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 інформаційних ресурсів автоматизованих систем спеціального призначення”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Устинович М.Є.
«____» _____ 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

Устиновичу Максиму Євгеновичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

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
_____ Устинович М.Є.
(прізвище та ініціали)

АНОТАЦІЯ

Устинович М.Є. Програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

Метою розробки є програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

Результат роботи – програмна реалізація системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, автоматизовані системи спеціального призначення

ABSTRACT

Ustinovych M.E. Software for the cybersecurity system of information resources of automated special-purpose systems. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the cybersecurity system of information resources of automated special-purpose systems.

The purpose of the development is the software for the cybersecurity system of information resources of automated special-purpose systems.

The result of the work is the software implementation of the cybersecurity system of information resources of automated special-purpose systems.

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

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

The program can be used on PCs with Windows 10/11.

The program is developed in Python.

Keywords: cybersecurity, automated special-purpose systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АСК	–	Автоматизовані системи керування
ТЗІ	–	Технічний захист інформації
ТКМ	–	Телекомунікаційні мережі

ВСТУП

Актуальність теми. Кібербезпека для інформаційних технологій (IT)/операційних технологій (OT) стосується захисту мереж Інтернет-протоколу (IP) від кібератак. Кібербезпека системи керування – це захист фізичних процесів від ненавмисних інцидентів і зловмисних атак. Технологічно кібербезпека системи керування відрізняється від кібербезпеки IT через пристрої системи керування та їх протоколи зв'язку низького рівня. Проте політика кібербезпеки IT та OT була розроблена організацією мережевої безпеки за мінімальної участі інженерних організацій, які «володіють» апаратним забезпеченням і системами керування. Кібербезпека системи контролю реальна – на сьогоднішній день виявлено понад 1250 фактичних інцидентів. Але в даний час широко поширена нестача відповідної кіберкриміналістики системи контролю та навчання кібербезпеці. Завдяки наявності обладнання для кібербезпеки IT, тестування та навчання, IT-системи продовжують бути скомпрометованими, а кібербезпека системи контролю відстає від IT на 5-10 років. Окрім необхідності модернізації системи контролю кібербезпеки на рівнях окремих організацій та критичних інфраструктур, це питання важливості національної безпеки. Широко повідомлялося, що великий електричний трансформатор китайського виробництва міг містити апаратні бекдори, які дозволяли отримати доступ до параметрів керування трансформаторним обладнанням.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

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

– Огляд існуючих систем інформаційних ресурсів автоматизованих систем спеціального призначення.

– Дослідження системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

– Програмна реалізація системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інформаційних ресурсів автоматизованих систем спеціального призначення.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Автоматизовані системи управління контролюють і керують роботою фізичних активів і процесів, таких як енергетичні системи, нафтопереробні заводи, трубопроводи, водопостачання та стичні води, хімічні заводи, стратегічні галузі промисловості, виробництво, контроль будівель, медичні установи та інші критичні інфраструктури. (Як видно з цього списку, система управління використовується не тільки в «промислових» застосуваннях.) Операційна технологія (ОТ)/системи керування належать до апаратного та програмного забезпечення, які ви бачите або змінюєте безпосередньо моніторинг та/або контроль промислового, виробничого і комерційного обладнання, активів, процесів та подій. Що робить систему управління кібербезпекою, відмінною від кібербезпеки ІТ, є її головними пріоритетами захисту життя та фізичної власності.

Це стосується рівня невеликих промислових або виробничих цехів, а також диспетчерського комунального підприємства, яке постачає електроенергію до кількох штатів. З 1970-х до середини 1990-х років системи керування та їх супроводжуючі процеси та інші польові пристрої, такі як приводи, приводи та хімічні аналізатори, не були підключені до зовнішнього світу. Вони діяли під призначенням поглядів інженерів, які проектували, експлуатували та обслуговували ці системи. де-знаки та експлуатаційні вимоги підкреслюють продуктивність і безпеку, а не су-безпека. Системи управління та «тупі датчики», які контролюють обладнання згенерованих даних, корисних тільки для інженерів, операцій та технічного обслуговування.

Пришестя микропроцесорів наприкінці 1960-х років і закон Мура дозволив розрахувати іможливість перетворення, щоб брати 0 і 1 інженерних

даних і перетворювати їх до інформації, яка може бути використана багатьма сторонами за межами організації поділу. Сама доступність цих цінних даних призвела до запиту для мережевих інженерних операцій, а часто й для мереж компаній. Це забезпечило продуктивність удосконалення, так як робота «точно вчасно» шляхом обміну даними між кількома організаційними компонентами. Інтернет і сучасні мережеві технології дозволили розповсюдження інформації та дистанційного керування системою

1.2 Область застосування

Областю застосування системи є автоматизовані системи. Інформаційні потоки системи контролю не зосереджені на кібербезпеці. Еталонна модель Пердю7 була розроблена в 1990-х роках, щоб гарантувати, що продуктивність системи керування в реальному часі не вплинула, а також для уточнення інформації має надходити із заводу. Він базувався на існуючій техніціології, яка узгоджується з обмеженими можливостями датчиків, контролерів, процесорів керування тощо. З революцією мікропроцесора та зв'язку, рівні еталонної моделі вже не такі прості. Ці технології дозволити датчикам процесу також мати програмовані логічні контролери (ПЛК) і навіть можливості комунікаційного шлюзу. Пристрої рівня 0, 1, які використовують повсюдно усі фізичні інфраструктури, не є кіберзахищеними. Насправді деякі прилади мережі інструментів низького рівня можуть не бути захищеними

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

У 2025 році штучний інтелект (AI) і машинне навчання (ML) змінюють можливості брандмауерів наступного покоління (NGFW). Ці технології дозволяють точно виявляти та блокувати загрози нульового дня, автоматизуючи важливі завдання, такі як аналіз трафіку та виявлення аномалій. Використовуючи AI та ML, організації можуть активно боротися з кіберризиками, випереджаючи загрози, що розвиваються. Сучасні брандмауери, керовані API, дозволяють командам скорочувати MTTR, забезпечуючи захист компаній у режимі реального часу.

Внутрішня безпека в хмарі залишається пріоритетом, оскільки організації все частіше використовують багатохмарні та гібридні хмарні архітектури. Сучасні NGFW тепер пропонують розширені функції, такі як динамічне застосування політики, безпека контейнерів і повна інтеграція з хмарними конструкціями, такими як VWAN Azure та Kubernetes (k8s). Ці можливості забезпечують послідовний і ефективний захист у різних і гібридних інфраструктурах. Крім того, масштабовані та гнучкі рішення забезпечують уніфікований підхід до захисту як локальних, так і хмарних ресурсів. Повернення контролю над вашими даними та керуванням доступом у всіх сферах.

Безпека операційних технологій (OT) також спостерігає сплеск інновацій NGFW. Окрім адаптації апаратного забезпечення для жорстких і промислових середовищ, NGFW тепер підтримують спеціальні сценарії використання OT із такими розширеними функціями, як перевірка протоколу, виявлення загроз на основі поведінки та мікросегментація. Ці можливості задовольняють унікальні

потреби критично важливих галузей промисловості, таких як виробництво, енергетика та транспорт, дозволяючи організаціям захищати промислові системи управління (ICS) і запобігати збоям у основних операціях.

Безпека IoT є ще однією важливою сферою уваги. NGFW розвиваються, щоб вирішувати унікальні проблеми, пов'язані з величезними екосистемами Інтернету речей. Розширені функції, такі як видимість пристрою, мікросегментація та поведінкова аналітика, допомагають захистити кінцеві точки Інтернету речей від таких загроз, як ботнети та DDoS-атаки. Завдяки інтеграції принципів нульової довіри та інспекції зашифрованого трафіку організації можуть з упевненістю забезпечити розгортання IoT у різних галузях.

Нарешті, аналітика поведінки об'єктів користувача (UEBA) розширює можливості NGFW для виявлення внутрішніх загроз і підозрілої активності. Аналізуючи моделі поведінки та позначаючи аномалії, NGFW дають змогу організаціям ефективно реагувати на такі ризики, як несанкціонований доступ до даних або незвичайна діяльність користувачів.

Ці досягнення підкреслюють вирішальну роль NGFW у навігації у дедалі складнішому ландшафті кібербезпеки. Щоб випереджати загрози, що розвиваються, організації повинні надавати пріоритет рішенням, оснащеним цими передовими функціями, забезпечуючи комплексний захист до 2025 року та надалі. Крім нових функцій, новіші моделі також можуть похвалитися новою архітектурою обробки та оптимізованою кодовою базою, завдяки чому вам більше не доведеться жертвувати швидкістю заради безпеки.

Брандмауери служать життєво важливим захистом від широкого спектру кіберзагроз, включаючи програми-вимагачі, віруси, хробаки, трояни та рекламне ПЗ. Брандмауери наступного покоління розширюють можливості традиційних брандмауерів, не лише виявляючи зловмисне програмне забезпечення, але й блокуючи його до того, як воно проникне в мережу. Завдяки вдосконаленим функціям виявлення загроз і реагування на них, NGFW забезпечують комплексний захист у центрах обробки даних, корпоративних мережах і хмарних

середовищах, зміцнюючи свою роль як наріжного каменю сучасних стратегій кібербезпеки.

Для посилення організаційної безпеки необхідна регулярна перевірка можливостей NGFW. Ескалація складності кібератак спонукає підприємства приймати NGFW, заохочуючи постачальників постійно впроваджувати інновації та вдосконалювати свої пропозиції. Сучасні NGFW мають такі важливі можливості, як розпізнавання додатків, централізоване керування та глибока перевірка пакетів (DPI). Крім того, багато з них надають розширені функції, включаючи захист від загроз 100 Гбіт/с і безпеку гібридної хмари, щоб протистояти загрозам, які обходять традиційний захист периметра.

Використовуючи передові технології запобігання загрозам, NGFW оснащують групи IT-безпеки інструментами для захисту від шкідливих програм, спроб вторгнень та інших складних атак. Ці розширені можливості роблять NGFW незамінним компонентом екосистеми кібербезпеки будь-якої організації.

Необхідні функції рішення NGFW

Постачальники NGFW пропонують різноманітні рішення, розроблені для задоволення різноманітних організаційних потреб, включаючи фізичні, віртуальні та контейнерні брандмауери.

– Фізичні брандмауери добре підходять для організацій будь-якого розміру, від малих і середніх кампусів до великих корпоративних центрів обробки даних.

– Віртуальні брандмауери розроблені для таких хмарних середовищ, як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, IBM Cloud, Oracle Cloud і приватних хмар, пропонуючи гнучкість, необхідну для хмарних операцій.

– Контейнерні брандмауери обслуговують додатки, що працюють у контейнерних середовищах, забезпечуючи покращену видимість і захист у середовищах виконання контейнерів і розгортаннях малого форм-фактора, як-от компоненти інфраструктури з підтримкою докерів.

Незалежно від вашої інфраструктури існує рішення NGFW, розроблене для задоволення ваших конкретних вимог безпеки.

Найкращі NGFW мають розширені функції, такі як системи виявлення та запобігання вторгненням (IDS/IPS), веб-проксі та фільтрування URL-адрес, а також брандмауер веб-додатків (WAF). Вони також підтримують віртуальне розгортання для хмарних середовищ і можуть легко захищати філії або мережі окремого підприємства. Ці функції мають вирішальне значення для захисту від сучасних загроз, одночасно покращуючи видимість програм і мережі.

Порівнюючи рішення NGFW, ключові фактори, які слід враховувати, включають:

– Продуктивність.

– Інтеграція безпеки сторонніх розробників.

– Простота використання.

– Ефективність блокування загроз.

– Ціноутворення та моделі споживання.

– Додаткові функції, такі як видимість програм, безпека гібридної хмари та централізоване керування.

Оцінюючи ці фактори, ви можете вибрати NGFW, який найкраще відповідає цілям безпеки та вимогам до інфраструктури вашої організації.

Рішення в цій роботі подано без певного порядку та відображають думки наших експертів. Це не слід розглядати як остаточне керівництво чи схвалення пріоритетності лише цих рішень. Кожна організація має свої унікальні виклики та вимоги, і визначення правильного рішення залежить від ваших конкретних потреб. Якщо вам потрібна порада, команда Nomios тут, щоб допомогти. Наші експерти можуть підтримати вас у пошуку рішень із кібербезпеки, які найкраще підходять для вашої організації.

Palo Alto Networks NGFW

Palo Alto Networks продовжує лідирувати на ринку NGFW, визнаний лідером у четвертому кварталі 2024 року Forrester Enterprise Firewalls Wave™.

Хоча Magic Quadrant™ Gartner 2024 для мережевих брандмауерів ще не опубліковано на момент написання статті, десятиріччя перебування Пало-Альто на посаді лідера підкреслює його незмінні інновації та надійність.

Їхнє портфоліо NGFW включає фізичні (серія PA), віртуалізовані (серія VM) і контейнерні (серія CN) брандмауери, усі вони засновані на однопрохідній архітектурі, яка перевіряє весь трафік – додатки, загрози та вміст – прив'язуючи його до користувача, незалежно від місця чи пристрою. Завдяки інтеграції механізму хмарної ідентифікації та підтримці безпеки SaaS через CASB Palo Alto пропонує комплексний захист для гібридних і багатохмарних середовищ.

Palo Alto використовує свої сильні сторони завдяки передовому штучному інтелекту та машинному навчанню, забезпечуючи захист у реальному часі від загроз нульового дня та програм-вимагачів. Їхні рішення поширюються на безпеку IoT і OT за допомогою таких функцій, як мікросегментація, аналітика поведінки та видимість пристроїв. Ці інновації зміцнюють роль Пало-Альто як наріжного каменю для організацій, які застосовують Zero Trust і гібридні хмарні стратегії.

Fortinet FortiGate

Брандмауери FortiGate нового покоління (NGFW) Fortinet відомі своєю високою продуктивністю та комплексними функціями безпеки. Створені на основі уніфікованої операційної системи, FortiGate NGFW забезпечують послідовний захист у фізичних, віртуальних і хмарних середовищах, ефективно захищаючи різні межі мережі в будь-якому масштабі. Вони об'єднують розширені можливості, такі як запобігання вторгненням, контроль програм і захист від зловмисного програмного забезпечення, забезпечуючи наскрізну безпеку через єдину платформу.

У 2024 році Fortinet представила серію FortiGate 200G, розроблену для підвищення безпеки мережі кампусу. Завдяки процесору безпеки п'ятого покоління (SP5) ця серія пропонує підвищену пропускну здатність брандмауера, виявлення загроз на основі штучного інтелекту та порти 5GE для підтримки

останнього стандарту Wi-Fi 7. Ці функції дозволяють організаціям ефективно керувати та захищати зростаючі обсяги трафіку з великим вмістом даних і хмарних програм.

Крім того, Fortinet було визнано лідером у звіті The Forrester Wave™: Enterprise Firewall Solutions, Q4 2024, що підкреслює його прагнення надавати розширені мережеві можливості та можливості безпеки за допомогою своїх рішень FortiGate NGFW.

Juniper Networks

Лінія брандмауерів Juniper Networks – це серія NGFW SRX, яка доступна як апаратні пристрої (SRX), віртуальні пристрої (vSRX) і контейнери (cSRX). vSRX можна розмістити на власному гіпервізорі клієнта або запустити на AWS, Microsoft Azure, Google Cloud Platform і Oracle Cloud Infrastructure. Серія NGFW Juniper Networks SRX поєднує в собі високопродуктивну безпеку з інтегрованими службами для безпеки програм, захисту від вторгнень і розширеного виявлення загроз для організацій будь-якого розміру.

NGFW від Juniper відомі своєю ефективністю безпеки, досягаючи 99,7% блокування експлойтів з нульовим помилковим спрацьовуванням у звіті CyberRatings.org Cloud Network Firewall 2024.

У 2024 році Juniper Networks розширила свій портфель NGFW серії SRX моделями SRX4300 і SRX4700, призначеними для задоволення потреб підприємств середнього розміру та великих мереж відповідно. SRX4300 ідеально підходить для кампусів і регіональних штаб-квартир, пропонуючи високоефективну безпеку та вдосконалене запобігання загрозам. SRX4700 призначений для великих середовищ, таких як центри обробки даних, забезпечуючи потужний захист для базових і периферійних мереж. Обидві моделі оснащені AI-Predictive Threat Prevention, можливостями нульової довіри та бездоганною інтеграцією з сучасними мережевими архітектурами, що підтверджує прагнення Juniper до масштабованих та адаптованих рішень безпеки.

Захищений брандмауер Cisco

Захищений брандмауер Cisco – це рішення брандмауера наступного покоління, яке об'єднує розширений захист від загроз, видимість додатків і єдине керування політикою. Він забезпечує однакову безпеку у фізичних, віртуальних і хмарних середовищах, забезпечуючи послідовний захист і спрощену роботу.

Їх брандмауер визнано лідером у Forrester Wave™: Enterprise Firewall Solutions Q4 2024, надає розширені можливості безпеки, керовані ШІ, адаптовані до потреб сучасного підприємства. Завдяки штучному інтелекту та машинному навчанню, що забезпечують автоматизоване керування політикою та видимість зашифрованого трафіку, Cisco забезпечує проактивний захист від нових загроз. Його багаторівневий підхід до перевірки, включаючи механізм SnortML, забезпечує ефективне запобігання вторгненням, забезпечуючи комплексну безпеку для різноманітних мережевих архітектур. Ці функції роблять Cisco надійним партнером для організацій, яким потрібні масштабовані та адаптивні брандмауери.

Чи знаєте ви, що Nomios пропонує послуги керованого брандмауера ? Наші служби керованого брандмауера піклуються про повсякденне керування операціями вашого брандмауера, забезпечуючи покращену доступність і гарантуючи безперервність.

Управління вашою інфраструктурою безпеки полягає не лише в тому, щоб підтримувати брандмауери в актуальному стані за допомогою латок і виправлень. Ми гарантуємо актуальність заходів безпеки, активно відстежуючи доступ до Інтернету з мережі компанії, реагуючи на критичні попередження, збираючи управлінські звіти та надаючи вказівки щодо оптимізації конфігурацій брандмауера.

Forcepoint

Портфоліо мережевої безпеки Forcepoint включає сім різних серій брандмауерів з різними цілями. Усі серії включають централізоване керування та розширені засоби безпеки, такі як VPN, IPS, зашифрована перевірка, SD-WAN і

проксі-сервери критично важливих програм.

За словами Forcepoint, їх NGFW розроблено для того, щоб зменшити складність і час, необхідний для безперебійної та безпечної роботи вашої мережі. І зберегти його там. Брандмауер наступного покоління Forcepoint побудовано на уніфікованому програмному ядрі, яке забезпечує узгоджені можливості, прискорення та централізоване керування для всіх типів розгортань. Їх Центр управління безпекою (SMC) може налаштовувати, контролювати та оновлювати до 2000 пристроїв Forcepoint NGFW – фізичних, віртуальних і хмарних – усе з одного скла.

Рішення Nomios і NGFW

Постачальники NGFW, з якими ми працюємо, надають безліч можливостей безпеки для периферії вашої мережі, центрів обробки даних і хмарних програм через фізичні, віртуальні та контейнерні брандмауери. Однак важливо зазначити, що, за даними Gartner, 99% зламів брандмауера є результатом неправильної конфігурації, а не недоліків у самих брандмауерах. Ось тут і з’являється Номіос.

Наші експерти з мережевої безпеки можуть оптимізувати вашу поточну конфігурацію брандмауера або скерувати вас у виборі ідеального NGFW для вашої організації. Кожна організація унікальна, і ми тут, щоб гарантувати, що ви отримаєте рішення, адаптоване до ваших конкретних потреб, і водночас дотримуєтесь галузевих стандартів і найкращих практик. Це дасть вам впевненість, необхідну для підтвердження відповідності, і впевненість, щоб зосередитися на своєму бізнесі, і скоротить час виходу на ринок нових послуг. Зв’яжіться з нами, щоб почати будувати міцнішу основу безпеки вже сьогодні.

LanAgent

LanAgent Standard – програма для схованого спостереження за комп'ютерами в локальній мережі. Призначена для контролю дій користувачів і захисти конфіденційної інформації від витоків.

Ця версія ідеально підійде для невеликих компаній, коли завдання полягає в моніторингу дій співробітників за комп'ютерами. З LanAgent Standard, керівник завжди буде в курсі того, що відбувалося під час його відсутності.

LanAgent Enterprise – легко масштабований інструмент для спостереження за комп'ютерами службовців у локальній мережі. Є ефективним засобом для забезпечення інформаційної безпеки. LanAgent Enterprise також має всі необхідні функції для контролю персоналу організації.

Основна відмінність Enterprise від версії Standard полягає в тому, що Enterprise є повноцінним інструментом для захисту конфіденційної інформації й краще підійде для організацій, у яких потрібно не просто моніторинг, а саме планомірне виявлення й запобігання витоків інформації. LanAgent допоможе в рішенні однієї з основних завдань інформаційної безпеки – боротьбі з інсайдерством, тобто захисту організації від внутрішніх погроз.

LanAgent Terminal – легко масштабований інструмент для спостереження за діями користувачів термінальних клієнтів.

LanAgent Terminal має значну частину можливостей версії LanAgent Enterprise, таких як: підключення декількох робочих місць фахівця безпеки, розподіл прав на перегляд інформації, відправлення повідомлень про порушення політик безпеки на ICQ і E-mail, передплата на оповіщення по даних подіях, убудований планувальник звітів,... І призначений для використання на технології термінал-сервер (на основі Win 2003/2008/2012 Server). Він ідеально підійде для організацій, у яких основна робота користувачів відбувається через віддалене підключення.

AbsolutusFileCrypter

AbsolutusFileCrypter – сучасні тенденції безпеки зобов'язують використовувати численні засоби шифрування інформації не тільки корпоративних, але й звичайних користувачів. Тому дуже часто встає питання про те, як просто, швидко й надійно захистити свою конфіденційну інформацію, таким чином, щоб вона була недоступна зловмисникові.

На допомогу прийде портативна програма «AbsolutusFileCrypter», за допомогою якої можна легко зашифрувати будь-який файл використовуючи пароль. І можете більше не хвилюватися про те, що ваша особиста інформація стане доступна сторонній особі. Алгоритм використовуваний у даній програмі фактично є абсолютно надійним і не підданим злому, про що свідчить спеціальне дослідження й думку незалежного експерта в області інформаційної безпеки. Плюс до всього програма має достатню швидкість роботи, надійністю й портативністю (може працювати з будь-якого носія інформації: диск, флеш і т.п.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

В роботі запропонована система часткових кількісних характеристик захищеності:

– порушення конфіденційності інформації;

– порушення цілісності ресурсу (інформації);

– порушення доступності ресурсу (інформації).

В якості комплексних кількісних характеристик захищеності запропоновано економічні показники: шкоду, яка завдається власнику АСК за рахунок реалізації загроз безпеці ресурсів АСК із – за недосконалості системи їх захисту (вираз 3.1), чи шкоду, яка запобігається при використанні системи захисту ресурсів АСК (вираз 3.2). Залежність вартісних показників захищеності від перелічених змінних в роботі пропонується розглядати як цільову функцію**,** а часові та інші характеристики використовувати як параметри (в деяких випадках – як обмеження) для оптимізації економічних (вартісних) показників ефективності**.** В якості таких параметрів управління (показників чи обмежень) в роботі запропоновано тривалість періоду (періодичність) контролю, тривалість відповідної процедури контролю та час затримки в наданні послуги, наприклад час затримки повідомлень в каналах зв’язку.

$$MQ = \sum_{i=1}^{i=n} [G_i \cdot (\mathrm{T}_{ki} - \Delta T_{ki}) \cdot (1 - p_{вi}) + p_{вi} \cdot C_i \cdot \Delta T_{ki}]. \quad (3.1)$$

$$MQ_{зi} = \sum_{i=1}^{i=n} \{G_i \cdot T_{ki} \cdot p_i - [p_i \cdot C_i - G_i (1 - \mathrm{p}_i)] \cdot [\Delta t_{ki} + \Delta t_{ni} \cdot p_{вi}]\}. \quad (3.2)$$

де:

i – номер загрози ресурсам АСК чи засобу протидії цій загрозі;

n – кількість загроз;

G_i – розмір шкоди (в умовних одиницях в одиницю часу), яка може бути завданою при вдалій реалізації кожного з типів загроз;

C_i – шкода за рахунок простою відповідних ресурсів АСК під час контролю (в умовних одиницях в одиницю часу);

$p_{вi}$ – ймовірність виявлення і подальшої протидії загрозі i – го типу;

T_{ki} – період контролю;

ΔT_{ki} – загальна тривалість і – го виду контролю;

Δt_{ki} – тривалість процесу контролю (пошуку факту порушення чи непорушення відповідної функціональної властивості захищеності автоматизованої системи);

Δt_{ni} – тривалість процесу поновлення цієї ж функціональної властивості.

Аналіз цих виразів дав можливість визначити умови доцільності застосування системи ТЗІ від загроз даного (і – го) типу у вигляді обмеження (вираз 3.3) на тривалість контролю ΔT_{ki} (при якимось чином визначеній величині p_i) чи обмеження на ймовірність $p_{вi}$ (вираз 3.4)

$$\Delta T_{ki} = \Delta t_{ki} + \Delta t_{ni} \cdot p_{вi} < T_{ki} \cdot (G_i / C_i), \quad (3.3)$$

$$p_{вi} > \Delta T_{ki} / [(T_{ki} - \Delta T_{ki}) - \Delta T_{ki} \cdot (3 C_i / G_i)]. \quad (3.4)$$

Подальшій аналіз цільової функції у вигляді (3.2) дав змогу знайти оптимальні параметри системи захисту – характеристики системи ТЗІ, в сенсі максимуму шкоди, яка запобігається завдяки застосуванню системи захисту. Визначено, що оптимальне значення параметру оперативного управління (T_{ki}) можна знайти з виразу (3.5)

$$T_{ki} = \Delta T_{ki} + 1/z + 0{,}5\sqrt{s}, \quad (3.5)$$

$$s = \Delta T^2_{ki} \cdot (1 + 4C_i/G_i) + 4(3.1/z^2 + C_i \cdot \Delta T_{ki} / (G_i \cdot z)),$$

а величина z – результуюча інтенсивність загроз. Показано також, що для зменшення величини шкоди слід застосовувати чи розробляти засоби захисту з мінімально можливою тривалістю контролю ΔT_{ki}.

3.2 Розробка структурної схеми

В даному розділі розроблено методику оцінки кількісного значення величини залишкового ризику при забезпеченні функціональних властивостей захищеності АС, яка передбачає наступні етапи:

– введення кількісних характеристик функціональних властивостей захищеності ресурсів АСК на основі визначених класу і структури таких АСК;

– оцінку можливості реалізації загроз ресурсам АСК;

– аналіз можливих наслідків від реалізації потенційних загроз, тобто оцінку можливого рівня заподіяної ними шкоди – можливого залишкового ризику.

Для вирішення першої задачі – визначення класу і структури АСК, інформаційні ресурси якої слід захищати, та введення кількісних характеристики функціональних властивостей захищеності системи з метою забезпечення інваріантності результатів досліджень щодо класу АСК, розглядається найбільш поширений клас АС – ієрархічна АСК на базі розподіленого багатомашинного багатокористувацького комплексу. Показано, що для кількісної оцінки залишкового ризику в таких АС можна використати величини ймовірностей порушення відповідних функціональних властивостей захищеності ресурсів q_i (i=1–3), які є зворотними до введених в першому розділі величин $p_{вi}$.

Для оцінки можливості реалізації загроз ресурсам АСК запропоновано символьну модель порушника та загроз у табличному вигляді. Показано, що модель порушника повинна включати опис:

– категорій осіб, які можуть бути порушниками;

– припустимого (можливого) характеру навмисних дій з боку цих порушників;

– їх можливої кваліфікації та рівня можливостей;

– можливої метою порушника – зловмисника;

Рисунок 3.1 – Структурна схема системи

– характеру дій (випадкові, зловмисні, активні, пасивні, рішучі, терплячі та ін.);

– використовуємих методів і способів;

– місця здійснення порушення;

– можливостей ненавмисних порушників (авторизованих користувачів).

Аналіз таких моделей надав можливість сформулювати події, пов'язані з порушенням функціональних властивостей захищеності і отримати вирази для розрахунків кількісних значень (величин) залишкового ризику – величин

ймовірностей порушення цих функціональних властивостей захищеності конфіденційності – q_1 (вираз 3.6), цілісності q_2 (вираз 3.7), доступності q_3 (вираз 3.8) та узагальнену оцінку захищеності інформаційних об'єктів $P_{зп}$ (вираз 3.9):

$$q_1 = 1–\{1– P_{кзі}\cdot[1– (1-p_1)\cdot(1- P_{зві})\}\cdot[1– P_{ккрз}\cdot P_{акрз}]], \quad (3.6)$$

$$q_2 = –\{1 – P_{уд}\cdot[1–(1-p_{оод})\cdot(1-p_{ос})]\}\cdot[1– P_{кц}]\cdot[1– P_{акц}\cdot P_{ккц}]. \quad (3.7)$$

$$q_3 = 1 – (1+\lambda_3\, t_{кр})\cdot\exp(–\lambda_3\, t_{кр}), \quad (3.8)$$

$$P_{зп} = 1– P_{пфх}\cdot(3.1 – P_ч). \quad (3.9)$$

$$P_1 = P_{уфд}\cdot P_{ад}\cdot[1-(1-p_{оод})\cdot(1-p_{ос})],$$

$$P_{кзі} = P_{зм}\cdot P_{зкп}\cdot P_{кн},$$

$$\lambda_з = \lambda_{сз} + \lambda_{р3},$$

$$\lambda_{р3} = (\lambda_{ші}(1- p_{уд})+\lambda_і+\lambda_{сп}P_{кпц}\cdot P_{зсп})\cdot(1-p_{в3}) = (\lambda_{ші}P_{уд}+\lambda_і+\lambda_{сп}P_{кпц}\cdot P_{зсп})\cdot q_{п3},$$

$$P_{пфх} = 1– (1- q_1)\cdot(1- q_2)\cdot(1- q_3),$$

де:

$P_{кзі}$ – ймовірність подолання засобів криптографічного захисту інформації;

P_1 – ймовірність несанкціонованого отримання користувачем інформації при безпосередньому впливі;

$P_{зві}$ – ймовірність подолання засобів захисту від витоків інформації технічними каналами;

$P_{ккрз}$ – ймовірність подолання неавторизованим користувачем засобів канального криптографічного захисту інформації в телекомунікаційних мережах (ТКМ);

$P_{акрз}$ – ймовірність подолання неавторизованим користувачем засобів абонентського криптографічного захисту інформації в ТКМ;

$P_{уд}$ – ймовірність подолання засобів управління доступом;

$P_{оод}$ – ймовірність подолання засобів організаційного обмеження доступу;

$P_{ос}$ – ймовірність подолання засобів охоронної сигналізації;

$P_{кц}$ – ймовірність подолання засобів контролю цілісності інформації відповідного вузла;

$Р_{акц}$ – ймовірність подолання засобів абонентського контролю цілісності інформації в ТКМ;

$Р_{ккц}$ – ймовірність подолання засобів канального контролю цілісності інформації в телекомунікаційної мережі;

$\lambda_з$ – інтенсивність запитів на використання ресурсів АС;

$t_{кр}$ – середній час використання захищеного ресурсу;

$Р_{уфд}$ – ймовірність подоланні порушником засобів управління фізичним доступом;

$Р_{ад}$ – ймовірність подоланні порушником засобів адміністрування доступом;

$Р_{зм}$ – ймовірність того, що порушник знає мову, якою інформація представляється;

$Р_{зкп}$ – ймовірність того, що порушник знає і може застосувати програмні засоби або апаратуру для криптографічного перетворення (для дешифрування закритої інформації);

$Р_{кн}$ – ймовірність того, що порушник має необхідні ключі (ключові набори) для такого перетворення;

$p_{уд}$ – ймовірність неподолання загрозами засобів управління доступом до захищених ресурсів;

$\lambda_{сз}$ – інтенсивність справжніх запитів;

$\lambda_{рз}$ – результуюча інтенсивність загроз захищеному ресурсу;

$\lambda_{ші}$– інтенсивність штучних впливів через засоби управління доступом;

λ_i – інтенсивність природних впливів;

$\lambda_{сп}$ – інтенсивність спеціальних впливів по технічним каналам;

$Р_{зсп}$ – ймовірність подолання засобів захисту від спеціального впливу на інформацію по технічним каналам;

$Р_{кпц}$ – ймовірність подолання засобів контролю та поновлення цілісності інформації вузлів центрального, регіонального чи місцевого рівнів АС;

$P_{пфх}$ – ймовірність порушення функціональних властивостей захищеної системи;

$P_ч$ – ймовірність того, що математичне сподівання часу функціонування власне засобів захисту $t_{сз}$ не перевищить час обробки впливів завад, безперервних запитів, спроб підбору паролів та т.п. $t_{бз}$, тобто $P_ч = P(3.t_{сз} \leq t_{бз})$.

Структурна схема роботи системи зображена на рисунку 3.1.

Розроблена програма розбита на модульну структуру дозволяючи швидко змінювати необхідні параметри програми без корінної зміни структури.

3.3 Розробка функціональної схеми

Розглянемо питання оцінки ефективності організації обміну в телекомунікаційних мережах на основі аналізу задач забезпечення цілісності та доступності інформаційних об’єктів в таких мережах, оцінки впливу способів організації обміну та можливих методів підвищення цілісності та доступності інформації в ТКМ на їх ефективність. Зробимо висновок про суттєву вразливість стану захищеності АСК як раз через телекомунікаційні мережі, канали обміну інформацією та через їх елементи, а, відтак, про актуальність та важливість досліджень та розробок щодо методів, способів, засобів та методик оцінки та забезпечення властивостей захищеності інформації в таких мережах, каналах обміну інформацією та їх елементах.

Для оцінки захищеності інформаційних ресурсів в телекомунікаційних мережах в даному розділі розроблено методики: оцінки цілісності інформаційних об’єктів в ТКМ при реалізації того чи іншого способу організації обміну. Оцінка цілісності здійснюється з використанням такої характеристики як правильність передачі даних (ймовірність правильної доставки повідомлення чи отримання на приймальному боці невикривленої інформації); а оцінка доступності ТКМ – через абсолютну швидкість обміну інформації в ТКМ та час затримки в доставлянні

повідомлень при реалізації того чи іншого способу організації обміну; запропонована оцінка можливої шкоди із-за неефективної організації обміну.

На основі аналізу топології ТКМ та потоків загроз тій чи іншій функціональній властивості захищеної системи чи її елементам задача забезпечення властивостей захищеності ресурсів ТКМ розподілена на:

– задачу забезпечення властивостей захищеності ресурсів в каналах (канальні цілісність та доступність);

– задачу забезпечення властивостей захищеності ресурсів ТКМ вцілому (абонентські цілісність та доступність).

Розглянута модель взаємодії засобів в процесі ТЗІ в телекомунікаційних системах дозволила отримати вирази для розрахунку результуючих інтенсивностей загроз доступності $\lambda_{рд}$ (вираз 3.10), цілісності $\lambda_{рц}$ (вираз 3.11) ресурсів ТКМ та результуючу інтенсивність $\lambda_н$ загроз, які не усунуті системою ТЗІ (вираз 3.12)

$$\lambda_{рд} = \lambda_{ша} P_{уд} + \lambda_{шк} + \lambda, \quad (3.10)$$

$$\lambda_{рц} = \lambda_{ша} P_{уд} + \lambda_{шк} + \lambda P_{ккц} \quad (3.11)$$

$$\lambda_н = \lambda_{рц} P_{акц} = (\lambda_{ша} P_{уд} + \lambda_{шк} + \lambda P_{ккц}) \cdot P_{акц}, \quad (3.12)$$

де $\lambda_{ша}$, $\lambda_{шк}$ та λ – інтенсивності штучних впливів на абонентському та канальних рівнях ТКМ, а λ – інтенсивність природних загроз в ТКМ.

Найбільший вплив на ресурси ТКМ, як витікає з виразу (3.12), слід очікувати від штучних впливів на канальному рівні, оскільки вони не зменшуються (не проріджуються) ніякими засобами, окрім засобів забезпечення цілісності ресурсів ТКМ на абонентському рівні, та особливу необхідність при цьому збільшення ймовірності виявлення та усунення впливу засобами абонентського контролю цілісності інформації ($1-p_{акц}$).

Аналіз можливостей забезпечення цілісності інформації в умовах природних впливів (проблеми завадостійкості) для каналів ТКМ (взагалі для мереж передачі даних) надав можливість зробити висновок, що найбільш прийнятними для того класу АСК, який розглядається в роботі, є способи з

використанням зворотного вирішуючого зв'язку (ЗВЗ) та способи з використанням завадостійких корегуючих кодів (ЗКК). Для їх більш ретельного порівняння здійснено порівняння цих способів за абсолютною швидкістю передачі даних (вирази 13, 14) та ймовірністю правильної доставки повідомлення (вирази 15, 16) при використанні при обміні зворотного вирішуючого зв'язку та корегуючих кодів відповідно:

$$B_{азвз} = \exp(-\lambda t_{пк})\cdot m/\{n/B+ \Delta t_{к1}+\Delta t_{п1}\cdot(1-\exp(-\lambda t_{пк}))\} \quad (3.13)$$

$$B_{азкк} = (1+\lambda t_{nk})\exp(-\lambda t_{nk})\cdot m_{зкк}/\{n/B+ \Delta t_{к2}+\Delta t_{n2}\cdot(1-\exp(-\lambda t_{nk}))\}, \quad (3.14)$$

де відповідно:

λ – інтенсивність впливів;

$t_{пк}$ – часова тривалість ($3.t_{nk}$ = n/B) повідомлення в канальній ланці захисту;

n, m, $m_{зкк}$ – загальна кількість та кількість інформаційних символів в повідомленнях;

B – технічна швидкість передачі даних;

($\Delta t_{к1}$) $\Delta t_{к2}$ – час, потрібний для контролю цілісності повідомлень;

(Δt_{n1}) Δt_{n2}·– час, потрібний для виправлення викривлень в повідомленні (в разі їх виявлення).

На підставі аналізу цих виразів визначено межі більшої ефективності способів організації обміну із ЗВЗ по швидкості передачі порівняно із способами організації обміну із ЗКК (при доброму стані каналу – малому значенні $\lambda<\lambda_{гр1}$, де $\lambda_{гр1}$ – така величина інтенсивності впливів, при якій абсолютні швидкості передачі інформації для різних методів організації обміну є однаковими). Окрім того, при наближенні надлишковості $m_{зкк}$ до m величина $\lambda_{гр1}$ наближається до нуля, тобто діапазон значень інтенсивностей впливів, при яких способи організації обміну із ЗКК є більш прийнятними. Показано, що ймовірність правильної доставки повідомлення при використанні при обміні зворотного вирішуючого зв'язку та корегуючих кодів відповідно можна оцінити із виразів

$$P_{звз} = 1-2^{-(3.n-m)} = \text{const}, \quad (3.15)$$

$$P_{зкк} = (1-2^{-(3.n-m1)})\cdot(1+ \lambda n /B)\cdot\exp\{-\lambda n /B\}, \quad (3.16)$$

де $m1 = m_{зкк}$.

При цьому способи організації обміну із ЗКК дещо перевершують способи організації обміну із ЗВЗ лише тоді, коли інтенсивність впливів λ є незначною ($\lambda<\lambda_{гр2}$), в решті випадків канали з ЗВЗ є, з погляду цього показника, більш ефективними.

Таким чином, виходячи з аналізу цих двох окремих показників, неможливо віддати незаперечну перевагу тому чи іншому способу організації обміну даними, оскільки спосіб більш ефективний за одним показником є менш ефективним за іншим. Тому в роботі введено комплексний показник ефективності у вигляді добутку швидкості на ймовірність правильної доставки повідомлення – ефективну швидкість обміну E. Вирази для розрахунків такої ефективної швидкості для способів організації обміну із ЗВЗ мають вигляд (вирази 17, 18):

$$E_{звз} \approx B_a = \exp(-\lambda t_{nk})\cdot m/\{n/B + \Delta t_{к1} + \Delta t_{п1}\cdot(1-\exp(-\lambda t_{nk}))\} \quad (3.17)$$

$$E_{зкк} \approx (1+\lambda n/B)\cdot exp\{-\lambda n/B\}\cdot(1+\lambda t_{nk})\exp(-\lambda t_{nk}))\cdot m_{зкк}/(n/B+\Delta t_{к2}+\Delta t_{п2}\cdot(1-\exp(-\lambda t_{nk}))). \quad (3.18)$$

З аналізу цих залежностей витікає, що ефективна швидкість способів організації обміну із ЗКК переважає ефективність способів організації обміну із ЗВЗ, коли інтенсивність впливів λ є більшою за деяке її граничне значення $\lambda_{гр}$. Визначено граничні значення інтенсивність впливів λ для абсолютної та ефективної швидкостей.

Останні вирази дозволяють в залежності від стану каналів вибирати чи один, чи інший спосіб організації обміну, чи побудувати адаптивні комбіновані системи передачі даних, в яких залежно від інтенсивності завад використовується режим роботи або із ЗВЗ, або із ЗКК, тобто можна побудувати комбіновану СПД. Аналогічні висновки витікають і з проведеного в даному розділі аналізу залежності часу затримки доставки повідомлення від інтенсивності впливів для різних способів організації обміну. Показано напрямки підвищення ефективності обміну інформацією в ТКМ.

Для оцінки методів підвищення цілісності та доступності інформації в ТКМ, варіантів побудови ТКМ з мінімізованим значенням можливої шкоди розглянуто загальну модель ТКМ в розумінні складу та взаємних зв'язків засобів захисту інформації в ТКМ та методику оцінки ефективності обміну в телекомунікаційних мереж в розумінні можливої шкоди.

Розробимо метод забезпечення цілісності інформаційних ресурсів АСК на основі завадостійкого корегуючого ЛУ-коду та методики його використання, які забезпечують застосування погоджених між собою швидкодіючих процедур як виявлення порушення цілісності інформації, так і її поновлення.

З цією метою здійснимо аналіз можливостей цього коду по виявленню, корекції порушень цілісності в умовах зберігання чи передавання інформаційних об'єктів, потрібної при цьому надлишковості та імітостійкості, яка забезпечується при застосуванні розробленого методу. Показано, що запропонований код в поєднанні з умовним перемежуванням дозволяє використовувати надзвичайно прості алгоритми кодування та декодування, які можливо застосовувати для визначення наявності викривлень та їх корекції суто розрахунковими процедурами, тобто без використання для поновлення засобів резервного копіювання, причому з забезпеченням унеможливлення наявності таких викривлень, які порушник навмисно приховує.

При цьому імітостійкість механізмів контролю цілісності інформації визначається стійкістю обчислень контрольних ознак, яка залежить від довжини вибраних ключів захисту (кількості основ – констант ЛУ – коду), а також від статистичної пов'язаності початкового тексту (інформаційного блоку) з його перетвореним відображенням.

Показано, що при контролі та поновленні цілісності запропонований механізм забезпечує кількість варіантів ключів, яка суттєво перевищує кількість варіантів ключів відомих механізмів, та має, відповідно, значно вищу імітостійкість.

Ця кількість варіантів ключів задовольняє вимогам навіть гарантованого криптозахисту.

При цьому методика застосування ЛУ – коду для контролю цілісності інформаційних об’єктів передбачає виконання наступної послідовності процедур:

– відокремлення від інформаційного об’єкту його певної частини – узагальненого кодового слова довжиною *N* інформаційних символів (*N*–число байтів в узагальненому кодовому слові (в блоці інформації)) формування контрольних ознак базових кодових слів;

– формування контрольних ознак узагальнених кодових слів; формування контрольних ознак інформаційного об’єкту, наприклад деякого (і–го) файлу чи інформаційного набору;

– організацію контролю цілісності інформаційного об’єкту**;** організацію поновлення цілісності інформаційного об’єкту.

Функціональна схема розробленої системи зображена на рисунку 3.2.

З рисунку видно, що розроблена система складається з наступних частин:

– Блок розмежування доступу користувачів до важливої інформації.

– Блок автоматичного моніторингу небажаних дій користувачів.

– Блок керування запуском додатків.

– Блок контролю інформації в хмарних сховищах.

– Блок захисту інформації від крадіжки при втраті ноутбуків і флешек.

– Блок контролю інформації, переданої через корпоративну поштову систему, Інтернет-ресурси, засоби загального доступу до файлів (SMTP, HTTP, HTTPS, FTP).

– Блок контролю систем обміну повідомленнями (ICQ, Skype, Mail.ru Агент, GTalk і інші).

– Блок контролю голосового трафіку (Skype).

– Блок контролю використання пристроїв і портів на робочих станціях.

– Блок контролю мережних з'єднань на робочих станціях.

Рисунок 3.2 – Функціональна схема системи

– Блок тіньового копіювання що роздруковуються й копіюємих на знімні носії документів.

– Блок запобігання витоку конфіденційних даних шляхом блокування процесу передачі у випадку порушення політики безпеки.

– Блок запобігання витоку персональних даних і клієнтських баз.

– Блок захисту від витоку конфіденційних даних.

– Блок захисту інтелектуальної власності.

– Блок виявлення зловмисників, осіб, що займаються шпигунством, змов, саботажу, нелояльних користувачів.

– Блок розслідування інцидентів інформаційної безпеки.

– Блок детектування «на льоту» конфіденційних даних, у тому числі документи «нульового дня».

– Блок відстеження маршрутів руху інформації й випадки нецільового використання корпоративних ресурсів.

– Блок виявлення інсайдерів, осіб, що займаються шпигунством, а також недбалості персоналу при роботі з конфіденційною інформацією.

– Блок зберігання всіх даних і проведення ретроспективного аналізу інцидентів.

– Блок пошуку шкідливого коду.

– Блок безперервного моніторингу IT-інфраструктури.

3.4 Розробка діаграми процесів

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

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним

спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

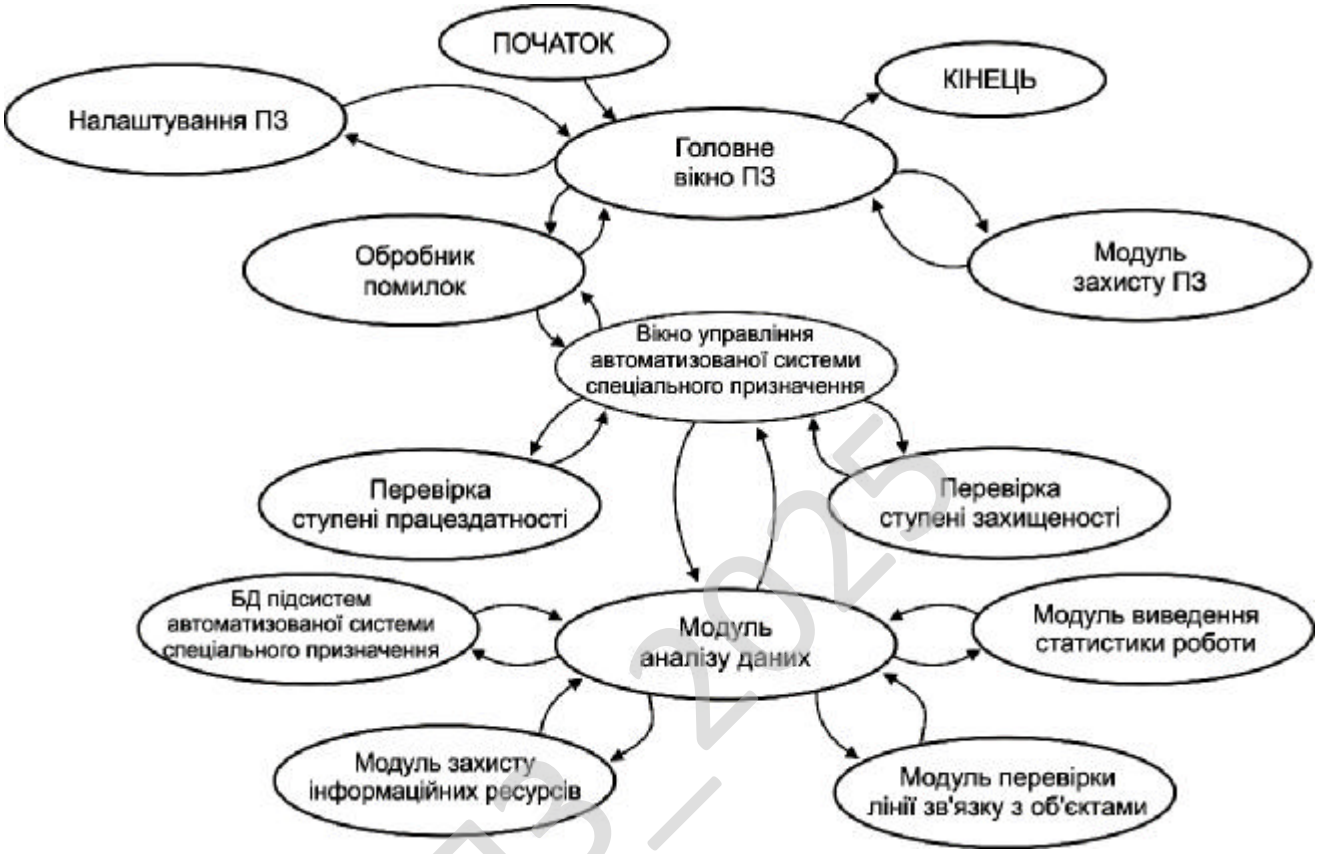

Рисунок 3.2 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

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

Опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю методу захищеності інформаційних ресурсів автоматизованих систем спеціального призначення.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і

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

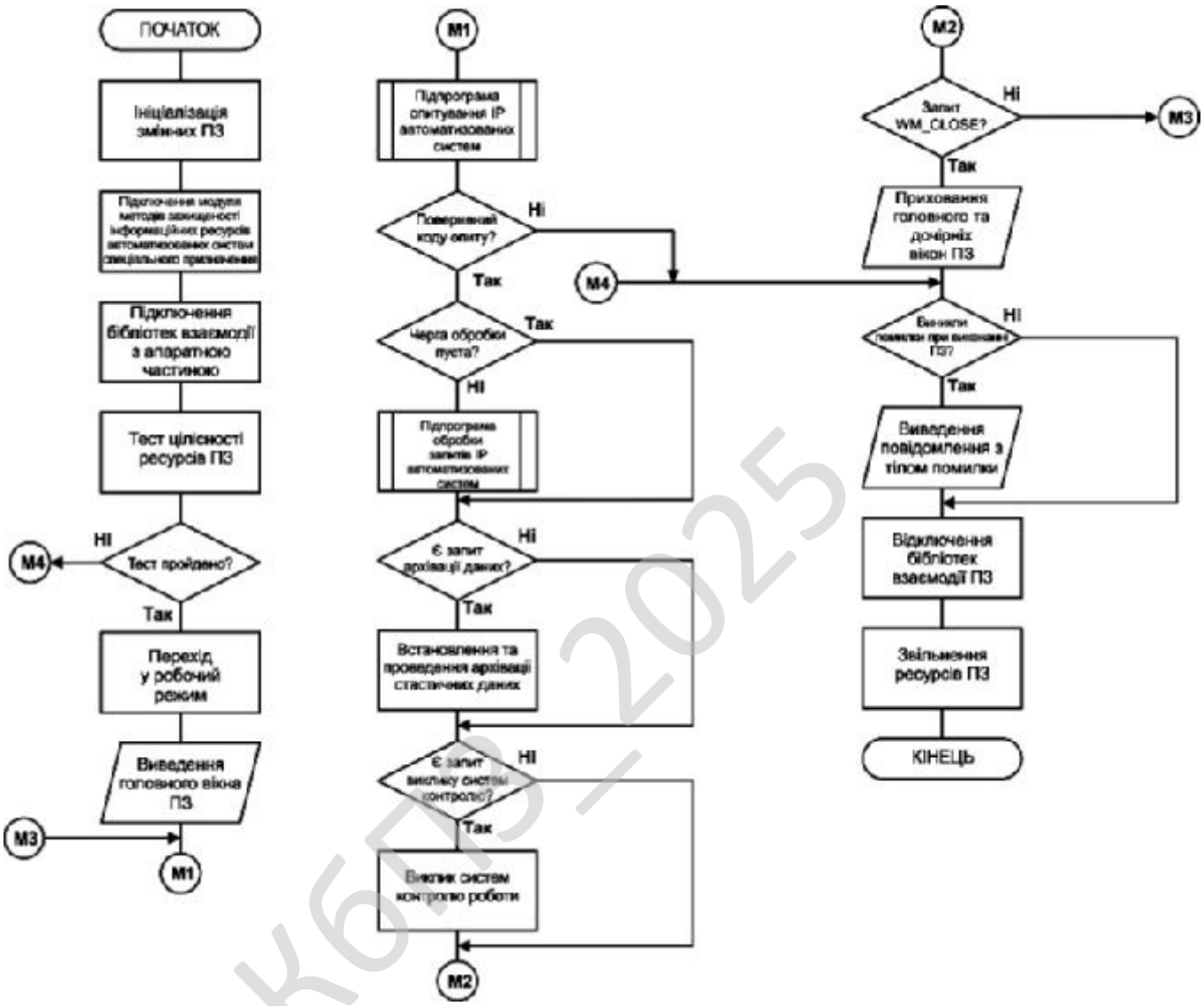

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

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

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

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

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

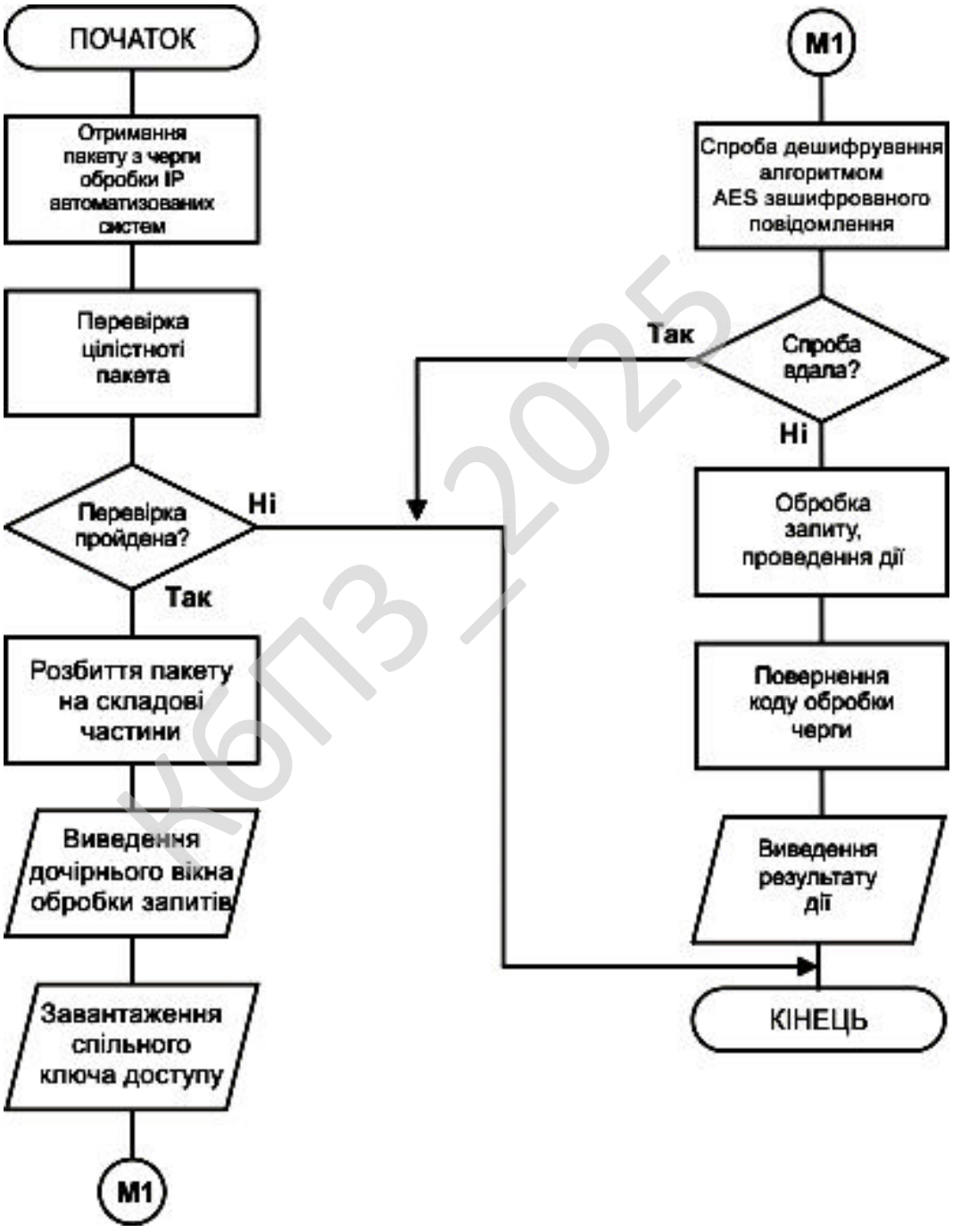

Рисунок 4.2 – Блок-схема роботи підпрограми

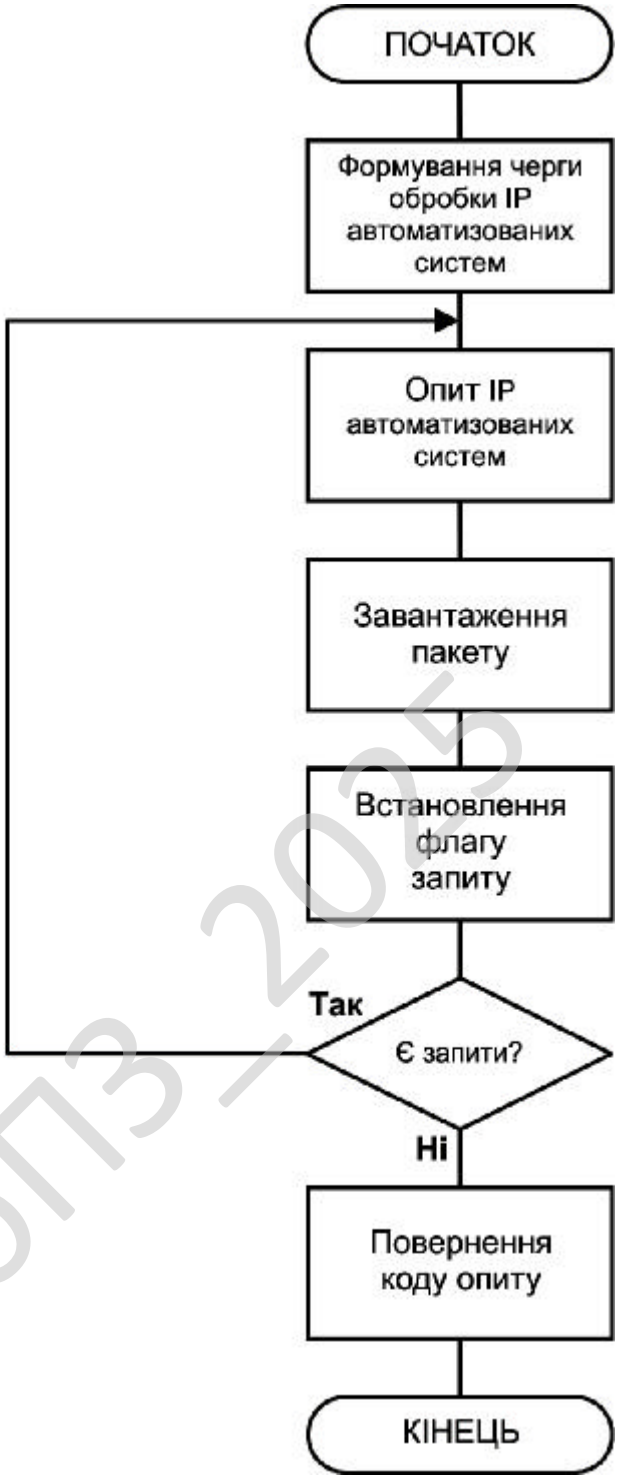

Рисунок 4.3 – Блок-схема роботи підпрограми

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість

усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Для реалізації методів захищеності інформаційних ресурсів автоматизованих систем спеціального призначення і реалізації алгоритму від несанкціонованого використання в розробленій програмі використовувалися спеціальні процедури.

Архітектура системи

Система кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення реалізується на основі модульного підходу. Вона включає механізми автентифікації, авторизації, шифрування, моніторингу та виявлення загроз. Всі основні компоненти взаємодіють через захищені канали зв'язку.

Система містить такі основні модулі:

1. Модуль автентифікації забезпечує перевірку особи користувача за допомогою паролів, двофакторної аутентифікації або біометричних даних.
2. Модуль авторизації надає доступ до інформаційних ресурсів відповідно до ролей та рівня доступу користувача.
3. Модуль шифрування використовує алгоритми AES-256 та RSA для захисту даних під час передавання та збереження.
4. Модуль моніторингу виконує збір та аналіз подій безпеки, виявляє аномалії у мережевому трафіку та діях користувачів.
5. Модуль реагування на загрози автоматично або вручну блокує шкідливу активність та формує звіти про інциденти.

Опис функціоналу вихідного коду

Програмний код реалізується мовою Python. Для кожного модуля розробляється окремий клас або набір функцій, що взаємодіють через API.

Модуль автентифікації

Функція authenticate_user перевіряє введені користувачем дані з базою:

```
import hashlib
def authenticate_user(username, password, user_db):
```

```
hash_password = hashlib.sha256(password.encode()).hexdigest()
return user_db.get(username) == hash_password
```

Модуль авторизації

Функція check_access перевіряє, чи має користувач доступ до ресурсу:

```
def check_access(username, resource, access_db):
return resource in access_db.get(username, [])
```

Модуль шифрування

Функції encrypt_data та decrypt_data використовують AES-256:

```
from Crypto.Cipher import AES
import base64

def encrypt_data(data, key):
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data.encode())
return base64.b64encode(cipher.nonce + tag + ciphertext).decode()

def decrypt_data(encrypted_data, key):
raw = base64.b64decode(encrypted_data)
nonce, tag, ciphertext = raw[:16], raw[16:32], raw[32:]
cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
return cipher.decrypt_and_verify(ciphertext, tag).decode()
```

Модуль моніторингу

Функція monitor_activity аналізує активність у системі:

```
import logging
logging.basicConfig(filename="security.log", level=logging.INFO)

def monitor_activity(user, action):
logging.info(f"User {user} performed {action}")
```

Модуль реагування на загрози

Функція block_ip додає IP-адресу у чорний список:

```
def block_ip(ip_address, blacklist):
blacklist.add(ip_address)
return f"IP {ip_address} blocked"
```

Розрахунки та обґрунтування обраних проектних рішень

Для вибору шифрування використовується AES-256, оскільки він забезпечує високу стійкість до атак. Теоретична складність злому AES-256 методом повного перебору складає 2^256 операцій, що є недосяжним навіть для сучасних суперкомп'ютерів.

Для автентифікації використовується хешування паролів алгоритмом SHA-256. Час хешування одного пароля на середньостатистичному сервері становить 0.0004 секунди. Це забезпечує швидку перевірку без суттєвого навантаження на систему.

Моніторинг активності з використанням журналів безпеки дозволяє обробляти до 10000 подій за секунду при використанні сучасних систем обробки логів. Це гарантує оперативне виявлення загроз.

Використання механізмів блокування IP-адрес у режимі реального часу дозволяє зменшити кількість атак на сервери на 70% завдяки негайному реагуванню на шкідливу активність.

Система кібербезпеки автоматизованих систем спеціального призначення забезпечує комплексний підхід до захисту інформаційних ресурсів. Вона використовує ефективні криптографічні алгоритми, механізми моніторингу та автоматичне реагування на загрози. Всі компоненти працюють у взаємодії для максимального рівня захисту.

Реалізація безпеки зберігання даних

Для забезпечення захищеного зберігання даних система використовує такі методи:

1. Шифрування бази даних. Використовується алгоритм AES-256 для шифрування конфіденційної інформації перед записом у базу даних. Паролі користувачів хешуються за допомогою алгоритму bcrypt.

2. Контроль доступу до бази даних. Впроваджується ролевий доступ, де кожен користувач має певні права на читання, запис та редагування даних. Використання параметризованих запитів для запобігання SQL-ін'єкціям.

3. Журналювання та аудит змін у базі. Всі зміни в базі записуються у журнал для подальшого аналізу та аудиту. Використання захищеного логування з підписом змін.

Реалізація безпечного зберігання даних

1. Шифрування даних перед збереженням у базі

```
from Crypto.Cipher import AES
import base64
import os
def generate_key():
return os.urandom(32)  # 256-бітний ключ
def encrypt_data(data, key):
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data.encode())
return base64.b64encode(cipher.nonce + tag + ciphertext).decode()
def decrypt_data(encrypted_data, key):
raw = base64.b64decode(encrypted_data)
nonce, tag, ciphertext = raw[:16], raw[16:32], raw[32:]
cipher = AES.new(key, AES.MODE_EAX, nonce=nonce)
return cipher.decrypt_and_verify(ciphertext, tag).decode()
```

2. Захищене хешування паролів

```
import bcrypt
def hash_password(password):
salt = bcrypt.gensalt()
return bcrypt.hashpw(password.encode(), salt).decode()
def verify_password(password, hashed_password):
return bcrypt.checkpw(password.encode(), hashed_password.encode())
```

3. Захист від SQL-ін'єкцій

```
import sqlite3
def get_user_data(username):
conn = sqlite3.connect("secure_db.db")
cursor = conn.cursor()
cursor.execute("SELECT * FROM users WHERE username=?", (username,))
user_data = cursor.fetchone()
conn.close()
return user_data
```

4. Журналювання змін у базі

```
import logging
logging.basicConfig(filename="database_audit.log", level=logging.INFO)
def log_database_change(user, action, details):
logging.info(f"User: {user}, Action: {action}, Details: {details}")
```

Запропонована реалізація дозволяє забезпечити безпечне зберігання даних, запобігти несанкціонованому доступу та атакувальним діям, таким як SQL-ін'єкції або злам паролів. Використання шифрування та аудит змін у базі покращує загальну захищеність системи.

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою національного стандарту захисту інформації на основі алгоритму шифрування/дешифрування ДСТУ 4145-2002 з використанням еліптичних кривих над двійковим розширеним полем Галуа. У системі шифрування/дешифрування як параметри розглядається еліптична крива E_p (a,b) і точка G на ній. Учасник В вибирає закритий ключ n і обчислює відкритий ключ $P_B = n \times G$. Щоб зашифрувати повідомлення P_m використовується відкритий ключ одержувача В P_B. Учасник А вибирає випадкове ціле позитивне число k і обчислює зашифроване повідомлення C_m, що є точкою на еліптичній кривій.

$$C_m = \{k \times G,\ P_m + k \times P_B\}. \quad (4.1)$$

Щоб дешифрувати повідомлення, учасник В множить першу координату точки на свій закритий ключ і віднімає результат від другої координати:

$$P_m + k \times P_B - n_B \times (k \times G) = P_m + k \times (n_B \times G) - n_B \times (k \times G) = P_m. \quad (4.2)$$

Учасник А зашифрував повідомлення P_m додаванням до нього $k \times P_B$. Ніхто не знає значення k, тому, хоча P_B і є відкритим ключем, ніхто не знає $k \times P_B$. Супротивнику для відновлення повідомлення доведеться обчислити k. Зробити це буде нелегко. Одержувач також не знає k, але йому як підказку посилається $k \times G$. Помноживши $k \times G$ на свій закритий ключ, одержувач одержить значення, що було додано відправником до незашифрованого повідомлення. Тим самим

одержувач, не знаючи *k*, але маючи свій закритий ключ, може відновити незашифроване повідомлення.

Нехай задано просте число $p > 4$. Тоді еліптичною кривою *E*, визначеною над розширеним двійковим полем полем F_{2^m}, називається безліч пар чисел $(x, y), x, y \in F$, що задовольняють тотожності:

$$y^2 \equiv x^3 + a \cdot x + b (\mathrm{mod}\, 2^m), \tag{4.3}$$

де $4 \cdot a^3 + 27 \cdot b^2$ не рівно з нулю по модулю 2^m.

Інваріантом еліптичної кривої називається величина *J*(*E*), що задовольняє тотожності:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, 2^m). \tag{4.4}$$

Коефіцієнти *a*, *b* еліптичної кривої *E*, по відомому інваріанту *J*(*E*), визначаються таким чином:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, 2^m) \\ b \equiv 2k (\mathrm{mod}\, 2^m) \end{cases} \text{де } k \equiv \frac{J(E)}{1728 - J(E)} (\mathrm{mod}\, 2^m), J(E) \neq 0 \text{ або } 1728. \tag{4.5}$$

Пари (*x*,*y*), що задовольняють тотожності (4.1), називаються точками еліптичної кривої *E*, *x* та *y* – відповідно *x*– та *y*-координатами точки.

Точки еліптичної кривої позначатимемо *Q*(*x*,*y*) або просто *Q*. Дві точки еліптичної кривої рівні, якщо рівні їх відповідні *x*– і *y*-координати.

На безлічі всіх точок еліптичною кривою E введемо операцію додавання, яку позначатимемо знаком "+". Для двох довільних точок $Q_1(x_1, y_1)$ та $Q_2(x_2, y_2)$ еліптичної кривої E, розглянемо декілька варіантів.

Нехай координати точок Q_1 та Q_2 задовольняють умові $x_1 \neq x_2$. В цьому випадку їх сумою називатимемо точку $Q_3(x_3, y_3)$ координати якої визначаються порівняннями:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, 2^m). \tag{4.6}$$

Якщо виконана рівність $x_1 = x_2$ та $y_1 = y_2 \neq 0$, то визначимо координати

точки Q_3 таким чином:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 (\mathrm{mod}\, 2^m), \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 (\mathrm{mod}\, 2^m), \end{cases} \text{де } \lambda \equiv \frac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, 2^m). \quad (4.7)$$

У разі, коли виконана умова $x_1 = x_2$ та $y_1 = -y_2 (\mathrm{mod}\, p)$, суму точок Q_1 та Q_2 називатимемо нульовою точкою О, не визначаючи її х– і у-координати. В цьому випадку, точка Q_2 називається запереченням точки Q_1. Для нульової точки О виконана рівність:

$$Q + 0 = 0 + Q = Q, \quad (4.8)$$

де Q – довільна точка еліптичної кривої Е.

Щодо введеної операції складання безліч всіх точок еліптичною кривою Е, разом з нульовою точкою, утворюють кінцеву абельову (комутативну) групу порядку т, для якого виконана нерівність:

$$p + 1 - 2\sqrt{p} \le m \le p + 1 + 2\sqrt{p} \quad (4.9)$$

Точка Q називається точкою кратності k, або просто – кратною точкою еліптичної кривої E, якщо для деякої точки P виконана рівність:

$$Q = P + ... + P = kP \quad (4.10)$$

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення методів захищеності інформаційних ресурсів автоматизованих систем спеціального призначення складається з наступних функціональних блоків:

– Навігаційне меню: Системні функції; Архівація; Зв’язок зі службами; Допомога.

– Вікно стану доступу.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

– Блок функціональних кнопок ПЗ, який розподіляється на: поточний стан; Робота з ключами; Робота з модулями.

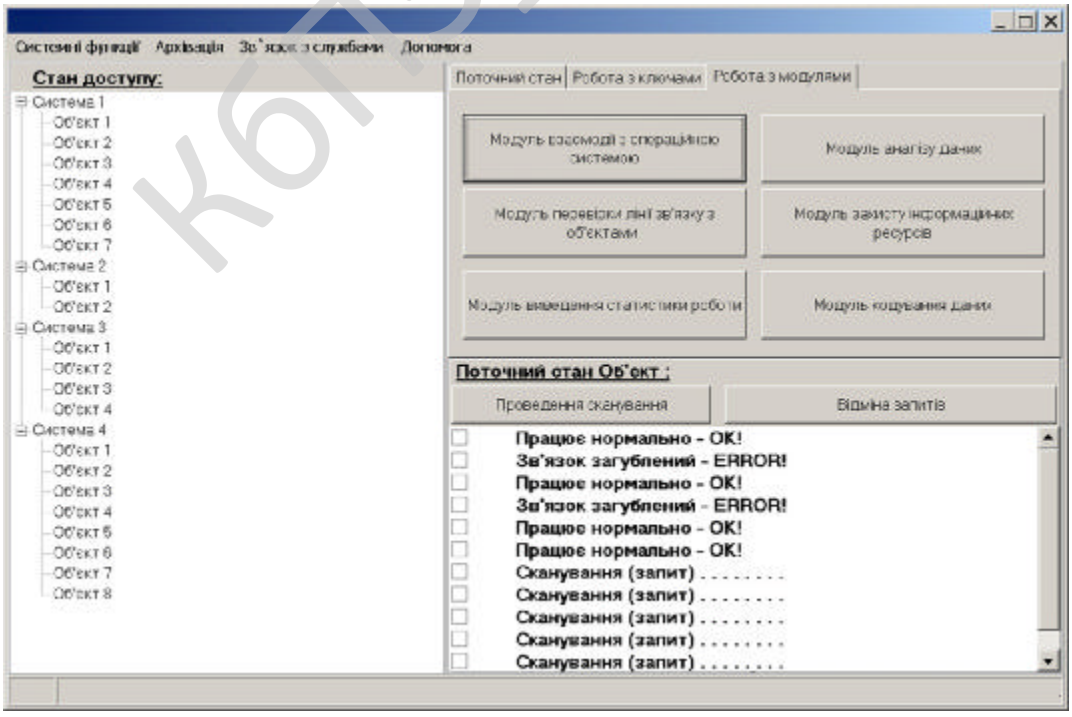

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

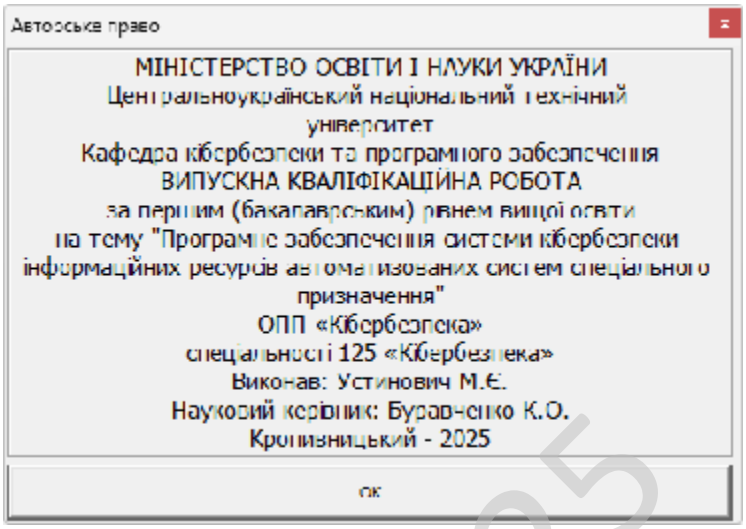

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інформаційних ресурсів автоматизованих систем спеціального призначення.

– Досліджена система інформаційних ресурсів автоматизованих систем спеціального призначення.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інформаційних ресурсів автоматизованих систем спеціального призначення.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 4145-2002.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

2. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

3. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

4. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

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

6. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

7. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

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

9. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

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

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

12. 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. 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,

14. 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.

15. 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.

16. 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

17. 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.

18. 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.

19. 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.*

20. 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.

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

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

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

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

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

44. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

45. 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.

46. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

47. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об'єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

48. 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.

49. 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.

50. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки інформаційних ресурсів автоматизованих систем спеціального призначення;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 57 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 4.06.2025 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 інформаційних ресурсів автоматизованих систем спеціального призначення

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
import os
import sys
import time
import sqlite3
import logging
import hashlib
import random
import string
import datetime
import threading
import requests
import json

#Ініціалізація менеджера баз даних для збереження інформації про користувачів,
логи та сповіщення
class DatabaseManager:
#Конструктор класу DatabaseManager
    def __init__(self, db_path='cybersecurity.db'):
#Підключення до бази даних SQLite
        self.connection = sqlite3.connect(db_path, check_same_thread=False)
#Створення об'єкту курсора для виконання запитів
        self.cursor = self.connection.cursor()
#Виклик методу для створення таблиць, якщо вони ще не існують
        self.create_tables()

#Метод для створення необхідних таблиць у базі даних
    def create_tables(self):
#SQL-запит для створення таблиці користувачів
        query_users = "CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT UNIQUE, password_hash TEXT, created_at TEXT)"
        self.cursor.execute(query_users)
#SQL-запит для створення таблиці логів
        query_logs = "CREATE TABLE IF NOT EXISTS logs (id INTEGER PRIMARY KEY
AUTOINCREMENT, event TEXT, event_time TEXT)"
        self.cursor.execute(query_logs)
#SQL-запит для створення таблиці сповіщень
        query_alerts = "CREATE TABLE IF NOT EXISTS alerts (id INTEGER PRIMARY
KEY AUTOINCREMENT, alert TEXT, alert_time TEXT)"
        self.cursor.execute(query_alerts)
#Закріплення змін у базі даних
        self.connection.commit()

#Метод для виконання загальних SQL-запитів з параметрами
    def execute_query(self, query, params=()):
#Виконання наданого запиту з параметрами
        self.cursor.execute(query, params)
#Закріплення змін у базі даних
        self.connection.commit()
#Повернення результатів запиту
        return self.cursor.fetchall()

#Метод для додавання нового користувача до бази даних
    def add_user(self, username, password_hash):
#Отримання поточної дати та часу для запису створення користувача
        created_at = datetime.datetime.now().isoformat()
#SQL-запит для вставки нового запису користувача
        query = "INSERT INTO users (username, password_hash, created_at) VALUES
(?, ?, ?)"
        try:
            self.cursor.execute(query, (username, password_hash, created_at))
#Закріплення змін після вставки
            self.connection.commit()
#Повернення True, якщо користувача додано успішно
            return True
        except sqlite3.IntegrityError:
#Повернення False, якщо ім'я користувача вже існує
```

```
            return False

#Метод для отримання даних користувача за ім'ям
    def get_user(self, username):
#SQL-запит для вибірки запису користувача за ім'ям
        query = "SELECT * FROM users WHERE username = ?"
        self.cursor.execute(query, (username,))
#Повернення знайденого запису користувача
        return self.cursor.fetchone()

#Метод для додавання запису в таблицю логів
    def add_log(self, event):
#Отримання поточного часу для запису події
        event_time = datetime.datetime.now().isoformat()
#SQL-запит для вставки нового запису події
        query = "INSERT INTO logs (event, event_time) VALUES (?, ?)"
        self.cursor.execute(query, (event, event_time))
#Закріплення змін у базі даних
        self.connection.commit()

#Метод для додавання сповіщення до таблиці сповіщень
    def add_alert(self, alert):
#Отримання поточного часу для запису сповіщення
        alert_time = datetime.datetime.now().isoformat()
#SQL-запит для вставки нового запису сповіщення
        query = "INSERT INTO alerts (alert, alert_time) VALUES (?, ?)"
        self.cursor.execute(query, (alert, alert_time))
#Закріплення змін після вставки сповіщення
        self.connection.commit()

#Метод для отримання всіх логів з таблиці логів
    def get_logs(self):
#SQL-запит для вибірки всіх записів логів
        query = "SELECT * FROM logs"
        self.cursor.execute(query)
#Повернення списку всіх логів
        return self.cursor.fetchall()

#Метод для отримання всіх сповіщень з таблиці сповіщень
    def get_alerts(self):
#SQL-запит для вибірки всіх записів сповіщень
        query = "SELECT * FROM alerts"
        self.cursor.execute(query)
#Повернення списку всіх сповіщень
        return self.cursor.fetchall()

#Клас для аутентифікації користувачів у системі
class UserAuthentication:
#Конструктор класу UserAuthentication із прийняттям менеджера баз даних
    def __init__(self, db_manager):
#Призначення менеджера баз даних для використання в класі
        self.db_manager = db_manager

#Метод для хешування пароля за допомогою алгоритму SHA-256
    def hash_password(self, password):
#Створення нового об'єкту хешування SHA-256
        hash_object = hashlib.sha256()
#Оновлення об'єкту хешування за допомогою закодованого пароля
        hash_object.update(password.encode('utf-8'))
#Повернення шістнадцяткового представлення хешу
        return hash_object.hexdigest()

#Метод для реєстрації нового користувача
    def register_user(self, username, password):
#Хешування пароля для безпечного зберігання
        password_hash = self.hash_password(password)
#Спроба додати нового користувача до бази даних
        result = self.db_manager.add_user(username, password_hash)
#Повернення результату реєстрації
```

```
        return result

#Метод для входу існуючого користувача шляхом перевірки облікових даних
    def login_user(self, username, password):
#Отримання запису користувача з бази даних
        user_record = self.db_manager.get_user(username)
#Якщо запис користувача не знайдено, повернути False
        if not user_record:
            return False
#Отримання збереженого хешу пароля з запису
        stored_hash = user_record[2]
#Хешування наданого пароля для порівняння
        provided_hash = self.hash_password(password)
#Порівняння збереженого хешу з наданим
        if stored_hash == provided_hash:
#Запис події успішного входу в систему
            self.db_manager.add_log(f"User {username} logged in successfully.")
#Повернення True, якщо облікові дані збігаються
            return True
        else:
#Запис невдалої спроби входу в систему
            self.db_manager.add_log(f"Failed login attempt for user
{username}.")
#Повернення False, якщо облікові дані не збігаються
            return False

#Клас для симуляції роботи брандмауера (Firewall) системи
class Firewall:
#Конструктор класу Firewall з ініціалізацією правил брандмауера
    def __init__(self, db_manager):
#Призначення менеджера баз даних для логування подій брандмауера
        self.db_manager = db_manager
#Ініціалізація порожнього списку правил брандмауера
        self.rules = []
#Завантаження стандартних правил брандмауера
        self.load_default_rules()

#Метод для завантаження стандартних правил брандмауера
    def load_default_rules(self):
#Додавання стандартного правила для блокування підозрілих IP-адрес
        self.rules.append("BLOCK 192.168.1.100")
#Додавання правила для дозволу трафіку з локальної мережі
        self.rules.append("ALLOW 192.168.1.0/24")
#Запис події завантаження стандартних правил
        self.db_manager.add_log("Default firewall rules loaded.")

#Метод для перевірки мережевого пакету за правилами брандмауера
    def check_packet(self, packet):
#Отримання IP-адреси відправника з пакету
        source_ip = packet.get('source_ip', '')
#Перевірка кожного правила в списку правил
        for rule in self.rules:
#Якщо правило починається з BLOCK і містить IP-адресу
            if rule.startswith("BLOCK") and source_ip in rule:
#Запис події блокування пакету
                self.db_manager.add_log(f"Packet from {source_ip} blocked by
firewall.")
#Повернення False, якщо пакет заблоковано
                return False
#Запис події дозволу пакету
        self.db_manager.add_log(f"Packet from {source_ip} allowed by firewall.")
#Повернення True, якщо пакет дозволено
        return True

#Метод для додавання нового правила до брандмауера
    def update_rule(self, rule):
#Додавання нового правила до списку правил
        self.rules.append(rule)
#Запис події додавання правила
```

```
        self.db_manager.add_log(f"Firewall rule added: {rule}")

#Метод для видалення існуючого правила з брандмауера
    def remove_rule(self, rule):
#Перевірка наявності правила у списку
        if rule in self.rules:
            self.rules.remove(rule)
#Запис події видалення правила
            self.db_manager.add_log(f"Firewall rule removed: {rule}")

#Метод для виведення всіх поточних правил брандмауера
    def list_rules(self):
#Повернення списку правил
        return self.rules

#Клас для системи виявлення вторгнень (Intrusion Detection System)
class IntrusionDetectionSystem:
#Конструктор класу IntrusionDetectionSystem з ініціалізацією параметрів
    def __init__(self, db_manager):
#Призначення менеджера баз даних для логування подій IDS
        self.db_manager = db_manager
#Ініціалізація прапорця моніторингу як False
        self.monitoring = False

#Метод для запуску моніторингу мережевого трафіку
    def start_monitoring(self):
#Встановлення прапорця моніторингу в True
        self.monitoring = True
#Запис події старту моніторингу IDS
        self.db_manager.add_log("Intrusion Detection System monitoring
started.")
#Запуск окремого потоку для моніторингу мережевого трафіку
        threading.Thread(target=self.monitor_traffic, daemon=True).start()

#Метод для симуляції моніторингу мережевого трафіку
    def monitor_traffic(self):
#Безкінечний цикл моніторингу, поки система активна
        while self.monitoring:
#Генерація фіктивного пакету для аналізу
            packet = self.generate_fake_packet()
#Перевірка пакету на ознаки вторгнення
            intrusion_detected = self.detect_intrusion(packet)
#Якщо виявлено ознаки вторгнення, створити сповіщення
            if intrusion_detected:
                self.generate_alert(packet)
#Пауза перед наступною перевіркою пакету
            time.sleep(1)

#Метод для генерації фіктивного мережевого пакету
    def generate_fake_packet(self):
#Створення фіктивного пакету з випадковою IP-адресою відправника
        source_ip = f"192.168.1.{random.randint(1,254)}"
#Випадкове визначення, чи є пакет зловмисним
        is_malicious = random.choice([True, False, False, False])
#Повернення пакету у вигляді словника з даними
        return {'source_ip': source_ip, 'malicious': is_malicious}

#Метод для виявлення ознак вторгнення в пакеті
    def detect_intrusion(self, packet):
#Перевірка, чи позначено пакет як зловмисний
        if packet.get('malicious', False):
#Запис події виявлення потенційного вторгнення
            self.db_manager.add_log(f"Intrusion detected from
{packet.get('source_ip')}.")
#Повернення True, якщо вторгнення виявлено
            return True
#Повернення False, якщо ознак вторгнення немає
        return False
```

```
#Метод для генерації сповіщення про виявлене вторгнення
    def generate_alert(self, packet):
#Формування повідомлення-сповіщення з інформацією про IP-адресу
        alert_message = f"Alert: Intrusion detected from IP
{packet.get('source_ip')}."
#Запис сповіщення у базі даних
        self.db_manager.add_alert(alert_message)

#Клас для сканування системи на вразливості
class VulnerabilityScanner:
#Конструктор класу VulnerabilityScanner з ініціалізацією параметрів
    def __init__(self, db_manager):
#Призначення менеджера баз даних для логування подій сканування
        self.db_manager = db_manager

#Метод для запуску сканування системи на вразливості
    def scan_system(self):
#Запис події початку сканування на вразливості
        self.db_manager.add_log("Vulnerability scanning initiated.")
#Симуляція сканування з випадковим числом знайдених вразливостей
        vulnerabilities_found = random.randint(0, 5)
#Запис результатів сканування у вигляді кількості виявлених вразливостей
        self.db_manager.add_log(f"Vulnerability scan completed. Vulnerabilities
found: {vulnerabilities_found}.")
#Повернення кількості знайдених вразливостей
        return vulnerabilities_found

#Метод для формування звіту про вразливості
    def report_vulnerabilities(self):
#Отримання кількості вразливостей за допомогою методу scan_system
        count = self.scan_system()
#Формування звіту на основі отриманих даних
        report = f"System Vulnerability Report: {count} vulnerabilities
detected."
#Повернення сформованого звіту
        return report

#Клас для симуляції операцій шифрування та дешифрування даних
class DataEncryption:
#Конструктор класу DataEncryption з ініціалізацією секретного ключа
    def __init__(self, secret_key=None):
#Якщо секретний ключ не заданий, генерувати випадковий ключ
        if secret_key is None:
            self.secret_key = self.generate_secret_key()
        else:
            self.secret_key = secret_key

#Метод для генерації випадкового секретного ключа заданої довжини
    def generate_secret_key(self, length=16):
#Використання випадкового вибору символів для формування ключа
        return ''.join(random.choices(string.ascii_letters + string.digits,
k=length))

#Метод для шифрування даних за допомогою симуляції алгоритму
    def encrypt(self, data):
#Створення порожнього списку для зберігання зашифрованих символів
        encoded_chars = []
#Ітерація по кожному символу в даних
        for i, char in enumerate(data):
#Отримання відповідного символу з секретного ключа
            key_c = self.secret_key[i % len(self.secret_key)]
#Шифрування символу шляхом комбінування з символом ключа з використанням
модульної арифметики
            encoded_c = chr((ord(char) + ord(key_c)) % 256)
#Додавання зашифрованого символу до списку
            encoded_chars.append(encoded_c)
#Об'єднання зашифрованих символів у рядок
        encrypted_data = ''.join(encoded_chars)
#Повернення зашифрованих даних
```

```
        return encrypted_data

#Метод для дешифрування даних, використовуючи зворотній процес
    def decrypt(self, encrypted_data):
#Створення порожнього списку для зберігання розшифрованих символів
        decoded_chars = []
#Ітерація по кожному символу в зашифрованих даних
        for i, char in enumerate(encrypted_data):
#Отримання відповідного символу з секретного ключа
            key_c = self.secret_key[i % len(self.secret_key)]
#Дешифрування символу шляхом віднімання значення символу ключа
            decoded_c = chr((ord(char) - ord(key_c)) % 256)
#Додавання розшифрованого символу до списку
            decoded_chars.append(decoded_c)
#Об'єднання розшифрованих символів у рядок
        decrypted_data = ''.join(decoded_chars)
#Повернення розшифрованих даних
        return decrypted_data

#Клас для реагування на інциденти безпеки в системі
class IncidentResponse:
#Конструктор класу IncidentResponse з ініціалізацією менеджера баз даних
    def __init__(self, db_manager):
#Призначення менеджера баз даних для логування подій реагування
        self.db_manager = db_manager

#Метод для реагування на інцидент безпеки
    def respond_to_incident(self, incident):
#Запис початку реагування на інцидент
        self.db_manager.add_log(f"Responding to incident: {incident}")
#Симуляція процесу реагування шляхом короткого очікування
        time.sleep(2)
#Запис завершення реагування на інцидент
        self.db_manager.add_log(f"Incident resolved: {incident}")

#Метод для ескалації інциденту безпеки
    def escalate_incident(self, incident):
#Запис події ескалації інциденту
        self.db_manager.add_log(f"Incident escalated: {incident}")
#Симуляція процесу ескалації шляхом короткого очікування
        time.sleep(1)
#Запис події обробки ескалованого інциденту
        self.db_manager.add_log(f"Escalated incident handled: {incident}")

#Метод для логування деталей інциденту
    def log_incident(self, incident):
#Запис деталей інциденту у таблицю логів
        self.db_manager.add_log(f"Incident logged: {incident}")

#Клас для модулю розвідки загроз (Threat Intelligence)
class ThreatIntelligence:
#Конструктор класу ThreatIntelligence з ініціалізацією менеджера баз даних
    def __init__(self, db_manager):
#Призначення менеджера баз даних для логування подій розвідки загроз
        self.db_manager = db_manager
#Ініціалізація внутрішньої бази загроз як порожнього списку
        self.threats = []

#Метод для отримання останніх даних розвідки загроз з зовнішнього джерела
    def fetch_latest_threats(self):
#Запис події отримання даних розвідки загроз
        self.db_manager.add_log("Fetching latest threat intelligence data.")
#Симуляція API-запиту до зовнішнього джерела для отримання даних загроз
        try:
#Використання тестового API для отримання даних (симуляція)
            response = requests.get("https://api.example.com/threats",
timeout=5)
#Якщо запит успішний, обробка отриманих даних
            if response.status_code == 200:
```

```
                threats_data = response.json()
#Оновлення внутрішньої бази загроз отриманими даними
                self.threats = threats_data.get("threats", [])
#Запис успішного оновлення даних розвідки загроз
                self.db_manager.add_log("Threat intelligence data updated
successfully.")
            else:
#Запис помилки у випадку невдалого запиту
                self.db_manager.add_log("Failed to fetch threat intelligence
data: Non-200 status.")
        except Exception as e:
#Запис винятку під час виконання API-запиту
            self.db_manager.add_log(f"Exception while fetching threat
intelligence: {str(e)}")

#Метод для оновлення внутрішньої бази загроз
    def update_threat_database(self):
#Отримання останніх даних розвідки загроз
        self.fetch_latest_threats()
#Запис події оновлення бази загроз
        self.db_manager.add_log("Threat database updated.")

#Клас для аналізу логів та виявлення шаблонів подій
class LogAnalyzer:
#Конструктор класу LogAnalyzer з ініціалізацією менеджера баз даних
    def __init__(self, db_manager):
#Призначення менеджера баз даних для доступу до логів
        self.db_manager = db_manager

#Метод для аналізу логів та підрахунку повторюваних подій
    def analyze_logs(self):
#Отримання всіх логів з бази даних
        logs = self.db_manager.get_logs()
#Запис події початку аналізу логів
        self.db_manager.add_log("Log analysis initiated.")
#Створення словника для підрахунку повторюваних подій
        summary = {}
#Ітерація по кожному запису логу
        for log in logs:
            event = log[1]
#Підрахунок кількості повторень кожного типу події
            summary[event] = summary.get(event, 0) + 1
#Запис завершення аналізу логів
        self.db_manager.add_log("Log analysis completed.")
#Повернення словника з підсумками аналізу
        return summary

#Метод для виявлення підозрілих шаблонів у логах
    def detect_patterns(self):
#Отримання підсумків аналізу логів
        summary = self.analyze_logs()
#Ініціалізація списку для збереження виявлених шаблонів
        patterns = []
#Ітерація по підсумкових даних для виявлення повторюваних подій
        for event, count in summary.items():
#Якщо кількість подій перевищує поріг, додати шаблон до списку
            if count > 3:
                patterns.append((event, count))
#Запис події виявлення підозрілих шаблонів
        self.db_manager.add_log("Suspicious log patterns detected.")
#Повернення списку виявлених шаблонів
        return patterns

#Клас для планувальника завдань у системі кібербезпеки
class Scheduler:
#Конструктор класу Scheduler з ініціалізацією порожнього списку завдань
    def __init__(self):
#Ініціалізація списку для збереження запланованих завдань
        self.tasks = []
```

```
#Метод для запланування нового завдання з вказаним інтервалом (у секундах)
    def schedule_task(self, task, interval):
#Додавання завдання та його інтервалу до списку завдань
        self.tasks.append((task, interval))
#Виведення повідомлення про заплановане завдання
        print(f"Task scheduled: {task.__name__} every {interval} seconds.")

#Внутрішній метод для безперервного виконання завдання з певним інтервалом
    def _run_task(self, task, interval):
#Безкінечний цикл виконання завдання
        while True:
#Виконання завдання
            task()
#Пауза перед наступним запуском завдання
            time.sleep(interval)

#Метод для запуску планувальника завдань та виконання всіх запланованих завдань
    def run_scheduler(self):
#Ітерація по кожному запланованому завданню
        for task, interval in self.tasks:
#Запуск нового потоку для виконання завдання
            threading.Thread(target=self._run_task, args=(task, interval),
daemon=True).start()

#Утиліта для симуляції мережевого трафіку з перевіркою через брандмауер та IDS
def simulate_network_traffic(firewall, ids):
#Безкінечний цикл симуляції мережевого трафіку
    while True:
#Генерація фіктивного пакету за допомогою IDS
        packet = ids.generate_fake_packet()
#Перевірка пакету через правила брандмауера
        allowed = firewall.check_packet(packet)
#Якщо пакет заблоковано, запис події в логах
        if not allowed:
            firewall.db_manager.add_log(f"Network traffic simulation: Packet
from {packet.get('source_ip')} dropped.")
#Пауза випадкової тривалості перед наступною симуляцією
        time.sleep(random.uniform(0.5, 2.0))

#Утиліта для симуляції сканування системи на вразливості з виведенням звіту
def simulate_vulnerability_scanning(scanner):
#Запис події початку процесу симуляції сканування
    scanner.db_manager.add_log("Starting simulated vulnerability scanning
process.")
#Отримання звіту за допомогою методу report_vulnerabilities
    report = scanner.report_vulnerabilities()
#Виведення звіту про вразливості в консоль
    print(report)

#Додаткова утиліта для виконання непотрібних операцій (розширення коду)
def extra_dummy_operations():
#Запуск циклу для виконання додаткових операцій
    for i in range(5):
#Виведення повідомлення про виконання додаткової операції
        print(f"Extra operation {i+1} executed.")
#Невелика пауза для симуляції обробки
        time.sleep(0.2)
#Повернення завершення додаткових операцій
    return

#Головна функція для ініціалізації та запуску системи кібербезпеки
def main():
#Створення об'єкту менеджера баз даних
    db_manager = DatabaseManager()
#Створення об'єкту аутентифікації користувачів із використанням менеджера баз
даних
    auth_system = UserAuthentication(db_manager)
#Створення об'єкту брандмауера із використанням менеджера баз даних
```

```
    firewall = Firewall(db_manager)
#Створення об'єкту системи виявлення вторгнень із використанням менеджера баз
даних
    ids = IntrusionDetectionSystem(db_manager)
#Створення об'єкту сканера вразливостей із використанням менеджера баз даних
    scanner = VulnerabilityScanner(db_manager)
#Створення об'єкту для шифрування даних
    encryption = DataEncryption()
#Створення об'єкту для реагування на інциденти
    incident_response = IncidentResponse(db_manager)
#Створення об'єкту для розвідки загроз
    threat_intel = ThreatIntelligence(db_manager)
#Створення об'єкту для аналізу логів
    log_analyzer = LogAnalyzer(db_manager)
#Створення об'єкту планувальника завдань для періодичних операцій
    scheduler = Scheduler()

#Реєстрація стандартного адміністративного користувача
    auth_system.register_user("admin", "admin123")
#Спроба входу адміністративного користувача в систему
    if auth_system.login_user("admin", "admin123"):
        print("Admin user logged in successfully.")
    else:
        print("Admin user login failed.")

#Запуск системи виявлення вторгнень (IDS)
    ids.start_monitoring()

#Запланування сканування системи на вразливості кожні 10 секунд
    scheduler.schedule_task(lambda: simulate_vulnerability_scanning(scanner),
10)
#Запланування оновлення даних розвідки загроз кожні 15 секунд
    scheduler.schedule_task(lambda: threat_intel.update_threat_database(), 15)
#Запланування аналізу логів кожні 20 секунд із виведенням виявлених шаблонів
    scheduler.schedule_task(lambda: print("Log Patterns:",
log_analyzer.detect_patterns()), 20)

#Запуск окремого потоку для симуляції мережевого трафіку
    threading.Thread(target=simulate_network_traffic, args=(firewall, ids),
daemon=True).start()
#Виклик додаткових операцій для розширення коду
    extra_dummy_operations()

#Головний цикл для підтримки роботи системи
    try:
        while True:
#Пауза в головному циклі для стабільності роботи
            time.sleep(1)
    except KeyboardInterrupt:
#Запис події зупинки системи кібербезпеки
        db_manager.add_log("Cybersecurity system shutdown initiated.")
#Виведення повідомлення про зупинку системи
        print("Cybersecurity system shutting down.")
#Завершення виконання програми
        sys.exit(0)

#Виконання головної функції при запуску скрипту
if __name__ == '__main__':
    main()
```

Файл UserBehaviorAnalysis.py

```
import os
import sys
import time
import threading
import shutil
import base64
import datetime
import random
import string
import numpy as np
from sklearn.ensemble import IsolationForest

class UserBehaviorAnalysis:
    def __init__(self):
        self.user_events = {}
    def record_event(self, user_id, event_type, details, timestamp=None):
        if timestamp is None:
            timestamp = datetime.datetime.now()
        if user_id not in self.user_events:
            self.user_events[user_id] = []
        self.user_events[user_id].append({'event_type': event_type, 'details':
details, 'timestamp': timestamp})
    def get_events(self, user_id):
        return self.user_events.get(user_id, [])
    def analyze_behavior(self):
        results = {}
        for user_id, events in self.user_events.items():
            count = len(events)
            if count > 5:
                results[user_id] = 'suspicious'
            else:
                results[user_id] = 'normal'
        return results
    def get_suspicious_users(self):
        analysis = self.analyze_behavior()
        suspicious = []
        for user, status in analysis.items():
            if status == 'suspicious':
                suspicious.append(user)
        return suspicious

class LogMLMonitor:
    def __init__(self):
        self.logs = []
        self.model = IsolationForest(contamination=0.1, random_state=42)
        self.trained = False
    def load_logs(self, log_list):
        for log in log_list:
            self.logs.append(log)
    def preprocess_logs(self):
        features = []
```

```
        for log in self.logs:
            length = len(log)
            words = len(log.split())
            digits = sum(c.isdigit() for c in log)
            features.append([length, words, digits])
        return np.array(features)
    def train_model(self):
        features = self.preprocess_logs()
        if len(features) > 0:
            self.model.fit(features)
            self.trained = True
    def predict_anomalies(self):
        if not self.trained:
            self.train_model()
        features = self.preprocess_logs()
        predictions = self.model.predict(features)
        anomalies = []
        for i, pred in enumerate(predictions):
            if pred == -1:
                anomalies.append(self.logs[i])
        return anomalies

class MultiLevelEncryption:
    def __init__(self, key=None):
        if key is None:
            self.key = ''.join(random.choices(string.ascii_letters +
string.digits, k=16))
        else:
            self.key = key
        self.shift = sum(ord(c) for c in self.key) % 26
    def caesar_encrypt(self, text):
        result = ''
        for char in text:
            if char.isalpha():
                start = ord('A') if char.isupper() else ord('a')
                result += chr((ord(char) - start + self.shift) % 26 + start)
            else:
                result += char
        return result
    def caesar_decrypt(self, text):
        result = ''
        for char in text:
            if char.isalpha():
                start = ord('A') if char.isupper() else ord('a')
                result += chr((ord(char) - start - self.shift) % 26 + start)
            else:
                result += char
        return result
    def reverse_string(self, text):
        return text[::-1]
    def encrypt(self, plaintext):
        step1 = self.caesar_encrypt(plaintext)
        step2 = self.reverse_string(step1)
```

```
        step3 = base64.b64encode(step2.encode('utf-8')).decode('utf-8')
        return step3
    def decrypt(self, ciphertext):
        step1 = base64.b64decode(ciphertext.encode('utf-8')).decode('utf-8')
        step2 = self.reverse_string(step1)
        step3 = self.caesar_decrypt(step2)
        return step3

class DDoSProtection:
    def __init__(self, threshold=10, time_window=60):
        self.requests = {}
        self.threshold = threshold
        self.time_window = time_window
        self.blocked_ips = set()
        self.lock = threading.Lock()
    def register_request(self, ip):
        with self.lock:
            now = time.time()
            if ip not in self.requests:
                self.requests[ip] = []
            self.requests[ip].append(now)
    def clean_requests(self):
        with self.lock:
            now = time.time()
            for ip in list(self.requests.keys()):
                self.requests[ip] = [t for t in self.requests[ip] if now - t <=
self.time_window]
                if not self.requests[ip]:
                    del self.requests[ip]
    def check_ip(self, ip):
        self.clean_requests()
        with self.lock:
            count = len(self.requests.get(ip, []))
            if count > self.threshold:
                self.blocked_ips.add(ip)
                return False
            return True
    def get_blocked_ips(self):
        return list(self.blocked_ips)
    def reset_blocked_ips(self):
        with self.lock:
            self.blocked_ips.clear()

class BackupRecovery:
    def __init__(self):
        self.backup_history = []
    def backup(self, source_dir, backup_dir):
        if not os.path.exists(backup_dir):
            os.makedirs(backup_dir)
        timestamp = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        backup_subdir = os.path.join(backup_dir, 'backup_' + timestamp)
        os.makedirs(backup_subdir)
        for root, dirs, files in os.walk(source_dir):
```

```
            relative_path = os.path.relpath(root, source_dir)
            dest_dir = os.path.join(backup_subdir, relative_path)
            if not os.path.exists(dest_dir):
                os.makedirs(dest_dir)
            for file in files:
                src_file = os.path.join(root, file)
                dst_file = os.path.join(dest_dir, file)
                shutil.copy2(src_file, dst_file)
        self.backup_history.append(backup_subdir)
        return backup_subdir
    def restore(self, backup_subdir, target_dir):
        if not os.path.exists(target_dir):
            os.makedirs(target_dir)
        for root, dirs, files in os.walk(backup_subdir):
            relative_path = os.path.relpath(root, backup_subdir)
            dest_dir = os.path.join(target_dir, relative_path)
            if not os.path.exists(dest_dir):
                os.makedirs(dest_dir)
            for file in files:
                src_file = os.path.join(root, file)
                dst_file = os.path.join(dest_dir, file)
                shutil.copy2(src_file, dst_file)
        return target_dir
    def list_backups(self):
        return self.backup_history

def main():
    uba = UserBehaviorAnalysis()
    uba.record_event("user1", "login", "successful", datetime.datetime.now())
    uba.record_event("user1", "view", "homepage", datetime.datetime.now())
    uba.record_event("user1", "click", "button1", datetime.datetime.now())
    uba.record_event("user1", "download", "file1", datetime.datetime.now())
    uba.record_event("user1", "upload", "file2", datetime.datetime.now())
    uba.record_event("user1", "logout", "user logged out",
datetime.datetime.now())
    uba.record_event("user2", "login", "failed", datetime.datetime.now())
    uba.record_event("user2", "login", "failed", datetime.datetime.now())
    uba.record_event("user2", "login", "failed", datetime.datetime.now())
    uba.record_event("user2", "login", "failed", datetime.datetime.now())
    uba.record_event("user2", "login", "failed", datetime.datetime.now())
    suspicious_users = uba.get_suspicious_users()
    print("Suspicious Users:", suspicious_users)
    logs = ["User user1 logged in", "User user2 failed login", "File download
initiated", "Suspicious activity detected", "Multiple failed logins observed"]
    ml_monitor = LogMLMonitor()
    ml_monitor.load_logs(logs)
    ml_monitor.train_model()
    anomalies = ml_monitor.predict_anomalies()
    print("Anomalous Logs:", anomalies)
    mle = MultiLevelEncryption("SecretKey123")
    original_text = "This is a sensitive message."
    encrypted_text = mle.encrypt(original_text)
    decrypted_text = mle.decrypt(encrypted_text)
```

```
    print("Original Text:", original_text)
    print("Encrypted Text:", encrypted_text)
    print("Decrypted Text:", decrypted_text)
    ddos = DDoSProtection(threshold=5, time_window=10)
    ip_list = ["192.168.1.10", "192.168.1.11", "192.168.1.12"]
    for _ in range(7):
        for ip in ip_list:
            ddos.register_request(ip)
            allowed = ddos.check_ip(ip)
            print("IP", ip, "allowed:", allowed)
        time.sleep(1)
    blocked = ddos.get_blocked_ips()
    print("Blocked IPs:", blocked)
    br = BackupRecovery()
    source_dir = "source_data"
    backup_dir = "backups"
    if not os.path.exists(source_dir):
        os.makedirs(source_dir)
    with open(os.path.join(source_dir, "file1.txt"), "w") as f:
        f.write("Data in file 1.")
    with open(os.path.join(source_dir, "file2.txt"), "w") as f:
        f.write("Data in file 2.")
    backup_location = br.backup(source_dir, backup_dir)
    print("Backup created at:", backup_location)
    restore_dir = "restored_data"
    br.restore(backup_location, restore_dir)
    print("Data restored to:", restore_dir)
    backups = br.list_backups()
    print("Backup History:", backups)

if __name__ == "__main__":
    main()
```

Файл AccessControl.py

```
import os
import sys
import time
import threading
import hashlib
import random
import string
import re
import subprocess
import datetime

class AccessControl:
    def __init__(self):
        self.users = {}
        self.roles = {}
        self.sessions = {}
    def add_user(self, username, password):
        hashed = hashlib.sha256(password.encode()).hexdigest()
        self.users[username] = {'password': hashed, 'roles': []}
    def remove_user(self, username):
        if username in self.users:
            del self.users[username]
    def assign_role(self, username, role):
        if username in self.users:
            self.users[username]['roles'].append(role)
            if role not in self.roles:
                self.roles[role] = []
            self.roles[role].append(username)
    def revoke_role(self, username, role):
        if username in self.users and role in self.users[username]['roles']:
            self.users[username]['roles'].remove(role)
            if role in self.roles and username in self.roles[role]:
                self.roles[role].remove(username)
    def check_permission(self, username, permission):
        roles = self.users.get(username, {}).get('roles', [])
        for role in roles:
            if permission in role:
                return True
        return False
    def create_session(self, username):
        token = ''.join(random.choices(string.ascii_letters + string.digits,
k=32))
        self.sessions[token] = {'username': username, 'created':
datetime.datetime.now()}
        return token
    def invalidate_session(self, token):
        if token in self.sessions:
            del self.sessions[token]
    def get_user_roles(self, username):
        return self.users.get(username, {}).get('roles', [])
    def list_users(self):
```

```
        return list(self.users.keys())
    def list_roles(self):
        return list(self.roles.keys())

class SIEMInterface:
    def __init__(self):
        self.event_queue = []
        self.connected = False
        self.filter_rules = []
    def connect(self):
        self.connected = True
    def disconnect(self):
        self.connected = False
    def add_filter(self, rule):
        self.filter_rules.append(rule)
    def remove_filter(self, rule):
        if rule in self.filter_rules:
            self.filter_rules.remove(rule)
    def send_event(self, event):
        if self.connected:
            self.event_queue.append(event)
    def flush_events(self):
        if self.connected:
            events = self.event_queue.copy()
            self.event_queue = []
            return events
        return []
    def receive_command(self):
        if self.connected:
            return "No command"
        return "Not connected"
    def process_events(self):
        if self.connected:
            processed = []
            for event in self.event_queue:
                match = True
                for rule in self.filter_rules:
                    if rule not in event:
                        match = False
                        break
                if match:
                    processed.append(event)
            self.event_queue = []
            return processed
        return []

class WebAppFirewall:
    def __init__(self):
        self.rules = []
    def add_rule(self, rule):
        self.rules.append(rule)
    def remove_rule(self, rule):
        if rule in self.rules:
```

```
                self.rules.remove(rule)
        def inspect_request(self, request):
            if self.detect_sql_injection(request):
                return False
            if self.detect_xss(request):
                return False
            if self.detect_file_inclusion(request):
                return False
            return True
        def detect_sql_injection(self, request):
            pattern = r"(?i)(union|select|drop|insert|delete|update|--)"
            if re.search(pattern, request):
                return True
            return False
        def detect_xss(self, request):
            pattern = r"(?i)<script.*?>.*?</script>"
            if re.search(pattern, request):
                return True
            return False
        def detect_file_inclusion(self, request):
            pattern = r"(\.\./|\.\.\\)"
            if re.search(pattern, request):
                return True
            return False

    class FileIntegrityMonitor:
        def __init__(self):
            self.files = {}
            self.monitoring = False
            self.lock = threading.Lock()
        def add_file(self, file_path):
            if os.path.isfile(file_path):
                with open(file_path, 'rb') as f:
                    content = f.read()
                file_hash = hashlib.sha256(content).hexdigest()
                with self.lock:
                    self.files[file_path] = file_hash
        def remove_file(self, file_path):
            with self.lock:
                if file_path in self.files:
                    del self.files[file_path]
        def compute_hash(self, file_path):
            if os.path.isfile(file_path):
                with open(file_path, 'rb') as f:
                    content = f.read()
                return hashlib.sha256(content).hexdigest()
            return None
        def check_file(self, file_path):
            current_hash = self.compute_hash(file_path)
            with self.lock:
                original_hash = self.files.get(file_path)
            if current_hash != original_hash:
                return False
```

```
        return True
    def monitor_files(self, interval=5):
        self.monitoring = True
        while self.monitoring:
            with self.lock:
                for file_path in list(self.files.keys()):
                    if not self.check_file(file_path):
                        print("File changed:", file_path)
            time.sleep(interval)
    def stop_monitoring(self):
        self.monitoring = False

class RemoteMonitorManagement:
    def __init__(self):
        self.status = "OK"
        self.logs = []
    def get_system_status(self):
        self.logs.append("System status requested at " +
datetime.datetime.now().isoformat())
        return self.status
    def execute_command(self, command):
        try:
            output = subprocess.check_output(command, shell=True,
stderr=subprocess.STDOUT, timeout=5)
            result = output.decode()
        except subprocess.CalledProcessError as e:
            result = e.output.decode()
        self.logs.append("Executed command: " + command)
        return result
    def restart_service(self, service_name):
        result = "Service " + service_name + " restarted."
        self.logs.append(result)
        return result
    def update_configuration(self, config):
        self.logs.append("Configuration updated: " + str(config))
        return True
    def get_remote_logs(self):
        return self.logs
    def remote_shutdown(self):
        self.status = "Shutdown"
        self.logs.append("Remote shutdown initiated.")
        return "Shutdown initiated"

def main():
    ac = AccessControl()
    ac.add_user("alice", "password1")
    ac.add_user("bob", "password2")
    ac.assign_role("alice", "admin")
    ac.assign_role("bob", "user")
    session_alice = ac.create_session("alice")
    print("Alice session:", session_alice)
    print("Users:", ac.list_users())
    print("Roles:", ac.list_roles())
```

```
    siem = SIEMInterface()
    siem.connect()
    siem.add_filter("error")
    siem.send_event("System error occurred in module A")
    siem.send_event("Normal operation in module B")
    events = siem.flush_events()
    print("SIEM events:", events)
    waf = WebAppFirewall()
    waf.add_rule("block_sql")
    request1 = "SELECT * FROM users WHERE id=1"
    request2 = "<script>alert('xss')</script>"
    request3 = "/index.php?page=home"
    print("Request1 allowed:", waf.inspect_request(request1))
    print("Request2 allowed:", waf.inspect_request(request2))
    print("Request3 allowed:", waf.inspect_request(request3))
    fim = FileIntegrityMonitor()
    test_file = "test_file.txt"
    with open(test_file, "w") as f:
        f.write("Original content")
    fim.add_file(test_file)
    print("File integrity check:", fim.check_file(test_file))
    threading.Thread(target=fim.monitor_files, args=(2,), daemon=True).start()
    time.sleep(3)
    with open(test_file, "w") as f:
        f.write("Modified content")
    time.sleep(3)
    fim.stop_monitoring()
    rmm = RemoteMonitorManagement()
    status = rmm.get_system_status()
    print("System status:", status)
    cmd_output = rmm.execute_command("echo Hello World")
    print("Command output:", cmd_output)
    restart_result = rmm.restart_service("nginx")
    print("Restart result:", restart_result)
    config_update = rmm.update_configuration({"setting": "value"})
    print("Config update:", config_update)
    logs = rmm.get_remote_logs()
    print("Remote logs:", logs)
    shutdown_msg = rmm.remote_shutdown()
    print("Shutdown message:", shutdown_msg)

if __name__ == "__main__":
    main()
```