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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Ходаковський Д.А.
«____» _____ 2025 р.

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

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

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

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

Ходаковському Даниїлу Анатолійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення для оцінки ризиків і виявлення загроз у системі кібербезпеки з використанням логіко-імовірнісних методів*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Алгоритм обробки вхідних даних	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз сучасних підходів до кібербезпеки та логіко-імовірнісних методів	11.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	17.03.2025 р.	
3.	Розробка логіко-імовірнісної моделі компонента системи	21.03.2025 р.	
4.	Побудова структури даних та підготовка вхідних параметрів	25.03.2025 р.	
5.	Розробка алгоритмів оцінки ризиків	30.03.2025 р.	
6.	Реалізація програмного модуля та тестування	10.04.2025 р.	
7.	Оформлення ПЗ	19.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

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

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Ходаковський Д.А.
(прізвище та ініціали)

АНОТАЦІЯ

Ходаковський Д.А. Програмне забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

У даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для автоматизованого аналізу кіберризиків з використанням логіко-імовірнісних методів.

Метою розробки є створення інтелектуального програмного модуля, що дозволяє виявляти, оцінювати ймовірність та прогнозувати розвиток кіберзагроз шляхом застосування імовірнісних моделей у реальному часі.

Результатом роботи є програмна реалізація системи аналізу ризиків інформаційної безпеки з урахуванням невизначеності та неповноти вхідних даних. Розроблена система здатна працювати з телеметричними подіями, логами, індикаторами атак та використовує логіко-імовірнісні структури для формування висновків про рівень загрози.

У процесі проєктування було застосовано моделі дерева атак, байєсівських мереж, логіко-імовірнісного оцінювання інцидентів, а також розроблено зручний графічний інтерфейс користувача. ПЗ забезпечує інтеграцію з системами моніторингу, має можливість розширення та адаптації під інфраструктуру підприємства.

Розроблене ПЗ функціонує в середовищі ОС Windows 10/11, реалізоване мовою Python із використанням бібліотек логіко-імовірнісного моделювання.

Ключові слова: кібербезпека, логіко-імовірнісні методи, ризик, атакуючі сценарії, Python.

ABSTRACT

Khodakovskyi D.A. Software for a cybersecurity system based on logical-probabilistic methods. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

This bachelor's thesis presents the development of software designed for automated risk assessment in cybersecurity using logical-probabilistic methods.

The goal of the project is to create an intelligent software module capable of detecting, evaluating the probability, and predicting the progression of cyber threats by applying probabilistic models in real time.

The result of the work is a software system for risk analysis in information security that takes into account uncertainty and incompleteness of input data. The developed system can process telemetry data, logs, attack indicators, and uses logical-probabilistic structures to generate conclusions about threat levels.

During the design process, models such as attack trees, Bayesian networks, and logical-probabilistic risk evaluation were used. A convenient graphical user interface was also developed. The software supports integration with monitoring systems and can be extended or adapted to a specific enterprise infrastructure.

The developed software operates in the Windows 10/11 environment and is implemented in Python using libraries for probabilistic modeling.

Keywords: cybersecurity, logical-probabilistic methods, risk, attack scenarios, Python.

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ - програмне забезпечення

ЛІМ - логіко-імовірнісна модель

BN - Bayesian Network – байєсівська мережа

LPT - Logical-Probabilistic Tree – логіко-імовірнісне дерево

IDS - Intrusion Detection System – система виявлення вторгнень

IPS - Intrusion Prevention System – система запобігання вторгненням

SIEM - Security Information and Event Management – система управління подіями безпеки

SOC - Security Operation Center – центр операцій безпеки

UEBA - User and Entity Behavior Analytics – аналітика поведінки користувачів і об’єктів

UAC - User Access Control – керування доступом користувача

IV - Initialization Vector – ініціалізаційний вектор

CVSS - Common Vulnerability Scoring System – система оцінювання вразливостей

ML - Machine Learning – машинне навчання

ВСТУП

Актуальність теми. У сучасному цифровому середовищі з кожним роком зростає кількість і складність кіберзагроз, які ставлять під загрозу безпеку інформаційних систем у всіх сферах діяльності — від державного управління до бізнесу. Поширення хмарних сервісів, мобільних пристроїв, віддаленої роботи, Інтернету речей (IoT) та цифрової трансформації створює нові вектори атак, які неможливо ефективно виявити за допомогою традиційних, сигнатурних або статичних систем захисту.

У таких умовах важливою стає потреба у системах, здатних не лише виявляти відомі загрози, але й прогнозувати потенційні інциденти, аналізуючи можливі сценарії розвитку атак із врахуванням неповних, нечітких або ймовірнісних даних. Особливої актуальності набувають **логіко-імовірнісні методи**, що дозволяють будувати моделі атак, враховувати взаємозв'язки між подіями та оцінювати ймовірність реалізації загроз. Це забезпечує новий рівень інтелектуального управління ризиками у сфері кібербезпеки.

Враховуючи тенденції переходу до моделей Zero Trust, активного використання SIEM, SOAR та аналітики загроз (threat intelligence), впровадження програмного забезпечення, яке підтримує логіко-імовірнісне оцінювання, є необхідним кроком для побудови гнучкої, адаптивної та надійної системи захисту інформації.

Мета й завдання дослідження. Метою цієї кваліфікаційної роботи є розробка програмного забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів, здатного оцінювати, прогнозувати та виявляти потенційні загрози в інформаційному середовищі організації.

Для досягнення поставленої мети визначено такі основні завдання дослідження:

- Провести огляд існуючих систем кібербезпеки, що використовують або можуть бути інтегровані з логіко-імовірнісним аналізом.

- Дослідити **математичні методи логіко-імовірнісного моделювання**, зокрема дерева атак, байєсівські мережі, дерева відмов.
- Розробити **програмну модель** оцінювання кіберризиків, яка здатна працювати з неповними або умовними даними.
- Реалізувати **графічний інтерфейс користувача** для побудови сценаріїв атак та перегляду результатів моделювання;
- Провести **тестування системи на прикладах типових атак**, оцінити її ефективність та адаптивність до нових загроз.

Практична цінність отриманих результатів розробленого програмного забезпечення полягає в можливості його інтеграції в інформаційно-аналітичні центри кіберзахисту (SOC), а також у використанні як допоміжного інструмента для прийняття рішень у сфері управління інформаційною безпекою. Реалізовані моделі дозволяють швидко реагувати на інциденти, оцінювати ймовірність поширення загрози, визначати пріоритетність реагування та підтримувати принципи ризик-орієнтованого підходу.

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

У сучасних умовах розвитку інформаційного суспільства питання забезпечення кібербезпеки набуває особливого значення. Кількість кібератак щороку зростає, а їх складність та прихованість змушують розробників систем захисту шукати нові підходи до виявлення та попередження загроз. У зв'язку з цим, надзвичайно актуальним стає впровадження інтелектуальних систем, що здатні самостійно аналізувати події, виявляти закономірності та формувати висновки щодо рівня ризику реалізації тієї чи іншої загрози. Одним із таких підходів є використання логіко-імовірнісних методів.

Призначенням даної системи є забезпечення комплексного аналізу та оцінки ризиків кібербезпеки шляхом застосування логіко-імовірнісного моделювання для виявлення, прогнозування та реагування на потенційні інциденти інформаційної безпеки.

Розроблене програмне забезпечення призначене для:

- Оцінки ймовірності реалізації кіберзагроз на основі вхідних подій (логів, телеметрії, поведінкових характеристик користувачів).
- Виявлення прихованих залежностей між подіями, які окремо не мають ознак загрози, але в сукупності формують ланцюг атаки.
- Моделювання сценаріїв розвитку подій, що дає змогу передбачити наслідки та вплив потенційних атак на інформаційну систему.
- Пріоритезації ризиків, що дозволяє службі безпеки або автоматизованим компонентам спрямовувати ресурси на найбільш критичні загрози.
- Інтеграції з існуючими засобами безпеки (SIEM, IDS, EDR тощо) для взаємного обміну інформацією про інциденти.

- Формування рекомендацій або автоматизованого реагування на основі визначеного рівня ризику.

Особливістю розробленої системи є те, що вона здатна працювати в умовах неповноти або невизначеності вхідних даних, що типово для реального середовища. Використання логіко-імовірнісних моделей, зокрема дерев рішень, байєсівських мереж або моделей сценаріїв, дозволяє ефективно компенсувати відсутні дані шляхом обчислення умовних ймовірностей.

Система орієнтована на використання у корпоративному середовищі, у службах інформаційної безпеки підприємств, які прагнуть автоматизувати процеси моніторингу, оцінки ризиків та прийняття рішень у сфері кіберзахисту. Окрім того, вона може бути використана як навчальний і дослідницький інструмент для аналізу ефективності різних стратегій захисту у змодельованих умовах.

Таким чином, головне функціональне призначення системи — надання фахівцям з інформаційної безпеки інструмента прогнозного аналізу, що ґрунтується на науково обґрунтованих методах обробки даних з високим ступенем невизначеності.

1.2 Область використання

У сучасному світі кіберзагрози є постійним викликом для безпеки інформаційних систем підприємств, установ і державних структур. Зловмисники дедалі частіше застосовують складні багаторівневі атаки, які важко виявити традиційними методами, особливо в умовах невизначеності або відсутності повної інформації. Саме тому виникає потреба у впровадженні інтелектуального програмного забезпечення, що використовує логіко-імовірнісні методи для оцінки ризиків, моделювання атак і підтримки прийняття рішень.

Програмне забезпечення, розроблене в рамках даної випускної кваліфікаційної роботи, має широке практичне застосування в різноманітних сферах, де інформаційна безпека є критично важливою. Його ключовою перевагою

є здатність працювати з неповними або нечіткими даними, виявляючи потенційні загрози на основі ймовірнісних моделей.

Прикладові області використання:

1. Корпоративний сектор

У великих компаніях, банках, страховому бізнесі та IT-компаніях, щодня фіксуються тисячі подій безпеки. Використання логіко-імовірнісного аналізу дає змогу автоматизувати виявлення загроз, які інакше могли б залишитися непоміченими через надлишок даних (data overload).

Наприклад, якщо користувач входить у систему з незвичного місця, паралельно змінює свій пароль і надсилає великі обсяги даних на зовнішній сервер, логіко-імовірнісна модель може визначити підвищений ризик компрометації та попередити адміністратора.

2. Державні установи та органи управління

У сфері національної безпеки важливо швидко і точно виявляти атаки, які можуть порушити роботу державних реєстрів, поширити дезінформацію або викрасти чутливі дані. Запровадження програмного забезпечення на базі логіко-імовірнісних моделей дозволяє оцінювати ймовірність успішного вторгнення за наявними індикаторами компрометації навіть у режимі реального часу.

3. Інфраструктура критичного значення

Енергетичні об’єкти, системи водопостачання, транспортні вузли — усі вони потребують надійного захисту. Атаки на такі системи часто є цілеспрямованими та багатоступеневими, і саме логіко-імовірнісні методи дозволяють моделювати весь можливий ланцюг розвитку інциденту.

Наприклад, якщо один вузол системи SCADA виходить з ладу або демонструє аномальну активність, система може визначити ймовірність того, що йдеться про початок атаки, та запропонувати запобіжні заходи.

4. Хмарні середовища та DevOps/DevSecOps-практики

У хмарній інфраструктурі, де постійно змінюються конфігурації, сервіси та користувачі, контроль ризиків є вкрай важливим. Логіко-імовірнісні моделі

дозволяють оцінювати наслідки змін доступу, оновлень або нових вразливостей ще до їх експлуатації зловмисниками.

5. Освітня сфера та наукові дослідження

Програмне забезпечення може використовуватись в університетах та науково-дослідних інститутах для моделювання кіберзагроз, навчання принципам побудови систем захисту та дослідження ефективності різних моделей ризику.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

У світі який стрімко розвивається, підключення до інтернету стає невід'ємною частиною програмного забезпечення. Існує безліч систем, технологій, архітектур, які дозволяються мати підключення до мережі інтернет, але потрібно не забувати важливість захисту підключення та конфіденційність інформації. Особлива увага до частини збереження інформації на хмарних середовищах. У цьому розділі будуть наведені існуючі технології реалізації системи кібербезпеки клієнтського доступу до бази даних та дієві аналоги хмарних платформ для збереження даних онлайн.

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми бакалаврської дипломної роботи

Огляд існуючих систем, технологій, архітектур та програмних рішень SIEM

Вступ до SIEM

Системи управління інформацією та подіями безпеки (SIEM — Security Information and Event Management) є критично важливою складовою сучасної інфраструктури кібербезпеки. Вони забезпечують централізований збір, зберігання, аналіз та кореляцію логів з різних джерел з метою виявлення підозрілої активності, інцидентів безпеки та відповідного реагування. Основна функція SIEM полягає в забезпеченні ситуаційної обізнаності про події в інформаційній системі в реальному часі, а також в історичному аналізі.

Основні функціональні компоненти SIEM

Типова SIEM-система складається з кількох ключових компонентів:

- Збір даних: лог-файли, події з мережевого обладнання, серверів, додатків, систем автентифікації.

- Нормалізація: приведення логів з різних джерел до єдиного формату.
- Збереження: централізоване та захищене збереження подій для подальшого аналізу та аудиту.
- Кореляція: виявлення закономірностей між подіями, які окремо можуть здаватися нешкідливими, але разом утворюють шаблон атаки.
- Моніторинг в реальному часі: виявлення інцидентів та оповіщення про них.
- Звітування та аудит: генерація звітів для аналізу, відповідності нормативам та оцінки ризиків.

Архітектурні підходи до побудови SIEM

Архітектура SIEM-систем зазвичай реалізується за такими моделями:

- Централізована — всі дані передаються до єдиного аналізуючого вузла. Має переваги в простоті управління, але слабкість — в масштабуванні та відмовостійкості.
- Децентралізована (розподілена) — компоненти збору та аналізу розміщені ближче до джерел подій. Краща масштабованість, адаптивність до великих інфраструктур.
- Гібридна — поєднання переваг обох підходів.

Архітектура може включати окремі компоненти для обробки великих даних (Big Data SIEM), машинного навчання, аналізу поведінки користувачів (UEBA — User and Entity Behavior Analytics) тощо.

Сучасні технології та тренди в SIEM

У відповідь на зростаючі складність та обсяг кіберзагроз, сучасні SIEM-системи інтегрують такі технології:

- Машинне навчання та штучний інтелект (AI/ML) для адаптивного виявлення аномалій.
- Логіко-імовірнісні методи, які дозволяють будувати моделі ризику на основі неповних чи нечітких даних.

- Інтеграція з SOAR (Security Orchestration, Automation and Response) для автоматизації процесів реагування.
- Хмарні SIEM (Cloud SIEM) — орієнтовані на організації з гібридною або хмарною інфраструктурою.
- Open-source рішення (наприклад, Wazuh, SIEMonster), які дозволяють кастомізацію та економію коштів.

Таблиця 2.1 – Популярні SIEM-рішення

№	Назва	Виробник	Особливості
1	Splunk Enterprise Security	Splunk Inc.	Потужний аналітичний двигун, гнучкий пошук, ML-аналітика
2	IBM QRadar	IBM	Потужна кореляція, інтеграція з іншими продуктами IBM
3	ArcSight	OpenText	Масштабованість, інструменти побудови сценаріїв
4	Azure Sentinel	Microsoft	Хмарна платформа, інтеграція з Microsoft 365
5	Wazuh	Open-source	Безкоштовна, легко інтегрується з ELK Stack
6	Elastic Security	Elastic.co	Побудована на Elasticsearch, зосереджена на швидкому пошуку та аналітиці

IDS (Intrusion Detection System)

Вступ до IDS

Системи виявлення вторгнень (IDS — Intrusion Detection System) є ключовими елементами інфраструктури інформаційної безпеки, які забезпечують моніторинг мережевого або системного трафіку з метою виявлення шкідливої чи аномальної активності. На відміну від систем запобігання вторгнень (IPS), IDS фіксують інциденти без втручання у трафік, часто працюючи в пасивному режимі.

Головна мета IDS — своєчасне виявлення спроб несанкціонованого доступу, зловмисної поведінки або політично некоректних дій у системі до того, як вони призведуть до серйозних наслідків.

Класифікація IDS

IDS-системи класифікуються за кількома ознаками:

- За методом виявлення:
  - Сигнатурні (Signature-based) — використовують базу відомих шаблонів атак (сигнатур).
  - Аномальні (Anomaly-based) — виявляють відхилення від нормальної поведінки, часто із використанням машинного навчання.
  - Гібридні — поєднують сигнатурний та аномальний підходи для підвищення точності.
- За місцем розгортання:
  - Мережеві IDS (NIDS) — аналізують трафік у мережі (наприклад, Snort).
  - Хостові IDS (HIDS) — контролюють активність на окремих пристроях (наприклад, OSSEC).
  - Гібридні — поєднують обидва підходи для покриття всієї IT-інфраструктури.

Архітектурні підходи до побудови IDS

Залежно від цілей і масштабу впровадження, IDS можуть мати такі архітектури:

- Централізована архітектура — всі дані від хостів або сенсорів передаються до центрального аналізатора.
- Розподілена архітектура — кожен вузол (хост чи мережевий сенсор) має локальний агент, який може самостійно приймати рішення або передавати дані до головного сервера.
- Багаторівнева архітектура — поєднує централізовану аналітику з локальним попереднім аналізом для зменшення навантаження на центральні ресурси.

Сучасні технології та тенденції в IDS

Сучасні IDS інтегрують інноваційні технології для покращення ефективності виявлення:

- Машинне навчання (ML) — для динамічного аналізу поведінки та виявлення невідомих атак.
- Логіко-імовірнісні методи — дозволяють ефективно працювати в умовах неповноти або невизначеності вхідних даних.
- Глибоке навчання (DL) — застосовується у високоточному виявленні складних атак у великому потоці даних.
- Інтеграція з SIEM — для централізованого збору логів і кореляції з іншими джерелами інформації.
- Контейнеризація та хмарна підтримка — підтримка DevOps-архітектур і хмарних середовищ.

Проблеми та виклики

Попри розвиток технологій, IDS залишаються чутливими до:

- Хибнопозитивних спрацьовувань — особливо в аномальних системах.
- Шифрування трафіку — ускладнює аналіз без дешифрування.
- Складності налаштування — потребують детального конфігурування правил та сценаріїв.

Таблиця 2.2 – Популярні SIEM-рішення

№	Назва	Тип	Особливості
1	Snort	NIDS	Сигнатурне виявлення, велика спільнота, гнучкість у налаштуванні
2	Suricata	NIDS	Підтримка багатопоточності, глибокий аналіз пакетів
3	Zeek(раніше Bro)	NIDS	Орієнтована на аналіз поведінки, створення складних сценаріїв
4	OSSEC	HIDS	Контроль файлів, аудит, сумісність із SIEM
5	Wazuh	HIDS/NIDS	Розширена версія OSSEC з веб-інтерфейсом та інтеграцією з ELK
6	Security Onion	Комплексна	Дистрибутив для глибокого моніторингу, включає Snort, Suricata, Zeek, Wazuh

Вступ до IPS

Системи запобігання вторгнень (IPS — Intrusion Prevention System) є розвитком і розширенням ідеї IDS, забезпечуючи не лише виявлення підозрілої активності, а й активне блокування потенційно небезпечних дій у реальному часі. IPS працюють в режимі *in-line*, тобто безпосередньо в мережевому трафіку, аналізуючи пакети на ходу і приймаючи рішення про їх дозволення, блокування або перенаправлення.

Завдяки цій властивості IPS-системи відіграють вирішальну роль у забезпеченні проактивного кіберзахисту — особливо в середовищах з високими вимогами до безперервності бізнес-процесів.

Основні функції IPS

Сучасні IPS-системи мають такі базові можливості:

- Аналіз трафіку в реальному часі.
- Виявлення атак за сигнатурами та поведінковими характеристиками.
- Автоматичне блокування атак на рівні мережі або хоста.

- Створення звітів і логів для подальшого аналізу та аудиту.
- Адаптивне оновлення баз загроз через хмарні сервіси або внутрішні системи обміну індикаторами компрометації (IoC).

Види IPS за принципом дії

- Мережеві IPS (NIPS) — розгортаються на мережевому периметрі або всередині сегментів для перевірки трафіку між вузлами.
- Хостові IPS (HIPS) — працюють на рівні операційної системи або додатків, контролюючи системні виклики, доступ до файлів, пам'яті тощо.
- Гібридні системи — поєднують мережевий і хостовий контроль.

Архітектура IPS

Типова архітектура IPS включає:

- Сенсори для моніторингу мережевого трафіку або хостових подій.
- Аналітичний рушій, що виконує сигнатурний, аномальний чи логіко-імовірнісний аналіз.
- Систему реагування, яка реалізує фільтрацію, перенаправлення або завершення сесій.
- Консоль керування для адміністрування, звітності, оновлень і політик.

Із впровадженням Zero Trust підходів архітектура IPS стає більш гнучкою, з можливістю роботи в гібридних і хмарних середовищах.

Методи виявлення загроз

- Сигнатурний аналіз — ефективний проти відомих атак, вимагає частого оновлення баз.
- Аналіз поведінки (аналітика аномалій) — визначає відхилення від нормального трафіку.
- Логіко-імовірнісні моделі — оцінюють імовірність того, що певна послідовність подій вказує на атаку, навіть якщо окремі компоненти виглядають безпечними.

- Машинне навчання та AI — застосовується для самонавчання та адаптації до нових загроз без попереднього визначення шаблонів.

Таблиця 2.3 – Популярні IPS-рішення

№	Назва	Тип	Особливості
1	Snort + Inline Mode	NIPS	Open-source, використовується з iptables для блокування
2	Suricata	NIPS	Підтримка багатопоточності, IPS-режим, TLS-декодування
3	Cisco Firepower NGIPS	NIPS	Глибока інтеграція з мережевими пристроями Cisco, потужна аналітика
4	Palo Alto Networks Threat Prevention	NIPS	Інтеграція з міжмережевими екранами, AI-аналітика
5	Trend Micro TippingPoint	NIPS	Ефективність для великих підприємств, швидке оновлення сигнатур
6	OSSEC (HIDS з IPS-функціями)	HIPS	Виявлення та реагування на рівні системи, інтегрується з SIEM

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програмного забезпечення, орієнтованого на моделювання та аналіз кіберризиків з використанням логіко-імовірнісних методів, було обрано мову програмування Python, а також низку відповідних бібліотек і технологій. Вибір цього інструментарію обґрунтовується рядом технічних, функціональних та практичних переваг, які забезпечують ефективність розробки та високу якість кінцевого продукту.

Python — це високорівнева мова програмування, яка є однією з найпопулярніших у світі. Її активно застосовують у компаніях таких як Google,

Meta, IBM, NASA, Dropbox, Netflix та інших. Вона ідеально підходить для проєктів, пов'язаних із кібербезпекою, обробкою даних, статистичним аналізом, машинним навчанням та побудовою інтелектуальних систем, до яких належить і ця дипломна робота.

Переваги Python для даного проєкту:

1. **Простий та інтуїтивний синтаксис** — дозволяє скоротити час на розробку, швидко перевіряти ітерації та зосередитися на логіці побудови ймовірнісних моделей.

2. **Інтерпретована природа мови** — дозволяє виконувати окремі частини програми без необхідності компіляції, що особливо зручно при створенні прототипів або експериментів із моделями.

3. **Динамічна типізація** — дає змогу створювати гнучкий і масштабований код, у якому можна ефективно працювати з різними типами даних.

4. **Підтримка об'єктноорієнтованого програмування (ООП)** — дозволяє структурувати програмне забезпечення у вигляді модулів, що спрощує тестування, оновлення та масштабування системи.

5. **Мультиплатформність** — код, написаний на Python, можна запускати як у Windows, так і в Linux-середовищах, що дозволяє легко адаптувати програму до потреб різних організацій.

Використані бібліотеки:

- pgmpy — потужна бібліотека для побудови графічних моделей (зокрема байєсівських мереж), які є основою для логіко-імовірнісного аналізу.
- pomegranate — використовується для роботи з ймовірнісними графами, деревами рішень, прихованими марковськими моделями (HMM).
- Tkinter — стандартна бібліотека для створення графічного інтерфейсу користувача (GUI) у Python.
- NumPy — забезпечує ефективну роботу з числовими масивами та лінійною алгеброю.

- Matplotlib / Seaborn — для візуалізації графів, розподілів ймовірностей, залежностей та індикаторів ризику.
- Scikit-learn (опціонально) — для реалізації класифікації та додаткового аналізу ризиків при потребі.

Розширення можливостей через фреймворки

Python має велику екосистему фреймворків, які можна застосувати у подальшому:

- Django — у разі створення веб-інтерфейсу для управління ризиками або інтеграції з базами даних.
- Flask — для побудови легкого REST API, що дозволяє підключати систему до SIEM або інших платформ.
- Dash / Streamlit — для створення інтерактивних дашбордів із результатами оцінки загроз.

Застосування Python у сфері кібербезпеки

Мова Python вже давно закріпилася в інструментарії спеціалістів з кіберзахисту. Вона активно використовується у створенні сканерів вразливостей, аналітичних інструментів, інтерфейсів до SIEM-систем та фреймворків для оцінювання подій безпеки. Це робить її ідеальною для розробки рішень, які потребують високої адаптивності, інтеграції та логічної обробки даних.

Прикладом є використання бібліотеки pgmpy для моделювання дерева атак, в якому вузли відповідають за окремі вектори загроз, а ймовірності — за їх можливу реалізацію. Такі структури дозволяють формувати гнучкі сценарії безпеки та автоматично обчислювати рівень ризику за кожним напрямком.

2.3 Розгорнута постановка завдання

У сучасних умовах стрімкого зростання кількості та складності кіберзагроз виникає необхідність у створенні програмного забезпечення, здатного не лише виявляти атаки на інформаційні системи, а й прогнозувати

можливі сценарії розвитку подій, оцінювати рівень ризику та підтримувати прийняття рішень службами інформаційної безпеки. З огляду на це, у даній роботі ставиться завдання створити програмне забезпечення, яке реалізує логіко-імовірнісні методи оцінки загроз.

Мета розробки:

Створити інтелектуальне програмне забезпечення, що дозволяє автоматизовано виявляти, аналізувати та оцінювати ймовірність реалізації загроз інформаційної безпеки з використанням логіко-імовірнісних моделей, таких як байєсівські мережі, дерева атак або дерева відмов.

Основні вимоги до системи:

1. Функціональні вимоги:

- Здійснювати введення початкових параметрів (вектори атак, умови, ймовірності, залежності).
- Побудова моделі у вигляді графічного дерева або ймовірнісної мережі.
- Розрахунок умовних та повних ймовірностей розвитку подій.
- Визначення рівня ризику для кожного сценарію або події.
- Побудова прогнозу розвитку атаки на основі вхідних даних.
- Формування звітів та візуалізація результатів у зручній формі.

2. Нефункціональні вимоги:

- Простий і зручний графічний інтерфейс для взаємодії з користувачем.
- Кросплатформенність (можливість запуску в середовищах Windows/Linux).
- Надійність: обробка некоректного введення, захист від аварійного завершення.
- Масштабованість: можливість додавання нових модулів без змін до базового ядра.
- Висока швидкодія при обчисленнях навіть у великих мережах загроз.

Завдання, які необхідно вирішити в процесі розробки:

1. Проаналізувати сучасні підходи до логіко-імовірнісного аналізу ризиків у сфері кібербезпеки.

2. Сформувати логічну модель сценарію атаки (дерево атак, байєсівська мережа або граф відмов).

3. Розробити програмний модуль, який дозволяє будувати структури залежностей між подіями.

4. Реалізувати обчислювальний модуль для автоматичного розрахунку ймовірностей виникнення подій.

5. Забезпечити зручний інтерфейс користувача з можливістю редагування вхідних даних, перегляду результатів та експорту звітів.

6. Провести тестування системи на прикладах реальних сценаріїв та оцінити її ефективність.

7. Оцінити переваги запропонованого підходу порівняно з традиційними методами оцінювання ризику (експертні оцінки, оцінки на основі правил тощо).

3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Архітектурний підхід

Запропоноване програмне забезпечення має **модульну архітектуру**, що дозволяє легко масштабувати систему, адаптувати її до потреб конкретної організації та інтегрувати з існуючими інструментами безпеки (SIEM, EDR, IDS/IPS тощо). Такий підхід забезпечує **гнучкість**, **розширюваність** і **мікросервісну сумісність** із корпоративними середовищами.

Система побудована з урахуванням розподіленої обробки подій та динамічного управління ризиками, використовуючи **логіко-імовірнісні методи**, що забезпечують роботу з неповними або суперечливими даними — типовими у сфері кібербезпеки.

Основні модулі системи

- **1. Модуль збору даних (Data Collection Module)**

Цей модуль є вхідною точкою всієї системи. Його завдання — **автоматизований збір інформації** з різноманітних джерел безпеки:

- системні логи серверів і робочих станцій (Syslog, Windows Event Log);
- трафік із мережевого рівня (PCAP, NetFlow, sFlow);
- журнали антивірусного ПЗ;
- сповіщення від IDS/IPS;
- дані від SIEM-систем;
- API EDR-рішень.

Зібрані дані проходять **агрегацію, нормалізацію** (до уніфікованого формату JSON/CEF/LEEF) та **маркування метаданими** (джерело, час, рівень критичності).

2. Модуль попередньої обробки (Preprocessing Module)

Відповідає за:

- **очищення** даних від шуму, дублікатів, нерелевантних записів;
- **фільтрацію** подій за критеріями важливості;
- **уніфікацію структури** даних — створення вхідного вектору параметрів для подальшого аналізу;
- збереження у проміжне сховище (наприклад, Elasticsearch або Apache Kafka для потокової обробки).

3. Аналітичний модуль (Analytics Engine)

Центральний елемент системи, що реалізує логіко-імовірнісний аналіз. Його основні функції:

- **побудова ієрархії загроз** за допомогою дерев рішень;
- **моделювання ймовірності атак** з використанням **байєсівських мереж** або **марковських процесів**;
- **кореляція подій** для виявлення складних багатоступеневих атак (APT);
- розрахунок метрик ризику (наприклад, CVSS × ймовірність) для кожного активу;
- виявлення **аномалій поведінки** на основі попередньо навчених профілів;
- вихід модуля — набір інцидентів із вказаною ймовірністю їх реалізації та ступенем критичності.

4. Модуль прийняття рішень (Decision Engine)

Цей модуль порівнює отримані оцінки ризику з **пороговими значеннями**, визначеними політиками безпеки організації. На основі цього:

- приймаються рішення щодо **типу реагування** (повідомлення, блокування, запуск скрипту, створення інциденту в SIEM);
- ініціюється **автоматична взаємодія з іншими системами**: firewall, IPS, AD, EDR;

- можуть бути **застосовані playbooks** згідно з SOAR-підходом (Security Orchestration, Automation and Response).

5. Модуль сповіщення (Alerting System)

Генерує повідомлення про загрози в режимі реального часу. Канали доставки:

- Email/SMS;
- інтеграція з месенджерами (Telegram, Slack, MS Teams);
- вебхуки в SIEM або SOC-платформи.

Повідомлення містять **рівень ризику**, **контекст події**, **рекомендації** до дій.

6. Модуль візуалізації (Visualization Dashboard)

Інтерактивна аналітична панель для:

- перегляду поточних та історичних інцидентів;
- візуалізації ймовірнісних моделей (графи, дерева, теплові карти);
- перегляду динаміки ризиків за активами/сегментами;
- налаштування політик і профілів безпеки.

Реалізується на базі веб-інтерфейсу (наприклад, з використанням React.js + D3.js або Grafana + Kibana).

7. Модуль інтеграції (Integration API)

Надає REST/GraphQL API та webhook-інтерфейси для:

- обміну даними з SIEM (Splunk, QRadar, ArcSight);
- передавання сигналів на EDR/IPS;
- взаємодії з зовнішніми базами загроз (Threat Intelligence);
- автоматичного оновлення моделей.

Основні функціональні можливості

1. **Автоматичний збір і обробка даних** з десятків джерел у реальному часі.

2. **Побудова ймовірнісних моделей атак** на основі логіко-імовірнісного аналізу.

3. **Виявлення аномалій** та створення контекстних інцидентів.

4. **Оцінка ризиків** для IT-активів компанії.

5. **Побудова дерев рішень** для моделювання сценаріїв вторгнень.

6. **Інтерактивна візуалізація** кіберзагроз та формування звітів для прийняття рішень.

7. **Масштабованість і гнучкість** — адаптація до організацій будь-якого рівня.

Розробка структурної схеми

Мета системи:

- **Моніторинг подій** та логів;
- **Виявлення аномальної активності**;
- **Ймовірнісний аналіз ризиків** на основі логіко-імовірнісних моделей;
- **Прийняття рішень** та ініціювання реагування на інциденти.

Використання логіко-імовірнісних методів:

Ці методи дозволяють:

- працювати з **неповною інформацією**;
- описувати **взаємозалежності** між подіями;
- формувати **прогнозовані сценарії атак**;
- приймати обґрунтовані рішення в умовах **невизначеності**.

Основні компоненти структурної схеми:

1. **Модуль збору даних**

○ Джерела: журнали подій, мережеві монітори, антивіруси, SIEM.

○ Дані: IP-адреси, час, тип події, користувач, контекст.

2. **Модуль попередньої обробки**

○ Фільтрація, нормалізація.

○ Побудова вхідного вектора (наприклад: <IP_src, Port_dst, Protocol, Event_type>).

3. **Аналітичний модуль**

- Байєсівські мережі та дерева рішень.
- Вихід: оцінка ймовірності загрози, тип ризику, індекс небезпеки.

4. **Модуль прийняття рішень**

- Порівняння з політиками.
- Вибір реакції: сповіщення, блокування, запуск скриптів.

5. **Модуль журналювання та звітності**

- Збереження подій, рішень, історії реагування.
- Автоматичне формування звітів (PDF, CSV, JSON).

6. **Інтерфейс адміністратора**

- Дашборд.
- Налаштування правил, моделей, політик.

3.2 Розробка структурної схеми

Розробка структурної схеми програмного забезпечення системи кібербезпеки є одним із ключових етапів проєктування, оскільки дозволяє візуалізувати основні функціональні модулі, визначити зв’язки між ними та забезпечити цілісність архітектури системи.

Структурна схема демонструє ієрархічну організацію компонентів ПЗ, розподілених відповідно до їхніх функцій та логіки взаємодії. Система має модульну структуру, що дозволяє легко масштабувати програму, змінювати або вдосконалювати окремі компоненти без повного перепроектування.

У структурі програмного забезпечення реалізовано такі основні функціональні блоки:

1. **Модуль введення даних**

Призначений для отримання вхідної інформації у вигляді логів, журналів подій, індикаторів атаки або інших телеметричних даних. Передбачає ручне або автоматизоване введення інформації з різних джерел (наприклад, SIEM або log-

файлів).

2. **Модуль попередньої обробки**

Виконує фільтрацію, нормалізацію та структурування вхідних даних. Це дозволяє зменшити «шум», виділити релевантні події та підготувати інформацію до моделювання.

3. **Модуль логіко-імовірнісного аналізу**

Є ядром системи. Реалізує побудову дерева атак, дерева відмов або байєсівської мережі залежно від сценарію. Виконує розрахунок умовних і повних ймовірностей виникнення інцидентів. Саме на цьому етапі система проводить інтелектуальну оцінку ризику.

4. **Модуль оцінювання ризику**

На основі результатів моделювання визначає рівень ризику для кожної виявленої загрози та ранжує події за критичністю. Дає змогу пріоритезувати реагування служб безпеки.

5. **Модуль візуалізації**

Формує графічне відображення моделі загроз: дерева рішень, графи, таблиці ризиків. Забезпечує інтерактивність для користувача та зручність сприйняття аналітичної інформації.

6. **Модуль збереження результатів та формування звітів**

Дозволяє експортувати результати в стандартні формати (PDF, DOCX, CSV), зберігати моделі та оцінки для подальшого аналізу або аудиту.

7. **Графічний інтерфейс користувача (GUI)**

Забезпечує взаємодію з користувачем, дозволяє вводити початкові дані, запускати аналіз, переглядати моделі та результати.

Взаємозв’язок між модулями реалізується за допомогою внутрішніх API або функціональних інтерфейсів, що забезпечує гнучкість та адаптивність системи до змін у конфігурації.

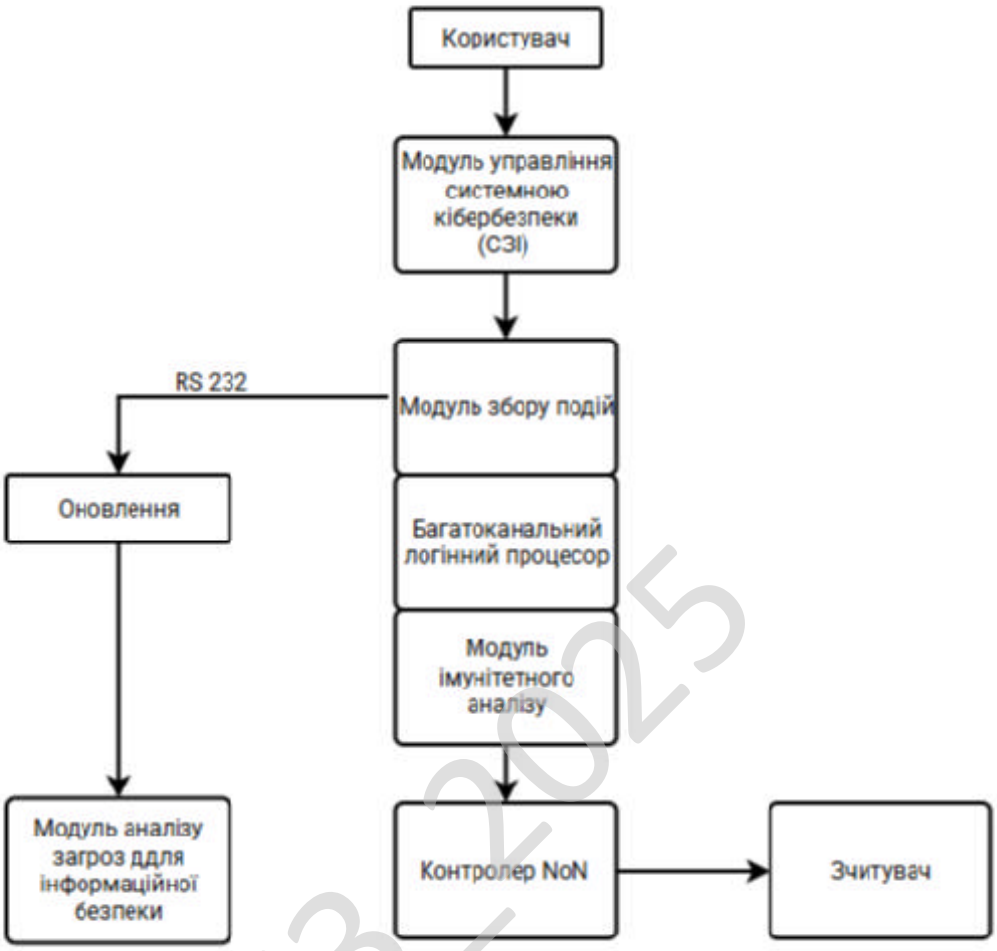

Рисунок 3.1 — Структурна схема системи кібербезпеки на основі логіко-імовірнісних методів

3.3 Розробка функціональної схеми

Функціональна схема програмного забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів відображає логіку обробки даних та основні етапи функціонування системи. Такий підхід дозволяє моделювати поведінку системи під час обробки подій безпеки та ухвалення рішень в умовах невизначеності.

Функціональна схема побудована з урахуванням вимог до систем виявлення та реагування на загрози, а також передбачає інтеграцію логіко-імовірнісних методів аналізу для підвищення точності діагностики потенційних інцидентів.

Клієнтська частина:

Клієнти — це кінцеві користувачі системи або прикладні сервіси, які мають доступ до її функціоналу через інтерфейс або API.

- **Інтерфейс користувача** — забезпечує зручну взаємодію з системою, візуалізацію результатів аналізу ризиків, перегляд подій та управління налаштуваннями.
- **Логування подій** — фіксує всі дії користувачів та системні події для подальшого аудиту.
- **API** — надає інтерфейс для взаємодії з іншими корпоративними системами.
- **Аутентифікація та авторизація** — забезпечують ідентифікацію користувачів та контроль доступу відповідно до ролей.
- **Захист даних** — включає шифрування, контроль доступу, захист від витоку інформації.
- **Перевірка цілісності** — дозволяє визначати зміни в даних або втручання.
- **Аудит** — забезпечує контроль за дотриманням політик безпеки та внутрішніх регламентів.
- **Серверна частина**

Сервер виконує роль центрального обчислювального ядра, де реалізуються алгоритми логіко-імовірнісного аналізу та шифрування.

- **Алгоритми шифрування** — відповідають за захист конфіденційної інформації під час передачі та зберігання.
- **Логіко-імовірнісний аналіз** — ключовий компонент, що виконує обробку подій безпеки, оцінку ризиків та побудову ймовірнісних сценаріїв атак на основі дерев рішень, байєсівських мереж тощо.

- **Основні функціональні модулі**

Ці модулі реалізують повний цикл управління інформаційною безпекою:

- **Аналіз загроз** — виявляє та класифікує потенційні загрози.
- **Моніторинг** — здійснює безперервний збір та фільтрацію подій у реальному часі.
- **Реагування** — автоматично або вручну активує відповідні протоколи безпеки.
- **Управління доступом** — керує правами користувачів та контролює вхід до критичних ресурсів.
- **Оцінка ризиків** — розраховує ймовірність реалізації загроз і їхній вплив на систему.
- **Моделювання атак** — дозволяє будувати сценарії атак та проводити тестування на стійкість системи.
- **Сповіщення** — інформує користувача або адміністратора про виявлені інциденти чи перевищення порогів ризику.
- **Звітність** — формує аналітичні та технічні звіти щодо поточного стану кібербезпеки.

Рисунок 3.2 - Функціональна схема системи

3.4 Розробка діаграми процесів

У межах проектування програмного забезпечення для системи кібербезпеки з використанням логіко-імовірнісних методів було розроблено детальну діаграму процесів. Ця діаграма візуалізує ключові етапи обробки даних, механізми прийняття рішень та алгоритми реагування на потенційні загрози в реальному часі.

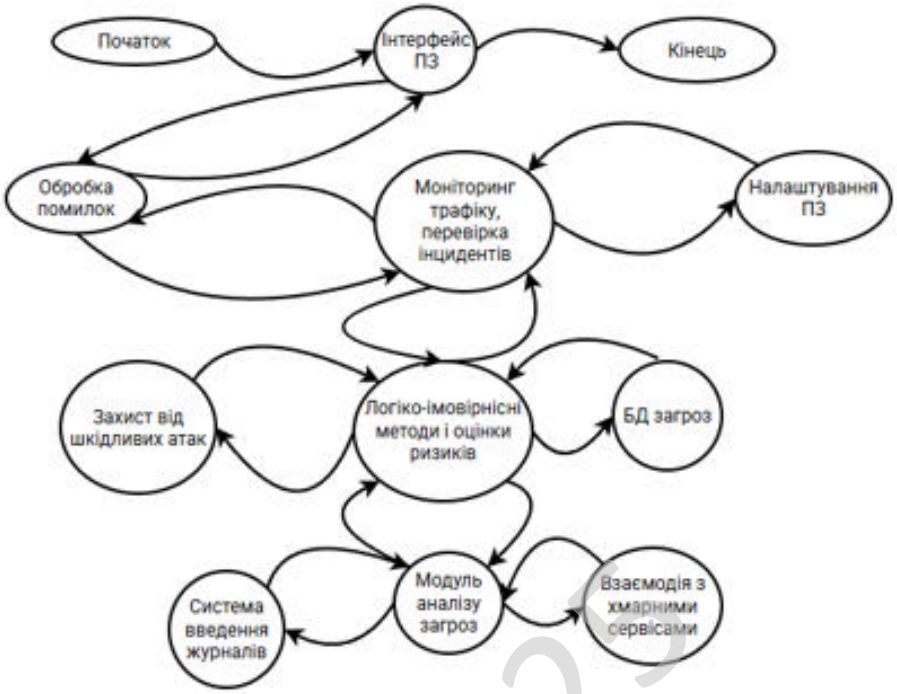

Рисунок 3.3 – Діаграма процесів системи

Основні етапи процесу:

Початок → Інтерфейс ПЗ

Робота системи починається з ініціалізації інтерфейсу. Інтерфейс ПЗ є стартовою точкою, через яку здійснюється взаємодія користувача з програмою. Він дозволяє налаштовувати параметри, запускати процеси аналізу та переглядати результати.

Інтерфейс ПЗ → Обробка помилок

З інтерфейсу користувач ініціює процес моніторингу вхідного та вихідного трафіку. Це необхідно для виявлення потенційних аномалій або вторгнень. На цьому етапі система виконує збір подій та логів.

Моніторинг трафіку → Налаштування ПЗ

В ході роботи може знадобитися доступ до налаштувань програмного забезпечення, наприклад, для коригування рівнів сповіщення або правил обробки даних.

Моніторинг трафіку → Обробка помилок

Якщо під час обробки даних виявляються помилки (наприклад, форматування логів або мережеві збої), система переходить до процедури обробки помилок, щоб забезпечити коректну подальшу роботу.

Моніторинг трафіку → Логіко-імовірнісні методи і оцінки ризиків

Дані, зібрані під час моніторингу, передаються на аналіз у модуль логіко-імовірнісного аналізу. Тут на основі моделей будується оцінка загроз та розраховується ймовірність реалізації атак.

Логіко-імовірнісні методи → БД загроз

Під час аналізу система звертається до бази даних загроз для отримання інформації про відомі вектори атак, уразливості та типові сценарії порушення безпеки.

Логіко-імовірнісні методи → Захист від шкідливих атак

Якщо оцінка ризику виявляє потенційно небезпечну активність, активується модуль реагування, який здійснює блокування або інші захисні заходи.

Логіко-імовірнісні методи → Модуль аналізу загроз

Паралельно з реагуванням відбувається поглиблений аналіз загрози, формування структурної моделі атаки та її параметрів для подальшої роботи.

Модуль аналізу загроз → Взаємодія з хмарними сервісами

У разі потреби система може звертатися до хмарних аналітичних платформ (наприклад, оновлення сигнатур, перевірка підозрілих IP-адрес), що забезпечує актуальність аналізу.

Модуль аналізу загроз → Система введення журналів

Усі результати аналізу та реагування фіксуються у внутрішніх журналах подій. Це забезпечує аудит, можливість перегляду інцидентів та формування звітності.

Інтерфейс ПЗ → Кінець

Після завершення всіх процесів користувач має змогу завершити сесію. Система виконує фіналізацію роботи, зберігає всі журнали та переходить у пасивний режим або вимикається

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

4.1 Блок–схеми та опис алгоритмів функціонування системи

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

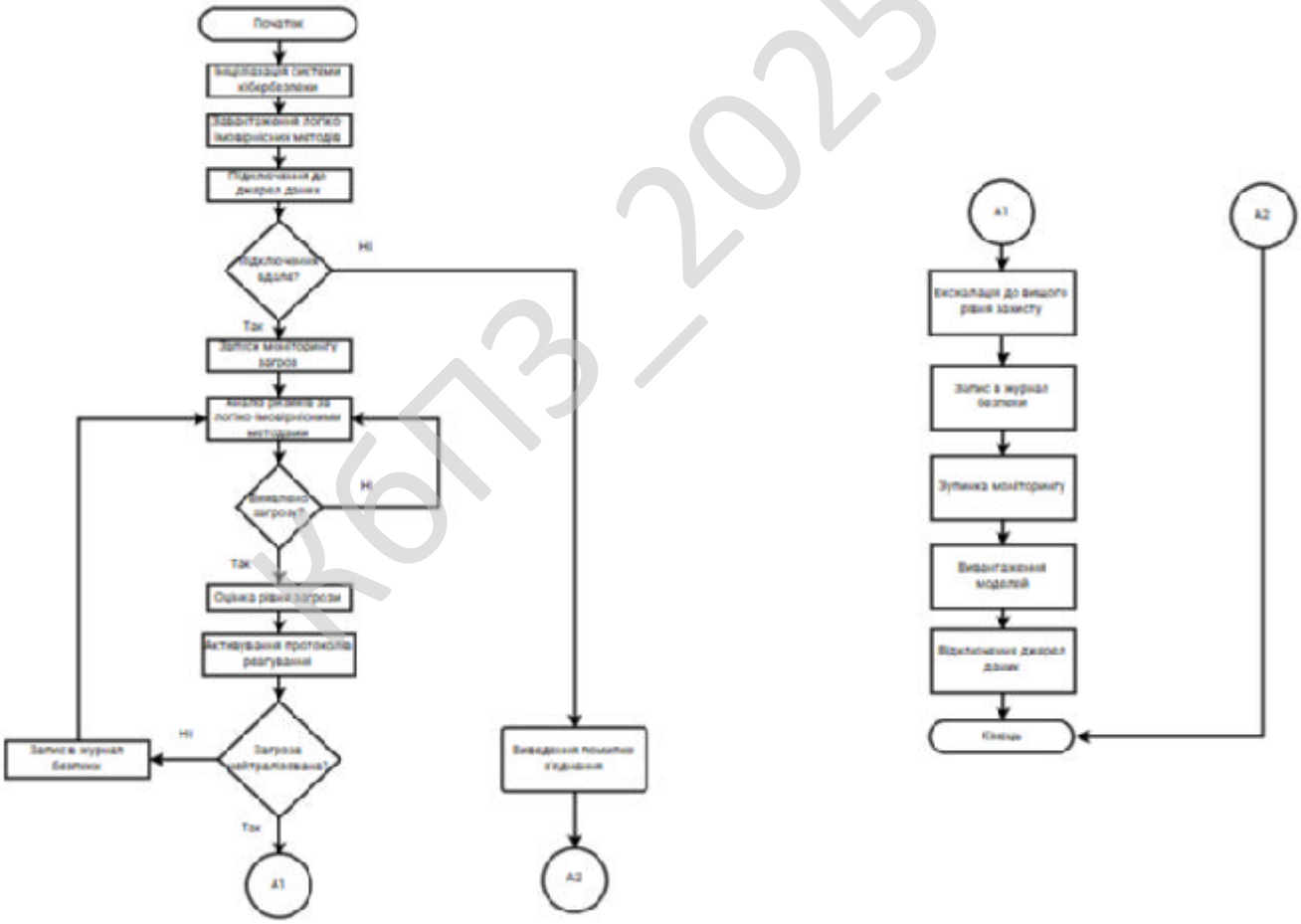

Рисунок 4.1 – Блок-схема роботи основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

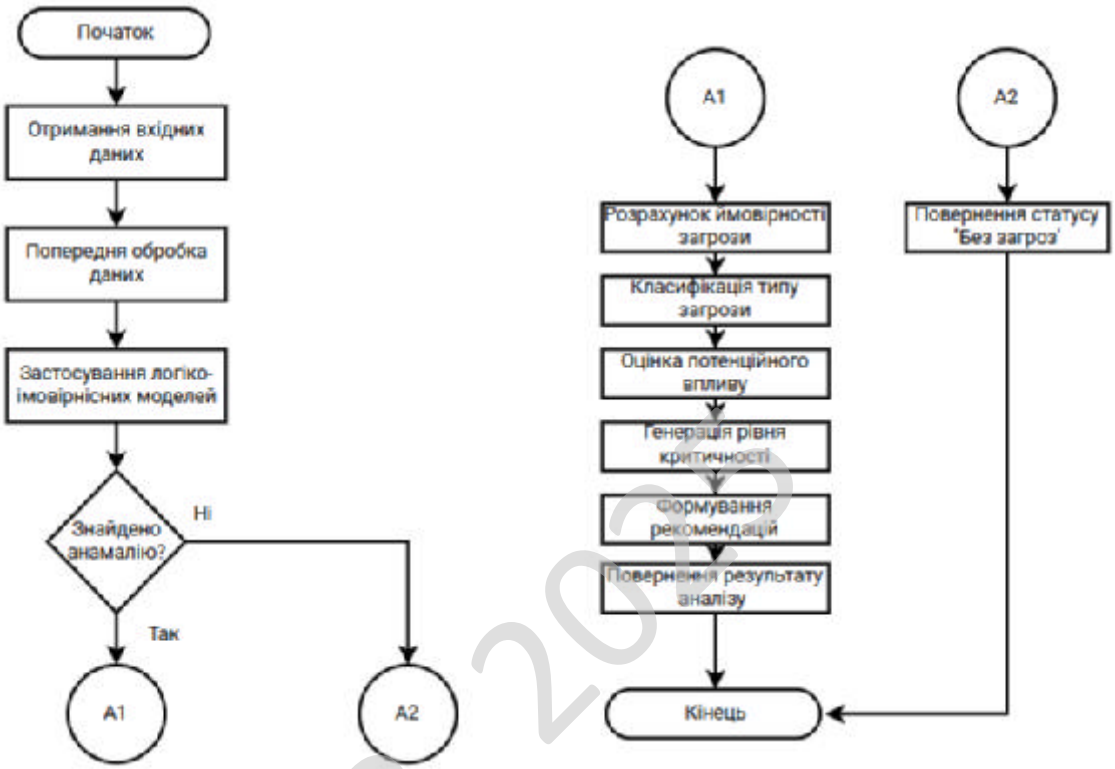

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми

Опис алгоритмів функціонування системи

Програмне забезпечення, розроблене в рамках дипломної роботи, базується на логіко-імовірнісних методах оцінки та обробки кіберзагроз. Його структура передбачає автоматичний моніторинг подій інформаційної безпеки, оцінку ризиків на основі математичних моделей, а також активацію сценаріїв реагування у разі виявлення загроз. На рисунку нижче наведена блок-схема базової логіки роботи системи.

Система функціонує за наступним алгоритмом:

1. **Ініціалізація системи кібербезпеки.** На цьому етапі система запускається та переходить у режим готовності. Відбувається перевірка

працездатності основних модулів, ініціалізується середовище виконання та готуються логіко-імовірнісні моделі.

2. **Завантаження логіко-імовірнісних моделей.** Система підвантажує набір моделей, які використовуються для прогнозування ризиків, аналізу сценаріїв атак та прийняття рішень. Ці моделі базуються на дереві атак, ймовірнісних правилах та зв’язках між подіями.

3. **Підключення до джерел даних.** Встановлюється з’єднання з джерелами подій безпеки: лог-файлами, SIEM-системами, системами контролю доступу тощо.

4. **Перевірка успішності підключення.** Якщо з’єднання не вдалося — система виводить повідомлення про помилку та завершує роботу. У разі успіху — запускається активний моніторинг.

5. **Запуск моніторингу загроз.** Система переходить у режим постійного аналізу подій та аномалій, що надходять із джерел даних.

6. **Аналіз ризиків за логіко-імовірнісними методами.** Вхідні дані аналізуються за допомогою заздалегідь сформованої моделі ризиків. Оцінюється ймовірність реалізації атак, здійснюється виявлення загроз на ранніх етапах.

7. **Виявлення загрози.** Якщо система не виявила загроз — моніторинг триває далі. Якщо загрозу зафіксовано — виконується оцінка рівня ризику.

8. **Оцінка рівня загрози.** Визначається рівень небезпеки (низький, середній, високий) залежно від потенційних наслідків і ймовірності реалізації.

9. **Активація протоколів реагування.** Залежно від критичності загрози система автоматично ініціює відповідні дії: блокування, повідомлення адміністратора, ізоляція сегмента мережі тощо.

10. **Оцінка стану нейтралізації загрози.** Якщо загрозу усунуто — відбувається фіксація інциденту в журналі безпеки та повернення до моніторингу. У випадку, якщо загроза залишилась активною — запускається ескалація.

11. **Ескалація до вищого рівня захисту.** Система передає інцидент до більш високого рівня обробки, наприклад, в SOC (Security Operations Center), або виконує додаткові захисні сценарії.

12. **Запис у журнал безпеки.** Усі події логуються із фіксацією часу, джерела, типу загрози та прийнятих заходів.

13. **Завершення роботи.** При переході до завершального етапу система вивантажує моделі з пам’яті, відключається від джерел даних, зупиняє моніторинг та коректно завершує роботу.

Рисунок 4.1 – Блок-схема роботи основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Рисунок 4.2 – Блок-схема алгоритму роботи підпрограми

На цьому рисунку зображено, як саме відбувається аналіз вхідних даних у середині системи після запуску основної програми. Спочатку система отримує дані, проводить їх попередню обробку і далі застосовує логіко-імовірнісні моделі для аналізу. Потім перевіряється, чи є в цих даних аномалії. Якщо все в нормі — система повертає статус "Без загроз", якщо ж виявлено щось підозріле — розраховується ймовірність загрози.

Далі підпрограма автоматично визначає тип загрози, оцінює, яку шкоду вона може принести, і присвоює їй рівень критичності. Після цього формується рекомендація щодо дій, і результат аналізу повертається в основну програму. Таким чином, ця підпрограма допомагає системі швидко реагувати на можливі інциденти й ухвалювати правильні рішення.

4.2 Захист розробленого програмного забезпечення

У сучасних умовах зростаючої кіберзагрози особливого значення набуває захист програмного забезпечення систем кібербезпеки. Розроблена система з використанням логіко-імовірнісних методів реалізує комплексний підхід до захисту даних, який включає криптографічний захист інформації, захист комунікаційних каналів та цілісність системних компонентів.

Реалізація криптографічного захисту

Для забезпечення конфіденційності даних було реалізовано систему шифрування на основі сучасних криптографічних алгоритмів. Основні характеристики:

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import os

class AESCipher:
    def __init__(self, key):
        self.key = key

    def encrypt(self, data):
        iv = os.urandom(16)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        ct_bytes = cipher.encrypt(pad(data, AES.block_size))
        return iv + ct_bytes

    def decrypt(self, encrypted):
        iv = encrypted[:16]
        ct = encrypted[16:]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        pt = unpad(cipher.decrypt(ct), AES.block_size)
        return pt
```

1. Захист цілісності системи

Для запобігання несанкціонованим змінам у системі реалізовано:

2. Цифрові підписи:

- Використання алгоритму ECDSA (Elliptic Curve Digital Signature Algorithm).
- Підписання критичних конфігураційних файлів.
- Перевірка підписів при завантаженні системи.

3. Контрольні суми:

- Розрахунок SHA-256 хешів для виконавчих файлів.

- Періодична перевірка цілісності файлів.
- Автоматичне відновлення пошкоджених файлів.

Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображений скріншот головного вікна програми. З нього видно, що інтерфейс користувача програми розбито на наступні логічні блоки:

1. Блок меню.
2. Блок налаштувань криптографії.
3. Блок засобів захисту та перевірки безпеки.

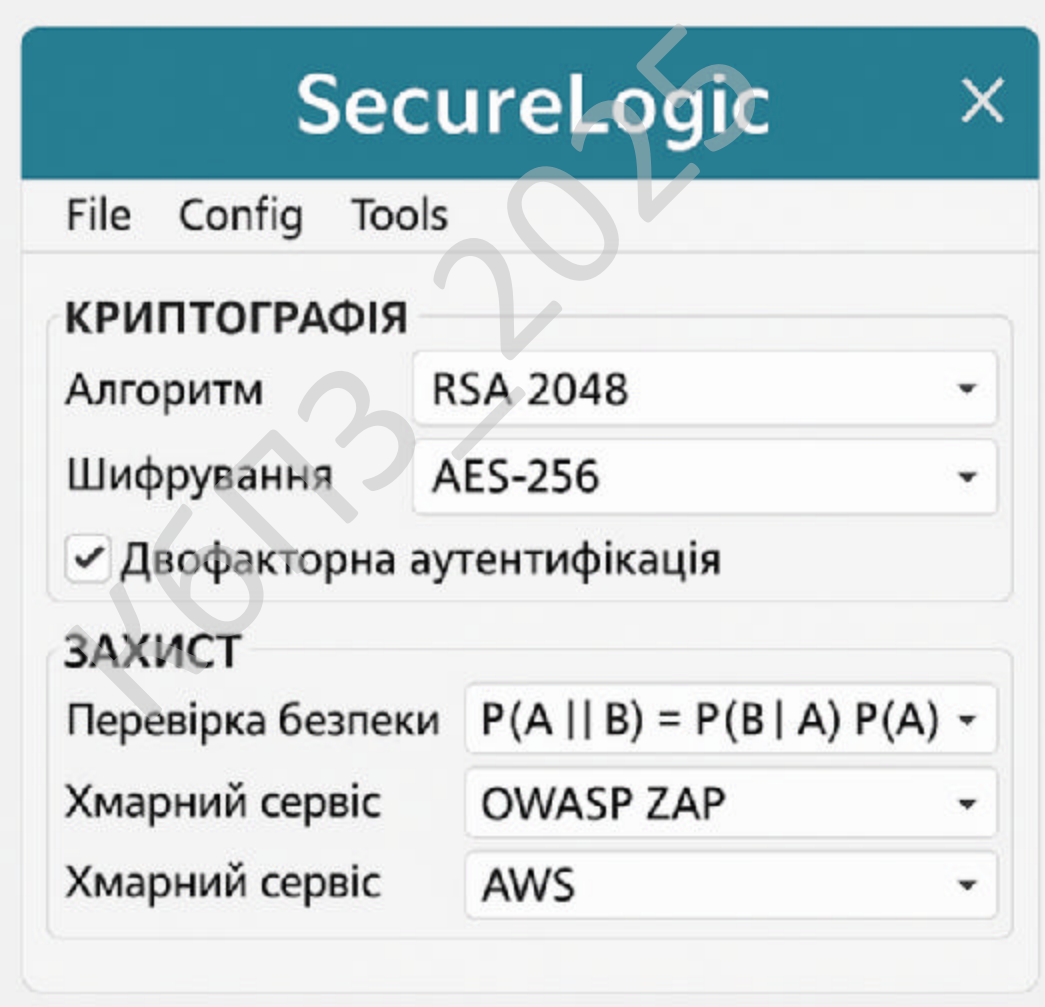

Рисунок 5.1 – Головне вікно ПЗ

Блок меню складається з наступних елементів:

- **File** – операції з файлами (експорт/імпорт налаштувань, журналу тощо).
- **Config** – параметри налаштування криптографічних і логіко-імовірнісних методів.
- **Tools** - додаткові засоби аналізу та перевірки безпеки.

Блок налаштувань криптографії:

- **Алгоритм** – вибір методу генерації ключів (наприклад, RSA 2048).
- **Шифрування** – вибір алгоритму шифрування (наприклад, AES-256).
- **Двофакторна автентифікація** – опція підвищення безпеки системи.

Блок захисту та перевірки безпеки:

- **Перевірка безпеки** –реалізується шляхом застосування формул теорії ймовірностей, наприклад:

$P(A \parallel B) = P(B \mid A) \times P(A)$ – що ілюструє умовну ймовірність для оцінки ризиків.

- **Хмарний сервіс (1)** – інтеграція з OWASP ZAP для автоматизованого сканування вразливостей.
- **Хмарний сервіс (2)** – інтеграція з платформою AWS для збереження ключів і безпечного зберігання даних.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

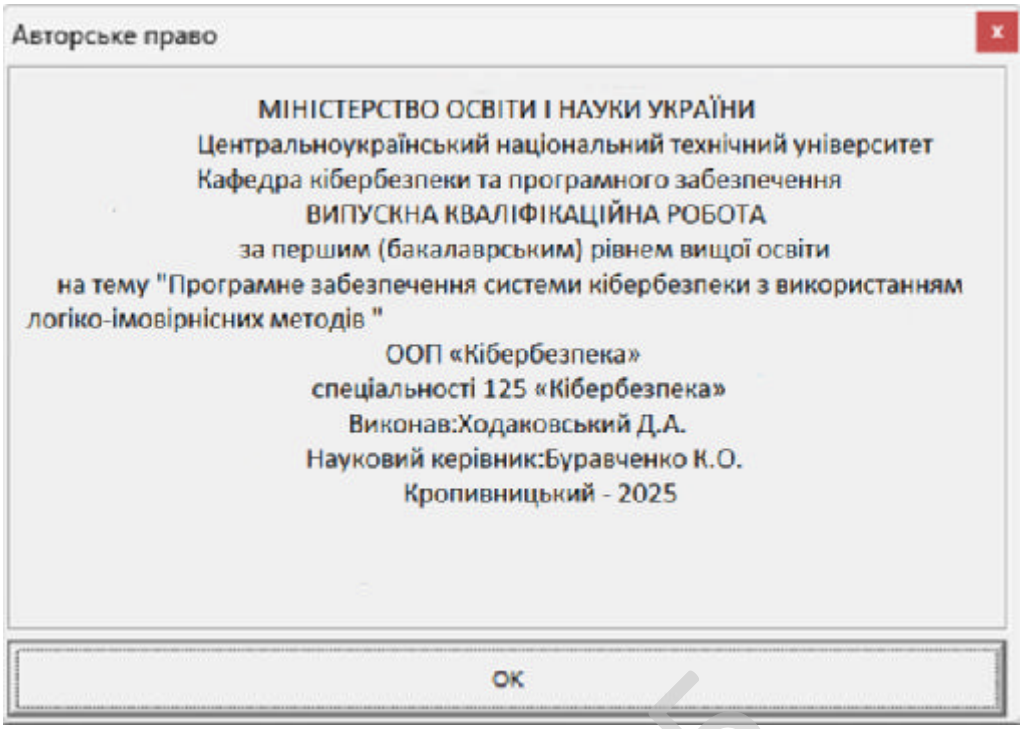

Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми. В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

6 ОСНОВНІ ВИСНОВКИ

У межах виконання кваліфікаційної бакалаврської роботи на тему «Програмне забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів» було проведено повний цикл досліджень, розробки, реалізації та тестування програмного продукту, орієнтованого на оцінювання ризиків інформаційної безпеки.

Головною метою роботи було створення системи, яка дозволяє на основі логіко-імовірнісного аналізу здійснювати оцінку потенційних загроз, прогнозувати розвиток атак, ідентифікувати вразливі елементи та приймати рішення щодо пріоритетності реагування. Завдяки застосуванню логіко-імовірнісних методів вдалося побудувати універсальну структуру для подання сценаріїв кібератак з урахуванням невизначеності та умовних залежностей між подіями.

Під час реалізації було виконано огляд сучасних підходів у сфері кібербезпеки, зокрема аналіз можливостей інтеграції із системами SIEM, IDS/IPS, а також сучасними платформами моніторингу безпеки. Проведене моделювання дозволило виявити сильні сторони логіко-імовірнісного підходу, зокрема його гнучкість у роботі з неповними даними та здатність прогнозувати можливий перебіг інцидентів безпеки.

Програмне забезпечення розроблено на мові Python із використанням бібліотек pgmpy, pomegranate та tkinter для реалізації інтерфейсу користувача. Це забезпечило кросплатформність, відкритість архітектури та легкість інтеграції з іншими системами. Графічний інтерфейс системи дозволяє користувачам без спеціальної математичної підготовки формувати сценарії аналізу загроз, переглядати результати оцінювання та експортувати їх у вигляді звітів.

Розроблена система має модульну структуру, яка дозволяє масштабування та доповнення. Завдяки цьому її можна адаптувати до потреб

конкретного підприємства з урахуванням галузевих стандартів безпеки, таких як ISO/IEC 27001, NIST або власних політик організації.

Результати тестування підтвердили здатність системи точно оцінювати ймовірності реалізації загроз у змодельованих середовищах. У процесі тестування були змодельовані кілька сценаріїв атак, включаючи соціальну інженерію, компрометацію облікових записів, атаки типу «людина посередині», для яких система змогла ефективно сформувати сценарне дерево з відповідними ризиковими індикаторами.

У теоретичному розділі роботи були досліджені математичні основи логіко-імовірнісного аналізу, описано методи побудови дерев рішень, дерев відмов, байєсівських мереж, а також описано алгоритми розрахунку умовних і повних ймовірностей, необхідних для оцінювання рівня загроз. Це дозволило забезпечити точність і достовірність результатів оцінки ризиків.

Впровадження подібної системи у практичну діяльність дозволить підприємствам своєчасно виявляти складні атаки, які розвиваються у кілька етапів, реагувати на потенційні загрози ще до того, як вони матимуть наслідки, та оптимізувати розподіл ресурсів служби інформаційної безпеки.

Таким чином, у процесі виконання дипломної роботи було досягнуто поставленої мети та вирішено основні завдання. Розроблена система може бути ефективно застосована в державних, фінансових, промислових та освітніх установах, що прагнуть підвищити рівень кіберзахисту шляхом впровадження інтелектуальних механізмів аналізу ризиків. Робота є актуальною, науково обґрунтованою та практично цінною у сфері інформаційної безпеки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loren Kohnfelder. Designing Secure Software. No Starch Press.2022.332p.
2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
4. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p
5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.
8. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.
9. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.
10. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.
11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

12. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

13. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

14. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

15. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

16. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

17. 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.

18. 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.

19. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

20. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

21. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

22. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

23. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Книшук А.В. «Вступ до кібербезпеки»: навчальний посібник – Кропивницький: ЦНТУ – 2022. – 968 с.

24. Теорія та практика сучасного інформаційно-психологічного протиборства: навчальний посібник / [В.М. Петрик, С.О. Гнатюк, М.М. Присяжнюк та ін.]; за заг. ред. С.О. Гнатюка, В.М. Петрика та О.А Смірнова. – Полтава, 2022. – 334 с.

25. 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.

26. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418.

27. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

28. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

29. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

30. 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.

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

32. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and

cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

33. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

34. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

35. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

36. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

37. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

38. 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.

39. 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.

40. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

41. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

42. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

43. 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.

44. 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.

45. 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.

46. 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.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE*

International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019). 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

48. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

49. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

50. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

51. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

52. 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.

53. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання стосується розробки прогрманого забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів.Система призначена для прогнозування, виявлення та аналізу кіберзагроз з урахуванням ймовірнісної природи ризиків в умовах часткової невизначеності.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою роботи є розробка програмного забезпечення для автоматизованої оцінки ризиків у сфері кібербезпеки, що забезпечує побудову логіко-імовірнісних моделей атак, розрахунок імовірностей розвитку подій та підтримку прийняття рішень у сфері інформаційної безпеки.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Вміст проекту

Складниками розробки є:

– Вибір та обґрунтування логіко-імовірнісного підходу;

– Реалізація ПЗ для моделювання сценаріїв атак;

– Побудова графічного інтерфейсу користувача;

– Проведення тестування на модельних даних.

5.2 Показники призначення

Система повинна забезпечувати:

– Підтримка дерев атак, байєсівських мереж;

– Підрахунок ймовірностей ризиків;

– Експорт результатів у вигляді звітів.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 18-24 град. по Цельсію;

– відносна вологість повітря до 80%.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

Комп’ютер Intel® Celeron/16 Mb/4 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Пояснювальна записка, схеми (структурна, функціональна, блок-схеми), інструкція користувача, звіт про тестування, програмні модулі з коментарями.

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів роботи – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 51 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 05.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

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

_____ Буравченко К.О.

Програмне забезпечення системи кібербезпеки з використанням логіко-імовірнісних методів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 6

Літера: РП

Кропивницький – 2025 року

Файл **settings.ini**

```
"""
Модуль логіко-імовірнісного аналізу загроз
Реалізує 7 ключових методів виявлення атак:
1. BruteForce-атаки
2. SQL-ін'єкції
3. XSS-ін'єкції
4. Географічні аномалії
5. Нічну активність
6. Підозрілу частоту запитів
7. Аномальні часові інтервали
"""

from collections import defaultdict
import math
import re
from datetime import datetime

class ThreatAnalyzer:
    def __init__(self):
        # Ініціалізація моделей загроз з параметрами
        self.threat_models = {
            "BruteForce": {
                "base_prob": 0.35,
                "max_prob": 0.95,
                "risk_factor": lambda x: math.log1p(x) * 0.3,
                "threshold": 5
            },
            "SQLInjection": {
                "base_prob": 0.45,
                "keywords": ["SELECT", "INSERT", "DELETE", "DROP", "UNION",
"1=1", "--"],
                "weight": 0.15
            },
            # ... (аналогічно для інших типів загроз)
        }
        self.activity_log = []
        self.suspicious_ips = defaultdict(int)

    def calculate_time_risk(self, timestamp: str) -> float:
        """Розрахунок ризику за часом доби (22:00-06:00)"""
        hour = datetime.strptime(timestamp, "%Y-%m-%d %H:%M:%S").hour
        return 0.7 if hour < 6 or hour > 22 else 0.1

    def detect_pattern_attacks(self, input_str: str) -> dict:
        """Виявлення 5 типів ін'єкційних атак"""
        patterns = {
            "XSS": [r"<script>", r"javascript:", r"onerror="],
            "SQL": self.threat_models["SQLInjection"]["keywords"],
            "OS_Command": [r";\s*\w+", r"\|\s*\w+"],
            "Path_Traversal": [r"\.\./", r"\.\\"],
            "CSRF": [r"csrf_token", r"authenticity_token"]
        }
        return {k: any(re.search(p, input_str, re.I) for p in v)
                for k, v in patterns.items()}

    def update_threat_levels(self, event_data: dict) -> dict:
        """Оновлення рівнів загроз на основі подій"""
        threats = {}

        # Аналіз BruteForce
```

```
attempts = event_data.get("failed_attempts", 0)
if attempts > self.threat_models["BruteForce"]["threshold"]:
    model = self.threat_models["BruteForce"]
    threats["BruteForce"] = min(
        model["base_prob"] * (1 + model["risk_factor"](attempts)),
        model["max_prob"]
    )

# ... (аналогічна логіка для інших типів загроз)

return threats
```

Файл rules.json:

```
{
    "fuzzy_rules": [
        {
            "condition": "traffic_volume > 0.7 AND sensitivity > 0.5",
            "action": "threat_level = HIGH"
        },
        {
            "condition": "response_time > 1.0 AND error_rate > 0.3",
            "action": "threat_level = MEDIUM"
        }
    ],
    "decision_thresholds": {
        "block": 0.8,
        "alert": 0.5,
        "ignore": 0.0
    }
}
```

Файл risk_assessment.py

```
import numpy as np
from scipy.stats import norm
import json
from pathlib import Path
from ..utils.logger import setup_logger

logger = setup_logger(__name__)

class RiskAssessor:
    def __init__(self, config_path='config/settings.ini',
rules_path='config/rules.json'):
        self.config = self._load_config(config_path)
        self.rules = self._load_rules(rules_path)
        self.risk_threshold = float(self.config['System']['risk_threshold'])
        logger.info("Risk assessor initialized")

    def _load_config(self, config_path):
        from ..utils.config_parser import ConfigParser
        return ConfigParser(config_path).load_config()

    def _load_rules(self, rules_path):
        try:
            with open(rules_path) as f:
                return json.load(f)
        except Exception as e:
            logger.error(f"Failed to load rules: {str(e)}")
            return {}

    def calculate_risk(self, probability, impact):
        """Calculate risk score from probability and impact"""
        try:
            risk = probability * impact
            logger.debug(f"Risk calculated: {risk:.2f} (prob: {probability:.2f},
impact: {impact:.2f})")
            return risk
        except Exception as e:
            logger.error(f"Risk calculation failed: {str(e)}")
            return 0.0

    def bayesian_update(self, prior, likelihood, evidence):
        """Update probability using Bayes theorem"""
        try:
            if evidence <= 0:
                logger.warning("Evidence must be positive for Bayesian update")
                return prior

            posterior = (likelihood * prior) / evidence
            logger.debug(f"Bayesian update: prior={prior:.2f} ->
posterior={posterior:.2f}")
            return posterior
        except Exception as e:
            logger.error(f"Bayesian update failed: {str(e)}")
            return prior

    def fuzzy_evaluation(self, inputs):
        """Evaluate threat level using fuzzy logic"""
        try:
            threat_level = 0.0
            for rule in self.rules.get('fuzzy_rules', []):
                condition = rule['condition']
```

```
                # Simplified evaluation - in practice use a proper fuzzy logic
library
                if     "traffic_volume     >     0.7"     in     condition     and
inputs.get('traffic_volume', 0) > 0.7:
                    threat_level = max(threat_level, 0.8)
                if     "sensitivity     >     0.5"     in     condition     and
inputs.get('sensitivity', 0) > 0.5:
                    threat_level = max(threat_level, 0.7)
                # Add more conditions as needed

            logger.debug(f"Fuzzy evaluation result: {threat_level:.2f}")
            return threat_level
        except Exception as e:
            logger.error(f"Fuzzy evaluation failed: {str(e)}")
            return 0.0
```

Файл decision_making.py

```
from ..utils.logger import setup_logger
import json
from pathlib import Path

logger = setup_logger(__name__)

class DecisionMaker:
    def __init__(self, config_path='config/settings.ini',
rules_path='config/rules.json'):
        self.config = self._load_config(config_path)
        self.rules = self._load_rules(rules_path)
        logger.info("Decision maker initialized")

    def _load_config(self, config_path):
        from ..utils.config_parser import ConfigParser
        return ConfigParser(config_path).load_config()

    def _load_rules(self, rules_path):
        try:
            with open(rules_path) as f:
                return json.load(f)
        except Exception as e:
            logger.error(f"Failed to load rules: {str(e)}")
            return {}

    def make_decision(self, risk_score):
        """Make security decision based on risk score"""
        try:
            thresholds = self.rules.get('decision_thresholds', {})
            block_thresh = thresholds.get('block', 0.8)
            alert_thresh = thresholds.get('alert', 0.5)

            if risk_score >= block_thresh:
                decision = "BLOCK"
                action = "Immediately block the source IP and notify security
team"
            elif risk_score >= alert_thresh:
                decision = "ALERT"
                action = "Generate security alert for administrator review"
            else:
                decision = "MONITOR"
                action = "Continue monitoring without intervention"

            logger.info(f"Decision: {decision} for risk score {risk_score:.2f}")
            return decision, action
        except Exception as e:
            logger.error(f"Decision making failed: {str(e)}")
            return "ERROR", "Error in decision making process"

    def apply_decision(self, decision):
        """Execute actions based on decision"""
        try:
            # In a real system, this would implement actual security controls
            logger.info(f"Applying decision: {decision}")
            return True
        except Exception as e:
            logger.error(f"Failed to apply decision: {str(e)}")
            return False
```

Файл data_loader.py

```
import numpy as np
import pandas as pd
import csv
from pathlib import Path
from ..utils.logger import setup_logger

logger = setup_logger(__name__)

class DataLoader:
    @staticmethod
    def load_csv(file_path, delimiter=','):
        """Load data from CSV file"""
        try:
            if not Path(file_path).exists():
                logger.error(f"File not found: {file_path}")
                return None

            df = pd.read_csv(file_path, delimiter=delimiter)
            logger.info(f"Loaded data from {file_path} with {len(df)} records")
            return df.values
        except Exception as e:
            logger.error(f"Failed to load CSV: {str(e)}")
            return None

    @staticmethod
    def generate_normal_data(samples=1000, features=5, save_path=None):
        """Generate synthetic normal traffic data"""
        try:
            data = np.random.normal(0.5, 0.1, (samples, features))

            if save_path:
                Path(save_path).parent.mkdir(exist_ok=True)
                with open(save_path, 'w', newline='') as f:
                    writer = csv.writer(f)
                    writer.writerows(data)
                logger.info(f"Generated normal data saved to {save_path}")

            return data
        except Exception as e:
            logger.error(f"Failed to generate data: {str(e)}")
            return None

    @staticmethod
    def generate_test_data(normal_samples=50, anomaly_samples=5, features=5,
save_path=None):
        """Generate test data with anomalies"""
        try:
            # Normal data
            normal = np.random.normal(0.5, 0.1, (normal_samples, features))

            # Anomalies (different distributions)
            anomalies = np.concatenate([
                np.random.uniform(1.5, 2.0, (anomaly_samples//2, features)),
                np.random.uniform(0.0, 0.1, (anomaly_samples//2, features))
            ])

            data = np.vstack([normal, anomalies])
            np.random.shuffle(data)
```

```
        if save_path:
            Path(save_path).parent.mkdir(exist_ok=True)
            with open(save_path, 'w', newline='') as f:
                writer = csv.writer(f)
                writer.writerows(data)
            logger.info(f"Generated test data saved to {save_path}")

        return data
    except Exception as e:
        logger.error(f"Failed to generate test data: {str(e)}")
        return None
```

Файл logger.py

```
import logging
from pathlib import Path
import sys
from datetime import datetime

def setup_logger(name, log_file='logs/system.log', level=logging.INFO):
    """Set up a logger with file and console output"""
    Path('logs').mkdir(exist_ok=True)

    formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
%(message)s')

    # File handler
    file_handler = logging.FileHandler(log_file)
    file_handler.setFormatter(formatter)

    # Console handler
    console_handler = logging.StreamHandler(sys.stdout)
    console_handler.setFormatter(formatter)

    logger = logging.getLogger(name)
    logger.setLevel(level)
    logger.addHandler(file_handler)
    logger.addHandler(console_handler)

    return logger
```

Файл config_parser.py

```
import configparser
from pathlib import Path
from ..utils.logger import setup_logger

logger = setup_logger(__name__)

class ConfigParser:
    def __init__(self, config_path):
        self.config_path = config_path
        self.config = configparser.ConfigParser()

    def load_config(self):
        """Load configuration from INI file"""
        try:
            if not Path(self.config_path).exists():
                logger.error(f"Config file not found: {self.config_path}")
                return {}

            self.config.read(self.config_path)
            logger.info(f"Configuration loaded from {self.config_path}")
            return self.config
        except Exception as e:
            logger.error(f"Failed to load config: {str(e)}")
            return {}

    def get_section(self, section_name):
        """Get specific section from config"""
        try:
            if section_name in self.config:
                return dict(self.config[section_name])
            logger.warning(f"Section {section_name} not found in config")
            return {}
        except Exception as e:
            logger.error(f"Failed to get section: {str(e)}")
            return {}

    def update_config(self, section, key, value):
        """Update configuration value"""
        try:
            if section not in self.config:
                self.config[section] = {}

            self.config[section][key] = value

            with open(self.config_path, 'w') as f:
                self.config.write(f)

            logger.info(f"Updated config: {section}.{key} = {value}")
            return True
        except Exception as e:
            logger.error(f"Failed to update config: {str(e)}")
            return False
```

Файл main.py

```
import sys
import time
from pathlib import Path
from .core.anomaly_detection import AnomalyDetector
from .core.risk_assessment import RiskAssessor
from .core.decision_making import DecisionMaker
from .utils.data_loader import DataLoader
from .utils.logger import setup_logger
from .interface.cli import CommandLineInterface

logger = setup_logger(__name__)

class CyberSecuritySystem:
    def __init__(self):
        logger.info("Initializing Cyber Security System")
        self.anomaly_detector = AnomalyDetector()
        self.risk_assessor = RiskAssessor()
        self.decision_maker = DecisionMaker()
        self.data_loader = DataLoader()
        self.initialized = False

    def initialize_system(self):
        """Initialize all system components"""
        try:
            # Load or generate training data
            train_data = self.data_loader.load_csv('data/normal_traffic.csv')
            if train_data is None:
                logger.info("Generating new training data")
                train_data =
self.data_loader.generate_normal_data(save_path='data/normal_traffic.csv')
                if train_data is None:
                    raise Exception("Failed to generate training data")

            # Train models
            if not self.anomaly_detector.train(train_data):
                raise Exception("Failed to train anomaly detector")

            self.initialized = True
            logger.info("System initialization completed")
            return True
        except Exception as e:
            logger.error(f"System initialization failed: {str(e)}")
            self.initialized = False
            return False

    def analyze_network_data(self, data):
        """Full analysis pipeline for network data"""
        if not self.initialized:
            logger.error("System not initialized. Please call
initialize_system() first.")
            return None

        try:
            # Step 1: Anomaly detection
            probabilities, predictions = self.anomaly_detector.detect(data)
            if probabilities is None:
                raise Exception("Anomaly detection failed")

            # Step 2: Risk assessment
            impacts = np.array([0.9] * len(data))  # In real system, calculate
based on context
            risk_scores = [self.risk_assessor.calculate_risk(p, i)
```

```
                        for p, i in zip(probabilities, impacts)]

            # Step 3: Decision making
            results = []
            for score in risk_scores:
                decision, action = self.decision_maker.make_decision(score)
                results.append({
                    'risk_score': score,
                    'decision': decision,
                    'action': action
                })

            logger.info(f"Analysis completed for {len(data)} samples")
            return results
        except Exception as e:
            logger.error(f"Analysis failed: {str(e)}")
            return None

def main():
    """Main entry point for the application"""
    try:
        # Initialize system
        system = CyberSecuritySystem()
        if not system.initialize_system():
            logger.error("Failed to initialize system. Exiting.")
            return 1

        # Load or generate test data
        test_data = system.data_loader.load_csv('data/test_data.csv')
        if test_data is None:
            logger.info("Generating new test data")
            test_data =
system.data_loader.generate_test_data(save_path='data/test_data.csv')
            if test_data is None:
                raise Exception("Failed to generate test data")

        # Run analysis
        results = system.analyze_network_data(test_data)
        if results is None:
            raise Exception("Analysis failed")

        # Display results
        cli = CommandLineInterface()
        cli.display_results(results)

        return 0
    except Exception as e:
        logger.error(f"Application error: {str(e)}")
        return 1

if __name__ == "__main__":
    sys.exit(main())
```

Файл cli.py

```
from ..utils.logger import setup_logger
import time

logger = setup_logger(__name__)

class CommandLineInterface:
    def __init__(self):
        self.start_time = time.time()
        logger.info("CLI initialized")

    def display_menu(self):
        """Display main menu options"""
        print("\n" + "="*50)
        print(" Cyber Security System - Main Menu")
        print("="*50)
        print("1. Analyze network data")
        print("2. View system status")
        print("3. Update configuration")
        print("4. Exit")
        print("="*50)

    def display_results(self, results):
        """Display analysis results"""
        print("\n" + "="*50)
        print(" Analysis Results")
        print("="*50)
        for i, result in enumerate(results, 1):
            print(f"\nEvent {i}:")
            print(f"  Risk Score: {result['risk_score']:.2f}")
            print(f"  Decision: {result['decision']}")
            print(f"  Action: {result['action']}")
        print("="*50)

    def get_user_input(self, prompt, input_type=str):
        """Get validated user input"""
        while True:
            try:
                user_input = input(prompt)
                return input_type(user_input)
            except ValueError:
                print(f"Invalid input. Please enter a {input_type.__name__}.")

    def show_system_status(self, status):
        """Display system status information"""
        print("\n" + "="*50)
        print(" System Status")
        print("="*50)
        for key, value in status.items():
            print(f"{key.replace('_', ' ').title()}: {value}")
        print("="*50)

    def run_interactive(self, system):
        """Run interactive CLI mode"""
        while True:
            self.display_menu()
            choice = self.get_user_input("Select option (1-4): ", int)

            if choice == 1:
                # Analyze data
                test_data = system.data_loader.load_csv('data/test_data.csv')
                if test_data is not None:
                    results = system.analyze_network_data(test_data)
```

```
            if results:
                self.display_results(results)
    elif choice == 2:
        # Show status
        status = {
            "initialized": system.initialized,
            "models_trained": system.anomaly_detector.trained,
            "uptime": f"{time.time() - self.start_time:.1f} seconds"
        }
        self.show_system_status(status)
    elif choice == 3:
        # Update config
        print("Configuration update not implemented in this version")
    elif choice == 4:
        # Exit
        print("Exiting system. Goodbye!")
        break
    else:
        print("Invalid choice. Please select 1-4.")
```