка містить сигнатури більшості атак. Існує три основні причини цього:

- нові сигнатури потрібно створювати вручну. Сигнатури відомих атак, які вже включені до БД, не можуть гарантувати надійного захисту без постійних оновлень;

- теоретично існує безліч методів і варіантів атак, і для їх виявлення знадобиться БД нескінченного розміру. Таким чином, є можливість того, що атака, не включена до бази даних, може бути успішно здійснена;

- сучасні методи виявлення атак викликають велику кількість помилкових мережевих подій. Таким чином можуть бути скомпрометовані легальні мережеві події.

Для аналізу мережного трафіку в СВА використовують різні підходи. У [28] автори пропонують використовувати аналіз методів, заснованих на нечіткій логіці, але використання таких підходів передбачає виконання додаткових дій, як-от: створення лінгвістичних змінних, складання бази правил і, зазвичай, вимагає наявність експертної системи.

В [29] пропонується використовувати двоетапну обробку вхідної інформації: на першому етапі відбувається зменшення розмірності вектора вхідних даних за допомогою нелінійної рециркуляційної нейронної мережі, а на другому етапі – виявлення атак з використанням багатошарового перцептрону, який здійснює обробку стиснутого простору вхідних образів з метою розпізнавання класу атаки. Такий підхід вимагає попереднього аналізу параметрів мережі для створення нелінійної рециркуляційної нейронної мережі.

Автори статті [30] пропонують звести завдання навчання нейромережевої системи до пошуку та вилучення інформативних ознак, їх стиснення за допомогою методу головних компонентів, подальшої обробки за допомогою рециркуляційної нейронної мережі та застосування двошарового перцептрону або мережі Кохонена на базі виділених інформаційних векторів ознак. Істотним

недоліком є збільшення часу навчання мережі розпізнаванню нових видів мережевих атак.

У роботах [46, 48] показана можливість використання багатошарового персептрона в якості вирішального блоку в СВА, що аналізує послідовно мережні пакети. Пропонований у роботі підхід передбачає використання нейромережі для виявлення атак, розподілених у часі та заснованих на спільному створенні штучного (аномального) потоку даних кількома джерелами.

Розподіленими за часом називаються атаки, які проводяться протягом одного тривалого періоду часу. При спільній атаці існує кілька зловмисників, які працюють паралельно. Кожен із них окремо може робити дії, які можуть здатися нешкідливими. Атаки стають очевидними лише тоді, коли всі події розглядаються разом.

Основна перевага СВА, що використовують нейронні мережі, у тому, що нейромережа не обмежена знаннями, які заклав у неї програміст. Вони мають можливість навчатися на попередніх подіях - як на аномальному, так і нормальному трафіку. За рахунок цього досягається висока ефективність та адаптивність СВА.

Існують різні варіанти застосування нейромережевих систем виявлення атак (НСВА) – аналіз всього мережевого трафіку в сегменті мережі, аналіз команд, що вводяться користувачем, аналіз переходів станів та ін.

Приклад сильно розподіленої за часом та адресами атаки було наведено у п. 2.2. Іншим, простішим прикладом розподіленої за часом атаки може бути злам пароля шляхом перебору [44].

Потоки даних були поділені на набори зі 150 пакетів, причому крім даних із заголовків пакетів мережного та транспортного рівнів, як пропонується в роботі [40], враховувалися також і перші 50 символів корисного навантаження прикладного рівня (в ASCII-коді). Крім того, проводиться нормалізація подання значень портів призначення для уточнення прикладного протоколу, що використовується (telnet, ftp і т.д.). Такий спосіб аналізу вихідних даних було описано в [45].

Як правило, жодна подія, яка становить трафік атаки, окремо не виглядає підозрілою. Крім того, навіть дві чи три події не є чимось незвичайним. Тим не менш, взяті разом, у певній послідовності, яка повторюються в певний час, ці події є вказівкою на те, що здійснюється атака. Подальше ускладнення виявлення полягає в тому, що ці події, хоч і в певній послідовності, можуть надходити з різних джерел і чергуватись з іншими, нормальними подіями.

2.2 Обґрунтування вибору технології на основі глибинної нейромережі для запобігання вторгнень в комп'ютерну систему

Система виявлення вторгнень - це програмний або апаратний засіб, призначений для виявлення несанкціонованого доступу (вторгнення або кібератаки) до комп'ютерної системи чи мережі.

На рисунку 2.1 зображена загальна схема системи виявлення та протидії мережевим атакам у локальних обчислювальних мережах (ЛОМ).

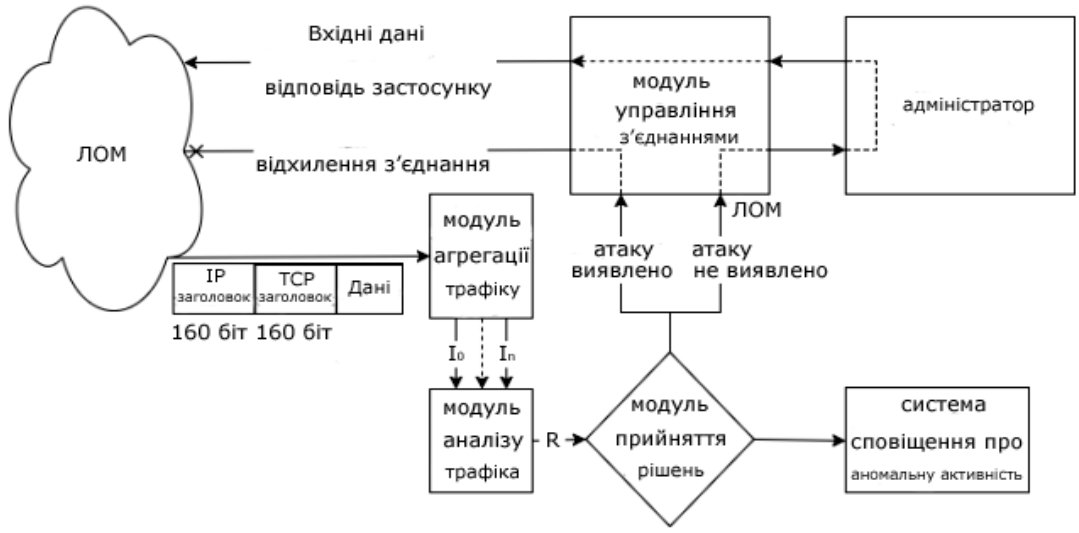


Рисунок 2.1 - Загальна схема системи виявлення мережевих атак

Модуль агрегації мережного трафіку використовується для перехоплення мережевих пакетів та їх аналізу на належність конкретному з'єднанню. Після виконання цих дій відбувається оновлення аналізованих параметрів

відповідного з'єднання.

Як відомо, ефективність методу виявлення мережевих вторгнень для системи із використанням мережі Кохонена на 15% перевищує існуючі рішення. Модуль аналізу трафіку включає нейронну мережу. Для цієї системи було обрано мережу Кохонена, що дозволяє працювати з вхідними векторами. Мережа здійснює визначення належності з'єднання до групи мережевих з'єднань, характерних для нормального або аномального стану мережі. Під аномальним станом мережі розуміється стан, у якому виконується протиправні дії, тобто. відбувається одна з мережевих атак, розглянутих у вибірці KDD99.

Результат роботи модуля аналізу передається у модуль прийняття рішень. На основі результату аналізу він або відхиляє з'єднання та повідомляє адміністратора про атаку, або дозволяє модулю управління з'єднаннями передати пакет додатку користувача для подальшої обробки. Модулі прийняття рішень утворюють розподілену систему, призначену для узгодження дій та донавчання нейронних мереж.

2.2.1 Метод виявлення низькоінтенсивних (low-rate) атак типу «відмова в обслуговуванні»

Особливістю методу є попередня кластеризація пакетів за допомогою карт Кохонена, що самоорганізуються. Вихідний вектор карти, що самоорганізується, є вхідним вектором багатошарового персептрона, який здійснює бінарну класифікацію – визначає, чи є набір мережевих пакетів нормальним або атакуючим. У результаті досягнуто помилки розпізнавання 0,84%. У статті [49] як ефективний інструмент виявлення кібератак запропоновано використовувати нейронну мережу. Для навчання та тестування нейронної мережі використовувався набір даних KDD99. Точність класифікації становила 97,87%. Зазначимо, що запропоновані підходи розраховані виявлення лише одного класу атак типу «відмова в обслуговуванні».

Значна кількість публікацій присвячена можливостям використання методів глибокого навчання виявлення і класифікації мережевих атак. У статті [50]

наводиться огляд сучасних публікацій на цю тему. У статті [51] розглянуто два завдання класифікації атак – бінарна класифікація та класифікація на 4 класи атак. Автори використовують рекурентні нейронні мережі для класифікації великого обсягу даних. В результаті для бінарної класифікації досягнуто точності менше 0,1% помилок, для класифікації за типом атак – 0,5%.

У статті [52] для виявлення атак здійснено порівняння рекурентних нейронних мереж, у тому числі LSTM мереж з традиційним методом випадкового лісу. LSTM мережі показали найвищу точність 98,4% правильного виявлення атак. У статті [50] використовуються мережі з автокодувальником зі стохастичним алгоритмом визначення порога спрацьовування. Цей метод дозволив збільшити точність виявлення атак на наборі NSL-KDD до 88,65%. У статті [53] пропонується система виявлення і класифікації як відомих, і невідомих аномалій за 4 класами. Експериментально визначено оптимальну архітектуру нейронної мережі. У статті [54] розглядається можливість автоматичної кластеризації пакетів системи виявлення аномалій в корпоративних мережах. Аномальними вважаються великі кластери з високою густиною, а також малі або розріджені кластери. Далі на цих даних навчаються алгоритми бінарної класифікації. На наборі KDD99 вдається отримати точність 88% класифікації.

У статті [50] розглядається класифікація атак у бездротових мережах IEEE 802.11. Атаки класифікуються на 3 класи за допомогою багатошарового автокодувальника. У роботі [51] для класифікації шкідливого трафіку використовуються згорткові нейронні мережі. Ідея полягає у тому, що сирі дані трафіку перетворюються на дані, які розпізнаються згортковими мережами. При цьому точність детектування атак досягає 99.41%. В роботі [52] для детектування типу атак, що важко виявляється, - сканування портів і пошуку вразливостей використовуються глибокі мережі, що комбінують підходи навчання з учителем і без вчителя. У статті [53] наведено метод бінарної класифікації атак на основі методу нечіткої кластеризації С-середніх. Для підвищення точності алгоритму використовується часткова ручна розмітка

невеликої частини навчальних даних. Детальне порівняння різних алгоритмів машинного навчання, що застосовуються у системах інформаційної безпеки, наведено у статті [54]. Автори розглядають три завдання – виявлення вторгнень, аналіз шкідливих програм та виявлення спаму. Висновки – для кожного завдання краще застосовувати свої методи, які потребують безперервного навчання та ретельного налаштування параметрів. У статті [55] розглядаються методи кластеризації виявлення вторгнень з урахуванням методу k-середніх.

Зазначимо, що методи глибокого навчання не мають високої продуктивності особливо на етапі навчання, також не досліджено методи, які дозволяють проводити класифікацію більш ніж за 4 класами атак.

2.2.2 Бінарна класифікація

Навчання класифікаторів відбувається із бібліотеки scikit-learn:

- DecisionTree – алгоритм вирішальних дерев;
- RandomForest – алгоритм випадкового лісу;
- AdaBoost – алгоритм AdaBoost для бустингу дерев.

У алгоритмі AdaBoost як базове використовувалося найкраще вирішальне дерево, отримане для DecisionTree.

Алгоритм VotingClassifier реалізує голосування кращих варіантів решти алгоритмів, які видають ймовірності приналежності об'єкта класам. При голосуванні використовується принцип “soft”, коли вибирається клас із найбільшою сумою ймовірності приналежності йому.

Як основну метрику оцінки точності класифікації використовується balanced_accuracy_score – збалансована точність.

Основна перевага даної метрики у адекватній оцінці точності алгоритмів класифікації із урахуванням сильного дисбалансу кількості розмічених записів по кожному класу набору даних.

Найбільшу точність класифікації, як у навчального, і на тестовому наборах забезпечують алгоритми RandomForest і VotingClassifier. Інші алгоритми показують також дуже близьку точність, більшу за 0.98. Із погляду практики

використання RandomForest більш виправдане, ніж VotingClassifier, оскільки не потрібно навчання та обрахунок алгоритмів всіх інших класифікаторів, достатньо лише одного.

Алгоритм RandomForest забезпечує 0.01% хибно-негативних, 2.6% хибно-позитивних спрацьовувань, що є цілком прийнятним. Для виключення частини хибно-позитивних спрацьовувань може бути проведений додатковий селективний аналіз із використанням подальшої класифікації трафіку за основними категоріями атак із подальшим аналізом із використанням додаткових інструментів.

Далі розглянемо класифікацію атак за категоріями.

2.2.3 Багатокласова класифікація атак

У розміченому наборі даних UNSW-NB15 існує 9 класів атак:

- Fuzzers – генерація випадкових даних, щоб викликати відмову програми чи мережі;

- Analysis – містить різні атаки, пов'язані зі скануванням портів, спамом та впровадженням у HT-ML-файли;

- Bakdoors – обхід механізмів захисту з метою прихованого доступу до даних або програм;

- DoS – відмова в обслуговуванні сервера чи мережного ресурсу;

- Exploits – експлуатація відомих атакуючого уразливостей в операційній системі чи програмі;

- Generic – техніка виявлення трафіку, шифрованого блоковим шифром;

- Reconnaissance – розвідувальні атаки;

- Shellcode - передача невеликих частин коду, що використовуються для вразливості програм;

- Worms - атаки, пов'язані з вірусами, що самореплікуються.

Для багатокласової класифікації були використані самі алгоритми із бібліотеки scikit-learn, які були раніше застосовані для бінарної класифікації. Аналогічно двійкової класифікації набір даних був поділений на навчальну та

тестову частини у відсотковому співвідношенні 75% та 25%.

2.2.4 Атака SYN FLOOD

Розглянемо атаку SYN Flood. Метою такої атаки є виведення із ладу мережевої служби так, щоб сервер, що атакується, не міг відповідати на запити «звичайних» користувачів. Для цього зловмисником на адресу жертви надсилається велика кількість пакетів даних, які переповнюють канал жертви. Точніше, атака ведеться на операційну систему з метою переповнення черги пакетів, що надходять. Так як розмір черги пакетів обмежений деякою величиною N, а на обробку кожного запиту йде певний час T, то момент відмови в обслуговуванні користувачів визначатиметься нерівністю:

$$v \geq N/T \tag{2.1}$$

де:

v - швидкість пакетів, що приходять, тобто кількість пакетів на секунду,

N - довжина черги вхідних з'єднань,

T - час очікування завершення з'єднання у секундах.

Значних збитків можна уникнути, виявивши атаку на ранній стадії. У цьому випадку системний адміністратор матиме час на вжиття заходів щодо запобігання атаці або заходів щодо мінімізації можливої шкоди. Для вирішення цього завдання необхідно аналізувати вхідний трафік і якщо черга пакетів даних, що надходить на сервер, має ознаки початку атаки, повідомити про це мережному адміністратору.

Розглянемо докладніше параметри черги вхідних пакетів даних, які підлягатимуть аналізу. Припустимо, для аналізу використовується послідовність N пакетів. Аналізуватимемо всю послідовність, а не кожен пакет окремо. Істотними для аналізу є такі параметри, як TCP прапори (SYN) (цікавими є пакети тільки із цими прапорами), IP адреси відправника та одержувача, напрямок пакета (вхідний або вихідний), час надходження пакета та порт одержувача. Також можна враховувати коректність заголовка:

надходження великої кількості пакетів із некоректними заголовками є одним із ознак атаки. Таким чином, на основі експериментальних даних, був складений вектор параметрів для аналізу, що складається з наступних компонентів:

- відсоток пакетів із різними зовнішніми IP адресами;
- відсоток пакетів із різними зовнішніми портами;
- середній час надходження одного пакета;
- відсоток пакетів із некоректними заголовками.

Експерименти проводилися на комп'ютері із процесором Pentium IV із частотою 3 ГГц, 512 Mb оперативної пам'яті. Для проведення атак використовувалася віртуальна машина VMware із запущеною на ній ОС LINUX Freespire. За допомогою віртуальної машини створювалася локальна мережа та сніффер, встановлений на хостовій операційній системі, фіксував параметри трафіку. Як сніфер використовувалася Trial версія програми SoftPerfect Analizer [50].

2.3 Розробка моделі глибинної нейромережі для виявлення вторгнень в комп’ютерну систему

Для обробки аномального трафіку пропонується застосовувати гібридну нейронну мережу, що складається з мережі Коханена, що самоорганізується (selforganizing map, SOM) і багатошарового персептрона.

За допомогою застосування карт Коханена відбувається кластеризація 50-символьних подій до вузлів матриці, в яких будуть згруповані події аналогічних числових символів. Фактично, окремі вузли будуть певними сценаріями атак.

На рисунку 2.2 представлено архітектуру пропонованої системи виявлення атак.

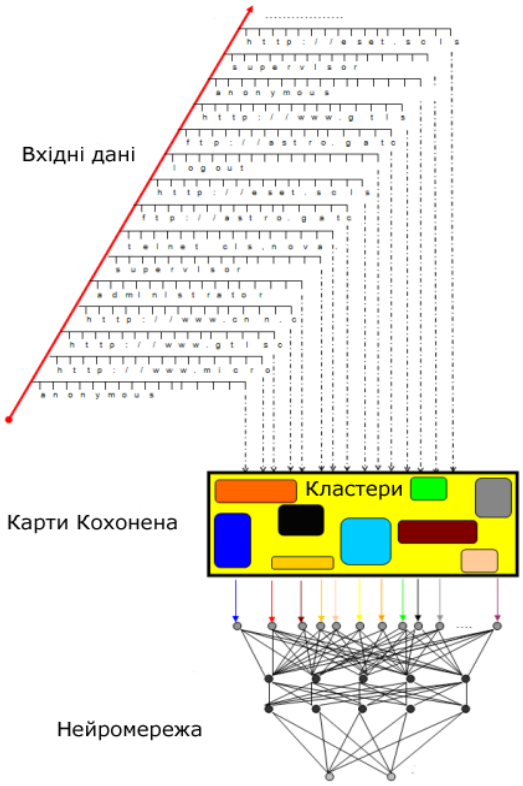


Рисунок 2.2 - Архітектура запропонованої системи виявлення атак

Вхідний вектор SOM містить наступні компоненти:

- 1–4 – байти, складові адресу пакета (для протоколу IPv4), наведені діапазон 0-1;

- 5 – порт пакета, наведений у діапазоні 0–1;

- 6-55 - перші 50 байтів даних пакета, наведені в діапазоні 0-1.

Після цього дані заголовків пакетів та інформація про угруповання подаються на вхід багатошарового персептрона, навченого розпізнавати аномальний трафік, але з урахуванням інформації про подію, тобто приналежності пакета тій чи іншій групі-сценарію. Це дозволяє не тільки виявляти аномалії в одиничних пакетах, але й виявляти належність пакета до розподіленої за часом атаки.

Формат вхідного вектора для мережі FF (персептрона) наступний. Вхідна вибірка розбивається на частини, що не перетинаються, довжиною 150 пакетів.

Усі пакети проходять через мережу Кохонена. Пакету ставиться у відповідність номер кластера (номер нейрона-переможця).

Кластери укрупнюються - 5 сусідніх об'єднуються воєдино. Виходить 100 укрупнених кластерів.

За укрупненими кластерами вважається гістограма для подання трафіку. Тобто, перший компонент вектора – це кількість пакетів, які потрапили до першого укрупненого кластера (кластери 1–5), тощо. Гістограма нормується, компоненти наводяться до діапазону 0-1. Після цього обчислюється дельта між сусідніми векторами.

2.4 Висновки до Розділу 2

Створення ефективної системи виявлення мережних атак вимагає застосування якісно нових підходів до обробки інформації, які повинні ґрунтуватися на адаптивних алгоритмах, здатних до самонавчання. Найбільш перспективним напрямом у створенні подібних систем виявлення атак на комп’ютерну мережу є застосування нейромережної технології.

Запропоновано модель для захисту від вторгнень. Модуль аналізу трафіку включає нейронну мережу. Для цієї системи було обрано мережу Кохонена. Це дозволяє працювати з вхідними векторами, які з дійсних чисел. Мережа здійснює визначення належності з'єднання до групи мережевих з'єднань, характерних для нормального або аномального стану мережі.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕННЯ НА ОСНОВІ ГЛИБИННОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Розмежування функцій вразливостей включення в комп’ютерну мережу та системи прийняття рішень

У запропонованому методі враховується два типи активності трафіку. Під час аналізу окремих вузлів кластера виявляються окремі сценарії атак. Крім того, аналізуються різкі перепади потужності мережного потоку виявлення сплесків навантаження. Спільний аналіз цих двох типів подій дозволяє точніше реагувати на появу DDoS-трафіку, при цьому знижуючи кількість помилкових спрацьовувань, пов'язаних з легальним зростанням використання смуги пропускання.

Для навчання штучної нейронної мережі моделювалися два типи мережного трафіку – нормальний та аномальний. Перший містив пакети, що з'являються в мережі за нормальної роботи, а другий імітував розподілену атаку.

Збір даних проводилося у 2 етапи.

1. Етап збирання нормальної поведінки. На сервері, що атакується, створений php-скрипт, який виконує «важкий» sql-запит (Цикл в N ітерацій). З атакуючого сервера Apache BanchMark запускався для створення легітимних підключень до сервера. 10 потоків за 100 запитів.

2. Етап збирання аномальної поведінки. Для імітації трафіку, що виникає під час атаки, на сервері, що атакує, запускається 3 копії скрипта slowloris з різницею в параметрі, що відповідає за затримку між повторними підключеннями.

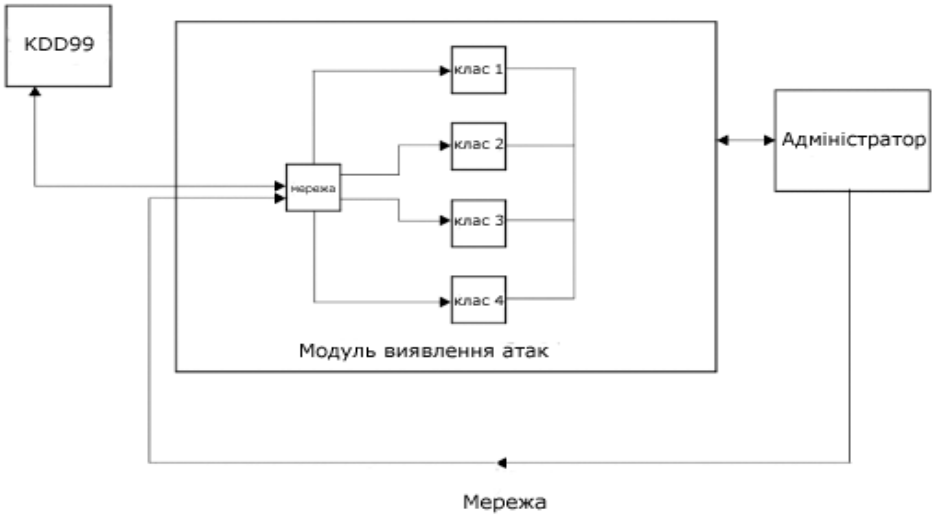


Рисунок 3.1 - Схема обробки даних та виявлення мережевих атак

У [9] виконано огляд наявних наборів даних, найбільш поширеним із яких є база даних NSL–KDD, що створена з ініціативи Управління перспективних дослідницьких проєктів Міністерства оборони США (DARPA) на основі бази даних KDD99 [57]. Набір даних складається з таких множин [56]: KDDTest+, KDDTrain+, KDDTrain+20 %.

Для експериментів будемо використовувати базу KDD99, де еталонний вектор містить 41 параметри трафіку. Отже, вхідним вектором для навчання є набір із 41 параметра TCP–з'єднання.

3.1.1 Нормалізація даних

Для моделювання нейромережі, яка призначена виявляти типи атаки на комп'ютерну мережу, потрібно виконати нормалізацію даних. Проведено аналіз за кількістю еталонів кожного класу атак. Атаки, які мали менше 200 еталонів, виключаємо із розгляду. У результаті залишилися такі класи атак:

- DoS;
- Probe;
- U2L;

- R2L.

Далі сформовано файли із еталонами для навчання мережі.

Перейдемо до перетворення параметрів, які містили нечислові типи даних, тобто типу String. Отже, необхідна заміна їх на числові коефіцієнти. До параметрів, що потребують конвертації, належить:

- тип протоколу;
- сервіс;
- flag;
- label.

Параметр “Тип протоколу” має 3 значення, кожному з яких присвоїмо числовий коефіцієнт:

- Icmp присвоєно значення 0;
- Tcp – значення 1;
- Udp – значення 2.

Отже, виявлення атак на комп’ютерну систему здійснено за допомогою використання багатошарової нейронної мережі Кохонена з донавчанням, на вхід якої подається 41 параметр із метою класифікації типів атак. Для моделювання нейромережі використовуються нормалізовані дані з відкритої бази KDD99.

3.2 Вибір середовища та системи для програмної реалізації

Для розробки даного програмного продукту потрібно обрати мову програмування, яка використовує принципи об'єктно-орієнтованого програмування.

До сучасних середовищ розробки програмного забезпечення, які підходять для вирішення поставленого завдання, можна віднести Microsoft Visual Studio, C++ Builder, Embarcadero Delphi тощо.

Microsoft Visual Studio – інтегроване середовище розробки, що включає інструментальні засоби для проектування, кодування, транслювання, налагодження та виконання програм [58]. Visual Studio дозволяє швидко

створювати та впроваджувати різноманітні програми на базі ОС Windows, веб-додатки та програми для мобільних пристроїв.

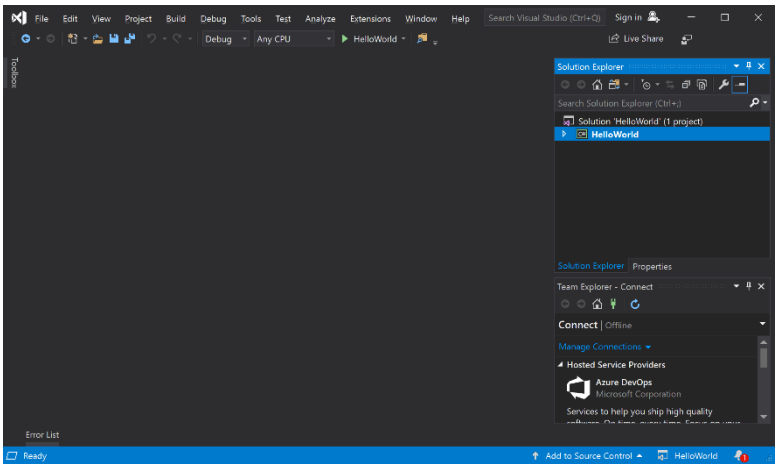


Рисунок 3.2 – Microsoft Visual Studio

У Visual Studio пропонується ціла низка шаблонів додатків, корисних під час створення програм, і кілька мов програмування, якими можна написати ці програми: Visual Basic, Visual C#, Visual C++, JScript тощо [58].

Visual Studio підтримує підключення бібліотеки DevExpress із широким спектром можливостей.

У додатки створювані за допомогою Visual Studio можна впроваджувати різні технології. Нижче наведено опис деяких із них:

- .NET Framework, .NET Framework 3.5, .NET Framework 3.0, .NET Compact Framework - це інтегрований компонент Windows, який підтримує створення та виконання нового покоління додатків та веб-служб XML.

- Windows Presentation Foundation (WPF) - WPF є набір типів .NET Framework, який можна використовувати для створення зовнішнього вигляду клієнтських програм Windows. WPF складається з таких компонентів, як розширювана мова виправлення для програм XAML, елементи керування, прив'язка даних, двовимірна та тривимірна графіка, анімація, стилі, шаблони, документи, мультимедійні дані, текст та типографічні засоби.

- Silverlight – це незалежна від оглядача та платформи технологія, що дозволяє проектувати, розробляти та постачати інтерфейси з підтримкою мультимедіа та багатофункціональні програми в Інтернеті.

- Windows Forms – дозволяє розробляти прості у розгортанні та оновленні програми з широкими графічними можливостями. Крім того, при доступі до Windows Forms до ресурсів на локальному комп'ютері забезпечується більш високий рівень безпеки, ніж при роботі традиційних програм Windows.

- Мова XAML – це мова розмітки для декларативної розробки програм. Windows Presentation Foundation (WPF) реалізує завантажувач XAML і забезпечує підтримку мови XAML для типів WPF, тому більшу частину інтерфейсу програми можна створювати за допомогою розмітки XAML.

- ASP.NET надає платформу, яку можна використовувати для створення веб-застосунків. До її складу входять такі служби, як керування станом, обробники HTTP, модулі HTTP та маршрутизація ASP.NET.

C++ Builder - програмний продукт, інструмент швидкої розробки програм (RAD), інтегрована середовище програмування (IDE), система, використовувана програмістами розробки програмного забезпечення мовою C++ [59].

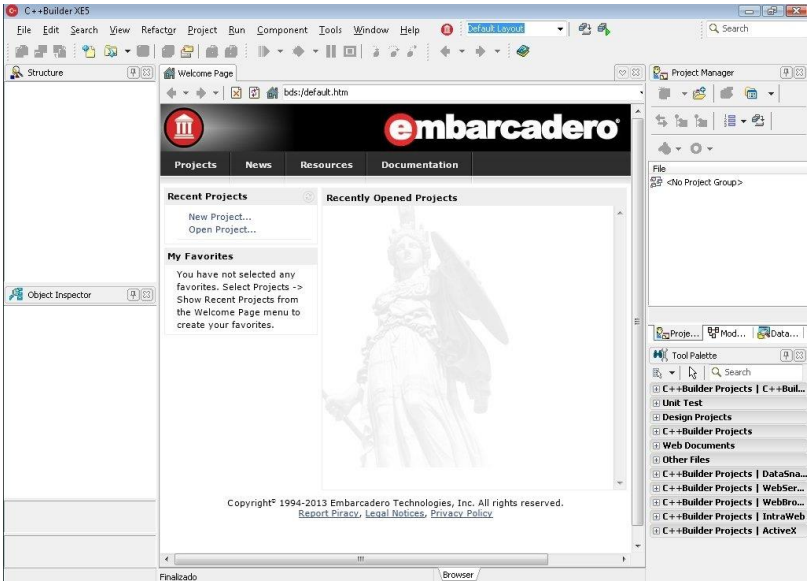


Рисунок 3.3 – C++ Builder

C++ Builder поєднує у собі комплекс об'єктних бібліотек (STL, VCL, CLX, MFC тощо), компілятор, налагоджувач, редактор коду та ще інші компоненти.

Цикл розробки аналогічний Delphi. Більшість компонентів, розроблених у Delphi, можна використовувати і C++ Builder без модифікації.

C++ Builder містить інструменти, які за допомогою drag-and-drop справді роблять розробку візуальною, спрощує програмування завдяки вбудованому WYSIWYG - редактору інтерфейсу тощо.

Embarcadero Delphi - просунуті програмні інструменти для розробників програм для Windows, Android, iOS, macOS та Linux. Потужне RAD-середовище для швидкої розробки високопродуктивних крос-платформних native-додатків із використанням потужних засобів візуального проектування та інтегрованих наборів інструментів, що подобається як незалежним, так і корпоративним розробникам (рис. 3.4).

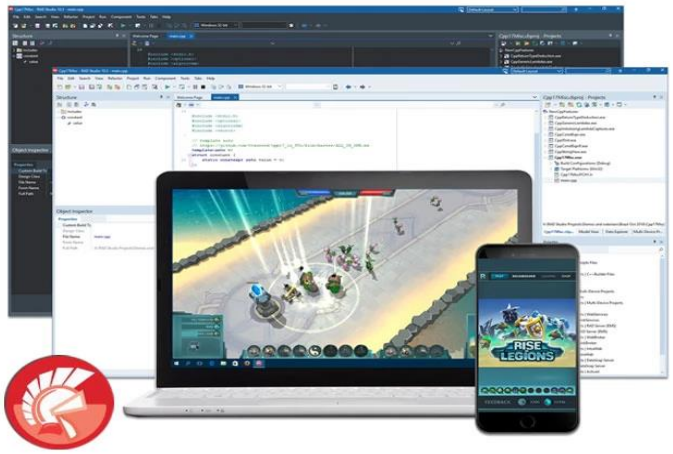


Рисунок 3.4 - Embarcadero Delphi

Фреймворк бібліотеки візуальних компонентів для Windows та візуальне середовище FireMonkey для міжплатформних інтерфейсів забезпечують основу для створення інтуїтивних і красивих інтерфейсів, що вражають на будь-якій платформі: Windows, macOS, iOS, Android та Linux [60].

Швидше добирайтеся до причини помилки за рахунок використання інтегрованого міжплатформного налагодження коду для інструментальної платформи. Швидкі цикли розробки не обов'язково призводять до погіршення

якості. Delphi включає безліч функцій, покликаних впровадити передові методи при написанні коду, знизити дублювання та допомогти стати суперрозробником.

Розглянувши середовища розробки програмного забезпечення, створено порівняльну таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз середовищ розробки ПЗ

Параметри	Visual Studio	C++ Builder	Embarcadero Delphi
Редактор користувальницького інтерфейсу	+	+	+
Вбудований профілювальник	+	+	+

3.3 Огляд розробленого програмного продукту та аналіз результатів

Виявлення активних атак на комп'ютерну систему відбувається за допомогою аналізу мережевого трафіку - дані, які надходять в систему або відправляються з неї. Для ясності процесу виявлення розглянемо параметри мережевого трафіку, які аналізуються для забезпечення безпеки комп'ютерних систем.

Дані в комп'ютерних мережах передаються у вигляді мережевих пакетів. У структурі мережевого пакету виділяють три основні поля (рис. 3.5): заголовок пакету, поле даних пакету, закінчення пакету.

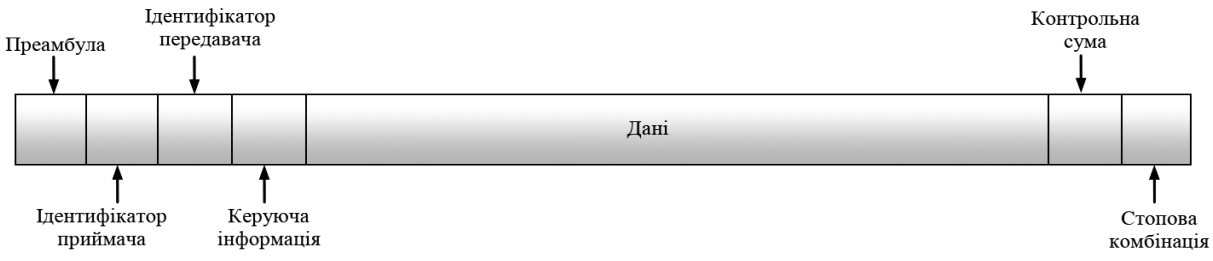


Рисунок 3.5 - Структура мережевого пакету

Заголовок пакету містить стартову комбінацію, яка забезпечує налаштування мережевого обладнання на прийом і обробку пакету, а також мережеві адреси приймача і передавача пакету і деяку загальну службову інформацію.

Поле даних пакету містить в собі, власне, інформацію, яка і передається від передавача до приймача. Закінчення пакету містить в собі контрольну суму, яка дозволяє судити про успішність передачі інформації, стопову комбінацію, яка служить для інформування про закінчення пакету, а також деяку службову інформацію.

Виділяють 41 параметр (або атрибут) мережевого з'єднання, які в свою чергу об'єднані в 3 групи:

1. Вбудовані атрибути. Ці атрибути отримуються із зони заголовку мережевих пакетів. Виділяють 9 вбудованих атрибутів, які містять інформацію про час роботи з'єднання, тип протоколу, кількість переданих байт і т.д.

2. Атрибути контенту, які отримуються із зони контенту і містять таку інформацію як: кількість невдалих спроб реєстрації, кількість невдалих спроб реєстрації в системі, кількість помилок, кількість операцій створення файлів і т.д. Існує 13 атрибутів контенту.

3. Атрибути трафіку. Обчислюються виходячи з попередніх з’єднань. У свою чергу виділяють атрибути тимчасового і машинного трафіку. 19 атрибутів трафіку містять наступну інформацію: кількість з'єднань до цієї ж IP-адреси, кількість з'єднань до цього ж номеру порту тощо.

У залежності від використовуваних технік при здійсненні несанкціонованих впливів на комп'ютерну систему, виділяють 4 типи активних атак.

DoS (denial of service) атаки - це активні атаки, спрямовані на виникнення ситуації, коли в системі, що атакується відбувається відмова в обслуговуванні. Дані атаки характеризуються генерацією великого обсягу трафіку, що призводить до перевантаження та блокування сервера. Виділяють шість DoS атак: back, land, neptune, Pod, Smurf, teardrop.

U2R (user-to-root) атаки передбачають отримання зареєстрованим користувачам привілеїв локального суперкористувача (адміністратора). Виділяють чотири типи U2R атак:

– buffer_overflow;
– loadmodule;
– perl;
– rootkit.

R2L (remote-to-local) атаки характеризуються отриманням доступу незареєстрованого користувача до комп'ютера з боку віддаленої машини. Виділяють вісім типів R2L атак:

– ftp_write;
– guess_passwd;
– imap;
– multihop;
– phf;
– spy;
– warezclient;
– warezmaster.

Probe атаки полягають в скануванні мережевих портів з метою отримання конфіденційної інформації. Виділяють чотири типи Probe атак:

– ipsweep;
– nmap;
– portsweep;
– satan.

Проведений аналіз пропонованих систем виявлення активних атак та захисту комп'ютерних систем від шкідливих впливів показує недосконалість існуючих методів захисту комп'ютерних систем від активних атак. У зв'язку з цим, розробка нових методів захисту інформації, що дозволяють підвищити рівень захищеності комп'ютерних систем від несанкціонованого впливу є актуальною.

Згідно з завданням необхідно реалізувати захист розроблюваного програмного засобу від несанкціонованого використання. Реалізовано захист програмного засобу від несанкціонованого використання за допомогою паролю (рис. 3.6):

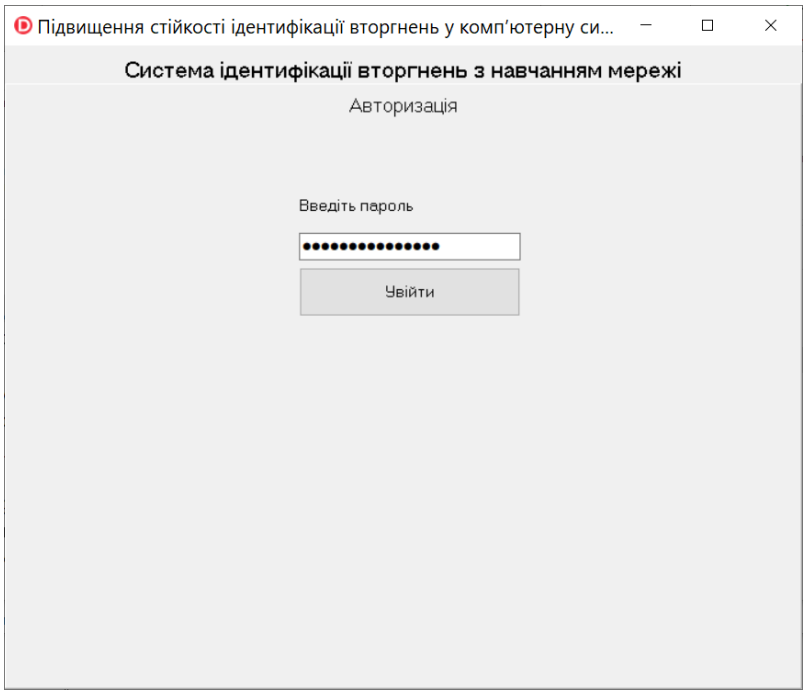


Рисунок 3.6 - Захист розробленого програмного засобу від несанкціонованого використання

Для входу в систему необхідно запустити файл Main.exe та ввести пароль: 111. У разі неуспішної авторизації система видає повідомлення (рис. 3.7):

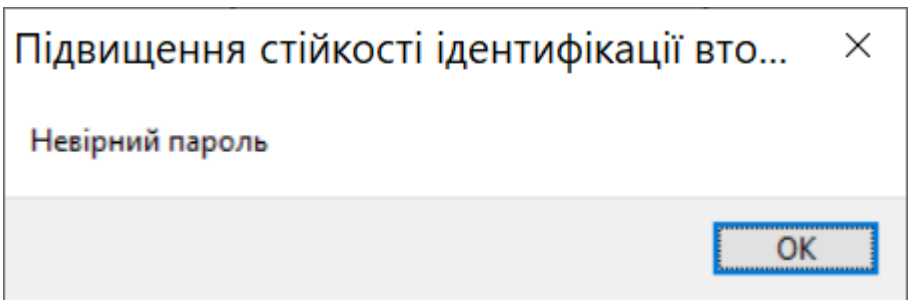


Рисунок 3.7 – Помилка авторизації

При успішній авторизації відкриється головне вікно програми (рис. 3.8):

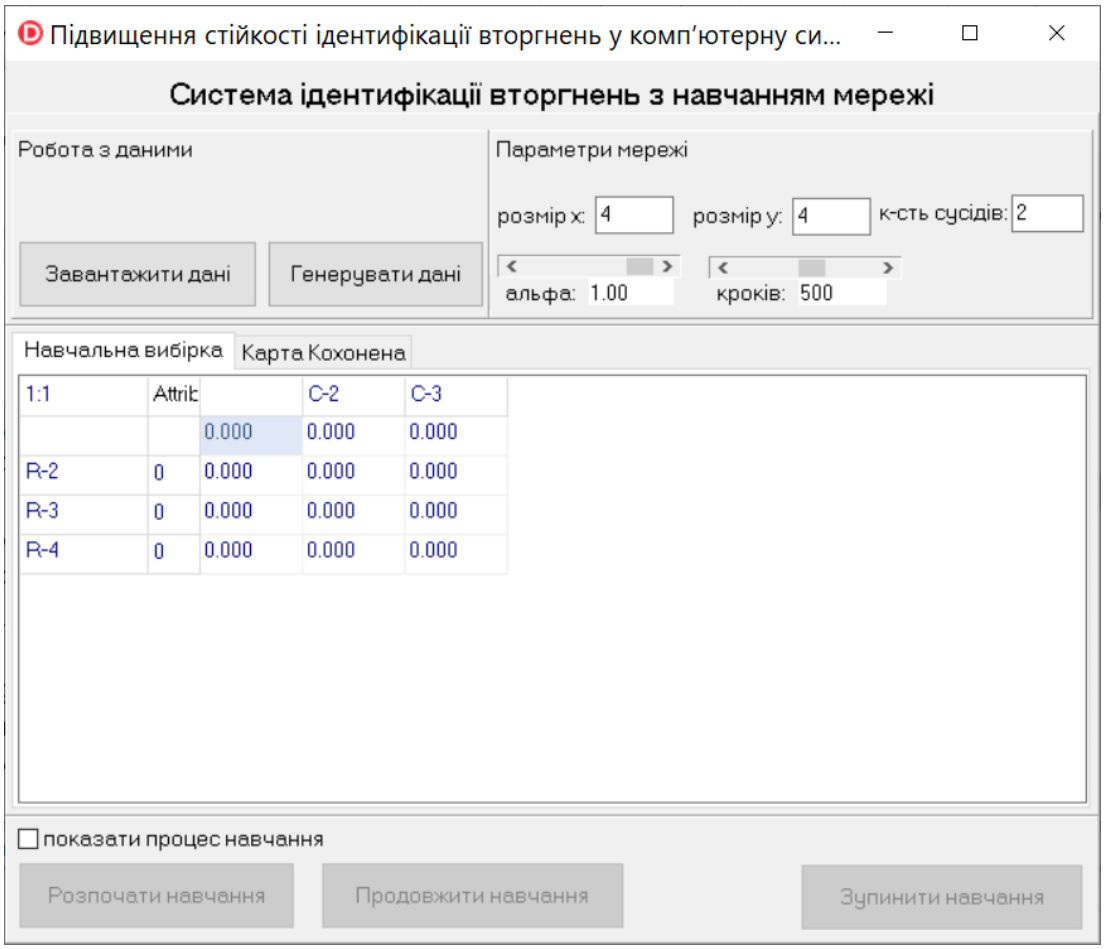


Рисунок 3.8 – Головне вікно програми

Розроблений програмний засіб має зручний інтерфейс користувача. Головне вікно програми логічно розділене на блоки:

– робота з даними;
– параметри мережі;
– навчальна вибірка;
– карта Кохонена;
– блок управління навчанням.

Блок «робота з даними» містить кнопки «Завантажити дані» та «Генерувати дані». Натиснувши кнопку «Завантажити дані», завантажуємо дані навчальної вибірки (рис. 3.9). Навчальні вибірки було заздалегідь підготовлені, використовуючи базу KDD99 [57].

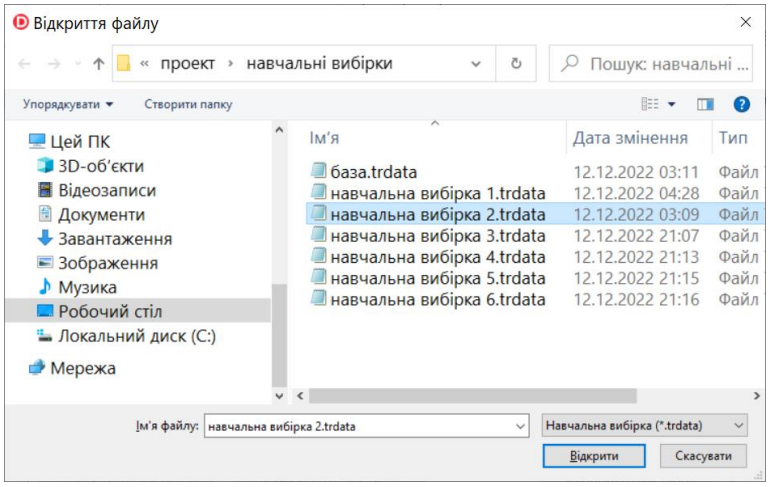


Рисунок 3.9 – Завантаження навчальної вибірки

Результат завантаження навчальної вибірки представлено на рисунку 3.10:

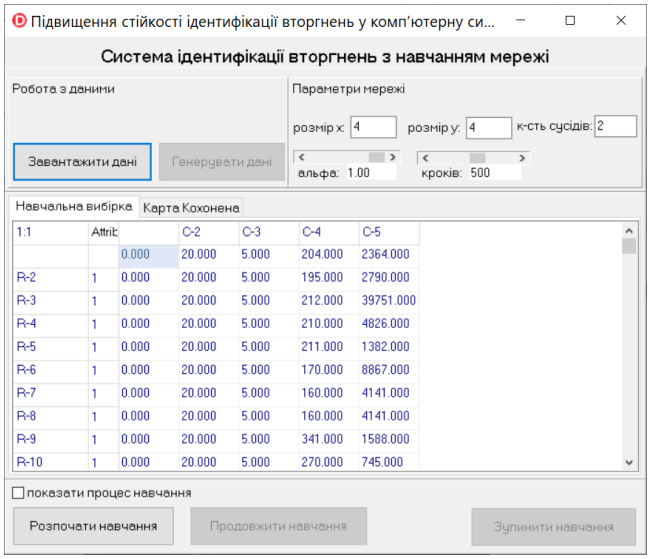


Рисунок 3.10 – Результат завантаження навчальної вибірки

Натиснувши кнопку «Генерувати дані», створюється тестова множина випадкових чисел (рис. 3.11).

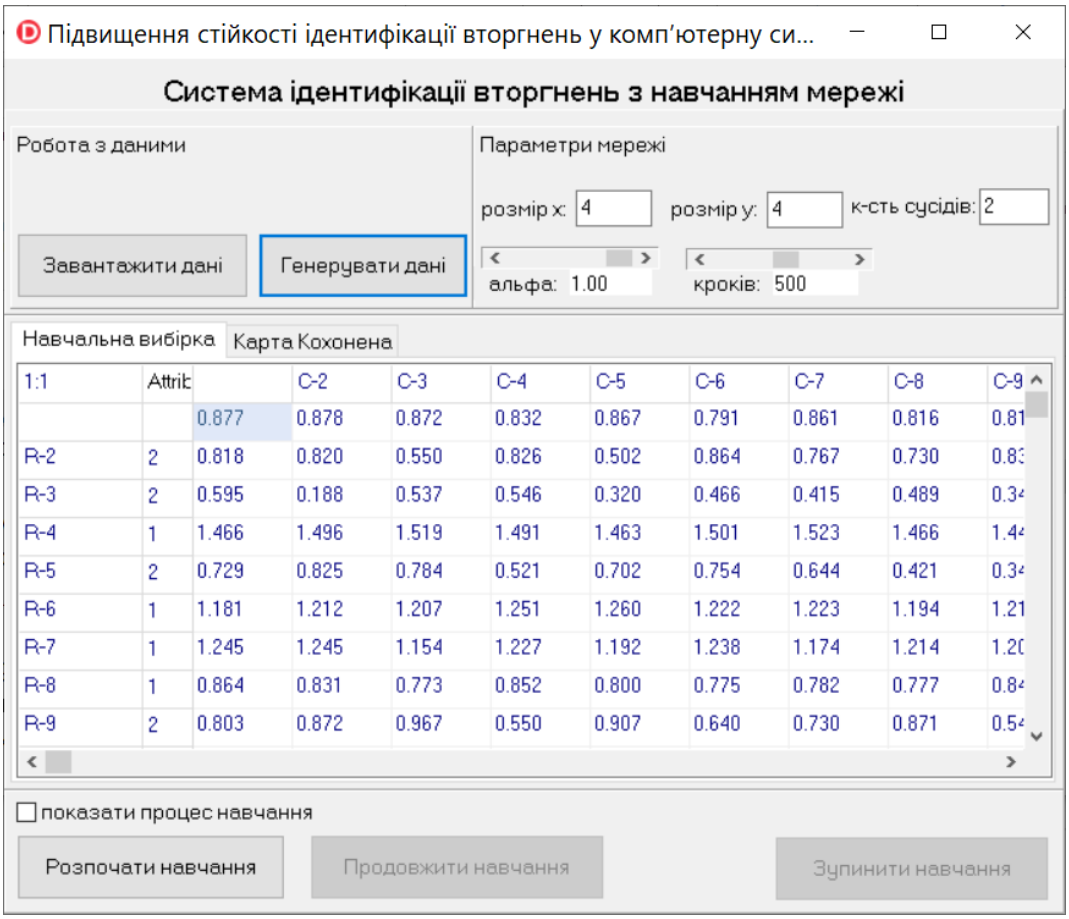


Рисунок 3.11 – Результат генерування даних

Після завантаження навчальних даних, переходимо до блоку «параметри мережі». У даному блоці можна вказати:

– розмір вихідної матриці;
– коефіцієнт альфа;
– кількість сусідів;
– кількість кроків навчання.

Враховуючи, що в базі KDD99 є 4 основні типи атак, визначаємо розмір вихідної матриці 2x2 та переходимо до процесу навчання мережі, використовуючи попередню нормалізацію вхідних даних відносно наступних параметрів: protocol_type, service, flag. Для цього потрібно натиснути кнопку «Розпочати навчання». Для візуалізації процесу необхідно відмітити прапорець «показати процес навчання» (рис. 3.12):

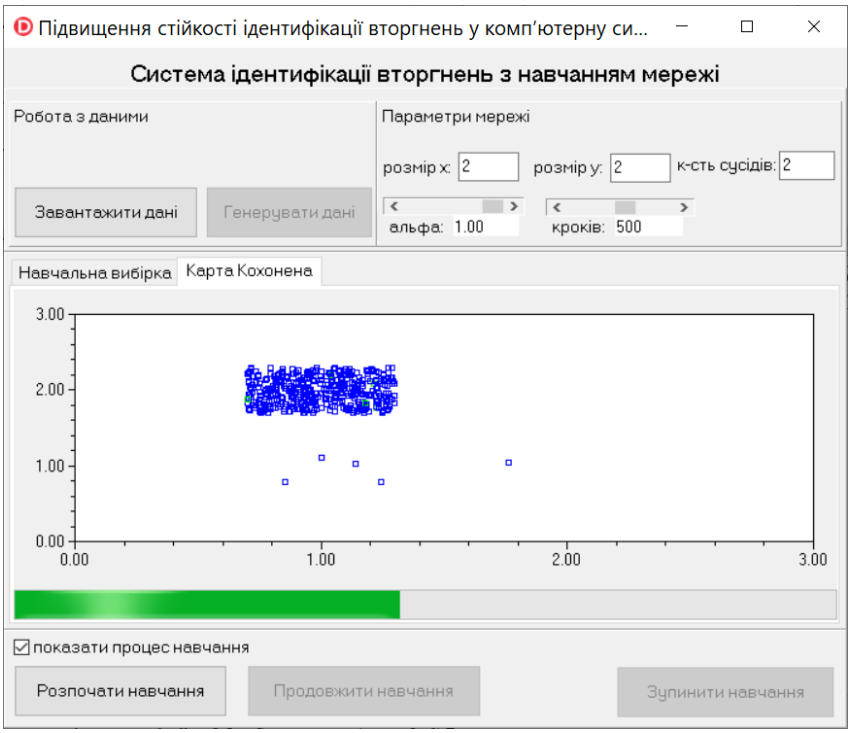


Рисунок 3.12 – Процес навчання мережі

Розроблений програмний засіб дозволяє зупиняти процес навчання та при необхідності проводити донавчання мережі. Для цього необхідно натиснути кнопку «Продовжити навчання». Після успішного навчання у блоці «Карта Кохонена» отримуємо результат навчання (рис. 3.13):

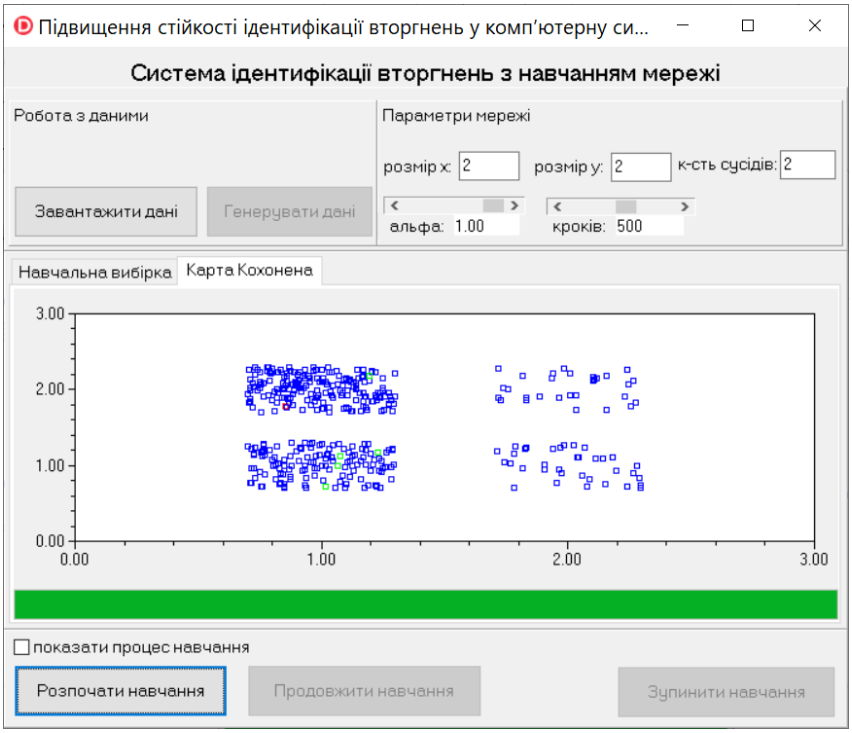


Рисунок 3.13 – Результат навчання мережі

Проаналізуємо отримані результати навчання мережі Кохонена для ідентифікації вторгнень у комп'ютерну систему (рис. 3.14):

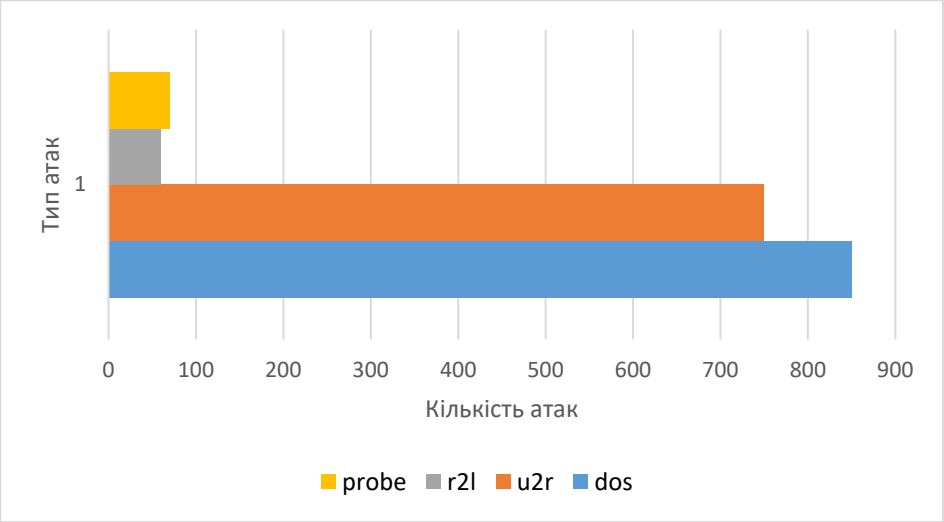


Рисунок 3.14 – Гістограма розподілення атак за типами

Таким чином, було розроблено програмний продукт для реалізації системи автоматизованого тестування стійкості ідентифікації вторгнень у комп'ютерну систему на основі глибинної нейромережі.

3.4 Висновки до Розділу 3

В розділі 3 запропоновано модель виявлення вразливостей на основі діагностики вторгнень у комп'ютерну мережу без оновлення сигнатур. Модель заснована на роботі з ознаками вторгнення, які за допомогою операторів глибинної нейромережі Кохонена приймають відповідні рішення про наявність або відсутність несанкціонованих дій в системі. Для виявлення вторгнень описані ознаки несанкціонованих дій, які найкраще характеризують дії зловмисника або нападу.

Проаналізовано сучасні математичні діагностичні моделі та визначено елементи, за допомогою яких можна виявити несанкціоновані дії в комп'ютерних мережах. Опис моделі вирішено виконати за допомогою глибинної нейромережі Кохонена, що забезпечить уникнення типових помилок при виявленні несанкціонованих.

Реалізована програма для виявлення вторгнень у комп'ютерну систему на основі нейромережі Кохонена. Результати тестування розробленого програмного

продукту демонструють, що нейромережа Кохонена виявляє типи атак на базі даних KDD99 коректно за рахунок навчання та донавчання мережі на основі встановлення кількості прошарків по вертикалі та горизонталі та налаштування вагових коефіцієнтів.

4. ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ СТВОРЕННЯ ПЗ ІДЕНТИФІКАЦІЇ ВТОРГНЕНЬ У КОМП'ЮТЕРНУ СИСТЕМУ НА ОСНОВІ ГЛИБИННОЇ НЕЙРОМЕРЕЖІ

4.1 Проведення наукового аудиту науково-дослідної роботи

Додаток для аналізу вразливостей комп'ютерної мережі за допомогою глибинної нейромережі було створено для підвищення ефективності роботи.

Ефективність впровадження системи полягає в автоматизації процесу захисту мережі.

Зниження витрат на оплату праці зменшить і фінансові витрати, що призведе до загального зростання продуктивності та економії.

Основною метою даного розділу є визначення вартості науково-дослідних робіт, витрат на визначення економічного ефекту від основних і відповідних результатів, отриманих при розв'язанні технічних задач у підсумковій ідентифікаційній роботі для використання в перевезеннях. При оцінці ефективності прийнятих науково-технічних рішень повинні бути враховані всі необхідні витрати і витрати, що вимагає проведення необхідних розрахунків за певним рішенням.

Комісійні витрати орієнтуються на цінову політику мережі, як показано в таблиці 4.1.

Таблиця 4.1 – Вартість програмного та апаратного забезпечення

Назва	Кількість	Ціна
Apache HTTP Server	1	0
СУБД MySql + phpMyAdmin	1	0

Назва	Кількість	Ціна
USB-флешка KINGSTON DataTraveler SE9 16Gb (DTSE9H)	1	450
Диск DVD-R 4.7Gb SlimCase VS	1	25

Таблиця 4.2 – Матеріали

Вартість електроенергії базується на тривалості періоду розробки програмного забезпечення, кількості кіловат-годин, витрачених на розробку програмного забезпечення, і вартості електроенергії за 1 кіловат-годину. Митний збір для юридичних осіб становить 4,68 грн. за кіловат годину. Витрати відображені в таблиці 4.3.

Елементи системи	Встановлена потужність, кВт	Вартість 1 кіловат за годину (грн.)	робочі години	загальне споживання
Aspire E5-532-C5SZ Сірий	0,057	4,68	398	106,18

Таблиця 4.3 - Плата за електроенергію

Амортизація обладнання оприбутковується в процесі його використання, тобто під час впровадження та створення ПЗ.

Грошова оцінка амортизації — це витрати на амортизацію, що входять до поточної собівартості.

4.2 Проведення комерційного та технологічного аудиту

Основна мета розробки техніко-економічного обґрунтування (ТЕО) – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку, економічну доцільність його розробки та впровадження.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості.

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF, \tag{4.1}$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Розмір програмного коду було підраховано за допомогою інструменту вбудованого у Visual Studio 2019 Community – Code Metrics Results.

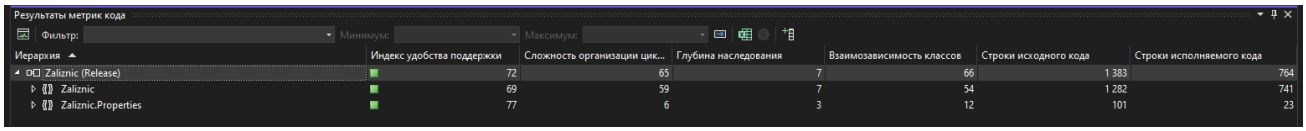


Рисунок 4.1 – Підрахунок розміру програмного коду

Отже розмір програмного коду становить 764 рядки:

$$E = 2{,}4 \cdot 0{,}764^{1{,}05} \cdot 1 = 1{,}81$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 1,81 людино-місяці.

Нижче наведені розрахунки вартості розробки «Автоматизована

система оцінки схожості програм». Основними статтями витрат прийняті:

- основна заробітна плата;
- відрахування на соціальні потреби;
- накладні витрати;
- витрати на персональний комп'ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 4.4.

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати грн
			чол	місяців	
1	інженер-програміст	15750	1	1,81	28507

Таблиця 4.4 – Фонд місячної заробітної плати

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

Описаний в проекті програмний продукт був розроблений одним програмістом в період з 27.09.21 до 22.11.21, що складає 40 днів або приблизно 8 робочих тиждень. Витрати робочого часу прийняті за 40 годин у тиждень. Погодинна ставка кваліфікованого інженера–програміста складає 93,75 грн/год. Таким чином, витрачено робочого часу:

$$t_{\text{розробки}} = N_{\text{чол}} \times N_{\text{тиж}} \times N_{\text{год}}, \quad (4.2)$$

де $N_{\text{чол}}$ – кількість виконавців, чол;

$N_{\text{тиж}}$ – тривалість розробки;

$N_{\text{год}}$ – витрати робочого часу, год;

$$t_{\text{розробки}} = 1 \cdot 8 \cdot 40 = 320 \text{ чол/год.}$$

ОЗП визначається за формулою:

$$\text{ОЗП} = t_{\text{розробки}} \cdot N \cdot K_{KB}, \quad (4.3)$$

де $t_{\text{розробки}}$ – витрати праці у чол/год;

N – погодинна ставка;

K_{KB} – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. Коефіцієнт кваліфікації розробника (k) - ступінь підготовленості виконавця до дорученої йому роботі (він визначається залежність від стажу праці та

становить:

- для працюючих до 2 років- 0,75;
- від 2 до 3 років 1,0;
- від 3 до 5 років - 1,1-1,2;
- від 5 до 7 років - 1,3-1,4;
- понад 7 років - 1,5-1,6.

В даному випадку KKB приймається 0,75. ОЗП складає:

$$\text{ОЗП} = 320 \cdot 93{,}75 \cdot 0{,}75 = 22500 \text{ грн.}$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати (22%):

$$C_{\text{соц}} = \frac{\text{ОЗП} \cdot 22\%}{100\%} \quad (4.4)$$

$$C_{\text{соц}} = \frac{22500 \cdot 22\%}{100\%} = 4950\text{грн.}$$

Отримані результати за (4.3) та (4.4) підсумовуються. Вони складають 27450 грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони

визначаються в процентах (30 – 40%) від суми прямих витрат:

$$С_{накл} = \frac{(ОЗП+С_{соц})\cdot 35\%}{100\%}; \quad (4.5)$$

$$С_{накл} = \frac{(22500+4950)\cdot 35\%}{100\%} = 9607{,}05 грн.$$

Протягом усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов'язані з її експлуатацією.

Експлуатаційні витрати на персональний комп'ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп'ютеру. В експлуатаційні витрати входять:

- вартість витратних матеріалів;
- витрати на ремонт;
- заробітна плата ремонтника;
- оренда приміщення;
- додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;
- амортизаційні витрати на персональний комп'ютер і програмне забезпечення;
- витрати на електроенергію ($С_{ел}$), які визначаються за формулою:

$$С_{ел} = P \cdot B \cdot T_{розр}, \quad (4.6)$$

де

Р – потужність комп'ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

B – вартість 1 кВт/годин для непобутових споживачів,

складає 1,8 грн;

Tрозр – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}45 \cdot 1{,}8 \cdot 320 = 259{,}2 \text{ грн.}$$

Витрати на витратні матеріали ($С_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп'ютеру. Вартість робочої станції приймається 18000 грн. [6], термін експлуатації – 5 років. Отже, можна визначити ці витрати за період створення програмного засобу:

$$С_{вм} = В_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%}, \quad (4.7)$$

де $B_{КОМ}$ – вартість персонального комп'ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп'ютеру.

Витрати на витратні матеріали визначаються так:

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 18000 \cdot \frac{40}{5 \cdot 365} \cdot \frac{10}{100} = 40 грн.$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп'ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 9000 грн. Тоді в перерахунку на один комп'ютер його заробітна плата за період розробки програмного продукту складає:

$$C_{рем} = \frac{C'_{рем}}{N_{КОМ}} \cdot T_{міс}, \quad (4.8)$$

де $C'_{рем}$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп'ютерів на одного ремонтника.

$Т_{мес}$ – час розробки програмного продукта, міс.

Заробітна плата ремонтника ($C_{рем}$) буде складати:

$$C_{рем} = \frac{9000}{50} \cdot 1{,}81 = 325{,}8 \text{ грн.}$$

За статистикою витрати на комплектуючи вироби ($С_{КОМ}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 40 грн. \quad (4.9)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює

терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$\text{АКП} = \text{В}_{КОМ} \cdot \frac{N_{\text{Д}}}{N_{\text{експ}} \cdot 365}; \quad (4.10)$$

$$\text{АКП} = 18000 \cdot \frac{1{,}81}{3 \cdot 12} = 905$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння для Windows 5 років та Visual Studio за 2 рік то амортизаційні відрахування на програмне забезпечення дорівнюють його вартості.

Для функціонування персонального комп'ютера використовувалася операційна система Windows 10, для написання програмного забезпечення - програмне середовище Visual Studio 2019 Community.

$$\text{АКП}_w = 13800 \cdot \frac{1{,}81}{5 \cdot 12} = 416{,}3$$

$$\text{АКП}_w = 0 \cdot \frac{1{,}81}{2 \cdot 12} = 0$$

Розрахунок амортизаційних відрахувань на програмне забезпечення зведений в табл. 5.2. Додаткові витрати ($C_{\text{дод}}$): прибирання приміщень, охорона, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера- програміст, тобто 7875 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 2500 гривень на місяць [5]. Тобто за весь період розробки – 4 100 грн. Сумарні експлуатаційні витрати на один персональний комп'ютер складають:

$$C_{\text{експ}} = C_{\text{ел}} + C_{\text{ВМ}} + C_{\text{рем}} + \text{АКП} + \text{АПО} + C_{\text{ор}} + C_{\text{дод}}; \quad (4.11)$$

$$C_{\text{експ}} = 259{,}2 + 40 + 325{,}8 + 905 + 416{,}3 + 4100 + 7875 = 13921{,}30 \text{ грн}$$

Результати розрахунків зведено у табл. 4.5.

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
Windows 10	13800	http://mtsoft.kiev.ua/product/windows-10-professional	416,3
Visual Studio 2019 Community	0	https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes	0
Всього:	13800		416,3

Таблиця 4.5 – Використовуване програмне забезпечення

Найменування витрат	Витрати, грн
Витрати на електроенергію	259
Вартість витратних матеріалів	40
Витрати на ремонт	326
Амортизація персонального комп’ютера	905
Амортизація програмного забезпечення	416
Оренда приміщення	4100
Додаткові витрати	7875
Всього	13921

Таблиця 4.6 – Експлуатаційні витрати на ПК і ПЗ.

Таким чином, витрати на створення програмного продукту складають:

$$C_{\text{розробки}} = \text{ОЗП} + C_{\text{соц}} + C_{\text{накл}} + C_{\text{експ}}; \qquad (4.12)$$

$$C_{\text{розробки}} = 22500 + 4950 + 9607 + 14051 = 51108 \text{ грн.}$$

Розрахунок витрат зведено у табл. 4.7.

Найменування витрат	Витрати, грн
Основна заробітна плата	22500
Відрахування на соціальні потреби	4950
Накладні витрати	9607
Експлуатаційні витрати	13921
Всього	50978

Таблиця 4.7 – Кошторис витрат на розробку програмного засобу

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 50978 грн.

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Час викладачів витрачається на невикористання додатків, розроблених у рамках даної роботи.

Також необхідно враховувати значення середньої вартості робочого часу працівників.

Оскільки програмне забезпечення, що розробляється, покликане полегшити роботу працівників, які також є клієнтами, окремо додаткових витрат на оплату праці не буде.

Опрацювання документів, складання тестів $t_{n1} = 3$г, за допомогою веб-додатку робота зі складання тестів буде меншою, $t_{n2} = 1$г.

Економія часу становитиме:

$$t_{n1} - t_{n2} = 3\text{г} - 1\text{г} = 2 \text{ години на день.}$$

За один місяць вчитель зекономив час:

$$T_{\text{рм1}} = 26 * 2\text{год} = 52\text{г}.$$

Це означає, що за рахунок економії часу можна збільшити години і вигода в грошовому вираженні складе: $P_{\text{м1}} = 168.26 * 52\text{г} = 8749{,}52\text{грн}.$ на місяць. Далі розраховуємо термін окупності системи: поточний $T_{\text{ok}} = 105141.84/8749.52 = 12.01$ місяць.

4.5 Висновок до Розділу 4

У четвертому розділі аналізується розроблене програмне забезпечення з точки зору актуальності та економічної доцільності його впровадження, а не використовуваних засобів. По-перше, визначається складність і час роботи по створенню програмного додатку.

Наступним кроком є розрахунок вартості впровадження та впровадження розробленого додатку. Завершальним етапом є розрахунок економічного ефекту від впровадження інформаційної системи.

Проведені розрахунки та метрики дозволяють говорити про доцільність та економічну вигоду із точки зору витрат часу впровадження інформаційної тестової системи.

ВИСНОВКИ

У магістерській роботі розв’язано наукову задачу підвищення стійкості ідентифікації вторгнень у комп’ютерну систему на основі нейронних мереж.

В ході написання роботи встановлено, що існуючі системи автоматизованого тестування не є в достатній мірі ефективними та функціональними щодо виявлення сучасних загроз та вразливостей, саме тому в роботі проведена розробка власної системи тестування. Аналіз мережі в системі відбувається за допомогою методів нейронної мережі Кохонена. Дані вузлів мережі записується в рядок, а потім через виклик методу класу відбувається аналіз даних вузлів, після аналізу вказується можливий відсоток вразливості. Цей підхід є ефективним та функціональним і дозволяє підвищити інформаційну безпеку програмного забезпечення.

У результаті написання роботи було виконано аналітичний огляд з теми дослідження. Досліджено проблеми захисту коду програмного забезпечення та визначено актуальність автоматизованого тестування. Здійснено огляд засобів захисту програмного забезпечення та розглянуто системи автоматизованого тестування. Виконано розробку та дослідження системи автоматизованого тестування безпеки комп’ютерної мережі за допомогою нейромережі.

Серед найбільш вагових наукових та практичних результатів варто відзначити те, що дістала подальший розвиток модель виявлення вторгнень на основі діагностики вторгнень у комп’ютерну мережу без оновлення сигнатур. Модель заснована на роботі із ознаками вторгнення, які за допомогою математичного апарату мережі Кохонена приймають відповідні рішення про наявність або відсутність несанкціонованих дій в системі.

Проаналізувавши сучасні математичні моделі, розвинуто принцип навчання мережі Кохонена за рахунок встановлення кількості прошарків за вертикаллю та горизонталлю і налаштування вагових коефіцієнтів. Крім того, вдосконалено методологію виявлення вторгнень за рахунок впровадження процесу донавчання нейромережі та розроблено відповідний програмний продукт.

ПЕРЕЛІК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Chaivat J., Naruemon W., Prasert K. Hybrid Neural Networks for Intrusion Detection System. 2002 [Електронний ресурс]. URL: https://www.researchgate.net/publication/266608342 (дата звернення: 17.11.2022).

2. A Review on Application of Machine Learning and Deep Learning [Електронний ресурс]. URL: https://www.ijert.org/a-review-on-application-of-machine-learning-and-deep-learning-for-intrusion-detection (дата звернення: 17.11.2022).

3. Grill M., Pevný T., Rehak M. Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. Journal of Computer and System Sciences. 2017. Vol. 83. Iss. 1. P. 43–57 [Електронний ресурс]. URL: https://doi.org/10.1016/j.jcss.2016.03.007 (дата звернення: 17.11.2022).

4. Gunes K. H., Nur Z.-H. A., Heywood M. I. A hierarchical SOM-based intrusion detection system. Engineering Applications of Artificial Intelligence. 2007. Vol. 20. Iss. 4. P. 439–451 [Електронний ресурс]. URL: https://doi.org/10.1016/j.engappai.2006.09.005 (дата звернення: 17.11.2022).

5. Kruti C., Bhavin S., Ompriya K. Improving user-to-root and remote-to-local attacks using growing hierarchical self organizing map. International Journal of Engineering Sciences & Research Technology. 2015. Vol. 4, № 6. P. 611–618.

6. Hindawi. Computer Network Intrusion Anomaly Detection with Recurrent Algorithms [Електронний ресурс]. URL: https://www.hindawi.com/journals/misy/2022/6576023/ (дата звернення: 17.11.2022).

7. Ortiz A. Improving Network Intrusion Detection with Growing Hierarchical Self-Organizing Maps. 2011 [Електронний ресурс]. URL: https://pdfs.semanticscholar.org/f3fb/cf7dfd84d9f2f2ace73580c32eb7c469b6e7.pdf (дата звернення: 17.11.2022).

8. Palomo E. J., Domínguez E., Luque R. M., Muñoz J. A new GHSOM Model applied to network security. Springer, Berlin, Heidelberg, 2008. P. 680–689

[Електронний ресурс]. URL: https://doi.org/10.1007/978-3-540-87536-9_70 (дата звернення: 17.11.2022).

9. Ring M., Wunderlich S., Scheuring D., Landes D., Hotho A. A Survey of Network-based Intrusion Detection Data Sets. Computers & Security. 2019. Vol. 86. P. 147–167 [Електронний ресурс]. URL: https://doi.org/10.1016/j.cose.2019.06.005 (дата звернення: 17.11.2022).

10. Kohonen, T. Sell-organised formation of topologically correct feature maps / T. Kohonen // Biological Cybernetics. – 1982. – N43. – P. 59-69.

11. A Hybrid Intrusion Detection Model Using EGA [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9414798/ (дата звернення: 17.11.2022).

12. Ross D. Securing IEEE802.11 Wireless LANs. PhD thesis, Queensland University of Technology, 2010 [Електронний ресурс]. URL: http://eprints.qut.edu.au/37638/1/David_Ross_Thesis.pdf (дата звернення 17.11.2022).

13. Nguyen T., Nguyen B., Pham H. Досить ефективним рішенням для вирішення дискусійного рішення на 802.11 мережах // 2012 Міжнародна конференція на Green Technology and Sustainable Development (GTSD2012): Journal of Engineering Technology1 2 .- P. 395-403.

14. Sinclair C., Pierce L., Matzner S. Application of Machine Learning to Network Intrusion Detection // Proceedings of Computer Security Applications Conference (ACSAC '99). - 1999. - P. 371-377.

15. Tang H., Cao Z. Machine Learning-based Intrusion Detection Algorithms // Journal of Computational Information Systems. - 2009. - P. 1825-1831.

16. Mukkamala S., Janoski G., Sung A. Intrusion Detection: Support Vector Machines and Neural Networks [Електронний ресурс]. URL: http://www.cs.uiuc.edu/class/fa05/cs591han/papers/mukkCNN02.pdf (дата звернення 17.11.2022).

17. Mulay S., Devale P., Garje G. Intrusion Detection System за допомогою Support Vector Machine and Decision Tree // International Journal of Computer

Applications. - 2010. - Vol. 3 № 3. - P. 40-43.

18. Deep learning vs. machine learning: What's the difference? [Електронний ресурс]. URL: https://www.zendesk.com/blog/machine-learning-and-deep-learning/ (дата звернення: 17.11.2022).

19. Aravindpai Pai. ANN vs CNN vs RNN | Types of Neural Networks [Електронний ресурс]. URL: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/ (дата звернення: 17.11.2022).

20. Методи аналізу та моделювання безпеки розподілених інформаційних систем: навч. посіб. / В.В. Литвинов, В.В. Казимир, І.В. Стеценко та ін. – Чернігів: Чернігівський національний технологічний університет, 2016. – 254 с.

21. A Beginner's Guide to Neural Networks and Deep Learning | Pathmind [Електронний ресурс]. URL: https://wiki.pathmind.com/neural-network (дата звернення: 17.11.2022).

22. AI vs. Machine Learning vs. Deep Learning vs. Neural Networks [Електронний ресурс]. URL: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks (дата звернення: 17.11.2022).

23. Deep learning [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Deep_learning (дата звернення: 17.11.2022).

24. Top 10 Deep Learning Algorithms You Should Know in 2023 [Електронний ресурс]. URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm (дата звернення: 17.11.2022).

25. Top 10 Deep Learning Algorithms in Machine Learning [2023] [Електронний ресурс]. URL: https://www.projectpro.io/article/deep-learning-algorithms/443 (дата звернення: 17.11.2022).

26. What is Deep Learning and How Does It Work? [Електронний ресурс]. URL: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network (дата звернення: 17.11.2022).

27. Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks [Електронний ресурс]. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 17.11.2022).

28. Risso F., Degioanni L. An architecture for high performance network analysis //Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on. - IEEE. 2001. P. 686-693.

29. Буханов Д. Г., Поляков В. М., Усков Д. А., Даєєф Ф. Виявлення SYN Flood атаки з використанням драйвера WinPCap //ISJ Theoretical & Applied Science. 2015. Т.1. №. 21. С. 139-144.

30. Carpenter GA, Grossberg S. Category learning and adaptive pattern recognition: Neural network model // Proceedings, Third Army Conference on Applied Mathematics and Computing, ARO Report. 1985. P. 86-101.

31. Carpenter GA, Grossberg S. ART 2: Selforganization з стабільною категорією recognition codes for analog input patterns //Applied optics. 1987. Т. 26. №23. С. 4919-4930.

32. Carpenter GA, Grossberg S., Rosen DB ART 2: An adaptative resonance algorithm for rapid category learning and recognition //Neural networks. 1991. Т. 4. №4. P. 493-504.

33. J. Ryan, M. Lin, і R. Miikkulainen. Intrusion Detection with Neural Networks. / AI.

34. Пристосування до Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop, Providence, RI. P. 72-79, 1997.

35. James Cannady, Artificial neural networks для misuse detection. / Proceedings of the 1998 National Information Systems Security Conference (NISSC'98), Arlington, VA, 1998.

36. Srinivas Mukkamala, Intrusion detection використовуючи neural networks і support vector machine. / Proceedings of the 2002 IEEE International Honolulu, HI, 2002.

37. P. Lichodzijewski, AN Zincir Heywood, i M.I. / Proceedings of the 2002 IEEE World Congress on Computational Intelligence, Honolulu, HI, pp. 1714-1719, 2002.

38. Зоріна Т.І. Системи виявлення і запобігання атак в комп'ютерних мережах / Т.І. Зоріна // Вісник східноукраїнського національного університету імені Володимира Даля. – 2013. – № 15 (204) ч.1. – С. 48 – 54.

39. Manage risk [Електронний ресурс]. URL: https://www.infoentrepreneurs.org/en/guides/manage-risk/ (дата звернення: 17.11.2022).

40. Phishing Attacks: A Recent Comprehensive Study [Електронний ресурс]. URL: https://www.frontiersin.org/articles/10.3389/fcomp.2021.563060/full (дата звернення: 17.11.2022).

41. Risk, resilience, and rebalancing in global value chains | McKinsey [Електронний ресурс]. URL: https://www.mckinsey.com/capabilities/operations/our-insights/risk-resilience-and-rebalancing-in-global-value-chains (дата звернення: 17.11.2022).

42. Slope stability analysis [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Slope_stability_analysis (дата звернення: 17.11.2022).

43. Stable face representations [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3130381/ (дата звернення: 17.11.2022).

44. Summary | Biometric Recognition: Challenges and Opportunities [Електронний ресурс]. URL: https://www.nap.edu/read/12720/chapter/2 (дата звернення: 17.11.2022).

45. Updated Guidelines for Evaluating Public Health Surveillance Systems [Електронний ресурс]. URL: https://www.cdc.gov/mmwr/preview/mmwrhtml/rr5013a1.htm (дата

звернення: 17.11.2022).

46. What is Risk Management and Why is it Important? [Електронний ресурс]. URL: https://www.techtarget.com/searchsecurity/definition/What-is-risk-management-and-why-is-it-important (дата звернення: 17.11.2022).

47. What is an Application Programming Interface (API)? | IBM [Електронний ресурс]. URL: https://www.ibm.com/cloud/learn/api (дата звернення: 17.11.2022).

48. Lincoln Kaffenberger, Emanuel Kopp. Cyber Risk Scenarios, the Financial System, and Systemic Risk [Електронний ресурс]. URL: https://carnegieendowment.org/2019/09/30/cyber-risk-scenarios-financial-system-and-systemic-risk-assessment-pub-79911 (дата звернення: 17.11.2022).

49. Paxson V. Bro: A System for Detecting Network Intruders в Real-me // Computer Networks. 1999. V. 31. P. 2435-2463.

50. A New Intrusion Detection System for the Internet of Things via Deep Learning [Електронний ресурс]. URL: https://www.mdpi.com/1424-8220/22/10/3607 (дата звернення: 17.11.2022).

51. Computational Intelligence Approaches in Developing Cyberattack [Електронний ресурс]. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8956412/ (дата звернення: 17.11.2022).

52. Intrusion Detection System Using Deep Neural Network [Електронний ресурс]. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155781 (дата звернення: 17.11.2022).

53. Survey of intrusion detection systems: techniques, datasets [Електронний ресурс]. URL: https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7 (дата звернення: 17.11.2022).

54. S. Thaseen and C. A. Kumar, "An analysis of supervised tree based

classifiers for intrusion detection system," in 2013 international conference on pattern recognition, informatics and Mobile engineering, 2013, pp. 294–299.

55. Why Artificial Neural Networks (ANN) Technology Offers a Promising [Електронний ресурс]. URL: https://resources.infosecinstitute.com/topic/why-artificial-neural-networks-ann-technology-offers-a-promising-future-in-idsips/ (дата звернення: 17.11.2022).

56. Набір даних NSL-KDD. Режим доступу: URL: https://www.kaggle.com/hassan06/nslkdd (дата звернення 14.10.2022).

57. Опис типів атак в наборі даних KDDCUP99. Режим доступу: URL: http://kdd.ics.uci.edu/databases/kddcup99/training_attack_types (дата звернення 14.10.2022).

58. Visual Studio: IDE and Code Editor for Software Developers and Teams [Електронний ресурс]. URL: https://visualstudio.microsoft.com/ (дата звернення 17.11.2022).

59. C++Builder: Software Overview - Embarcadero [Електронний ресурс]. URL: https://www.embarcadero.com/products/cbuilder (дата звернення 17.11.2022).

60. Delphi: IDE Software Overview - Embarcadero [Електронний ресурс]. URL: https://www.embarcadero.com/ru/products/delphi (дата звернення 17.11.2022).

ДОДАТКИ

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

ЗАТВЕРДЖУЮ
Голова секції "Управління інформаційною безпекою" кафедри МБІС
д.т.н., професор
_____ Юрій ЯРЕМЧУК
"24" вересня 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

«Підвищення стійкості ідентифікації вторгнень у комп'ютерну систему на основі нейронних мереж

08-72.МКР.003.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
д.ф., ст. викл. каф. МБІС
_____ Салієва О. В

Вінниця – 2022 р.

Додаток А. Технічне завдання

1. Найменування та область застосування

Програмний засіб для підвищення стійкості ідентифікації несанкціонованих дій і атак в комп'ютерних мережах. Область застосування: захист інформаційних ресурсів від несанкціонованого доступу до комп’ютерних мереж.

2. Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №203 від 14. 09. 2022 р.

3. Мета та призначення розробки

3.1 Мета розробки: розробка ефективного захисту від атаки на комп’ютерні мережі за допомогою нейромережі.

3.2 Призначення: розроблений програмний засіб виконує захист від втручань за допомогою навченої нейромережі.

4. Джерела розробки

4.1. Зоріна Т.І. Системи виявлення і запобігання атак в комп’ютерних мережах / Т.І. Зоріна // Вісник східноукраїнського національного університету імені Володимира Даля. – 2013. – № 15 (204) ч.1. – С. 48 – 54

4.2. Методи аналізу та моделювання безпеки розподілених інформаційних систем: навч. посіб. / В.В. Литвинов, В.В. Казимир, І.В. Стеценко та ін. – Чернігів: Чернігівський національний технологічний університет, 2016. – 254 с.

4.3. Diogenes Y. Cybersecurity - Attack and Defense Strategies / Yuri Diogenes, Erdal Ozkaya. - Packt Publishing Ltd., Livery Place,35 Livery Street, Birmingham, B3 2PB, UK, 2018. - 354 pp.

4.4. Joseph Migga Kizza J.M. Guide to Computer Network Security, Fourth Edition. - Springer International Publishing AG 2017. - 569 pp

5. Вимоги до програми

5.1 Вимоги до функціональних характеристик:

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

5.1.2 Програма має бути реалізована за допомогою нейромережі;

5.1.3 Реалізація методу не повинна вимагати спеціальних ліцензійних програмних додатків;

5.2 Вимоги до надійності:

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

5.2.2 Дані стосовно навчання нейромережі, мають бути зберігатися в окремому файлі;

5.2.3 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

для сервера бази даних:

Процесор - 2 x IntelXeon3 ГГц;

обсяг оперативної пам'яті – 16 ГБ;

дискова підсистема – 4 x 146 ГБ;

Дисковод компакт-дисків (DVD-ROM);

Мережевий адаптер - 100 Мбіт/с.

для ПК користувача:

процесор - Intel Pentium 1,5 ГГц;

обсяг оперативної пам'яті – 256 МБ;

дискова пам'ять – 40 ГБ;

мережевий адаптер - 100 Мбіт/с.

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

6. Вимоги до програмної документації

Обов’язкова поетапна інструкція для майбутніх користувачів, наведена в розділі 3.3

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

8. Техніко-економічні показники

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

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

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	01.09.2022	15.09.2022
2	Аналіз предметної області обраної теми	15.09.2022	20.09.2022
3	Апробація отриманих результатів	20.09.2022	30.09.2022
4	Розробка алгоритму роботи	01.10.2022	10.10.2022
5	Написання магістерської роботи на основі розробленої теми	01.10.2022	20.11.2022
6	Розробка економічної частини	15.11.2022	20.11.2022
7	Попередній захист магістерської кваліфікаційної роботи	24.11.2022	25.11.2022
8	Перевірка магістерської кваліфікаційної роботи на наявність плагіату	28.11.2022	30.11.2022
9	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	01.12.2022	10.12.2022
10	Захист магістерської кваліфікаційної роботи	19.12.2022	21.12.2022

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ В.О. Гайдей

Проведемо постановку технічного завдання магістерської роботи:

1. Проаналізувати сучасний стан проблеми захисту комп’ютерної мережі.

2. Окреслити головні напрямки розробки складових системи захисту:

– мережний периметр вузлів та каналів передачі даних;

– маршрутизатори з фільтрацією пакетів;

– транслятори мережних адрес;

– транслятори адрес основних та альтернативних портів.

3. Провести розробку підсистеми захисту від розподілених мережних атак за допомогою нейромережі:

– комп’ютери з явними вразливостями;

– мережі з вразливостями.

4. Запропонувати перелік рекомендацій щодо застосування систем силового, розподіленого захисту комп’ютерної мережі.

5. В результаті виконання роботи має бути розроблено систему захисту комп'ютерної корпоративної мережі з різними принципами протидії атакам та несанкціонованим вторгненням за допомогою нейромережі.

Додаток Б. Лістинг програми

```
unit Main;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, DateUtils, Grids, ComCtrls, Tabnotbk, StdCtrls, Buttons, ExtCtrls, SDL_vector, SDL_kohonen, SDL_rchart, SDL_ntabed, SDL_NumIO, SDL_NumLab, SDL_math1, SDL_filesys;

type

 TMainForm = class(TForm)

   Panel1: TPanel;

   NoteBk: TTabbedNotebook;

   RCKohMap: TRChart;

   TEData: TNTabEd;

   OpenDialog1: TOpenDialog;

   SBAlpha: TScrollBar;

   SBSteps: TScrollBar;

   NLAlpha: TNumLab;

   NLSteps: TNumLab;

   NIOysize: TNumIO;

   NIOxsize: TNumIO;

   Label1: TLabel;

   Label2: TLabel;

   NIONeighb: TNumIO;

   Label3: TLabel;

   SaveDialog1: TSaveDialog;

   KohMap1: TKohonen;

   RowVect: TVector;

   Panel2: TPanel;

   Panel3: TPanel;

   Label4: TLabel;

   BButLoad: TButton;

   BButGenerate: TButton;

   Splitter2: TSplitter;

   Panel4: TPanel;

   Label5: TLabel;
```

```
    Label6: TLabel;
    BButTrainIt: TButton;
    BButContTrain: TButton;
    CBShowTraining: TCheckBox;
    BButAbort: TButton;
    ProgressBar1: TProgressBar;
    Panel5: TPanel;
    procedure SBAlphaChange(Sender: TObject);
    procedure SBStepsChange(Sender: TObject);
    procedure KohMap1Feedback(Sender: TObject; PercentDone: Integer);
    procedure BButLoadClick(Sender: TObject);
    procedure BButGenerateClick(Sender: TObject);
    procedure BButTrainItClick(Sender: TObject);
    procedure BButContTrainClick(Sender: TObject);
    procedure BButAbortClick(Sender: TObject);
  private
    procedure ShowCurrentNetResponse;
  public
  end;

var
  MainForm: TMainForm;

implementation

{$R *.DFM}

//кроки навчанна
const
  MaxSteps = 20;
  Steps : array[1..MaxSteps] of integer =
                        (5,10,15,20,30,50,75,
                         100,150,200,300,500,750,
                         1000,1500,2000,3000,5000,7500,10000);
//масив кольорів на графі
  MaxPreDefColors = 15;
  PredefinedColors : array [0..MaxPreDefColors-1] of TColor =
            (clRed, clBlue, clLime, clYellow, clFuchsia,
```

```
            clBlack, clMaroon, clGreen, clOlive, clNavy,
            clPurple, clTeal, clGray, clSilver, clAqua);

//завантаження даних
procedure TMainForm.BButLoadClick(Sender: TObject);
var
  i,j,k,ParamsCount,StringsCount: integer;
  f : TextFile;
  s,val : string;
  str :TStringList;
  lastval : double;
begin
  NoteBk.PageIndex := 0;
  OpenDialog1.Filter := 'Навчальна вибірка|*.trdata';
  if OpenDialog1.Execute then
  begin
    BButGenerate.Enabled := false;

    //читаємо файл даних з навчальною вибіркою
    AssignFile(f, OpenDialog1.FileName);
    Reset(f);
    ParamsCount := 0;
    StringsCount := 0;
    //визначаємо кількість параметрів
    ReadLn(f, s);
    for i := 2 to length(s) do
    if (s[i] <> ' ') and (s[i-1] = ' ') then inc(ParamsCount);
    //задаємо к-сть параметрів
    TEData.NrOfColumns := ParamsCount;

    //визначаємо кількість строк даних
    while not eof(f) do
    begin
      ReadLn(f, s);
      inc(StringsCount);
    end;
    CloseFile(f);
```

```
//задаємо к-сть строк даних
TEData.NrOfRows := StringsCount;

NoteBk.PageIndex := 0;
TEData.SuppressPaint := true;

//зчитуємо навчальну вибірку
AssignFile(f, OpenDialog1.FileName);
Reset(f);

i := 0;
while not eof(f) do
begin
 ReadLn(f, s);
 inc(i);

 //якщо не строка заголовку, тобто перша, читаємо дані
 if i > 1 then
 begin
  j := 0;
  //розбиваємо строку
  str := TStringList.create;
  str.text := stringReplace(s, ' ', #13#10, [rfReplaceAll]);
  for j := 0 to str.count-1 do
  begin
   val := str[j];

   //аналізуємо отриманні значення

   //protocol
   if val = 'tcp' then val := '0';
   if val = 'udp' then val := '1';
   if val = 'icmp' then val := '2';
   //service
   if val = 'auth' then val := '1';
   if val = 'bgp' then val := '2';
   if val = 'courier' then val := '3';
   if val = 'csnet_ns' then val := '4';
```

```
if val = 'ctf' then val := '5';

if val = 'daytime' then val := '6';

if val = 'discard' then val := '7';

if val = 'domain' then val := '8';

if val = 'domain_u' then val := '9';

if val = 'echo' then val := '10';

if val = 'eco_i' then val := '11';

if val = 'ecr_i' then val := '12';

if val = 'efs' then val := '13';

if val = 'exec' then val := '14';

if val = 'finger' then val := '15';

if val = 'ftp' then val := '16';

if val = 'ftp_data' then val := '17';

if val = 'gopher' then val := '18';

if val = 'hostnames' then val := '19';

if val = 'http' then val := '20';

if val = 'http_443' then val := '21';

if val = 'imap4' then val := '22';

if val = 'IRC' then val := '23';

if val = 'iso_tsap' then val := '24';

if val = 'klogin' then val := '25';

if val = 'kshell' then val := '26';

if val = 'ldap' then val := '27';

if val = 'link' then val := '28';

if val = 'login' then val := '29';

if val = 'mtp' then val := '30';

if val = 'name' then val := '31';

if val = 'netbios_dgm' then val := '32';

if val = 'netbios_ns' then val := '33';

if val = 'netbios_ssn' then val := '34';

if val = 'netstat' then val := '35';

if val = 'nnsp' then val := '36';

if val = 'nntp' then val := '37';

if val = 'ntp_u' then val := '38';

if val = 'other' then val := '39';

if val = 'pm_dump' then val := '40';

if val = 'pop_2' then val := '41';

if val = 'pop_3' then val := '42';
```

```
if val = 'printer' then val := '43';
if val = 'private' then val := '44';
if val = 'red_i' then val := '45';
if val = 'remote_job' then val := '46';
if val = 'rje' then val := '47';
if val = 'service' then val := '48';
if val = 'shell' then val := '49';
if val = 'smtp' then val := '50';
if val = 'sql_net' then val := '51';
if val = 'ssh' then val := '52';
if val = 'sunrpc' then val := '53';
if val = 'supdup' then val := '54';
if val = 'systat' then val := '55';
if val = 'systat' then val := '56';
if val = 'telnet' then val := '57';
if val = 'tftp_u' then val := '58';
if val = 'tim_i' then val := '59';
if val = 'time' then val := '60';
if val = 'urh_i' then val := '61';
if val = 'uucp' then val := '62';
if val = 'uucp_path' then val := '63';
if val = 'vmnet' then val := '64';
if val = 'whois' then val := '65';
if val = 'X11' then val := '66';
if val = 'Z39_50' then val := '67';
if val = 'https' then val := '68';
//flag
if val = 'OTH' then val := '0';
if val = 'REJ' then val := '1';
if val = 'RSTO' then val := '2';
if val = 'RSTR' then val := '3';
if val = 'S0' then val := '4';
if val = 'SF' then val := '5';
if val = 'SH' then val := '6';
//label
if val = 'back' then val := '0';
if val = 'buffer_overflow' then val := '1';
if val = 'ftp_write' then val := '2';
```

```
        if val = 'guess_passwd' then val := '3';
        if val = 'imap' then val := '4';
        if val = 'ipsweep' then val := '5';
        if val = 'neptune' then val := '6';
        if val = 'normal' then val := '7';
        if val = 'phf' then val := '8';
        if val = 'pod' then val := '9';
        if val = 'portsweep' then val := '10';
        if val = 'rootkit' then val := '11';
        if val = 'satan' then val := '12';
        if val = 'smurf' then val := '13';
        if val = 'teardrop' then val := '14';
        if val = 'warezclient' then val := '15';

        for k := 1 to Length(val) do
        begin
          if val[k] = '.' then val[k] := ',';
        end;

        lastval := val.ToDouble;

        //нульові та ненульові за тривалістю дані позначимо різними кольорами
        if str[0] = '0' then TEData.RowAttrib[i] := 1
        else TEData.RowAttrib[i] := 2;

        //додаємо на вхід мережі
        TEData.Data[j,i-1] := lastval;
      end;
      str.free;
    end;
  end;
  CloseFile(f);

  TEData.SuppressPaint := false;
  BButTrainit.Enabled := true;
 end;
end;
```

```
//генерація випадкового набору даних для навчання
procedure TMainForm.BButGenerateClick(Sender: TObject);
var
  i,j : integer;
  rnd : double;
begin
  NoteBk.PageIndex := 0;
  TEData.SuppressPaint := true;
  //задаємо 10 параметрів
  TEData.NrOfColumns := 10;
  //генеруємо 200 строк даних
  TEData.NrOfRows := 200;
  for i:=1 to TEData.NrOfRows do
    begin
    //генеруємо випадкове число
    rnd := random;
    if rnd > 0.5
      then begin
           TEData.RowAttrib[i] := 1;
           for j:=1 to TEData.NrOfColumns do
             begin
             TEData.Data[j,i] := 0.5 + sqr(rnd) + 0.1*random;
             end;
          end
      else begin
           TEData.RowAttrib[i] := 2;
           for j:=1 to TEData.NrOfColumns do
             begin
             TEData.Data[j,i] := sqrt(rnd) + 0.5*random;
             end;
          end;
    end;
  TEData.SuppressPaint := false;
  BButTrainit.Enabled := true;
end;

//регулювання коефіцієнту альфа
procedure TMainForm.SBAlphaChange(Sender: TObject);
```

```
begin
  NLAlpha.Value := SBAlpha.Position/100;
end;

//регулювання кількості кроків навчання мережі
procedure TMainForm.SBStepsChange(Sender: TObject);
begin
  NLSteps.Value := Steps[SBSteps.Position];
end;

//візуалізація результату навчання мережі
procedure TMainForm.ShowCurrentNetResponse;
var
  ix, iy : integer;
  dist   : double;
  i      : integer;
begin
  //очищуємо граф
  RCKohMap.ClearGraf;
  RowVect.NrOfElem := TEData.NrOfColumns;
  //заповнюємо граф
  for i:=1 to TEData.NrOfRows do
    begin
    TEData.Data.NumericData.CopyRowToVec (RowVect,i,1,TEData.NrOfColumns);
    KohMap1.ApplyIt (RowVect, Ix, Iy, Dist);
    RCKohMap.DataColor := PredefinedColors[TEData.RowAttrib[i] mod 16];
    RCKohMap.MarkAt (ix+0.6*random-0.3, iy+0.6*random-0.3, 11);
    end;
  //виводимо граф
  RCKohMap.SetRange (1, 0, 0, KohMap1.SizeX+1,KohMap1.SizeY+1);
  RCKohMap.ShowGraf;
end;

//процес навчання мережі
procedure TMainForm.BButTrainItClick(Sender: TObject);
begin
  BButContTrain.Enabled := false;
  BButAbort.Enabled := true;
```

```
  NoteBk.PageIndex := 1;
  SBAlpha.Enabled := false;
  SBSteps.Enabled := false;
  //передаємо дані на вхід мережі
  KohMap1.TrainData := TEData.Data.NumericData;
  //стандартизуємо дані
  KohMap1.StandardizeData;
  //встановлюємо архітектуру мережі
  KohMap1.SizeX := round(NIOxSize.Value);
  KohMap1.SizeY := round(NIOySize.Value);
  //встановлюємо параметри мережі
  KohMap1.InitialAlpha := SBAlpha.Position/100; //коефіцієнт альфа
  KohMap1.NrOfTrnSteps := Steps[SBSteps.Position]; //кількість кроків навчання
  KohMap1.InitialNeighbors := round(NIONeighb.Value); //кількість сусідів
  //запускаємо процес навчання мережі
  KohMap1.TrainIt;
  Show;
  //показуємо результат навчання
  ShowCurrentNetResponse;
  SBAlpha.Enabled := true;
  SBSteps.Enabled := true;
  BButContTrain.Enabled := false;
  BButAbort.Enabled := false;
end;

//продовження навчання мережі
procedure TMainForm.BButContTrainClick(Sender: TObject);
begin
  Enabled := false;
  //передаємо дані мережі
  KohMap1.TrainData := TEData.Data.NumericData;
  //стандартизуємо дані
  KohMap1.StandardizeData;
  //продовжуємо навчання
  KohMap1.ContinueTraining;
  Enabled := true;
  //показуємо результат навчання
  ShowCurrentNetResponse;
```

```
  SBAlpha.Enabled := true;
  SBSteps.Enabled := true;
  Show;
end;

//перервати процес навчання
procedure TMainForm.BButAbortClick(Sender: TObject);
begin
  BButContTrain.Enabled := true;
  KohMap1.AbortTraining;
end;

//зворотний зв'язок від мережі
procedure TMainForm.KohMap1Feedback(Sender: TObject; PercentDone: Integer);
begin
  //показуємо процент прогресу навчання мережі
  ProgressBar1.Position := PercentDone;
  //якщо відмічено пропорець "показати процес навчання"
  if CBShowTraining.Checked then
  //візуалізуємо прогрес навчання мережі
    ShowCurrentNetResponse;
  Application.ProcessMessages;
end;
end.
```

Додаток В. Ілюстративний матеріал (презентація)

Актуальність дослідження

- Комп'ютерні системи вразливі до вторгнень.
- **Вторгнення** - це отримання несанкціонованого доступу до системи з метою порушення її безпеки. Мета кібератаки – отримати несанкціонований доступ до системи з метою отримання конфіденційної інформації. Зловмисники також можуть спробувати викрасти або змінити інформацію, отриману із системи, до якої вони отримали несанкціонований доступ. Крім того, зловмисники також прагнуть скомпрометувати доступність, цілісність і конфіденційність інформації в системі.
- Існує кілька підходів до виявлення вторгнень, які можна використовувати для реалізації системи. Ці підходи включають статистичні аномалії, зіставлення шаблонів, інтелектуальний аналіз даних і застосування підходу машинного навчання. Зокрема, варто зосередити увагу на підвищенні стійкості ідентифікації вторгнень в комп'ютерну систему.
- Вирішенням даного питання може бути використання апарату штучних нейронних мереж. Адже завдяки навченим **штучним нейронним мережам** легше аналізувати вторгнення в комп'ютерну мережу.

Мета і задачі дослідження

- **Метою і задачею** роботи є підвищення стійкості ідентифікації вторгнень в комп'ютерну систему за рахунок штучної нейронної мережі.

Основні завдання

- Проаналізувати проблеми безпеки мережі.
- Проаналізувати існуючі методи виявлення вторгнень в мережу і вибрати прототип.
- Вдосконалення методології за рахунок впровадження процесу донавчання.
- Провести комплексне моделювання та аналітичну оцінку запропонованого рішення.

Об'єкт, предмет, методи дослідження

- **Об'єкт дослідження** – застосування глибокого навчання для підвищення стійкості ідентифікації вторгнень в комп'ютерну систему.
- **Предмет дослідження** – застосування глибинної нейромережі для виявлення вторгнень в комп'ютерну систему.
- **Методи дослідження**: методи математичного аналізу для створення моделі нейроподібної мережі, апарат штучних нейронних мереж для створення структури нейроподібної мережі, методи контрольованого та неконтрольованого навчання штучних нейронних мереж для розробки підконтрольного навчання для мережі Кохонена.

Наукова новизна та практична цінність

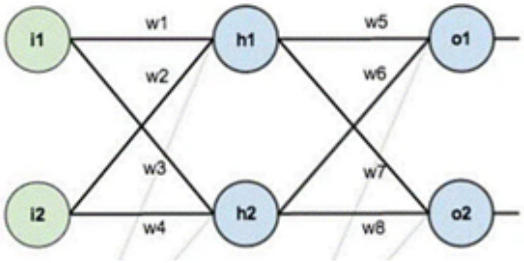
- **Наукова новизна:**

- підвищення стійкості ідентифікації вторгнень в комп'ютерну систему за рахунок використання розробленої архітектури глибинної нейронної мережі.

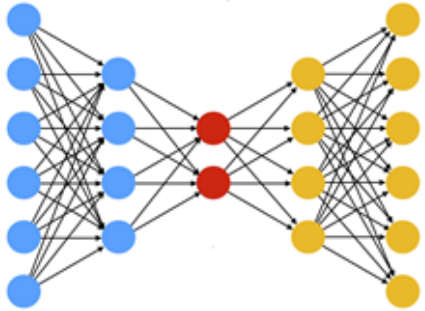
- **Практична цінність одержаних результатів:**

- розроблено програмний продукт для реалізації системи автоматизованого тестування стійкості ідентифікації вторгнень у комп'ютерну систему на основі глибинної нейромережі.

Моделі глибинних нейромереж



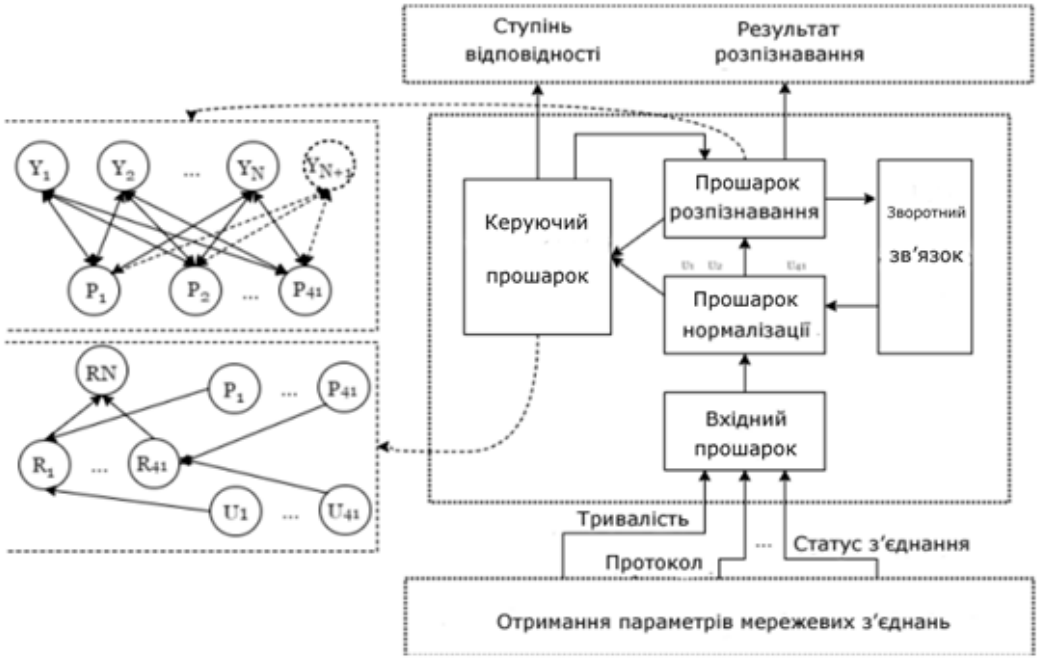
- Модель глибокого навчання



- Структура автокодування

7

Виявлення мережевих атак



- Схема мережі АРТ-2 виявлення мережевих атак

8

Навчання глибоких нейронних мереж

- *Навчання* глибоких нейронних мереж можна здійснити за допомогою алгоритму **зворотного поширення**. Ваговий коефіцієнт визначається:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \frac{\partial C}{\partial \omega_{ij}}$$

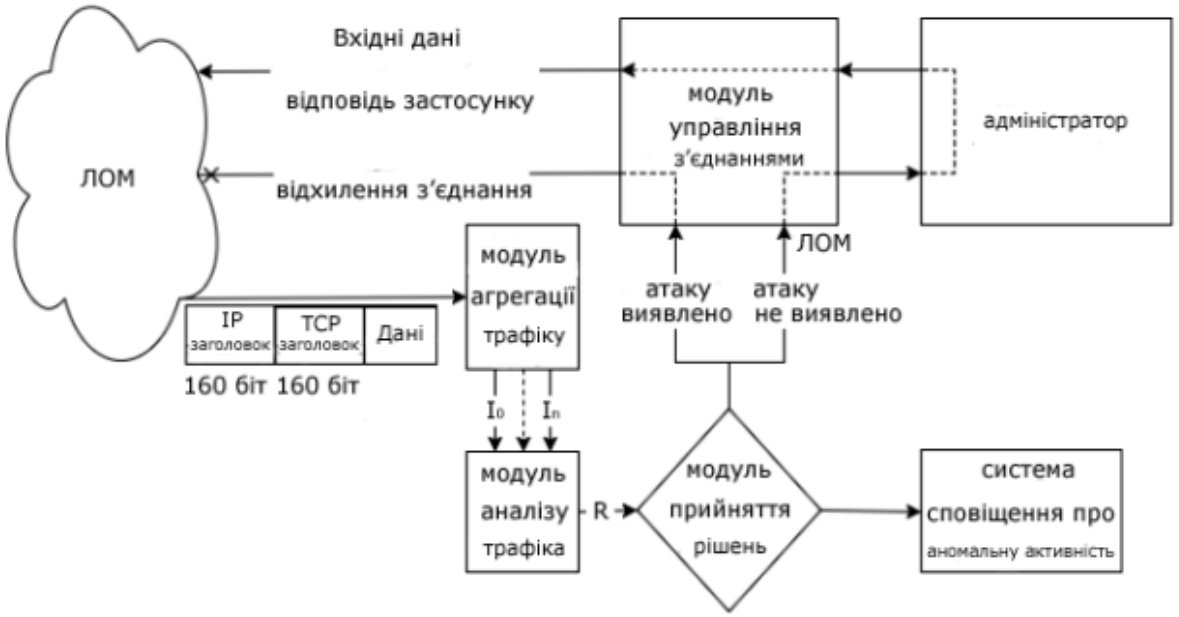
9

Порівняння методів виявлення вторгнень

Метод	Точність	Помилки першого роду	Невідомі атаки	Використання обчислювального ресурсу	Складність розгортання	Складність обслуговування
Сигнатурні	10	ні	ні	низьке	низька	висока
Патерні	10	ні	ні	низьке	низька	висока
Модель орієнтовані	10	ні	ні	низьке	висока	висока
Статистичні	7	так	так	низьке	висока	низька
Data Mining	7-9	так	так	високе	низька	низька
Нейромережеві	7-9	так	так	дуже високе	низька	низька
Нейромережеві імунні	9-10	так	так	дуже високе	низька	низька

10

Ідентифікація входу в комп'ютерну систему



- Загальна схема системи виявлення мережевих атак

Обґрунтування вибору мережі

- **Мережа Кохонена**
- Використовуючи карти Кохонена, що самоорганізуються, ефективність системи методів виявлення мережевих вторгнень на 15% перевищує існуючі рішення.

+15%

Структура мережевого пакету

- Структура мережевого пакету

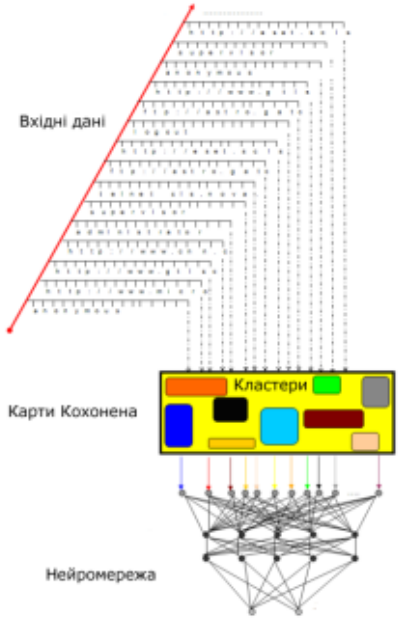
Типи активних атак

- **DoS (denial of service)** атаки - це активні атаки, спрямовані на виникнення ситуації, коли в системі, що атакується відбувається відмова в обслуговуванні. Дані атаки характеризуються генерацією великого обсягу трафіку, що призводить до перевантаження та блокування сервера. Виділяють **шість** підтипів DoS атак: back, land, neptune, Pod, smurf, teardrop.

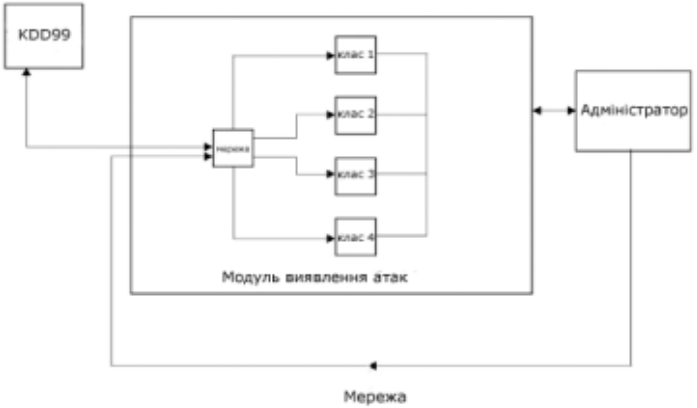
- **U2R (user-to-root)** атаки передбачають отримання зареєстрованим користувачам привілеїв локального суперкористувача (адміністратора). Виділяють **чотири** підтипи U2R атак: buffer_overflow, loadmodule, perl, rootkit.

- **R2L (remote-to-local)** атаки характеризуються отриманням доступу незареєстрованого користувача до комп'ютера з боку віддаленої машини. Виділяють **вісім** підтипів R2L атак: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster.

Модель мережі для виявлення вторгнень

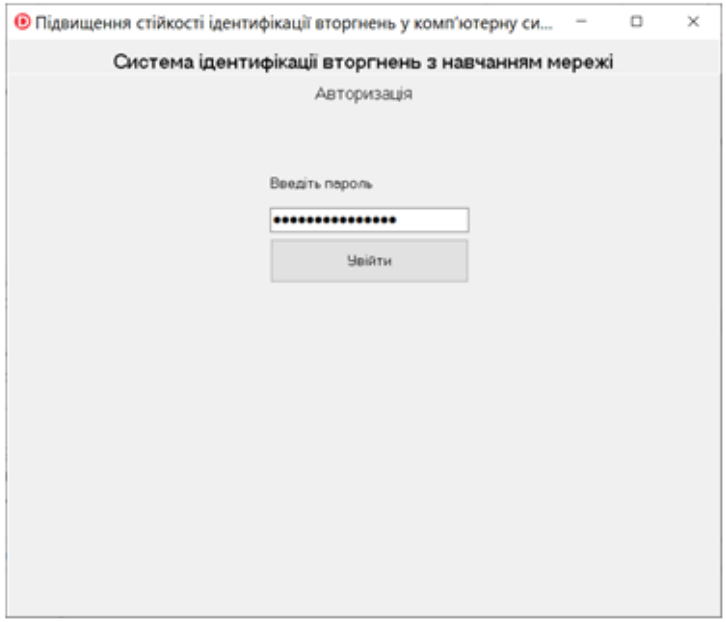


- Архітектура пропонованої системи виявлення атак

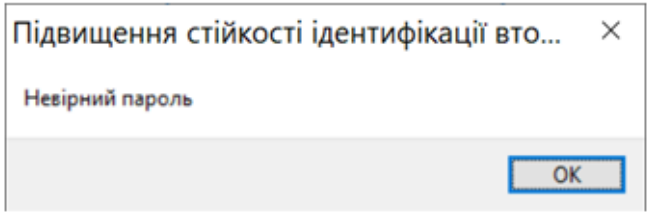


- Схема обробки даних та виявлення мережевих атак

Захист від несанкціонованого використання



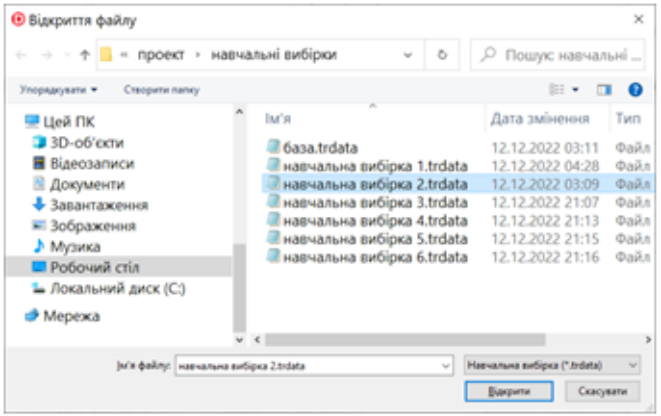
- Захист розробленого програмного засобу від несанкціонованого використання



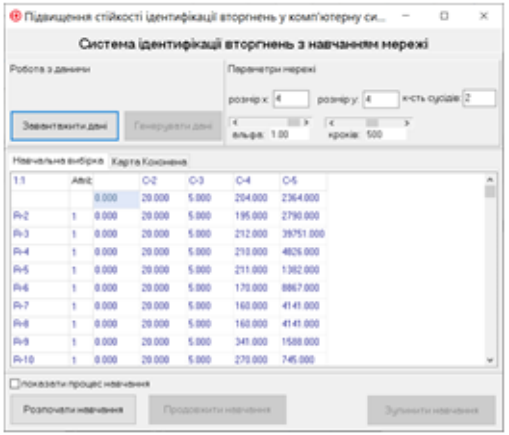
- Помилка авторизації

16

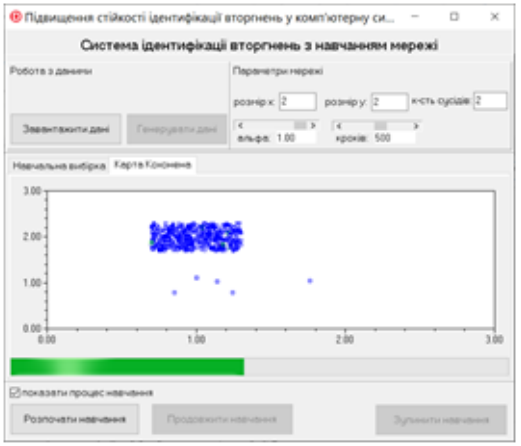
Процес навчання мережі



- Завантаження навчальної вибірки

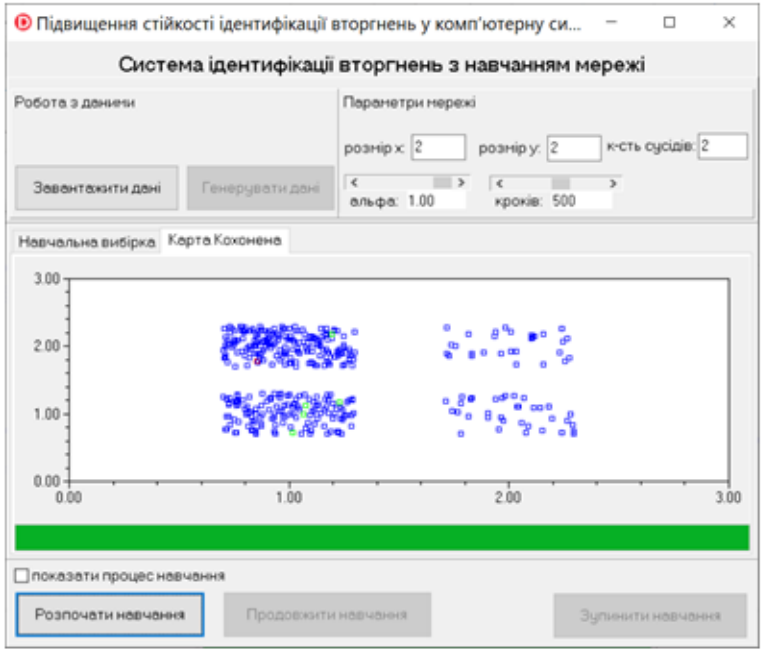


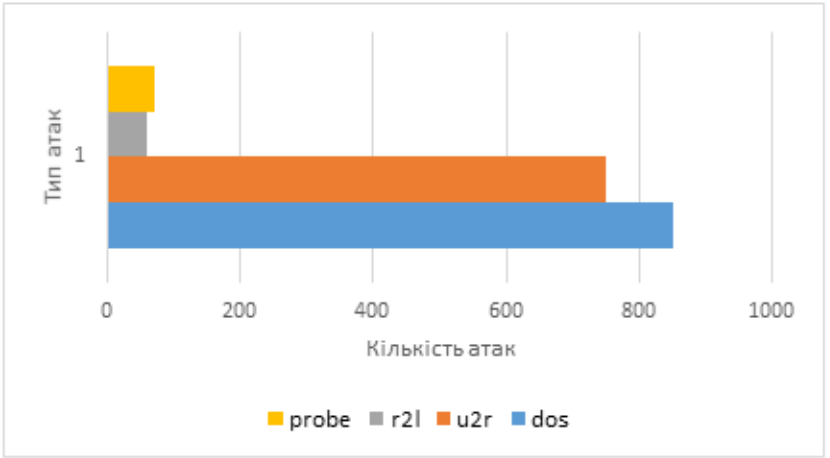
- Результат завантаження навчальної вибірки



- Процес навчання мережі

Результат навчання мережі





- Результат навчання мережі
- Гістограма розподілення атак за типами

18

Висновки

- У магістерській роботі *розв'язано наукову задачу* підвищення стійкості ідентифікації вторгнень у комп'ютерну систему на основі нейронних мереж.

- 1. **Дістала подальший розвиток** модель виявлення вторгнень на основі діагностики вторгнень у комп'ютерну мережу без оновлення сигнатур. Модель заснована на роботі із ознаками вторгнення, які за допомогою математичного апарату мережі Кохонена приймають відповідні рішення про наявність або відсутність несанкціонованих дій в системі.

- 2. Проаналізувавши сучасні математичні моделі, **розвинуто принцип** навчання мережі Кохонена за рахунок встановлення кількості прошарків за вертикаллю та горизонталлю і налаштування вагових коефіцієнтів.

- 3. **Вдосконалено методологію** виявлення вторгнень за рахунок впровадження процесу донавчання нейромережі.

- 4. **Розроблено програмний продукт** із захистом від несанкціонованого використання, який використовує математичний апарат мережі Кохонена для ефективного виявлення вторгнень у мережу.

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ
ЗАПОЗИЧЕНЬ

Назва роботи: Підвищення стійкості ідентифікації вторгнень у комп'ютерну систему на основі нейронних мереж

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ: Кафедра менеджменту та безпеки інформаційних систем
Факультет менеджменту та інформаційної безпеки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 99 % Схожість 1 %

Аналіз звіту подібності (відмітити потрібне):
1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Гайдей В.О.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Салієва О.В.
(підпис) (прізвище, ініціали)