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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі”

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

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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

Спеціальність *125 “*Кібербезпека*”*

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

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

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

Крутікову Олександру Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

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

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

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

КАЛЕНДАРНИЙ ПЛАН

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

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Крутіков О.А.
(прізвище та ініціали)

АНОТАЦІЯ

Крутіков О.А. Програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

Метою розробки є програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

Результат роботи – програмна реалізація системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, безпечний доступ

ABSTRACT

Krutikov O.A. Software for a cybersecurity system for secure access from a mobile device to corporate network resources. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system for secure access from a mobile device to corporate network resources.

The purpose of the development is to develop software for a cybersecurity system for secure access from a mobile device to corporate network resources.

The result of the work is a software implementation of a cybersecurity system for secure access from a mobile device to corporate network resources.

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

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

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

The program is developed in the Python environment.

Keywords: cybersecurity, secure access

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
УЦ	–	удостовірюючий центр
ЦС	–	центр сертифікації
AH	–	Authentication Header – протокол автентифікуючого заголовку
DES	–	симетричний алгоритм шифрування
ESP	–	протокол інкапсулюючий захист вмісту
IKE	–	протокол обміну ключами в Інтернеті
ISAKMP	–	протокол керування ключами й контекстами безпеки Інтернету
HMAC	–	геш-код повідомлення
MAC	–	код автентифікації
MD5	–	геш-функція
MLA	–	агент списку розсилання
MOSS	–	MIME Object Security Services
OAKLEY	–	протокол обчислення ключів
PEM	–	Privacy Enhanced Mail
PGP	–	Pretty Good Privacy
PKI	–	інфраструктура відкритих ключів
RSA	–	несиметричний алгоритм шифрування
S/MIME	–	Secure Multipurpose Internet Mail Extension
SHA-1	–	геш-функція
SPI	–	Security Parameters Index – індекс параметрів безпеки
SSL	–	Secure Socket Layer
TCP	–	транспортний протокол
TLS	–	Transport Layer Security – протокол безпеки транспортного рівня
VPN	–	Virtual Private Networks – віртуальна приватна мережа

ВСТУП

Актуальність теми. По деяких експертних оцінках, попит на проекти по інформаційній безпеці щорічно росте на 27-30%, і до 2029 року цей тренд буде зберігатися. 2025 рік продовжить основні тенденції попередніх років. Українські компанії найчастіше зіштовхуються зі шкідливими програмами, спробами несанкціонованого проникнення в систему й фішингом. Серйозні проблеми викликають уразливості у встановленому ПЗ й використання співробітниками власних мобільних пристроїв при доступі до корпоративної мережі.

Рівень захищеності бізнесу як і раніше недостатньо високий, а підвищити його заважають обмежений бюджет, нерозуміння керівництвом завдань інформаційної безпеки, недостача кваліфікованих кадрів.

Кожна третя компанія дотепер не повною мірою користується засобами захисту від шкідливих програм, а одна зі ста й зовсім не захищена. На думку фахівців, для створення по-справжньому ефективної системи ІБ необхідно інвестувати в даний напрямок як мінімум на 25% більше, ніж витрачається сьогодні.

Ряд експертів називають 2025-й «роком автентифікації», тому що багато компаній, усвідомивши недостатність традиційного захисту за допомогою логіна/пароля, стануть розгортати розвинені засоби автентифікації. «Парольна модель захисту мертва, – заявляють в Fortinet. – Їй на зміну прийдеться двофакторна автентифікація».

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

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

– Огляд існуючих систем безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

– Дослідження системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

– Програмна реалізація системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі, 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 реалізації програмного забезпечення безпечного доступу з мобільного пристрою до ресурсів корпоративної мереж**.** Як відзначають, хоча як і раніше домінують безсистемні атаки, спрямовані на крадіжку особистої інформації у випадкових користувачів, помітним явищем стали цілеспрямовані акції, спроби проникнення в корпоративну мережу конкретної організації. 16% компаній вважають, що в майбутньому ця погроза стане для них самою актуальною. Нерідко головним завданням є збір конфіденційних даних, які можна з вигодою продати.

Цільові атаки зовсім не обов'язково пов'язані із застосуванням витончених методів: у більшості випадків слабкою ланкою стає «людський фактор».

В 2025 році ще більш розповсюдженим явищем стане кібершпигунство, причому мішенню може виявитися будь-яка організація. Крім того, за останній рік близько 40% українських підприємств зіштовхувалися з випадковим витоком даних через необережні дії персоналу, а в 43% випадків причиною витоку стали вірусні інциденти. Число витоків буде рости.

При цьому завдання забезпечення безпеки ускладнюється внаслідок все більшої розмаїтості платформ, росту популярності концепції BYOD і ставки хакерів на так звані соціальні атаки, в основі яких – обман користувачів.

Поряд з корисливими мотивами проведення атак: крадіжкою засобів з банківських рахунків або викраденням конфіденційних даних з метою наживи – будуть у ще більшому ступені проявлятися й інші стимули, наприклад бажання привернути увагу громадськості до політичної або соціальної проблеми. Однак експерти McAfee уважають, що в 2026 році хакерський рух Anonymous розгубить свій вплив і практично зникне.

1.2 Область застосування

Оскільки віддалені та гібридні схеми роботи стають все більш популярними, команди інформаційних технологій (IT) повинні мати можливість віддалено отримувати доступ до робочих столів. Таким чином вони можуть будь-коли переглядати або змінювати файли, вирішувати технічні проблеми або налаштовувати параметри. Однак ця зручність супроводжується підвищеним ризиком кіберзагроз і проблем із безпекою. Забезпечення безпечного віддаленого доступу має важливе значення для захисту вашого бізнесу від витоку даних та інших вразливостей безпеки.

Захищений віддалений доступ – це технологія, яка часто використовується IT-спеціалістами для допомоги колегам у вирішенні технічних проблем на відстані. Це також корисно для працівників, які мають увійти до захищеного пристрою, підключеного до серверу компанії в офісі, працюючи з особистого пристрою вдома. Інше використання для менеджерів, які хочуть контролювати, до чого їхні співробітники мають доступ на пристроях своєї компанії.

Якщо ви надаєте членам команди доступ до пристроїв або систем компанії, важливо вжити заходів безпеки для запобігання несанкціонованому доступу до ваших програм і даних. Системи безпечного віддаленого доступу включають найкраще програмне забезпечення для віддаленого доступу до ПК та інші стратегії, розроблені для захисту комп’ютерів і мереж від потрапляння в чужі руки.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі, 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

Dashlane

Найкращий менеджер паролів, представлений на ринку. Програма дозволяє змінити відразу кілька паролів в один клік і підтримує двофакторну автентифікацію. Крім того, є можливість використання загальних паролів усередині однієї команди.

Dashlane з'явився на ринку недавно, але за свій дружелюбний інтерфейс уже завоював симпатії представників малих і середніх бизнесів.

Вартість Dashlane – 40$ доларів у рік на один користувача, але є безкоштовна версія для одного пристрою.

Існує багато випадків, коли справа доходить до менеджерів паролів, і власники бизнесів завжди можуть вирішити, чи варто купити повну версію або використовувати безкоштовну залежно від потреб і пріоритетів. Гарна новина у тому, що навіть у безкоштовних версіях є безліч функцій.

У безкоштовній версії Dashlane ви зможете «пощупати» інтерфейс, включаючи менеджер паролів і автоматичний заповнювач форм, і спільно використовувати до 5 облікових записів.

В Dashlane можна призначити доступ до сховища аккаунтів (і до заповнювача форм) достовірній особі, якщо раптом з вами щось трапиться. Так що вам не прийдеться щораз уводити свою адресу.

Функція зміни пачки паролів в один клік на даний момент присутній тільки в Dashlane. Сюди входять більш ніж 160 найбільш популярного сайту, включаючи Facebook, Twitter, LinkedIn, Pinterest, Amazon, Dropbox і Evernote.

Дана можливість дуже придасться в тому випадку, якщо раптом ви запідозрите, що ваші облікові записи скомпрометовані.

При спільному використанні паролів ви повністю контролюєте, хто користується загальними аккаунтами (при необхідності можна змінити рівень доступу). Дана функція дуже корисна, якщо декілька співробітників вашої компанії спільно працюють під одним аккаунтом у соціальній мережі.

Автоматична авторизація працює навіть із багатосторінковими сайтами (наприклад, банківськими сайтами). Паролі, інформація у формах і замітки шифруються й зберігаються в будь-якому місці на ваш вибір (локально або в хмарі).

До речі, компанія Apple тільки недавно дозволила доступ до мобільного додатка Safari для сторонніх менеджерів пролий. Раніше користувачі в iOS повинні були копіювати й вставляти інформацію у формах вручну.

LastPass

Раніше LastPass був моїм фаворитом, хоча я й зараз продовжую користуватися цим менеджером паролів.

Існує корпоративна версія LastPass із синхронізацією через службу Active Directory, що набудовуються політиками (підключення, відключення й додавання нових користувачів) і єдиною авторизацією в популярних хмарних сервісах, включаючи Office 365, Google Apps, Salesforce, Wordpress і інших.

Підтримуючи як програмна, так і апаратна багатофакторна автентифікація, включаючи YubiKey USB keyfob, toopher і Duo Security.

Крім того, в LastPass є безкоштовна функція моніторингу кредитної історії. При роботі в сумнівних системах і мережах генеруються одноразові паролі.

На даний момент у компанії більше 10 тисяч корпоративних клієнтів, компанії деякі з яких входять у список Fortune 500.

Ціна коливається від 18$ до 24$ у рік за один користувача залежно від обсягу замовлення.

Як і Dashlane, LastPass дозволяє зберігати паролі під час авторизації на нових сайтах. Однак можуть виникнути проблеми, якщо співробітник користується сервісами, що не мають відношення до діяльності компанії.

Варто подбати про користувачів, які на робочому місці користуються особистими обліковими записами, наприклад, для доступу до банківських рахунків, щоб потім не зштовхнутися з позовами в суд. В LastPass є рішення цієї проблеми.

Співробітники можуть використовувати корпоративне й особисте сховище паролів, однак керівництво компанії має доступ тільки корпоративному сховищу.

Коли співробітник іде з компанії, всі корпоративні облікові записи можуть бути вилучені – а персональні паролі залишаються недоторканими.

Особисто я користуюся персональною версією LastPass, але інтерфейс цього менеджера досить громіздкий і незручний. Можливо в майбутньому, коли закінчиться підписка, я перейду на Dashlane.

Приміром, щоб в LastPass поміняти пачку паролів потрібно створювати й міняти пароль по черзі для кожного сайту. Весь процес досить добре автоматизований, але однаково не так зручний, як в DashLane, де можна поміняти кілька паролів одним кличем миші.

Хоча корпоративна версія LastPass найбільш сильний конкурент для DashLane.

KeePass

KeepPass – безкоштовний менеджер паролів, з відкритим вихідним кодом, інтегрований з Windows User Account Control і не є плагіном до браузера. З погляду безпечно не зовсім добре, коли найбільш важлива інформація – паролі й номери кредитних кар – зберігається в браузері, що легко може стати жертвою шкідливої програми.

KeePass – окремий додаток і не є плагіном браузера.

KeePass підтримує скрипти для PowerShell, що дозволяє створювати рішення під спеціальні потреби.

KeePass – безкоштовна альтернатива для «аскетів», не особливо причепливих до зручності й функціональності, але бажаючих серйозно підвищити свою захищеність.

1Password

Ще один менеджер паролів, що дозволяє використовувати спільні облікові записи – 1Password.

Я настійно рекомендую менеджер паролів із захищеним сховищем. Подібні додатки дозволяють вам автоматично генерувати повністю унікальні паролі для кожного сайту й автоматично заповнювати форми під час роботи на стаціонарних комп'ютерах, мобільних пристроях і в інших додатках.

Як Dashlane і LastPass, 1Password підтримує більшість браузерів, автоматично заповнює форми й має версії як для iOS, так і для Android платформ.

Blur

Особливість Blur у тому, що, крім генерації довгого унікального пароля, цей менеджер створює одноразову адресу для маскування вашої реальної електронної пошти.

Як і в інших платних менеджерах, в Blur передбачена безкоштовна версія. Розширена версія (за рахунок якої компанія одержує доходи) вартістю 40$ дозволяє генерувати одноразові номери кредитних карт із видатковими лімітами, що захищають користувача від схованих комісій і крадіжки інформації. Крім того, можна замаскувати номера телефонів.

Всім користувачам інтернету варто завести менеджер паролів. Менеджери паролів стають усе більше зручними, заощаджують час, але в той же час добре захищають конфіденційну інформацію. Поза залежністю від розміру вашого бізнесу, варто звернути увагу на ці додатки.

PasswordBox

PasswordBox недавно був придбаний компанією Intel. На даний момент повна версія безкоштовна для всіх користувачів.

Крім того, у майбутньому планується реалізація функції «True Key», що замінить майстер-пароль на біометричну ідентифікацію (наприклад, по особі).

Більшість менеджерів паролів використовують майстер-пароль, тобто пароль для розблокування доступу до всього сховища. Основна ідея полягає у тому, що набагато простіше запам'ятати один дуже довгий пароль, чим десятки й сотні паролів для кожного сайту.

Однак майстер-пароль також не зовсім зручний, особливо якщо менеджер блокується через неактивність комп'ютера або мобільного пристрою (хоча, з погляду безпечно, блокування бажане).

Всебічна підтримка вкрай важлива. Доступ до сховища паролів завжди буде однією з першорядних цілей зловмисників.

Стверджується, що PasswordBox найбільш надійний менеджер паролів. На даний момент кількість завантажень перевищує 14 мільйонів. Для порівняння: стверджується, що LastPass використовують близько 6 мільйонів користувачів.

Крім того, в PasswordBox є функція призначення достовірного користувача, що зможе одержати доступ до сховища, якщо з вами що-небудь трапиться. Також можна спільно використовувати облікові записи серед співробітників або членів родини.

RoboForm

RoboForm – самий старий серед менеджерів паролів, згаданих у цьому огляді. Перша версія RoboForm була випущена наприкінці 1999 року.

В RoboForm є одна унікальна функція, що дозволяє користувачеві одночасно авторизуватися на декількох сайтах. Дуже зручно, якщо ви щодня користуєтеся декількома сервісами. Також є портативна версія RoboForm2Go, яку можна встановити на флешку.

Як і інші менеджери паролів, RoboForm підтримує всі основні браузери й пристрої. Можна вибрати хмарний сервіс для синхронізації на всіх пристроях або зберігати дані локально. Однак в останньому випадку ви не зможете одержати доступ до сховища з інших комп'ютерів і мобільних пристроїв.

У корпоративній версії RoboForm можна набудовувати групові політики, інтеграцію з Active Directory, відновлення майстер-пароля й спільне використання облікових записів. Підтримується автоматичне створення аккаунтів для груп користувачів і облікових записів, обмежених за часом.

StickyPassword

В StickyPassword є унікальна функція для синхронізації за допомогою Wi-Fi мережі (якщо ви з якихось причин не хочете використовувати хмарний сервіс).

Підтримується робота з USB-пристроями, біометрія й автоматичне заповнення форм. StickyPassword адаптований для всіх найбільш популярних платформ, браузерів і пристроїв.

Вартість повної версії з підтримкою синхронізації через Wi-Fi усього лише 20$ у рік (найдешевший варіант серед всіх інших менеджерів із цього огляду).

У компанії затверджують, що StickyPassword використовують 2 мільйони користувачів. Крім того, на основі StickyPassword була розроблена технологія VIPRE Password Vault (компанією ThreatTrack Security).

На жаль, в StickyPassword не передбачена корпоративна версія, що робить цей менеджер паролів не особливо придатним для бізнесу.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

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

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми,

яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

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

Це мова об'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 кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

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

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

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

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

г) організувати інтерфейс користувача з метою формування та виводу на

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

За прогнозами компанії Sophos, поява нових інструментів для створення шкідливих програм і платформ для їхнього тестування істотно полегшить зусилля зловмисникам. Якщо до справжнього моменту безліч хакерських атак не наносили істотного збитку, то в 2025 році з розвитком платформ для тестування шкідливого ПЗ ситуація може кардинально змінитися й традиційні системи корпоративної безпеки перестануть захищати від нових погроз. У результаті можна чекати збільшення кількості інцидентів, коли шахраї одержують доступ до корпоративних мереж і активно ним користуються.

Згідно із прогнозами Symantec, в 2025 році атаки стануть агресивніші й будуть проводитися не тільки з метою заробітку або шпигунства, але й для демонстрації сили атакуючих, а конфлікти між державами, організаціями й окремими особами в значній мірі перейдуть у кіберпростір.

Уряду, організації або навіть окремі групи осіб почнуть використовувати кібератаки, щоб продемонструвати свою міць і заявити про себе. Мало того, якщо подібні атаки будуть фінансуватися державами, вони можуть відкрити епоху «холодної кібервійни». Ще одна очікувана тенденція – усе більше широке використання засобів спостереження з боку правоохоронних органів.

У міру поширення хмарних сервісів вони всі частіше будуть використовуватися для розміщення й поширення шкідливих програм. Хмарні ЦОД – приваблива мішень для кіберзлодіїв. На їхніх серверах розміщені більші обсяги особистих даних, які у випадку успішної атаки на провайдеру можуть цілком потрапити в руки кіберзлодіїв. Однак хмарні інфраструктури, як правило, досить надійно й професійно захищені від погроз. За прогнозами Garther, 80% інцидентів, що відбуваються в них, безпеки стануть можливі лише через

некоректні дії системних адміністраторів, що працюють у провайдерів, або помилок користувачів хмарних сервісів при керуванні послугами.

Так чи інакше, кількість IT-фахівців в Україні, що розглядають хмару як погрозу бізнесу, знизилося на 8%, а 41% опитаних бачить у хмарних сервісах можливість підсилити захист компанії шляхом передачі частини IT-інфраструктури в руки експертів.

Лише 21% вважає, що використання хмари може являти загрозу безпеки – у першу чергу мова йде про схоронність конфіденційної інформації. В 2025 році довіра до хмар буде рости, однак небезпека підстерігає й з іншої сторони.

Доступ до даних, що зберігається в хмарі, нерідко можливий з мобільних пристроїв, які найчастіше не так надійно захищені, як звичайні вузли мережі. Ризик ще вище в тому випадку, коли той самий мобільний апарат використовується як в особистих цілях, так і для рішення бізнес-завдань (за моделлю BYOD). Актуальність цієї проблеми росте, третина компаній бачить серйозну погрозу у відсутності контролю за мобільними пристроями, а 47% опитаних повідомили, що їхньої компанії почали приділяти більше увагу даному питанню. Із втратою критично важливих даних у результаті крадіжки або втрати смартфону або планшета вже зіштовхувалися 5% компаній. І це одна із причин збільшення кількості випадків втрати важливої конфіденційної інформації. Тим часом, по оцінках J'son & Partners Consulting, в III кварталі 2024 року в Україні було продано близько 3,4 млн смартфонів – у порівнянні з початком 2024 року цей ринок виріс удвічі.

Біля третини компаній надають співробітникам необмежений доступ з особистих пристроїв до корпоративної мережі і її ресурсів. Надалі кількість таких пристроїв буде тільки рости: 35% опитаних компаній планують заохочувати їхнє застосування в бізнесі, і лише 8%, навпроти, мають намір увести строгі обмеження. Характерно, що попередження появи проломів у системі безпеки замикає п'ятірку пріоритетних завдань, що стоять перед IT-фахівцями в Україні (у той час як в усьому світі ця проблема стоїть на першому місці), а контроль за

мобільними пристроями на останньому (7%). Підключення до IT-інфраструктури компанії із приналежним співробітникам незахищених пристроїв значно підвищує ризик витоку або цілеспрямованого захвата даних. Сучасні інструменти захисту, такі як окрема політика безпеки для ноутбуків і знімних носіїв, системи керування мобільними пристроями й шифрування даних, застосовують поки деякі українські організації. Поширення мобільності й концепції BYOD ведуть до зміни фундаментальних принципів доставки додатків і захисту користувальницьких даних у тому, що стосується авторизації, автентифікації, контролю доступу й функцій керування.

За останні півтора року різко зросла кількість мобільних шкідливих програм. Більше 90% з них націлені на пристрої на базі Android. Розробка шкідливих додатків для цієї ОС, швидше за все, буде тривати з тією же інтенсивністю. Крім того, кіберзлочинці продовжать інтенсивно експлуатувати уразливості в Java. Досить імовірно, що вже в 2025 році мобільні пристрої будуть використовуватися для організації DDoS-атак на корпоративні Web-сайти.

Експерти Symantec очікують росту використання мобільного рекламного ПЗ, що збирає різні відомості: інформацію про місце розташування й контактах, а також ідентифікаційні дані пристрої. Кількість програм такого типу буде збільшуватися у зв'язку з бажанням компаній підвищити доходи за рахунок мобільної реклами. Дані про місцезнаходження й характеристики пристрої можуть бути законним образом придбані агрегаторами інтернет-реклами, щоб надавати більше релевантні оголошення.

Неминуче буде розширюватися використання підроблених і крадених сертифікатів, упевнені експерти. Користувачі схильні довіряти Web-сайтам, які мають сертифікат безпеки, виданий відомим центром сертифікації, а також додаткам, підписаним за допомогою дійсного цифрового сертифіката. Однак кіберзлочинці тепер не тільки фабрикують фальшиві сертифікати для шкідливого ПЗ, але й успішно зламують сервери центрів сертифікації, щоб завіряти краденими сертифікатами свій код.

У той час як поширеність лжеантивірусів повільно сходить на ні, з'являються ще більш тверді типи погроз. В усьому світі усе ширше використовуються програми-здирники, добре відомі в Україні. В 2025 році виросло кількість троянців здирників, які шифрують дані на диску або блокують доступ до системи, а потім вимагають заплатити викуп. У майбутньому варто очікувати збільшення числа шкідливих програм такого типу.

Блокувальники вийдуть за рамки простого вимагання й будуть націлені на лякання своїх жертв, для чого стануть використовувати способи, при застосуванні яких виявиться набагато складніше відновити систему. Уже цілком реальні погрози для комп'ютерів Mac, і надалі їхнє число, швидше за все, буде тільки рости.

Забезпечення безпеки електронних платежів – ще одне актуальне завдання. У міру повсюдного впровадження технологій оплати послуг за допомогою мобільних пристроїв, останні стануть представляти ще більшу цінність. Варто очікувати появи програм, які будуть перехоплювати платіжну інформацію користувачів. Експерти Symantec очікують росту числа атак, спрямованих на крадіжку платіжних даних у соціальних мережах і прагнення змусити користувачів шляхом обману повідомити ці й інші дані підробленим соцмережам. Хоча надання нефінансової інформації може здатися справою необразливим, зловмисники торгують і обмінюються такими даними, поєднуючи їх із уже наявними, що найчастіше дозволяє їм одержувати доступ до коштовної інформації.

За прогнозами Sophos, в 2025 році збільшиться кількість критичних для безпеки помилок у настроюваннях серверів Web, зростуть обсяги шкідливого ПЗ, що важко аналізувати, спроститься виявлення експлойтів. Механізми виявлення проникнення будуть поліпшені, але з'являться нові типи атак, що використовують сервіси геопозиціонування й мобільних платежів.

Шлюзи інформаційної безпеки – спеціалізовані програмно-апаратні комплекси для захисту мережі від небажаного трафіку – всі частіше стануть

установлювати не тільки на фізичних, але й на віртуальних серверах, або їх будуть розробляти спеціально для віртуального середовища контролю трафіку між віртуальними машинами. Форм-фактор Virtual Security Appliance (VSA) дозволяє застосовувати на одній апаратній платформі кілька віртуальних засобів захисту, контролювати певне число мережних сегментів з різними правилами доступу й керуванням, захищати самі віртуалізовані інфраструктури з урахуванням їх особливостей. У міру розгортання приватних хмар переваги VSA почнуть здобувати все більшу значимість.

3.2 Розробка структурної схеми

Віддалений доступ – це простий процес, який включає наступні кроки:

1. Встановіть програмне забезпечення віддаленого доступу на пристрій, який ви використовуєте.

2. Встановіть ту саму програму на пристрій, до якого потрібно отримати віддалений доступ.

3. Увімкніть обидва пристрої.

4. Увійдіть у програмне забезпечення та виберіть пристрій, до якого потрібно отримати доступ.

5. Програмне забезпечення буде віддзеркалювати екран віддаленого пристрою, і авторизований користувач зможе отримувати доступ, обмінюватися та завантажувати файли цього пристрою.

Які є різні типи безпечних систем віддаленого доступу?

Існує кілька варіантів безпечного доступу до пристрою чи програми з іншого місця. Деякі рішення можна використовувати одночасно, залежно від необхідного рівня доступу та безпеки.

Вибирайте рішення, які прості для роботи з вашим IT-відділом і співробітниками [і можуть] легко адаптуватися до потреб їхніх розподілених команд.

Ось деякі з найпоширеніших типів захищеного віддаленого доступу до ПК :

– Віртуальна приватна мережа (VPN): VPN додає рівень безпеки для будь-якого пристрою, що працює віддалено від домашньої системи. Він забезпечує безпечний доступ до Інтернету шляхом маршрутизації з'єднань через зашифрований сервер. Тоді користувачі можуть працювати через будь-яку домашню або громадську мережу Wi-Fi для своїх конфіденційних бізнес-потреб без обмежень і страху злому.

– Безпека кінцевої точки: система безпеки кінцевої точки захищає різні пристрої компанії в мережі. Ці пристрої можуть включати мобільні пристрої, ноутбуки, настільні комп'ютери та сервери. Як правило, на кожному пристрої встановлюють найкраще антивірусне програмне забезпечення та брандмауери, щоб захистити дані.

– Доступ до мережі з нульовою довірою (ZTNA): Цей тип системи забезпечує безпечний віддалений доступ до програм і послуг на основі політики компанії. Система ZTNA за замовчуванням забороняє доступ, на відміну від VPN, яка зазвичай надає користувачам відкритий доступ із відповідними обліковими даними. Тип даних, системи та рівень доступу користувача визначаються налаштуваннями системи компанії. Підприємства можуть застосовувати свої політики на основі розташування або типу пристрою, що дозволяє деяким організаціям обмежувати доступ до незнайомих мереж.

– Контроль доступу до мережі: контроль доступу до мережі зазвичай визначається менеджерами організації або керівником відділу користувача. Це дозволяє їм обмежити доступ зовнішнім користувачам і користувачам всередині організації, яким він не потрібен.

– Єдиний вхід (SSO): технологія SSO – це процес автентифікації користувача, який надає користувачеві доступ до кількох пристроїв за допомогою одного набору облікових даних. Програмне забезпечення дозволяє користувачам входити до свого комп'ютера, електронної пошти, мобільного пристрою та будь-

якої іншої системи чи пристрою за допомогою тих самих облікових даних. Система єдиного входу дозволяє користувачам швидко входити в систему, дозволяючи організації керувати автентифікацією користувачів і доступом.

– Керування привілейованим доступом (PAM): PAM – це набір інструментів, які менеджери використовують для захисту та моніторингу доступу до конфіденційних системних даних шляхом відстеження активності користувачів у різних облікових записах. PAM допомагає забезпечити безпеку, дозволяючи адміністраторам контролювати доступ користувачів і виявляти будь-яку незвичну поведінку.

Основні компоненти комплексної системи включають зашифровані з'єднання, наприклад через VPN, безпеку кінцевих точок для захисту пристроїв і ZTNA, який аутентифікує користувачів для надання доступу в кожній точці дотику. Усі ці елементи працюють разом, щоб мінімізувати вразливі місця, зберігаючи безперешкодний доступ для співробітників.

Чому безпечний віддалений доступ важливий?

Хегде підкреслив важливість безпечного віддаленого доступу для будь-якої організації з компонентами віддаленої роботи. «Захист віддаленого доступу більше не є обов'язковим; це стратегічний імператив для будь-якого перспективного бізнесу», – сказав Хегде.

Коли роботодавці вживають належних заходів безпеки для захисту своїх бізнес-даних у віддаленому робочому середовищі, вони отримують наступні переваги.

1. Безпечний віддалений доступ забезпечує безпечну віддалену роботу

Рішення безпечного віддаленого доступу надають авторизованим користувачам доступ до мережі вашої компанії на будь-якому сумісному пристрої. Співробітники мають безперебійний доступ до всіх необхідних даних і файлів, а дані організації залишаються в безпеці.

Крім того, IT та інженерні експерти можуть дистанційно вирішувати проблеми користувачів. Щоразу, коли виникає технічна проблема,

спеціалізований працівник може віддалено отримати доступ до пристрою користувача та допомогти йому швидко та ефективно знайти рішення.

2. Безпечний віддалений доступ забезпечує безпечний перегляд Інтернету

У багатьох сучасних компаніях велика частина щоденних робочих процесів їхніх співробітників відбувається в Інтернеті. Безпечна система віддаленого доступу захищає ваших співробітників від веб-загроз. Такі небезпеки включають фішингові загрози, атаки програм-вимагачів і зловмисне програмне забезпечення, коли вони ввійшли в мережу вашої компанії. Без захисту ці кіберінциденти можуть призвести до несанкціонованого доступу та використання як бізнес-даних компанії, так і особистих даних співробітників.

3. Безпечний віддалений доступ захищає кілька кінцевих точок

Сучасні працівники часто використовують кілька пристроїв для виконання робочих завдань. Вони можуть відповідати на електронні листи зі смартфона, вводити дані на своїх планшетах і входити на відеозустрічі на своїх ноутбуках. Кожен із цих пристроїв, відомих як кінцеві точки, становить потенційну загрозу безпеці, якщо вони не захищені належним чином.

Захист віддаленого доступу важливий, оскільки все більше співробітників вирішують працювати віддалено, а захист традиційного корпоративного периметра більше не існує. Рішення безпечного віддаленого доступу може захистити всі пристрої, підключені до мережі та систем вашої компанії. Ця практика гарантує, що конфіденційна інформація залишається в безпеці, незалежно від того, де та як співробітники входять у систему.

4. Безпечний віддалений доступ підвищує обізнаність співробітників щодо безпеки

План кібербезпеки будь-якої організації має включати навчання працівників передовим практикам кібербезпеки та тому, як їхні дії можуть вплинути на безпеку даних компанії та клієнтів. Обізнаність співробітників є ще більш важливою для компаній з компонентом віддаленої роботи.

Оскільки рішення для безпечного віддаленого доступу вимагають від співробітників дотримання протоколів безпеки, вони за своєю суттю навчають членів команди безпечним цифровим практикам. «Підвищена обізнаність співробітників є важливою, оскільки нам все ще потрібно керувати безпекою наших окремих пристроїв як фізично, так і з усвідомленням того, що ми можемо створювати ненавмисний ризик, коли натискаємо шкідливе посилання або завантажуємо неавторизоване програмне забезпечення», – пояснив Басу.

Захист від кіберзлочинності є надзвичайно важливою проблемою. Згідно зі звітом CompTIA про стан кібербезпеки за 2025 рік, майже 60 відсотків компаній повідомляють, що інциденти з кібербезпекою за останній рік мали помірний або серйозний вплив – негативно вплинули як на фінанси, так і на продуктивність.

Які недоліки безпечного віддаленого доступу?

Захищений віддалений доступ – або віддалений доступ, який не є достатньо безпечним – також може спричинити кілька проблем для організації, зокрема такі недоліки:

1. Безпечний віддалений доступ створює потенційну вразливість системи безпеки

Коли ви налаштовуєте безпечну систему віддаленого доступу, слово «безпечний» є ключовим. Без ретельного впровадження та дотримання найкращих практик запровадження нової системи може відкрити двері для ризиків безпеки кожного зовнішнього пристрою, який підключається до системи компанії.

За своєю конструкцією безпечні системи віддаленого доступу мають шифрування даних, журнали активності для моніторингу використання співробітниками та кілька рівнів дозволів. Проте забезпечити доступ лише авторизованим користувачам може бути складно.

2. Безпечний віддалений доступ має технічні вимоги

Для успішного впровадження програмного забезпечення віддаленого

доступу потрібна надійна технічна інфраструктура; налаштування включає безпечне та надійне підключення до Інтернету, як-от широкосмугове з'єднання для бізнесу з достатньою швидкістю Інтернету. Без належного технічного налаштування ваші співробітники не зможуть ефективно виконувати свою роботу.

Хоча ефективні безпечні системи віддаленого доступу вимагають значної технічної інфраструктури та обслуговування. Погані конфігурації відкривають слабкі місця в системах, постійно потрібні оновлення для запобігання кіберзагрозам, що розвиваються, і досить часто існує операційна проблема між комфортом для користувача та безпекою.

Крім того, постачальники рішень можуть пред'являти інші технічні вимоги, такі як конкретні специфікації хост-комп'ютера, які компанії повинні виконати для впровадження технології.

3. Безпечний віддалений доступ вимагає постійного обслуговування

Системи віддаленого доступу вимагають регулярного обслуговування для вирішення таких проблем, як системні збої або необхідні оновлення. Це означає, що компанії повинні мати IT-фахівця або технічно кваліфікованого працівника в режимі очікування для вирішення системних проблем – дистанційно або особисто, залежно від проблеми – у міру їх виникнення.

Окрім операційних витрат і витрат на персонал, пов'язаних із обслуговуванням програмного забезпечення, збої в системі, наприклад збої, спричинені втратою живлення, можуть вплинути на щоденні робочі процеси та призвести до втрати продуктивності.

S/MIME

Для забезпечення безпечного доступу з мобільного пристрою до ресурсів корпоративної мереж запропоновано використовувати S/MIME.

Зазвичай відправник повідомлення шифрує його один раз за допомогою симетричного ключа. Потім відправник повинен зашифрувати симетричний ключ окремо, використовуючи відкритий ключ того адресата, якому він направляє своє

повідомлення. Якщо кількість адресатів велика, виникає необхідність делегувати функції шифрування довіреному серверу. Такий сервер називають агентом списку розсилання (Mail List Agent – MLA). Якщо є агент MLA, то відправник може послати зашифроване повідомлення йому, а той, у свою чергу, після виконання відповідних операцій виконує розсилання повідомлення всім адресатам зі списку відправника. При цьому MLA не одержує доступ до ключа шифрування повідомлення й не розшифровує повідомлення, що пересилається.

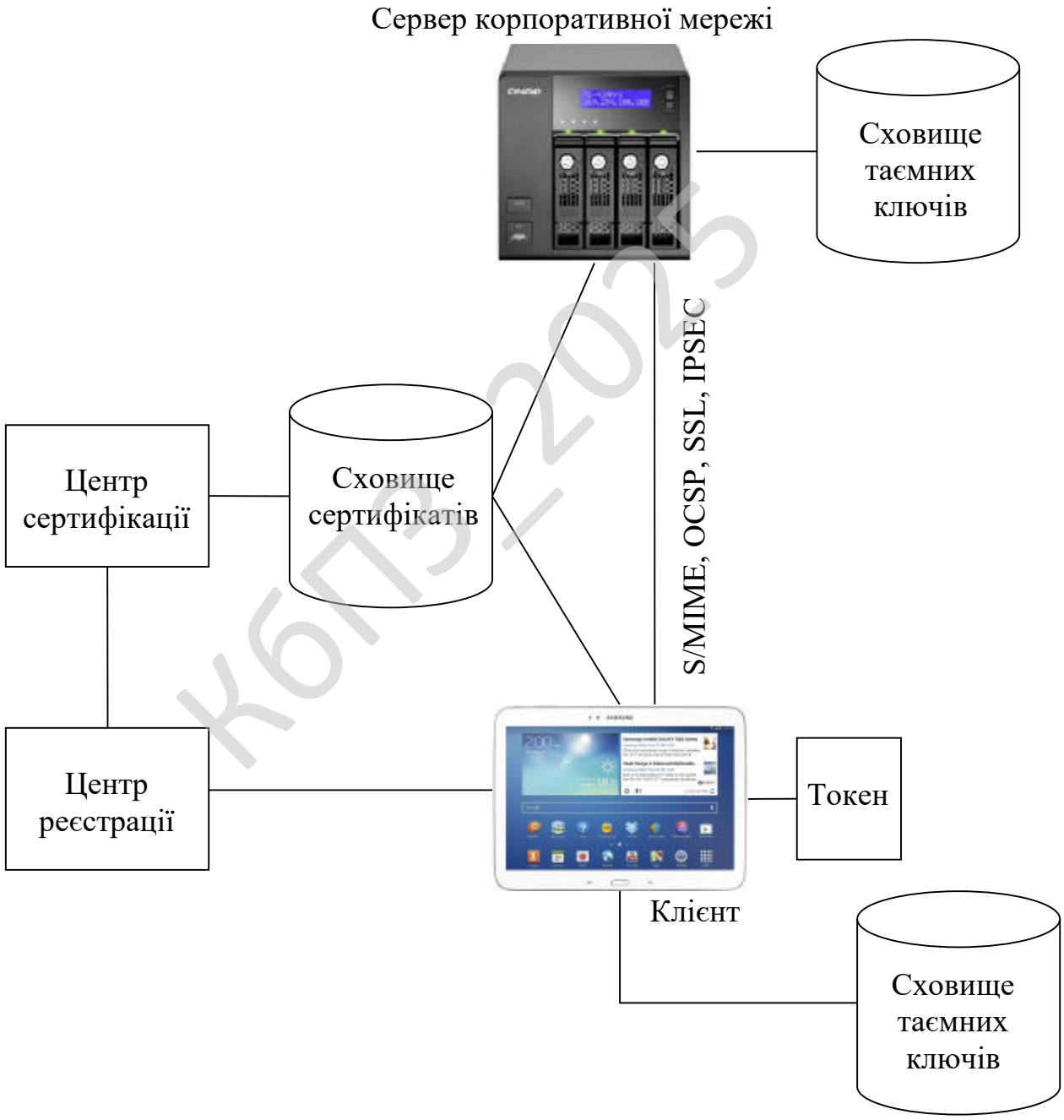

Рисунок 3.1 – Структурна схема системи

Якщо адресати повідомлення територіально розподілені (наприклад, перебувають на різних континентах), то відправник посилає зашифроване повідомлення головному агентові MLA, головний агент пересилає його регіональним агентам MLA, і, нарешті, кожний регіональний агент доставляє зашифроване повідомлення локальним одержувачам [7]. У цьому випадку повідомлення бере участь у кожній міжконтинентальній комунікації тільки один раз. Правда, якщо в списки розсилання кожного регіонального MLA включені адреси всіх інших регіональних агентів, може відбуватися багаторазове пересилання повідомлення по колу. Для виявлення циклів аналізується атрибут "історія поширення списку розсилання", що втримується в зовнішньому конверті повідомлення. Коли агент MLA одержує повідомлення, то перевіряє історію поширення, щоб визначити, чи не оброблялося вже дане повідомлення. Якщо повідомлення оброблялося, воно просто віддаляється. Зовнішній конверт із цифровим підписом забезпечує цілісність історії поширення списку розсилання.

S/MIME v2 забезпечує тільки транспортування ключів шифрування повідомлень, а S/MIME v3 додатково підтримує механізми узгодження ключів і зовнішнього поширення симетричних ключів шифрування ключів.

3.3 Розробка функціональної схеми

S/MIME – відмінний приклад гібридного рішення шифрування, у якому об'єднані достоїнства симетричного й асиметричного шифрів і функцій гешування. На рисунку 3.2 показана функціональна схема системи.

Рисунок 3.2 – Функціональна схема використання S/MIME

Відправник хоче послати безпечне повідомлення (з гарантованими конфіденційністю, цілісністю, дійсністю даних) користувачеві Одержувач. Обмін S/MIME складається з наступних кроків:

1. Відправник створює цифрову сигнатуру для повідомлення з використанням свого приватного ключа.

2. Відправник використовує єдиний ключ симетричного шифрування для шифрування повідомлення.

3. Щоб сформувати безпечний канал, у якому буде захищена конфіденційність ключа шифрування при його передачі через загальнодоступний канал зв'язку, Відправник використовує відкритий ключ Одержувача (із сертифіката Одержувача), щоб зашифрувати ключ шифрування. Результат цього етапу – захищений ящик (lockbox), у якому втримується зашифрована копія ключа шифрування.

4. Одержувач використовує свій приватний ключ, щоб розшифрувати захищений ящик. У процесі дешифрування формується єдиний ключ шифрування.

5. Одержувач розшифровує повідомлення за допомогою єдиного ключа шифрування. Тепер Одержувач може прочитати повідомлення.

6. Одержувач використовує відкритий ключ Відправника для перевірки дійсності й цілісності повідомлення, засвідчуючи цифрову сигнатуру, що є в сертифікаті Відправника. S/MIME забезпечує стійке наскрізне шифрування поштових повідомлень. Це значить, що при використанні S/MIME повідомлення зашифровані не тільки на етапі пересилання, але й при зберіганні в локальній особистій папці й поштовій скриньці.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.
– Підсистема ключів.
– Створення ключа для шифрування.
– Налаштування підсистеми ключів.
– Моніторинг ресурсів корпоративної мережі.
– Отримання листа.
– Перегляд листів.
– Дешифрування листа.
– Формування листа.
– Шифрування листа.
– Відправка листа.

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

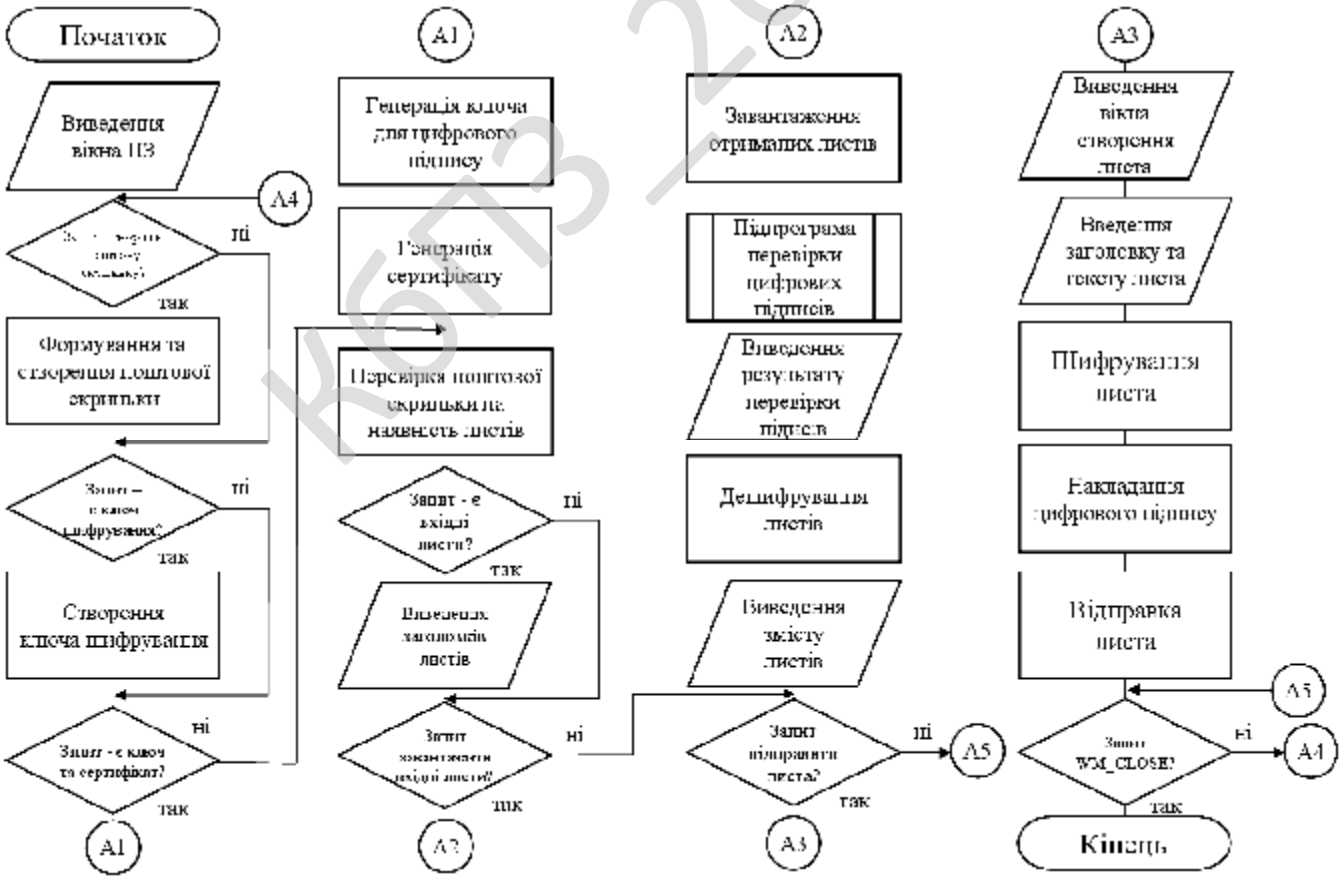

Рисунок 4.1 – Блок схема основної програми

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

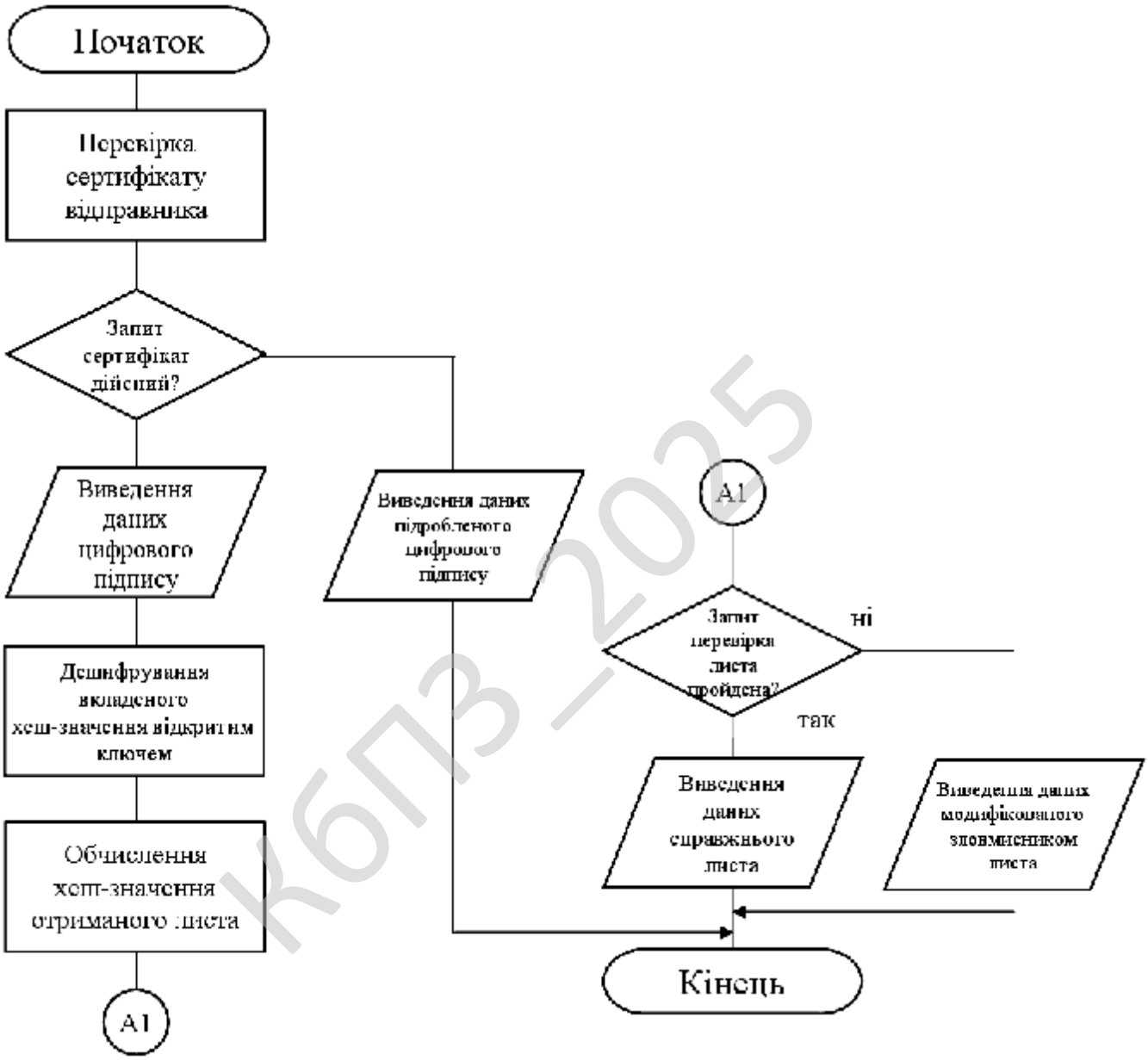

Рисунок 4.2 – Блок схема підпрограми

Опис розробленого алгоритму доступу та шифрування ресурсів корпоративної мережі

Алгоритм RSA являє собою блоковий алгоритм шифрування, де зашифруванні й незашифруванні дані є цілими між 0 і n-1 для деякого n.

Дані шифруються блоками, кожний блок розглядається як число, менше деякого числа n. Шифрування і дешифрування мають наступний вигляд для деякого незашифрованого блоку M та зашифрованого блоку C:

$$C = M^e \pmod n, \tag{4.1}$$

$$M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n. \tag{4.2}$$

Як відправник, так і одержувач повинні знати значення n. Відправник знає значення e, одержувач знає значення d. Таким чином, відкритий ключ є KU = {e, n} і закритий ключ є KR = {d, n}. При цьому повинні виконуватися наступні умови:

1. Можливість знайти значення e, d і n такі, що $M^{ed} = M \bmod n$ для всіх $M < n$.

2. Відносна легкість обчислення M^e й C^d для всіх значень $M < n$.

3. Неможливість визначити d, знаючи e та n.

Опис створення ключів.

Вибрати прості p та q.

Обчислити $n = p \cdot q$.

Обчислити функцію Ейлера $\Phi(n) = (p - 1) * (q - 1)$.

Вибрати e взаємно просте з $\Phi(n)$: НСД $(\Phi(n), e) = 1$; $1 < e < \Phi(n)$.

Обчислити d: $d \cdot e \equiv 1 \bmod \Phi(n)$.

Відкритий ключ KU = {e, n} – публікується

Закритий ключ KR = {d, n} – тримається в таємниці

Примітка: p, q тримаються в таємниці, якщо вони стають відомі зловмиснику, то протокол дискредитовано.

Шифрування даних.

Незашифрований текст: $M < n$.

Зашифрований текст: $C = M^e \pmod n$.

Дешифрування даних.

Зашифрування текст: $C < n$.

Незашифрування текст: $M = C^d \pmod n$.

Обчислювальні аспекти.

Розглянемо складність обчислень в алгоритмі RSA при створенні ключів і при шифруванні/дешифруванні.

Шифрування/дешифрування даних. Як шифрування, так і дешифрування включають піднесення цілого числа в цілий ступінь по модулю n. При цьому проміжні значення будуть величезними. Для того, щоб частково цього уникнути (тобто, щоб проміжні результати не виходили за границі n), використовується наступна властивість модульної арифметики:

$$(a \cdot b) \bmod n = (a \bmod n) \cdot (b \bmod n). \quad (4.3)$$

Також алгоритм можна оптимізувати за допомогою ефективного використання показника ступеня, тому що у випадку RSA показники ступеня дуже великі. Припустимо, що необхідно обчислити x^{16}. Прямий підхід вимагає 15 множень. Однак можна домогтися того ж кінцевого результату за допомогою тільки чотирьох множень, якщо використовувати квадрат кожного проміжного результату: x^2, x^4, x^8, x^{16}.

Створення ключів. Створення ключів включає наступні задачі:

1. Визначити два простих числа p та q.

2. Вибрати e й обчислити d.

Насамперед, розглянемо проблеми, пов'язані з вибором p та q. Тому що значення $n = p \cdot q$ буде відомо будь-якому потенційному зловмисникові, для запобігання розкриття p та q ці прості числа повинні бути обрані з досить великої множини, тобто p та q повинні бути великими числами.

З іншого боку, метод, що використовується для пошуку великого простого числа, повинен бути досить ефективним. У наш час не відомі алгоритми, які створюють довільно великі прості числа.

Процедура, що використовується для цього, вибирає випадкове непарне число з необхідного діапазону й перевіряє, чи є воно простим. Якщо число не є простим, то знову вибирається випадкове число доти, поки не буде знайдено просте.

Були розроблені різні тести для визначення того, чи є число простим. Це тести імовірнісні, тобто тест показує, що дане число ймовірно є простим. Незважаючи на це, вони можуть виконуватися таким чином, що зроблять імовірність близькою до 1. Якщо n "провалює" тест, то воно не є простим. Якщо n "пропускає" тест, то n може як бути, так і не бути простим. Якщо n пропускає багато таких тестів, то можна з високим ступенем вірогідності сказати, що n є простим. Це досить довга процедура, але вона виконується відносно рідко: тільки при створенні нової пари (KU, KR).

На складність обчислень також впливає те, яка кількість чисел буде відкинута перед тим, як буде знайдене просте число. Результат з теорії чисел, відомий як теорема простого числа, говорить, що простих чисел, розташованих біля n у середньому одне на кожні ln (n) чисел. Таким чином, у середньому потрібно перевірити послідовність із ln (n) цілих, перш ніж буде знайдене просте число. Тому що всі парні числа можуть бути відкинуті без перевірки, то потрібно виконати приблизно ln (n)/2 перевірок. Наприклад, якщо просте число шукається в діапазоні величин 2^{200}, то необхідно виконати біля ln (2^{200}) / 2 = 70 перевірок.

Вибравши прості числа p та q, далі необхідно вибрати значення e так, щоб НСД(Φ(n), e) = 1 і обчислити значення d, $d = e^{-1} \bmod \Phi(n)$. Існує єдиний алгоритм, який називається розширеним алгоритмом Евкліда, що за фіксований час обчислює найбільший спільний дільник двох цілих і якщо цей спільний дільник дорівнює одиниці, визначає інверсне значення одного за модулем іншого. Таким чином, процедура полягає в генерації серії випадкових чисел і перевірці кожного відносно Φ(n) доти, поки не буде знайдене число, взаємопросте з Φ(n). Виникає питання, як багато випадкових чисел доведеться перевірити доти, поки не знайдеться потрібне число, що буде взаємопростим з Φ(n). Результати показують, що ймовірність того, що два випадкових числа є взаємопростими, дорівнює 0.6.

Опис алгоритму геш-функції

Криптографічна геш-функція SHA-1 (SHA – Secure Hash Algorithm) призначена для обчислення з повідомленням довжиною до 2^{64} бітів його ущільненого представлення довжиною 160 бітів.

Геш-функція SHA-1 опрацьовує вхідне повідомлення послідовними блоками по 512 бітів, тому якщо довжина повідомлення не є кратною 512 бітам, здійснюється його вирівнювання на границю блока.

Для цього в кінець повідомлення дописується біт 1, а потім дописуються біти 0; це дописування припиняється, коли до кінця блока залишається 64 біта. В останні 64 біта блока записується початкова довжина повідомлення (до вирівнювання); при цьому найменш значущі біти довжини повинні знаходитись у кінці блока.

В процесі обчислення геш-функції застосовуються дві групи по п'ять 32-бітних змінних – A, B, C, D, E і H_0, H_1, H_2, H_3, H_4, а також вісімдесят 32-бітних змінних $W_0,\ldots,W_{79}$. Крім цього, в процесі роботи використовується змінна T довжиною 32 біта.

Нехай вирівняне повідомлення складається з 512-бітних блоків $M_1,\ldots,M_n$. Процес обробки кожного блока M_i включає 80 ітерацій. Перед початком обробки блоків здійснюється ініціалізація змінних H_0, H_1, H_2, H_3, H_4: H_0=67452301; H_1=EFCDAB89; H_2=98BADCFE; H_3=10325476;H_4=C3D2E1F0.

Обробка блока M_i полягає в наступному:

1. Блок M_i послідовно розбивається на шістнадцять 32-бітних частин, які записуються в змінні $W_0,\ldots,W_{15}$.

2. Для i від 16 до 79 виконуються обчислення:

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) <<<1 \qquad (4.3)$$

3. Здійснюються присвоювання:

$$A = H_0,\ B = H_1,\ C = H_2,\ D = H_3,\ E = H_4. \qquad (4.4)$$

Для i від 0 до 79 виконуються наступні обчислення:

$$T = (A<<<5) \oplus_{32} f(B,C,D) \oplus_{32} E \oplus_{32} W_i \oplus_{32} K, \quad (4.5)$$

$$E = D,\ D = C,\ C = B<<<30,\ B = A,\ A = T, \quad (4.6)$$

$$T = (A<<<5) \oplus_{32} f(B,C,D) \oplus_{32} E \oplus_{32} W_i \oplus_{32} K, \quad (4.7)$$

$$E = D,\ D = C,\ C = B<<<30,\ B = A,\ A = T. \quad (4.8)$$

Здійснюються переприсвоювання:

$$H_0 = H_0 \oplus_{32} A,\ H_1 = H_1 \oplus_{32} B,\ H_2 = H_2 \oplus_{32} C,$$

$$H_3 = H_3 \oplus_{32} D,\ H_4 = H_4 \oplus_{32} E. \quad 4.9)$$

В процесі обробки блоків використовується функція *f*, причому в залежності від номера *i* ітерації ця функція має різний вигляд. Можливі типи функції *f*, наведені нижче.

$$f(B,C,D) = (B\&C) \vee ((\oplus B)\&D) \quad (4.10)$$

$$f(B,C,D) = B \oplus C \oplus D \quad (4.11)$$

$$f(B,C,D) = (B\&C) \vee (B\&D) \vee (C\&D) \quad (4.12)$$

В ітераціях 0-19 використовується функція *f*,типу (4.9), в ітераціях 20-39, 60-79 – функція *f*, типу (4.10), в ітераціях 40-59 – функція *f*, типу (4.11). Крім функції *f*, від номера ітерації *i* також залежить значення константи *K*. Можливі значення цієї константи наведені нижче.

1) *K* = 5A827999 (ціла частина числа [$2^{30} \times 2^{1/2}$]);

2) *K* = 6ED9EBA1 (ціла частина числа [$2^{30} \times 3^{1/2}$]);

3) *K* = 8F1BBCDC (ціла частина числа [$2^{30} \times 5^{1/2}$]);

4) *K* = CA62C1D6 (ціла частина числа [$2^{30} \times 10^{1/2}$]).

В ітераціях 0-19 використовується константа *K* типу 1, в ітераціях 20-39 – константа *K* типу 2, в ітераціях 40-59 – константа *K* типу 3, в ітераціях 60-79 – константа *K* типу 4.

Після обробки останнього блока повідомлення – блока M_n здійснюється конкатенація змінних H_0, H_1, H_2, H_3, H_4, і таким чином отримується 160-бітний результат обчислення геш-функції від повідомлення.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Khufu. Khufu – це 64-бітовий блоковий шифр. 64-бітовий відкритий тест спочатку розщеплюється на дві 32-бітові половини, *L* і *R.* Над обома половинами й певними частинами ключа виконується операція XOR. Потім, аналогічно DES, результати проходять деяку послідовність раундів. У кожному раунді молодший значущий байт *L* використовується як вхід S-блоку. У кожного S-блоку 8 вхідних біт і 32 вихідних біта. Далі обраний в S-блоці 32-бітовий елемент піддається операції XOR з *R.* Потім *L* циклічно зрушується на число, кратним восьми биткам, *L* і *R* міняються місцями, і раунд завершується. Сам S-блок не статичний, він міняється кожні вісім раундів. Нарешті, по закінченні останнього раунду, над *L* і *R* виконується операція XOR з іншими частинами ключа, і половини поєднуються, утворюючи блок шифртексту.

Хоча частини ключа використовуються для операції XOR із блоком шифрування на початку й кінці виконання алгоритму, головне призначення ключа – генерація S-блоків. Ці S-блоки секретні, по суті, це частина ключа. Повний розмір ключа алгоритму Khufu дорівнює 512 біт (64 байт), алгоритм надає спосіб генерації S-блоків по ключу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

– Розділ локальних даних: SD карта; Внутрішній накопичувач.

– Розділ віддалених даних – ресурси підключеної корпоративної мережі.

– Розділ представлення обраних даних.

– Розділ вспливаючого меню (показується при довгому натисканні на сенсорному екрані) з наступним функціоналом: Лист; Пошта Шифрування; Ключі; Параметри ; Довідка.

Для генерації сертифікату і ключа підпису та створення ключа шифрування треба вибрати пункт «Параметри». Для створення листа необхідно вибрати пункт «Лист».

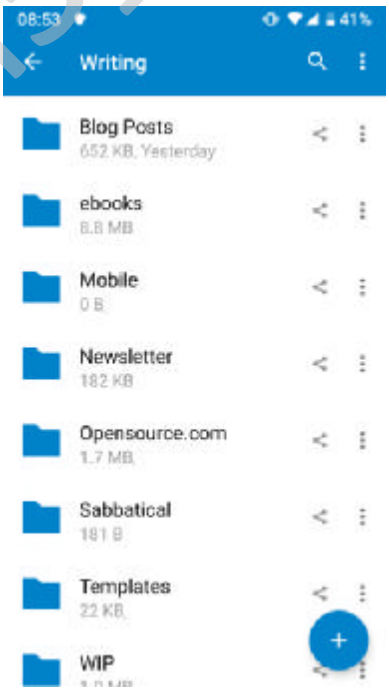

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

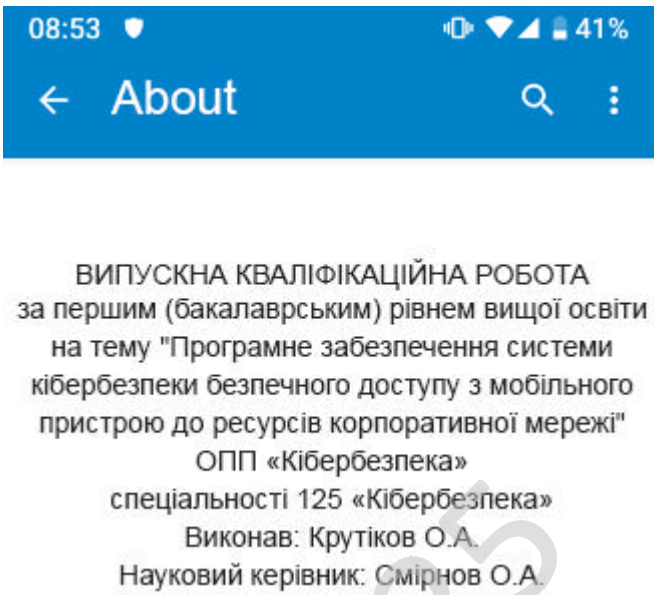

Рисунок 5.2 – Авторське право

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

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

– Досліджена система безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

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

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

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

системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

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

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Khufu.

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. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

11. 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.

12. 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.

13. 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.

14. 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.

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

16. 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

17. 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.

18. 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.

19. 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,

20. 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.

21. 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.

22. 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

23. 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.

24. 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.

25. 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.*

26. 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.

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

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі.

4 Джерела розробки

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

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

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

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 50 аркушів.

8 Етапи розробки

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

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

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

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

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 7.06.2025 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

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки безпечного доступу з мобільного пристрою до ресурсів корпоративної мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 18

Літера: РП

Кропивницький – 2025 року

Основна програма

```
# Імпорт необхідних бібліотек
import hashlib
import hmac
import random
import string
import time
import base64
import json
import socket
import ssl
from datetime import datetime

# Ініціалізація глобальних змінних для емуляції конфігурації
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 8443
BUFFER_SIZE = 4096
TIMEOUT = 5

# Ініціалізація секретного ключа для HMAC
GLOBAL_SECRET_KEY = "SuperSecretKeyForHMAC12345"

# Клас Logger для збереження журналу подій системи
class Logger:
# Конструктор класу Logger
    def __init__(self):
# Ініціалізація списку журналу
        self.logs = []
# Метод для додавання запису в журнал
    def log(self, message):
# Отримання поточного часу у форматі
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
# Формування запису журналу
        entry = f"{timestamp} - {message}"
# Додавання запису до списку
        self.logs.append(entry)
# Метод для виведення всіх записів журналу
    def show_logs(self):
# Ітерування по кожному запису
        for log in self.logs:
            print(log)
# Метод для збереження журналу у файл
    def save_logs(self, filename="system_logs.txt"):
# Відкриття файлу для запису
        with open(filename, "w") as f:
# Запис кожного запису в файл
            for log in self.logs:
                f.write(log + "\n")

# Клас для управління аутентифікацією користувачів
class AuthenticationManager:
# Конструктор класу AuthenticationManager
    def __init__(self, logger):
# Ініціалізація бази даних користувачів у вигляді словника
        self.users_db = {}
# Збереження об’єкта журналу
        self.logger = logger
# Метод для реєстрації користувача
    def register_user(self, username, password):
# Перевірка чи існує користувач
        if username in self.users_db:
```

```
            self.logger.log(f"Реєстрація неуспішна: Користувач {username} вже
існує")
            return False
# Хешування пароля з використанням SHA-256
        hashed_password = hashlib.sha256(password.encode()).hexdigest()
# Збереження користувача у базі даних
        self.users_db[username] = hashed_password
# Запис журналу про успішну реєстрацію
        self.logger.log(f"Користувач {username} успішно зареєстрований")
        return True
# Метод для автентифікації користувача
    def authenticate_user(self, username, password):
# Перевірка наявності користувача
        if username not in self.users_db:
            self.logger.log(f"Аутентифікація неуспішна: Користувача {username}
не знайдено")
            return False
# Отримання збереженого хеша пароля
        stored_hash = self.users_db[username]
# Розрахунок хеша для введеного пароля
        current_hash = hashlib.sha256(password.encode()).hexdigest()
# Порівняння збереженого та введеного хеша
        if hmac.compare_digest(stored_hash, current_hash):
            self.logger.log(f"Користувач {username} успішно автентифікований")
            return True
        else:
            self.logger.log(f"Аутентифікація неуспішна для користувача
{username}")
            return False

# Клас для управління криптографічними операціями
class CryptoManager:
# Конструктор класу CryptoManager
    def __init__(self, secret_key):
# Збереження секретного ключа для HMAC
        self.secret_key = secret_key.encode()
# Метод для генерації HMAC-підпису
    def generate_signature(self, message):
# Створення HMAC підпису з використанням SHA-256
        signature = hmac.new(self.secret_key, message.encode(),
hashlib.sha256).hexdigest()
# Повернення підпису
        return signature
# Метод для перевірки HMAC-підпису
    def verify_signature(self, message, signature):
# Порівняння згенерованого підпису з отриманим
        expected_signature = self.generate_signature(message)
        return hmac.compare_digest(expected_signature, signature)

# Клас для моделювання мобільного клієнта
class MobileClient:
# Конструктор класу MobileClient
    def __init__(self, username, password, auth_manager, crypto_manager,
logger):
# Збереження імені користувача
        self.username = username
# Збереження пароля користувача
        self.password = password
# Збереження посилання на менеджер аутентифікації
        self.auth_manager = auth_manager
# Збереження посилання на менеджер криптографії
        self.crypto_manager = crypto_manager
# Збереження посилання на журнал подій
```

```
        self.logger = logger
# Змінна для зберігання сесійного токена
        self.session_token = None
# Метод для ініціалізації з’єднання з сервером
    def connect_to_server(self):
# Спроба встановлення сокетного з’єднання
        try:
            context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)
            sock = socket.create_connection((SERVER_HOST, SERVER_PORT),
timeout=TIMEOUT)
            self.ssl_sock = context.wrap_socket(sock,
server_hostname=SERVER_HOST)
            self.logger.log("Мобільний клієнт успішно підключився до сервера")
        except Exception as e:
            self.logger.log(f"Помилка підключення до сервера: {str(e)}")
            self.ssl_sock = None
# Метод для закриття з’єднання з сервером
    def disconnect(self):
# Закриття SSL сокета
        if self.ssl_sock:
            self.ssl_sock.close()
            self.logger.log("Мобільний клієнт розірвав з’єднання з сервером")
# Метод для реєстрації на сервері
    def register(self):
# Відправлення запиту на реєстрацію
        if self.ssl_sock:
            registration_data = {
                "action": "register",
                "username": self.username,
                "password": self.password
            }
# Серіалізація даних у формат JSON
            message = json.dumps(registration_data)
# Генерація підпису для повідомлення
            signature = self.crypto_manager.generate_signature(message)
# Формування запиту з підписом
            request = json.dumps({"payload": message, "signature": signature})
# Відправлення запиту на сервер
            self.ssl_sock.send(request.encode())
# Отримання відповіді від сервера
            response = self.ssl_sock.recv(BUFFER_SIZE).decode()
            self.logger.log(f"Отримано відповідь на реєстрацію: {response}")
# Метод для аутентифікації на сервері
    def login(self):
# Відправлення запиту на аутентифікацію
        if self.ssl_sock:
            login_data = {
                "action": "login",
                "username": self.username,
                "password": self.password
            }
# Серіалізація даних у формат JSON
            message = json.dumps(login_data)
# Генерація підпису для повідомлення
            signature = self.crypto_manager.generate_signature(message)
# Формування запиту з підписом
            request = json.dumps({"payload": message, "signature": signature})
# Відправлення запиту на сервер
            self.ssl_sock.send(request.encode())
# Отримання відповіді від сервера
            response = self.ssl_sock.recv(BUFFER_SIZE).decode()
# Обробка відповіді та встановлення сесійного токена
            try:
```

```
                data = json.loads(response)
                if data.get("status") == "success":
                    self.session_token = data.get("session_token")
                    self.logger.log("Аутентифікація пройшла успішно, сесійний
токен отримано")
                else:
                    self.logger.log("Аутентифікація не вдалася")
            except Exception as e:
                self.logger.log(f"Помилка обробки відповіді: {str(e)}")
# Метод для отримання захищених ресурсів з корпоративної мережі
    def request_resource(self, resource_name):
# Перевірка наявності сесійного токена
        if not self.session_token:
            self.logger.log("Запит не може бути виконаний: відсутній сесійний
токен")
            return
# Формування запиту на отримання ресурсу
        resource_request = {
            "action": "get_resource",
            "resource": resource_name,
            "session_token": self.session_token
        }
# Серіалізація запиту у формат JSON
        message = json.dumps(resource_request)
# Генерація підпису для запиту
        signature = self.crypto_manager.generate_signature(message)
# Формування остаточного запиту з підписом
        request = json.dumps({"payload": message, "signature": signature})
# Відправлення запиту на сервер
        self.ssl_sock.send(request.encode())
# Отримання відповіді від сервера
        response = self.ssl_sock.recv(BUFFER_SIZE).decode()
        self.logger.log(f"Отримано відповідь для ресурсу {resource_name}:
{response}")

# Клас для симуляції корпоративного сервера
class CorporateServer:
# Конструктор класу CorporateServer
    def __init__(self, host, port, auth_manager, crypto_manager, logger):
# Збереження хоста сервера
        self.host = host
# Збереження порту сервера
        self.port = port
# Збереження менеджера аутентифікації
        self.auth_manager = auth_manager
# Збереження менеджера криптографії
        self.crypto_manager = crypto_manager
# Збереження журналу подій
        self.logger = logger
# Ініціалізація словника для сесійних токенів
        self.sessions = {}
# Ініціалізація доступних ресурсів
        self.resources = {
            "financial_report": "Фінансовий звіт корпорації за поточний рік",
            "hr_policy": "Документ політики відділу кадрів",
            "it_security": "Звіт з аудиту IT-безпеки"
        }
# Метод для запуску сервера
    def start_server(self):
# Створення сокета
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Встановлення опцій сокета для повторного використання адреси
        self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
# Прив’язка сокета до вказаного хоста та порту
        self.server_socket.bind((self.host, self.port))
# Початок прослуховування вхідних з’єднань
        self.server_socket.listen(5)
# Запис у журнал про запуск сервера
        self.logger.log(f"Сервер запущено на {self.host}:{self.port}")
# Метод для зупинки сервера
    def stop_server(self):
# Закриття сокета сервера
        self.server_socket.close()
        self.logger.log("Сервер зупинено")
# Метод для обробки запитів від клієнтів
    def handle_client(self, client_socket):
# Отримання даних від клієнта
        try:
            data = client_socket.recv(BUFFER_SIZE).decode()
# Розпаковка отриманих даних
            request_dict = json.loads(data)
            payload = request_dict.get("payload")
            signature = request_dict.get("signature")
# Перевірка підпису повідомлення
            if not self.crypto_manager.verify_signature(payload, signature):
                self.logger.log("Отримано запит з неправильним підписом")
                client_socket.send(json.dumps({"status": "failure", "reason":
"Invalid signature"}).encode())
                client_socket.close()
                return
# Десеріалізація корисного навантаження
            request = json.loads(payload)
            action = request.get("action")
# Обробка різних типів запитів
            if action == "register":
                username = request.get("username")
                password = request.get("password")
                if self.auth_manager.register_user(username, password):
                    response = {"status": "success", "message": "Registration
successful"}
                else:
                    response = {"status": "failure", "message": "User already
exists"}
                client_socket.send(json.dumps(response).encode())
# Обробка запиту на логін
            elif action == "login":
                username = request.get("username")
                password = request.get("password")
                if self.auth_manager.authenticate_user(username, password):
# Генерація сесійного токена
                    session_token = self.generate_session_token(username)
                    response = {"status": "success", "session_token":
session_token}
                else:
                    response = {"status": "failure", "message": "Authentication
failed"}
                client_socket.send(json.dumps(response).encode())
# Обробка запиту на отримання ресурсу
            elif action == "get_resource":
                session_token = request.get("session_token")
                resource = request.get("resource")
                if self.validate_session_token(session_token):
                    resource_content = self.resources.get(resource, "Запитуваний
ресурс не знайдено")
                    response = {"status": "success", "resource_content":
resource_content}
```

```
                else:
                    response = {"status": "failure", "message": "Invalid session
token"}
                client_socket.send(json.dumps(response).encode())
            else:
                response = {"status": "failure", "message": "Unknown action"}
                client_socket.send(json.dumps(response).encode())
        except Exception as e:
            self.logger.log(f"Помилка обробки клієнтського запиту: {str(e)}")
        finally:
            client_socket.close()
# Метод для генерації сесійного токена
    def generate_session_token(self, username):
# Створення випадкового рядка для токена
        token = ''.join(random.choices(string.ascii_letters + string.digits,
k=32))
# Збереження токена разом з іменем користувача та часом створення
        self.sessions[token] = {"username": username, "timestamp": time.time()}
# Повернення згенерованого токена
        return token
# Метод для перевірки валідності сесійного токена
    def validate_session_token(self, token):
# Перевірка чи існує токен у сесіях
        if token in self.sessions:
# Перевірка часу сесії (термін дії 1 година)
            session_age = time.time() - self.sessions[token]["timestamp"]
            if session_age < 3600:
                return True
            else:
                del self.sessions[token]
                return False
        else:
            return False
# Метод для обробки вхідних з’єднань у нескінченному циклі
    def run(self):
# Запис у журнал про запуск циклу обробки
        self.logger.log("Сервер готовий приймати клієнтські запити")
        while True:
            try:
                client_socket, addr = self.server_socket.accept()
# Запис інформації про підключення клієнта
                self.logger.log(f"Підключився клієнт: {addr}")
# Обробка запиту у новому потоці
                self.handle_client(client_socket)
            except Exception as e:
                self.logger.log(f"Помилка прийому з’єднання: {str(e)}")
                break

# Головна функція для запуску системи
def main():
# Створення об’єкта Logger
    logger = Logger()
# Створення об’єкта AuthenticationManager з журналом подій
    auth_manager = AuthenticationManager(logger)
# Створення об’єкта CryptoManager з глобальним секретним ключем
    crypto_manager = CryptoManager(GLOBAL_SECRET_KEY)
# Створення об’єкта CorporateServer для обробки запитів
    server = CorporateServer(SERVER_HOST, SERVER_PORT, auth_manager,
crypto_manager, logger)
# Запуск сервера
    server.start_server()
# Запуск сервера у окремому потоці для паралельної обробки (емулюємо окремий
процес)
```

```
    import threading
    server_thread = threading.Thread(target=server.run)
    server_thread.daemon = True
    server_thread.start()
# Затримка для забезпечення запуску сервера
    time.sleep(1)
# Ініціалізація мобільного клієнта з даними для входу
    mobile_client = MobileClient("user1", "password123", auth_manager,
crypto_manager, logger)
# Підключення мобільного клієнта до сервера
    mobile_client.connect_to_server()
# Виконання реєстрації користувача з мобільного пристрою
    mobile_client.register()
# Затримка для обробки запиту
    time.sleep(0.5)
# Виконання логіну користувача
    mobile_client.login()
# Затримка для обробки запиту
    time.sleep(0.5)
# Запит на отримання ресурсу "financial_report"
    mobile_client.request_resource("financial_report")
# Затримка для обробки запиту
    time.sleep(0.5)
# Запит на отримання ресурсу "hr_policy"
    mobile_client.request_resource("hr_policy")
# Затримка для обробки запиту
    time.sleep(0.5)
# Запит на отримання ресурсу "it_security"
    mobile_client.request_resource("it_security")
# Закриття з’єднання мобільного клієнта
    mobile_client.disconnect()
# Затримка перед завершенням роботи сервера
    time.sleep(1)
# Зупинка сервера
    server.stop_server()
# Виведення журналу подій
    logger.show_logs()
# Збереження журналу у файл
    logger.save_logs()

# Виклик головної функції для запуску програми
if __name__ == "__main__":
# Запис повідомлення про старт програми
    print("Запуск системи кібербезпеки для мобільного доступу до корпоративної
мережі")
    main()
    print("Завершення роботи програми")
```

Файл NetworkMonitor.py

```
import threading
import time
import random
import re
import socket
import ssl
import json
import os
from flask import Flask, request, render_template_string

class NetworkMonitor:
    def __init__(self):
        self.active_connections = []
        self.running = False
    def start(self):
        self.running = True
        self.thread = threading.Thread(target=self.monitor)
        self.thread.start()
    def stop(self):
        self.running = False
        self.thread.join()
    def monitor(self):
        while self.running:
            self.active_connections = self.get_connections()
            time.sleep(1)
    def get_connections(self):
        connections = []
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.bind(('localhost', 0))
            s.listen(1)
            port = s.getsockname()[1]
            connections.append(('127.0.0.1', port))
            s.close()
        except:
            pass
        return connections
    def get_active_connections(self):
        return self.active_connections

class SecurityLogAnalyzer:
    def __init__(self, log_file='system_logs.txt'):
        self.log_file = log_file
        self.suspicious_patterns = [r'error', r'failed', r'неуспішно',
r'вразлив']
        self.results = []
    def analyze(self):
        self.results = []
        if os.path.exists(self.log_file):
            with open(self.log_file, 'r') as f:
                lines = f.readlines()
                for line in lines:
                    for pattern in self.suspicious_patterns:
                        if re.search(pattern, line, re.IGNORECASE):
                            self.results.append(line.strip())
        return self.results
    def print_results(self):
        for result in self.results:
            print(result)

class AdminWebInterface:
```

```
    def __init__(self, host='0.0.0.0', port=5000):
        self.app = Flask(__name__)
        self.host = host
        self.port = port
        self.setup_routes()
    def setup_routes(self):
        @self.app.route('/')
        def index():
            return render_template_string("<h1>Admin Panel</h1><p>Welcome to the
admin interface</p>")
        @self.app.route('/logs')
        def logs():
            logs = ""
            if os.path.exists('system_logs.txt'):
                with open('system_logs.txt', 'r') as f:
                    logs = f.read().replace('\n', '<br>')
            return render_template_string("<h1>System Logs</h1><p>{{logs}}</p>",
logs=logs)
        @self.app.route('/config')
        def config():
            config_data = ""
            if os.path.exists('config_changes.txt'):
                with open('config_changes.txt', 'r') as f:
                    config_data = f.read().replace('\n', '<br>')
            return render_template_string("<h1>Configuration
Changes</h1><p>{{config}}</p>", config=config_data)
        @self.app.route('/analyze_logs')
        def analyze_logs():
            analyzer = SecurityLogAnalyzer()
            results = analyzer.analyze()
            result_str = "<br>".join(results)
            return render_template_string("<h1>Log
Analysis</h1><p>{{results}}</p>", results=result_str)
    def run(self):
        self.app.run(host=self.host, port=self.port)

class IntrusionDetectionModule:
    def __init__(self):
        self.alerts = []
        self.running = False
    def start(self):
        self.running = True
        self.thread = threading.Thread(target=self.detect)
        self.thread.start()
    def stop(self):
        self.running = False
        self.thread.join()
    def detect(self):
        while self.running:
            traffic = self.simulate_traffic()
            if self.analyze_traffic(traffic):
                alert = f"Intrusion detected at {time.strftime('%Y-%m-%d
%H:%M:%S')}"
                self.alerts.append(alert)
                self.log_alert(alert)
            time.sleep(2)
    def simulate_traffic(self):
        traffic = random.choice(["normal", "normal", "attack", "normal"])
        return traffic
    def analyze_traffic(self, traffic):
        if traffic == "attack":
            return True
        return False
```

```
    def log_alert(self, alert):
        with open('intrusion_alerts.txt', 'a') as f:
            f.write(alert + "\n")
    def get_alerts(self):
        return self.alerts

class ConfigurationChangeLogger:
    def __init__(self, config_file='config_changes.txt'):
        self.config_file = config_file
        self.changes = []
    def log_change(self, change_detail):
        timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
        entry = f"{timestamp} - {change_detail}"
        self.changes.append(entry)
        with open(self.config_file, 'a') as f:
            f.write(entry + "\n")
    def get_changes(self):
        return self.changes

def main():
    net_monitor = NetworkMonitor()
    net_monitor.start()
    id_module = IntrusionDetectionModule()
    id_module.start()
    config_logger = ConfigurationChangeLogger()
    config_logger.log_change("Initial configuration loaded")
    config_logger.log_change("Updated network settings")
    config_logger.log_change("Changed authentication parameters")
    admin_interface = AdminWebInterface(port=5001)
    admin_thread = threading.Thread(target=admin_interface.run)
    admin_thread.daemon = True
    admin_thread.start()
    start_time = time.time()
    while time.time() - start_time < 10:
        active = net_monitor.get_active_connections()
        print("Active connections:", active)
        time.sleep(1)
    id_module.stop()
    net_monitor.stop()
    analyzer = SecurityLogAnalyzer()
    results = analyzer.analyze()
    print("Security Log Analysis Results:")
    for res in results:
        print(res)
    print("Intrusion Alerts:")
    for alert in id_module.get_alerts():
        print(alert)
    print("Configuration Changes:")
    for change in config_logger.get_changes():
        print(change)
    time.sleep(2)

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

Файл NotificationSystem.py

```
import random
import string
import time
import hashlib
import base64
from flask import Flask, request, jsonify
import threading
import smtplib
from email.mime.text import MIMEText
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

class TwoFactorAuth:
    def __init__(self):
        self.codes = {}
    def generate_code(self, user):
        code = ''.join(random.choices(string.digits, k=6))
        self.codes[user] = (code, time.time())
        return code
    def verify_code(self, user, code):
        if user not in self.codes:
            return False
        stored_code, timestamp = self.codes[user]
        if time.time() - timestamp > 300:
            del self.codes[user]
            return False
        if stored_code == code:
            del self.codes[user]
            return True
        return False

class DataEncryption:
    def __init__(self, key=None):
        self.key = key if key else get_random_bytes(16)
    def pad(self, s):
        pad_len = 16 - len(s) % 16
        return s + bytes([pad_len]) * pad_len
    def unpad(self, s):
        pad_len = s[-1]
        return s[:-pad_len]
    def encrypt(self, plaintext):
        plaintext_bytes = plaintext.encode('utf-8')
        plaintext_padded = self.pad(plaintext_bytes)
        iv = get_random_bytes(16)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        encrypted = cipher.encrypt(plaintext_padded)
        return base64.b64encode(iv + encrypted).decode('utf-8')
    def decrypt(self, ciphertext):
        raw = base64.b64decode(ciphertext)
        iv = raw[:16]
        encrypted = raw[16:]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        decrypted_padded = cipher.decrypt(encrypted)
        return self.unpad(decrypted_padded).decode('utf-8')

class NotificationSystem:
    def __init__(self, smtp_server='smtp.example.com', smtp_port=587,
smtp_user='user@example.com', smtp_password='password'):
        self.smtp_server = smtp_server
        self.smtp_port = smtp_port
        self.smtp_user = smtp_user
```

```
        self.smtp_password = smtp_password
    def send_email(self, to_address, subject, message):
        msg = MIMEText(message)
        msg['Subject'] = subject
        msg['From'] = self.smtp_user
        msg['To'] = to_address
        server = smtplib.SMTP(self.smtp_server, self.smtp_port)
        server.starttls()
        server.login(self.smtp_user, self.smtp_password)
        server.send_message(msg)
        server.quit()
    def send_sms(self, phone_number, message):
        print(f"Sending SMS to {phone_number}: {message}")

class AccessControlManager:
    def __init__(self):
        self.user_roles = {}
        self.role_permissions = {}
    def add_role(self, role, permissions):
        self.role_permissions[role] = permissions
    def assign_role(self, user, role):
        self.user_roles[user] = role
    def check_permission(self, user, permission):
        role = self.user_roles.get(user, None)
        if not role:
            return False
        permissions = self.role_permissions.get(role, [])
        return permission in permissions

class ExtendedAPI:
    def __init__(self, host='0.0.0.0', port=5002):
        self.app = Flask(__name__)
        self.host = host
        self.port = port
        self.twofa = TwoFactorAuth()
        self.data_encryption = DataEncryption()
        self.notification = NotificationSystem()
        self.access_control = AccessControlManager()
        self.setup_roles()
        self.setup_routes()
    def setup_roles(self):
        self.access_control.add_role('admin', ['create', 'read', 'update',
'delete'])
        self.access_control.add_role('user', ['read'])
    def setup_routes(self):
        @self.app.route('/api/encrypt', methods=['POST'])
        def encrypt_text():
            data = request.get_json()
            user = data.get('user')
            if not self.access_control.check_permission(user, 'create'):
                return jsonify({'status': 'failure', 'message': 'Permission
denied'}), 403
            plaintext = data.get('text', '')
            ciphertext = self.data_encryption.encrypt(plaintext)
            return jsonify({'status': 'success', 'ciphertext': ciphertext})
        @self.app.route('/api/decrypt', methods=['POST'])
        def decrypt_text():
            data = request.get_json()
            user = data.get('user')
            if not self.access_control.check_permission(user, 'read'):
                return jsonify({'status': 'failure', 'message': 'Permission
denied'}), 403
            ciphertext = data.get('ciphertext', '')
```

```
            try:
                plaintext = self.data_encryption.decrypt(ciphertext)
                return jsonify({'status': 'success', 'plaintext': plaintext})
            except Exception as e:
                return jsonify({'status': 'failure', 'message': 'Decryption
error'})
        @self.app.route('/api/2fa/generate', methods=['POST'])
        def generate_2fa():
            data = request.get_json()
            user = data.get('user')
            code = self.twofa.generate_code(user)
            return jsonify({'status': 'success', 'code': code})
        @self.app.route('/api/2fa/verify', methods=['POST'])
        def verify_2fa():
            data = request.get_json()
            user = data.get('user')
            code = data.get('code')
            if self.twofa.verify_code(user, code):
                return jsonify({'status': 'success', 'message': 'Code
verified'})
            else:
                return jsonify({'status': 'failure', 'message': 'Invalid code'})
        @self.app.route('/api/notify/email', methods=['POST'])
        def notify_email():
            data = request.get_json()
            to_address = data.get('to')
            subject = data.get('subject', 'Notification')
            message = data.get('message', '')
            try:
                self.notification.send_email(to_address, subject, message)
                return jsonify({'status': 'success', 'message': 'Email sent'})
            except Exception as e:
                return jsonify({'status': 'failure', 'message': 'Email sending
failed'})
        @self.app.route('/api/notify/sms', methods=['POST'])
        def notify_sms():
            data = request.get_json()
            phone = data.get('phone')
            message = data.get('message', '')
            self.notification.send_sms(phone, message)
            return jsonify({'status': 'success', 'message': 'SMS sent'})
    def run(self):
        self.app.run(host=self.host, port=self.port)

def extended_api_main():
    extended_api = ExtendedAPI()
    api_thread = threading.Thread(target=extended_api.run)
    api_thread.daemon = True
    api_thread.start()
    time.sleep(2)
    print("Extended API is running on port 5002")
    while True:
        time.sleep(5)

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

Файл SecurityAuditModule.py

```
import threading
import time
import json
import os
import random
import string
import requests

class LDAPIntegration:
    def __init__(self, server_url='ldap://example.com',
base_dn='dc=example,dc=com'):
        self.server_url = server_url
        self.base_dn = base_dn
    def bind(self, username, password):
        if username == "admin" and password == "adminpass":
            return True
        return False
    def search_user(self, search_filter):
        dummy_users = [{"cn": "Alice", "uid": "alice"}, {"cn": "Bob", "uid":
"bob"}]
        result = [user for user in dummy_users if search_filter.lower() in
user["cn"].lower()]
        return result
    def get_user_info(self, uid):
        dummy_users = {"alice": {"cn": "Alice", "mail": "alice@example.com"},
"bob": {"cn": "Bob", "mail": "bob@example.com"}}
        return dummy_users.get(uid, {})

class BackupModule:
    def __init__(self, backup_dir='backups'):
        self.backup_dir = backup_dir
        if not os.path.exists(self.backup_dir):
            os.makedirs(self.backup_dir)
    def backup_file(self, file_path):
        if os.path.exists(file_path):
            timestamp = time.strftime('%Y%m%d%H%M%S')
            backup_file_name = os.path.join(self.backup_dir,
os.path.basename(file_path) + "." + timestamp + ".bak")
            with open(file_path, 'r') as src:
                data = src.read()
            with open(backup_file_name, 'w') as dst:
                dst.write(data)
            return backup_file_name
        return None
    def restore_file(self, backup_file, target_file):
        if os.path.exists(backup_file):
            with open(backup_file, 'r') as src:
                data = src.read()
            with open(target_file, 'w') as dst:
                dst.write(data)
            return True
        return False
    def list_backups(self):
        return os.listdir(self.backup_dir)

class AutoUpdateSystem:
    def __init__(self, update_url='http://updates.example.com/latest',
current_version='1.0.0'):
        self.update_url = update_url
        self.current_version = current_version
    def check_for_updates(self):
```

```
            latest_version = "1.0.1"
            if latest_version != self.current_version:
                return latest_version
            return None
        def download_update(self, version):
            update_file = f"update_{version}.zip"
            with open(update_file, 'w') as f:
                f.write("Dummy update content for version " + version)
            return update_file
        def apply_update(self, update_file):
            if os.path.exists(update_file):
                self.current_version = update_file.split('_')[1].split('.zip')[0]
                return True
            return False
        def run_update_cycle(self):
            latest = self.check_for_updates()
            if latest:
                update_file = self.download_update(latest)
                applied = self.apply_update(update_file)
                return applied
            return False

    class WebContentFilter:
        def __init__(self):
            self.blocked_keywords = ["malware", "phishing", "adult"]
        def is_allowed(self, url):
            for keyword in self.blocked_keywords:
                if keyword in url.lower():
                    return False
            return True
        def filter_content(self, content):
            for keyword in self.blocked_keywords:
                content = content.replace(keyword, "[blocked]")
            return content
        def fetch_and_filter(self, url):
            if not self.is_allowed(url):
                return "Content blocked"
            try:
                response = requests.get(url, timeout=5)
                if response.status_code == 200:
                    return self.filter_content(response.text)
                else:
                    return "Error fetching content"
            except Exception as e:
                return "Error fetching content"

    class SecurityAuditModule:
        def __init__(self, audit_file='security_audit.log'):
            self.audit_file = audit_file
        def record_event(self, event_description):
            timestamp = time.strftime('%Y-%m-%d %H:%M:%S')
            entry = json.dumps({"timestamp": timestamp, "event": event_description})
            with open(self.audit_file, 'a') as f:
                f.write(entry + "\n")
        def perform_audit(self):
            events = []
            if os.path.exists(self.audit_file):
                with open(self.audit_file, 'r') as f:
                    lines = f.readlines()
                    for line in lines:
                        try:
                            event = json.loads(line.strip())
                            events.append(event)
```

```
                    except:
                        pass
        return events
    def clear_audit_log(self):
        if os.path.exists(self.audit_file):
            os.remove(self.audit_file)
            return True
        return False

def run_ldap_integration():
    ldap_integration = LDAPIntegration()
    bound = ldap_integration.bind("admin", "adminpass")
    users = ldap_integration.search_user("Alice")
    user_info = ldap_integration.get_user_info("alice")
    print("LDAP Bound:", bound)
    print("LDAP Search:", users)
    print("LDAP User Info:", user_info)

def run_backup_module():
    backup_module = BackupModule()
    sample_file = "sample.txt"
    with open(sample_file, 'w') as f:
        f.write("This is a sample file for backup.")
    backup_file_name = backup_module.backup_file(sample_file)
    backups = backup_module.list_backups()
    restored = backup_module.restore_file(backup_file_name,
"restored_sample.txt")
    print("Backup file:", backup_file_name)
    print("Backup list:", backups)
    print("Restoration success:", restored)

def run_auto_update_system():
    auto_update = AutoUpdateSystem()
    update_available = auto_update.check_for_updates()
    update_applied = auto_update.run_update_cycle()
    print("Update available:", update_available)
    print("Update applied:", update_applied)
    print("Current version:", auto_update.current_version)

def run_web_content_filter():
    filter_module = WebContentFilter()
    allowed = filter_module.is_allowed("http://example.com/safe-content")
    blocked = filter_module.is_allowed("http://example.com/malware-content")
    content = filter_module.fetch_and_filter("http://example.com")
    print("URL allowed (safe):", allowed)
    print("URL allowed (blocked):", blocked)
    print("Filtered content:", content[:100])

def run_security_audit_module():
    audit_module = SecurityAuditModule()
    audit_module.record_event("User logged in")
    audit_module.record_event("Configuration changed")
    events = audit_module.perform_audit()
    cleared = audit_module.clear_audit_log()
    print("Audit events:", events)
    print("Audit log cleared:", cleared)

def main():
    run_ldap_integration()
    time.sleep(1)
    run_backup_module()
    time.sleep(1)
    run_auto_update_system()
```

```
    time.sleep(1)
    run_web_content_filter()
    time.sleep(1)
    run_security_audit_module()

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