Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки на основі нейромережевих технологій”

КБПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Чорний В.Р.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Буравченко К.О.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*“Інформаційні технології”*_____
Спеціальність *125 “Кібербезпека”*
Освітньо-професійна (освітньо-наукова) програма _*“Кібербезпека”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Чорному Владиславу Романовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки на основі нейромережевих технологій*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки на основі нейромережевих технологій*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

КБПЗ-2025

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

КБПЗ_2025

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Чорний В.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Чорний В.Р. Програмне забезпечення системи кібербезпеки на основі нейромережевих технологій. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки на основі нейромережевих технологій.

Метою розробки є програмне забезпечення системи кібербезпеки на основі нейромережевих технологій.

Результат роботи – програмна реалізація системи кібербезпеки на основі нейромережевих технологій.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, нейромережеві технології

КБПЗ_2025

ABSTRACT

Chorny V.R. Software for a cybersecurity system based on neural network technologies. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a cybersecurity system based on neural network technologies.

The purpose of the development is software for a cybersecurity system based on neural network technologies.

The result of the work is a software implementation of a cybersecurity system based on neural network technologies.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with OS Windows 10/11.

The program was developed in the Python environment.

Keywords: cybersecurity, neural network technologies

КБП3_2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 6
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 7
2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування 10
2.3 Розгорнута постановка завдання 13
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 14
3.1 Опис функціонування системи 14
3.2 Розробка структурної схеми 15
3.3 Розробка функціональної схеми 17
3.4 Розробка діаграми процесів 29
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 30
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 30
4.2 Захист розробленого програмного забезпечення 38
5 ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 43
6 ОСНОВНІ ВИСНОВКИ 47
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 49

КБПЗ_2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЗІ	–	захист інформації
КЗ	–	канал зв'язку
НКД	–	нейромережевий класифікатор-декодер
НМ	–	штучні нейронні мережі
СЗІ	–	системи захисту інформації
ФМТФ	–	функціонал на множині табличних функцій

К6ПЗ_2025

ВСТУП

Актуальність теми. Кібербезпека – це сфера, яка дуже розвивається, і захищає системи, мережі та дані від цифрових атак. Зі збільшенням масштабів Інтернету та розвитком кібератак розробка нових інструментів кібербезпеки стала важливою, особливо для мереж Інтернету речей (IoT). Ця робота містить систематичний огляд застосування підходів глибокого навчання (DL) для кібербезпеки. У цьому документі наведено короткий опис методів DL, які використовуються в кібербезпеці, включаючи мережі глибоких переконань, генеративні змагальні мережі, рекурентні нейронні мережі та інші. Далі ми проілюструємо відмінності між поверхневим навчанням і DL. Крім того, обговорюється переважаючі на даний момент кібератаки в IoT та інших мережах, а також ефективність методів DL для управління цими атаками. Крім того, у цьому документі описано дослідження, які висвітлюють техніку DL, програми кібербезпеки та джерело наборів даних. Далі обговорюється доцільність систем DL для виявлення та класифікації шкідливих програм, виявлення вторгнень та інших частих кібератак, включаючи ідентифікацію типу файлу, спаму та мережевого трафіку. Наш огляд показує, що висока точність класифікації 99,72% досягається за допомогою обмеженої машини Больцмана (RBM) при застосуванні до спеціального набору даних, тоді як довготривала короткочасна пам’ять (LSTM) досягає точності 99,80% для набору даних KDD Cup 99. Нарешті, у цій статті обговорюється важливість кібербезпеки для надійних і практичних систем охорони здоров’я, що керуються IoT.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки на основі нейромережевих технологій.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем на основі нейромережевих технологій.

КБПЗ_2025

– Дослідження системи кібербезпеки на основі нейромережевих технологій.

– Програмна реалізація системи кібербезпеки на основі нейромережевих технологій.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі на основі нейромережевих технологій.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки на основі нейромережевих технологій, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

К6ПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для реалізації методів захисту інформації на основі нейромережевих технологій. Кібербезпека – це повний пакет усіх методів і технологій, що відносяться до захисту мережі, програмного забезпечення та атаки [1, 2]. Механізм кіберзахистів доступний на рівнях мережі, даних, хостів і програм. Деякі інструменти кібербезпеки подобаються брандмауері, в системі звиявлення вторгнень, в системі захисту від проникнення тощо., є активними на кожному кінці, щоб порушити безпеку та зупинити атаки [3, 4]. Проте, зі збільшенням кількості систем, маючих Інтернет-зв'язку, існує ризик атаки з кожним днем. З реалізацією Інтернету речі (Інтернет речей) мережі, кібербезпека є становлення більш важливо, ніж будь-коли. Комп'ютерні мережі IoT є вразливі до багатьох загроз безпеці. Деякі атаки свідомий шаблон можна легко керувати. Проте, нападник розвивається нуль-денні підйоми, де атакує місце, яке тільки буде виявлено слабке місце в системі. Такого нападника немає попереднього запису і атака може пошкодити комп’ютерну систему до вирішення проблеми. Крім того, систему необхідно захищати не тільки від зовнішніх загроз, але також потрібно захищати від внутрішніх загроз, таких як неправильне використання з уповноваженим доступом, який може бути індивідуальним або середнім бути частиною організації. Основним викликом є ознака поза в компроматичних індикаторах системи з життєвого циклу атаки, які можуть мати значущі ознаки майбутнього нападу. Проте, це могло бути складною роботою через величезну кількість даних, що безперервно генеруються з багатьох кібер-увімкнених пристроїв.

КБПЗ_2025

1.2 Область застосування

Data Science використовує широкий спектр даних, створених системою кіберзахисту, в т.ч. інформацію про схему безпеку та управління подіями (SIEM), іноді переповнюється фахівець з охорони з попередніми подіями, що ідентифікують закономірності, пов’язані з виявленням ненормальних подій. Поведінки покращити в кібербезпеці. Гібридне виявлення в об’єднаних аномаліях безпеки виявлення неправильного використання. Ця система в основному використовується для зменшення норми хибно-позитивного значення анонімних атак і підвищення показника виявлення розпізнаних вторгнень. Максимальний DL підходи є гібридними методами [5, 6]. Попередні огляди, тобто в [7-9] мають ілюстроване застосування машинного навчання(ML) для вирішення кіберпов’язаних проблем, але методи глибокого навчання (DL) не були зосереджені на цих документах. Деякі роботи ілюструють підходи DL для кібербезпеки. Ці підходи мають деякі обмеження в програмі кібербезпеки [10, 11].

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки на основі нейромережевих технологій, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ-2025

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Boole Server

Boole Server – це безпечна масштабована система, розроблена для захисту будь-якої делікатної інформації, що дозволяє обмінюватися цією інформацією між колегами, партнерами й клієнтами, коли це необхідно. Ця система гарантує, що дані не будуть викрадені або загублені при передачі або на сервері, і що тільки авторизований одержувач може використовувати, редагувати й переглядати специфічні файли.

Boole Server захищає збережені файли, використовуючи алгоритм шифрування AES з 256-ключем. Ключ для дешифрування файлів зберігається в сховище Boole Server окремо від зашифрованих файлів. Розшифрувати зашифровані файли без авторизації на Boole Server неможливо.

При роботі через глобальну мережу Інтернет браузер користувача встановлює захищене з'єднання з компонентами IIS 6.0/7.0, використовуючи протокол HTTPS і SSL шифрування, далі ISS установлює з'єднання з Boole Server.

Користувачі Boole Server мають можливість доступу до інформації в будь-якому місці (метро, вокзал, готель і т.д.) і в будь-який час доби, без ризику зниження безпеки і їхніх інформацій при наявності доступу до мережі Інтернет.

Boole Server захищає файли, електронну пошту й текст, використовуючи алгоритм шифрування AES 256 біт. Інформація зберігається у вигляді окремих зашифрованих файлів.

КБПЗ-2025

Boole Server забезпечує захист інформації за межами традиційного "периметра безпеки".

Адміністратор сховища даних може бачити докладний звіт про всі операції з даними й відслідковувати дії інших користувачів.

Boole Server Agent забезпечує захист інформації від копіювання, блокуючи функції "створення знімка екрана" і печатки відкритого файлу.

Web Client призначений для підключення до Boole Server шляхом організації захищеного каналу через мережу Інтернет.

Worry-Free Business Security Advanced

Рішення Worry-Free Business Security Advanced дозволяє працювати на ПК, комп'ютерах Mac і мобільних пристроях без побоювань із приводу простоїв і втрати даних. Це програмне забезпечення не вимагає наявності спеціальних знань у сфері IT і не знижує продуктивність мережі й пристроїв.

Основні переваги:

– Можливість зосередитися на рішенні бізнес-завдань, не турбуючись про безпеку IT-систем.

– Простота установки й зручність використання на підприємствах малого бізнесу – не потрібен досвід у сфері IT.

– Блокування погроз в «хмарі», перш ніж вони проникнуть у вашу мережу й на пристрої.

– Фільтрація URL-адрес для запобігання доступу до небажаних веб-сайтам.

– Запобігання поширення ділової інформації за допомогою поштових повідомлень і USB-накопичувачів.

– Захист від фішингу й атак з використанням прийомів соціотехники.

– Зручне керування безпекою мобільних пристроїв.

– Блокування спаму, перш ніж він досягне поштових серверів, за допомогою вхідні до складу пакета рішення Hosted Email Security.

Об'єкти захисту:

– ПК.

КбПЗ_2025

– Комп'ютери Mac.

– Мобільні пристрої.

– Ноутбуки.

– Поштові й файлові сервери.

– Касові термінали.

– USB-накопичувачі.

Worry-Free Business Security Advanced – це єдине рішення для малого бізнесу, що забезпечує повний захист користувачів. Воно блокує погрози ще до того, як вони проникнуть у мережу підприємства. Воно розроблено спеціально для представників малого бізнесу, і його без праці можуть використовувати люди, що не є фахівцями в області IT.

Крім того, це єдине рішення для забезпечення безпеки малих підприємств із інтегрованими функціями захисту мобільних пристроїв.

От ще ряд важливих причин, по яких Worry-Free Business Security можна назвати провідним рішенням для захисту даних малих підприємств:

– Повний захист комп'ютерів всіх ваших співробітників.

– Захист при роботі з електронною поштою, перегляді веб-сайтів, обміні файлами й іншими діями.

– Фільтрація URL-адрес для запобігання доступу до небажаних веб-сайтам і підвищення ефективності роботи співробітників.

– Запобігання втрати даних через поштові повідомлення й USB-накопичувачі.

– Блокування спаму й захист від фішингу й атак з використанням прийомів соціотехники.

– Захист комп'ютерів Windows, Mac і мобільних пристроїв.

– Простої у використанні й невимогливе до ресурсів рішення.

– Простота установки й використання.

– Рішення, розроблене спеціально для малих підприємств і не потребуючих глибоких знань в області IT.

– Забезпечення централізованого контролю й прозорості.

КБПЗ_2025

– Забезпечення захисту з мінімальним впливом на продуктивність.

– Блокування погроз в «хмарі», перш ніж вони проникнуть у вашу мережу й на пристрої.

– «Хмарний» захист на основі інфраструктури Trend Micro Smart Protection Network.

– Блокування погроз ще до їхнього проникнення в мережу підприємства, що знижує навантаження на системи.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктно-орієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними

КбПЗ_2025

у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

– у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;

– гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;

– розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;

– Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;

– у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив’язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп’ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

КБПЗ_2025

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

КБПЗ_2025

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки на основі нейромережевих технологій.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

КБПЗ_2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Глибоке навчання (DL) і машинне навчання (ML)

І ML, і DL є підмножинами штучного інтелекту (AI). Відмінності між ML і DL включають наступне:

– Залежності даних: Продуктивність моделей DL порівняно не краще, ніж традиційні моделі ML для маленького-масштабувати обсяги даних. Причиною цього є DL моделям потрібна велика частина даних, щоб зрозуміти дані бездоганно. З іншого боку, традиційні алгоритми ML використовують встановлені правила [14].

– Інтерфейс обладнання: Графічна обробка одиниці (GPU) можна правильно вважати основним обладнанням для навчання моделей DL. Графічний процесор є в основному для оптимізації матричних процесів ефективнішими, а решту моделей DL вимагають багатоматричні операції. Увімкнено в іншу руку, традиційні MLалгоритми зазвичай не вимагають високої продуктивності машин із графічним процесором [18].

– Обробка особливостей: Процедура керування доменом у функціях екстрактора, щоб зменшити складність даних, названих обробкою ознаки. Візерунки єзазвичай генеруються під час обробки функцій, а отже, ML і Алгоритми DL працюють краще. однак, цей етап – час-трудомісткий, і в цьому випадку потрібні спеціальні знання. Більшість моделей ML складається на функціональність (наприклад, значення пікселів, текстури, форми, розташування тощо) видобутий. Спроба отримати функції рівня відкритих персональних даних є основною відмінністю традиційних Алгоритмів ML і DL [17]. Відповідно, DL зменшує зусилля проектування для вирішення функцій для кожної проблеми.

– Час виконання: великий час виконання потрібен для навчання а Модель

КБПЗ_2025

DL із різноманітними параметрами. Крок навчання займається також більше часу. Навпаки, менше часу виконання (від секунди до кількох годин) необхідний для навчання а ML model. Тим не менше, час, потрібний на етапі тестування, достатній контраст. Моделі DL дуже потрібні короткі порівняння часу тестування з деякими моделями ML.

DL підходи в кібербезпеці

Мережі глибокого навчання

Мережі глибокого навчання (DBNs) представлені в основоположній статті Джеффрі Хінтон. DBN є класом Deep NeuralМережі (DNN). БДН складається з кількох шарів RS прихованих випадкових змін. Крім того, зв'язки між одними шарами та відсутні зв'язки між одиницями всередині шару [12]. Це комбінація ймовірності та статистики з ML і нейронні мережі.

Автокодувальник

Неконтрольований метод – це автокодер, де є вхідний як вектор. Мережа намагається зіставити та вихід так само, як у вхідний вектор. Можна створити зображення нижчої чи вищої розмірності даних збереження вхідних даних і варіювання відтворення вхідних даних за допомогою його розмірності. Дані операції кодування (тобто функціонування) виконуються в мережі з маленьким розміром прихованих шарів. Може відтворюватися автоматичний кодер із шумозаглушенням важливої ролі в порядку до ліквідації в шумі і реконструювати оригінальний вхід із зашумленого введення.

КБПЗ_2025

3.2 Розробка структурної схеми

Моделі DL продемонстрували значні покращення завершено традиційного ML-рішення на основі, підписані методи і правильно-на основі методи в порядку проблеми кібербезпеки. Програма показує результати в кінцевому підсумку, частоту помилкових негативних результатів (FNR), класифікаційну точність, F1-бал, істинно позитивний показник (TPR) тощо. З огляду видно, що дослідники мають зосереджені на шкідливе програмне забезпечення класифікації виявлення

різних типів використання в мережі. Кібер-фізичні автономні системи, які є не тільки датчиками на основі комунікації-увімкнено (наприклад, автомобільні системи), біометрія поведінки (тобто перед динамікою) розглядається як розширення областей для застосування DL.

Коли ми все більше залежимо від мережевих підключених пристроїв, ми побачимо збільшення кількості кібер-фізичних систем та обчислювальних систем, також з якими є свої набір векторні атаки за рахунок унікальної базовій лінії. Для шкідливих програм вторгнення де RBM використовувалися всю DL техніку. RNN були іншим популярним рішенням для вирішення найширшого спектру проблем кібербезпеки (тобто мережеві включення, кібер-фізичні включення, зловмисне програмне забезпечення, хост-включення та ім'я шкідливого домену). Широке використання RBM і автокодерів, близько 50%, є за все, через дефіцит мічених даних і немаркованих даних, попередньо-навчених і добре-налаштованих за допомогою невеликої кількості розпізнаних даних.

Роботи або дані про безперервність можна розглядати як проблему часових рядів. Це вигідно для RNN. Однак деякі тенденції є з зауваженнями. Діяльність різних напрямків охоронного бізнесу дуже різнилася. Домени, створені за допомогою різноманітних методів, здаються, мають найбільшу послідовність DGA-виробляються небезпечні домени з TPR в діапазоні від 1% до 1,5% визначення точності змінюється від 0,9959 до 0,9969, що еквівалентно 96,01 до 99,86%. Методи виявлення мережевих вторгнень, з іншого боку, мають продуктивність від 92,33 до 100 відсоток з TPR від 1,58 до 2,3 відсотка та точністю діапазон від 44 до 99 відсотків. Висока класифікаційна точність99,72% повідомляється для RBM у разі застосування до спеціального набору даних[34], тоді як точність 99,80% досягнута LSTM для Набору даних KDD Cup 99 [36]. Історична здатність виявлення вторгнення в мережі значно була залежно від кількості здійснених атак. Ще один важливий компонент, що приливає в загальну продуктивність, був у навчанні набору відносини між доброякісними і небезпечними даними.

КбПЗ_2025

КБПЗ_2025

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Опишемо методи нейромережевої класифікації для декодування інформації при передачі КЗ із завадами та надамо опис нейромережевого

класифікатора-декодера (НКД) з врахуванням можливості оптимізації його функціонування за критерієм швидкодії обробки зашумленої інформації.

Використання однотипних компонент на основі НМ ФМТФ в структурі НКД дозволило поєднати процес корекції помилок з декодуванням, що впливає на ефективність відтворення інформації при застосуванні кодів великої розмірності.

Аналіз розвитку інформаційних систем та мереж показав тенденцію до пропорційного зростання рівня зашумленості та складності реалізації математичного апарату відтворення інформації.

Введення надлишковості, що лежить в основі методів завадостійкого кодування, суттєво збільшує об'єм вихідних даних відповідно до кратності помилок, ускладнюючи задачу декодування. Отже, необхідною є розробка таких методів та алгоритмів, які дозволяють зменшити тривалість процесу відтворення інформації без втрат у показниках цілісності.

Метод передачі N повідомлень задається довільною парою $[K_N(n), D_N(n)]$, де $K_N(n)$ представляє собою код, $D_N(n)$ – метод декодування, як набір функцій $q_1(\tilde{v}), q_2(\tilde{v}), \ldots, q_N(\tilde{v})$, таких, що $0 \le q_i(\tilde{v}) \le 1$, $i = \overline{1, N}$, $\sum_{i=1}^{N} q_i(\tilde{v}) = 1$. Канал передачі інформації характеризується тим, що для всіх n, i та j задані системи умовних ймовірностей P_{ij} переходу вхідного слова v_i у вихідне $\tilde{v}_j$. Тоді функції $q_i(\tilde{v})$ трактуються як імовірність, з якою приймається рішення, що було передане $i-$те повідомлення, а імовірність помилки для методу передачі дорівнює:

$$P[K_N(n), D_N(n)] = \frac{1}{N} \sum_{v_i \in K_N(n)} \sum_{\tilde{v}_j \in \tilde{V}(n)} P_{ij} \left[1 - q_i(\tilde{v}_j)\right].$$

Відомо, що для заданого коду $K_N(n)$ метод декодування є оптимальним, якщо $q_i(\tilde{v}_j) = 0$ при $\max_{1 \le l \le N} P_{lj} \ne P_{ij}$. Отже, операція декодування полягає у визначенні $\tilde{\Omega}_i \subset \tilde{V}(n)$ для кожного v_i, тоді всі $\tilde{v}_j \in \tilde{\Omega}_i$ вірно декодуються в v_i. Для середньої імовірності правильного декодування слова $v_i \in K_N(n)$ (тобто, вірогідність

КбПЗ_2025

інформації) справедливо:

$$P_{пр}\left[K_N(n), D_N(n)\right] = \frac{1}{N}\sum_{v_i \in K_N(n)}\sum_{\tilde{v}_j \in \tilde{V}(n)} P_{ij} q_i\left(\tilde{v}_j\right) = \frac{1}{N}\sum_{v_i \in K_N(n)}\sum_{\tilde{v}_j \in \tilde{\Omega}_i} P_{ij}.$$

З розглянутих в цьому розділі положень теорії інформації та потенційної завадостійкості випливають принципи функціонування оптимального декодера, що мінімізує імовірність появи помилки при передачі та представляє собою класифікатор. На відміну від поширених нейромережевих класифікаторів, таких як класифікатор Хеммінга, використання НМ ФМТФ для нейромережевого декодування обумовлено наявністю додаткового параметру, оперування яким дозволяє розв'язати проблему відмови від класифікації.

Процес навчання НМ

Процес навчання НМ відбувається на створеній навколо кожного кодового слова v_i області, причому кодове слово вважається еталонним. Області охоплюють можливі помилки при передачі вихідного слова, а кількість помилок, які можливо виправити, визначається кодовою відстанню та обмежується радіусом r_i. Отже, навчання НМ можна розглядати як процес кластеризації за еталоном.

Існує принципова відмінність у навчанні нейромереж з використанням методу зворотного поширення похибки і мереж на основі моделі ФМТФ. При використанні останніх НМ для відтворення складних, неоднорідних на різних ділянках гіперповерхонь, спостерігається ефект погіршення точності. Вихід з такої ситуації полягає в необхідності попереднього розділення вхідних даних, просторово близьких між собою, на групи реалізацій, що обумовлює використання блокової структури з окремо навчених однотипних НМ в запропонованому НКД. Внаслідок ідентичності процесів навчання та функціонування блоків НМ, що входять до складу НКД, надалі будемо розглядати алгоритм стосовно окремого блоку.

Формується навчальна підмножина $\tilde{\Omega} \subset \tilde{V}(n)$ для кодового слова

КБПЗ_2025

$v^0 = \left(a_1^0, a_2^0, \ldots, a_n^0\right)$, що містить слова з можливими помилками при передачі та виглядає наступним чином:

$$\widetilde{\Omega} = \begin{Vmatrix} a_1^0 & a_2^0 & \cdots & a_n^0 \\ a_1^1 & a_2^1 & \cdots & a_n^1 \\ a_1^2 & a_2^2 & \cdots & a_n^2 \\ - & - & - & - \\ a_1^c & a_2^c & \cdots & a_n^c \end{Vmatrix}, \tag{3.1}$$

де $\widetilde{v}^j = \left(a_1^j, a_2^j, \ldots, a_n^j\right)$, $j = \overline{1,c}$ – слова з можливими помилками в вихідному кодовому слові; c – кількість елементів підмножини кодових слів. Перший елемент створеної навчальної вибірки представляє собою, за аналогією з класифікацією, еталон класу, що визначається елементами підмножини.

В режимі навчання елементи підмножини $\widetilde{\Omega}$ подаються на вхід НМ наступного вигляду.

В процесі навчання НМ визначається базовий рядок i_b в матриці (3.1), сума квадратів елементів якого є максимальною. Обчислюється коефіцієнт

$$k_i = \frac{\sum_{j=0}^{c} a_{ij} \cdot a_{i_b j}}{\sum_{j=0}^{c} \left(a_{i_b j}\right)^2}, \quad i = \overline{1,n}, \tag{3.2}$$

за допомогою якого навчальна матриця видозмінюється згідно:

$$a'_{ij} = a_{ij} - k_i \cdot a_{i_b j}, \quad j = \overline{0,c},$$

для забезпечення мінімуму різниці в сенсі критерію найменших квадратів, тобто

$$\sum_{j=1}^{c} \left(a_{ij} - k_i \cdot a_{i_b j}\right)^2 \to \min_i$$

З одержаної величини вираховується максимально допустимий радіус сфери, що обмежує створений клас елементів

$$r_i = \sum_{j=1}^{c} a'_{ij}$$

КбПЗ 2025

На виході НМ формується величина середньоквадратичного відхилення між компонентами класу та еталоном, що базується на обчисленні евклідової віддалі.

Описані вище процедури здійснюються для всіх навчальних множин; таким чином, кожному з класів ставиться у відповідність навчена мережа, що виступає базовим елементом реалізації розробленого НКД.

Відповідно до методу класифікації за критерієм мінімуму віддалі, функціонування НКД полягає у визначенні відстані від слова на вході схеми та всіх еталонів класів, кожен з яких представляє собою окремо навчену НМ. Тривіальна модель для декодування зашумлених слів передбачає використання одного блоку НМ, що навчається на всій області кодових слів. В процесі досліджень встановлено доцільність розпаралелення процесу декодування шляхом створення системи однотипних блоків НМ загальною кількістю m у відповідності з кількістю класів, причому кожен блок при навчанні оперує тільки одним кодовим словом та областю навколо нього. Результати обчислення розглянутої відстані подаються на вхід мінімізатора, який визначає найменшу величину з поданих даних. Подальше функціонування НКД полягає в співставленні еталона визначеного класу згідно попередніх результатів.

Проведені дослідження з метою створення оптимальної структури НКД дозволили виявити ряд характерних особливостей в геометричному представленні кодів, які можна представити окремими областями, що не перетинаються. Область навколо кодового слова можна повністю описати та визначити, використовуючи поняття центру області та її радіусу, що обумовлює використання неповної навчальної множини при одержанні достатньої точності результатів.

На основі порівняльного аналізу та експериментальних досліджень розробленого в дисертації методу та інших відомих методів обробки зашумленої інформації встановлено, що запропонований метод нейромережевого

КБПЗ-2025

декодування суттєво підвищує ефективність відтворення інформації та дозволяє продуктивно використовувати коди великої розмірності.

Представимо розроблені методи та алгоритми забезпечення конфіденційності в СЗІ на основі процесів обробки інформації в моделі біологічного нейрона.

Розроблений нейромережевий алгоритм шифрування використовує принципи формування вагових коефіцієнтів при неітераційному навчанні НМ ФМТФ та оперує ключовою інформацією великої розмірності, що відповідає критеріям стійкості шифрів.

Проведений в першому розділі аналіз нейропарадигм дозволив виділити особливість нейромереж прямого поширення за архітектурою зв'язків між шарами, що полягає в здатності до навчання за прикладами навчальної множини. Процес навчання розглядається як налаштування архітектури мережі та формування вагових коефіцієнтів зв'язків між нейронами відповідно до даних навчальної множини (у вигляді <біжучий вхід X^0>, <бажаний вихід Y^0>).

Отримані в процесі навчання вагові коефіцієнти визначають результат функціонування Y нейромережі у відповідному режимі при оперуванні вхідними даними X. Таким чином, НМ повністю описується ваговими коефіцієнтами та передатною функцією, за допомогою якої відбувається перетворення сумарного імпульсу $Y = f(X, W)$.

Незважаючи на те, що алгоритм навчання за допомогою процедури зворотного поширення відносять до загальноприйнятих та з його допомогою були розв'язано багато практичних задач, він характеризується низькою швидкодією. Це пов'язано з ітераційною зміною значень параметрів мережі, що оперують тільки одним вектором-реалізацією з навчальної вибірки, а також із лінійним характером досягнення точності самого алгоритму.

Новий принципово відмінний підхід до побудови, налаштування та використання штучних НМ базується на аналогіях між просторовим представленням об'єкта досліджень та моделлю мережі ФМТФ. Особливістю

КБПЗ_2025

навчання цієї мережі є попереднє встановлення нульових вагових коефіцієнтів, що усуває залежність функціонування від зовнішніх оцінок.

Принцип неітераційного навчання НМ прямого поширення, проаналізований в бакалаврській роботі, дозволив розглядати елементи криптографічних перетворень шеннонівської секретної системи як складові процесу налаштування параметрів мережі.

Секретна система визначається абстрактно як деяка множина відображень одного простору (множини можливих повідомлень) в інший простір (множину можливих шифртекстів).

Кожне відображення з цієї множини відповідає способу шифрування за допомогою ключа.

Відкритий текст M перетворюється за допомогою алгоритму шифрування E та ключа K в зашифрований текст (шифртекст) $C = E(K, M)$.

Зворотне перетворення шифртексту у відкритий текст відбувається за допомогою алгоритму дешифрування D, тобто $M = D(K, C)$.

Якщо для базових режимів нейромережі покласти, що $X^0 \cong K$, $Y^0 \cong M$ та $W \cong C$ то задача неітераційного навчання НМ з подальшим функціонуванням переходить у задачу симетричного шифрування.

Алгоритм шифрування

Запропонований в бакалаврській роботі алгоритм шифрування можна класифікувати як моноалфавітний блочний шифр внаслідок наявності процесу попереднього розбиття відкритого тексту $M = (M_1, M_2, \ldots, M_l)$ на n-грами $M_p = \left(M_1^{(p)}, M_2^{(p)}, \ldots, M_n^{(p)}\right)$, $p = \overline{1, l}$ за аналогією з розбиттям навчальної вибірки нейромережі на вектори-реалізації.

Ключ

Ключем служить матриця $K = \left|k_{ij}\right|$, де $k_{ij} \in Q, i = \overline{1, n}, j = \overline{1, m}$ згенеровані випадковим чином, а простір ключів $Q = [-1; 1]$, обумовлений специфікою функцій активації нейромережі.

КБПЗ_2025

Шифрування

Процес шифрування відбувається в фазі навчання нейромережі за принципом обчислення вагових коефіцієнтів синаптичних ваг:

$$\begin{matrix} k_{11} & k_{12} & \dots & k_{1m} & M_1^{(p)} & c_1^{(p)} \\ k_{21} & k_{22} & \dots & k_{2m} & M_2^{(p)} & c_2^{(p)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ k_{n1} & k_{n2} & \dots & k_{nm} & M_n^{(p)} & c_n^{(p)} \end{matrix}$$

Для всіх n-грам відкритого тексту створюється матриця шифртексту шляхом перетворення елементів ключової матриці функцією активації на входах мережі:

$$E(M_p) = f(K, M_p) = C_p, \quad p = \overline{1, l},$$

де $C_p = \left(c_1^{(p)}, c_2^{(p)}, \dots, c_n^{(p)}\right)$, $c_i^{(p)} = f\left(k_{i1}, k_{i2}, \dots, k_{im}, M_i^{(p)}\right)$, $i = \overline{1, n}$.

Дешифрування

Дешифрування здійснюється в режимі функціонування мережі із використанням оберненої функції f^{-1}, а саме функції активації на виходах мережі

$$D(C_p) = f^{-1}(K, C_p) = M_p, \quad p = \overline{1, l}$$

Криптоаналіз

Якщо припустити, що оптимальним методом криптоаналізу алгоритму шифрування є метод прямого перебору всіх можливих ключів, то його швидкодія повністю визначається довжиною ключа та потужністю обчислювальних засобів.

Враховуючи тенденцію стрімкого зростання потужності комп'ютерних технологій, оптимальне дешифрування перехопленого повідомлення необхідно починати в момент, коли часові затрати на нього будуть мінімальними. Тобто, стійкість алгоритмів шифрування можна оцінювати за новим об'єктивним критерієм.

Нехай $x(t)$ – неперервна функція швидкодії обчислювальних засобів в момент часу t. Виходячи з припущення Б.Шнайєра про ріст потужності комп'ютерних технології (10 разів за кожні 5 років) та взявши за початкову точку

КБПЗ 2025

відліку 1946 рік (поява першого комп'ютера, $t = 0$), функція виглядає наступним чином $x(t) = P_0 \cdot 10^{t/5}$, де P_0 – потужність першого комп'ютера. Модель для розрахунку часу життя ключа складається з двох діючих факторів: моменту початку дешифрування та складності дешифрування. Мінімум цієї функції визначає максимальний час життя

$$t_{\max} = \left(t + \frac{|Q|}{2 \cdot x(t) \cdot P_1} \right) \to \min ,$$

де $|Q|$ – потужність множини ключів; P_1 – коефіцієнт для переходу від операцій/секунду до операцій/рік.

Аналіз запропонованого нейромережевого алгоритму шифрування проводився за шеннонівськими критеріями оцінки алгоритмів криптографічного захисту. Встановлено, що цей алгоритм відповідає оптимальному критерію кількості секретності, оскільки довільний об'єм перехопленої інформації не дозволяє ефективно провести криптоаналіз внаслідок великого обсягу ключових даних. Алгоритм навчання та функціонування нейромережі базується на використанні простих математичних операцій, тому процеси шифрування і дешифрування не потребують значних затрат часу.

На основі розроблених у бакалаврській роботі методів та алгоритмів запропоновано і створено СЗІ для забезпечення комплексного захисту інформації. На основі описаної структурної схеми, що ілюструють роботу системи, проаналізуємо функціональні можливості і базові задачі забезпечення основних компонент СЗІ.

Проаналізовано особливості побудови програмного інтерфейсу сучасних технічних засобів захисту, що дозволило розробити відповідний аналог при створенні експериментальних зразків програмних засобів на основі запропонованих алгоритмів.

Розроблена схема ЗІ включає в себе симетричну нейросистему, що реалізує нейромережевий алгоритм шифрування.

Система характеризується простотою використання внаслідок

КБПЗ_2025

автоматичності процесів налаштування параметрів НМ та швидкодією за рахунок використання блокової структури НКД.

Вхідною інформацією (блок "Джерело повідомлень") служать файли даних, ефективна обробка яких забезпечується шляхом оперування побітовим машинним кодом.

Внутрішні параметри програмних модулів (для алгоритму кодування, генерації ключа та ін.) можуть вводитись в інтерактивному режимі або встановлюватись з попередньо створених бібліотек.

Для експериментальних досліджень і подальшого гнучкого функціонування в програмних модулях передбачена можливість зміни параметрів, таких як рівень завад в КЗ та довжина ключа.

Наведено дані щодо застосування створених спеціалізованих програмних реалізацій для розв'язування практичних завдань.

Кожна компонента наведеної схеми містить елементи, що передбачають використання нейромережі та реалізують задачі підтримки інформаційних характеристик з наперед обумовленим ступенем захисту.

В залежності від вибраної задачі (забезпечення цілісності, конфіденційності) в процесі роботи використовується відповідний модуль схеми і певна конфігурація кожної складової.

Задача забезпечення конфіденційності інформації

Розроблений програмний продукт, що реалізує запропоновані в бакалаврській роботі алгоритми шифрування/дешифрування на основі навчання/функціонування НМ ФМТФ, містить процедури:

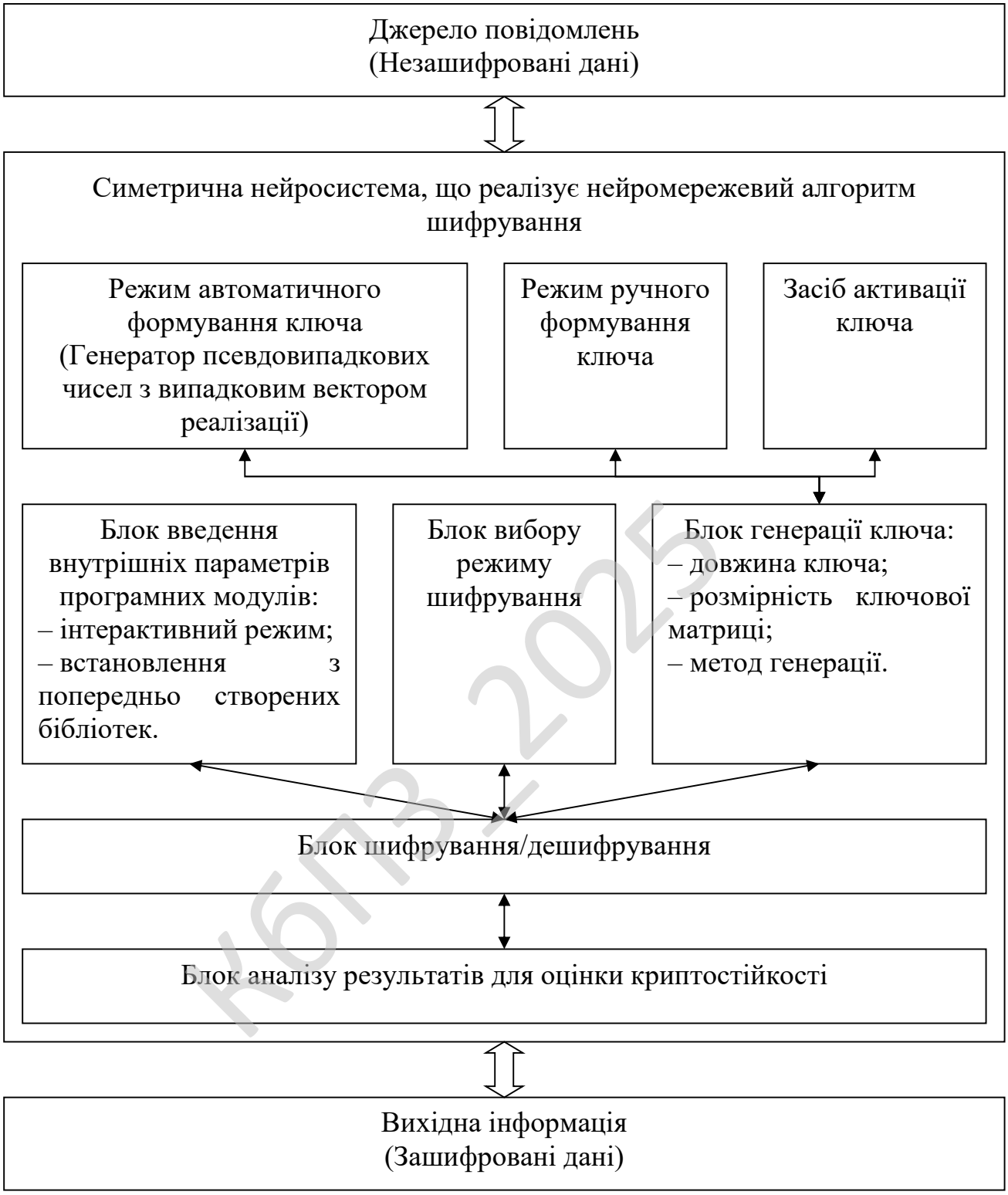
– генерації ключа;

– шифрування;

– дешифрування;

– аналізу результатів для оцінки криптостійкості.

КБПЗ_2025



КБПЗ – 2025

Рисунок 3.2 – Функціональна схема системи

Оскільки сучасні дослідження в галузі секретних систем виявили єдиний критерій оцінки стійкості алгоритму шифрування, що залежить від властивостей

ключової інформації, розглянемо процес генерування ключа в розробленій системі.

Процес генерування ключа може відбуватись як в бінарному, так і в дійсному режимах з врахуванням розмірності ключової матриці та методу генерації.

В режимі автоматичного формування ключа використовується генератор псевдовипадкових чисел з випадковим вектором реалізації. Враховуючи недоліки існуючих генераторів, в програмі введено режим ручного формування ключа, що дозволяє змінювати згенеровану послідовність чисел для підвищення надійності системи. Для встановлення створеного ключа в алгоритмах шифрування та дешифрування, що реалізовані в програмі, у вікні формування ключа представлено засіб його активації. Наявність додаткових ключів дозволяє підвищити рівень захищеності шляхом використання перестановок блоків шифртексту. Параметри, які використовуються при генерації, також обумовлюють і режим шифрування та розмірність ключової матриці. В залежності від їх значень, нейромережевий алгоритм шифрування можна трактувати як блоковий або потоковий, що дозволяє провести аналіз функціонування в порівнянні з ефективними блоковими або потоковими шифрами. Використання дійсного режиму в процесі генерації ключа обумовлюється представленням даних, якими оперує НМ. Внаслідок цього при програмній реалізації алгоритму виникає потреба в збільшенні обчислювальних ресурсів.

КБПЗ-2025

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

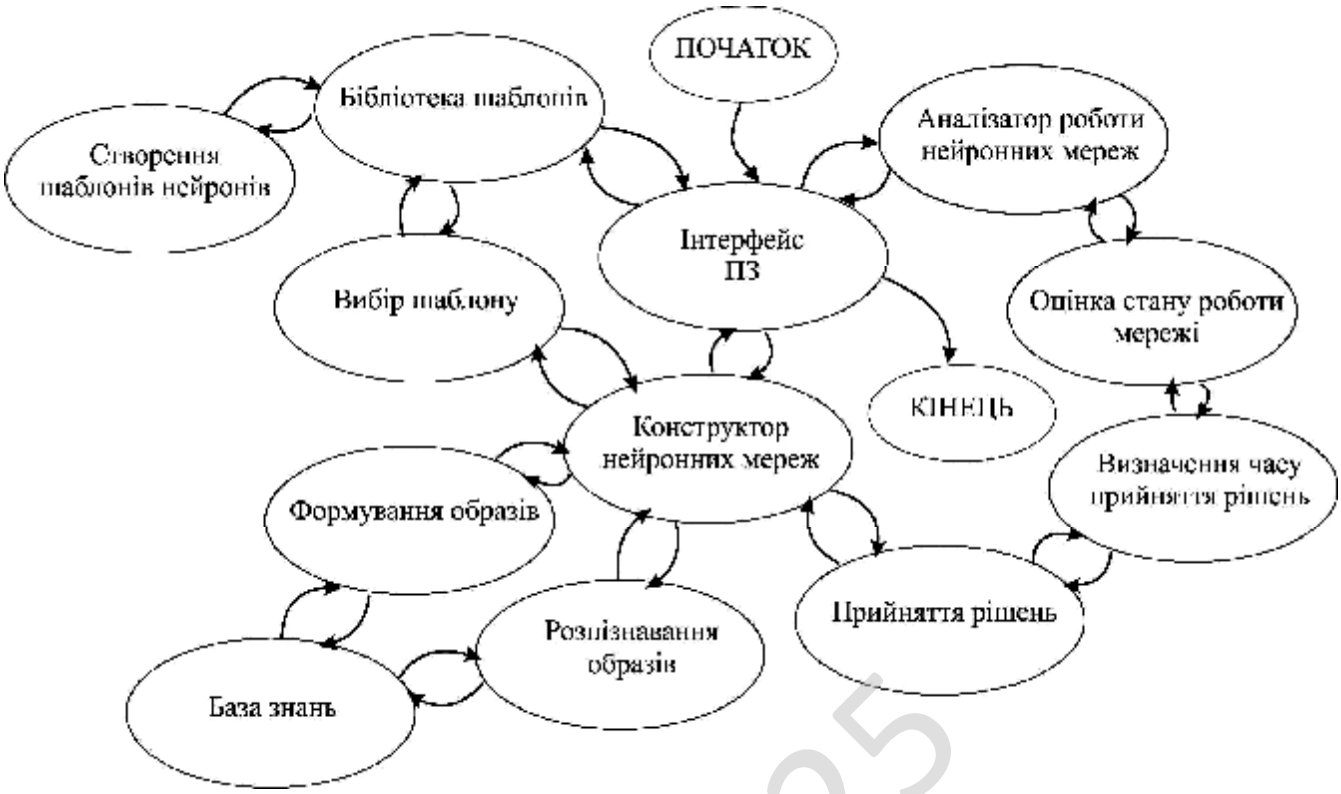


Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

КБГЗ – 2025

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту інформації на основі нейромережевих технологій.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

КБПЗ_2025

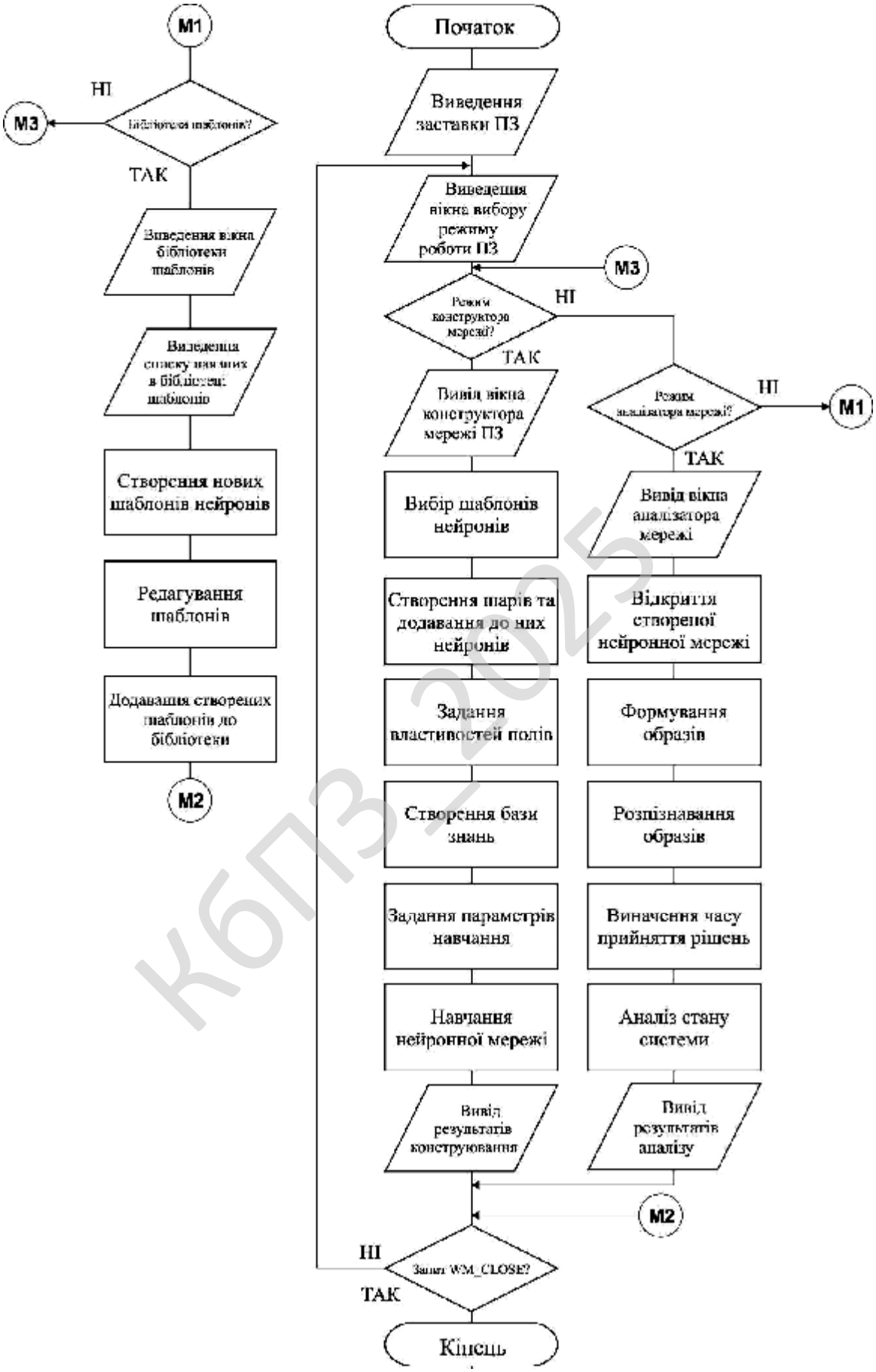


Рисунок 4.1 – Блок-схема основної програми

КБПЗ-2025

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

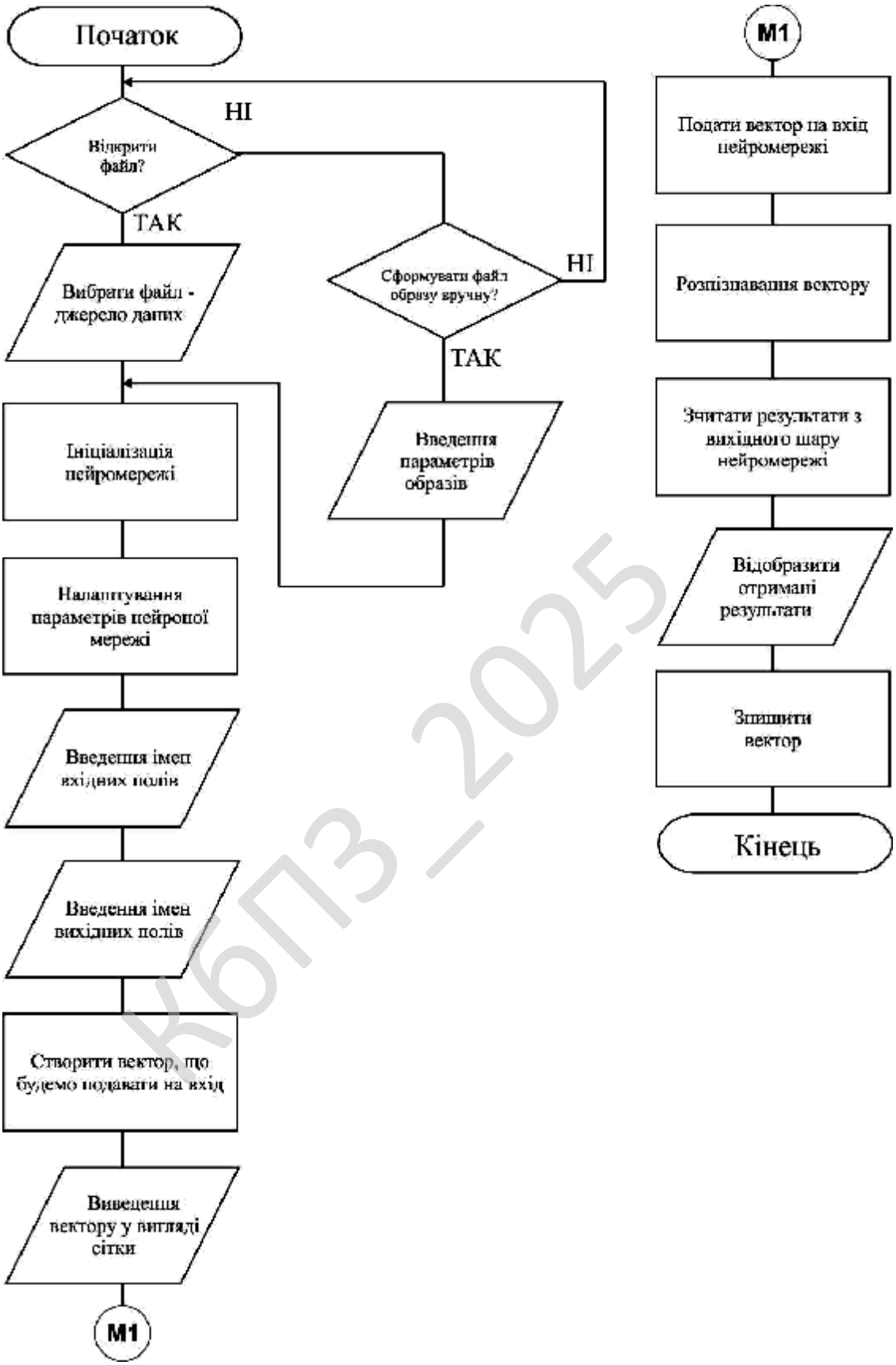
Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

КБПЗ_2025



КБПЗ_2025

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

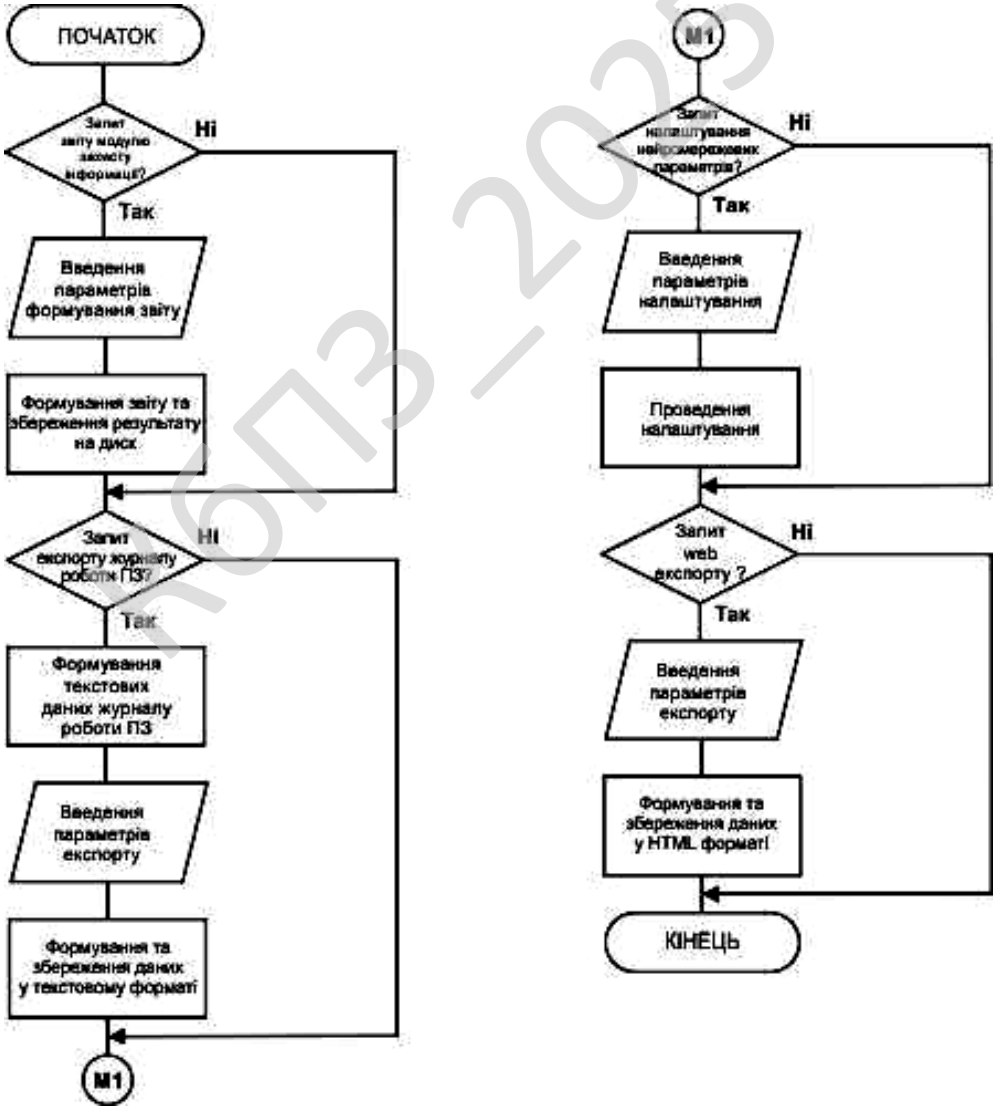


Рисунок 4.3 – Блок-схема роботи підпрограми

КбПЗ_2025

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо роль використання конструктора мережі та бібліотеки шаблонів для захисту інформації на основі нейромережевих технологій

Як уже було відзначено, конструктори мереж ядра призначені для створення внутрішнього подання мережі в пам'яті комп'ютера по різних джерелах. Тут буде розглянутий тільки один – конструктор по файлу-специфікації мережі, але ми не виключаємо можливості створення конструкторів, що використовують інші джерела.

Конструктор по суті є фабрикою об'єктів класу ЕлементМережі. Ідея фабрики полягає в наступному. Оскільки конструювання мережі складається в породженні величезного числа різнорідних об'єктів ЕлементМережі, то необхідний об'єкт для регулювання процесу породження й смерті цих об'єктів, або фабрика елементів мережі.

Тобто на Фабрику також покладені функції збирача сміття. Регулювання або управління процесом породження полягає в наступному. Ми маємо багато різних нащадків класу ЕлементаМережі, наприклад, Нейрон, що, у свою чергу має кілька підкласів, відповідних кожній з різновидів формальних моделей, а також інші елементи мережі Блок, Джерело, що має також кілька своїх підкласів і т.д.

Припустимо, ми модифікували або створили нову версію класу А з перерахованих класів – A'. Тоді в кожному місці вихідного тексту ми повинні замінити оператор породження А на оператор породження A'. Більш гнучкою є наступна схема. Фабрика1 уміє, або точніше виражаючись, має методи для породження об'єктів класів А, В, С і т.д. При повідомленні про породження, наприклад, об'єкта типу А, вона породжує насправді об'єкт нащадка А: A', а Фабрика2 породжує в цьому випадку A''. Таким чином, заміною тільки фабрик

КБПЗ_2025

ми можемо міняти класи породжуваних об'єктів. Посилання на ФабрикуЕлементівМережі, що вміє породжувати кожний з кінцевих нащадків ЕлементМережі, зберігає об'єкт Мережа.

При ініціалізації КонструктораМережі йому повідомляється посилання на Мережу. Природно, Мережа ще не містить ЕлементівМережі, але вже повинна мати посилання на ФабрикуЕлементівМережі. При конструюванні мережі за файлом специфікації КонструкторМережіПоФайлу (підклас КонструктораМережі) використовує методи породження об'єктів ФабрикиЕлементовМережі, посилання на яку він бере в Мережі.

Відзначимо тут, як була вирішена проблема передачі параметрів конструкторові (ініціалізатору, особливому методу, що викликається першим після розміщення об'єкта в пам'яті) елемента мережі й, взагалі, конструкторові будь-якого об'єкта, породження й видалення якого перебуває під управлінням фабрики. Проблема складається в уніфікації типу переданих параметрів: вони повинні бути ті самі для всіх типів елементів мережі. Був уведений клас Атрибут і методу породження об'єкта у фабрики й, відповідно, конструкторові об'єкта передавався список Атрибутів.

Кожний Атрибут має ім'я, і конструктор кожного елемента мережі розпізнає тільки деяку підмножину підкласів Атрибута, «свої атрибути», які довідається по ім'ю. Наприклад, АтрибутЙмовірності є підкласом Атрибута, має своє поле раціонального типу ймовірність. Конструктор Нейрона2 розпізнає АтрибутЇмовірності в переданому списку атрибутів і використовує значення його поля ймовірність для ініціалізації Нейрона2. Для атрибутів також знадобилася ФабрикаАтрибутов.

При створенні Фабрики був використаний ще один об'єктний шаблон, так званий Синглетон.

Синглетон вирішує задачу забезпечення одиничності екземпляра класу й управляє доступом до цього екземпляра.

КБПЗ_2025

Тепер про саму специфікацію. При створенні мови специфікації ставилися наступні задачі:

– Максимум широти спектра описуваних мереж, або максимальна гнучкість мови;

– Відносна простота;

– Мінімальна довжина специфікацій;

– Можливість розвитку мови;

– Специфікація мережі.

Специфікація складається із секцій. Секція починається із вказівки імені секції у квадратних дужках і складається з операторів специфікації. В Meta секції зібрані оператори, область застосування яких – вся специфікація, тобто всі секції. Тут, наприклад, можна задати імена-синоніми для шаблонів.

У секціях Inputs і Outputs просто перераховуються входи й виходи мережі. Найбільша секція, звичайно, Net Topology, де описується топологія мережі. Посилання на елемент мережі в специфікації складаються із двох злитих слів: вказівки ім'я типу елемента мережі (або його синоніма) і його деякого порядкового номера, причому нумерація для кожного типу своя. Опис топології складається з операторів опису топології, у яких вказується деякий елемент мережі, його входи у квадратних дужках, причому в круглих дужках після вказівки кожного входу може стояти асоціативний список (тобто список пар ім'я-значення) атрибутів зв'язку, і, додатково, після перерахування входів, може бути, у круглих дужках асоціативний список атрибутів елемента.

Кожний зв'язок двох елементів мережі може характеризуватися деякою множиною атрибутів зв'язку. Наприклад, атрибутом зв'язку може бути синаптична затримка. Кожний елемент мережі розуміє свій набір атрибутів зв'язку, деяка множина атрибутів зв'язку обробляється ядром системи, і, як ми вже згадали, кожний елемент мережі розуміє свій набір атрибутів (елемента мережі), значення яких можуть передаватися в списку атрибутів елемента.

КПІ-2025

Загалом кажучи, ядром підтримується деяка множина вбудованих базових шаблонів і в мові є конструкція для визначення нових шаблонів через уже певні або убудовані. Інформація про вже певні шаблони зберігається в спеціальному об'єкті БібліотекаШаблонів, що є Синглетоном і має методи для одержання ідентифікатора шаблона по його ім'ю, добування інформації про шаблон по його ідентифікаторі, а також додавання/видалення шаблонів. Якщо КонструкторМережі при читанні специфікації зустрічає ім'я шаблона або конструкцію визначення нового шаблона, то він звертається до БібліотеціШаблонів для, відповідно, одержання інформації про шаблон по ім'ю або для додавання нового шаблона.

Таким чином, користувач може створювати бібліотеки своїх шаблонів, визначення яких зберігаються, наприклад, у файлах і при їхньому використанні просто включати відповідні файли у свої програми.

Відзначимо, що бібліотечні файли шаблонів можуть створюватися спеціальним інструментом, конструктором шаблонів.

Як приклад, створені компоненти, що реалізують дві нейромережні парадигми: рекурентну нейронну мережу, у нашому випадку - це мережа Хопфілда й багатошарову нейронну мережу з навчанням по алгоритму зворотного поширення помилки (back propagation).

Реалізація нейронних мереж у вигляді компонентів дозволяє легко вбудовувати їх в інші програми.

Об'єктно-орієнтоване виконання надає особливу гнучкість, та можливість одержати компонент, оптимізований під будь-які задачі.

КБПЗ-2025

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 7564:2014 («Купина»). В Україні на основі консервативного підходу із залученням відомих і добре досліджених конструкцій була розроблена ґеш-функція, що базується на новому блоковому шифрі „Калина” (ДСТУ 7624:2014).

Національний стандарт ДСТУ 7564:2014 визначає криптографічну функцію ґешування „Купина”, додатковий режим її застосування для формування коду автентифікації повідомлення (імітовставки), а також значення для перевірки реалізацій.

Для скорочення обсягу тексту національного стандарту була застосована математична нотація, що дозволяє отримати точний і компактний запис.

Водночас, такий підхід може ускладнювати сприйняття сутності алгоритму для фахівців, що не мають фундаментальної криптологічної освіти.

У роботі наводиться розгорнутий альтернативний опис функції ґешування „Купина” із позначеннями, традиційними для галузі комп’ютерних наук.

Термінологія та позначення

Вектор ініціалізації – бітова послідовність фіксованої довжини (512 або 1024 біта), що використовується як початкове значення при обчисленні ґеш-значення.

Внутрішній стан – бітова послідовність фіксованої довжини (512 або 1024 біта), що є проміжним значенням на кожній ітерації перетворення функції ґешування, а також вхідним та вихідним значеннями перетворень P і Q; для цих перетворень внутрішній стан подається як матриця розміром $8\times c$ байт.

Ґеш-значення (ґеш-вектор) – бітова послідовність фіксованої довжини ($n = 8\cdot s$, $s\in\{1,2,...,64\}$), що є результатом роботи функції ґешування.

Доповнення – вставка додаткових біт у кінець повідомлення для отримання кратності довжини бітової послідовності довжині внутрішнього стану функції ґешування.

КБПЗ_2025

Повідомлення – бітова послідовність довжини від 0 біт (порожній рядок) до 2^{96}-1 біт.

Функція стиснення – ітеративне перетворення, що відображає *l*-бітний блок повідомлення та *l*-бітне значення, отримане функцією стиснення на попередньому кроці, у нове *l*-бітне значення.

Далі використовуються наступні позначення:

– ⊕ – додавання за модулем 2 (XOR);

– 0x – префікс числа, що записане у шістнадцятковій системі числення;

– $a \bmod b$ – ціле невід'ємне число, що дорівнює залишку від ділення цілого числа *a* на натуральне число *b*;

– B_i – *i* -й байт вхідної послідовності;

– C^i – константа перетворення XORRoundKey або Add64RoundKey для *i* -го циклу;

– *c* – кількість стовпців внутрішнього стану в матричному поданні;

– φ – функція стиснення;

– *H* – визначена у стандарті функція ґешування;

– *H*(*M*) – результат обчислення функції ґешування для повідомлення *M* (ґешзначення);

– IV – вектор ініціалізації;

– *l* – розмір внутрішнього стану функції ґешування (у бітах), $l \in \{512,1024\}$;

– *M* – повідомлення;

– m_i – *i* -й блок повідомлення *M*;

– *n* – довжина обчисленого ґеш-значення;

– *N* – довжина повідомлення *M* без доповнення;

– *P*, *Q* – складові перетворення функції стиснення;

– P_{512} – перетворення *P* для 512-бітного внутрішнього стану;

– P_{1024} – перетворення *P* для 1024-бітного внутрішнього стану;

– Q_{512} – перетворення *Q* для 512-бітного внутрішнього стану;

КБПЗ 2025

– Q_{1024} – перетворення Q для 1024-бітного внутрішнього стану;

– r – кількість ітерацій у перетвореннях P і Q ($r = t$ в ДСТУ 7564:2014);

– S – внутрішній стан ґеш-функції;

– t – кількість блоків m, з яких складається повідомлення M, включаючи доповнення;

– v_i – i -й біт вхідної послідовності;

– Ω – завершальне перетворення;

– Купина-n – режим використання функції ґешування з усіченням обчисленого ґешзначення до розміру n біт.

Загальні положення

Під функцією ґешування H розуміється залежне від вектора ініціалізації IV відображення послідовності біт M у ґеш-значення $H(M)$ фіксованої довжини n.

ДСТУ 7564:2014 визначає функцію ґешування, яка виконує перетворення «Купина-256» або «Купина-512», що забезпечують обчислення ґеш-значення з довжинами 256 або 512 біт відповідно.

Ґеш-значення довжиною 256 бітів додатково може бути усічено до бітової послідовності довжиною від 8 до 248 біт з кроком у 8 біт, 512 бітів може бути усічене до бітової послідовності довжиною від 264 до 504 біт з кроком у 8 біт.

Режим роботи для формування ґеш-значення довжиною n біт позначається як «Купина-n».

Основними режимами роботи функції ґешування, що рекомендуються до застосування, є «Купина-256», «Купина-384» і «Купина-512».

Структура перетворення

Функція ґешування, визначена в ДСТУ 7564:2014, формує ґеш-значення для повідомлення, що складається з бітової послідовності довжини від 0 біт (порожній рядок) до 2^{96}-1 біт.

КБПЗ_2025

При формуванні ґеш-значення повідомлення доповнюється, далі поділяється на l-бітні блоки $m_0,\ldots, m_t$, після чого виконується обробка кожного блоку шляхом ітеративного виконання функції стиснення φ.

При цьому формуються значення $h_i = \varphi(h_{i-1}, m_i)$ де $i = 1,\ldots,t$, а початкове значення h_0 = IV.

Після обробки останнього блоку повідомлення результуюче ґеш-значення обчислюється як $H(M) = \Omega(h_t)$, де Ω – завершальне перетворення, що повертає n - бітне значення, кратне 8 ($0 < n \leq l/2$).

К6ПЗ_2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської роботи.

Розроблене програмне забезпечення захисту інформації на основі нейромережевих технологій складається з наступних функціональних блоків:

– Навігаційне меню: Бібліотека шаблонів; Конструктор НМ; Налаштування; Довідка.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ: Аналізатор роботи НМ; База знань; Навчання нейронної мережі; Редактор властивостей; Налаштування умов зупинки навчання.

КПЗ_2025

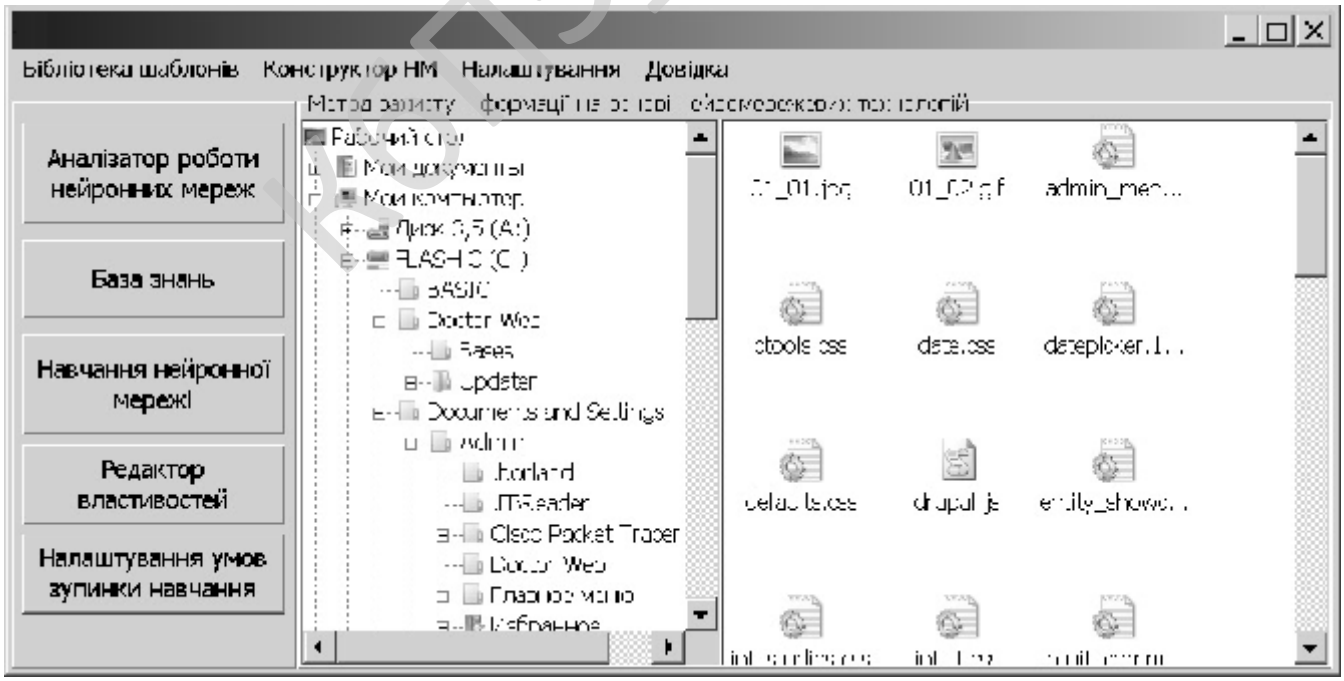


Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

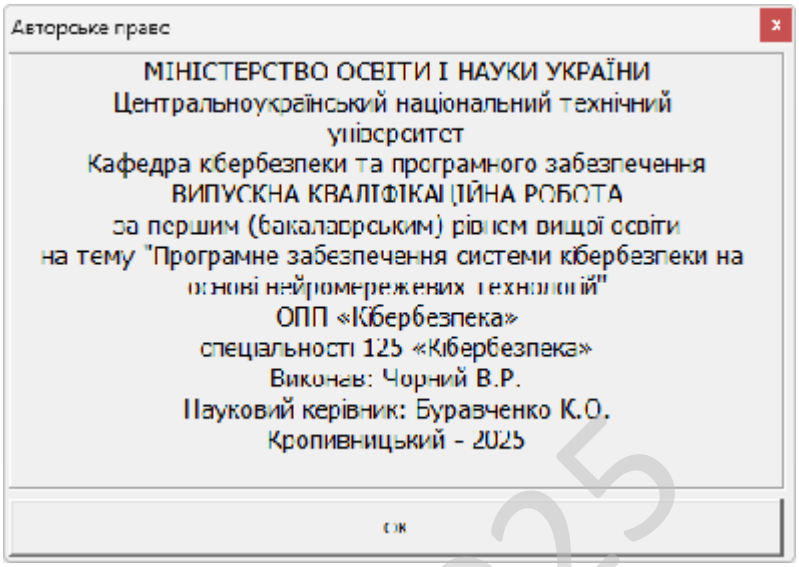


Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

КБПЗ-2025

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки» пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

КБПЗ_2025

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ - «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

КБПЗ_2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки на основі нейромережевих технологій.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем на основі нейромережевих технологій.

– Досліджена система на основі нейромережевих технологій.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки на основі нейромережевих технологій.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання на основі нейромережевих технологій.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки на основі нейромережевих технологій. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної

КБПЗ_2025

техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 7564:2014.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

КБПЗ_2025

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.

2. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.

3. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.

4. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O'Reilly. 2019. – 252 p.

5. Rajasekaran S., Vijayalakshmi Pai G.A. Neural networks, fuzzy logic, and genetic algorithms: synthesis and applications (with cd-rom) Kindle Edition. – PHI, 2013. – 628 p.

6. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

7. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

8. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

9. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

КБПЗ-2025

10. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

12. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022, pp. 1-12.

14. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

15. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

16. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

17. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

КБПЗ_2025

18. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207.

19. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

20. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

21. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

22. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

23. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

24. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

25. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties».

КБПЗ-2025

International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

26. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

27. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

28. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

30. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

31. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

32. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

КБПЗ-2025

Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

33. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

34. Smirnov, S., Bulekbaeva, G., Kikvidze, O.G., Lakhno, V., Brzhanov, R., Tabylov, A. «Computer simulation in the MathCAD package of plastic deformation of the deposited layer on the flat surface of the part». Journal of Theoretical and Applied Information Technology Volume 97, Issue 20, 2019, Pages 2467-2484. (Scopus).

35. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

36. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

38. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв’язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

39. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

КБПЗ_2025

комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

40. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

41. Смірнов, С.А., Смірнова, Т.В., Минайленко, Р.М., Доренський, О.П., Сисоєнко С.В. «Хмарна автоматизована система інтелектуальної підтримки прийняття рішень для технологічних процесів». Вісник Черкаського державного технологічного університету. Технічні науки. №4, 2020, С. 84-92.

42. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

43. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

44. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура № 1 (11). с. 48-57, 2019.

45. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

46. Смірнова Т.В., Дрєєв О.М., Смірнов О.А. Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал»

КБПЗ-2025

електродуговим напиленням. Системи управління, навігації та зв’язку, № 2 (54). С. 149-154, 2019.

47. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

48. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

49. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). - Полтава: ПолтНТУ. - 2016. - С. 98-103

50. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). - Харків: ХУПС. - 2016. - С.96-100.

51. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". - Випуск 8 (145). - Х.: ХУПС - 2016. - С. 77-80.

52. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". - Випуск 6 (143). - Х.: ХУПС - 2016. - С. 216-220.

53. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). - Харків: ХУПС. - 2016. - С. 128-133.

КБПЗ-2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування....2
2 Підстава для розробки....2
3 Мета та призначення розробки....2
4 Джерела розробки....2
5 Технічні вимоги....2
5.1 Вміст проекту....2
5.2 Показники призначення....3
5.3 Вимоги до функціональних характеристик....3
5.4 Вимоги до архітектури....3
5.5 Вимоги до надійності....3
5.6 Умови експлуатації....4
5.7 Вимоги до складу та параметрів технічних засобів....4
5.8 Вимоги до інформаційної і програмної сумісності....4
5.8.1 Обладнання....4
5.8.2 Мова програмування....4
5.8.3 Вхідні дані....5
5.8.4 Вихідні дані....5
6 Вимоги до програмної документації....5
7 Перелік документів, що розробляються....5
8 Етапи розробки....6
9 Порядок контролю та приймання....6

КБПЗ-2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки на основі нейромережевих технологій.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки на основі нейромережевих технологій.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

КБПЗ_2025

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки на основі нейромережевих технологій;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КБПЗ_2025

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

КБПЗ_2025

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 55 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

КБПЗ_2025

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 4.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Буравченко К.О.

КБПЗ_2025

Програмне забезпечення системи кібербезпеки на основі нейромережевих технологій

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
#Імпорт необхідних бібліотек для роботи системи
import numpy as np
#Імпорт бібліотеки для обробки даних
import pandas as pd
#Імпорт TensorFlow для побудови нейронних мереж
import tensorflow as tf

#Імпорт класів з TensorFlow Keras для створення моделей
from tensorflow.keras.models import Sequential
#Імпорт шарів для побудови моделі
from tensorflow.keras.layers import Dense, Dropout

#Імпорт оптимізатора Adam для навчання моделі
from tensorflow.keras.optimizers import Adam
#Імпорт бібліотеки для генерації випадкових чисел
import random
#Імпорт бібліотеки для роботи з часом
import time

#Імпорт бібліотеки для роботи з потоками
import threading
#Імпорт бібліотеки для виконання HTTP запитів
import requests

#Імпорт бібліотеки для логування подій системи
import logging
#Імпорт Flask для створення веб-сервера API
from flask import Flask, request, jsonify

#Налаштування базового рівня логування
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s - %(levelname)s - %(message)s')

#Клас для реалізації нейронної мережі для виявлення вторгнень
class NeuralNetworkIntrusionDetector:

#Конструктор класу для ініціалізації параметрів моделі
    def __init__(self, input_dim, model_path='intrusion_model.h5'):

#Встановлення розміру вхідного шару
        self.input_dim = input_dim

#Встановлення шляху збереження моделі
        self.model_path = model_path

#Створення нейронної мережі за допомогою методу build_model
        self.model = self.build_model()

#Прапорець, що показує стан навчання моделі
        self.trained = False

#Метод для побудови архітектури моделі нейронної мережі
    def build_model(self):

#Створення послідовної моделі
        model = Sequential()

#Додавання вхідного шару з 64 нейронами та функцією активації relu
        model.add(Dense(64, input_dim=self.input_dim, activation='relu'))
```

КБПЗ_2025

```
#Додавання шару Dropout для зменшення перенавчання
        model.add(Dropout(0.3))
#Додавання прихованого шару з 32 нейронами та relu активацією
        model.add(Dense(32, activation='relu'))

#Додавання ще одного шару Dropout для кращої регуляризації
        model.add(Dropout(0.3))
#Додавання вихідного шару з 1 нейроном та сигмоїдальною активацією для виявлення
аномалій
        model.add(Dense(1, activation='sigmoid'))

#Компіляція моделі з використанням функції втрат binary_crossentropy та
оптимізатора Adam
        model.compile(loss='binary_crossentropy',
optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
#Повернення побудованої моделі
        return model

#Метод для навчання моделі на тренувальних даних
    def train_model(self, X_train, y_train, epochs=50, batch_size=32):

#Логування початку процесу навчання моделі
        logging.info("Starting model training.")
#Навчання моделі з використанням наданих даних
        history = self.model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, verbose=0)

#Оновлення прапорця про завершення навчання
        self.trained = True
#Збереження навченої моделі на диску
        self.save_model()
#Логування завершення навчання
        logging.info("Model training completed.")
#Повернення історії навчання для подальшого аналізу
        return history

#Метод для виконання прогнозування аномалій
    def predict(self, X):
#Перевірка, чи модель була навчена або завантажена
        if not self.trained:

#Логування попередження про спробу прогнозування без навчання
            logging.warning("Model not trained. Prediction may be inaccurate.")

#Виконання прогнозування за допомогою моделі нейронної мережі
        predictions = self.model.predict(X)

#Повернення результатів прогнозування у вигляді масиву
        return predictions

#Метод для збереження моделі на диску
    def save_model(self):
#Збереження архітектури та ваг моделі
        self.model.save(self.model_path)
#Логування успішного збереження моделі
        logging.info("Model saved to disk at %s", self.model_path)

#Метод для завантаження моделі з диску
    def load_model(self):
```

КбПЗ - 2025

```
#Завантаження моделі за допомогою TensorFlow
        self.model = tf.keras.models.load_model(self.model_path)

#Оновлення прапорця про доступність моделі
        self.trained = True

#Логування успішного завантаження моделі
        logging.info("Model loaded from disk at %s", self.model_path)

#Додатковий метод для попередньої обробки даних (резервний)
    def preprocess_data(self, data):

#Логування початку процесу попередньої обробки даних
        logging.debug("Starting data preprocessing.")

#Перетворення даних у формат numpy array
        processed_data = np.array(data)
#Логування завершення попередньої обробки
        logging.debug("Data preprocessing completed.")
#Повернення оброблених даних
        return processed_data

#Додатковий метод для детального логування роботи моделі
    def detailed_logging(self):
#Логування внутрішнього стану моделі для налагодження
        logging.debug("Model details: %s", self.model.summary())
#Завершення методу додаткового логування
        return

#Клас для аналізу системних логів
class LogAnalyzer:
#Конструктор класу з вказанням файлу логів
    def __init__(self, log_file='system_logs.txt'):
#Встановлення шляху до файлу логів
        self.log_file = log_file

#Метод для аналізу лог-файлу на предмет підозрілих записів
    def analyze_logs(self):

#Ініціалізація списку для збереження підозрілих записів
        suspicious_entries = []
#Спроба відкрити файл логів для читання
        try:

#Відкриття файлу логів у режимі читання
            with open(self.log_file, 'r') as file:
#Читання всіх рядків з файлу
                lines = file.readlines()

#Ітерація по кожному рядку файлу
                for line in lines:
#Перевірка наявності ключових слів, що свідчать про аномальні події
                    if "error" in line.lower() or "failed" in line.lower() or
"unauthorized" in line.lower():
#Додавання підозрілого рядка до списку
                        suspicious_entries.append(line.strip())
        except Exception as e:
#Логування помилки у разі неможливості читання файлу логів
            logging.error("Error reading log file: %s", e)

#Повернення списку підозрілих записів
        return suspicious_entries
```

КБПЗ_2025

```
#Додатковий метод для очищення логів (резервний)
    def cleanup_logs(self):
#Логування початку процесу очищення логів
        logging.debug("Starting log cleanup process.")
#Симуляція процесу очищення логів (без реальної реалізації)
        time.sleep(0.1)
#Логування завершення процесу очищення логів
        logging.debug("Log cleanup process completed.")
#Завершення методу cleanup_logs
        return

#Клас для взаємодії з зовнішніми джерелами розвідки загроз
class ThreatIntelligenceClient:

#Конструктор класу з налаштуванням API кінцевої точки
    def __init__(self, api_endpoint='https://example-threat-intel.com/api'):
#Встановлення кінцевої точки для API
        self.api_endpoint = api_endpoint

#Метод для запиту даних розвідки за індикатором (наприклад, IP адресою)
    def query_threat_intelligence(self, indicator):

#Формування URL запиту для отримання інформації про загрозу
        url = f"{self.api_endpoint}/lookup?indicator={indicator}"

#Логування формування URL для налагодження
        logging.debug("Querying threat intelligence for indicator: %s",
indicator)
#Ініціалізація змінної для збереження відповіді
        response = None
#Спроба виконати HTTP GET запит до API
        try:

#Виконання GET запиту з таймаутом 5 секунд
            response = requests.get(url, timeout=5)
#Перевірка статусу відповіді від API
            if response.status_code == 200:
#Повернення даних у форматі JSON при успішному запиті
                return response.json()
            else:
#Логування попередження, якщо статус відповіді не успішний
                logging.warning("Threat intelligence API returned status: %s",
response.status_code)

#Повернення порожнього словника у разі невдачі
                return {}
        except Exception as e:

#Логування помилки при виконанні запиту до API
            logging.error("Error querying threat intelligence: %s", e)
#Повернення порожнього словника при виникненні помилки
            return {}

#Додатковий метод для отримання розширеної інформації з API (резервний)
    def extended_api_details(self):
#Логування початку запиту розширених даних API
        logging.debug("Requesting extended API details.")
#Симуляція затримки для отримання розширених даних
        time.sleep(0.2)
#Логування завершення запиту розширених даних API
        logging.debug("Extended API details received.")
```

КОПЗ_2025

```
#Повернення симульованих даних
        return {"detail": "Simulated extended threat intelligence data"}

#Клас для симуляції мережевого трафіку з метою тестування системи
class NetworkTrafficSimulator:
#Конструктор класу для ініціалізації списків IP адрес та ознак
    def __init__(self):
#Список нормальних IP адрес для симуляції
        self.normal_ips = ["192.168.1." + str(i) for i in range(2, 255)]
#Список аномальних IP адрес для симуляції
        self.anomalous_ips = ["10.0.0." + str(i) for i in range(2, 50)]

#Список імен ознак, що використовуються для симуляції трафіку
        self.features = ['feature1', 'feature2', 'feature3', 'feature4',
'feature5']
#Додаткова ініціалізація для симуляції додаткових параметрів
        self.extra_params = {"latency": 0.05, "packet_loss": 0.01}

#Метод для симуляції одного зразка мережевого трафіку
    def simulate_sample(self):

#Випадкове визначення, чи є зразок аномальним
        is_anomalous = random.choice([True, False, False, False])
#Вибір IP адреси в залежності від типу трафіку
        ip_address = random.choice(self.anomalous_ips) if is_anomalous else
random.choice(self.normal_ips)

#Генерація випадкових значень для ознак трафіку
        sample_features = [random.random() for _ in self.features]
#Повернення словника з даними зразка
        return {"ip": ip_address, "features": sample_features, "anomaly": 1 if
is_anomalous else 0}

#Метод для симуляції пакету мережевого трафіку
    def simulate_traffic(self, num_samples=100):

#Ініціалізація списку для зберігання зразків трафіку
        samples = []
#Цикл для генерації заданої кількості зразків
        for _ in range(num_samples):

#Симуляція одного зразка трафіку
            sample = self.simulate_sample()
#Додавання зразка до списку
            samples.append(sample)
#Повернення списку з симульованими зразками трафіку
        return samples

#Додатковий метод для симуляції затримок мережевого трафіку
    def simulate_delay(self):

#Логування початку симуляції затримки
        logging.debug("Simulating network delay.")
#Симуляція затримки
        time.sleep(self.extra_params["latency"])
#Логування завершення симуляції затримки
        logging.debug("Network delay simulation completed.")

#Завершення методу simulate_delay
        return
```

КБПЗ_2025

```
#Основний клас системи кібербезпеки, що інтегрує всі компоненти
class CyberSecuritySystem:
#Конструктор класу для ініціалізації системних компонентів
    def __init__(self):

#Встановлення розміру вхідного шару згідно кількості ознак
        self.input_dim = 5

#Ініціалізація детектора вторгнень на основі нейронної мережі
        self.intrusion_detector =
NeuralNetworkIntrusionDetector(input_dim=self.input_dim)

#Ініціалізація аналізатора логів
        self.log_analyzer = LogAnalyzer()

#Ініціалізація клієнта для розвідки загроз
        self.threat_client = ThreatIntelligenceClient()

#Ініціалізація симулятора мережевого трафіку
        self.traffic_simulator = NetworkTrafficSimulator()

#Список для зберігання згенерованих сповіщень про загрози
        self.alerts = []

#Створення екземпляру Flask для API інтерфейсу
        self.app = Flask(__name__)

#Налаштування маршрутів API
        self.setup_routes()

#Додаткова ініціалізація додаткових параметрів системи
        self.system_parameters = {"monitor_interval": 1, "log_interval": 5}

#Метод для налаштування API маршрутів у Flask
    def setup_routes(self):

#Визначення маршруту для отримання списку сповіщень
        @self.app.route('/alerts', methods=['GET'])
        def get_alerts():
#Повернення сповіщень у форматі JSON
            return jsonify({"alerts": self.alerts})
#Визначення маршруту для подачі нового зразка мережевого трафіку
        @self.app.route('/submit', methods=['POST'])
        def submit_sample():

#Отримання даних у форматі JSON з POST запиту
            data = request.get_json()

#Перевірка наявності необхідних ключів у отриманих даних
            if 'features' not in data or 'ip' not in data:

#Повернення повідомлення про помилковий формат даних
                return jsonify({"error": "Invalid data format"}), 400

#Виконання прогнозування аномалії для отриманих ознак
            prediction =
self.intrusion_detector.predict(np.array([data['features']]))
#Перевірка, чи перевищує результат поріг аномалії
            if prediction[0][0] > 0.5:
```

КБПЗ_2025

```
#Формування повідомлення про виявлену аномалію
                alert_message = f"Anomalous activity detected from IP:
{data['ip']}"
#Додавання сповіщення до списку сповіщень
                self.alerts.append(alert_message)
#Повернення відповіді з інформацією про сповіщення
                return jsonify({"alert": alert_message}), 200
            else:
#Повернення відповіді, що трафік нормальний
                return jsonify({"message": "Traffic is normal"}), 200

#Метод для навчання детектора вторгнень з використанням симульованого трафіку
    def train_intrusion_detector(self):
#Логування початку процесу навчання моделі
        logging.info("Training the intrusion detection model with simulated
traffic.")
#Отримання симульованих зразків мережевого трафіку для навчання
        samples = self.traffic_simulator.simulate_traffic(num_samples=1000)
#Ініціалізація списків для ознак та міток
        X_train = []
        y_train = []
#Обхід кожного зразка для формування навчальних даних
        for sample in samples:
#Додавання ознак зразка до списку ознак
            X_train.append(sample['features'])
#Додавання мітки аномалії до списку міток
            y_train.append(sample['anomaly'])
#Перетворення списків у numpy масиви
        X_train = np.array(X_train)
        y_train = np.array(y_train)
#Навчання моделі детектора вторгнень за допомогою тренувальних даних
        self.intrusion_detector.train_model(X_train, y_train, epochs=20,
batch_size=16)
#Логування завершення процесу навчання моделі
        logging.info("Intrusion detection model training completed.")

#Метод для моніторингу мережевого трафіку у реальному часі
    def monitor_network(self):
#Логування початку моніторингу мережевого трафіку
        logging.info("Starting network traffic monitoring.")
#Безкінечний цикл для постійного моніторингу
        while True:
#Отримання симульованого зразка мережевого трафіку
            sample = self.traffic_simulator.simulate_sample()
#Отримання ознак зразка
            features = np.array([sample['features']])
#Прогнозування аномалії за допомогою моделі
            prediction = self.intrusion_detector.predict(features)
#Перевірка, чи перевищує прогнозований результат поріг аномалії
            if prediction[0][0] > 0.5:
#Формування сповіщення про виявлення аномалії
                alert_message = f"Detected anomaly from IP: {sample['ip']} with
score {prediction[0][0]:.2f}"
#Додавання сповіщення до системного списку
                self.alerts.append(alert_message)
#Логування створеного сповіщення
                logging.info("Alert generated: %s", alert_message)
#Виклик додаткової симуляції затримки трафіку
            self.traffic_simulator.simulate_delay()
#Затримка між обробкою зразків для зменшення навантаження
            time.sleep(self.system_parameters["monitor_interval"])

#Метод для моніторингу системних логів
```

КБПЗ_2025

```
    def monitor_logs(self):
#Логування початку моніторингу логів
        logging.info("Starting system log monitoring.")
#Безкінечний цикл для постійного аналізу логів
        while True:
#Аналіз логів за допомогою класу LogAnalyzer
            suspicious_entries = self.log_analyzer.analyze_logs()
#Якщо знайдено підозрілі записи, обробити кожен з них
            if suspicious_entries:
                for entry in suspicious_entries:
#Формування сповіщення на основі підозрілого запису
                    alert_message = f"Suspicious log entry detected: {entry}"
#Додавання сповіщення до загального списку
                    self.alerts.append(alert_message)
#Логування сповіщення про підозрілий запис
                    logging.info("Log alert: %s", alert_message)
#Затримка перед наступною перевіркою логів
            time.sleep(self.system_parameters["log_interval"])

#Метод для запуску API сервера Flask у окремому потоці
    def start_api_server(self):
#Логування запуску API сервера
        logging.info("Starting Flask API server for system interface.")
#Запуск Flask додатку на вказаних параметрах
        self.app.run(host='0.0.0.0', port=5000, debug=False, use_reloader=False)

#Метод для періодичного запиту розвідки загроз
    def threat_intelligence_monitor(self):
#Логування початку моніторингу розвідки загроз
        logging.info("Starting threat intelligence monitoring.")
#Список зразкових IP адрес для запиту розвідки
        sample_ips = self.traffic_simulator.normal_ips[:10] +
self.traffic_simulator.anomalous_ips[:5]
#Безкінечний цикл для запиту інформації
        while True:
#Обхід кожного IP для запиту розширених даних
            for ip in sample_ips:
#Запит даних розвідки за допомогою ThreatIntelligenceClient
                intel_data = self.threat_client.query_threat_intelligence(ip)
#Якщо отримано дані, що свідчать про загрозу, створити сповіщення
                if intel_data.get("threat", False):
                    alert_message = f"Threat intelligence reported threat for
IP: {ip}"
                    self.alerts.append(alert_message)
                    logging.info("Threat alert: %s", alert_message)
#Невелика затримка між запитами для уникнення перевантаження API
                time.sleep(0.5)
#Затримка перед наступною ітерацією запиту розвідки
            time.sleep(10)

#Метод для запуску всіх компонентів системи кібербезпеки
    def run(self):
#Навчання детектора вторгнень з використанням симульованого трафіку
        self.train_intrusion_detector()
#Створення потоку для моніторингу мережевого трафіку
        network_thread = threading.Thread(target=self.monitor_network,
daemon=True)
#Створення потоку для моніторингу системних логів
        log_thread = threading.Thread(target=self.monitor_logs, daemon=True)
#Створення потоку для моніторингу розвідки загроз
        threat_thread =
threading.Thread(target=self.threat_intelligence_monitor, daemon=True)
#Створення потоку для запуску API сервера
```

КБПЗ_2025

```
        api_thread = threading.Thread(target=self.start_api_server, daemon=True)
#Запуск потоку моніторингу мережевого трафіку
        network_thread.start()
#Запуск потоку моніторингу логів
        log_thread.start()
#Запуск потоку моніторингу розвідки загроз
        threat_thread.start()
#Запуск потоку API сервера
        api_thread.start()
#Логування запуску всіх компонентів системи
        logging.info("Cybersecurity system components are running.")
#Безкінечний цикл для підтримки роботи головного потоку
        while True:
#Додаткова затримка для зменшення навантаження на процесор
            time.sleep(1)
#Завершення методу run

#Головна точка входу в програму
if __name__ == '__main__':
#Логування ініціалізації системи кібербезпеки
    logging.info("Initializing the Cybersecurity System.")
#Створення екземпляру системи кібербезпеки
    cyber_system = CyberSecuritySystem()
#Логування завершення ініціалізації системи
    logging.info("Cybersecurity System initialization complete.")
#Запуск системи кібербезпеки
    cyber_system.run()
#Кінець вихідного коду для системи кібербезпеки на основі нейромережевих
технологій
```

КБПЗ_2025

Файл RealTimeDashboard.py

```
import threading
import time
import random
import json
import socket
import select
import sys
import os
from flask import Flask, jsonify, request, render_template_string

class RealTimeDashboard:
    def __init__(self, host='0.0.0.0', port=8000):
        self.app = Flask(__name__)
        self.host = host
        self.port = port
        self.data_lock = threading.Lock()
        self.metrics = {"cpu": [], "memory": [], "network": []}
        self.setup_routes()
        self.running = False

    def setup_routes(self):
        @self.app.route('/')
        def dashboard():
            html = self.get_dashboard_html()
            return html

        @self.app.route('/data')
        def data():
            with self.data_lock:
                return jsonify(self.metrics)

    def get_dashboard_html(self):
        html_template = """
        <html>
        <head>
        <title>Real Time Dashboard</title>
        <script>
        function fetchData() {
        fetch('/data')
        .then(response => response.json())
        .then(data => {
        document.getElementById('cpu').innerHTML = data.cpu.join(', ');
        document.getElementById('memory').innerHTML = data.memory.join(', ');
        document.getElementById('network').innerHTML = data.network.join(', ');
        });
        }
        setInterval(fetchData, 1000);
        </script>
        </head>
        <body>
        <h1>Real Time Metrics</h1>
        <div>
        <h2>CPU Usage</h2>
        <p id="cpu"></p>
        <h2>Memory Usage</h2>
        <p id="memory"></p>
        <h2>Network Traffic</h2>
        <p id="network"></p>
        </div>
        </body>
        </html>
```

КБПЗ_2025

```
        """
        return render_template_string(html_template)

    def update_metrics(self):
        while self.running:
            with self.data_lock:
                self.metrics["cpu"].append(random.randint(0, 100))
                if len(self.metrics["cpu"]) > 20:
                    self.metrics["cpu"].pop(0)
                self.metrics["memory"].append(random.randint(0, 100))
                if len(self.metrics["memory"]) > 20:
                    self.metrics["memory"].pop(0)
                self.metrics["network"].append(random.randint(0, 1000))
                if len(self.metrics["network"]) > 20:
                    self.metrics["network"].pop(0)
            time.sleep(1)

    def start(self):
        self.running = True
        t = threading.Thread(target=self.update_metrics)
        t.daemon = True
        t.start()
        self.app.run(host=self.host, port=self.port, debug=False,
use_reloader=False)

class IncidentResponseSystem:
    def __init__(self):
        self.blocked_ips = set()
        self.incident_queue = []
        self.response_lock = threading.Lock()

    def add_incident(self, incident):
        with self.response_lock:
            self.incident_queue.append(incident)

    def process_incidents(self):
        while True:
            with self.response_lock:
                if self.incident_queue:
                    incident = self.incident_queue.pop(0)
                    self.respond_to_incident(incident)
            time.sleep(1)

    def respond_to_incident(self, incident):
        if "IP:" in incident:
            ip = incident.split("IP:")[1].split()[0]
            if ip not in self.blocked_ips:
                self.block_ip(ip)

    def block_ip(self, ip):
        self.blocked_ips.add(ip)
        self.log_block(ip)

    def log_block(self, ip):
        with open("blocked_ips.txt", "a") as f:
            f.write(ip + "\n")

    def start(self):
        t = threading.Thread(target=self.process_incidents)
        t.daemon = True
        t.start()

class MLModelManagement:
```

КБТЗ_2025

```
    def __init__(self):
        self.models = {}
        self.current_version = 0
        self.training_lock = threading.Lock()

    def add_model(self, model_name, model):
        self.current_version += 1
        version_key = f"{model_name}_v{self.current_version}"
        self.models[version_key] = model

    def train_model(self, model_name, data, labels, epochs=10):
        dummy_model = {"name": model_name, "version": self.current_version + 1,
"accuracy": 0}
        for epoch in range(epochs):
            time.sleep(0.1)
            dummy_model["accuracy"] += random.random() * 0.1
        with self.training_lock:
            self.current_version += 1
            version_key = f"{model_name}_v{self.current_version}"
            dummy_model["version"] = self.current_version
            self.models[version_key] = dummy_model
        return dummy_model

    def evaluate_model(self, model_key, test_data, test_labels):
        if model_key in self.models:
            model = self.models[model_key]
            score = random.random()
            return score
        return None

    def list_models(self):
        return list(self.models.keys())

    def auto_retrain(self, model_name, data, labels):
        while True:
            self.train_model(model_name, data, labels, epochs=5)
            time.sleep(10)

    def start_auto_retrain(self, model_name, data, labels):
        t = threading.Thread(target=self.auto_retrain, args=(model_name, data,
labels))
        t.daemon = True
        t.start()

class ForensicAnalysisModule:
    def __init__(self):
        self.log_collection = []
        self.network_data = []
        self.memory_dumps = []
        self.analysis_reports = []

    def collect_logs(self):
        logs = []
        try:
            with open("system_logs.txt", "r") as f:
                logs = f.readlines()
        except:
            logs = ["No logs available"]
        self.log_collection.extend(logs)
        return logs

    def collect_network_data(self):
        data = []
```

R6П3_2025

```
        for i in range(10):
            data.append({"packet": i, "data": random.randint(0, 1000)})
        self.network_data.extend(data)
        return data

    def collect_memory_dump(self):
        dump = {}
        for i in range(5):
            dump[f"region_{i}"] = random.randint(1000, 5000)
        self.memory_dumps.append(dump)
        return dump

    def analyze_forensics(self):
        report = {}
        logs = self.collect_logs()
        net_data = self.collect_network_data()
        mem_dump = self.collect_memory_dump()
        report["log_summary"] = len(logs)
        report["network_summary"] = len(net_data)
        report["memory_summary"] = sum(mem_dump.values()) if mem_dump else 0
        self.analysis_reports.append(report)
        return report

    def generate_report(self):
        report = self.analyze_forensics()
        report_json = json.dumps(report, indent=4)
        with open("forensic_report.json", "w") as f:
            f.write(report_json)
        return report_json

    def start_periodic_analysis(self):
        while True:
            self.generate_report()
            time.sleep(15)

class HoneypotDeployment:
    def __init__(self, host='0.0.0.0', port=9000):
        self.host = host
        self.port = port
        self.running = False
        self.connections = []
        self.server_socket = None

    def start_server(self):
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.server_socket.bind((self.host, self.port))
        self.server_socket.listen(5)
        self.running = True
        while self.running:
            r, _, _ = select.select([self.server_socket], [], [], 1)
            if self.server_socket in r:
                client_socket, addr = self.server_socket.accept()
                self.log_connection(addr)
                t = threading.Thread(target=self.handle_client,
args=(client_socket,))
                t.daemon = True
                t.start()

    def handle_client(self, client_socket):
        try:
            data = client_socket.recv(1024)
            if data:
```

КбП3_2025

```
                response = b"HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<h1>Honeypot Response</h1>"
                client_socket.sendall(response)
        except:
            pass
        finally:
            client_socket.close()

    def log_connection(self, addr):
        self.connections.append(addr)
        with open("honeypot_connections.txt", "a") as f:
            f.write(str(addr) + "\n")

    def stop_server(self):
        self.running = False
        if self.server_socket:
            self.server_socket.close()

    def start(self):
        t = threading.Thread(target=self.start_server)
        t.daemon = True
        t.start()

class CyberSecurityEnhancementsSystem:
    def __init__(self):
        self.dashboard = RealTimeDashboard(host='0.0.0.0', port=8000)
        self.incident_response = IncidentResponseSystem()
        self.ml_manager = MLModelManagement()
        self.forensics = ForensicAnalysisModule()
        self.honeypot = HoneypotDeployment(host='0.0.0.0', port=9000)
        self.running = True

    def simulate_incidents(self):
        incidents = ["Alert from sensor IP:192.168.1.100", "Alert from sensor
IP:10.0.0.5", "Alert from sensor IP:172.16.0.10"]
        while self.running:
            incident = random.choice(incidents)
            self.incident_response.add_incident(incident)
            time.sleep(2)

    def simulate_ml_training(self):
        dummy_data = [random.random() for _ in range(100)]
        dummy_labels = [random.randint(0, 1) for _ in range(100)]
        while self.running:
            self.ml_manager.train_model("intrusion", dummy_data, dummy_labels,
epochs=3)
            time.sleep(5)

    def simulate_forensic_collection(self):
        while self.running:
            self.forensics.collect_logs()
            self.forensics.collect_network_data()
            self.forensics.collect_memory_dump()
            time.sleep(10)

    def run(self):
        self.incident_response.start()
        self.honeypot.start()
        t1 = threading.Thread(target=self.dashboard.start)
        t1.daemon = True
        t1.start()
        t2 = threading.Thread(target=self.simulate_incidents)
        t2.daemon = True
```

КСТВ_2025

```
        t2.start()
        t3 = threading.Thread(target=self.simulate_ml_training)
        t3.daemon = True
        t3.start()
        t4 = threading.Thread(target=self.forensics.start_periodic_analysis)
        t4.daemon = True
        t4.start()
        t5 = threading.Thread(target=self.simulate_forensic_collection)
        t5.daemon = True
        t5.start()
        while True:
            time.sleep(1)

if __name__ == "__main__":
    system = CyberSecurityEnhancementsSystem()
    system.run()
```

К6ПЗ_2025

Файл MultiThreatIntelligenceIntegration.py

```
import threading
import time
import random
import json
import requests
import os
import networkx as nx
import statistics
from flask import Flask, jsonify, request

class MultiThreatIntelligenceIntegration:
    def __init__(self):
        self.sources = ["SourceA", "SourceB", "SourceC"]
    def query_source(self, source, indicator):
        time.sleep(random.uniform(0.1, 0.3))
        result = {"source": source, "indicator": indicator, "threat_level":
random.choice(["low", "medium", "high"]), "confidence": random.uniform(0.5,
1.0)}
        return result

    def query_all_sources(self, indicator):
        results = []
        for source in self.sources:
            result = self.query_source(source, indicator)
            results.append(result)
        return results

    def aggregate_threat_data(self, indicator):
        data = self.query_all_sources(indicator)
        threat_counts = {"low": 0, "medium": 0, "high": 0}
        confidence_values = []
        for entry in data:
            level = entry["threat_level"]
            threat_counts[level] += 1
            confidence_values.append(entry["confidence"])
        avg_confidence = statistics.mean(confidence_values) if confidence_values
else 0
        aggregated = {"indicator": indicator, "threat_counts": threat_counts,
"average_confidence": avg_confidence}
        return aggregated

class UserBehaviorAnalysis:
    def __init__(self):
        self.user_events = []
        self.lock = threading.Lock()

    def record_event(self, user_id, event_type, timestamp, details):
        event = {"user_id": user_id, "event_type": event_type, "timestamp":
timestamp, "details": details}
        with self.lock:
            self.user_events.append(event)
    def get_events(self):
        with self.lock:
            return list(self.user_events)


    def analyze_behavior(self):
        events = self.get_events()
        user_sessions = {}
```

КБПЗ_2025

```
        for event in events:
            uid = event["user_id"]
            if uid not in user_sessions:
                user_sessions[uid] = []
            user_sessions[uid].append(event)
        session_analysis = {}

        for uid, ev_list in user_sessions.items():
            ev_list.sort(key=lambda x: x["timestamp"])
            durations = []

            for i in range(1, len(ev_list)):
                duration = ev_list[i]["timestamp"] - ev_list[i-1]["timestamp"]
                durations.append(duration)
            avg_duration = statistics.mean(durations) if durations else 0
            session_analysis[uid] = {"total_events": len(ev_list),
"average_interval": avg_duration}
        return session_analysis

class ThreatHuntingModule:

    def __init__(self, log_file="historical_logs.txt"):
        self.log_file = log_file
        self.hunting_results = []

    def load_logs(self):
        if not os.path.exists(self.log_file):
            return []
        with open(self.log_file, "r") as f:
            lines = f.readlines()
        logs = [line.strip() for line in lines]
        return logs

    def process_logs(self):
        logs = self.load_logs()
        ip_occurrences = {}
        for log in logs:
            parts = log.split(" ")
            for part in parts:
                if part.count(".") == 3:
                    ip = part
                    if ip not in ip_occurrences:
                        ip_occurrences[ip] = 0
                    ip_occurrences[ip] += 1

        suspicious_ips = [ip for ip, count in ip_occurrences.items() if count >
3]
        self.hunting_results = suspicious_ips
        return suspicious_ips
    def run_hunting(self):
        while True:
            self.process_logs()
            time.sleep(5)

class IncidentReportGenerator:
    def __init__(self, report_file="incident_report.json"):
        self.incidents = []
        self.report_file = report_file
        self.lock = threading.Lock()

    def add_incident(self, incident):
        with self.lock:
            self.incidents.append(incident)
```

КеПЗ_2025

```
    def generate_report(self):
        with self.lock:
            report = {"timestamp": time.time(), "incident_count":
len(self.incidents), "incidents": list(self.incidents)}
        with open(self.report_file, "w") as f:
            json.dump(report, f, indent=4)
        return report
    def periodic_report(self):
        while True:
            self.generate_report()
            time.sleep(10)

class NetworkGraphAnalyzer:
    def __init__(self):
        self.graph = nx.Graph()

    def add_connection(self, source, target, weight):
        self.graph.add_edge(source, target, weight=weight)

    def simulate_network(self, num_nodes=10, num_edges=15):
        for i in range(num_nodes):
            self.graph.add_node(f"Node_{i}")

        for _ in range(num_edges):
            s = f"Node_{random.randint(0, num_nodes-1)}"
            t = f"Node_{random.randint(0, num_nodes-1)}"
            if s != t:
                w = random.randint(1, 10)
                self.add_connection(s, t, w)
    def analyze_graph(self):
        centrality = nx.degree_centrality(self.graph)
        clustering = nx.clustering(self.graph)
        analysis = {"centrality": centrality, "clustering": clustering}
        return analysis

    def export_graph(self, filename="network_graph.json"):
        data = nx.node_link_data(self.graph)
        with open(filename, "w") as f:
            json.dump(data, f, indent=4)
        return data

def run_multi_threat_intel():
    mti = MultiThreatIntelligenceIntegration()
    while True:
        indicator = f"192.168.1.{random.randint(1,254)}"
        aggregated = mti.aggregate_threat_data(indicator)
        print("Threat Intel:", aggregated)
        time.sleep(3)

def run_user_behavior_analysis():
    uba = UserBehaviorAnalysis()
    def simulate_events():
        while True:
            uid = random.randint(1, 5)
            event_type = random.choice(["login", "click", "logout", "download"])
            timestamp = time.time()
            details = {"info": f"Event details {random.randint(1,100)}"}
            uba.record_event(uid, event_type, timestamp, details)
            time.sleep(random.uniform(0.5, 2))
```

КОПЗ_2025

```
    t = threading.Thread(target=simulate_events)
    t.daemon = True
    t.start()
    while True:
        analysis = uba.analyze_behavior()
        print("User Behavior Analysis:", analysis)
        time.sleep(5)

def run_threat_hunting():
    thm = ThreatHuntingModule()
    if not os.path.exists(thm.log_file):
        with open(thm.log_file, "w") as f:
            for i in range(50):
                ip =
f"{random.randint(1,255)}.{random.randint(1,255)}.{random.randint(1,255)}.{rando
m.randint(1,255)}"
                f.write(f"Log entry from {ip}\n")
    thm.run_hunting()

def run_incident_report():
    irr = IncidentReportGenerator()
    def simulate_incidents():
        while True:

            incident = {"timestamp": time.time(), "description": f"Incident
{random.randint(1,100)}", "severity": random.choice(["low", "medium", "high"])}
            irr.add_incident(incident)
            time.sleep(2)
    t = threading.Thread(target=simulate_incidents)
    t.daemon = True
    t.start()
    irr.periodic_report()

def run_network_graph_analysis():
    nga = NetworkGraphAnalyzer()
    while True:
        nga.simulate_network(num_nodes=15, num_edges=25)
        analysis = nga.analyze_graph()
        print("Network Graph Analysis:", analysis)
        nga.export_graph()
        time.sleep(7)

if __name__ == "__main__":
    t1 = threading.Thread(target=run_multi_threat_intel)
    t2 = threading.Thread(target=run_user_behavior_analysis)
    t3 = threading.Thread(target=run_threat_hunting)
    t4 = threading.Thread(target=run_incident_report)
    t5 = threading.Thread(target=run_network_graph_analysis)

    t1.daemon = True
    t2.daemon = True
    t3.daemon = True
    t4.daemon = True
    t5.daemon = True

    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t5.start()

    while True:
        time.sleep(1)
```

КБП3_2025