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
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-25М
ОПП «Комп’ютерні науки»
спеціальності F3 «Комп’ютерні науки»
_____ Третьяков Н.Г.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2026 р.

Нормоконтроль проекту
_____ Коваленко А.С.
«____» _____ 2026 р.

Рецензент _____
_____

Підписано КЕП
Центральноукраїнський національний технічний університет
Третьяков Нікіта Геннадійович
Сертифікат: 514B5C86A1E5DA1104000000741F2E000DCBF704
Дійсний з 02.12.2025 22:18:51 по 02.12.2026 22:18:50

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _F_____*"Інформаційні технології"*_____

Спеціальність F3 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 1 » вересня 2026 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Третьякову Нікіті Геннадійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи мутаційного тестування програмних об'єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту*

2. Керівник роботи *Коваленко Анна Степанівна, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 42-13 від 17.08.2026 року

3. Строк подання студентом роботи до захисту *2.12.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи мутаційного тестування програмних об'єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту*

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

1. Призначення та область використання.	*5. Впровадження системи в промислову експлуатацію*
2. Перегляд аналогічних існуючих систем.	*6. Наукова новизна.*
3. Опис і обґрунтування проектних рішень.	*7. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*8. Висновки.*

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

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

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2026	16.11.2026

8. Дата видачі завдання « 1 » вересня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2026 р.	
3.	Розробка моделі компонента	20.10.2026 р.	
4.	Розробка структур даних	25.10.2026 р.	
5.	Розробка алгоритмів системи	30.10.2026 р.	
6.	Програмування алгоритмів	10.11.2026 р.	
7.	Розрахунки з охорони праці та техніки безпеки	15.11.2026 р.	
8.	Оформлення ПЗ	17.11.2026 р.	
9.	Попередній захист роботи	2.12.2026 р.	

Дата видачі завдання
« 1 » вересня 2026 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 1 » вересня 2026 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Третьяков Н.Г. Дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту. ФЗ Комп’ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Метою розробки є дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Об’єктом дослідження є процес мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Предметом дослідження є методи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

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: комп’ютерні науки, тестування програмних об’єктів, штучний інтелект, антивірусного захисту

ABSTRACT

Tretyakov N.G. Research and Software Implementation of a Mutation Testing System for Software Objects Using Artificial Intelligence to Detect Blind Spots in Antivirus Protection. F3 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This final qualification thesis, submitted for the second (master's) level of higher education, presents the development of software designed for a mutation testing system for software objects, utilizing artificial intelligence to identify blind spots in antivirus protection.

The objective of this development is to research and implement a software-based mutation testing system for software objects that employs artificial intelligence to detect blind spots in antivirus protection.

The object of the research is the process of mutation testing for software objects, using artificial intelligence to identify blind spots in antivirus protection.

The subject of the research consists of methods for mutation testing of software objects, utilizing artificial intelligence to detect blind spots in antivirus protection.

The research methodology is based on techniques from artificial intelligence, mathematical statistics, and software engineering.

The outcome of the work is a software implementation of a mutation testing system for software objects that uses artificial intelligence to detect blind spots in antivirus protection.

During the development of the software model, an analysis of existing hardware and software tools was conducted. All components of the developed software are described in full detail.

A user-friendly interface has been developed. Instructions for using the software tools are provided.

The program can be used on PCs running the Windows 10/11 operating system.

The program was developed using the Python environment.

Keywords: computer science, software testing, artificial intelligence, antivirus protection

ВІДОМІСТЬ ПРОЕКТУ

		Текстові документи			
1	А4	Пояснювальна записка	97	–	
2	А4	Робоча програма	26	–	Додаток А
		Графічні документи			
3	А4	Наукова новизна	1	–	
4	А4	Структурна схема	1	–	
5	А4	Функціональна схема роботи системи	1	–	
6	А4	Діаграма взаємодії процесів	1	–	
7	А4	Блок-схема основної програми	1	–	
8	А4	Блок-схема роботи підпрограми	1	–	

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 10
ВСТУП..... 11
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 14
1.1 Призначення системи..... 14
1.2 Область застосування 15
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 16
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 16
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 22
2.3 Розгорнута постановка завдання 23
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 25
3.1 Опис функціонування системи 25
3.2 Розробка структурної схеми..... 28
3.3 Розробка функціональної схеми 41
3.4 Розробка діаграми процесів 43
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 45
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 45
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 73
6 НАУКОВА НОВИЗНА 78

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 79
7.1 Вступ 79
7.2 Шкідливі і небезпечні фактори при роботі з комп’ютером 80
7.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 81
7.4 Розробка заходів з умов поліпшення охорони праці 84
7.5 Розрахункова частина 85
8 ОСНОВНІ ВИСНОВКИ 88
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 90

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КМ	–	комп’ютерна мережа
КСАЗ	–	комплексна система антивірусного захисту
МЕ	–	міжмережний екран
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ACL	–	Access Control List
FTP	–	File Transfer Protocol
http	–	HyperText Transfer Protocol
POP3	–	Post Office Protocol Version 3
SMTP	–	Simple Mail Transfer Protocol
VLAN	–	Virtual Local Area Network

ВСТУП

Актуальність теми. П'ять років тому золотим стандартом були периметральний брандмауер та антивірус на основі сигнатур. Ця епоха закінчилася. Зловмисники тепер використовують моделі великих мов програмування для створення поліморфного шкідливого програмного забезпечення, яке перезаписує власну сигнатуру з кожним циклом виконання. Якщо ваш стек безпеки спирається на вчорашні методи виявлення, ви вже скомпрометовані. Перш ніж розглядати конкретні інструменти, потрібно зрозуміти зміну парадигми, яка змінює галузь. Основна проблема полягає в асиметрії швидкості. Корисне навантаження програми-вимагача може зашифрувати весь мережевий ресурс менш ніж за чотири хвилини. Аналітику-людині потрібно від п'ятнадцяти до тридцяти хвилин лише для сортування сповіщення. Автоматизоване реагування стосується платформ безпеки, які виявляють, класифікують та нейтралізують загрози без втручання людини, працюючи в мілісекундних часових інтервалах, що відповідають швидкості сучасних атак.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

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

– Огляд існуючих систем мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– Дослідження системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– Програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Об’єктом дослідження є процес мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Предметом дослідження є методи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– Розроблено вітчизняний продукт мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVIII науково-технічній конференції здобувачів вищої освіти «Наука-виробництву», основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №17.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Штучний інтелект тепер бере безпосередню участь на кожному етапі ланцюжка атак, від написання шкідливого програмного забезпечення до виконання команд усередині активних мереж з мінімальним людським керівництвом між кроками. Найяскравішим прикладом минулого року було порушення безпеки дев'яти урядових установ Мексики [1] між кінцем 2025 та початком 2026 року. Один оператор паралельно запускав Claude Code та GPT-4.1, один з яких керував активним використанням шкідливої програми протягом 34 сеансів, інший аналізував викрадені дані та автоматично керував подальшою активністю. Людина запустила архітектуру. Штучний інтелект керував операцією, створивши понад 5000 виконаних команд та розкривши приблизно 400 мільйонів записів.

Штучний інтелект також індустріалізував ринок інструментів для боротьби з шахрайством. Платформи фішингу як послуги [2] тепер вбудовують мовні моделі безпосередньо в робочий процес атаки, автоматично скануючи викрадені облікові записи, імітуючи стиль письма жертви та генеруючи переконливі подальші шахрайські електронні листи. Платформи голосового шахрайства [3] використовують повністю автоматизовані агенти ШІ, які проводять цільові дані через сценарні виклики для відновлення облікового запису, щоб викрасти одноразові паролі, без участі людини-абонента. Джейлбрейк вбудований у продукт, тому покупцеві взагалі не потрібні навички ШІ, щоб здійснити складну багатоетапну атаку.

1.2 Область застосування

Штучний інтелект (ШІ) тепер достатньо здатний міркувати про код, що одночасно пришвидшує обидві сторони гонки. Група розвідки загроз Google повідомила про першу атаку нульового дня за допомогою ШІ, створену для масового використання, тоді як інші дослідження показали, що передові моделі створюють робочі експлойти нульового дня у великих масштабах. Практичний ефект полягає в стисненні: розкриття вразливості, яке колись давало захисникам дні на реагування, тепер дає їм години. Уряд США CISA відповів, вимагаючи від установ усунути вразливості з найвищим рівнем ризику протягом трьох днів. Індійський CERT-In пішов далі, порадивши організаціям виправити критичні системи протягом 12 годин.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Захист у 2026 році вимірюється одним показником: швидкість реагування. Якщо ваші інструменти не можуть виявити шкідливий процес і завершити його протягом мілісекунд, шифрування вже триває. Наведені нижче платформи представляють сучасний стан справ.

SentinelOne та CrowdStrike: розширене виявлення та реагування (XDR)

Технічне визначення: Розширене виявлення та реагування (XDR) виходить за рамки традиційного виявлення кінцевих точок (EDR). Хоча EDR зосереджується виключно на кінцевих точках (ноутбуках, серверах, робочих станціях), XDR інтегрує телеметрію з поштових шлюзів, хмарних робочих навантажень, постачальників ідентифікації та мережевих пристроїв в єдину платформу.

Логіка роботи: платформи XDR покладаються на поведінкову евристику, а не на статичні сигнатури. Вони відстежують виклики API, зміни реєстру, операції з файлами та мережеві підключення, щоб ідентифікувати «поведінкові історії».

Коли цей ланцюжок поведінки виявляється, XDR виконує команду Rollback, використовуючи власну технологію створення знімків для відновлення зашифрованих файлів протягом кількох секунд. Пошкодження усувається до того, як оператор дізнається про те, що відбулася атака.

Під час оцінювання постачальників запитуйте демонстрацію в реальному часі, де вони запускають реальні зразки програм-вимагачів. Зверніть особливу

увагу на швидкість відкату. Будь-що понад 30 секунд вказує на архітектурні обмеження.

Darktrace: самонавчальний штучний інтелект для захисту мережі

Технічне визначення: Darktrace – це платформа автономного реагування, яка застосовує машинне навчання без учителя для аналізу мережевого трафіку. На відміну від систем на основі сигнатур, що потребують постійного оновлення правил, Darktrace вивчає, як виглядає «норма» для вашого середовища, та виявляє відхилення в режимі реального часу.

Логіка роботи: Darktrace створює шаблон життя для кожного користувача, пристрою та підмережі.

Коли стажер-маркетолог, який зазвичай користується LinkedIn, раптово ініціює сканування портів фінансових серверів за протоколом SMB, модуль Antigena від Darktrace хірургічно обмежує це з'єднання, водночас пропускаючи легітимний трафік. Така точність забезпечує безперервність бізнесу, нейтралізуючи загрози.

Розвідка загроз та SOAR

Сире виявлення нічого не означає без контексту. Платформи розвідки загроз повідомляють вашим захисним системам, що шукати, тоді як платформи SOAR повідомляють їм, що робити, коли вони це знаходять.

Платформи розвідки загроз: MISP та OpenCTI

Технічне визначення: Платформи розвідки загроз (TIP) агрегують, нормалізують та розповсюджують індикатори компрометації (IOC) та тактики, методи та процедури (TTP) з різних джерел. Варіанти з відкритим кодом, такі як MISP та OpenCTI, конкурують з комерційними альтернативами.

Логіка роботи: Сучасні TIP спілкуються за допомогою протоколів STIX/TAXII.

Ніколи не споживайте канали загроз пасивно. Налаштуйте свій TIP на автоматичне збагачення IOC даними WHOIS, пасивним DNS та оцінками VirusTotal перед надсиланням до систем виявлення. Необроблені канали без контексту призводять до втоми від сповіщень.

Splunk SOAR та Tines: автоматизована оркестрація реагування

Технічне визначення: SOAR-платформи об’єднують ваші інструменти безпеки за допомогою автоматизованих плейбуків, що дозволяє скоординовано реагувати через брандмауери, EDR, шлюзи електронної пошти та системи продажу заявок.

Логіка роботи: Коли фішинговий електронний лист обходить ваш шлюз, автоматично виконується сценарій SOAR.

Splunk SOAR пропонує візуальний редактор сценаріїв. Tines використовує автоматизацію на основі JSON.

Хмарна безпека: платформи CNAPP для багатохмарної видимості

Традиційні засоби захисту периметра руйнуються в хмарних середовищах, де інфраструктура розгортається за лічені хвилини. Платформи захисту додатків, орієнтовані на хмару (CNAPP), забезпечують єдину видимість в AWS, Azure та Google Cloud.

Безпека Wiz та Orca: хмарна безагентна безпека

Технічне визначення: Інструменти CNAPP об'єднують керування станом безпеки хмари (CSPM), захист робочого навантаження хмари (CWPP) та сканування вразливостей в єдину платформу. Безагентні архітектури сканують хмарні середовища через інтеграцію API без встановлення програмного забезпечення на екземпляри.

Логіка роботи: платформи CNAPP виконують постійну оцінку ризиків.

Wiz чудово справляється з візуалізацією шляхів атаки, показуючи, як публічний веб-сервер з відомою вразливістю підключається до бази даних, що містить записи клієнтів. Orca Security зосереджується на глибокій видимості середовища виконання, виявляючи активні загрози в хмарних робочих навантаженнях.

Спочатку ввімкніть інструменти CNAPP у режимі «лише для читання». Багато організацій виявляють тисячі неправильних конфігурацій з першого дня. Перш ніж намагатися масово виправити помилки, розставте пріоритети за ступенем ризику та чутливістю даних.

Атакувальна безпека: інструменти для червоних команд та тестувальників на проникнення

Розуміння наступальних інструментів не є необов'язковим. Ви не можете захистити те, чого не розумієте. Наведені нижче платформи представляють стандарт для авторизованого тестування безпеки.

Metasploit та Cobalt Strike: Фреймворки для експлуатації

Технічне визначення: Фреймворки експлуатації автоматизують виявлення, перетворення на зброю та здійснення атак на вразливі системи. Metasploit слугує стандартом з відкритим кодом, тоді як Cobalt Strike домінує у комерційних взаємодіях червоних команд.

Логіка роботи: Сучасна експлуатація відбувається за багатоетапним процесом.

Cobalt Strike представляє корисні навантаження «Beacon», призначені для уникнення виявлення за допомогою таймерів сну, джиттера (рандомізованих інтервалів зв'язку) та фронтування домену (приховування трафіку C2 за легітимними сервісами CDN).

Ніколи не використовуйте конфігурацію Cobalt Strike за замовчуванням. Defenders підтримують бази даних сигнатур для стандартних маячків. Налаштуйте свій профіль Malleable C2, щоб імітувати законні моделі трафіку, такі як виклики Slack API або завантаження на Google Диск.

Burp Suite Professional: Тестування веб-застосунків

Технічне визначення: Burp Suite – це проксі-сервер-перехоплювач, який розміщується між вашим браузером та цільовими веб-додатками, дозволяючи вам перехоплювати, змінювати та відтворювати HTTP-запити для виявлення вразливостей, таких як SQL-ін'єкції та XSS.

Логіка роботи: робочий процес Burp зосереджений на ручному та автоматизованому тестуванні.

Поєднуйте Burp з Академією веб-безпеки PortSwigger. Кожну техніку, яку ви вивчите в безкоштовних навчальних лабораторіях, можна безпосередньо застосувати до реальних оцінювань.

Конфіденційність та автентифікація: фундаментальна гігієна безпеки

Атакуючі та захисні інструменти нічого не варті, якщо ваші облікові дані викрадено через фішинг. Наведені нижче технології представляють мінімально прийнятні стандарти для всіх, хто працює у сфері безпеки.

YubiKey та апаратна автентифікація

Технічне визначення: Апаратні ключі безпеки, що використовують протоколи FIDO2/WebAuthn, забезпечують стійку до фішингу багатофакторну автентифікацію, вимагаючи фізичного володіння криптографічним пристроєм для входу.

Логіка роботи: Традиційна 2FA надсилає код через SMS або додаток-автентифікатор. Зловмисник, який використовує фішинговий проксі-сервер у режимі реального часу, може перехопити та миттєво відтворити його. FIDO2 Коли ви автентифікуєтесь за допомогою YubiKey, ваш браузер надсилає криптографічний запит, унікальний для домену, який ви відвідуєте (наприклад, github.com). YubiKey підписує цей запит закритим ключем, який ніколи не залишає пристрій. Навіть на ідеальному клоні сторінки входу GitHub запит не збігатиметься, і автентифікація не вдасться.

Зареєструйте кілька YubiKey для ваших критично важливих облікових записів (основного та резервного). Зберігайте резервну копію в окремому фізичному місці. Втрата єдиного ключа без кодів відновлення означає безповоротне блокування облікового запису.

ProtonVPN та мережі, орієнтовані на конфіденційність

Технічне визначення: Віртуальна приватна мережа (VPN) шифрує ваш інтернет-трафік і направляє його через посередницький сервер, маскуючи вашу IP-адресу від веб-сайтів і інтернет-провайдерів.

Логіка роботи: VPN-протоколи визначають безпеку та продуктивність.

ProtonVPN працює відповідно до швейцарських законів про конфіденційність із перевіреною політикою відмови від журналів та підтримує

Secure Core (багатострибкова маршрутизація) та NetShield (блокування реклами/трекерів).

VPN переносять довіру з вашого інтернет-провайдера на постачальника VPN. Для справжньої анонімності використовуйте браузер Tor. Для забезпечення конфіденційності від вашого інтернет-провайдера, зберігаючи при цьому нормальну швидкість, використовуйте надійний VPN.

Створення професійної лабораторії: Free Stack

Вам не потрібен корпоративний бюджет для розвитку професійних навичок. Наведена нижче конфігурація працює на одному ноутбуці та забезпечує практичний досвід роботи з інструментами корпоративного рівня.

Технічне визначення: Домашня лабораторія кібербезпеки – це ізольоване віртуальне середовище, де ви легально відпрацьовуєте наступальні методи проти навмисно вразливих систем, не ризикуючи виробничими мережами.

Професійний досвід не вимагає корпоративних бюджетів. Ви можете створити функціональний SOC на одному ноутбуці, використовуючи безкоштовне програмне забезпечення з відкритим кодом.

Робочий процес конфігурації лабораторії:

Крок 1: Встановіть VirtualBox з мережевим адаптером «Тільки для хоста». Це ізолює вашу лабораторію від домашньої мережі та Інтернету.

Крок 2: Розгорніть Kali Linux як платформу для атаки:

sudo apt update && sudo apt install metasploit-framework

Крок 3. Розгорніть Windows за допомогою безкоштовних образів Microsoft «Enterprise Evaluation» (90-денні ліцензії для тестування).

Крок 4: Встановіть агент Wazuh у Windows. Під час сканування з Kali:

nmap -sV [Windows_VM_IP]

Дані сканування відображаються на вашій інформаційній панелі Wazuh, генеруючи сповіщення, що відображають спостереження за виробничим SOC.

Документуйте кожне лабораторне заняття в особистій вікі. Включайте виконані команди, виявлені помилки та знайдені рішення. Це стає безцінним під час співбесід та реагування на реальні інциденти.

Висновок: Автоматизація як конкурентна перевага

Інструмент не робить хакера справжнім, але у 2026 році ви не зможете конкурувати, не опанувавши найкращі інструменти кібербезпеки на основі штучного інтелекту 2026 року, які визначають сучасний ландшафт. Незалежно від того, чи захищаєте ви глобальне підприємство, чи навчаєтесь у домашній лабораторії, імператив залишається актуальним: автоматизувати повторюване, щоб люди могли зосередитися на винятковому.

Збір журналів, керування виправленнями та моніторинг базових показників – це вирішувані проблеми. Штучний інтелект вирішує їх швидше, ніж будь-яка людська команда. Ваша цінність полягає в питаннях, на які машини не можуть відповісти: пошук загроз з урахуванням бізнес-контексту, реагування на інциденти з урахуванням організаційної політики, балансування архітектури безпеки з захистом та операціями.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для розробки програмного комплексу мутаційного тестування і виявлення сліпих зон антивірусного захисту було обрано мову програмування високого рівня Python. Вибір цього інструментарію зумовлений його гнучкістю, лаконічністю синтаксису, кросплатформовістю та потужною екосистемою бібліотек для роботи з алгоритмами штучного інтелекту, машинного навчання, аналізу структури файлів і взаємодії з операційною системою.

Python є інтерпретованою мовою з динамічною типізацією, що суттєво прискорює процес прототипування складних аналітичних алгоритмів та реалізації генераторів мутацій програмних об'єктів. Попри те, що інтерпретований код поступається за швидкістю виконання нативним компільованим мовам (таким як C/C++), продуктивність Python є достатньою для побудови ефективних тестових конвеєрів. Крім того, архітектура мови дозволяє за потреби легко інтегрувати оптимізовані модулі, написані на C/C++,

для виконання ресурсомістких обчислень.

Завдяки розвиненому набору вбудованих та сторонніх бібліотек, Python широко застосовується в наукових дослідженнях з кібербезпеки, автоматизованому аналізі вразливостей, розробці інструментів реверс-інжинірингу та створенні інтелектуальних агентів на базі методів штучного інтелекту. У межах даної роботи Python забезпечує реалізацію таких ключових компонентів системи:

– модулів генерації мутацій: програмної трансформації досліджуваних об'єктів та генерації варіантів з метою тестування стійкості антивірусних детекторів;

– інтелектуальних компонентів ШІ: взаємодії з нейромережевими моделями для виявлення «сліпих зон» та оптимізації процесів мутаційного тестування;

– автоматизованих конвеєрів перевірки: взаємодії із захисними механізмами, збору телеметрії та формування звітів про ефективність виявлення загроз.

Для написання, відлагодження та супроводу програмного забезпечення використовувалися сучасні інтегровані середовища розробки (IDE) та текстові редактори, зокрема PyCharm та Visual Studio Code, які забезпечують зручні інструменти контролю версій, аналізу синтаксису та організації модульної структури проєкту.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

В процесі розробки випускної кваліфікаційної роботи за другим

(магістерським) 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 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 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.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Оскільки організації впроваджували штучний інтелект в електронну пошту, документи, код та основні бізнес-процеси, стек штучного інтелекту сам по собі став мішенню. Непряме впровадження запитів, коли шкідливі інструкції приховані всередині контенту, який ШІ читає в рамках своєї звичайної роботи, перейшло від підтвердження концепції до операційної загрози. Check Point AI Security зафіксувала приблизно п'ятикратне збільшення виявлень великих шкідливих корисних навантажень впровадження запитів між березнем і травнем 2026 року, що відповідає тому, що непряме впровадження стало рутинним шляхом атаки, а не теоретичним.

Інфраструктура штучного інтелекту також стала мішенню за допомогою звичайних засобів. Критичний недолік в Ollama [4] призвів до витоку запитів, ключів та змінних середовища приблизно з 300 000 серверів моделей, підключених до Інтернету. GreyNoise [5] зафіксував близько 91 000 сеансів атаки, що досліджували розгортання LLM, протягом одного кварталу. Ланцюг постачання програмного забезпечення штучного інтелекту виявився однаково вразливим: черв'як Shai-Hulud у листопаді 2025 року скомпрометував сотні широко використовуваних пакетів коду та десятки тисяч репозиторіїв, викравши облікові дані розробників, автоматично поширюючись через конвеєри збірки.

Що штучний інтелект зробив із цифровою ідентифікацією

Голос, обличчя, документи та відео в реальному часі тепер можна переконливо синтезувати, а це означає, що жоден з них не може служити самостоятельним доказом особи. За останній рік [6] обмін обличчями в реальному часі перейшов від операцій на рівні національних держав до індустріального шахрайства. Підробка документів стала комерціалізованою до такої міри, що один сервіс продав понад 10 000 підроблених посвідчень особи, згенерованих

штучним інтелектом, здатних проходити перевірки KYC у 56 країнах. Пов'язана з Північною Кореєю група зайшла далі в цьому, використовуючи створені штучним інтелектом персони, щоб найняти оперативників у західні компанії як законних віддалених працівників, заробивши близько 800 мільйонів доларів на програми озброєння режиму.

Скільки конфіденційних даних витікає через інструменти штучного інтелекту підприємства

За минулий рік кількість запитів GenAI з високим рівнем ризику подвоїлася з 2 до 4 відсотків. В середньому організація запускає 10 застосувань ШІ на місяць, багато з яких без офіційного схвалення. Сфера бізнес-послуг мала найвищий показник серед усіх галузей: майже кожна 17-та взаємодія зі ШІ несла реальний ризик витоку конфіденційних даних, а до травня 2026 року цей показник зріс до 1-ї взаємодії зі ШІ. Більшість цього ризику виникає не внаслідок атак, а внаслідок звичайного схваленого використання, коли співробітники діляться більшою кількістю контексту, ніж вони усвідомлюють, щоб отримати корисну відповідь.

Ризики у цьому розділі поділяються на три категорії, і кожна з них вимагає різного виду захисту: захист самого ШІ, узгодження швидкості атак на основі ШІ та регулювання того, як ШІ фактично використовується в робочій силі.

Захист систем штучного інтелекту

Більшість команд безпеки не можуть бачити найризикованішу частину власної поверхні атаки ШІ, тому захист починається з видимості та поширюється на те, як поводяться агенти після їх запуску:

– Безпека агентів на основі штучного інтелекту керує тим, як агенти взаємодіють із запитами, інструментами, даними та діями в режимі реального часу, запобігаючи маніпуляціям через впровадження запитів, отруєні конфігурації та небезпечне використання інструментів.

– AI Red Teaming перевіряє, чи можна обманом змусити додатки штучного інтелекту розкрити конфіденційні дані або обійти політики, перш ніж

зловмисники отримають таку можливість, і знову перевіряє безпеку після кожної суттєвої зміни моделей, підказок або дозволів.

– WAF, що працює на базі двошарового двигуна машинного навчання, блокує оперативні впровадження та небезпечний контент на периметрі без потреби в підписах та без простоїв.

– AI Factory Security забезпечує багаторівневий захист для організацій, що створюють власну інфраструктуру штучного інтелекту, охоплюючи обладнання, робочі навантаження, контейнери, API-інтерфейси виводу та кінцеві точки.

– Управління експозицією виявляє кожен ресурс штучного інтелекту, що підключений до Інтернету, включаючи відкриті сервери моделей та панелі керування агентами, і позначає щойно відкриту інфраструктуру в момент її появи.

Відповідність темпам атак на основі штучного інтелекту

Тепер вторгнення охоплюють десятки цілей одночасно, а ШІ виконує оперативну роботу між реєстраціями. Команди безпеки, які працюють зі швидкістю людини, просто не можуть встигати за таким темпом, тому захист також має працювати на ШІ:

– Штучний інтелект ThreatCloud працює одночасно на двох швидкостях, генеруючи безперервний фоновий аналіз та відповідаючи на запити в режимі реального часу від датчиків Check Point по всьому світу, підключених через мережі, електронну пошту, кінцеві точки, мобільні пристрої та хмару.

– Програма готовності моделей Frontier AI, включаючи внутрішню, незалежну від моделі технологію Check Point BLAST, проактивно виявляє та усуває вразливості в моделях Frontier, які живлять власний захист Check Point, усуваючи прогалину, перш ніж їх можна буде використати як зброю деінде.

Управління використанням ШІ зсередини

Значна частина розкриття інформації, про яку йдеться у цьому звіті, взагалі не була результатом атаки. Вона сталася внаслідок звичайного, дозволеного використання, коли співробітники поділилися більшою кількістю

інформації, ніж вони усвідомлювали, лише для того, щоб отримати корисну відповідь:

– Служба захисту робочої сили на основі штучного інтелекту виявляє як санкціоновані, так і несанкціоновані програми штучного інтелекту по всій організації та застосовує запобігання втраті даних у режимі реального часу до запитів GenAI, щоб облікові дані, вихідний код і дані клієнтів не виходили через щоденне використання штучного інтелекту.

– Керування експозицією розширює цю ж видимість на зовнішню поверхню, ранжуючи експозиції за тим, що дійсно придатне для використання, а не за обсягом сканування, а завдяки своїм шарам захисту бренду та розвідки загроз він також виявляє фішингові сторінки, клоновані сайти та викрадені облікові дані, включаючи логіни для служб штучного інтелекту, які вже циркулюють у глибокому та темному вебі.

3.2 Розробка структурної схеми

Працівниця читає електронний лист, який відображає спосіб написання її менеджера, приєднується до відеодзвінка з обличчями, які вона впізнає, та схвалює платіж, який не викликає тривоги. Ніщо в цій послідовності не спрацьовує проти попереджувальних знаків, які навчає персонал розпізнавати навчання з безпеки. Той самий клас програмного забезпечення, який створював електронний лист, також забезпечує інструменти, які її команда безпеки використовує для виявлення підробок. Штучний інтелект тепер служить як зловмиснику, так і захиснику в одному інциденті, і 2026 рік дав перші вагомі докази того, що дає ця домовленість з кожної сторони. Компанії, державні установи та звичайні споживачі стикаються зі зловмисниками, чиї інструменти вдосконалюються за тією ж кривою, що й захист, який їх переслідує.

Швидкість та обсяг – ось що зловмисники купують за допомогою ШІ. Модель пише чистий текст будь-якою мовою за лічені секунди, піднімає голос із публічного запису та створює код, який виглядає по-різному на кожному

запуску. Невеликі злочинні групи, яким колись потрібна була зарплата для організації серйозної кампанії, тепер розтягують кількох операторів на десятки цілей одночасно. Захисники стикаються зі зворотною арифметикою. Одне велике підприємство може залишити кілька сотень тисяч систем, звернених до Інтернету, серед яких хмарні сервери, покинуті тестові середовища та все, що прибуло з останнім придбанням, і жоден штат спеціалістів з безпеки реального розміру не може перевірити цю нерухомість вручну. Наймання не скорочує розрив. Нестача кваліфікованих кіберфахівців найбільше тисне на фірми, що базуються в регіонах Африки на південь від Сахари, Латинської Америки та Карибського басейну. Обидві сторони тепер несуть свою ціну, одна враховує вартість порушення, а інша – те, скільки заощаджує сильний захист.

У новому звіті Всесвітнього економічного форуму цифри наводяться як для атак, так і для захисту. Дослідження «Розширення можливостей захисників: ШІ для кібербезпеки» було опубліковано у травні спільно з KPMG. У ньому зазначається, що 94% кіберлідерів називають цю технологію визначальною силою у своїй роботі, тоді як 77% організацій вже використовують ШІ в кібероперація̈х. В основі роботи лежать двадцять задокументованих тематичних досліджень, а також внески 105 представників 84 організацій з 15 галузей. У звіті пронизує аргумент, що технологія надає захисникам перевагу лише тоді, коли організації застосовують її з планом, даними для його обробки та управлінням, щоб виявляти її помилки. Сім слайдів нижче демонструють боротьбу на практиці, від тактик, які використовують злочинці в польових умовах, до того, як реагують команди системної безпеки, називаючи інструменти, інциденти та результати, що стоять за кожним із них.

1. Фішинг, згенерований штучним інтелектом, перевершує старіші загрози

Кібершахрайство зараз перевершує програми-вимагачі як головну проблему для корпоративних керівників, і ця тенденція змінилася у січні 2026 року, і саме фішинг є головною причиною такого шахрайства. Мовні моделі забезпечують приманку. Один оператор може генерувати тисячі повідомлень

одночасно, кожне з яких позбавлене незграбності, яка колись видавала шахрайство.

Навчання з питань безпеки побудувало свою навчальну програму на цій незграбності. Персонал навчився позначати загальні звернення, граматичні помилки та дивне форматування – сигнали, які працювали, оскільки більшість шахрайських листів надходили поспіхом, часто від авторів, які працюють не рідною мовою. Модель стирає повідомлення, створюючи португальську, японську або англійську версію за лічені секунди, яка відображає реєстр банку, постачальника чи відділу кадрів.

WormGPT з'явився на форумах кіберзлочинності у 2023 році, його продавці пропонували чат-бота без захисних рейок, встановлених основними постачальниками. FraudGPT продовжував використовувати списки в даркнеті та канали Telegram, продаючи допомогу з фальшивими веб-сторінками та готовими приманками. Якість варіювалася, але обидва довели, що існує платний ринок для штучного інтелекту, побудованого на обмані.

Персоналізація знаменує собою значніший зсув. Колись фішинг через спецпідказки змушував зловмисника досліджувати кожну жертву вручну, що обмежувало кількість людей, з якими міг охопити один оператор. Модель переглядає профіль LinkedIn, прес-релізи компанії та публічні публікації жертви, а потім складає чернетку записки, згадуючи реальний проект, ім'я колеги або майбутню поїздку. Години ручної роботи на кожну жертву перетворюються на миттєві дії. Захисники відповіли, дивлячись далі за слова. Продукти безпеки електронної пошти натомість враховують поведінкові сигнали, серед яких платіжні інструкції, що змінюються без попередження, домени, зареєстровані на кілька днів раніше, та зв'язки з відправниками без попередньої історії. Компанії переробляють інструкції для персоналу на тій самій логіці, радячи співробітникам перевіряти незвичайні запити через другий канал.

У папці "Вхідні" більше не можна визначити, хто написав повідомлення. Підтвердження особи надходить з іншого місця, наприклад, із номера телефону, що вже є в файлі, внутрішнього запиту або прогулянки коридором.

2. Шахрайство Deepfake виснажує корпоративні рахунки

У 2024 році злочинці, які під час прямої трансляції видавали себе за керівників вищої ланки компанії, заволоділи приблизно 25 мільйонами доларів у британської інженерної фірми Arup, що стало найдорожчою такою крадіжкою за всю історію публічних даних. Один із фінансових співробітників Arup у Гонконзі увійшов у відеозустріч, де самозванці замінили фінансового директора та кількох колег. Кожне обличчя та голос на екрані, окрім одного справжнього учасника, були результатом генеративної моделі. На момент розкриття інформації про шахрайство працівник уже перерахував 200 мільйонів гонконгських доларів через 15 окремих платежів. Влада Гонконгу виклала справу в публічній заяві, а сама Arup підтвердила, що компанія стала ціллю атаки.

Сфальсифікований дзвінок пройшов перевірку, якій навчили довіряти співробітників: підняти слухавку або стати на камеру з особою, яка робить запит. Тут підробкою була сама камера, тому етап перевірки надав фальшиву підтримку замість реального доказу.

Синтез мовлення знизив початкові витрати. Сучасні системи клонують переконливий голос з короткого аудіозразка, а керівники, політики та співробітники, що працюють з громадськістю, розкидають корисні записи під час телефонних розмов про прибутки, інтерв'ю та конференцій.

Імітатори використовували те саме обладнання для державного сектору. У травні 2025 року ФБР попереджало, що з квітня злочинці надсилали текстові повідомлення та синтезовані голосові повідомлення, видаючи себе за високопосадовців США, отримувачи доступ до чинних та колишніх посадовців федерального та державного урядів. Оператори завойовували довіру кожної жертви, перш ніж переслідувати особисті акаунти.

Жоден брандмауер не зупиняє інструкцію, яку, на думку працівника, надіслала начальник, тому контрзаходи є частиною процедури. Тепер компанії вимагають зворотного дзвінка на номер, що зберігається у файлі, перш ніж переказати гроші. Деякі призначають кодові слова запитам з високою вартістю

або вимагають схвалення двох осіб для великих сум. Банки та постачальники ідентифікаційних даних додали функцію виявлення живих облич, яка шукає підписи штучно створеного обличчя.

Програмне забезпечення, створене для виявлення підроблених відео, вдосконалюється, як і генератори, які воно переслідує. Елементи керування, які ніколи не просять людину оцінити, чи справжнє обличчя, переживають кожен раунд цього змагання.

3. Агентний ШІ здійснює вторгнення з мінімальним керівництвом

Anthropic відстежила дві кампанії з вторгнення у 2025 році до операторів, які передали більшу частину роботи своєму інструменту Claude Code, і у випадку з листопадом програмне забезпечення взяло на себе від 80% до 90% зусиль проти приблизно 30 цілей. Компанія опублікувала обидва звіти, перетворивши теоретичне занепокоєння щодо автономних агентів атаки на задокументовані інциденти з пов'язаними жертвами.

У серпневому звіті описується злочинець-одинак, який націлив кодувальний агент на щонайменше 17 організацій, серед яких були служби екстреної допомоги, медичні заклади, державні органи та релігійні установи. Інструмент розвідував цих жертв, збирав облікові дані та складав наступні записки з вимогою вимагання, деякі з яких вимагали понад 500 000 доларів. Один оператор склав список цілей, який міг би зайняти цілу команду протягом місяців.

Листопадове розкриття інформації мало високий рівень достовірності, пов'язаний з китайською державною групою, цілями якої були технологічні фірми, виробники хімічної продукції, фінансові установи та урядові установи. Лише кілька спроб злому були успішними. Оператори залишали собі кілька моментів прийняття рішень, залишаючи агенту розбиратися з усім навколо них.

Ці зловмисники подолали захист моделі лише за допомогою фреймінгу, розділивши операцію на невеликі завдання, кожне з яких самостійно вважалося рутинним, та повідомляючи системі, що вона обслуговує охоронну фірму, яка проводить авторизовані тести. Агент виконав вимоги, хоча й вигадав результати,

позначаючи захоплені логіни, які виявилися недійсними.

Розвідувальні та експлойт-написання, які б поглинули людську команду протягом кількох днів, натомість виконувалося як тисячі автоматизованих запитів, звужуючи вікно між тим, як вразливість стане публічною, і тим, як злочинці з'являться, щоб її використати. Цикли виправлень, побудовані на кількох тижнях очікування, втрачають цю подушку безпеки.

Anthropic тепер виявляє зловживання, блокує облікові записи, пов'язані з атаками, і публікує результати своєї роботи, що розміщує комерційних постачальників штучного інтелекту в ланцюжку захисту поруч із клієнтами, які купують їхні продукти. Захисник, який оцінює загрозу за кількістю персоналу, що стоїть за нею, неправильно оцінить майже кожну операцію, спрямовану на нього.

4. Виявлення за допомогою штучного інтелекту зменшує вартість порушення

Компанії, які використовують штучний інтелект та автоматизацію безпеки за повною програмою, скорочують витрати приблизно на 1,9 мільйона доларів за кожне порушення та вирішують інциденти приблизно на 80 днів раніше, згідно з даними IBM 2025 року аналіз виявлених витрат на витік даних. Обидві цифри порівнюються із середніми світовими витратами на витік даних у розмірі 4,44 мільйона доларів, загальна сума яких знизилася в тому ж дослідженні, оскільки почалася швидша ідентифікація та стримування.

Виявлення аномалій виконує більшу частину цієї роботи. Модель машинного навчання вивчає, як виглядає звичайний день у мережі, відображаючи місця, з яких співробітники входять у систему, сервери, які обмінюються даними один з одним, та звичайний обсяг, що переміщується між ними. Будь-що, що виходить за рамки шаблону, наприклад, один обліковий запис, що видаляє кілька тисяч файлів до світанку, викликає тривогу.

Зловживання інсайдерами, викрадені логіни та штами шкідливого програмного забезпечення: жоден постачальник не каталогізував усі продукти, що прослизають, що зіставляють код з відомими сигнатурами. Поведінкові

базові показники виявляють їх. Охоплення також сягає території, яку жоден штатний персонал не міг би перевірити вручну, включаючи хмарні робочі навантаження, що розширюються на вимогу, та парки ноутбуків, що налічують тисячі.

Генеративні асистенти розташовані над рівнем виявлення. Шарлотта ШІ з CrowdStrike, Security Copilot від Microsoft і Google Twins з відділу операцій безпеки відповідають на запитання звичайною англійською мовою, дозволяючи аналітику допитати підозрілий сценарій, запросити короткий виклад інциденту або попросити про чернетку правила виявлення.

Тиск на персонал значною мірою пояснює цю привабливість. У звіті було виявлено системну нестачу кваліфікованих фахівців з кібербезпеки, причому найбільший дефіцит талантів спостерігається в організаціях, розташованих у країнах Африки на південь від Сахари, Латинській Америці та Карибському басейні. Молодший співробітник, який працює разом з асистентом, тепер може завершувати справи, які раніше обслуговувалися старшим спеціалістом.

Хибнопозитивні результати обмежують вигоди. Неправильно калібрована система заглушає своїх аналітиків шумом, а нерівномірний збір журналів позбавляє модель доказів, необхідних для будь-яких висновків. 1,9 мільйона доларів від IBM належать фірмам, які впровадили цю технологію в цілу програму безпеки, тому один інструмент, прикріплений до недоторканого стеку, не отримує жодної користі. Економіка порушень стала мірилом того, наскільки глибоко сягає ця проводка.

5. Агентське сортування звільняє години аналітиків

ІБМ. Платформа ATOM компанії щомісяця виконує понад 850 годин рутинної роботи аналітиків і скорочує на 37% проміжок часу між початком розслідування та його завершенням. Ці години приблизно дорівнюють щомісячній продуктивності п'яти штатних аналітиків, команди, яку більшість команд моніторингу навряд чи зможуть найняти та утримати. IBM покладається на платформу для забезпечення стабільного виявлення та реагування в будь-який час для клієнтів по всьому світу.

Брандмауери, програмне забезпечення для кінцевих точок, хмарні сервіси та платформи ідентифікації разом щодня надсилають тисячі сповіщень до черги моніторингу великого підприємства, і майже всі вони нічого не варті. Аналітики все одно відкривають кожне з них. Повторення притупляє увагу та дозволяє справжнім вторгненням пропустити повз увагу.

Агент обробляє перший прохід, перш ніж людина щось побачить. Він збирає записи журналу, переглядає канали інформації про загрози, визначає, чий обліковий запис і яка машина спрацювали, та оцінює серйозність знайденого. Потім аналітик відкриває письмовий звіт про інцидент із запропонованим наступним кроком замість простого попередження, яке коштувало б години копання.

Продукти оркестрації зі сценаріями попереднього покоління зіставляли кожну подію, що запускала подію, з однією попередньо встановленою дією та зупинялися на ній. Агентна система вибирає наступне питання з будь-якої останньої відповіді, що з'явилася, опрацьовуючи справу так само, як це робить людина-слідчий. Покрокове обмірковування інциденту знаменує собою розрив із попередньою автоматизацією.

Агент, якому дозволено закривати облікові записи або переміщувати машини на карантин, може зупинити легітимні операції в той момент, коли він неправильно оцінить ситуацію, тому команди залишають важливі рішення за схваленням людиною. Програмне забезпечення досліджує та пропонує. Людина авторизує відповідь.

Зловмисники обирають ночі, вихідні та свята, коли списки рятувальників не вистачає. Агент починає копати, щойно надходить сигнал, скорочуючи затримку між виявленням та першим захисним ходом.

Кваліфіковані фахівці залишаються незамінними. Хтось налаштовує агента, перевіряє його висновки та працює над справами, які він передає. Дефіцитний ресурс переходить від годин роботи консолі до оцінки, необхідної для перевірки машинного виводу, що змінює те, для чого повинна наймати команда безпеки.

6. Сканування за допомогою штучного інтелекту скорочує час виявлення вразливостей

Скоротився час перевірки кожного сайту в інвентаризації понад 100 000 об'єктів, підключених до Інтернету, скоротивши те, що колись займало 15 хвилин, приблизно до хвилини або менше. Відкритий слід у великому підприємстві сягає сотень тисяч активів, значна частина яких накопичується, і ніхто не вирішує їх зберегти. Покинуті тестові середовища, хмарні сервери, запущені для одного проекту, та інфраструктура, що з'явилася в результаті придбання, відкривають нові можливості, і багато компаній не можуть перелічити їх усі. Машинні сканери без зупинки сканують всю нерухомість, сортуючи активи за рівнем експозиції та просуваючи вперед ті, що мають неправильну конфігурацію.

Big Sleep, агент, створений спільно інженерами Google. Підрозділи Project Zero та DeepMind виявили вразливість безпеки пам'яті всередині механізму баз даних SQLite наприкінці 2024 року. У липні 2025 року Google повідомив, що той самий інструмент виявив окрему вразливість у SQLite, зареєстровану як CVE-2025-6965, яку хакери вже обговорювали. Компанія назвала цей момент віхою. Агент втрутився, щоб заблокувати спробу живої атаки, перш ніж вона досягла успіху.

Дворічний конкурс DARPA на кібербезпеку штучного інтелекту (AI Cyber Challenge) спонукав учасників розробляти системи, здатні знаходити та виправляти недоліки у відкритому коді без втручання людини на кожному кроці. Команда Атланти отримала головний приз у розмірі 4 мільйони доларів після завершення змагань на виставці DEF CON у Лас-Вегасі у серпні 2025 року. Виправлення важливе так само, як і виявлення. У багатьох організаціях переліки відомих вразливостей залишаються недоторканими, а агенти, які розробляють та тестують виправлення, зменшують їх кількість.

Агенти з пошуку помилок підзвітні тому, хто ними керує, тому слабкість, яку агент виявляє команді безпеки, є очевидною для зловмисника, який використовує аналогічний інструмент. Розробники програмного забезпечення

відреагували, перенісши перевірку коду штучного інтелекту на ранніший етап розробки, де виправлення коштує набагато менше, ніж виправлення, видане після того, як злочинці знаходять прогалину у відправленому продукті. Швидкість виявлення вирішує суперечку, що не дозволяє жодній компанії перерахувати власні виявлені системи, що відстають, ще до початку перегонів.

7. Розвідка загроз на основі штучного інтелекту покращує рішення аналітиків

KPMG зафіксувала 25% зростання операційної ефективності у своїй сфері розвідки загроз після того, як передала штучному інтелекту завдання зчитування та сортування необроблених даних зловмисників. Відділ розвідки відстежує активні хакерські команди, їхні інструменти, обрані цілі та недоліки, на які вони спираються, а потім перетворює результат на заблоковані домени та перетасовану чергу виправлень.

Обсяг створює вузьке місце. Урядові бюлетені, дослідження постачальників, трекери з відкритим кодом, пісочниці шкідливих програм, балачки на кримінальних форумах та власні записи про інциденти компанії – все це потрапляє в одну чергу, значна частина якої написана кількома мовами. Один аналітик очищає лише частину даних за один день, перш ніж надійде наступний.

Мовні моделі несуть більшу частину цього навантаження на читання. Вони вилучають індикатори компрометації, IP-адреси та хеші файлів, пов'язані з відомими операціями, документи, які ніхто не дочитав, перетворюють російськомовні та китайськомовні дописи на форумах на англійську мову та групують розрізнені статті, які описують одну кампанію. Вони також відображають спостережувану поведінку на MITRE ATT&CK, каталозі, який захисники використовують для позначення дій зловмисника.

Ті самі результати потрапляють до операційного центру як правило виявлення, на стіл керівника служби безпеки як оцінка ризиків, а в залу засідань – як кілька простих абзаців. Інструменти машинного навчання зважують інциденти один з одним, зіставляючи фрагменти коду, налаштування хостингу та звички операторів, щоб запропонувати посилання, які жоден

аналітик не намалював від руки.

Вигадка – це ціна швидкості. Модель неправильно інтерпретує контекст або вигадує деталь, а аналітичний продукт, що містить неперевірений індикатор, може підштовхнути команду блокувати нешкідливий трафік або переслідувати супротивника, якого ніколи не існувало. Аналітичний огляд має стояти між моделлю та будь-яким важливим рішенням.

Розвідувальні дані мають короткий термін придатності. Звістка про новий фішинговий домен або про активну експлуатацію вразливості приносить найбільшу користь протягом кількох годин після появи, і час, який заощаджує ШІ, припадає на цей період. Вигода спостерігається лише там, де перевірка людьми йде в ногу з часом, оскільки черга на перевірку, яка відстає, дає захисникам можливість повернути залишки, для розв'язання яких вони придбали ШІ.

Мутаційне тестування

Тестування мутацій, яке вводить невеликі модифікації коду для оцінки ефективності тестування, стикається з труднощами в машинному навчанні через недетермінований характер моделей та потребу в спеціалізованих операторах мутації [3]. Велика кількість згенерованих мутантів також робить тестування обчислювально дорогим. Мутанти можна класифікувати на мутанти, що завершують аварію, знищені та виживаючі, кожен з яких по-різному впливає на оптимізацію меж тверджень тесту. Виявлення ефективних мутантів має вирішальне значення для підвищення ефективності та результативності тестування.

Це дослідження досліджує тестування мутацій у машинному навчанні шляхом аналізу нестабільних тестових випадків з популярних проектів машинного навчання. Статистичний аналіз з використанням дивергенції, ексцесу та асиметрії JS досліджує поведінку мутантів для підвищення ефективності тестування.

Емпіричне дослідження виявляє такі результати:

1. Мутанти, що призводять до аварійного завершення, становлять 38,79%

усіх мутантів, що в основному спричинено значними відмінностями в намірі оператора, причому мутації параметрів методу найімовірніше призводять до збоїв виконання.

2. Вбиті мутанти становлять лише 4,39%, що вказує на низький рівень знищення мутацій. Числові та операторні мутації частіше виявляються в тестових випадках.

3. Вижили мутанти показують, що 41,62% не можуть оптимізувати межі тверджень тестових випадків, що призводить до збільшення витрат на тестування мутацій.

4. Виявлення та пропуск виконання мутантів, що призвели до аварійного завершення, та вбитих мутантів може значно зменшити обчислювальні витрати.

На завершення, багато мутантів є неефективними в оптимізації меж тверджень, а ідентифікація впливових мутантів на основі місць розташування мутацій та операторів може значно покращити ефективність тестування машинним навчанням.

Тестування машинного навчання

У галузі програмної інженерії твердження традиційно використовуються для перевірки заздалегідь визначених умов змінних або станів програми, забезпечуючи правильність поведінки програмного забезпечення [4]. Ці твердження оцінюються під час виконання, а невдалі твердження вказують на наявність помилок у програмі. При переході до галузі машинного навчання (ML) застосування тверджень зазнає фундаментальної трансформації, що в першу чергу відображає перевірку кількох вимірів, включаючи результати прогнозування моделі, показники продуктивності та якість даних [5]. У той час як твердження в традиційній програмній інженерії служать для забезпечення очікуваної поведінки системи, твердження в ML використовуються для перевірки відповідності поведінки моделі заздалегідь визначеним очікуванням. Однак через притаманну моделям машинного навчання невизначеність результати виконання тестування тверджень в ML часто є нестабільними.

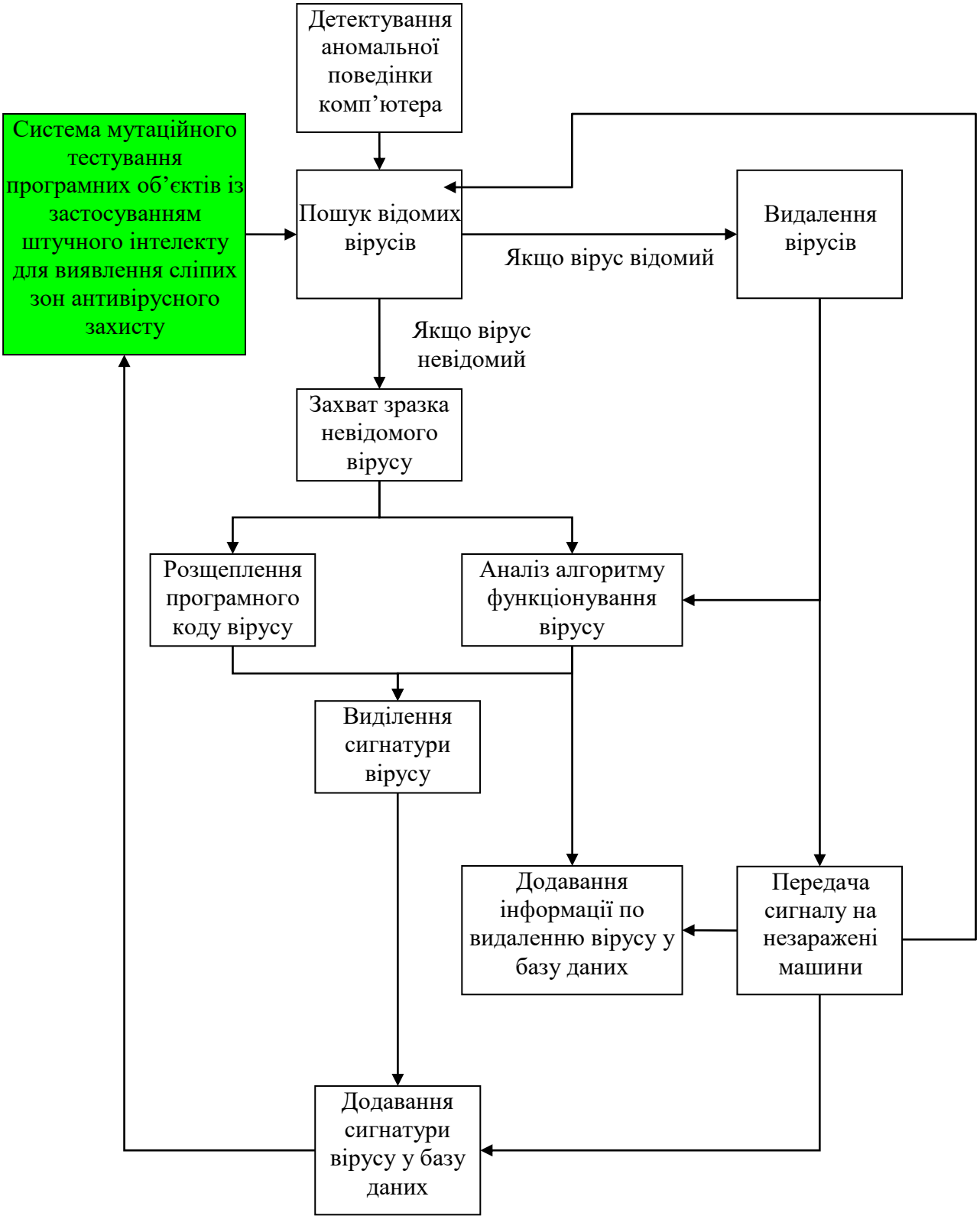

Рисунок 3.1 – Структурна схема системи

Тестування мутацій

Тестування мутацій оцінює здатність набору тестів програмного

забезпечення виявляти помилки шляхом введення невеликих помилок (мутацій) для створення мутантів, що імітують реальні помилки [6]. Ефективність тесту вимірюється його здатністю виявляти (знищувати) ці мутанти.

Хоча тестування мутацій у машинному навчанні (ML) широко застосовується в традиційному програмному забезпеченні, воно все ще перебуває на стадії зародження [7]. Невизначеність моделей ML та їхня залежність від даних створюють такі проблеми, як розробка ефективних операторів мутацій та обробка недетермінованих виходів [8].

3.3 Розробка функціональної схеми

Функціональна архітектура розробленого програмного комплексу складається з двох ключових макрокомпонентів:

1. Блок протидії детектуванню: призначений для моделювання методів обходу антивірусного контролю та генерації мутованих програмних об'єктів.

2. Блок антивірусного захисту: розроблений модуль, який виконує функції детектування, аналізу поведінки та захисту операційної системи.

3. Блок протидії детектуванню. Цей модуль об'єднує механізми маскування та модифікації виконавчих файлів для перевірки стійкості захисних систем і включає такі підкомпоненти:

– обфускація (динамічне дешифрування шелл-коду);

– пакувальники та криптори з реалізацією механізмів PE-ін'єкції;

– виявлення динамічного аналізу та засобів пісочниць;

– контроль привілеїв секцій та аналіз наявності нестандартних секцій у виконуваних файлах;

– перевірка сигнатур і контрольних сум файлів.

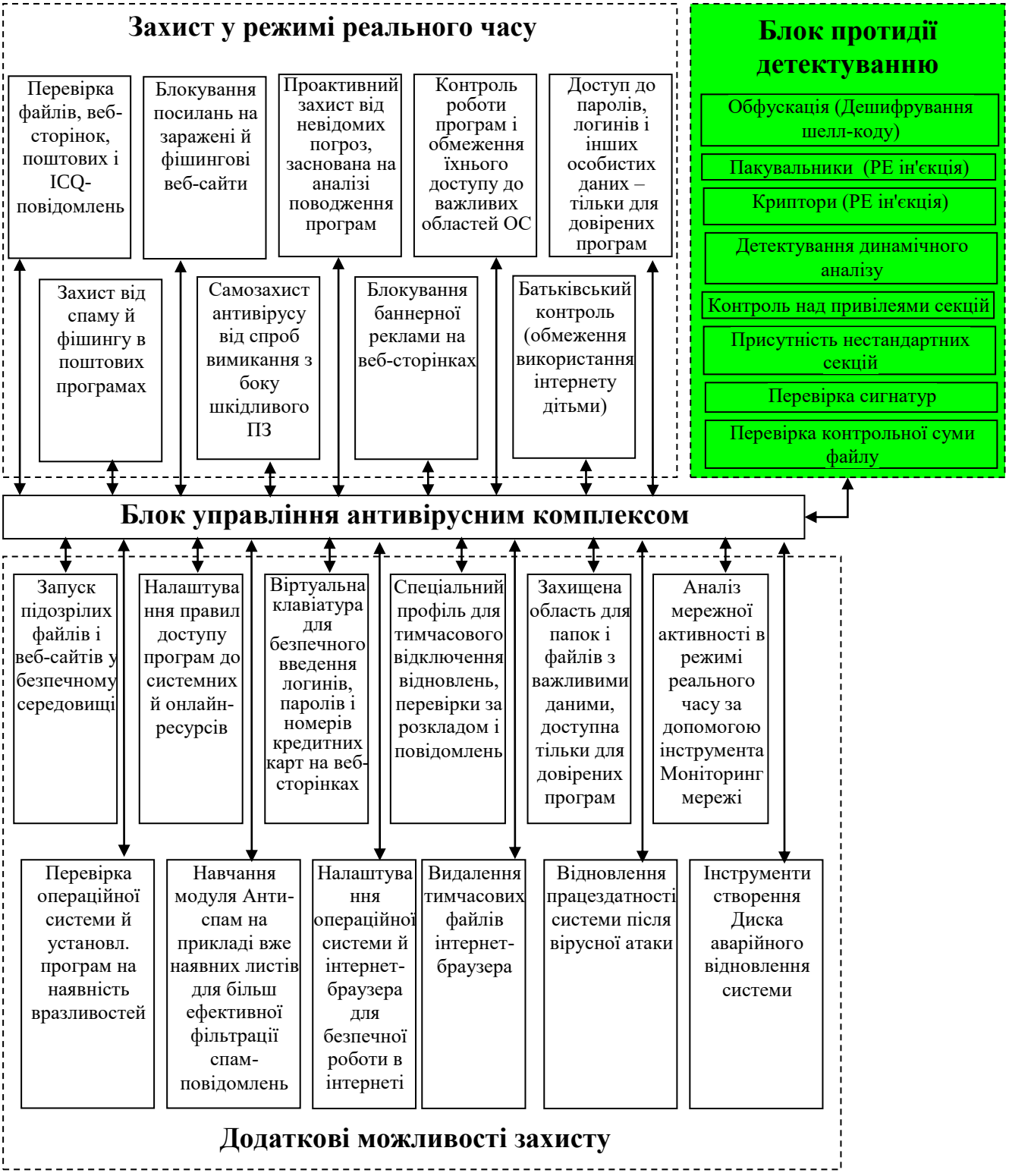

Рисунок 3.2 – Функціональна схема системи

4. Блок антивірусу. Розроблений антивірусний модуль забезпечує комплексний захист і стабільне функціонування комп'ютерної системи під час проведення тестових випробувань і мутаційного аналізу. До основних функцій та

сервісів безпеки належать:

– моніторинг у реальному часі: сканування файлів, вебсторінок та перевірка мережевої активності;

– проактивний захист нового покоління: евристичний аналіз поведінки програм для виявлення невідомих загроз і сигнатурних сліпих зон;

– система швидкого реагування: миттєве блокування шкідливих кодів та небезпечних посилань, що здійснюють перехоплення інформації;

– самозахист антивірусного ядра: запобігання несанкціонованому вимкненню або модифікації захисних служб з боку зловмисного ПЗ;

– додаткові інструменти безпеки: віртуальна клавіатура для захищеного введення конфіденційних даних, сканування ОС та встановленого софту на наявність уразливостей, а також модулі відновлення системи після деструктивного впливу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

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. У проєктній розробці застосовується модель графічного представлення потоків даних (Data Flow Diagram, DFD), яка відображає процеси передачі, трансформації та збереження інформації в інформаційній системі.

Діаграми взаємодії процесів є фундаментальним інструментом структурного проєктування для візуалізації алгоритмів обробки даних. Традиційно розробка розпочинається з побудови контекстної діаграми потоків даних, яка ілюструє загальну взаємодію розроблюваної системи мутаційного тестування та виявлення сліпих зон антивірусного захисту із зовнішнім середовищем (користувачем, тестовими середовищами та аналітичними модулями ШІ).

У подальшому ця модель деталізується шляхом поетапної декомпозиції процесів і потоків даних з метою всебічної демонстрації архітектури комплексу.



Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Алгоритмічне забезпечення системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту призначене для дослідження повноти антивірусного контролю, визначення недостатньо перевірених сценаріїв і виявлення потенційних сліпих зон захисних механізмів. Основним завданням є автоматизація формування контрольованих варіантів тестових програмних об’єктів, організація їхньої перевірки, накопичення результатів та інтелектуальний аналіз отриманих відомостей. Для цього передбачається поєднання методів мутаційного тестування, машинного навчання та статистичного оцінювання результатів. Особливістю запропонованого підходу є використання безпечних тестових артефактів і контрольованих сценаріїв, які дозволяють досліджувати працездатність антивірусної інфраструктури без необхідності створення шкідливого програмного забезпечення.

Загальний алгоритм функціонування програмного комплексу представлено на рисунку 4.1. На початковому етапі виконується підготовка тестового середовища, реєстрація контрольованих засобів захисту та завантаження набору початкових програмних об’єктів. Далі визначаються допустимі оператори мутації, за допомогою яких формуються контрольовані варіанти тестових артефактів. Після перевірки правильності отриманих об’єктів створюються завдання тестування. Антивірусний засіб або його тестовий інтерфейс обробляє відповідні об’єкти, після чого система накопичує результати. На основі отриманих даних виконуються класифікація спостережень, оцінювання покриття тестових сценаріїв та визначення напрямів подальшої перевірки.

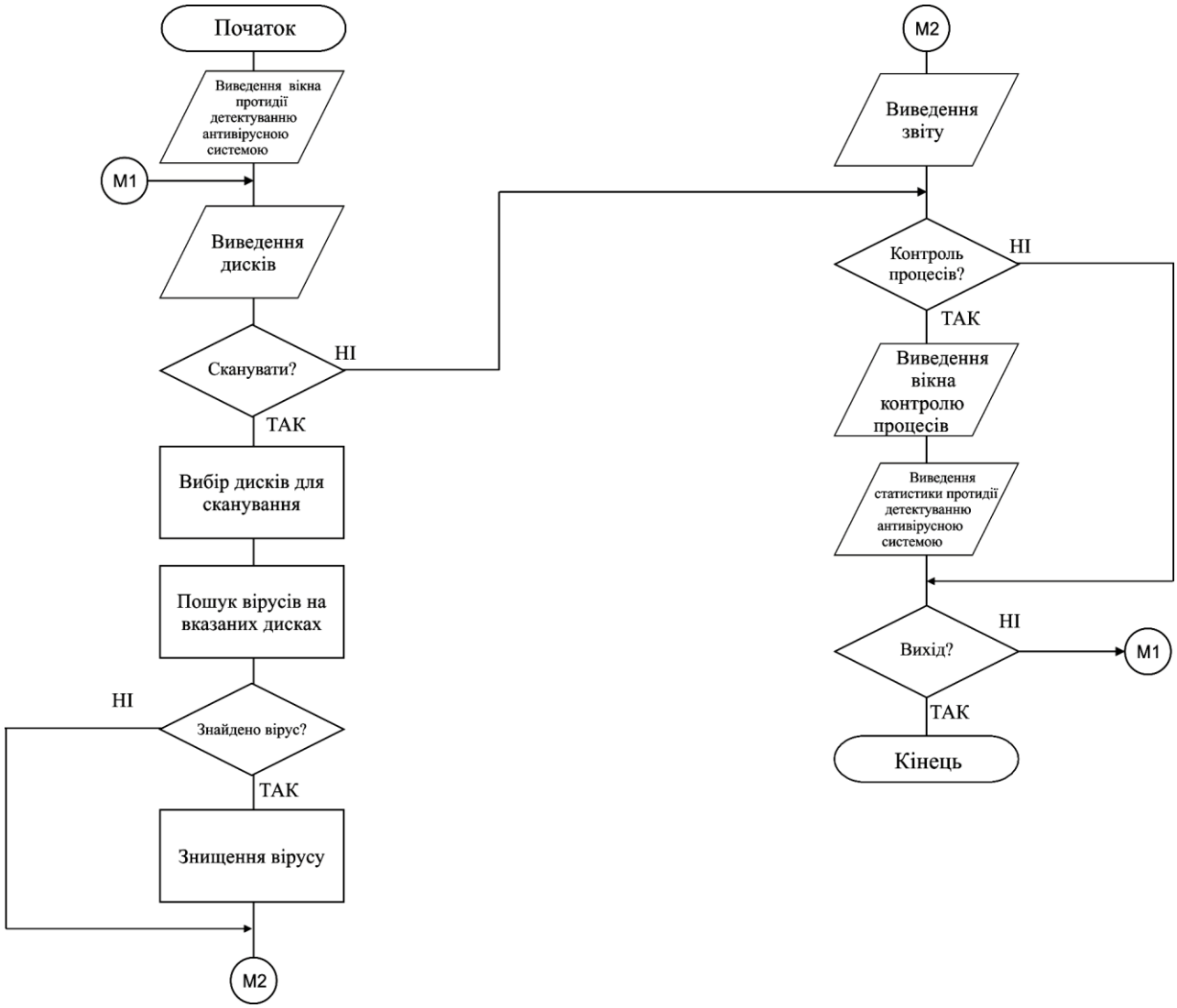

Рисунок 4.1 – Блок-схема основної програми

Для реалізації відповідних функцій доцільно передбачити модульну архітектуру програмного комплексу. Компонент підготовки тестів відповідає за реєстрацію вихідних об’єктів і правил їхнього перетворення. Модуль мутаційного тестування створює допустимі варіанти безпечних програмних артефактів. Планувальник визначає послідовність перевірок і контролює використання ресурсів. Компонент взаємодії з антивірусними засобами отримує результати відповідно до підтримуваних інтерфейсів. Аналітична підсистема використовує методи машинного навчання для групування сценаріїв, визначення недостатньо досліджених категорій і формування рекомендацій щодо розширення тестового

набору. Окремий компонент зберігання забезпечує реєстрацію конфігурацій експериментів, результатів перевірки та характеристик використаних моделей.

Важливим принципом системи є відокремлення тестових програмних об’єктів від засобів антивірусного захисту. Для дослідження можуть використовуватися спеціально створені нешкідливі програми, синтетичні журнали подій, контрольні файли та дозволені тестові сигнатури. Такі об’єкти дають змогу перевіряти окремі властивості процесу сканування, реєстрації подій та застосування політик безпеки. Водночас результати тестування синтетичних об’єктів не можна безпосередньо ототожнювати з ефективністю виявлення реального шкідливого програмного забезпечення. Для кожного тесту необхідно визначати, яку саме функцію захисту він перевіряє.

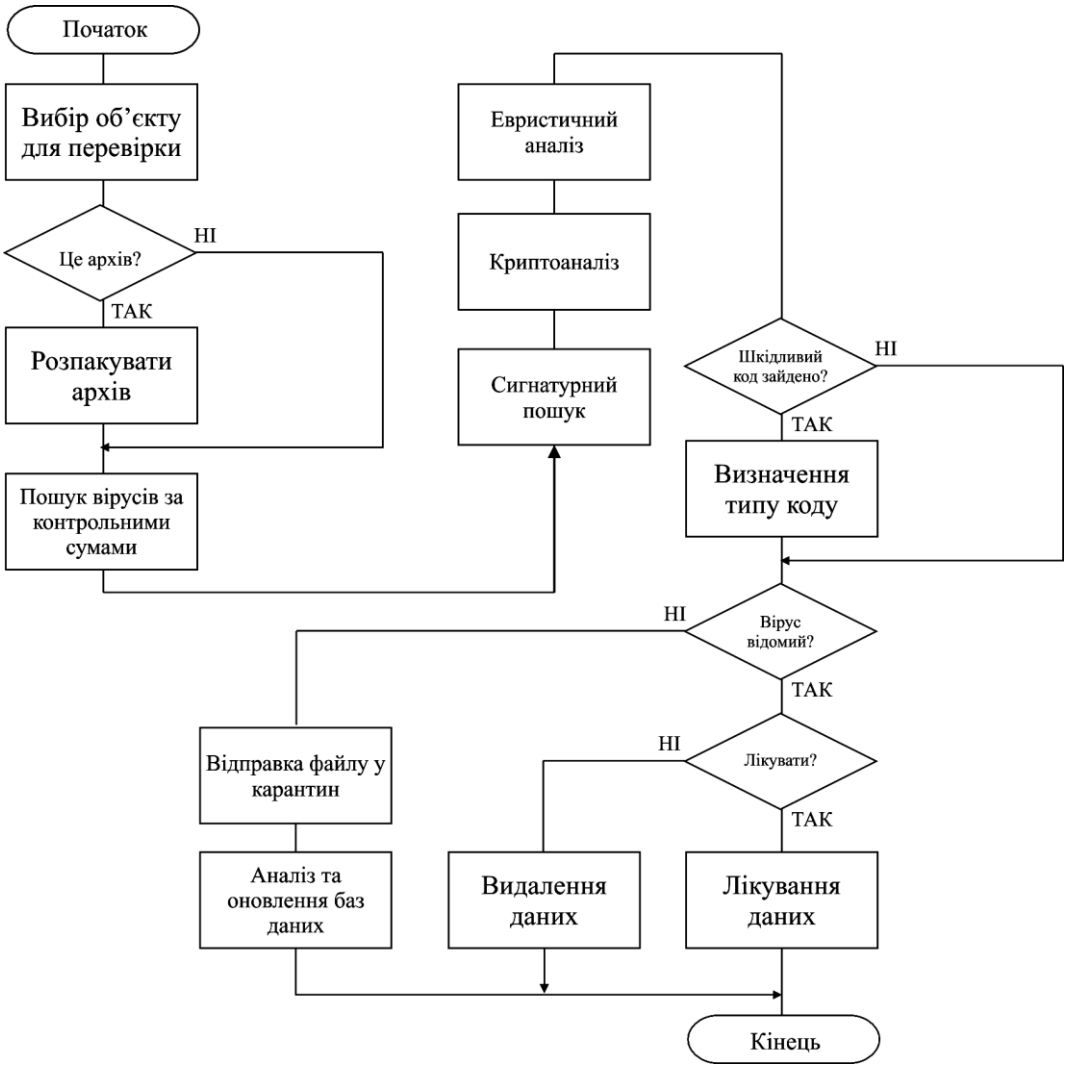

Рисунок 4.2 – Блок-схема роботи підпрограми

Окремим функціональним блоком є підпрограма інтелектуального аналізу результатів мутаційного тестування, алгоритм якої відображено на рисунку 4.2. Вона отримує характеристики виконаних тестів, відомості про використані оператори мутації та результати відповідних антивірусних перевірок. На першому етапі виконується нормалізація даних і визначення категорій тестових сценаріїв. Далі модель машинного навчання групує подібні результати та оцінює достатність покриття. Якщо для певної категорії бракує перевірених спостережень або виявлено невідповідність очікуваній реакції, формується повідомлення про необхідність додаткового тестування. Результат передається до модуля формування звітів.

Під час дослідження необхідно розрізняти класичне мутаційне тестування програмного забезпечення та оцінювання антивірусного захисту. У традиційному підході мутація передбачає контрольоване внесення змін до програми з метою перевірки здатності тестового набору виявляти помилки. Відповідні принципи систематизовано в роботі Y. Jia та M. Harman «An Analysis and Survey of the Development of Mutation Testing». У досліджуваній системі ці принципи застосовуються до формування різних контрольованих умов перевірки програмних об’єктів. При цьому відсутність антивірусного спрацювання не завжди означає порушення захисту. Такий висновок допустимий лише за наявності підтвердженої очікуваної реакції відповідного механізму.

Алгоритм ініціалізації середовища мутаційного тестування

Функціонування системи починається з перевірки готовності тестового середовища. Його основним призначенням є забезпечення відтворюваних умов проведення експериментів та ізоляція тестових процесів від продуктивної інфраструктури.

Вхідними даними алгоритму є конфігурація тестового стенда, перелік доступних програмних компонентів, параметри захисних засобів та правила виконання тестів. На першому етапі перевіряється наявність необхідних обчислювальних ресурсів. Далі визначається доступність компонентів зберігання, планувальника та аналітичної підсистеми.

Для виконання контрольованих перевірок можуть використовуватися ізольовані віртуальні машини або окремі тестові контейнери. Конкретний спосіб ізоляції залежить від функцій, які підлягають перевірці. Наприклад, для аналізу поведінки процесу контейнеризація не завжди забезпечує такий самий рівень відокремлення, як повноцінна віртуальна машина.

На наступному етапі перевіряється конфігурація мережі. Тестові компоненти повинні отримувати лише ті мережеві повноваження, які необхідні для виконання дозволених операцій. Для експериментів, що не потребують зовнішнього доступу, доцільно використовувати ізольований мережевий сегмент.

Після перевірки середовища формується запис про його поточний стан. Він може містити ідентифікатор тестового вузла, доступні ресурси, характеристики засобів захисту та версії необхідних програмних компонентів.

Результатом алгоритму є підтверджена готовність тестового середовища або перелік виявлених обмежень, які перешкоджають початку перевірок.

Алгоритм реєстрації програмних об’єктів

Для забезпечення відтворюваності експериментів кожний тестовий програмний об’єкт повинен мати визначений набір характеристик. До таких об’єктів можуть належати невеликі навчальні програми, тестові бібліотеки, конфігураційні файли та спеціально сформовані безпечні артефакти.

Вхідними даними алгоритму є вихідний об’єкт, його тип, призначення та відомості про дозволені процедури тестування. На першому етапі виконується перевірка формату. Далі визначається належність об’єкта до підтримуваної категорії.

Для кожного файлу може обчислюватися криптографічне контрольне значення SHA-256. Воно використовується для підтвердження тотожності артефакта під час повторних перевірок. Окремо реєструються розмір файлу, його логічна назва та походження.

Важливим етапом є перевірка допустимості використання об’єкта. Система повинна працювати з матеріалами, для яких визначено дозволений порядок

тестування. Доцільно виключати довільно отримані виконувані файли невідомого походження.

Для синтетичних об’єктів можуть зберігатися очікувані властивості. Наприклад, тестова програма може мати визначений результат виконання, який використовується для перевірки збереження функціональності після допустимого перетворення.

Результатом алгоритму є реєстр початкових програмних об’єктів із підтвердженими характеристиками та правилами подальшого використання.

Алгоритм формування набору операторів мутації

Мутаційне тестування передбачає використання визначених правил перетворення програмних об’єктів. Для коректного дослідження необхідно відокремлювати оператори, які змінюють поведінку програми, від перетворень, що зберігають її функціональний результат.

Вхідними даними алгоритму є тип програмного об’єкта, його структурні характеристики та перелік дозволених операторів. На початковому етапі визначається сумісність кожного оператора з відповідним форматом.

Для навчальних програм можуть розглядатися контрольовані зміни вихідного коду, які моделюють типові помилки програмування. Вони можуть використовуватися для перевірки якості функціональних тестів. Такі мутації не призначені для дослідження обходу антивірусних механізмів.

Окрему категорію становлять нейтральні перетворення безпечних тестових об’єктів. Вони можуть включати зміни допоміжних описових полів, форматування тестових даних або інших властивостей, які не впливають на визначений функціональний результат. Для кожного перетворення необхідно встановити очікувану незмінність або зміну відповідних характеристик.

Для перевірки антивірусної інфраструктури можуть використовуватися стандартні контрольні файли. Зокрема, EICAR Anti-Virus Test File призначений для перевірки реакції антивірусного програмного забезпечення без використання реального вірусу. Такий файл доцільно застосовувати відповідно до офіційного

визначення тестового зразка. Довільна зміна його вмісту не зберігає автоматично очікувані властивості контрольної сигнатури.

Після класифікації операторів формується їхній реєстр. Для кожного запису визначаються область застосування, очікуваний результат і процедура перевірки коректності.

Результатом алгоритму є набір дозволених операторів мутації, який може використовуватися для формування контрольованих сценаріїв тестування.

Алгоритм генерації контрольованих мутантів

Після визначення операторів система переходить до формування варіантів тестових об’єктів. Метою цього етапу є отримання набору артефактів із відомими та документованими відмінностями від початкового представлення.

Вхідними даними є зареєстрований об’єкт, вибраний оператор і параметри відповідного тестового сценарію. На першому етапі перевіряється допустимість застосування оператора.

Після цього створюється незалежна копія вихідного об’єкта. Усі зміни виконуються лише над копією. Початковий артефакт повинен залишатися незмінним для забезпечення можливості повторного дослідження.

Далі застосовується визначене перетворення. Для синтетичних об’єктів воно може стосуватися структури тестових даних, незначущих атрибутів або спеціально передбачених властивостей навчального програмного об’єкта.

Після перетворення перевіряється правильність отриманого результату. Якщо об’єкт має визначену функціональність, необхідно підтвердити, що він відповідає умовам конкретного сценарію. Якщо перетворення змінює очікувану поведінку, це повинно відображатися в його описі.

Для кожного мутанта створюється окремий ідентифікатор. У метаданих зберігаються зв’язок із початковим об’єктом, застосований оператор, параметри перетворення та контрольне значення результату.

Результатом алгоритму є набір контрольованих тестових артефактів, придатних для подальшого оцінювання.

Алгоритм перевірки коректності мутантів

Не кожний результат перетворення можна використовувати для подальшого тестування. Частина змін може призводити до пошкодження формату, порушення очікуваної функціональності або утворення артефакта, який не відповідає початковій меті експерименту.

Вхідними даними алгоритму є сформований мутант, характеристики вихідного об’єкта та визначені правила перевірки.

На першому етапі контролюється відповідність формату. Для текстових файлів перевіряється коректність кодування. Для структурованих даних може застосовуватися відповідний синтаксичний аналізатор.

Якщо мутація стосується навчальної програми, виконується відповідний набір функціональних тестів. Він дозволяє визначити, чи збереглася очікувана поведінка або чи виникла передбачена тестова відмінність.

Окремо перевіряється відсутність непередбачених побічних ефектів. Тестовий об’єкт не повинен отримувати нові мережеві повноваження, доступ до конфіденційних ресурсів або інші можливості, які не визначені сценарієм.

За невідповідності результат позначається як непридатний для відповідної перевірки. Такий мутант не повинен включатися до розрахунку ефективності антивірусного контролю.

Результатом алгоритму є перелік валідованих мутантів із визначеними властивостями та допустимими способами використання.

Алгоритм усунення повторюваних та еквівалентних об’єктів

Під час генерації декілька операторів можуть створювати однакові або функціонально еквівалентні результати. Надмірна кількість таких об’єктів збільшує тривалість дослідження без пропорційного розширення тестового покриття.

На початковому етапі для кожного сформованого артефакта обчислюється криптографічне контрольне значення. Однакові значення для тотожного вмісту дозволяють виявляти точні дублікати.

Окремо аналізуються функціонально еквівалентні програмні об’єкти. Вони можуть відрізнятися структурою, але забезпечувати однакові результати для визначеного набору вхідних даних.

Необхідно враховувати, що повне доведення еквівалентності довільних програм у загальному випадку є складною проблемою. Тому автоматизоване тестування може підтвердити лише відповідність у межах перевірених сценаріїв, якщо не використовуються спеціальні методи формального доведення.

Після аналізу формуються групи однакових і потенційно еквівалентних артефактів. Для кожної групи зберігаються відомості про застосовані оператори.

Результатом алгоритму є скорочений набір тестових об’єктів, у якому зменшено кількість повторних перевірок без втрати документованого різноманіття сценаріїв.

Алгоритм створення завдань антивірусного тестування

Після підготовки валідованих артефактів необхідно визначити порядок їхньої перевірки. Завдання повинно містити достатньо інформації для відтворення експерименту та правильного зіставлення результатів.

Вхідними даними є тестовий артефакт, цільовий захисний засіб і конфігурація стенда. На першому етапі перевіряється сумісність відповідного тестового сценарію з можливостями антивірусного продукту.

Далі визначається тип перевірки. Це може бути контроль сканування файлу, перевірка оброблення стандартного тестового зразка або оцінювання реєстрації синтетичної події інформаційної безпеки.

Для кожного завдання створюється ідентифікатор, який дозволяє зіставляти його виконання з конкретним артефактом. Окремо фіксуються версія антивірусного засобу, поточний стан його оновлень та активна політика.

Після формування завдання передається до планувальника. До початку виконання перевіряється готовність тестового середовища та відсутність несумісних паралельних операцій.

Результатом алгоритму є формалізоване завдання антивірусної перевірки з визначеними вхідними умовами та очікуваним способом отримання результату.

Алгоритм планування та керування тестовими завданнями

Виконання великої кількості перевірок потребує контролю обчислювального навантаження та послідовності використання тестових стендів.

Вхідними даними алгоритму є черга сформованих завдань, характеристики доступних середовищ і встановлені обмеження виконання.

На першому етапі визначається стан кожного тестового вузла. Якщо середовище зайняте або перебуває на етапі відновлення, нове завдання до нього не передається.

Далі перевіряється пріоритет завдання. Він може визначатися необхідністю завершення певної групи тестів, обмеженнями часу або попередньо встановленим планом дослідження.

За наявності декількох придатних вузлів завдання може передаватися до менш завантаженого. При цьому необхідно враховувати, що однакові артефакти, перевірені за різних конфігурацій, не завжди формують безпосередньо порівнянні результати.

Після призначення завдання змінює стан з очікування на виконання. У випадку перевищення допустимого часу формується повідомлення про помилку або невизначений результат.

Результатом алгоритму є керований процес виконання перевірок із визначеними станами та журналом переходів.

Алгоритм взаємодії з антивірусним засобом

Спосіб отримання результатів залежить від можливостей конкретного програмного продукту. Антивірусні системи можуть надавати інтерфейси командного рядка, програмні API, журнали подій або централізовані засоби адміністрування.

Вхідними даними алгоритму є завдання тестування та параметри відповідного адаптера. На першому етапі перевіряється доступність захисного засобу.

Далі виконується дозволена операція перевірки тестового артефакта. Для цього необхідно використовувати документований інтерфейс, який не потребує вимкнення захисних механізмів або втручання в роботу антивірусного ядра.

Після завершення перевірки система отримує доступний результат. Він може містити повідомлення про виявлення, назву категорії загрози, стан сканування або відомості про застосовану політику.

Окремо обробляються випадки, коли захисний продукт не надає явного результату. Відсутність повідомлення в журналі не завжди означає, що перевірка завершилася без виявлення. Вона може бути наслідком помилки інтеграції, неповного журналювання або невиконаного завдання.

Тому алгоритм повинен перевіряти наявність підтвердження завершення та відповідність результату конкретному ідентифікатору тесту.

Результатом є структурований запис про реакцію антивірусного засобу або інформація про неможливість достовірного оцінювання.

Алгоритм перевірки працездатності захисного механізму

Перед дослідженням складних тестових сценаріїв необхідно підтвердити, що захисний компонент працює та правильно передає результати.

Для цього використовується базова контрольна перевірка. Вона повинна мати попередньо визначену очікувану реакцію.

Одним із можливих засобів є стандартний EICAR Anti-Virus Test File, розроблений EICAR та CARO. Його призначення полягає у перевірці реакції антивірусного програмного забезпечення без використання реального шкідливого коду.

На початковому етапі перевіряється, чи підтримує відповідний антивірусний продукт обраний контрольний сценарій. Далі виконується перевірка відповідно до офіційного призначення тестового файлу.

Після отримання результату визначається правильність реєстрації події. Якщо очікуване спрацювання відсутнє, необхідно перевірити конфігурацію продукту, активність захисту та працездатність інтеграційного компонента.

Слід враховувати, що успішне виявлення контрольного файлу підтверджує лише роботу відповідного механізму в межах тестового сценарію. Воно не доводить здатність продукту виявляти всі категорії шкідливого програмного забезпечення.

Результатом алгоритму є підтвердження базової працездатності контрольованого механізму або повідомлення про необхідність діагностики.

Алгоритм нормалізації результатів тестування

Різні захисні продукти можуть формувати неоднакові повідомлення. Для спільного аналізу необхідно перетворювати їх до єдиного внутрішнього формату.

Вхідними даними є службові відповіді антивірусних засобів, журнали перевірок і характеристики виконаних завдань.

На першому етапі визначається тип повідомлення. Далі виділяються ідентифікатор тесту, час перевірки, стан завершення, категорія результату та доступні службові характеристики.

Для внутрішнього аналізу доцільно використовувати окремі стани підтвердженого виявлення, підтвердженої відсутності виявлення, технічної помилки та невизначеного результату.

Важливо не об’єднувати технічні помилки з негативними результатами сканування. Якщо засіб захисту не зміг відкрити файл або завершити перевірку, це не підтверджує його здатності чи нездатності розпізнавати відповідний об’єкт.

Окремо зберігається початкове повідомлення продукту. Це дозволяє уточнювати результати після зміни правил нормалізації.

Результатом алгоритму є уніфікований набір записів, придатний для статистичного аналізу та машинного навчання.

Алгоритм формування векторів ознак тестових сценаріїв

Для застосування методів штучного інтелекту необхідно перетворити результати експериментів на структуровані числові характеристики.

Вхідними даними є метадані тестових артефактів, відомості про оператори мутації, конфігурації стендів і нормалізовані результати перевірок.

На першому етапі визначається перелік ознак. До них можуть належати категорія програмного об’єкта, його розмір, тип контрольованого перетворення та характеристики застосованого тестового сценарію.

Окремо можуть використовуватися параметри захисного середовища. Наприклад, тип операції сканування, доступність журналювання та належність тесту до визначеної політики захисту.

Категоріальні характеристики перетворюються до числового представлення. Для цього можуть застосовуватися One-Hot Encoding або інші методи кодування.

Числові показники приводяться до узгодженого масштабу. Параметри відповідного перетворення повинні визначатися на навчальному наборі та зберігатися разом із моделлю.

Під час формування ознак необхідно виключати характеристики, які стають відомими лише після завершення тесту, якщо модель призначена для попереднього оцінювання. Інакше виникає витік цільової інформації.

Результатом алгоритму є вектор характеристик тестового сценарію, який може використовуватися для кластеризації та аналізу покриття.

Алгоритм кластеризації тестових сценаріїв

Кластеризація дозволяє визначати групи перевірок із подібними характеристиками. Це необхідно для аналізу різноманітності тестового набору та виявлення категорій, які представлені недостатньою кількістю спостережень.

Вхідними даними алгоритму є нормалізовані вектори ознак тестових сценаріїв. На початковому етапі перевіряється їхня структура та відсутність некоректних значень.

Для групування може застосовуватися K-means. Він дозволяє розподіляти спостереження між визначеною кількістю груп за відстанню до відповідних центрів.

Як альтернативний метод може використовуватися HDBSCAN. Його доцільно розглядати для наборів із неоднорідною щільністю, у яких частина тестових сценаріїв утворює невеликі спеціалізовані групи.

Після кластеризації визначаються характеристики кожної групи. До них належать кількість тестів, розподіл типів об’єктів і представленість відповідних операторів мутації.

Окремо аналізуються групи з недостатньою кількістю спостережень. Їхня наявність може свідчити про необхідність розширення тестового набору.

Слід враховувати, що належність до одного кластера не доводить еквівалентності тестів. Алгоритм визначає подібність за вибраними ознаками, а не повну однаковість їхнього функціонального змісту.

Результатом алгоритму є структура тестових сценаріїв, придатна для аналізу покриття та планування додаткових перевірок.

Алгоритм оцінювання покриття антивірусних перевірок

Для визначення повноти тестування необхідно встановити, які захисні функції та умови роботи вже були перевірені.

Вхідними даними алгоритму є перелік запланованих сценаріїв, результати виконаних завдань та правила очікуваної поведінки захисного засобу.

На першому етапі формується матриця покриття. Її вимірами можуть бути тип тестового об’єкта, категорія перевірки, конфігурація захисту та очікуваний результат.

Далі визначається наявність завершених перевірок для кожної комбінації. Якщо тест не виконувався або його результат є невизначеним, відповідна категорія не може вважатися підтверджено перевіреною.

Окремо аналізується відповідність фактичного результату очікуваній реакції. Наприклад, якщо конфігурація передбачає реєстрацію визначеної контрольної події, але вона не з’явилася в журналі, необхідно встановити причину невідповідності.

При цьому слід відрізняти неперевірені сценарії від підтверджених дефектів. Відсутність тестових даних свідчить про недостатнє покриття, але не про фактичну неспроможність захисту.

Результатом алгоритму є матриця перевірених, неперевірених і проблемних сценаріїв.

Алгоритм визначення потенційних сліпих зон захисту

Сліпа зона антивірусного захисту в межах досліджуваної системи розглядається як недостатньо контрольована або непідтверджено оброблена категорія подій, для якої визначено відповідну вимогу до захисного механізму.

Вхідними даними є матриця покриття, очікувані результати тестових сценаріїв та фактичні повідомлення антивірусних засобів.

На першому етапі визначаються перевірки, які мають підтверджений очікуваний результат. Далі аналізуються випадки невідповідності.

Якщо захисний механізм повинен сформувати контрольне сповіщення, але за коректно виконаного тесту його не отримано, формується кандидат на сліпу зону.

Наступним етапом є перевірка достовірності такого висновку. Необхідно підтвердити правильність тестового об’єкта, налаштування захисту, завершення операції сканування та працездатність журналювання.

Якщо причина полягає в помилці інтеграції або неповному отриманні телеметрії, відповідний випадок класифікується як проблема інструментування.

Якщо невідповідність відтворюється за правильних умов, вона може бути зареєстрована як підтверджений недолік конкретного механізму або політики в межах виконаного сценарію.

Результатом алгоритму є перелік потенційних і підтверджених проблем контролю з визначеними умовами їхнього відтворення.

Алгоритм інтелектуального визначення пріоритетів тестування

За великої кількості можливих сценаріїв доцільно автоматизувати вибір напрямів подальших перевірок. Для цього може використовуватися модель машинного навчання, яка оцінює недостатність покриття та інформативність тестових категорій.

Вхідними даними є характеристики попередніх тестів, кількість спостережень у відповідних групах та результати перевірки їхньої достовірності.

На початковому етапі визначається представленість кожної категорії. Якщо група містить невелику кількість завершених перевірок, вона може отримувати підвищений пріоритет.

Далі враховується різноманітність тестів. Повторне виконання тотожних сценаріїв не завжди забезпечує додаткову інформацію. Тому перевага може надаватися дозволеним контрольним перевіркам, які розширюють уже сформовану матрицю покриття.

Окремо аналізується кількість невизначених результатів. Значна їхня частка може свідчити про проблеми тестового середовища, які потрібно усунути до подальшого дослідження.

Для автоматизованого ранжування можуть використовуватися регресійні моделі або алгоритми класифікації. Їхнє навчання здійснюється на історичних даних про інформативність тестових сценаріїв.

Завданням моделі є визначення пріоритетів безпечних перевірок, а не створення програмних об’єктів із метою уникнення антивірусного виявлення.

Результатом алгоритму є впорядкований перелік сценаріїв для подальшого тестування.

Алгоритм навчання моделі штучного інтелекту

Для використання машинного навчання необхідно сформувати навчальний набір, який містить структуровані характеристики тестів і перевірені оцінки їхньої інформативності.

Вхідними даними є історичні результати експериментів, категорії сценаріїв, дані про повноту покриття та висновки щодо якості отриманих спостережень.

На першому етапі виконується очищення набору. Видаляються технічно некоректні записи та дублікати, які не мають окремого експериментального значення.

Далі визначається цільова характеристика. Для задачі пріоритизації це може бути категорія необхідності додаткового тестування або оцінка інформаційної цінності сценарію.

Як можливі методи розглядаються Random Forest і Gradient Boosting. Вони дозволяють оцінювати залежності між різними характеристиками тестів без необхідності задавати всі правила вручну.

Під час формування навчальної та тестової вибірок необхідно враховувати спорідненість мутантів. Варіанти одного початкового об’єкта можуть мати дуже подібні характеристики. Якщо вони одночасно потрапляють до навчальної та тестової частин, оцінка якості моделі може бути завищеною.

Тому доцільно виконувати груповий поділ за ідентифікатором початкового об’єкта або сімейством тестових сценаріїв.

Після навчання модель перевіряється на окремому наборі. Результати порівнюються з простими евристичними правилами пріоритизації.

Результатом алгоритму є підготовлена модель, придатна для визначення напрямів розширення безпечного тестового набору.

Алгоритм оцінювання достовірності висновків

Результати машинного навчання можуть містити помилки, особливо за недостатньої кількості спостережень або суттєвої зміни умов тестування.

Для зменшення ризику некоректних висновків передбачається окремий механізм контролю достовірності.

Вхідними даними є результат моделі, характеристика відповідної категорії та статистика навчального набору.

На першому етапі перевіряється, чи відповідає новий сценарій області даних, на якій навчалася модель. Значні відмінності можуть свідчити про ненадійність прогнозу.

Далі визначається кількість перевірених спостережень у відповідній групі. Якщо даних недостатньо, результат не повинен автоматично використовуватися для остаточного висновку.

Окремо аналізується узгодженість між моделлю та формальними правилами перевірки. Якщо модель позначає сценарій як проблемний, але фактичні дані не підтверджують порушення, необхідно додатково перевірити причину розбіжності.

Результатом алгоритму є оцінка можливості використання висновку для подальшого аналізу або рекомендація щодо отримання додаткових даних.

Приклад програмної реалізації оцінювання тестових результатів

Для пояснення принципу оброблення результатів мутаційного тестування може використовуватися функція мовою Python. Вона аналізує записи виконаних перевірок, визначає частку підтверджених сценаріїв і формує перелік випадків, які потребують додаткового дослідження. Наведений фрагмент є ілюстративним прикладом. Він не виконує модифікації програмних файлів, не взаємодіє безпосередньо з антивірусними механізмами та не є підтвердженням фактичної програмної реалізації комплексу.

```
from dataclasses import dataclass
from collections import defaultdict


@dataclass(frozen=True)
class TestResult:
    # Ідентифікатор контрольованого сценарію
    test_id: str

    # Категорія перевірки захисного механізму
    category: str

    # Очікувана реакція за правилами тесту
    expected_alert: bool

    # Фактичний результат, якщо тест завершено
    observed_alert: bool | None

    # Ознака технічно коректного завершення тесту
    completed: bool


def evaluate_test_coverage(
    results: list[TestResult]
) -> dict:
    """Аналіз повноти та коректності контрольованих перевірок."""

    categories = defaultdict(
        lambda: {
            "completed": 0,
            "matched": 0,
            "mismatched": 0,
            "inconclusive": 0
        }
    )

    review_cases = []

    for item in results:
        # Отримуємо статистику відповідної категорії
        stats = categories[item.category]
```

```
        # Незавершені перевірки не вважаємо негативними
        if not item.completed or item.observed_alert is None:
            stats["inconclusive"] += 1
            continue

        stats["completed"] += 1

        # Порівнюємо фактичну реакцію з очікуваною
        if item.observed_alert == item.expected_alert:
            stats["matched"] += 1
        else:
            stats["mismatched"] += 1

            # Передаємо розбіжність на додаткову перевірку
            review_cases.append({
                "test_id": item.test_id,
                "category": item.category,
                "reason": "Невідповідність очікуваній реакції"
            })

    summary = {}

    for category, stats in categories.items():
        completed = stats["completed"]

        # Обчислюємо відповідність лише для завершених тестів
        agreement = (
            stats["matched"] / completed
            if completed > 0
            else None
        )

        summary[category] = {
            **stats,
            "agreement_rate": agreement
        }

    return {
        "categories": summary,
        "cases_for_review": review_cases
    }
```

У наведеному прикладі кожен результат містить категорію тесту, очікувану реакцію захисного механізму та фактичний стан перевірки. Перед обчисленням показників визначається, чи завершилося завдання коректно. Якщо результат відсутній, запис відноситься до невизначених і не використовується як підтвердження виявлення або невиявлення.

Для завершених перевірок виконується порівняння очікуваного й фактичного результатів. Випадки невідповідності передаються до окремого переліку. Вони не оголошуються автоматично підтвердженими сліпими зонами, оскільки потребують перевірки правильності конфігурації та достовірності контрольного сценарію.

Обчислений показник agreement_rate характеризує частку перевірок, результати яких відповідають очікуваній поведінці. Він не є універсальною оцінкою ефективності антивірусного продукту. Для такого висновку необхідний репрезентативний набір сценаріїв та окрема методика оцінювання.

Алгоритм оцінювання ефективності тестового набору

Для аналізу якості дослідження необхідно використовувати декілька взаємодоповнювальних показників. Вони повинні характеризувати повноту тестування, достовірність результатів і обсяг виявлених невідповідностей.

На першому етапі визначається кількість сформованих і валідованих мутантів. Окремо враховуються непридатні артефакти, точні дублікати та результати з невизначеним станом.

Далі обчислюється частка виконаних перевірок відносно запланованих. Цей показник відображає повноту виконання тестової програми.

Окремо визначається частка результатів, які відповідають очікуваній реакції. Для сценаріїв із підтвердженою вимогою до спрацювання можуть розглядатися випадки невиявлення. Для безпечних об’єктів, щодо яких спрацювання не очікується, аналізуються помилкові позитивні результати.

Класичний mutation score визначає частку нееквівалентних мутантів, які були виявлені відповідним набором функціональних тестів. Його не слід безпосередньо переносити на оцінювання антивірусних продуктів без визначення того, що саме означає виявлення мутації.

Для досліджуваної системи доцільно окремо оцінювати покриття сценаріїв, частку коректно завершених перевірок і кількість підтверджених невідповідностей.

Результатом алгоритму є набір показників, які дозволяють аналізувати якість тестування без змішування різних за змістом характеристик.

Алгоритм повторної перевірки підозрілих результатів

Одинична невідповідність не завжди дозволяє встановити стійкий недолік захисного механізму. На результат можуть впливати тимчасові помилки, стан оновлень або особливості конфігурації тестового середовища.

Вхідними даними алгоритму є перелік проблемних перевірок і відомості про умови їхнього виконання.

На першому етапі визначається необхідність повторного тестування. Далі перевіряється можливість відновлення початкової конфігурації стенда.

Після відновлення середовища відповідний контрольний сценарій виконується повторно. Для кожної перевірки зберігаються незалежні відомості про її завершення та отриманий результат.

Якщо невідповідність не відтворюється, випадок позначається як нестабільний. Після цього можуть аналізуватися технічні характеристики середовища та журнали виконання.

Якщо невідповідність повторюється, формується запис для поглибленого аналізу. Він повинен містити лише ті висновки, які підтверджені виконаними перевірками.

Результатом алгоритму є уточнений статус проблемного сценарію, який використовується під час формування звітів.

Алгоритм ізоляції та відновлення тестового середовища

Для забезпечення відтворюваності необхідно контролювати стан середовища між окремими перевірками. Антивірусні засоби можуть змінювати локальні журнали, створювати записи про попередні події або переміщувати контрольні файли відповідно до своїх політик.

Тому після виконання завдання передбачається процедура очищення та відновлення тестового вузла.

На початковому етапі перевіряється завершення всіх операцій. Далі результати передаються до централізованого сховища.

Після підтвердження збереження даних тестове середовище може повертатися до попереднього контрольного стану. Для віртуальних машин можливим способом є використання погоджених знімків.

Водночас відновлення знімка не повинно порушувати ліцензійні умови, механізми керування захистом або політики оновлення програмного продукту.

Після відновлення перевіряється готовність середовища до нового завдання. Якщо початковий стан не підтверджено, подальше тестування призупиняється.

Результатом алгоритму є підготовлений тестовий вузол із контрольованою конфігурацією.

Алгоритм централізованого зберігання результатів

Для подальшого інтелектуального аналізу необхідно забезпечити накопичення результатів у структурованому вигляді. При цьому важливо зберігати зв’язок між вихідними об’єктами, мутантами, завданнями та відповідями захисних засобів.

Вхідними даними алгоритму є записи перевірок, метадані артефактів та службова інформація про тестові середовища.

На першому етапі виконується перевірка коректності результату. Далі визначається відповідний запис завдання та встановлюється зв’язок із конкретним мутантом.

Для зберігання структурованих відомостей може використовуватися реляційна система керування базами даних, зокрема PostgreSQL. Великі службові файли та розширені журнали можуть розміщуватися в окремому об’єктному сховищі.

Необхідно забезпечити незмінність первинних результатів після завершення експерименту. Уточнення класифікації повинно реєструватися окремо від вихідного повідомлення.

Для кожного запису доцільно зберігати версію правил аналізу та модель машинного навчання, яка використовувалася під час його оброблення.

Результатом алгоритму є історія експериментів, придатна для повторного аналізу та перевірки відтворюваності.

Алгоритм формування аналітичних звітів

Завершальним етапом оброблення результатів є формування звітної інформації. Її призначення полягає в представленні структури проведених перевірок і визначенні напрямів удосконалення контролю.

Вхідними даними є нормалізовані результати, показники покриття, кластерна структура та перелік підтверджених невідповідностей.

На першому етапі визначається загальна кількість запланованих і завершених тестів. Далі формуються відомості про використані категорії об’єктів та оператори мутації.

Окремо представляється структура результатів за контрольованими захисними засобами. Порівняння допустиме лише за однакових або документовано зіставних умов тестування.

Для кожної потенційної сліпої зони необхідно наводити характеристики контрольного сценарію, очікувану реакцію, фактичний результат та статус повторної перевірки.

Додатково може формуватися перелік рекомендацій щодо вдосконалення тестової програми. До нього належать розширення недостатньо представлених категорій, уточнення правил журналювання та повторна перевірка технічно невизначених результатів.

Результатом алгоритму є структурований звіт, який дозволяє оцінювати повноту проведеного дослідження.

Особливості програмної реалізації системи

Програмну реалізацію доцільно організувати з функціональним розділенням тестового конвеєра та аналітичної підсистеми. Такий підхід дозволяє змінювати алгоритми машинного навчання без втручання в механізми виконання контрольованих перевірок.

Для реалізації компонентів оброблення даних як можливе технологічне рішення може використовуватися Python. Він забезпечує засоби роботи зі структурованими повідомленнями, автоматизації тестових завдань та підготовки даних машинного навчання.

Для кластеризації й побудови моделей пріоритизації можуть застосовуватися бібліотеки scikit-learn. Оброблення результатів і формування агрегованих характеристик може виконуватися засобами pandas.

Для ізоляції тестових середовищ можуть застосовуватися віртуальні машини. Конкретний механізм визначається типом антивірусного засобу та вимогами до відтворюваності експериментів.

Важливо забезпечити контроль доступу до керувальних інтерфейсів. Компонент машинного навчання повинен отримувати підготовлені метадані й результати тестування, але не потребує необмеженого доступу до виконуваних об’єктів.

Окремо необхідно передбачити версіонування алгоритмів. Зміна правил нормалізації або складу ознак може впливати на результати кластеризації та пріоритизації. Тому відповідні конфігурації повинні зберігатися разом із результатами експериментів.

Алгоритм оброблення помилок та відмов компонентів

У процесі роботи можливі пошкодження тестових артефактів, недоступність середовища, помилки взаємодії з антивірусним продуктом і порушення роботи аналітичного компонента.

На першому етапі алгоритм визначає категорію помилки. Для технічних відмов можуть застосовуватися повторні спроби з обмеженою кількістю виконань.

Якщо тестовий артефакт не пройшов перевірку коректності, він не передається до антивірусного компонента. Відповідна причина зберігається в журналі.

У разі відмови тестового вузла завдання позначається як незавершене. Після відновлення середовища воно може виконуватися повторно.

Якщо компонент машинного навчання недоступний, система може продовжувати накопичення результатів без інтелектуальної пріоритизації. Такий режим не повинен впливати на достовірність первинних записів.

За недоступності сховища необхідно призупиняти запуск нових перевірок, якщо неможливо гарантувати збереження їхніх результатів.

Результатом алгоритму є контрольоване відновлення роботи без змішування помилок системи з результатами оцінювання захисту.

Забезпечення безпеки проведення експериментів

Дослідження антивірусних засобів потребує визначення допустимих меж тестування. Особливу увагу необхідно приділити ізоляції середовища, походженню тестових об’єктів і контролю виконуваних операцій.

Початковим етапом є визначення дозволеної області експерименту. Усі перевірки повинні виконуватися на власних або належним чином авторизованих ресурсах.

Для базового тестування доцільно використовувати нешкідливі навчальні програми, синтетичні події та офіційні контрольні файли. Використання цих матеріалів дозволяє досліджувати роботу окремих функцій захисту без створення дієвих механізмів шкідливої активності.

Окремо необхідно обмежити взаємодію тестових вузлів із продуктивною корпоративною мережею. Передавання результатів до централізованого сховища повинно здійснюватися через визначені інтерфейси.

Алгоритми штучного інтелекту доцільно застосовувати для аналізу покриття, класифікації результатів і вибору безпечних сценаріїв. Їхнє використання не повинно передбачати автоматичного створення шкідливих програмних варіантів або оптимізації програмного коду для уникнення виявлення.

Результатом застосування відповідних обмежень є контрольований дослідницький процес із визначеними вимогами до безпеки та відтворюваності.

Перевірка коректності алгоритмів та оцінювання ефективності

Для експериментальної перевірки запропонованого алгоритмічного забезпечення необхідно передбачити окреме оцінювання генерації мутантів, виконання тестових сценаріїв та інтелектуального аналізу результатів.

Початковим етапом є перевірка реєстрації програмних об’єктів. Для цього використовуються контрольовані файли з відомими характеристиками. Необхідно підтвердити правильність визначення формату, обчислення контрольних значень та збереження метаданих.

Далі перевіряються оператори мутації. Для кожного з них створюється набір тестових прикладів, у якому заздалегідь визначено очікувану зміну. Після

перетворення необхідно перевірити відповідність отриманого об’єкта встановленим умовам.

Окремо оцінюється механізм усунення дублікатів. Для цього формуються однакові артефакти та декілька контрольованих варіантів, які відрізняються за вмістом. Система повинна правильно розрізняти точні збіги та незалежні результати.

Наступним етапом є перевірка планувальника. Для цього можуть використовуватися синтетичні завдання з відомими параметрами тривалості й доступності тестових вузлів. Необхідно перевірити правильність переходів між станами та оброблення незавершених операцій.

Для оцінювання інтеграції з антивірусним засобом доцільно використовувати офіційно підтримувані контрольні сценарії. Під час перевірки необхідно підтвердити отримання результатів, правильність їхнього зв’язку із завданнями та можливість повторного виконання.

Особливу увагу потрібно приділити розмежуванню негативних і невизначених результатів. Для цього можуть моделюватися помилки доступу до тестового файлу, затримка відповіді та недоступність журналу подій. Система не повинна трактувати такі ситуації як підтверджену відсутність антивірусного виявлення.

Для перевірки кластеризації формується набір синтетичних тестових сценаріїв із визначеними категоріями. Після групування аналізується структура отриманих кластерів і представленість різних типів перевірок.

Як показники якості кластеризації можуть використовуватися Silhouette Score та Davies-Bouldin Index. Вони дозволяють оцінювати структуру групування, але не характеризують безпосередньо здатність антивірусних продуктів виявляти загрози.

Для моделі інтелектуальної пріоритизації необхідно сформувати навчальну та тестову вибірки. Доцільно забезпечити розділення споріднених мутантів, щоб уникнути завищення якості прогнозування.

За наявності перевірених категоріальних міток можуть використовуватися Precision, Recall і F1-score. Якщо модель прогнозує числову оцінку інформативності тесту, відповідні результати можуть перевірятися за допомогою Mean Absolute Error.

Для визначення практичної доцільності машинного навчання необхідно порівнювати його результати з евристичними правилами вибору тестів. Наприклад, базовий підхід може надавати перевагу категоріям із найменшою кількістю виконаних перевірок.

Під час порівняння доцільно використовувати однакові набори сценаріїв та обмеження кількості тестів. Це дозволяє визначити, чи забезпечує інтелектуальна пріоритизація додаткове покриття відносно простих методів.

Окремо необхідно оцінювати відтворюваність виявлених невідповідностей. Для цього контрольовані сценарії повторюються за однакових умов. Результати перевіряються на узгодженість.

Для оцінювання продуктивності можуть вимірюватися час підготовки тестового артефакта, тривалість окремої антивірусної перевірки, затримка нормалізації та час формування аналітичного звіту.

За збільшення кількості паралельних перевірок необхідно контролювати використання процесорних ресурсів, оперативної пам’яті та пропускної здатності службових каналів.

Важливим етапом є перевірка працездатності після відмови компонентів. Для цього можуть моделюватися тимчасова недоступність тестового середовища, переривання роботи планувальника й порушення з’єднання зі сховищем.

Очікуваним результатом є правильне збереження стану завдань і відсутність некоректного включення незавершених тестів до статистики успішних або невдалих перевірок.

Кількісні висновки щодо повноти покриття, ефективності мутаційного тестування, якості машинного навчання та продуктивності програмного комплексу можуть бути сформовані лише після проведення відповідних експериментальних досліджень.

Запропоноване алгоритмічне забезпечення формує послідовний процес мутаційного тестування програмних об’єктів від підготовки безпечних вихідних артефактів до аналізу реакції захисних механізмів та визначення недостатньо перевірених сценаріїв.

Основними складовими є контрольоване формування мутантів, перевірка їхньої коректності, автоматизація тестових завдань, нормалізація результатів і застосування машинного навчання для оцінювання покриття. Особливе значення має розмежування підтверджених невідповідностей, технічних помилок і відсутності достатньої інформації. Сукупність розглянутих алгоритмів створює основу для програмної реалізації системи, призначеної для дослідження та вдосконалення процедур перевірки антивірусного захисту в контрольованому середовищі.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програмне забезпечення системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту призначене для автоматизованого дослідження повноти антивірусних перевірок і визначення потенційних сліпих зон захисних механізмів. Система забезпечує формування контрольованих варіантів безпечних тестових об’єктів, виконання перевірок, накопичення результатів та їхній інтелектуальний аналіз. Використання алгоритмів машинного навчання дозволяє групувати тестові сценарії, оцінювати їхнє покриття та визначати пріоритети подальшого дослідження. Головне вікно програми показане на рисунку 5.1.

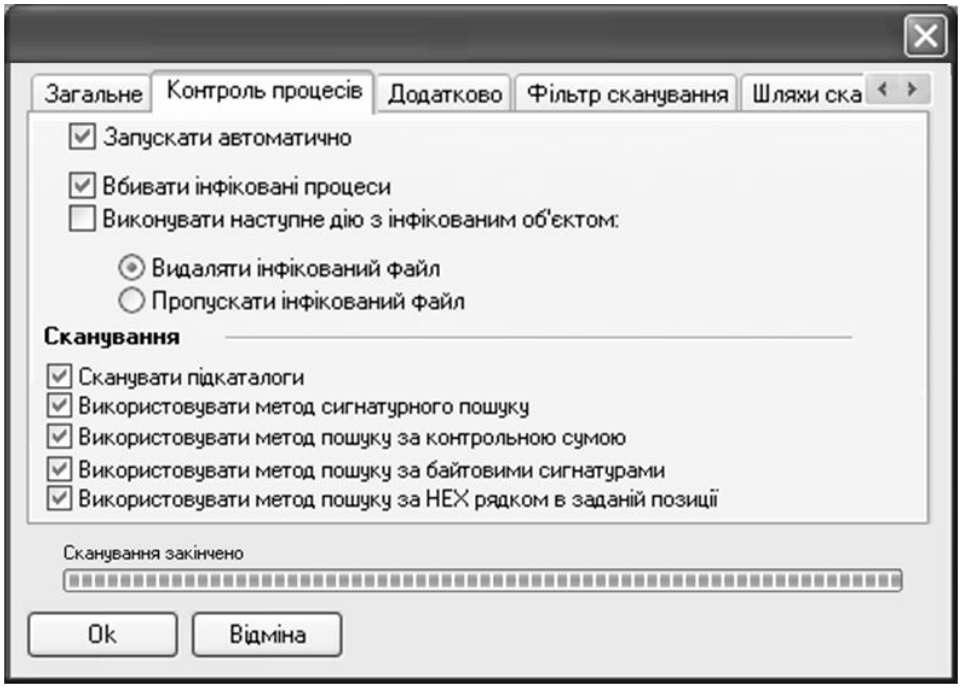

Рисунок 5.1 – Головне вікно ПЗ

Впровадження програмного комплексу передбачає підготовку ізольованого тестового середовища, встановлення необхідних програмних

компонентів, налаштування механізмів взаємодії з антивірусними засобами та перевірку працездатності аналітичної підсистеми. Основними функціональними складовими є модуль керування тестовими об’єктами, генератор контрольованих мутацій, планувальник перевірок, компонент отримання результатів і підсистема машинного навчання. Окремо передбачається сховище метаданих і результатів експериментів.

Особливу увагу необхідно приділити ізоляції тестового середовища. Для цього можуть використовуватися окремі віртуальні машини з попередньо підготовленими конфігураціями. Мережеві з’єднання тестових вузлів повинні обмежуватися необхідними службовими каналами. Якщо перевірка не потребує доступу до зовнішніх ресурсів, доцільно використовувати ізольований мережевий сегмент. Відновлення початкового стану середовища може здійснюватися за допомогою контрольних знімків віртуальних машин.

Після підготовки середовища необхідно сформувати набір початкових програмних об’єктів. Для тестування доцільно використовувати нешкідливі навчальні програми, синтетичні файли та інші контрольовані артефакти. Кожний об’єкт реєструється із зазначенням типу, призначення та допустимих способів оброблення. Для перевірки його незмінності може використовуватися криптографічне контрольне значення SHA-256. Початкові об’єкти доцільно зберігати окремо від результатів мутацій.

Наступним етапом є налаштування операторів мутації. Для кожного типу об’єкта визначаються дозволені перетворення та правила перевірки їхньої коректності. Генератор повинен створювати окрему копію вихідного артефакта без зміни оригіналу. Після виконання перетворення необхідно перевірити структуру отриманого файлу та відповідність установленим умовам. Об’єкти, які не пройшли перевірку, виключаються з подальшого тестування.

Для перевірки базової працездатності антивірусної інфраструктури може застосовуватися стандартний EICAR Anti-Virus Test File. Його використання дозволяє перевірити реакцію підтримуваного антивірусного засобу без застосування реального шкідливого програмного забезпечення. Контрольний

файл повинен використовуватися відповідно до його офіційного призначення. Довільна зміна вмісту такого файлу не гарантує збереження властивостей тестової сигнатури.

Підключення антивірусних засобів передбачає налаштування підтримуваного способу отримання результатів перевірки. Залежно від програмного продукту можуть використовуватися документовані інтерфейси командного рядка, API або журнали подій. Для кожного адаптера необхідно встановити параметри доступу, допустимі операції та правила інтерпретації повідомлень. Службові облікові дані повинні зберігатися окремо від вихідного коду.

Для централізованого накопичення результатів необхідно підготувати сховище метаданих. Як можливий варіант може використовуватися PostgreSQL. Сховище повинно забезпечувати зв’язок між початковим об’єктом, його мутантами, завданнями та отриманими результатами. Окремо реєструються версії антивірусних засобів і відповідні конфігурації. Це необхідно для відтворення умов перевірки та коректного порівняння результатів.

Після підготовки сховища налаштовується планувальник тестових завдань. Для нього визначаються доступні тестові вузли, обмеження паралельності та допустима тривалість виконання перевірок. Кожне завдання повинно мати унікальний ідентифікатор і поточний стан. Повторне виконання після технічної помилки не повинно призводити до неконтрольованого дублювання результатів.

Аналітичний компонент потребує окремого налаштування. Для його роботи формується набір характеристик тестових сценаріїв, який може включати тип об’єкта, категорію мутації, параметри перевірки та відомості про завершення тесту. Для групування подібних сценаріїв може використовуватися K-means або HDBSCAN. Після навчання необхідно зберегти параметри моделі та правила попереднього оброблення ознак. Для визначення пріоритетів подальшого тестування може застосовуватися окрема модель класифікації або регресії.

Рекомендована послідовність запуску починається з перевірки доступності сховища та тестових вузлів. Далі активуються модулі реєстрації об’єктів і

формування контрольованих мутацій. Після цього запускається планувальник та перевіряється взаємодія з антивірусними адаптерами. Наступним етапом є завантаження моделей машинного навчання. Завершується запуск активацією аналітичного компонента та перевіркою доступності користувацького інтерфейсу.

Первинну функціональну перевірку доцільно виконувати на невеликому наборі безпечних тестових об’єктів. Спочатку перевіряється правильність їхньої реєстрації та створення контрольованих мутантів. Після цього контролюється збереження метаданих і можливість визначення зв’язку між початковим об’єктом та його варіантами. Необхідно переконатися, що некоректні або повторювані артефакти обробляються відповідно до встановлених правил.

Наступним етапом є перевірка взаємодії з антивірусним засобом. Використовується контрольний сценарій із визначеною очікуваною реакцією. Після виконання перевірки система повинна отримати відповідне повідомлення та зіставити його з ідентифікатором завдання. Окремо необхідно перевірити оброблення випадків відсутності відповіді, помилки доступу та незавершеного сканування. Такі результати повинні позначатися як невизначені, а не як підтверджене невиявлення.

Для перевірки машинного навчання можуть використовуватися попередньо підготовлені набори тестових сценаріїв. Необхідно контролювати правильність формування числових ознак, завантаження навченої моделі та групування результатів. Додатково перевіряється формування матриці покриття. Система повинна розрізняти виконані, невиконані та технічно некоректні перевірки. Випадки невідповідності очікуваній реакції повинні передаватися на додатковий аналіз.

У процесі експлуатації можливі помилки запуску тестового вузла, недоступність антивірусного інтерфейсу, порушення роботи сховища та невідповідність вхідних ознак параметрам навченої моделі. Для діагностики використовуються службові журнали програмних компонентів. За тимчасової відмови аналітичної підсистеми допускається накопичення первинних результатів

без машинного навчання. Перед продовженням тестування необхідно перевірити готовність середовища та можливість збереження нових записів.

Перед введенням комплексу в регулярну експлуатацію необхідно перевірити ізоляцію тестових вузлів, розмежування прав доступу, правильність відновлення початкового стану середовища та збереження історії експериментів. Окремо контролюється працездатність після перезапуску, відповідність конфігурацій моделей і можливість повторного виконання контрольних сценаріїв. Результати цих перевірок використовуються для оцінювання експлуатаційної готовності системи.

Для документування програмного продукту передбачається представлення відомостей про його розробника. Відповідна інформація використовується для ідентифікації авторства, супроводження програмного комплексу та уточнення його призначення. Авторські дані системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту представлені на рисунку 5.2.

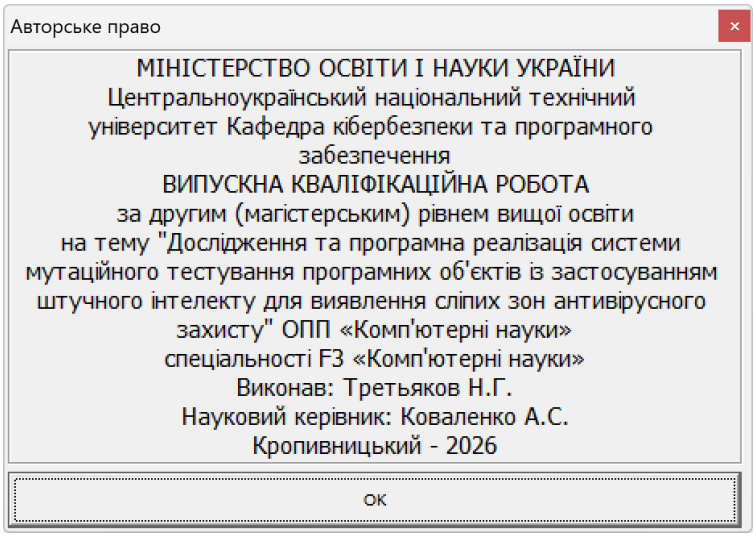

Рисунок 5.2 – Авторське право

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Метою розробки є дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Об’єктом дослідження є процес мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Предметом дослідження є методи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– Розроблено вітчизняний продукт мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

7.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-97.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-17.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

7.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Оскільки робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у тому числі програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-97.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у тому числі високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

Лікарі вважають, що з'явився новий тип захворювання – синдром стресу оператора дисплея, дія якого проявляється у вигляді головного болю, підвищеної стомлюваності та слабкості, порушення сну, болю в області спини, шиї та зап'ястків, напруженого стану м'язів, відчуття скутості, запалення сухожиль, запальних процесів м'язів, що керують рухами передпліч і пальців, неприємних відчуттів в області очей, що супроводжуються різзю, алергії. Можливі астматичні прояви, пригніченість, млявість, депресія. про аналітичного паралічу, тобто. нездатності до аналізу одержуваної інформації. Напружений характер праці, висока вартість помилки оператора, перебування у стані тривожно-томного очікування за відсутності емоційної розрядки ведуть до невротизації особистості, що може несприятливо позначатися на спадковості як невротизації потомства.

7.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 7.1. У зазначеному приміщенні працюють двоє людей. За даними, які наведено у таблиць 7.1-7.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207,

від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Таблиця 7.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

Таблиця 7.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

Таким чином, можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря

в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 7.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 7.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість, %	Швидкість повітря, м/с	Температура,°С	Вологість %	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні. У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України “Природне і штучне освітлення” – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у тому числі світлодіодних.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – це основна гігієнічна вимога. Оскільки яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

7.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних допустимих значень.

Таким чином, можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Оскільки при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

7.5 Розрахункова частина

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,6 м, висота – 3 м. У зазначеному приміщенні працює 1 особа.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=E\cdot S\cdot K\cdot Z/n,$$

де

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S = 3х4,6 = 13,8 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп, обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i = S/(h\cdot(A+B)),$$

де

S – площа приміщення, S = 13,8 м^2;

h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, оскільки лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 3 м;

B – довжина приміщення, B = 4,6 м.

Підставимо всі значення у формулу та визначимо індекс приміщення: i=0,43.

Знаючи індекс приміщення, знаходимо n = 0,23 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним

типом ламп). Підставимо всі значення у формулу, визначимо світловий потік: F=29700 Лм.

Для розрахунку будемо використовувати світлодіодні панелі LED панель 42 Вт 6000 К SUNLED 000000127, світловий потік яких $F_л$ = 3990 Лм.

Число ламп визначається за формулою:

$$N=F/F_л$$

де

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначаємо кількість ламп:

$$N = 29700/ 3990 =7,4 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 8 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

8 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– Досліджена система мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

– На основі отриманих результатів досліджень створена програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання мутаційного тестування програмних об’єктів із

застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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. 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 техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи 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.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Третьяков Н.Г. Дослідження та програмна реалізація системи мутаційного тестування програмних об’єктів із застосуванням штучного інтелекту для виявлення сліпих зон антивірусного захисту // Збірник праць молодих науковців ЦНТУ. – Вип. 17. – Кропивницький: ЦНТУ, 2027.
2. Hurbans R. Grokking Artificial Intelligence Algorithms. – Manning, 2020. – 631 p.
3. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.
4. Kotu V., Deshpande B. Data Science: Concepts and Practice. – Elsevier Science, 2018. – 953 p.
5. Knowledge Base A Complete Guide – 2021 Edition // The Art of Service – Knowledge Base Publishing, 2020. – 306 p.
6. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.
7. Mattmann C. Machine Learning with TensorFlow, Second Edition. – Manning, 2020. – 1124 p.
8. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.
9. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.
10. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.
11. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O’Reilly. 2019. – 252 p.
12. 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.

13. Vintenko, B., Myronets, I., Smirnov, O., Smirnova, T. «Enhancing the Reliability of Input Data for Nuclear Power Plant Operator Support Systems». *International Conference on Sensors, Devices and Systems. SDaS 2025. Lecture Notes in Electrical Engineering,* 2026. vol 1617. pp 3-12 Springer, Cham.

14. Богаченко, С., Смірнов, О., Буравченко, К., Смірнова, Т., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С. «Обмеження ресурсів мобільних та edge-платформ при розробці програмного забезпечення для реалізації та оптимізації моделей машинного навчання». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2026, т. 2 № 34, С. 677-686.

15. Смірнов, О., Ткачук, Р., Козірова, Н., Константинова, Л., Конопліцька-Слободенюк, О., Якименко, Н., Смірнов, С.. «Дослідження застосування векторних баз даних у генеративному штучному інтелекті». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*, 2026, Том 1 № 33, 667-682.

16. Вінтенко Б.Ю., Смірнова Т.В., Миронець І.В., Смірнов О.А., Буравченко К.О. «Метод оцінки функціональної стійкості комп'ютерно-орієнтованих процедур системи підтримки оперативного персоналу АЕС». *Системи управління, навігації та зв'язку,* 2026, вип. 2(84), С. 62-69.

17. Смірнов О., Заріцкий В., Буравченко К., Конопліцька-Слободенюк О., Константинова Л., Якименко Н., Смірнов С. «Оптимізація розпізнавання облич за допомогою CUDA прискореної бібліотеки dlib». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2026, Том 4, № 32, С. 573-582.

18. Лисенко, І., Минайленко, Р., Смірнов, С., Буравченко, К., Якименко, Н., Смірнов, О. «Дослідження інструментів штучного інтелекту для інтелектуального аналізу даних». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2025, № 3(31), С. 227–241.

19. Kuznetsov O., Frontoni E., Arnesano M., Smirnov O., Khruskov B. «Biometric Authentication in TinyML: Opportunities and Challenges». *Tiny Machine*

Learning: Design Principles and Applications, 2026, pp. 587-633.

20. Kuznetsov O., Frontoni E., Kuznetsova Y., Chevardin V., Smirnov O. «Architectural foundations for adaptive security in edge computing systems». *Cybersecurity Defensive Walls in Edge Computing*, 2025. pp. 21-61.

21. Вінтенко, Б.Ю., Миронець, І.В., Смірнов, О.А., Коваленко, О.В., Усік, П.С., Буравченко, К.О., Лисенко, І.А. «Логіко-структурна модель комп'ютерно-орієнтованої процедури системи підтримки оперативного персоналу АЕС». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С. 413-427, 2025.

22. Смірнова, Т.В. «Дослідження методів, моделей та сучасних ІТ-рішень для підтримки технологічних процесів у критичній інфраструктурі держави». *Кібербезпека: освіта, наука, техніка*. 2025. Том 2 № 30. С.195-208, 2025.

23. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка*. 2025. Том 1 № 29. С.704–716, 2025

24. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

25. Kuznetsov, O., Smirnov, O., Akhmetov, B., Alimseitova, Z., Imoize, A.L. «Deep Learning Frontiers in Copy-Move Forgery Detection: Advances, Challenges, and Future Directions». *Advancements in Cybersecurity Next Generation Systems and Applications*, 2025. 202-229.

26. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

27. Вінтенко Б.Ю., Смірнов О.А., Миронець І.В., Смірнова Т.В.,

Коваленко О.В., Мацуй А.М. «Модель шляхів отримання вхідних даних комп'ютерної інтелектуальної системи підтримки оперативного персоналу АЕС». *Центральноукраїнський науковий вісник. Технічні науки.* 2025. Вип. 11(42), ч. II. С.52-62.

28. Смірнов, О.А., Константинова, Л.В., Конопліцька-Слободенюк, О.К., Козірова, Н.В, Якименко, Н.М., Доренський, О.П., Буравченко, К.О. «Дослідження інструментів штучного інтелекту для роботи з базами даних та аналізу даних». *Кібербезпека: освіта, наука, техніка.* 2025. №3(27), С. 429–448.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

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

45. 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.

46. 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.

47. 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.

48. 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.

49. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

50. 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.

51. 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.

52. 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.

53. 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.

54. 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.

55. 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.

56. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

57. 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.

58. 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.

59. 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.

60. 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.

61. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

62. 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.

63. 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.

64. 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.