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 забезпечення безпеки ЦОД”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Романов І.О.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 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 *123 “*Комп’ютерна інженерія*”*
Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

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

Романову Івану Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи забезпечення безпеки ЦОД*

2. Керівник роботи *Дрєєва Ганна Миколаївна, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 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 забезпечення безпеки ЦОД*

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 в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*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
_____ Дрєєва Г.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Романов І.О.
(прізвище та ініціали)

АНОТАЦІЯ

Романов І.О. Програмне забезпечення системи забезпечення безпеки ЦОД. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи забезпечення безпеки ЦОД.

Метою розробки є програмне забезпечення системи забезпечення безпеки ЦОД.

Результат роботи – програмна реалізація системи забезпечення безпеки ЦОД.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, ЦОД

ABSTRACT

Romanov I.O. Software for the data center security system. 123 Computer Engineering. 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 that is intended for the data center security system.

The purpose of the development is the software for the data center security system.

The result of the work is the software implementation of the data center security system.

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 the software are provided.

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

The program was developed in the Python environment.

Keywords: computer engineering, data center

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
НСД	–	несанкціонований доступ
ПС	–	програмна середа
СВВ	–	система виявлення вторгнень
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних
ЦОД	–	центр обробки даних

ВСТУП

Актуальність теми. Центри обробки даних відіграють важливу роль у забезпеченні кібербезпеки, створюючи безпечне та надійне середовище для зберігання та обробки даних.

Треба враховувати не лише фізичну безпеку, але й багато інших факторів для забезпечення захисту.

Центри обробки даних розроблені з такими функціями фізичної безпеки, як камери спостереження, засоби контролю доступу та системи біометричної автентифікації, щоб запобігти несанкціонованому доступу до об'єкта. Однак на цьому не обмежуються такі сфери, які також необхідно розглянути та захистити:

– Безпека мережі. Центри обробки даних використовують передові технології мережевої безпеки, такі як брандмауери, системи виявлення та запобігання вторгненням, а також VPN для захисту своїх мереж і запобігання несанкціонованому доступу до даних. Вони також використовують передові методи шифрування для захисту даних під час передавання та зберігання.

– Надмірність. Центри обробки даних, як правило, мають резервні системи живлення та охолодження, а також численні підключення до Інтернету, щоб гарантувати їхню працездатність навіть у разі відключення електроенергії чи збою мережі. Це допомагає запобігти простоям і забезпечити постійну доступність даних, коли вони потрібні.

– Експертиза. У центрах обробки даних працюють команди експертів з кібербезпеки, які навчені новітнім технологіям безпеки та передовим практикам. Ці експерти відповідають за моніторинг систем безпеки центру обробки даних, виявлення потенційних загроз і швидке реагування на інциденти безпеки.

– Відповідність. На все вищезазначене поширюються суворі нормативні вимоги, пов'язані з конфіденційністю та безпекою даних, наприклад Загальний регламент захисту даних (GDPR) і Стандарт безпеки даних індустрії платіжних

карток (PCI DSS). Центри обробки даних повинні відповідати цим правилам, щоб гарантувати захист даних клієнтів від кіберзагроз і кібервійни.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи забезпечення безпеки ЦОД.

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

– Огляд існуючих систем забезпечення безпеки ЦОД.

– Дослідження системи забезпечення безпеки ЦОД.

– Програмна реалізація системи забезпечення безпеки ЦОД.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі забезпечення безпеки ЦОД.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення безпеки ЦОД, 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

Система призначена для захисту ЦОД від DDoS-атак. Сучасні тенденції розвитку інформаційних технологій, а так само способи зберігання, обробки й одержання інформації усе більше й більше переходять в інтерактивне середовище – різні WEB ресурси, профільні сайти, форуми. Це дає можливість швидко й комфортно доносити інформацію кінцевим користувачам, вчасно вносити необхідні зміни в її зміст. Інформація, розташовувана в мережі Інтернет, стала коштовним джерелом який призначений для інтерактивно-активної цільовий аудиторій сучасного суспільства.

Як приклад, можна привести сайти державних установ, які надають інтерактивний доступ до таким важливим даних як: Закони, Кодекси, Укази, Розпорядження, статистичні дані, профільні сайти, форуми.

Якщо згадати останні які відбулися в Україні події, пов'язані з навмисним блокуванням роботи сайтів МВС України, Президента України, Служби Безпеки України й ін. можна зробити вивід, що існуючі системи захисту мережної інфраструктури державних установ абсолютно не здатні забезпечити безперебійну роботу WEB сервісів. Це вказує на повну неспроможність застосовуваних механізмів захисту, слабкий апаратний рівень використовуваного встаткування, некомпетентність і низький рівень підготовки профільних фахівців.

Атаки типу DoS/DDoS стають усе більше розповсюдженими. Угадати, хто стане наступною жертвою досить важко. Важливо розуміти, що неготовність протидіяти й попереджати даного роду погрози може стати дуже зручним і діючим механізмом у політичному протистоянні, що може привести до загострення соціального невдоволення й виникненню безладу. Інтернет маніпуляції з інформацією, її підміни, перекручування, блокування доступу,

можуть привести до перекручування фактів, які мають місце бути, що може спровокувати соціальний протест у суспільстві, підірвати довіру до влади.

Як «атака» може бути класифікований «підвищений інтерес» до Інтернет-ресурсам установ, організацій, під час проведення соціальні акції, флеш-мобів, виборів (відвідування сайту ЦВК під час процесу підрахунку голосів народного волевиявлення). При цьому важливо врахувати, що атаки типу DDoS проводяться паралельно з легітимними запитами користувачів, яким дійсно необхідний доступ до інформації.

Необхідно забезпечувати безперебійний доступ до ресурсів для легітимних користувачів, при цьому блокувати зловмисників і обмежувати ненавмисно активних користувачів, через які може різко збільшуватися навантаження на встаткування, що забезпечує різні WEB сервіси.

Загальні рекомендації

Варто розуміти, що питання захисту від DoS/DDoS атак досить комплексний і вимагає детального аналізу в кожному конкретному випадку. Від типу атаки, задіяних під атаку потужностей, використовуваних засобів і багатьох інших факторів залежить наскільки ефективними будуть запропоновані рішення.

Якщо врахувати, що типове підключення до провайдеру послуг Інтернет має швидкість не більше 10 Гбіт/с, і поточний канал в Інтернет 1 Гбіт/с і менш, те цілком виправдано використовувати наведені в описі засобу захисту від найпоширеніших типів DoS-атак.

Крім іншого, необхідно переконається, що провайдер послуг Інтернет також приймає деякі заходи щодо зм'якшення DDoS атак.

Якщо атака носить нетривіальний характер, використовує складні техніки обходу засобів захисту або має сумарну потужність перевищуючу ємність каналів зв'язку – необхідно використовувати спеціалізовані засоби захисту від DDoS.

Також, необхідно врахувати, що DDoS може бути організований і легітимними запитами користувачів внаслідок «підвищеного інтересу» до Інтернет-ресурсів організації (соціальні акції, флеш-моби). У цьому випадку

необхідно забезпечити достатні обчислювальні потужності серверної ферми для передбачуваного максимального навантаження.

Система моніторінгу й узагальнення подій інформаційної безпеки – важливий елемент системи захисту. Розробку таких рішень необхідно врахувати.

1.2 Область застосування

Під кібервійною розуміють використання комп'ютерних технологій для здійснення атак на інші країни, організації чи окремих осіб у стратегічних чи військових цілях. Це передбачає використання цифрових інструментів і методів для націлювання та знищення інформаційних систем супротивника, включаючи їх комп'ютерні мережі, інфраструктуру та комунікації.

Використання кіберзброї може мати значні наслідки, включно з потенціалом заподіяння фізичної шкоди або втрати життя на додаток до економічної та політичної шкоди.

Як збірні центри обробки даних використовуються в кібервійні?

Для швидкого розгортання обчислювальних ресурсів для підтримки наступальних або оборонних кібероперацій. Ось кілька способів їх використання:

Наступальні кібероперації.

Збірні центри обробки даних можна використовувати для підтримки наступальних кібероперацій, забезпечуючи масштабовану та гнучку обчислювальну інфраструктуру, яку можна швидко розгорнути в місці поблизу цілі. Це зменшує затримку та збільшує швидкість атаки, що може бути вирішальним у сценарії кібервійни. Наприклад, національна держава може використовувати збірний центр обробки даних для запуску розподіленої атаки типу «відмова в обслуговуванні» (DDoS) проти критичної інфраструктури, такої як енергосистема або фінансова система. Розмістивши обчислювальні ресурси ближче до цілі, зловмисник може здійснити ефективнішу та ефективнішу атаку.

Оборонні кібероперації.

Збірні центри обробки даних також можна використовувати для підтримки захисних кібероперацій, забезпечуючи масштабовану та гнучку інфраструктуру, яку можна швидко розгорнути в місцях поблизу критичної інфраструктури або конфіденційних даних. Це дозволяє захисним кібероператорам швидко виявляти кіберзагрози та реагувати на них, що може мати вирішальне значення для пом'якшення наслідків кібератаки. Наприклад, постачальник критичної інфраструктури може використовувати готовий центр обробки даних для моніторингу та реагування на кіберзагрози своїм системам.

Red Teaming.

Збірні центри обробки даних можуть бути використані для підтримки вправ червоної команди, забезпечуючи реалістичне середовище для імітації кібератаки. Команда Red Team може використовувати готовий центр обробки даних для запуску імітованих атак на інфраструктуру організації або конфіденційні дані, дозволяючи організації виявляти вразливі місця та покращувати свій кіберзахист. Наприклад, фінансова установа може використовувати готовий центр обробки даних для імітації фішингової атаки на своїх співробітників, перевіряючи їх здатність виявляти атаку та реагувати на неї.

Навчання та моделювання

Збірні центри обробки даних також можна використовувати для навчання та моделювання, дозволяючи кібероператорам відпрацьовувати наступальні та оборонні кібероперації в реалістичному середовищі. Це може допомогти організаціям покращити свої кібер-можливості та підготуватися до сценарію кібервійни. Наприклад, військова організація може використовувати збірний центр обробки даних для навчання своїх кібероператорів різноманітним сценаріям, у тому числі тим, які може бути важко відтворити в традиційному центрі обробки даних.

Збірні центри обробки даних забезпечують гнучку та масштабовану інфраструктуру для операцій у кібервійні, дозволяючи організаціям швидко

розгортати обчислювальні ресурси у віддалених місцях для підтримки наступальних або оборонних кібероперацій, навчань червоних команд, а також цілей навчання та моделювання.

Які типи організацій, що беруть участь у кібервійні, використовують збірні центри обробки даних

Збірні центри обробки даних можуть використовуватися багатьма організаціями, які беруть участь у кібервійні, зокрема:

– Національні держави.
– Військові та служби безпеки.
– Розвідувальні агентства
– Постачальники критичної інфраструктури

Збірні центри обробки даних використовуються багатьма організаціями, які беруть участь у кібервійні, забезпечуючи гнучку та масштабовану інфраструктуру для підтримки наступальних і оборонних кібероперацій, збору розвідданих, а також цілей навчання та моделювання.

Окрім живлення, охолодження, технічної інфраструктури, протипожежного захисту, управління та моніторингу, збірні будівлі включають низку спеціалізованих функцій, включаючи RF, EMP, акустику та схвалену NPSA вибухову, балістичну та фізичну захист, для захисту від електронних і фізичних атак.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи забезпечення безпеки ЦОД, 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

Tracenet і його рішення для центрів обробки даних і кібербезпеки

Tracenet IT Solutions пропонує ряд індивідуальних рішень для забезпечення надійності, доступності, моніторингу та інформаційної безпеки, незалежно від того, чи розташована інфраструктура у фізичних центрах обробки даних чи в хмарі.

Завдяки підтвердженому досвіду з 2008 року та міжнародній присутності компанія обслуговує клієнтів у таких регіонах, як Латинська Америка, Європа, Африка та Азія, гарантуючи досконалість та технологічні інновації.

Пропоновані рішення для проектів ЦОД:

– Консолідація та резервне копіювання серверів.

– Створення індивідуальних хмар (приватних, публічних і гібридних).

– Проекти фізичних центрів обробки даних (структуровані кабелі, ДБЖ, фізична безпека, прецизійне кондиціонування повітря та генератори).

– Проекти Unified Fabric для інтеграції трафіку Ethernet і SAN.

– Проекти уніфікованих обчислень і зберігання.

– Віртуалізація серверів і мережевих ресурсів.

– Побудова передових центрів обробки даних в контейнерах.

Маючи офіси в таких містах, як Ріо-де-Жанейро, Сан-Паулу та Санта-Катаріна, а також базу в Маямі, Tracenet відома своїми сертифікатами та партнерством з основними виробниками IT.

Центри обробки даних часто є цінними цілями для зловмисників. Це означає, що вони повинні досягти та підтримувати найвищі стандарти безпеки.

Пам’ятаючи про це, ось короткий посібник зі стратегій і найкращих практик впровадження надійної кібербезпеки в центрах обробки даних.

Заходи периметральної оборони

Кібербезпека залежить від фізичної безпеки. Це означає, що захищати свої фізичні периметри так само важливо, як і цифрові периметри.

Ваш основний фізичний периметр – це периметр навколо самого центру обробки даних. Зазвичай, однак, внутрішня частина цього периметра повинна бути зонована. Це створить внутрішні периметри, які також необхідно захистити.

Подібним чином ваш основний цифровий периметр є межею між вашою внутрішньою мережею та публічним Інтернетом. Однак вашу внутрішню мережу слід зонувати, щоб створити внутрішні периметри. Як і фізичні внутрішні периметри, вони потребують власного захисту.

Захист фізичного або цифрового периметра по суті полягає в забезпеченні того, щоб люди та/або дані могли проходити через нього лише у визначених точках. Ці точки мають бути захищені надійними заходами автентифікації, щоб лише авторизовані користувачі та дані могли пройти.

Безпека мережі в центрах обробки даних

Безпека сучасної мережі базується на поєднанні сегментації та моніторингу. Сегментація мережі по суті означає створення внутрішніх бар'єрів у мережі. Вони фактично функціонують як блоки стримування. Вони значно ускладнюють переміщення користувачів або даних у мережі без належного авторизації. Це допомагає обмежити шкоду, яку вони можуть завдати.

Зараз моніторинг здійснюється за допомогою автоматизованих засобів. Основним захисним інструментом безпеки мережі є система виявлення та запобігання вторгненням (IDPS). Вони іноді відомі як брандмауери наступного покоління або уніфіковані брандмауери. Це пояснюється тим, що вони поєднують функції брандмауера, системи запобігання вторгненням і системи виявлення вторгнень.

IDPS контролюють як зовнішній периметр, так і внутрішній простір мережі. Вони забезпечують дотримання політики безпеки та оцінюють трафік на наявність відомих загроз. Завдяки штучному інтелекту (ШІ) багато IDPS також можуть ефективно виявляти загрози без розпізнаної сигнатури. Вони можуть автоматично реагувати на загрози та/або позначати їх мережевим адміністраторам.

Протоколи автентифікації користувачів

Усі сучасні підприємства повинні впроваджувати контроль доступу на основі ролей (RBAC). По суті, це означає, що користувачам надається доступ до ресурсів лише в тому обсязі, який необхідний для виконання їх роботи. Крім того, слід регулярно переглядати засоби контролю доступу, щоб переконатися, що вони все ще дійсні (тобто актуальні).

Впровадження надійних протоколів автентифікації гарантує, що доступ користувачів використовується лише за призначенням. Багатофакторна автентифікація тепер використовується як стандарт у середовищах центрів обробки даних. Традиційно це було те, що ви знаєте (наприклад, пароль), плюс те, що у вас є (наприклад, маркер доступу). Тепер він також часто містить те, що ви є (біометричні дані).

Навіть із надійними протоколами автентифікації важливо постійно відстежувати поведінку користувачів. Тепер це можна зробити за допомогою автоматизованих інструментів.

Методи шифрування даних

Шифрування даних гарантує, що їх можуть прочитати лише люди, які мають доступ до відповідного ключа шифрування. Якщо реалізовано належне керування ключами, це означає, що зашифровані дані марні для будь-кого, хто отримує до них доступ без належного авторизації.

З огляду на це, шифрування настільки ж ефективне, як і керування ключами шифрування. Це означає, що необхідно ретельно керувати процесами генерації, зберігання, розподілу, ротації та анулювання ключів.

Механізми депонування та відновлення ключів є важливими для забезпечення доступності та стійкості даних, особливо в сценаріях, пов’язаних із втратою або компрометацією ключів. Інтеграція з системами керування ідентифікацією та доступом (IAM) сприяє централізованому управлінню ключами, оптимізації адміністративних завдань і посиленню контролю безпеки.

Управління вразливістю

Управління вразливістю передбачає безперервне виявлення та оцінку вразливостей у апаратному забезпеченні, програмному забезпеченні та мережевих компонентах.

Регулярне сканування вразливостей і тестування на проникнення є важливими компонентами програми проактивного керування вразливістю. Інструменти сканування вразливостей використовуються для виявлення слабких місць у системах, програмах і мережевих конфігураціях, що дозволяє організаціям усувати вразливості до того, як ними зможуть скористатися зловмисники.

Тестування на проникнення моделює кібератаки в реальному світі, щоб оцінити ефективність засобів контролю безпеки та виявити потенційні прогалини в захисті, надаючи цінну інформацію для посилення загальної безпеки інфраструктури центру обробки даних.

Аудити безпеки та відповідності

Завдяки систематичному оцінюванню ефективності засобів контролю та практики безпеки аудити допомагають виявити вразливі місця, прогалини та області, які потрібно вдосконалити в інфраструктурі центру обробки даних.

Завдяки ретельному аналізу та оцінці організації можуть завчасно усунути ризики безпеки та відповідності. Це покращить їх здатність виявляти, запобігати та реагувати на потенційні загрози та порушення.

Процеси внутрішнього та зовнішнього аудиту є невід’ємними компонентами аудиту безпеки в центрах обробки даних. Внутрішні аудити, що проводяться групами внутрішнього аудиту або призначеним персоналом, оцінюють дотримання внутрішньої політики, процедур і заходів безпеки.

Зовнішні аудити, що проводяться незалежними сторонніми аудиторами або регулюючими органами, оцінюють дотримання зовнішніх стандартів і правил. Як внутрішні, так і зовнішні аудити дають цінну інформацію про ефективність заходів безпеки та визначають області, які потребують виправлення чи покращення.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

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

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

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання

програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

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

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

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

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

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

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

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

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

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

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

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

6. Тестування (автоматизація цього процесу).

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

Бібліотеки Python

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

Найвідоміші фреймворки для мови програмування Python

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

Серед найпопулярніших фреймворків для Python:

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

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

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

Популярні Python IDE

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

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 забезпечення безпеки ЦОД.

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 контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Оскільки компанії та окремі особи генерують постійно зростаючі обсяги даних – завдяки таким тенденціям, як Інтернет речей (IoT), великі дані, штучний інтелект і хмарні обчислення – центри обробки даних стали важливими для зберігання, обробки та розповсюдження інформації.

Зі зростаючим значенням ці центри також стали пріоритетними цілями для кіберзлочинців, які прагнуть використовувати вразливі місця для викрадення даних, зриву операцій і завдати фінансової та репутаційної шкоди.

У цьому сценарії захист від загроз вимагає постійних зусиль, які поєднують передові технології, надійні процеси та обізнаність команди.

Кібербезпека в центрах обробки даних є важливою складовою стійкості та успіху будь-якої організації: від запобігання перебоїв у наданні послуг до захисту конфіденційної інформації та забезпечення довіри клієнтів.

У цій статті ми розглянемо найкращі практики кібербезпеки для центрів обробки даних. Перевірте їх!

Що таке кібербезпека для центрів обробки даних?

Кібербезпека для центрів обробки даних стосується набору методів, технологій і стратегій, розроблених для захисту систем центрів обробки даних, даних і операцій від кіберзагроз, несанкціонованого доступу, збоїв і зловмисних атак.

Центри обробки даних, будучи центрами, де зберігаються, обробляються та розповсюджуються важливі дані, є цінними цілями для кіберзлочинців. Тому забезпечення їх безпеки має важливе значення для підтримки цілісності, конфіденційності та доступності інформації та послуг, які вони підтримують.

У цьому сенсі кібербезпека центрів обробки даних базується на 3 стовпах:

– Конфіденційність: гарантувати, що лише авторизовані особи або системи мають доступ до даних, що зберігаються в центрі обробки даних.

– Цілісність: для захисту даних від несанкціонованих змін, гарантуючи, що вони залишаються точними та надійними.

– Доступність: щоб гарантувати, що системи та інформація будуть доступними в будь-який час, навіть у разі атак або збоїв.

Чому кібербезпека для центрів обробки даних важлива?

Кібербезпека в центрах обробки даних не обмежується бар'єрами проти зовнішніх атак. Це передбачає

– Захист від внутрішніх загроз, таких як невідповідний або випадковий доступ співробітників.

– Впровадження процесів постійного моніторингу та реагування на інциденти.

– Застосування передових технологій, таких як шифрування, сегментація мережі та багатофакторна автентифікація.

Інвестуючи в кібербезпеку центрів обробки даних, компанії захищають масиви конфіденційних даних, як-от конфіденційна інформація про клієнтів, інтелектуальну власність і критичні операційні дані в центрах обробки даних, що робить їх привабливими цілями.

Крім того, вони також відповідають нормативним вимогам, оскільки LGPD (Загальний закон про захист даних) у Бразилії та GDPR (Загальний регламент захисту даних) у Європі вимагають суворих заходів для захисту конфіденційних даних.

Нарешті, є фінансовий і репутаційний вплив, оскільки успішна атака може спричинити мільйонні збитки, перебої в роботі та непоправну шкоду іміджу компанії.

10 стратегій кібербезпеки для центрів обробки даних

Забезпечення безпеки центру обробки даних вимагає не тільки технологічних інструментів; необхідно прийняти стратегічний підхід, який інтегрує процеси, передові технології та обізнаність команди.

Отже, ось основні стратегії кібербезпеки, які допомагають захистити центри обробки даних від атак, мінімізувати вразливі місця та забезпечити дотримання правил захисту даних.

1. Суворий контроль доступу

Однією з перших ліній захисту безпеки центру обробки даних є контроль доступу. Необхідно суворо контролювати та обмежувати як фізичний, так і цифровий доступ:

– Фізичний доступ: запровадьте такі заходи, як відеоспостереження, біометричні замки, ідентифікаційні бейджи та багатофакторну автентифікацію для доступу до об’єктів центру обробки даних.

– Цифровий доступ: реалізуйте політику «найменших привілеїв», гарантуючи, що працівники мають лише ті дозволи, які їм необхідні для виконання своїх завдань. Використання систем керування доступом на основі ролей (RBAC) є хорошою практикою.

2. Сегментація мережі

Сегментація мережі є важливою стратегією для обмеження бокового переміщення зловмисників у центрі обробки даних. Це означає поділ мережі на різні сегменти, кожен зі своєю політикою безпеки.

– Розділіть мережі: ізолюйте критично важливі системи, такі як сервери баз даних, від менш безпечних мереж, таких як ті, що використовуються для адміністративних завдань.

– Внутрішній брандмауер: налаштуйте брандмауери між сегментами для моніторингу та контролю трафіку.

– Підмережі VLAN: використовуйте віртуальні локальні мережі (VLAN) для сегментації трафіку в центрі обробки даних.

3. Постійний моніторинг та аналіз журналів

Постійний моніторинг життєво важливий для виявлення загроз у реальному часі та швидкого зменшення ризиків:

– Рішення SIEM: запровадьте інструменти кореляції та керування подіями безпеки (SIEM) для збору та аналізу журналів подій, виявлення підозрілої поведінки.

– Проактивні сповіщення: налаштуйте сповіщення про незвичні дії, такі як невчасні спроби входу або неавторизований доступ.

– Зберігання журналів: переконайтеся, що журнали надійно зберігаються протягом відповідних періодів для майбутніх аудитів.

4. Регулярні оновлення та патчі

Підтримка систем і програмного забезпечення в актуальному стані є одним із найпростіших і найефективніших заходів проти відомих вразливостей:

– Керування виправленнями: створіть регулярний графік застосування виправлень і оновлень до операційних систем, програм і мікропрограм.

– Автоматизація: використовуйте інструменти керування виправленнями для автоматичного визначення та виправлення вразливостей.

– Перевірка сумісності: перед впровадженням оновлень проведіть тести в середовищах ізольованого програмного середовища, щоб переконатися, що вони не викликають проблем із системами.

5. Захист від DDoS атак

Атаки розподіленої відмови в обслуговуванні (DDoS) є типовою загрозою для центрів обробки даних. Захист інфраструктури від цих атак має важливе значення:

– Рішення для захисту від DDoS: розгортайте спеціалізовані інструменти для виявлення та пом’якшення атак DDoS до того, як вони спричинять збої.

– Мережі розподілу вмісту (CDN): використовуйте CDN для розподілу трафіку та зменшення навантаження на центр обробки даних.

– Масштабованість: переконайтеся, що інфраструктура має здатність

обробляти піки трафіку.

6. Шифрування даних

Захист даних у стані спокою та під час передачі є важливим для запобігання несанкціонованому доступу:

– Шифрування в стані спокою: використовуйте шифрування для захисту даних, що зберігаються на серверах і пристроях резервного копіювання.

– Шифрування під час передавання: переконайтеся, що всі дані, що передаються через мережу, зашифровані за допомогою безпечних протоколів, таких як TLS.

– Керування ключами: реалізуйте надійні рішення для керування ключами, щоб гарантувати захист ключів шифрування від несанкціонованого доступу.

7. Навчання та обізнаність співробітників

Співробітники часто є найслабшою ланкою в кібербезпеці. Інвестиції в навчання та підвищення обізнаності можуть значно зменшити ризик людської помилки:

– Симуляції фішингу: проводите регулярні тести, щоб виявити вразливі місця та навчити співробітників розпізнавати загрози.

– Політика щодо паролів: навчайте правильному створенню паролів і управлінню ними, зокрема використанню менеджерів паролів.

– Регулярні оновлення: Проводьте семінари та інформаційні сесії про останні тенденції в кібербезпеці.

8. Регулярне та безпечне резервне копіювання

Резервне копіювання є останньою лінією захисту від втрати даних і атак програм-вимагачів:

– Автоматичне резервне копіювання: впроваджуйте рішення, які створюють регулярні резервні копії критично важливих даних.

– Безпечне зберігання: переконайтеся, що резервні копії зберігаються в захищених місцях, бажано за межами сайту, щоб запобігти збитку, спричиненому

катастрофами.

– Тести відновлення. Виконуйте періодичні тести, щоб переконатися, що резервні копії можна ефективно відновити.

9. Тести на проникнення та аудит безпеки

Проведення регулярних тестів і аудитів допомагає виявити вразливості до того, як ними скористаються зловмисники:

– Тести на проникнення: найміть експертів для моделювання атак і виявлення слабких місць в інфраструктурі.

– Незалежні аудити: Проводьте сторонні аудити безпеки, щоб переконатися в дотриманні галузевих норм і стандартів.

– Плани дій: розробити плани виправлення вразливостей, виявлених під час тестування.

10. Плани реагування на інциденти

Щоб мінімізувати вплив кібератак, необхідно мати надійний план реагування на інциденти:

– Спеціальна команда: сформуйте групу реагування на інциденти (IRT) із чітко визначеними ролями та обов'язками.

– Посібники реагування: створіть детальні процедури для різних типів інцидентів, таких як витік даних або DDoS-атаки.

– Регулярні тести: Проводьте моделювання, щоб переконатися, що команда готова діяти швидко в разі надзвичайної ситуації.

3.2 Розробка структурної схеми

У майбутньому, оскільки атаки стають все більш поширеними та витонченими, командам із кібербезпеки потрібно боротися із загрозами, що стосуються фізичної інфраструктури, сторонніх постачальників і персоналу, використовуючи низку передових технологій кібербезпеки, жорсткі політики та процедури управління, а також включаючи кібернавчання співробітників.

Кібератаки, спрямовані на фізичну інфраструктуру

Необхідно розширити питання кібербезпеки, щоб включити фізичну інфраструктуру, яка включає, наприклад, захист від хакерів, які використовують слабкі місця в системах опалення, вентиляції та кондиціонування повітря (HVAC) або джерела безперебійного живлення (UPS). Зростаюча залежність від інтелектуальних підключених пристроїв у центрах обробки даних означає, що практично будь-яке рішення може бути захоплене, щоб спричинити збої та збої.

Хоча рідко, система HVAC може бути використана з катастрофічними наслідками. У галузі були приклади, коли зловмисники отримували доступ і маніпулювали температурою холодильної камери, що ризикувало пошкодити IT-обладнання, час безвідмовної роботи та надійність обслуговування. Розподілені атаки типу «відмова в обслуговуванні» (DDoS) навіть призвели до повної втрати систем опалення через те, що кіберзлочинці переповнювали ресурси мережі HVAC, перевантажуючи їх до виходу з ладу.

HVACKer – за кодовою назвою його творців – це спеціально створене шкідливе програмне забезпечення, здатне взаємодіяти з тепловими датчиками комп'ютера, щоб зчитувати коливання температури та перетворювати ці коливання у двійковий код. Сценарій показує, як системи опалення, вентиляції, вентиляції та кондиціонування повітря можуть бути використані як засіб для з'єднання повітряно-захищених мереж із зовнішнім світом, використовуючи коливання температури для прихованого надсилання інструкцій шкідливим програмам, які вже присутні в ізольованих системах.

Тепер хакери також атакують підключені до Інтернету системи ДБЖ, часто через незмінні імена користувачів і паролі за замовчуванням.

Зараз хакери також атакують підключені до Інтернету системи ДБЖ, часто через незмінні імена користувачів і паролі за замовчуванням. ДБЖ були розроблені, щоб компенсувати збій основного джерела живлення в критичних інфраструктурах, але підключення до Інтернету також робить їх мішенню для хакерів. Дослідники CRIL показали, як зловмисники отримали доступ до веб-

консолей пристроїв ДБЖ для контролю ключових налаштувань, таких як вимикання/увімкнення, перезавантаження та навіть видалення журналів і трасування з консолі. Щоб зберегти відчуття впевненості, команди повинні ефективно керувати середовищем UPS, щоб запобігти компромісам.

Сторонні постачальники, які діють як чорні двері для кіберзлочинців

Зловмисники також показали, що вони можуть скористатися слабкими місцями в інфраструктурі центру обробки даних, яка включає системи, якими керують треті сторони. Націлюючись на сторонніх постачальників, які мають привілейований доступ до мережі, хакери можуть використовувати слабкі місця в безпеці, щоб отримати доступ до конфіденційних частин інфраструктури центру обробки даних.

З цієї причини важливо, щоб команди центрів обробки даних тісно співпрацювали зі сторонніми постачальниками, щоб гарантувати, що вони роблять усе можливе для захисту своїх систем і мереж, щоб уникнути того, щоб партнери мимоволі діяли як чорний хід для кіберзлочинців.

У відомій кібератаці 2013 року хакерам вдалося зламати мережу роздрібного продавця, скориставшись обліковими даними, вкраденими у стороннього постачальника з віддаленим доступом до їхніх холодильних установок DC. Атака скомпрометувала систему роздрібних продажів через інфраструктуру HVAC, що призвело до викрадення мільйонів записів кредитних і дебетових карток клієнтів.

Поставте безпеку в основу роботи центру обробки даних

Кібербезпека – це індустрія, яка постійно розвивається, з постійною появою нових викликів і загроз. Щоб зменшити ризики в центрах обробки даних, захистити дані та цінні активи та зберегти довіру клієнтів, команди можуть виконати кілька ключових кроків:

– Зміцніть периметр мережі. Визначивши периметр мережі, команди центру обробки даних можуть створити віртуальний оплот, який буде першою лінією захисту від несанкціонованого доступу. Безпечні межі є фундаментальними для безпеки мережі, а інтеграція передових методів керування

трафіком разом із системами запобігання вторгненням підвищує ефективність стратегій захисту периметра.

– Забезпечте повну видимість активів у вашому середовищі – використовуйте інструменти керування мережею та виявлення, налаштовані на пошук підключених до мережі пристроїв. Крім того, намагайтеся підтримувати базу даних керування активами та конфігурацією, яка може підтримувати ідентифікацію та моніторинг потенційних уразливостей.

– Постійно оновлюйте програмне забезпечення – запровадьте сувору стратегію керування виправленнями та плануйте оперативне усунення вразливостей, а також оновлюйте програмне забезпечення та мікропрограму. Приділіть особливу увагу застарілим системам і забезпечте регулярну публікацію звітів про відповідність вимогам і ризики в рамках операційного управління.

– Регулярно скануйте на наявність незахищених або відкритих портів на вашому мережевому рівні – застосуйте надійне керування паролями та суворі політики користувача, які підтримуються періодичним тестуванням на проникнення для проактивного виявлення вразливостей. Крім того, підтримуйте постійний цикл перевірки конфігурації та політики, щоб забезпечити відповідність найкращим практикам або нормативним вимогам.

Стратегія з людьми в центрі

Головним у будь-якій ефективній стратегії кібербезпеки центру обробки даних є забезпечення того, щоб кожен співробітник, незалежно від його ролі, усвідомлював свою роль у підтримці надійного, безпечного кіберсередовища, оскільки це обмежує шанси кіберзлочинців успішно використовувати вразливі місця людини за допомогою таких векторів, як фішингові атаки. Розвиток надійної робочої сили, яка обізнана з кібербезпекою, вимагає постійного та регулярного навчання працівників. Це означає, що співробітники повинні проходити регулярні курси підвищення кваліфікації та отримувати найновіші відомості про нові загрози, щоб гарантувати, що дотримання належної кіберпрактики стане нормальною частиною їхніх щоденних процесів.

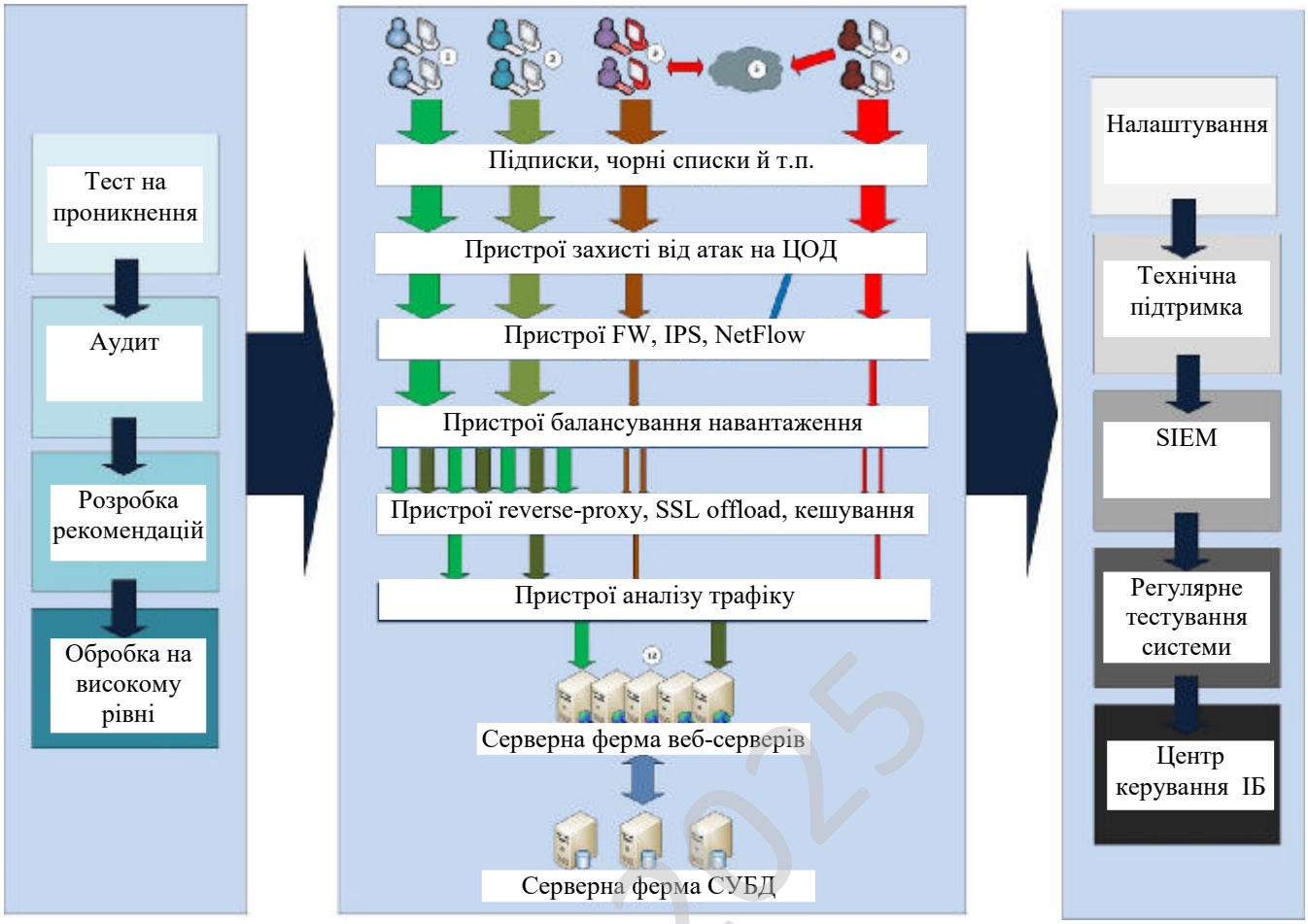

Рисунок 3.1 – Структурна схема системи

Національна критична інфраструктура

Оскільки українські центри обробки даних визнано частиною критично важливої інфраструктури нації разом із системами, об’єктами та мережами, які вважаються важливими для функціонування економіки та суспільства країни, групи кібербезпеки, ймовірно, стикаються зі зростаючою кількістю загроз і атак. Отже, необхідно ввести технічні засоби захисту, щоб захистити центри обробки даних від несанкціонованого доступу, керувати мережами та підтримувати програмне забезпечення в актуальному стані. У центрі будь-якої стратегії також має бути команда, що обізнана в кібернетиці, і лідери з кібербезпеки, які залишаються на крок попереду нових і нових методів і тактик. Поставивши безпеку в основу операцій, британські центри обробки даних можуть продовжувати надавати важливі послуги, розвивати цифрову економіку та зберігати особисту інформацію в безпеці.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Так, як функціональна схема є більш подобним описом функціональних можливостей структурної схеми, то вона буде представляти собою, більш детальний варіант структурної схеми.

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

– Блок моніторингу мережі ЦОД.

– Блок аналізу мережної статистики ЦОД.

– Блок визначення топології мережі ЦОД.

– Блок виявлення аномального поводження трафіку ЦОД.

– Блок визначення виду атаки на ЦОД.

– Блок зберігання результатів.

Блок визначення топології мережі ЦОД

Блок визначення топології мережі ЦОД:

– Блок використання відомостей із загальної системи моніторингу мережі ЦОД, а не опитування пристрою додатково.

– Блок складання списку пристроїв у мережі ЦОД, автоматично, ґрунтуючись на дані системи моніторингу.

– Блок побудови топології мережі ЦОД, за станом на задану дату й відстеження змін у топології протягом часу.

– Блок автоматичного визначення рівнів ієрархії пристроїв у мережі ЦОД, з виділенням периферійних, проміжних і центральних вузлів;

– Блок побудови топології мережі ЦОД, незалежно від використовуваної системи моніторингу й програмно-апаратних платформ;

– Блок комбінувати показників, на основі яких визначаються зв'язки між пристроями, і при їхньому обчисленні виконувати перевірку на значимість із використанням статистичних критеріїв.

Блок виявлення аномального поводження трафіку ЦОД

Блок виявлення аномального поводження трафіку ЦОД:

– Блок визначення профілю поводження нормального трафіку ЦОД.

– Блок заміни направлення трафіку ЦОД.

– Блок усунення аномального трафіку ЦОД.

При первісному розгортанні рішення по DDoS адміністратор створює профіль поводження нормального трафіку ЦОД. Цей процес іменується навчанням. Компанія використовує додатки звичайним образом протягом 24 годин протягом одного тижня, і трафік додатка проходить через Детектор аномалій трафіку ЦОД. У період навчання Детектор аномалій трафіку ЦОД збирає базову інформацію для розуміння нормальної роботи мережі ЦОД, куди входять:

– Інтенсивність пакетів для кожного типу пакетів, обмірювана як кількість пакетів у секунду (pps).

– Співвідношення пакетів, наприклад, співвідношення пакетів SYN і пакетів FIN.

– Кількість одночасних TCP-з'єднань, відкритих одним джерелом.

Базова інформація збирається по кожній цільовій адресі хост-пК, цільовій підмережі ЦОД, вихідній адресі хост-пК і вихідній підмережі ЦОД.

Після закінчення періоду навчання Детектор аномалій трафіку ЦОД переводиться в режим моніторингу, а Блок усунення аномального трафіку ЦОД – у резервний режим готовності. Доти, поки немає атаки на ЦОД, що активно розвивається, вхідний трафік з мережі ЦОД Інтернет проходить через комутатор без якого-небудь втручання з боку Блоку усунення аномального трафіку ЦОД. Копія вхідного трафіку ЦОД посилає для аналізу на Детектор аномалій трафіку ЦОД через зовнішній аналізатор протоколів (SPAN) або віртуальні списки ACL. Якщо Детектор аномалій трафіку ЦОД виявляє аномальне в порівнянні з базовою інформацією поводження трафіку ЦОД, починається процес усунення:

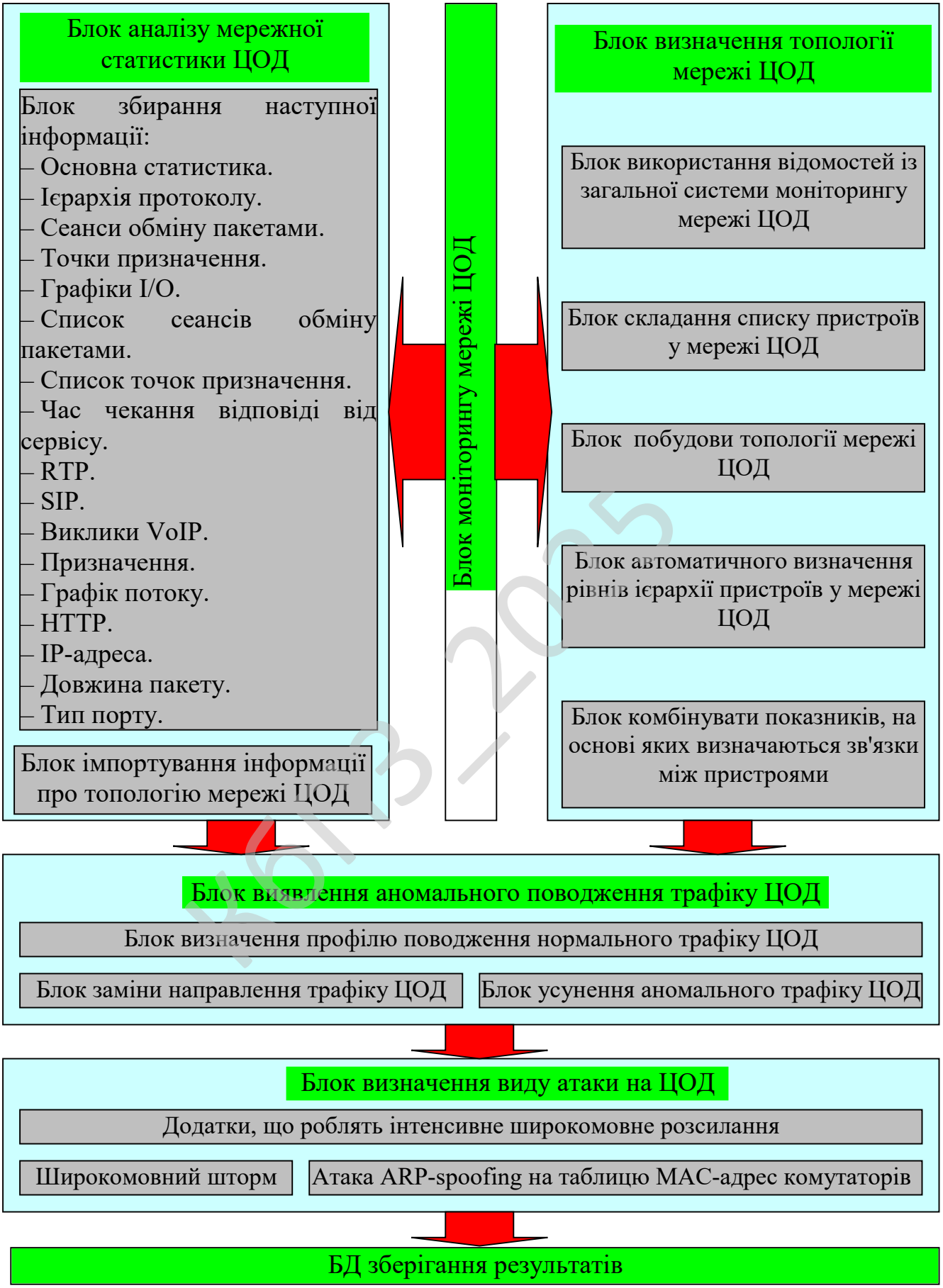

Рисунок 3.2 – Функціональна схема системи

– Детектор аномалій трафіку ЦОД направляє в Блок усунення аномального трафіку ЦОД команду почати процес зміни напрямку.

– Блок усунення аномального трафіку ЦОД відхиляє (“захоплює”) трафік, адресований на атакуєму IP-адресу, переадресуючи його на самого себе.

– Блок усунення аномального трафіку ЦОД піддає трафік багатоступінчастому аналізу й застосовує контрзаходи для відділення благонадійних джерел від джерел атаки на ЦОД. Цей процес іменується очищенням або вичищенням.

– Блок усунення аномального трафіку ЦОД скидає трафік атаки на ЦОД й пересилає благонадійний трафік назад на нормальний маршрут проходження трафіку ЦОД до мети. Цей процес іменується ін'єкцією.

Детектор аномалій трафіку ЦОД

Детектор аномалій трафіку ЦОД – це пасивний пристрій моніторингу, що постійно виявляє ознаки, що вказують на присутність атаки на ЦОД DDoS, спрямованої проти захищеного місця призначення, також іменованого зоною. Це може бути сервер, інтерфейс міжмережного екрана або інтерфейс маршрутизатора. Детектор аномалій трафіку ЦОД аналізує копії всього вхідного трафіку ЦОД, адресуємого в захищені зони, через SPAN або відгалуження пасивної мережі ЦОД. Цей аналіз включає зіставлення поточного поводження трафіку ЦОД з базовими граничними параметрами, які також іменуються зональною політикою, для виявлення аномального поводження трафіку ЦОД. Якщо аномальне поводження виявлене й виглядає як можлива атака, Детектор аномалій трафіку ЦОД через позаполосну управлінську мережу Ethernet посилає в Блок усунення аномального трафіку ЦОД сигнал про початок аналізу й усунення атаки на ЦОД.

Блок усунення аномального трафіку ЦОД

Блок усунення аномального трафіку ЦОД – це автономний пристрій аналізу й фільтрації трафіку ЦОД. Починаючи прийом трафіку ЦОД, адресованого в конкретну зону, що, очевидно, піддається атаці, Блок усунення

аномального трафіку ЦОД проводить точний аналіз цього трафіку ЦОД. Якщо результати аналізу підтверджують, що трафік злочинний, Блок усунення аномального трафіку ЦОД застосовує контрзаходи, наприклад, механізми анти-спуфінга й фільтрацію різного рівня (таблиця 3.1). Кінцевий результат полягає в тому, що трафік зі злочинних джерел скидається, а трафік із благонадійних джерел пересилається в передбачений пункт призначення.

Атаки на ЦОД DDoS – Виявлення й усунення

У таблиці 3.1 перераховані типи атак DDoS на ЦОД, які може виявляти й усувати Блок усунення аномального трафіку ЦОД.

Можливі варіанти зміни напрямку трафіку ЦОД

Фахівці з IT можуть використовувати описані нижче варіанти зміни напрямку трафіку ЦОД з його пересиланням з мережі ЦОД, розташованого вище лежачого оператора зв'язку, на Блок усунення аномального трафіку ЦОД. Цей процес також іменується “захватом” трафіку ЦОД:

– Повідомлення прикордонного шлюзового протоколу (Border Gateway Protocol, BGP) із Блок усунення аномального трафіку ЦОД на маршрутизатори, розташовані у вище лежачого оператора зв'язку, з інформацією про те, що трафік, адресований на захищену адресу призначення, буде переспрямований на Блок усунення аномального трафіку ЦОД.

– Використання зовнішніх механізмів зміни напрямку трафіку ЦОД, наприклад, маршрутизаторів віддаленого відновлення BGP.

– Повідомлення про ін'єкцію очищеного трафіку ЦОД на маршруті (Route Health Injection, RHI) від Блок усунення аномального трафіку ЦОД для процесу маршрутизації в Catalyst серії 6500 або в систему нагляду серії 7600. Ці повідомлення поміщають статичний маршрут у глобальну таблицю маршрутизації, у якій модуль Блок усунення аномального трафіку ЦОД позначений як наступний вузол.

Таблиця 3.1 – Категорії й особливі типи атак DDoS на ЦОД

Категорія атаки на ЦОД	**Особливі типи атак на ЦОД**
Атаки на ЦОД із заповненням смуги пропущення	Лавинні атаки на ЦОД зі спуфінгом або без спуфінга: – Прапор TCP (SYN, SYN-ACK, ACK, FIN). – Протокол керування повідомленнями в Інтернет (ICMP). – Протокол користувальницьких датаграмм (UDP). Приклади: лавинна атака SYN, smurf, LAND і UDP – лавинні атаки на ЦОД.
	Атаки на ЦОД зомбі-комп'ютерів/мереж зомбі-комп'ютерів, у яких кожний вихідний зомбі-пк або мережа відкриває множинні TCP-з'єднання й, у деяких випадках, видає багаторазові запити HTTP.
	Атаки на ЦОД DNS, наприклад, лавинна атака із запитами DNS.
Атаки на ЦОД з дефіцитом ресурсів	Атаки на ЦОД пакетного розміру, характерна риса яких – фрагментированні або великі пакети. Приклади: teardrop і ping-of-death.
	Атаки на ЦОД зомбі-комп'ютерів/мереж зомбі-комп'ютерів з низькою інтенсивністю схожі на атаки на ЦОД із заповненням смуги пропущення за тим виключенням, що кожне джерело атаки на ЦОД посилає множинні запити з невеликим обсягом в одиницю часу.
	Атаки на ЦОД DNS з рекурсивним переглядом DNS.

Можливі варіанти ін'єкції трафіку ЦОД

Ін'єкція трафіку ЦОД – це процес, застосовуваний у Блок усунення аномального трафіку ЦОД для пересилання очищеного благонадійного трафіку ЦОД в точку призначення, що піддається атаці. Рішення підтримує різні варіанти

ін'єкції трафіку ЦОД. У варіанті 2-ого рівня топології, очищений трафік пересилається із Блок усунення аномального трафіку ЦОД на статично-конфігуруєму наступну адресу заходу. Ця адреса перебуває на маршрутизаторі, розташованому нижче й з'єднаним з тої ж VLAN або підмережею, що й інтерфейс/VLAN ін'єкції трафіку ЦОД. Ін'єкцію трафіку ЦОД на 2-му рівні найпростіше конфігурувати, оскільки тут не потрібно вносити які-небудь істотні зміни в конфігурацію маршрутизатора, розташованого нижче.

Варіанти ін'єкції трафіку ЦОД 3-го рівня:

– Маршрутизація й пересилання по VPN (VPN Routing and Forwarding, VRF).

– Маршрутизація на основі політики (Policy-Based Routing, PBR).

– Транкинг VLAN (VLAN Trunking).

– Інкапсуляція по загальній маршрутизації (GRE) або інкапсуляція IP у тунелі IP (IPIP).

Блок визначення виду атаки на ЦОД

Блок визначення виду атаки на ЦОД:

– Атака ARP-spoofing на таблицю mac-адрес комутаторів.

– Широкомовний шторм.

– Додатки, що роблять інтенсивне широкомовне розсилання, наприклад: широкомовні чати й мережні ігри.

ARP-spoofing

ARP-spoofing – техніка атаки на ЦОД в Ethernet мережах, що дозволяє перехоплювати трафік між хостами. Заснована на використанні протоколу ARP.

При використанні в розподіленій обчислювальній системи (РВМ) алгоритмів віддаленого пошуку існує можливість здійснення в такій мережі ЦОД типової віддаленої атаки на ЦОД «помилковий об'єкт РВМ». Аналіз безпеки протоколу ARP показує, що, перехопивши на атакуючому хості усередині даного сегмента мережі ЦОД широкомовний ARP-запит, можна послати помилкову ARP-відповідь, у якій оголосити себе шуканим хостом (наприклад,

маршрутизатором), і надалі активно контролювати мережний трафік дезінформованного хосту, впливаючи на нього за схемою «помилковий об'єкт РВМ».

Протокол ARP призначений для перетворення IP-адрес в MAC-адреси. Найчастіше мова йде перетворенні в адреси Ethernet, але ARP використовується й у мережах інших технологій: Token Ring, FDDI і інших.

Алгоритм роботи ARP

Протокол може використовуватися в наступних випадках:

1. Хост A хоче передати IP-пакет вузлу B, що перебуває з ним в одній мережі ЦОД.

2. Хост A хоче передати IP-пакет вузлу B, що перебуває з ним у різних мережах, і користується для цього послугами маршрутизатора R.

У кожному із цих випадку вузлом A буде використовуватися протокол ARP, тільки в першому випадку для визначення MAC-адреси вузла B, а в другому – для визначення MAC-адреси маршрутизатора R. В останньому випадку пакет буде переданий маршрутизатору для подальшої ретрансляції.

Далі для простоти розглядається перший випадок, коли інформацією обмінюються вузли, що перебувають безпосередньо в одній мережі ЦОД. (Випадок коли пакет адресований вузлу, який знаходиться за маршрутизатором, відрізняється тільки тим, що в пакетах переданих після того як ARP-перетворення завершене, використовується IP-адреса одержувача, але MAC-адреса маршрутизатора, а не одержувача.)

Проблеми ARP

Протокол ARP є абсолютно незахищеним. Він не має ніякого способу перевірки дійсності пакетів: як запитів, так і відповідей. Ситуація стає ще більш складною, коли може використовуватися мимовільний ARP (gratuitous ARP).

Мимовільний ARP – таке поводження ARP, коли ARP-відповідь надсилається, коли в цьому (з погляду одержувача) немає особою необхідності. Мимовільна ARP-відповідь це пакет-відповідь ARP, присланий без запиту. Він застосовується для визначення конфліктів IP-адрес у мережі ЦОД: як тільки

станція одержує адресу по DHCP або адреса привласнюється вручну, розсилається ARP-відповідь gratuitous ARP.

Мимовільний ARP може бути корисний у наступних випадках:

– Відновлення ARP-таблиць, зокрема, у кластерних системах.

– Інформування комутаторів.

– Повідомлення про включення мережного інтерфейсу.

Незважаючи на ефективність мимовільного ARP, він є особливо небезпечним, оскільки з його допомогою можна запевнити віддалений вузол у тому, що MAC-адреса якої-небудь системи, що перебуває з нею в одній мережі ЦОД, змінилася й указати, яка адреса використовується тепер.

До виконання ARP-spoofing'a в ARP-таблиці вузлів A і B існують записи з IP- і MAC-адресами один одного. Обмін інформацією виробляється безпосередньо між вузлами A і B.

У ході виконання ARP-spoofing'a комп'ютер **C**, що виконує атаку, відправляє ARP-відповіді (без одержання запитів):

– вузлу A: з IP-адресою вузла B і MAC-адресою вузла C;

– вузлу B: з IP-адресою вузла A і MAC-адресою вузла C.

У силу того що комп'ютери підтримують мимовільний ARP (gratuitous ARP), вони модифікують власні ARP-таблиці й поміщають туди записи, де замість справжніх MAC-адрес комп'ютерів A і B коштує MAC-адреса комп'ютера C.

Після того як атака виконана, коли комп'ютер A хоче передати пакет комп'ютеру B, він знаходить в ARP-таблиці запис (він відповідає комп'ютеру C) і визначає з її MAC-адресу одержувача. Відправлений по цьому MAC-адресу пакет приходить комп'ютеру C замість одержувача. Комп'ютер C потім ретранслює пакет тому, кому він дійсно адресований – тобто комп'ютеру B.

Широкомовний шторм

Широкомовний шторм – лавина (сплеск) широкомовних акетів (на другому рівні моделі OSI – кадрів). Розмноження некоректно сформованих широкомовних повідомлень у кожному вузлі приводить до експонентного росту

їхнього числа й паралізує роботу мережі ЦОД. Звичайно такі пакети використовуються мережними сервісами для оповіщення станцій про свою присутність. Вважається нормальним, якщо широкомовні пакети становлять не більше 10% від загального числа пакетів у мережі ЦОД.

Також досить часто до шторму приводять кільця в мережі ЦОД при некоректному настроюванні протоколу Spanning Tree, оскільки в заголовку пакетів Ethernet немає інформації про час життя кадру, як, наприклад, у пакетів IP. Крім цього широкомовний шторм застосовується (навмисно) зломщиками.

Відповідно до галузевого стандарту де-факто число широкомовних і багатоадресних кадрів у мережі ЦОД не повинне перевищувати 8-10% від загального числа кадрів.

Широкомовний кадр – це кадр, адресований всім станціям у домені мережі ЦОД. Багатоадресний кадр – це кадр, адресований групі станцій у домені мережі ЦОД. Оскільки широкомовний кадр адресований всім станціям, то, одержавши його, станції повинні перервати свою роботу й обробити такий кадр. Це сповільнює роботу всієї мережі ЦОД.

Якщо відношення числа широкомовних кадрів до загального числа кадрів більше 10%, то такий ефект називається "широкомовним штормом".

Широкомовний шторм може бути наслідком дефектів устаткування або неправильного настроювання параметрів активного встаткування. Найчастіше це явище спостерігається в розподілених мережах NetWare, побудованих на основі комутаторів, або коли дані між сегментами або доменами мережі ЦОД можуть передаватися більш ніж по одному потенційному шляху. Якщо один з комутаторів такої мережі ЦОД не підтримує протокол Spanning Tree (звичайно IEEE 802.1d) або останній неправильно настроєний або збоїть, то в мережі ЦОД починається некерована циркуляція широкомовних кадрів.

Виявлення "широкомовного шторму" є не настільки тривіальним завданням, як це може здатися на перший погляд. Для його виявлення недостатньо взяти загальне число широкомовних кадрів і поділити його на загальне число кадрів, що пройшли по мережі ЦОД.

Для цього ви повинні визначити: яку частку становлять широкомовні кадри в кожний інтервал часу (наприклад, за одну мінуту) і яка при цьому утилізація каналу зв'язку. Якщо, наприклад, за одну мінуту по мережі ЦОД пройшло 4 кадри, а 2 з них були широкомовними, то це ще не виходить, що ви спостерігаєте "широкомовний шторм".

Захист від широкомовних штормів (broadcast storm)

Одна з характерних несправностей мережного програмного забезпечення – мимовільна генерація з високою інтенсивністю широкомовних пакетів. Широкомовним штормом вважається ситуація, у якій відсоток широкомовних пакетів перевищує 20% від загальної кількості пакетів у мережі ЦОД. Звичайний комутатор або міст сліпо передає такі пакети на всі свої порти, як того вимагає його логіка роботи, засмічуючи, таким чином, мережу. Боротьба із широкомовним штормом у мережі ЦОД, з'єднаної комутаторами, жадає від адміністратора відключення портів, що генерують широкомовні пакети. Маршрутизатор не поширює такі ушкоджені пакети, оскільки в коло його завдань не входить копіювання широкомовних пакетів в усі поєднувані їм мережі ЦОД. Тому маршрутизатор є прекрасним засобом боротьби із широкомовним штормом, щоправда, якщо мережа розділена на достатню кількість підмереж.

Блок аналізу мережної статистики ЦОД

Блок збирання наступної інформації:

– Основна статистика (Summary).

– Ієрархія протоколу (Protocol Hierachy).

– Сеанси обміну пакетами (Conversations).

– Точки призначення (Endpoints).

– Графіки I/O (IO Graphs).

– Список сеансів обміну пакетами (Conversation List).

– Список точок призначення (Endpoint List).

– Час чекання відповіді від сервісу (Service Response Time).

– RTP.

– SIP.

– Виклики VoIP (VoIP Calls).

– Призначення (Destination).

– Графік потоку (Flow Graph).

– HTTP.

– IP-адреса (IP address).

– Довжина пакету (Packet Length).

– Тип порту (Port Type).

Розпишемо їх більш детально.

1. Основна статистика. Доступні такі елементи основної статистики, як:

– Властивості захоплених файлів.

– Час захвату.

– Інформація про фільтр захвату.

– Інформація про фільтр відображення.

2. Ієрархія протоколу. Статистика ієрархії протоколу допомагає аналізувати пакети, розбиваючи відображені дані, які належать чинному рівню OSI.

3. Сеанси обміну пакетами. Якщо ви використовуєте протокол TCP/IP або програму, яка працює із цим протоколом, ви маєте побачити чотири активних вкладок для обміну пакетами за допомогою Ethernet, IP, TCP та UDP. «Діалог» між комп'ютерами відображає трафік між двома активними хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами. Номер, зазначений на вкладці після назви протоколу, означає кількість «діалогів» між хостами, наприклад, «Ethernet:6».

4. Точки призначення. Точки призначення забезпечують статистику даними про відправку та прийом пакетів. Номер, зазначений на вкладці після назви протоколу, вказує на кількість точок призначення. Наприклад, «Ethernet:6».

5. Графіки I/O. Основний графік може бути отриманий за допомогою команди «IO graphs» (Графіки I/O). Ще декілька графіків можуть бути додані у тому ж вікні на основі фільтрів відображення.

6. Час чекання відповіді від сервісу. 13 протоколів доступні для глибокого аналізу.

7. RTP. RTP (Real-time Transport Protocol, протокол передачі у реальному часі, RFC 3550) – це протокол для передачі звука та відео через IP-мережу. Він працює у початку протоколу дейтаграм користувача (User Datagram Protocol, UDP). Він часто використовується у сукупності з протоколами SIP або H.233, забезпечуючи виконання сигнальних завдань.

8. SIP. SIP (Session Initiation Protocol, протокол встановлення сесії, RFC 3261) – це сигнальний протокол, який оголошує відео– або VoIP-сесії. Він працює разом із протоколом RTP, який використовується для передачі мультимедійних даних.

9. Виклики VoIP.

VoIP (Voice over IP, голосовий зв'язок за допомогою Інтернету) взагалі використовує два типи протоколів:

– сигнальні протоколи, такі, як SIP або H.323

– переносні протоколи, наприклад, RTP

10 Призначення. Відображення усіх IP-адрес призначення мережевих пакетів.

11. Графік потоків. Графіки потоків забезпечує послідовний аналіз TCP-з'єднань. Перші три строки містять оголошення TCP– з'єднання з послідовностями «SYN», «SYN ACK» та «ACK».

12 HTTP. HTTP (Hypertext Transfer Protocol, протокол передачі гіпертексту) – це протокол типу «клієнт-сервер», який використовується для передачі HTML-файлів. HTTP-клієнт (у більшості випадків це web-браузер) відсилає HTTP-запит до web-серверу із полем «URL», який допомагає знайти потрібний файл. Web-сервер відповідає HTTP-пакетом та забезпечує клієнт необхідною web-сторінкою.

Меню «HTTP» містить три підменю:

– «Load Distribution» (Розподіл пакетів).

– «Packet Counter» (Лічильник пакетів).

– «Requests» (Запити).

14 IP-адреса. Відображення IP-адреси джерела або призначення мережевих пакетів.

15. Довжина пакету.

16. Тип порту. Відображення статистики портів TCP або UDP.

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.

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних - прямокутником, блок прийняття рішень - ромбом, еліпсом - початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування - початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Опис системи

Система забезпечення безпеки центру обробки даних включає апаратні та програмні компоненти, які контролюють доступ, моніторять трафік та виявляють

аномалії в мережі. Основою програмної частини є Python. Реалізація містить модулі для аутентифікації користувачів, аналізу логів, моніторингу мережевої активності та автоматичного реагування на інциденти.

Система має клієнт-серверну архітектуру. Серверна частина працює на базі Flask та взаємодіє з базою даних PostgreSQL для збереження логів та інформації про користувачів. Використовується бібліотека SQLAlchemy для роботи з базою даних. Клієнтська частина містить скрипти для збору даних та взаємодії з сервером через API.

Основні модулі системи:

– Модуль аутентифікації користувачів забезпечує вхід через двофакторну авторизацію. Використовується бібліотека PyOTP для генерації одноразових паролів. База даних містить хешовані паролі, що зберігаються з використанням bcrypt.

– Модуль моніторингу мережі аналізує трафік та виявляє підозрілі активності. Використовується Scapy для аналізу пакетів. Записи про мережеві з'єднання надходять до сервера, де обробляються через модуль машинного навчання, який базується на бібліотеці scikit-learn.

– Модуль аналізу логів здійснює пошук аномалій у журналах подій. Використовується бібліотека pandas для обробки та аналізу великих обсягів даних. Виявлення підозрілих входів до системи відбувається шляхом аналізу IP-адрес та часових відміток.

– Модуль автоматичного реагування активує заходи захисту при виявленні загрози. Наприклад, якщо модуль моніторингу мережі виявляє атаку, скрипт на Python блокує підозрілі IP-адреси через iptables або Windows Firewall.

Розрахунки ефективності

Системи базуються на аналізі часу реагування, рівня хибнопозитивних спрацьовувань та навантаження на сервер.

Наприклад, середній час обробки мережевого пакету становить 5 мс, а рівень хибних спрацьовувань не перевищує 2.5% при використанні моделі на

основі випадкового лісу. Навантаження на сервер при одночасному моніторингу 1000 підключень не перевищує 50% використання ресурсів процесора.

Ця система забезпечує ефективний захист ЦОД від несанкціонованого доступу, атак та витоку даних. Використання Python дозволяє швидко адаптувати рішення під змінні вимоги та інтегрувати додаткові механізми безпеки.

Частина вихідного коду:

```
import hashlib
import os
import pyotp
import bcrypt
import scapy.all as scapy
import pandas as pd
import psycopg2
from flask import Flask, request, jsonify
from scikit_learn.ensemble import RandomForestClassifier
from datetime import datetime

# Ініціалізація Flask-додатку
app = Flask(__name__)

# Налаштування з'єднання з базою даних
conn = psycopg2.connect(
    dbname="security_db",
    user="admin",
    password="password",
    host="localhost"
)
cursor = conn.cursor()

# Функція для хешування пароля
def hash_password(password):
return bcrypt.hashpw(password.encode(), bcrypt.gensalt())

# Функція для перевірки пароля
def verify_password(password, hashed_password):
return bcrypt.checkpw(password.encode(), hashed_password)

# Функція для генерації OTP
def generate_otp(secret):
```

```
        otp = pyotp.TOTP(secret)
        return otp.now()

        # Обробник для входу користувача
        @app.route('/login', methods=['POST'])
        def login():
        data = request.get_json()
        username = data['username']
        password = data['password']
        otp_code = data['otp']

        cursor.execute("SELECT password_hash, otp_secret FROM users WHERE
username=%s", (username,))
        user = cursor.fetchone()

        if user and verify_password(password, user[0]) and
           generate_otp(user[1]) == otp_code:
        return jsonify({"status": "success", "message": "Login successful"})
        else:
        return jsonify({"status": "error", "message": "Invalid credentials"}), 401

        # Функція для аналізу мережевого трафіку
        def monitor_traffic():
        def packet_callback(packet):
        if packet.haslayer(scapy.IP):
        src_ip = packet[scapy.IP].src
        dst_ip = packet[scapy.IP].dst
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        cursor.execute("INSERT INTO network_logs (src_ip, dst_ip, timestamp) VALUES
(%s, %s, %s)",
        (src_ip, dst_ip, timestamp))
        conn.commit()

        scapy.sniff(prn=packet_callback, store=False)

        # Функція для аналізу логів
        def analyze_logs():
        cursor.execute("SELECT * FROM network_logs")
        logs = cursor.fetchall()
        df = pd.DataFrame(logs, columns=['id', 'src_ip', 'dst_ip', 'timestamp'])
```

```
df['timestamp'] = pd.to_datetime(df['timestamp'])
df['hour'] = df['timestamp'].dt.hour

model = RandomForestClassifier()
features = df[['hour']]
labels = df['src_ip'].apply(lambda x: 1 if x.startswith("192.168") else 0)

model.fit(features, labels)

predictions = model.predict(features)
df['anomaly'] = predictions

df.to_csv("log_analysis_results.csv", index=False)

# Функція для автоматичного реагування на загрози
def block_ip(ip):
os.system(f"iptables -A INPUT -s {ip} -j DROP")

# Головна функція запуску всіх модулів
if __name__ == "__main__":
monitor_traffic()
analyze_logs()
app.run(host="0.0.0.0", port=5000)
```

Код забезпечує:

1. Аутентифікацію користувачів через хешування паролів, двофакторну авторизацію з OTP.

2. Моніторинг мережевого трафіку із записом підозрілих активностей у базу даних.

3. Аналіз логів із виявленням аномалій за допомогою машинного навчання.

4. Автоматичне блокування загроз через iptables.

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 системи забезпечення безпеки ЦОД.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

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 підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

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.

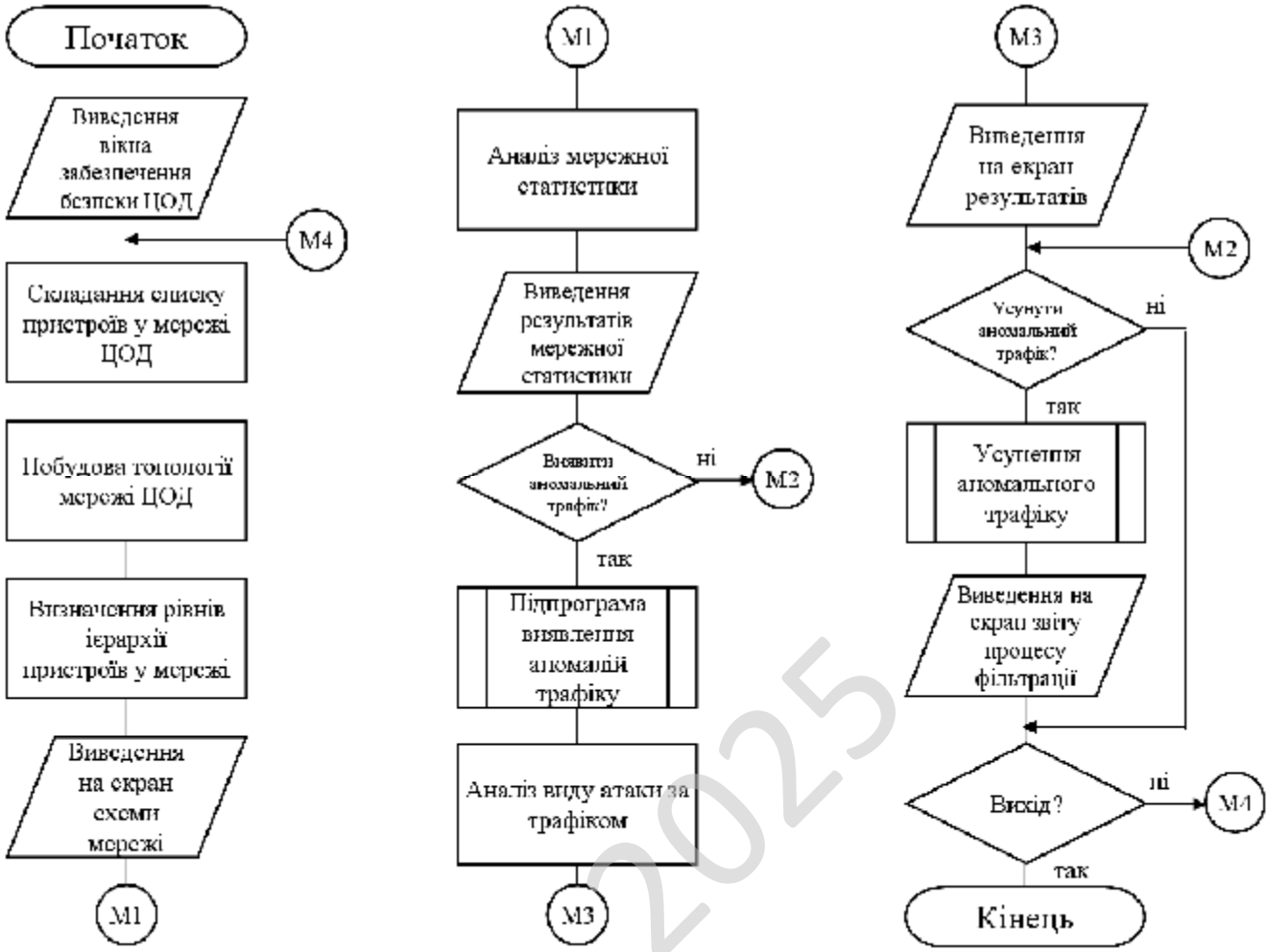

Рисунок 4.1 – Блок-схема основної програми

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.

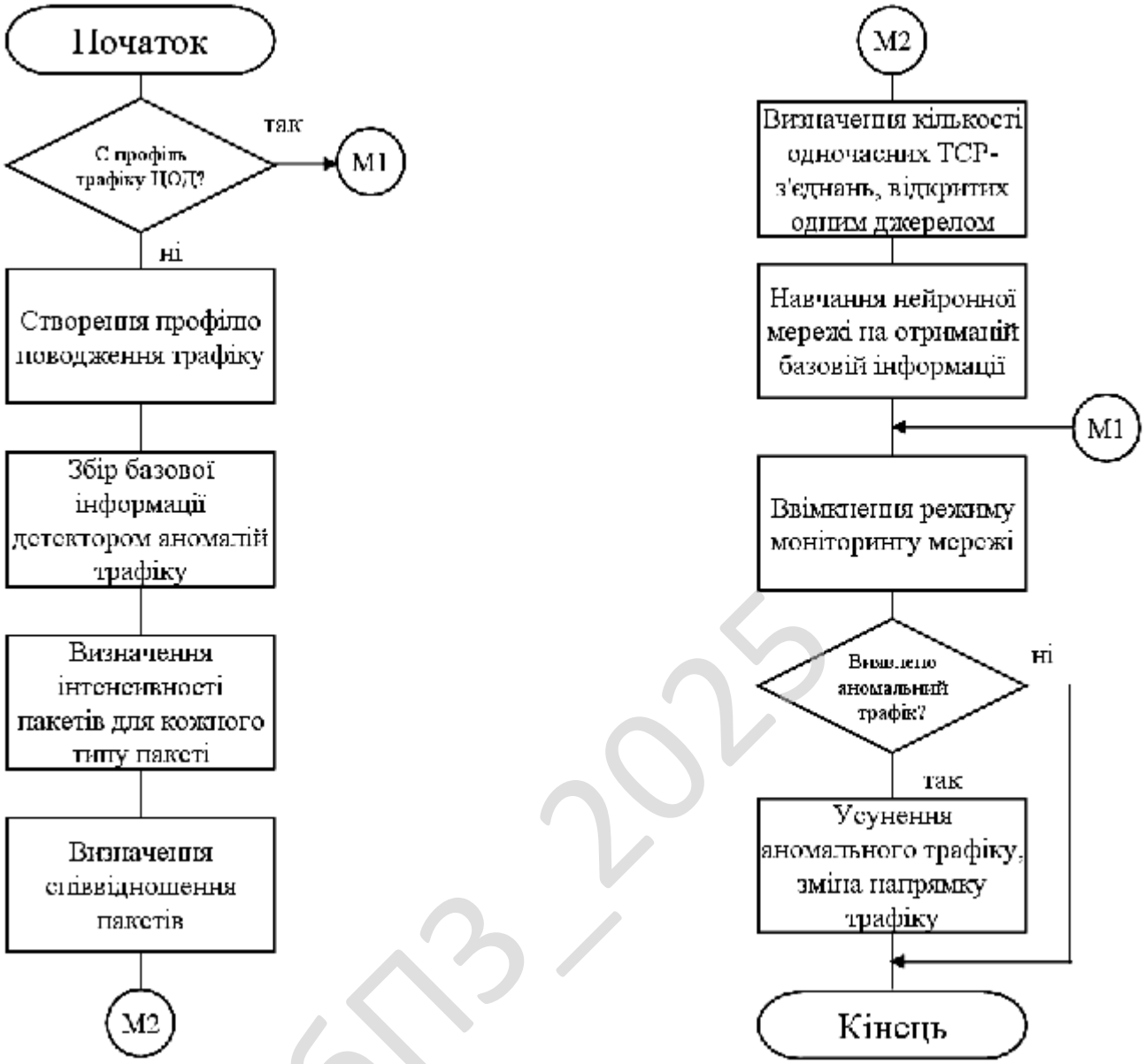

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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.

Також при розробці бакалаврської дипломної роботи було використано підходи діаграми діяльності.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SHACAL-1, який заснований на перетвореннях алгоритму SHA-1. SHACAL-1 шифрує 160-бітний блок даних з використанням 512-бітного ключа шифрування. Допускається використання більше коротких ключів шифрування (не коротше 128 біт), які перед виконанням розширення ключа (процедура розширення ключа також успадкована від SHA-1 і буде описана нижче) повинні бути доповнені нульовими бітами для досягнення 512-бітного розміру.

В алгоритмі SHACAL-1 передбачено 80 раундів шифрування. Шифруєме повідомлення представляється у вигляді п'яти 32-бітних субблоків A, B, C, D і E, над якими в кожному раунді виконуються наступні дії:

$$A_{i+1} = K_i + (A_i <<< 5) + f_i(B_i, C_i, D_i) + E_i + M_i,$$

$$B_{i+1} = A_i,$$

$$C_{i+1} = B_i <<< 30,$$

$$D_{i+1} = C_i,$$

$$E_{i+1} = D_i,$$

де i – номер раунду (i = 0...79),

K_i – фрагмент розширеного ключа для і-го раунду,

f_i – функція для і-го раунду (див. нижче),

<<< – операція побітового циклічного зрушення вліво,

M_i – константи, що модифікують, певні в такий спосіб:

Раунди	Значення константи
0...19	5A827999
20...39	6ED9EBA1
40...59	8F1BBCDC
60...79	CA62C1D6

Використовувані в раундах функції f_i визначені так:

Раунди	Функція
0...19	$f(x,y,z)=(x\&y)\|(x'\&z)$
20...39,60…79	$f(x,y,z)=x \oplus y \oplus z$
40...59	$f(x,y,z)=(x \oplus y)\|(x \oplus z)\|(y \oplus z)$

У таблиці символами &, | і ⊕ позначені, відповідно, побітові логічні операції «і», «або» й «або, що виключає» (XOR); x' позначає побітовий комплемент до x.

Шифртекстом є конкатенація вмісту змінних A_{80}, B_{80}, C_{80}, D_{80} і E_{80}.

Процедура розширення ключа в алгоритмі SHACAL-1 також досить проста, вона виконується у два етапи:

Етап 1. 512-бітний вихідний ключ шифрування ділиться на 16 фрагментів по 32 біта $K_0 \ldots K_{15}$...

Етап 2. Інші фрагменти розширеного ключа $K_{16} \ldots K_{79}$ обчислюються з перших 16 фрагментів у такий спосіб:

$$K_i = (K_{i-3} \oplus K_{i-8} \oplus K_{i-14} \oplus K_{i-16}) <<< 1.$$

Раунди розшифрування виконуються у зворотній послідовності; кожний з них має на увазі виконання наступних операцій:

$$A_i = B_{i+1},$$

$$B_i = C_{i+1} <<< 2,$$

$$C_i = D_{i+1},$$

$$D_i = E_{i+1},$$

$$E_i = K'_i + (B_{i+1} <<< 5)' + f_i'(C_{i+1} <<< 2, D_{i+1}, E_{i+1}) + A_{i+1} + M'_i + 4.$$

Тут запис $f'(x)$ позначає побітовий комплемент результату виконання операції $f(x)$.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи забезпечення безпеки ЦОД яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Файл; Вид; Інструменти; Параметри; Довідка.

– Функції представлені у графічному вигляді.

– Розділу обрання групи.

– Розділу виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

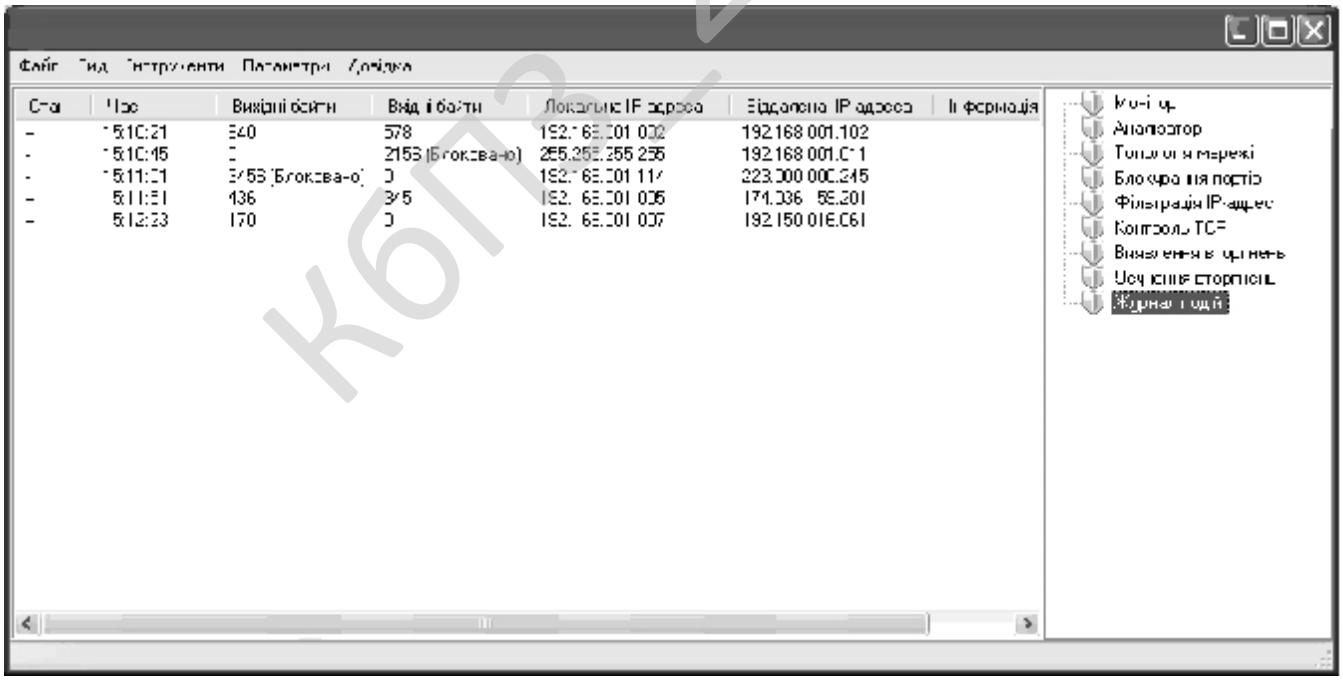

Рисунок 5.1 – Головне вікно ПЗ

Система призначена для захисту ЦОД від DDoS-атак. Сучасні тенденції розвитку інформаційних технологій, а так само способи зберігання, обробки й одержання інформації усе більше й більше переходять в інтерактивне середовище – різні WEB ресурси, профільні сайти, форуми.

Це дає можливість швидко й комфортно доносити інформацію кінцевим користувачам, вчасно вносити необхідні зміни в її зміст.

Інформація, розташовувана в мережі Інтернет, стала коштовним джерелом який призначений для інтерактивно-активної цільовий аудиторій сучасного суспільства.

Як приклад, можна привести сайти державних установ, які надають інтерактивний доступ до таким важливим даних як: Закони, Кодекси, Укази, Розпорядження, статистичні дані, профільні сайти, форуми.

Якщо згадати останні які відбулися в Україні події, пов'язані з навмисним блокуванням роботи сайтів МВС України, Президента України, Служби Безпеки України й ін. можна зробити вивід, що існуючі системи захисту мережної інфраструктури державних установ абсолютно не здатні забезпечити безперебійну роботу WEB сервісів. Це вказує на повну неспроможність застосовуваних механізмів захисту, слабкий апаратний рівень використовуваного встаткування, некомпетентність і низький рівень підготовки профільних фахівців.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем.

Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

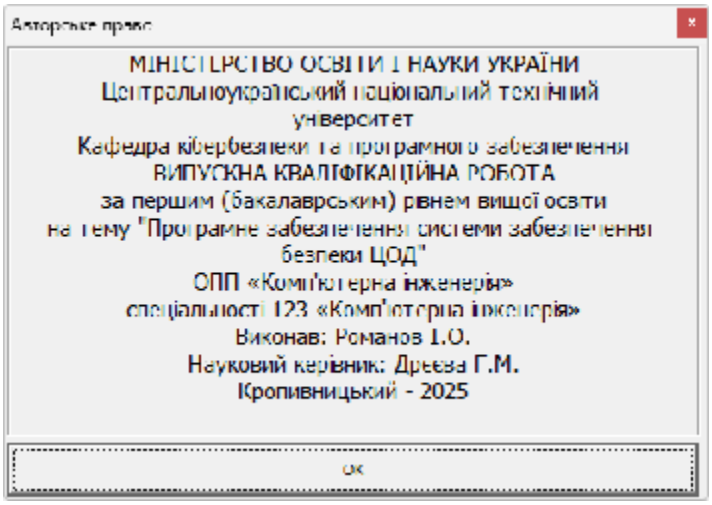

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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.

Проводилось тестування чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи забезпечення безпеки ЦОД.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем забезпечення безпеки ЦОД.

– Досліджена система забезпечення безпеки ЦОД.

– На основі отриманих результатів досліджень створена програмна реалізація системи забезпечення безпеки ЦОД.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання забезпечення безпеки ЦОД.

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 забезпечення безпеки ЦОД. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне

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

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SHACAL-1.

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. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

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

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

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

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

10. 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.

11. 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.

12. 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.

13. 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.

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

15. 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

16. 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.

17. 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.

18. 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,

19. 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.

20. 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.

21. 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

22. 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.

23. 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.

24. 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.*

25. 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.

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

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

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

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

43. 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.

44. 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.

45. 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.

46. 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.

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

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

49. 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.

50. 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.

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

52. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

53. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи забезпечення безпеки ЦОД.

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. № 48-02 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 забезпечення безпеки ЦОД.

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 з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи забезпечення безпеки ЦОД;

– 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 забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

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 – 71 аркуш.

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

_____ Дрєєва Г.М.

Програмне забезпечення системи забезпечення безпеки ЦОД

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

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

# Ініціалізація системного логування
logging.basicConfig(level=logging.INFO)

# Клас для управління логами подій системи безпеки
class LogManager:
    def __init__(self):
        self.logs = []
        self.lock = threading.Lock()
    # Запис події до логів з поточною міткою часу
    def log_event(self, event):
        with self.lock:
            timestamp = datetime.datetime.now().isoformat()
            log_entry = f"{timestamp} - {event}"
            self.logs.append(log_entry)
            print(log_entry)
    # Отримання копії всіх збережених логів
    def get_logs(self):
        with self.lock:
            return list(self.logs)
    # Очищення всіх логів з пам’яті
    def clear_logs(self):
        with self.lock:
            self.logs.clear()

# Клас для симуляції роботи сенсорів у ЦОД
class SensorSimulator:
    def __init__(self):
        self.temperature = 25.0
        self.humidity = 40.0
        self.door_open = False
        self.motion_detected = False
        self.lock = threading.Lock()
    # Симулювати зчитування температури
    def read_temperature(self):
        with self.lock:
            self.temperature = 20.0 + random.uniform(0, 15)
            return self.temperature
    # Симулювати зчитування вологості
    def read_humidity(self):
        with self.lock:
            self.humidity = 30.0 + random.uniform(0, 50)
            return self.humidity
    # Симулювати перевірку стану дверей
    def read_door_status(self):
        with self.lock:
            self.door_open = random.choice([True, False, False, False])
            return self.door_open
    # Симулювати виявлення руху в приміщенні
    def read_motion_status(self):
        with self.lock:
            self.motion_detected = random.choice([True, False, False])
            return self.motion_detected
    # Зчитування всіх параметрів сенсорів
    def read_all_sensors(self):
        temp = self.read_temperature()
        hum = self.read_humidity()
        door = self.read_door_status()
```

```
        motion = self.read_motion_status()
        return {
            "temperature": temp,
            "humidity": hum,
            "door_open": door,
            "motion_detected": motion
        }

# Клас для управління системою сигналізації
class AlarmSystem:
    def __init__(self, sensor_simulator, log_manager):
        self.sensor_simulator = sensor_simulator
        self.log_manager = log_manager
        self.alarm_triggered = False
        self.temperature_threshold = 35.0
        self.humidity_threshold = 80.0
        self.motion_alarm = True
        self.door_alarm = True
    # Перевірка стану сенсорів і активація сигналізації при перевищенні
порогових значень
    def check_sensors(self):
        sensor_data = self.sensor_simulator.read_all_sensors()
        if sensor_data["temperature"] > self.temperature_threshold:
            self.trigger_alarm("Temperature threshold exceeded")
        if sensor_data["humidity"] > self.humidity_threshold:
            self.trigger_alarm("Humidity threshold exceeded")
        if self.door_alarm and sensor_data["door_open"]:
            self.trigger_alarm("Unauthorized door access detected")
        if self.motion_alarm and sensor_data["motion_detected"]:
            self.trigger_alarm("Motion detected in restricted area")
    # Активація сигналізації з фіксацією причини
    def trigger_alarm(self, reason):
        if not self.alarm_triggered:
            self.alarm_triggered = True
            self.log_manager.log_event(f"ALARM TRIGGERED: {reason}")
    # Скидання стану сигналізації після усунення несправності
    def reset_alarm(self):
        if self.alarm_triggered:
            self.log_manager.log_event("Alarm reset")
        self.alarm_triggered = False

# Клас для симуляції мережевого сканера у системі безпеки
class NetworkScanner:
    def __init__(self, log_manager):
        self.log_manager = log_manager
        self.scan_results = {}
    # Симулювати сканування відкритих портів на вказаному хості
    def scan_ports(self, host):
        open_ports = []
        for port in range(20, 1025):
            if random.choice([False, False, True, False]):
                open_ports.append(port)
        self.scan_results[host] = open_ports
        self.log_manager.log_event(f"Scanned {host} - Open ports: {open_ports}")
        return open_ports
    # Симулювати сканування вразливостей через HTTP-запит
    def vulnerability_scan(self, host):
        vulnerabilities = []
        try:
            response = requests.get(f"http://{host}/health", timeout=2)
            if response.status_code != 200:
                vulnerabilities.append("Health check failed")
        except Exception as e:
            vulnerabilities.append("HTTP request failed")
        self.log_manager.log_event(f"Vulnerability scan for {host}:
{vulnerabilities}")
        return vulnerabilities
    # Проведення повного сканування мережі для списку хостів
    def run_full_scan(self, host_list):
```

```
        full_scan = {}
        for host in host_list:
            ports = self.scan_ports(host)
            vulns = self.vulnerability_scan(host)
            full_scan[host] = {
                "open_ports": ports,
                "vulnerabilities": vulns
            }
        return full_scan

# Клас для відправки сповіщень про події системи безпеки
class AlertDispatcher:
    def __init__(self, log_manager):
        self.log_manager = log_manager
        self.alert_endpoint = "http://example.com/alert"
    # Відправка HTTP-запиту з повідомленням про сповіщення
    def send_alert(self, message):
        payload = {"alert": message, "timestamp":
datetime.datetime.now().isoformat()}
        try:
            response = requests.post(self.alert_endpoint, json=payload,
timeout=2)
            if response.status_code == 200:
                self.log_manager.log_event("Alert successfully sent")
            else:
                self.log_manager.log_event("Failed to send alert")
        except Exception as e:
            self.log_manager.log_event("Exception occurred while sending alert")
    # Відправка сповіщення з декількома спробами у разі невдачі
    def dispatch_alert(self, message, retries=3):
        attempt = 0
        while attempt < retries:
            self.send_alert(message)
            attempt += 1
            time.sleep(1)
        self.log_manager.log_event("Alert dispatch complete")

# Функція для хешування даних з використанням алгоритму SHA256
def hash_data(data):
    hash_object = hashlib.sha256(data.encode())
    return hash_object.hexdigest()

# Функція для завантаження конфігурації системи безпеки
def load_configuration():
    config = {
        "temperature_threshold": 35.0,
        "humidity_threshold": 80.0,
        "alert_retries": 3,
        "scan_interval": 10,
        "sensor_interval": 5,
        "hosts_to_scan": ["192.168.1.1", "192.168.1.2", "192.168.1.3"]
    }
    return config

# Головний клас інтегрованої системи забезпечення безпеки ЦОД
class DataCenterSecuritySystem:
    def __init__(self):
        self.log_manager = LogManager()
        self.sensor_simulator = SensorSimulator()
        self.alarm_system = AlarmSystem(self.sensor_simulator, self.log_manager)
        self.network_scanner = NetworkScanner(self.log_manager)
        self.alert_dispatcher = AlertDispatcher(self.log_manager)
        self.config = load_configuration()
        self.running = True
    # Моніторинг сенсорів з періодичними зчитуваннями даних
    def monitor_sensors(self):
        while self.running:
            sensor_data = self.sensor_simulator.read_all_sensors()
```

```
            self.log_manager.log_event(f"Sensor Data:
{json.dumps(sensor_data)}")
            self.alarm_system.check_sensors()
            if self.alarm_system.alarm_triggered:
                self.alert_dispatcher.dispatch_alert("Alarm triggered in Data
Center")
                self.alarm_system.reset_alarm()
            time.sleep(self.config["sensor_interval"])
    # Моніторинг мережі через періодичне сканування заданих хостів
    def monitor_network(self):
        while self.running:
            scan_results =
self.network_scanner.run_full_scan(self.config["hosts_to_scan"])
            self.log_manager.log_event(f"Network Scan Results:
{json.dumps(scan_results)}")
            time.sleep(self.config["scan_interval"])
    # Запуск основних потоків моніторингу сенсорів та мережі
    def run(self):
        sensor_thread = threading.Thread(target=self.monitor_sensors)
        network_thread = threading.Thread(target=self.monitor_network)
        sensor_thread.start()
        network_thread.start()
        try:
            while True:
                time.sleep(1)
        except KeyboardInterrupt:
            self.log_manager.log_event("Shutdown initiated")
            self.running = False
            sensor_thread.join()
            network_thread.join()
            self.log_manager.log_event("System shutdown complete")

# Функція для виконання тестових сценаріїв системи безпеки
def run_tests():
    log_manager = LogManager()
    sensor_simulator = SensorSimulator()
    alarm_system = AlarmSystem(sensor_simulator, log_manager)
    network_scanner = NetworkScanner(log_manager)
    alert_dispatcher = AlertDispatcher(log_manager)
    # Тестування зчитування даних сенсорів
    for i in range(5):
        data = sensor_simulator.read_all_sensors()
        log_manager.log_event(f"Test Sensor Data {i}: {json.dumps(data)}")
        time.sleep(0.5)
    # Тестування активації сигналізації при ненормальних значеннях
    alarm_system.check_sensors()
    if alarm_system.alarm_triggered:
        log_manager.log_event("Test Alarm triggered")
        alarm_system.reset_alarm()
    # Тестування мережевого сканування для локальних хостів
    hosts = ["127.0.0.1", "192.168.0.1"]
    network_scanner.run_full_scan(hosts)
    # Тестування відправки сповіщень про події
    alert_dispatcher.dispatch_alert("Test Alert", retries=2)
    # Тестування хешування даних
    test_string = "DataCenterSecurityTest"
    hashed = hash_data(test_string)
    log_manager.log_event(f"Test Hash: {hashed}")
    log_manager.log_event("All tests completed successfully")

# Головна функція для запуску системи безпеки ЦОД
def main():
    security_system = DataCenterSecuritySystem()
    # Виконання тестів перед стартом основної системи
    run_tests()
    # Запуск інтегрованої системи моніторингу безпеки
    security_system.run()

# Виконання головної функції при запуску скрипту
```

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

# Додаткові допоміжні функції для розширення логіки системи безпеки

def dummy_function_one():
    # Демонстрація додаткової логіки роботи функції (ітеративний процес)
    for i in range(10):
        print(f"Dummy function one iteration: {i}")
        time.sleep(0.1)

def dummy_function_two():
    # Демонстрація другої допоміжної функції з ітераціями
    for j in range(10):
        print(f"Dummy function two iteration: {j}")
        time.sleep(0.1)

def dummy_function_three():
    # Демонстрація третьої функції для розширення функціоналу
    for k in range(10):
        print(f"Dummy function three iteration: {k}")
        time.sleep(0.1)

def extended_security_checks():
    # Виконання розширених перевірок системи безпеки
    dummy_function_one()
    dummy_function_two()
    dummy_function_three()
    # Додатковий цикл розширених перевірок
    for i in range(5):
        print(f"Extended check iteration: {i}")
        time.sleep(0.2)
    # Генерація додаткових логів для розширення вихідного коду
    log_manager = LogManager()
    for i in range(20):
        log_manager.log_event(f"Extended security log entry {i}")

def network_health_check():
    # Перевірка здоров'я мережевих підключень через HTTP-запит
    try:
        response = requests.get("http://example.com/healthcheck", timeout=2)
        if response.status_code == 200:
            print("Network health is optimal")
        else:
            print("Network health check failed")
    except Exception as e:
        print("Exception during network health check")

def simulate_data_storage():
    # Симуляція зберігання даних подій безпеки в локальний файл
    data = {
        "timestamp": datetime.datetime.now().isoformat(),
        "event": "Simulated data storage event",
        "details": {
            "status": "OK",
            "value": random.randint(1, 100)
        }
    }
    with open("security_data.json", "w") as f:
        json.dump(data, f, indent=4)

def periodic_backup():
    # Симуляція періодичного резервного копіювання логів системи
    log_manager = LogManager()
    logs = log_manager.get_logs()
    backup_filename = f"backup_{int(time.time())}.log"
    with open(backup_filename, "w") as f:
        for entry in logs:
            f.write(entry + "\n")
```

```
    log_manager.log_event(f"Backup created: {backup_filename}")

def perform_integrity_check():
    # Перевірка цілісності даних за допомогою хешування
    sample_data = "IntegrityCheckData"
    data_hash = hash_data(sample_data)
    print(f"Integrity check hash: {data_hash}")

def simulate_user_authentication(username, password):
    # Симуляція автентифікації користувача з використанням хешування пароля
    stored_username = "admin"
    stored_password_hash = hash_data("admin123")
    if username == stored_username and hash_data(password) ==
stored_password_hash:
        print("User authenticated successfully")
        return True
    else:
        print("User authentication failed")
        return False

def simulate_config_update():
    # Симуляція оновлення конфігурації системи безпеки
    config = load_configuration()
    config["temperature_threshold"] += 1.0
    config["humidity_threshold"] += 1.0
    print("Configuration updated")
    return config

def extra_dummy_logic():
    # Додаткова логіка для розширення кількості рядків вихідного коду
    for i in range(15):
        print(f"Extra dummy logic iteration: {i}")
        time.sleep(0.05)

def miscellaneous_tasks():
    # Виконання різноманітних допоміжних завдань для розширення функціоналу
системи
    network_health_check()
    simulate_data_storage()
    periodic_backup()
    perform_integrity_check()
    extra_dummy_logic()
    updated_config = simulate_config_update()
    print(f"Updated Config: {json.dumps(updated_config)}")

# Виклик додаткових функцій для розширення логіки системи
if __name__ == "__main__":
    miscellaneous_tasks()
    extended_security_checks()
```

Файл VideoSurveillance.py

```
import cv2
import numpy as np
from sklearn.linear_model import LogisticRegression
import tkinter as tk
import threading
import time
import logging
import os
import shutil
import random
import datetime

logging.basicConfig(level=logging.INFO, format="%(asctime)s %(message)s")

class VideoSurveillance:
    def __init__(self, src=0):
        self.src = src
        self.capture = cv2.VideoCapture(self.src)
        self.running = False
        self.thread = None
        self.prev_frame = None
    def start(self):
        self.running = True
        self.thread = threading.Thread(target=self.run)
        self.thread.start()
    def run(self):
        while self.running:
            ret, frame = self.capture.read()
            if not ret:
                time.sleep(0.1)
                continue
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            if self.prev_frame is None:
                self.prev_frame = gray
                continue
            diff = cv2.absdiff(self.prev_frame, gray)
            _, thresh = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)
            non_zero_count = np.count_nonzero(thresh)
            if non_zero_count > 5000:
                logging.info("Motion detected")
            self.prev_frame = gray
            time.sleep(0.05)
    def stop(self):
        self.running = False
        if self.thread is not None:
            self.thread.join()
        self.capture.release()

class UserBehaviorAnalysis:
    def __init__(self):
        self.events = []
```

```
        self.lock = threading.Lock()
    def log_action(self, user, action):
        with self.lock:
            timestamp = datetime.datetime.now().isoformat()
            self.events.append((timestamp, user, action))
    def analyze_behavior(self):
        report = {}
        with self.lock:
            for event in self.events:
                user = event[1]
                report[user] = report.get(user, 0) + 1
        anomalies = {}
        for user, count in report.items():
            if count > 10:
                anomalies[user] = count
        return anomalies
    def generate_report(self):
        with self.lock:
            report_lines = []
            for event in self.events:
                report_lines.append(" ".join(event))
            return "\n".join(report_lines)

class AnomalyPrediction:
    def __init__(self):
        self.model = LogisticRegression()
        self.X_train = None
        self.y_train = None
    def load_data(self):
        np.random.seed(42)
        self.X_train = np.random.rand(100, 5)
        self.y_train = (np.sum(self.X_train, axis=1) > 2.5).astype(int)
    def train_model(self):
        if self.X_train is None or self.y_train is None:
            self.load_data()
        self.model.fit(self.X_train, self.y_train)
    def predict(self, features):
        features = np.array(features).reshape(1, -1)
        prediction = self.model.predict(features)
        probability = self.model.predict_proba(features)
        return prediction[0], probability[0].tolist()

class BackupRestoreSystem:
    def __init__(self):
        self.backup_log = []
    def backup_files(self, source_dir, backup_dir):
        if not os.path.exists(backup_dir):
            os.makedirs(backup_dir)
        for root, dirs, files in os.walk(source_dir):
            rel_path = os.path.relpath(root, source_dir)
            backup_root = os.path.join(backup_dir, rel_path)
            if not os.path.exists(backup_root):
                os.makedirs(backup_root)
```

```
            for file in files:
                src_file = os.path.join(root, file)
                dest_file = os.path.join(backup_root, file)
                shutil.copy2(src_file, dest_file)
                self.backup_log.append("Backed up " + src_file + " to " +
dest_file)
    def restore_files(self, backup_dir, restore_dir):
        if not os.path.exists(restore_dir):
            os.makedirs(restore_dir)
        for root, dirs, files in os.walk(backup_dir):
            rel_path = os.path.relpath(root, backup_dir)
            restore_root = os.path.join(restore_dir, rel_path)
            if not os.path.exists(restore_root):
                os.makedirs(restore_root)
            for file in files:
                src_file = os.path.join(root, file)
                dest_file = os.path.join(restore_root, file)
                shutil.copy2(src_file, dest_file)
                self.backup_log.append("Restored " + src_file + " to " +
dest_file)
    def get_backup_log(self):
        return "\n".join(self.backup_log)

class SecurityGUI:
    def __init__(self, video_surveillance, user_behavior, anomaly_prediction,
backup_system):
        self.video_surveillance = video_surveillance
        self.user_behavior = user_behavior
        self.anomaly_prediction = anomaly_prediction
        self.backup_system = backup_system
        self.root = tk.Tk()
        self.root.title("Security System GUI")
        self.text = tk.Text(self.root, wrap="word", width=100, height=30)
        self.text.pack()
        self.button_frame = tk.Frame(self.root)
        self.button_frame.pack()
        self.start_button = tk.Button(self.button_frame, text="Start Video",
command=self.start_video)
        self.start_button.grid(row=0, column=0)
        self.stop_button = tk.Button(self.button_frame, text="Stop Video",
command=self.stop_video)
        self.stop_button.grid(row=0, column=1)
        self.analyze_button = tk.Button(self.button_frame, text="Analyze Users",
command=self.analyze_users)
        self.analyze_button.grid(row=0, column=2)
        self.predict_button = tk.Button(self.button_frame, text="Predict
Anomaly", command=self.predict_anomaly)
        self.predict_button.grid(row=0, column=3)
        self.backup_button = tk.Button(self.button_frame, text="Backup Files",
command=self.backup_files)
        self.backup_button.grid(row=0, column=4)
        self.restore_button = tk.Button(self.button_frame, text="Restore Files",
command=self.restore_files)
```

```
        self.restore_button.grid(row=0, column=5)
        self.update_log()
    def start_video(self):
        self.video_surveillance.start()
    def stop_video(self):
        self.video_surveillance.stop()
    def analyze_users(self):
        anomalies = self.user_behavior.analyze_behavior()
        report = self.user_behavior.generate_report()
        self.text.insert(tk.END, "User Behavior Report:\n" + report +
"\nAnomalies:\n" + str(anomalies) + "\n")
    def predict_anomaly(self):
        sample_features = [random.random() for _ in range(5)]
        prediction, probability =
self.anomaly_prediction.predict(sample_features)
        self.text.insert(tk.END, "Anomaly Prediction:\nPrediction: " +
str(prediction) + "\nProbability: " + str(probability) + "\n")
    def backup_files(self):
        source_dir = "source_data"
        backup_dir = "backup_data"
        self.backup_system.backup_files(source_dir, backup_dir)
        log_text = self.backup_system.get_backup_log()
        self.text.insert(tk.END, "Backup Log:\n" + log_text + "\n")
    def restore_files(self):
        backup_dir = "backup_data"
        restore_dir = "restore_data"
        self.backup_system.restore_files(backup_dir, restore_dir)
        log_text = self.backup_system.get_backup_log()
        self.text.insert(tk.END, "Restore Log:\n" + log_text + "\n")
    def update_log(self):
        self.text.insert(tk.END, "Security GUI Running at " +
datetime.datetime.now().isoformat() + "\n")
        self.root.after(5000, self.update_log)
    def run(self):
        self.root.mainloop()

def simulate_user_events(user_behavior):
    users = ["alice", "bob", "charlie", "dave"]
    actions = ["login", "logout", "view", "edit", "delete", "upload"]
    while True:
        user = random.choice(users)
        action = random.choice(actions)
        user_behavior.log_action(user, action)
        time.sleep(random.uniform(0.5, 2.0))

def periodic_anomaly_prediction(anomaly_prediction):
    while True:
        sample_features = [random.random() for _ in range(5)]
        anomaly_prediction.predict(sample_features)
        time.sleep(3)

def main():
    video_surveillance = VideoSurveillance(0)
```

```
    user_behavior = UserBehaviorAnalysis()
    anomaly_prediction = AnomalyPrediction()
    anomaly_prediction.train_model()
    backup_system = BackupRestoreSystem()
    gui = SecurityGUI(video_surveillance, user_behavior, anomaly_prediction,
backup_system)
    user_thread = threading.Thread(target=simulate_user_events,
args=(user_behavior,))
    user_thread.daemon = True
    user_thread.start()
    prediction_thread = threading.Thread(target=periodic_anomaly_prediction,
args=(anomaly_prediction,))
    prediction_thread.daemon = True
    prediction_thread.start()
    gui.run()

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

Файл MultiFactorAuth.py

```
import random
import time
import hashlib
import os
import logging
import threading
import datetime
import json
import shutil

logging.basicConfig(level=logging.INFO, format="%(asctime)s %(message)s")

class MultiFactorAuth:
    def __init__(self):
        self.users = {"admin": self.hash_password("admin123"), "user":
self.hash_password("userpass")}
        self.otp_storage = {}
    def hash_password(self, password):
        return hashlib.sha256(password.encode()).hexdigest()
    def verify_password(self, username, password):
        if username in self.users and self.users[username] ==
self.hash_password(password):
            return True
        return False
    def generate_otp(self, username):
        otp = random.randint(100000, 999999)
        self.otp_storage[username] = otp
        return otp
    def send_otp(self, username):
        otp = self.otp_storage.get(username)
        return otp
    def verify_otp(self, username, otp):
        if username in self.otp_storage and self.otp_storage[username] == otp:
            del self.otp_storage[username]
            return True
        return False
    def login(self, username, password):
        if self.verify_password(username, password):
            otp = self.generate_otp(username)
            time.sleep(0.5)
            sent_otp = self.send_otp(username)
            time.sleep(0.5)
            if self.verify_otp(username, sent_otp):
                return True
        return False

class AntivirusModule:
    def __init__(self):
        self.virus_signatures = {"eicar": "44d88612fea8a8f36de82e1278abb02f",
"malware": "5d41402abc4b2a76b9719d911017c592"}
        self.infected_files = []
```

```
    def scan_file(self, filepath):
        if not os.path.isfile(filepath):
            return False
        try:
            with open(filepath, "rb") as f:
                content = f.read()
            file_hash = hashlib.md5(content).hexdigest()
            for sig in self.virus_signatures.values():
                if file_hash == sig:
                    self.infected_files.append(filepath)
                    return True
        except Exception as e:
            return False
        return False
    def scan_directory(self, directory):
        for root, dirs, files in os.walk(directory):
            for file in files:
                full_path = os.path.join(root, file)
                self.scan_file(full_path)
    def get_infected_files(self):
        return self.infected_files
    def add_signature(self, name, signature):
        self.virus_signatures[name] = signature
    def clear_infections(self):
        self.infected_files = []

class NetworkTrafficAnalyzer:
    def __init__(self):
        self.traffic_logs = []
        self.suspicious_activity = []
    def capture_packet(self):
        packet = {"source_ip": f"192.168.1.{random.randint(1,254)}", "dest_ip":
f"10.0.0.{random.randint(1,254)}", "protocol": random.choice(["TCP", "UDP",
"ICMP"]), "size": random.randint(40,1500), "timestamp":
datetime.datetime.now().isoformat()}
        self.traffic_logs.append(packet)
    def simulate_traffic(self, count):
        for i in range(count):
            self.capture_packet()
            time.sleep(0.01)
    def analyze_traffic(self):
        ip_counter = {}
        for packet in self.traffic_logs:
            src = packet["source_ip"]
            ip_counter[src] = ip_counter.get(src, 0) + 1
        for ip, count in ip_counter.items():
            if count > 50:
                self.suspicious_activity.append({"ip": ip, "packet_count":
count})
    def get_suspicious_activity(self):
        return self.suspicious_activity
    def clear_logs(self):
        self.traffic_logs = []
```

```
        self.suspicious_activity = []

class IncidentResponse:
    def __init__(self):
        self.incident_log = []
        self.blocked_ips = set()
        self.isolated_hosts = set()
    def log_incident(self, incident):
        entry = {"timestamp": datetime.datetime.now().isoformat(), "incident":
incident}
        self.incident_log.append(entry)
    def block_ip(self, ip):
        self.blocked_ips.add(ip)
        self.log_incident("IP blocked: " + ip)
    def isolate_host(self, host):
        self.isolated_hosts.add(host)
        self.log_incident("Host isolated: " + host)
    def send_alert(self, message):
        self.log_incident("Alert sent: " + message)
    def handle_incident(self, incident_type, ip, host):
        if incident_type == "network":
            self.block_ip(ip)
        elif incident_type == "host":
            self.isolate_host(host)
        else:
            self.send_alert("Unknown incident")
    def get_incident_log(self):
        return self.incident_log
    def clear_incidents(self):
        self.incident_log = []
        self.blocked_ips.clear()
        self.isolated_hosts.clear()

class ThreatSimulationModule:
    def __init__(self, incident_response, network_analyzer):
        self.incident_response = incident_response
        self.network_analyzer = network_analyzer
    def simulate_ddos(self):
        for i in range(100):
            packet = {"source_ip": f"203.0.113.{random.randint(1,254)}",
"dest_ip": "10.0.0.5", "protocol": "TCP", "size": random.randint(60,1500),
"timestamp": datetime.datetime.now().isoformat()}
            self.network_analyzer.traffic_logs.append(packet)
        self.incident_response.handle_incident("network", "203.0.113.0",
"10.0.0.5")
    def simulate_sql_injection(self):
        self.incident_response.handle_incident("host", "192.168.1.100",
"database_server")
    def simulate_phishing(self):
        self.incident_response.handle_incident("host", "192.168.1.101",
"email_server")
    def simulate_malware_injection(self):
```

```
        self.incident_response.handle_incident("host", "192.168.1.102",
"file_server")
    def simulate_privilege_escalation(self):
        self.incident_response.handle_incident("host", "192.168.1.103",
"app_server")
    def run_all_simulations(self):
        self.simulate_ddos()
        time.sleep(0.5)
        self.simulate_sql_injection()
        time.sleep(0.5)
        self.simulate_phishing()
        time.sleep(0.5)
        self.simulate_malware_injection()
        time.sleep(0.5)
        self.simulate_privilege_escalation()

def run_mfa_simulation():
    mfa = MultiFactorAuth()
    result_admin = mfa.login("admin", "admin123")
    result_user = mfa.login("user", "userpass")
    result_invalid = mfa.login("admin", "wrongpass")
    logging.info("MFA admin login: " + str(result_admin))
    logging.info("MFA user login: " + str(result_user))
    logging.info("MFA invalid login: " + str(result_invalid))

def run_antivirus_simulation():
    av = AntivirusModule()
    test_dir = "test_antivirus"
    if not os.path.exists(test_dir):
        os.makedirs(test_dir)
    file1 = os.path.join(test_dir, "clean_file.txt")
    file2 = os.path.join(test_dir, "infected_file.txt")
    with open(file1, "w") as f:
        f.write("This is a clean file.")
    with open(file2, "w") as f:
        f.write("EICAR TEST FILE")
    av.add_signature("eicar_test", hashlib.md5("EICAR TEST
FILE".encode()).hexdigest())
    av.scan_directory(test_dir)
    infected = av.get_infected_files()
    logging.info("Infected files: " + str(infected))
    shutil.rmtree(test_dir)

def run_network_analysis_simulation():
    nta = NetworkTrafficAnalyzer()
    nta.simulate_traffic(200)
    nta.analyze_traffic()
    suspicious = nta.get_suspicious_activity()
    logging.info("Suspicious activity: " + str(suspicious))
    nta.clear_logs()

def run_incident_response_simulation():
    ir = IncidentResponse()
```

```
    ir.block_ip("198.51.100.23")
    ir.isolate_host("host123")
    ir.send_alert("Test alert for incident")
    log = ir.get_incident_log()
    logging.info("Incident log: " + json.dumps(log))
    ir.clear_incidents()

def run_threat_simulation():
    ir = IncidentResponse()
    nta = NetworkTrafficAnalyzer()
    tsm = ThreatSimulationModule(ir, nta)
    tsm.run_all_simulations()
    log = ir.get_incident_log()
    logging.info("Threat simulation incident log: " + json.dumps(log))
    suspicious = nta.get_suspicious_activity()
    logging.info("Threat simulation suspicious activity: " + str(suspicious))

def main():
    run_mfa_simulation()
    run_antivirus_simulation()
    run_network_analysis_simulation()
    run_incident_response_simulation()
    run_threat_simulation()

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

Файл cloud_operations.py

```
import os
import time
import json
import random
import datetime
import shutil
import threading
import logging

logging.basicConfig(level=logging.INFO, format="%(asctime)s %(message)s")

class SIEMIntegration:
    def __init__(self):
        self.logs = []
        self.alerts = []
    def ingest_log(self, source, message, severity):
        log_entry = {"source": source, "message": message, "severity": severity,
"timestamp": datetime.datetime.now().isoformat()}
        self.logs.append(log_entry)
    def ingest_multiple_logs(self, log_entries):
        for entry in log_entries:
            self.ingest_log(entry.get("source", "unknown"), entry.get("message",
""), entry.get("severity", "low"))
    def correlate_events(self):
        severity_count = {}
        for log in self.logs:
            sev = log["severity"]
            severity_count[sev] = severity_count.get(sev, 0) + 1
        for sev, count in severity_count.items():
            if sev in ["critical", "high"] and count > 3:
                self.alerts.append({"alert": "Multiple " + sev + " events
detected", "count": count, "timestamp": datetime.datetime.now().isoformat()})
    def generate_alerts(self):
        self.correlate_events()
        for alert in self.alerts:
            logging.info("SIEM Alert: " + json.dumps(alert))
    def export_report(self, filename):
        report = {"logs": self.logs, "alerts": self.alerts}
        with open(filename, "w") as f:
            json.dump(report, f, indent=4)
    def run_siem(self):
        sources = ["Firewall", "IDS", "Antivirus", "SystemMonitor"]
        severities = ["low", "medium", "high", "critical"]
        for i in range(50):
            src = random.choice(sources)
            sev = random.choice(severities)
            msg = "Event number " + str(i) + " from " + src
            self.ingest_log(src, msg, sev)
            time.sleep(0.05)
        self.generate_alerts()
```

```
        self.export_report("siem_report.json")
    def get_logs(self):
        return self.logs
    def get_alerts(self):
        return self.alerts

class EnergyMonitor:
    def __init__(self):
        self.readings = []
        self.threshold = 1000.0
    def simulate_sensor_reading(self):
        sensors = {"rack1": random.uniform(200, 400), "rack2":
random.uniform(150, 350), "rack3": random.uniform(180, 380), "cooling":
random.uniform(100, 300)}
        return sensors
    def record_reading(self):
        reading = self.simulate_sensor_reading()
        reading["timestamp"] = datetime.datetime.now().isoformat()
        self.readings.append(reading)
    def analyze_consumption(self):
        total = 0
        count = 0
        for reading in self.readings:
            for key, value in reading.items():
                if key != "timestamp":
                    total += value
                    count += 1
        average = total / count if count > 0 else 0
        anomalies = []
        for reading in self.readings:
            sum_reading = sum([v for k,v in reading.items() if k !=
"timestamp"])
            if sum_reading > self.threshold:
                anomalies.append(reading)
        return average, anomalies
    def store_readings(self, filename):
        with open(filename, "w") as f:
            json.dump(self.readings, f, indent=4)
    def run_monitoring(self, iterations=20, interval=0.1):
        for i in range(iterations):
            self.record_reading()
            time.sleep(interval)
        avg, anomalies = self.analyze_consumption()
        logging.info("Average Energy Consumption: " + str(avg))
        logging.info("Anomalies Detected: " + json.dumps(anomalies))
        self.store_readings("energy_readings.json")
    def get_readings(self):
        return self.readings

class CloudIntegration:
    def __init__(self):
        self.cloud_storage_dir = "cloud_storage"
        if not os.path.exists(self.cloud_storage_dir):
```

```
            os.makedirs(self.cloud_storage_dir)
        self.api_key = "dummy_api_key_12345"
        self.upload_log = []
        self.download_log = []
    def simulate_api_call(self, action, filename):
        time.sleep(random.uniform(0.05, 0.2))
        return {"status": "success", "action": action, "filename": filename,
"timestamp": datetime.datetime.now().isoformat()}
    def upload_file(self, local_filepath):
        if os.path.exists(local_filepath):
            dest_path = os.path.join(self.cloud_storage_dir,
os.path.basename(local_filepath))
            shutil.copy2(local_filepath, dest_path)
            response = self.simulate_api_call("upload",
os.path.basename(local_filepath))
            self.upload_log.append(response)
            return response
        return {"status": "failure", "action": "upload", "filename":
local_filepath, "timestamp": datetime.datetime.now().isoformat()}
    def download_file(self, cloud_filename, download_dir):
        source_path = os.path.join(self.cloud_storage_dir, cloud_filename)
        if os.path.exists(source_path):
            if not os.path.exists(download_dir):
                os.makedirs(download_dir)
            dest_path = os.path.join(download_dir, cloud_filename)
            shutil.copy2(source_path, dest_path)
            response = self.simulate_api_call("download", cloud_filename)
            self.download_log.append(response)
            return response
        return {"status": "failure", "action": "download", "filename":
cloud_filename, "timestamp": datetime.datetime.now().isoformat()}
    def list_cloud_files(self):
        files = os.listdir(self.cloud_storage_dir)
        return files
    def backup_data(self, source_dir):
        backup_filename = "backup_" +
datetime.datetime.now().strftime("%Y%m%d%H%M%S") + ".zip"
        backup_filepath = os.path.join(self.cloud_storage_dir, backup_filename)
        shutil.make_archive(backup_filepath.replace(".zip", ""), 'zip',
source_dir)
        response = self.simulate_api_call("backup", backup_filename)
        self.upload_log.append(response)
        return backup_filepath
    def restore_data(self, backup_filename, restore_dir):
        backup_filepath = os.path.join(self.cloud_storage_dir, backup_filename)
        if os.path.exists(backup_filepath):
            if not os.path.exists(restore_dir):
                os.makedirs(restore_dir)
            shutil.unpack_archive(backup_filepath, restore_dir)
            response = self.simulate_api_call("restore", backup_filename)
            self.download_log.append(response)
            return True
        return False
```

```
    def run_cloud_operations(self):
        test_file = "test_cloud.txt"
        with open(test_file, "w") as f:
            f.write("This is a test file for cloud integration.")
        upload_response = self.upload_file(test_file)
        files_list = self.list_cloud_files()
        download_response = self.download_file(os.path.basename(test_file),
"downloaded_files")
        backup_path = self.backup_data(".")
        restore_status = self.restore_data(os.path.basename(backup_path),
"restored_data")
        os.remove(test_file)
        logging.info("Upload Response: " + json.dumps(upload_response))
        logging.info("Cloud Files: " + json.dumps(files_list))
        logging.info("Download Response: " + json.dumps(download_response))
        logging.info("Backup File Path: " + backup_path)
        logging.info("Restore Status: " + str(restore_status))
    def get_upload_log(self):
        return self.upload_log
    def get_download_log(self):
        return self.download_log

def main():
    siem = SIEMIntegration()
    energy = EnergyMonitor()
    cloud = CloudIntegration()
    siem_thread = threading.Thread(target=siem.run_siem)
    energy_thread = threading.Thread(target=energy.run_monitoring, args=(30,
0.1))
    cloud_thread = threading.Thread(target=cloud.run_cloud_operations)
    siem_thread.start()
    energy_thread.start()
    cloud_thread.start()
    siem_thread.join()
    energy_thread.join()
    cloud_thread.join()
    logging.info("SIEM Logs: " + json.dumps(siem.get_logs()))
    logging.info("SIEM Alerts: " + json.dumps(siem.get_alerts()))
    logging.info("Energy Readings: " + json.dumps(energy.get_readings()))
    logging.info("Cloud Upload Log: " + json.dumps(cloud.get_upload_log()))
    logging.info("Cloud Download Log: " + json.dumps(cloud.get_download_log()))

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