МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем та мереж»

Група: 4КБ-02

Дипломний проєкт

здобувача освіти денної форми навчання
КБ.02.09.000.ДП

ТАТАРИНСЬКОГО АНДРІЯ ОЛЕКСІЙОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка програмної моделі відновлення даних після атаки на інформаційну систему

Проектний матеріал складається з пояснювальної записки на 74 сторінках та графічного (презентаційного) матеріалу на 13 аркушах (слайдах)

Дипломник _____ (Татаринський А.О)

Керівник _____ (Залапін О.І)

Консультанти:

з економічного розділу _____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «28» червня 2025 р. Протокол ЕК № 7

Оцінка ЕК 4 (добре) / 75 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Татаринського Андрія Олексійовича*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка програмної моделі відновлення даних після атаки на інформаційну систему*

затверджена наказом по коледжу від "14" 11 2024 р. №

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *1. Реалізувати програмну модель мовою програмування Python; 2. Програмне забезпечення реалізувати засобами Hashlib та Watchdog у інтерактивній консолі Python Shell; 3. Застосувати SMTP-протокол в програмній моделі (синхронізацію даних на Email); 4. Реалізувати синхронізацію та відновлення даних в режимі реального часу.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз принципу дії та розповсюдження програм вимагачів; Огляд існуючих програмних рішень для відновлення даних; Впровадження синхронізації даних; Інтеграція хешування в роботу програмної моделі; Впровадження системи сповіщень на електрону адресу; Впровадження системи моніторингу та сканування; Тестування та аналіз роботи програмної моделі.

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Загальна модель взаємодії програмної моделі з користувачем та мережею; Логіка дії програмної моделі відновлення; Схема роботи бібліотеки SMtp (синхронізація даних на Email); Задіяння програми, що змінює хеші файлу на випадкові; Запуск роботи програмної моделі відновлення; Можливості та перспективи подальшого розвитку програмної моделі відновлення даних після атаки на інформаційну систему.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Залапін О.І.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Залапін О.І.* _____
(підпис)

Завдання прийняв до виконання *Татаринський А.О.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Постановка задачі проектування*	19.05.2025	Виконано
2	*Аналіз технічного завдання та пошук літератури*	20.05.2025	Виконано
3	*Аналіз принципу дії та розповсюдження програм вимагачів*	21.05.2025	Виконано
4	*Аналіз структури ransomware;*	22.05.2025	Виконано
5	*Огляд існуючих програмних рішень для відновлення даних;*	23.05.2025	Виконано
6	*Проектування програмної моделі відновлення даних;*	26.05.2025	Виконано
7	*Впровадження синхронізації даних;*	26.05.2025	Виконано
8	*Інтеграція хешування в роботу програмної моделі;*	27.05.2025	Виконано
9	*Впровадження системи сповіщень на електрону адресу;*	28.05.2025	Виконано
10	*Впровадження системи моніторингу та сканування;*	29.05.2025	Виконано
11	*Тестування та аналіз роботи програмної моделі*	30.05.2025	Виконано
12	*Виконання економічних розрахунків*	13.06.2025	Виконано
13	*Розробка питань з охорони праці та техніки безпеки*	14.06.2025	Виконано
14	*Підготовка мультимедійної презентації проекту*	16.06.2025	Виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 02. 21 000. 00 ДП	Дипломний проект		
А4			КБ 02. 21 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 21 000. 00 ДП		
Розробив		Татаринський А.		16.05.25	Розробка програмної моделі відновлення даних після атаки на інформаційну систему	Літ.	Аркуш 4 / Аркушів 74
Перевірив		Залапін О.І.		16.05.25		Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.4КБ-02	
Затверд.		Кривченко Ю.В.		23.06			

Зміст

ВСТУП

Сучасні інформаційні системи цілодобово зустрічаються із різними видами кібератак, серед яких неабияку загрозу становлять атаки, що націлені на шифрування, знищення чи крадіжку даних користувачів. Згідно даним звіту Verizon DBIR 2023, більше ніж 74 відсотки вдалих кібератак включали в себе компрометацію даних, також чималий збиток з одного інциденту склав більш ніж 4.4млн$ (IBM, 2023). Попри це все класичні методи безпеки, такі як антивіруси, далеко не завжди здатні запобігти ушкодженню інформації, що робить резервне копіювання та відновлення даних ключовими елементами запобігання кібератакам.

З часів пандемії COVID-19 й по сьогоднішній день зловмисники почали набирати неабияких обертів й головними мішенями стали люди, що працюють віддалено та великі компанії.

Особливу загрозу становлять програми-вимагачі (ransomware), котрі шифрують й блокують доступ до файлів з подальшим вимаганням. Найпопулярнішими прикладами в сучасній історії кібератак є віруси шифрувальники: «WannaCry» та «Petya.A» які завдали чималої шкоди банкам, державним установам, корпоративним мережам.

На сьогоднішній день існує небагато рішень запобігання та боротьби проти програм-вимагачів, незважаючи на це, ці рішення є одні з найдієвіших щодо запобігань кібератакам. Одне з цих рішень буде представлено у цій дипломній роботі.

Метою дипломної роботи є розробка програмної моделі відновлення даних після кібератаки, а також наочне демонстрування важливості створення резервних копій даних на різних рівнях задля запобігання втрати інформації, а саме: автоматизоване резервне копіювання та синхронізацію даних в режимі реального часу. Демонстрування на практиці роботи програмної моделі відновлення даних після керованого шифрування однієї з папок із файлами у якості демонстрації спроможності програми виявлення та усунення потенційних кіберзагроз.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз принципу дії та розповсюдження програм вимагачів

Програма-вимагач, програма-здирник, програма-шифрувальник (англ. ransomware, ransom — викуп і software — програмне забезпечення) — це тип шкідливого програмного забезпечення, що приховано інсталюється на комп'ютери користувачів здійснюючи несанкційовані злочинні дії. Програми-вимагачі надають зловмисникам можливість віддалено блокувати комп'ютер та змінювати вміст даних (шифрувати). Після вдалої атаки на систему програма виводить на спливаючому вікні повідомлення, про те, що комп'ютер було заблоковано, і що до нього немає доступу, доки не буде користувачем або підприємством сплачено викуп. На сьогодні існує кілька значно різних способів роботи програм-вимагачів:

1. Шифрування файлової системи
2. Блокування роботи або чинення перешкод системі
3. Блокування роботи або чинення перешкод у використанні браузеру

1.1.1 Аналіз способів ураження пристрою шкідливим ПЗ

Схованка погрози в межах іншого файлу чи програми виглядає настільки невинно для користувача, що він просто вікриває її: прикріплення до електронної пошти, відео з сторінок сумнівного походження, навіть оновлення системи від нібито тих же надійних програм, як Windows чи Adobe Flash. Як тільки завантажується на комп'ютер, шкідлива програма активується, блокує операційну систему, чи окремі показує попередження із загрозою і вимогами щодо оплати викупу. До відомих програм-зловмисників належать наступні програми.

Trojan-Ransom.Win32.Cryzip - це шкідливий програмний засіб належить до класу ransomware, тобто програм, що здійснюють шифрування користувацьких файлів із наступною вимогою викупу за їхнє відновлення. Зараження зазвичай відбувається через відкриття вкладень у фішингових електронних листах,

встановлення піратського програмного забезпечення, використання фальшивих оновлень або експлуатацію вразливостей у системі.

Після запуску шкідливий код інтегрується у системні процеси, що дозволяє уникнути виявлення антивірусними засобами. Надалі програма сканує жорсткі диски, мережеві каталоги та зовнішні накопичувачі у пошуках файлів певних форматів (наприклад, документи, бази даних, зображення тощо). Для процесу шифрування застосовується гібридна криптографічна схема: симетричний алгоритм (наприклад, AES) забезпечує швидкість шифрування, а асиметричний (RSA) – захист ключа шифрування.

Після завершення шифрування програма видаляє тіньові копії (Volume Shadow Copies), що унеможливлює відновлення даних стандартними засобами Windows. На пристрої залишаються текстові або HTML-файли з інструкціями щодо сплати викупу, зазвичай у формі криптовалюти (наприклад, біткойн). Окремі модифікації подібних програм можуть також містити функціонал для здійснення DDoS-атак, викрадення даних або поширення на інші пристрої в локальній мережі.

Trojan-Ransom.Win32.Gpcode - це один із найбільш показових прикладів ранніх програм-здирників, що з’явився приблизно у 2006 році. Від самого початку цей шкідливий код вирізнявся технологічною новизною: на відміну від типових троянів того часу, він застосовував асиметричне шифрування (RSA), що суттєво ускладнювало відновлення даних без приватного ключа. Початкові версії використовували 660-бітні ключі, але згодом розробники збільшили їхню довжину до 1024 біт, підвищуючи захист від дешифрування.

Процес зараження зазвичай розпочинався або з соціальної інженерії, або через експлуатацію вразливостей програмного забезпечення. Після проникнення у систему Gpcode ретельно маскував свою присутність, ін’єктуючи власний код у легітимні системні процеси. Далі вірус сканував файлову систему, вибірково знаходячи найбільш цінні для користувача дані: документи, архіви, бази даних тощо. Важливо зазначити, що Gpcode не унеможливлював роботу системи

повністю, а залишав її функціональною, аби користувач міг виконати інструкції щодо сплати викупу.

Технічна реалізація шифрування передбачала використання випадкового сеансового AES-ключа для безпосереднього шифрування даних, після чого сам ключ шифрувався відкритим RSA-ключем. Після завершення процесу жертва отримувала детальну інструкцію щодо подальших дій та вимог до викупу, часто через анонімні платіжні системи чи криптовалютні сервіси. Для унеможливлення відновлення інформації троян видаляв тіньові копії (Volume Shadow Copy Service) та блокував доступ до системних утиліт.

Складність аналізу та протидії Gpcode визначалась використанням сучасних на той час криптографічних алгоритмів. Проте, дослідникам згодом вдалось виявити уразливості в реалізації RSA, що надало змогу створити утиліти для відновлення даних у певних версіях вірусу. У підсумку, Gpcode став вагомою віхою у розвитку програм-здирників, вплинувши як на кіберзлочинність, так і на підходи до захисту даних у галузі інформаційної безпеки.

Trojan-Ransom.Win32.Rector - цей вірус являє собою високотехнологічний зразок сучасної кіберзагрози, що поєднує функції криптографічного шантажу та крадіжки конфіденційної інформації. Його механізми проникнення в систему зазвичай базуються на соціальній інженерії: шкідливий код часто вбудовується в підроблені документи (DOCX, PDF), які користувач отримує через цілеспрямовані атаки, або розповсюджується через експлойт-кіти, що використовують вразливості застарілих браузерних плагінів.

Після потрапляння в систему Rector проявляє значний рівень складності в уникненні детектування: він впроваджує (підлаштовує) свій код у довірені процеси операційної системи (зокрема explorer.exe та svchost.exe), модифікуючи записи в реєстрі для забезпечення тривалого збереження на пристрої, а також активно використовує методи “living off the land” (використання легітимного програмного забезпечення для здійснення деструктивних дій), застосовуючи легітимні системні утиліти (як PowerShell) для виконання шкідливих операцій.

Основною метою діяльності цього шкідливого ПЗ є шифрування користувацьких даних із подальшою вимогою викупу. Для цього Rector здійснює сканування всіх доступних носіїв, включаючи мережеві диски та зовнішні накопичувачі, ідентифікуючи цінні файли (документи, бази даних, архіви тощо). Процес шифрування реалізовано із застосуванням гібридної криптосистеми: дані спершу шифруються алгоритмом AES-256 з унікальним сеансовим ключем, який додатково кодується за допомогою RSA-2048. Власна реалізація криптобібліотек (окремий модуль, який несе відповідальність за підписання та шифрування даних) ускладнює процес аналізу та відновлення інформації.

У процесі шифрування вірус паралельно ліквідує засоби відновлення системи: видаляє тіньові копії за допомогою vssadmin, усуває точки відновлення, блокує доступ до диспетчера завдань і редактора реєстру через політики груп. Після завершення основного етапу атаки в кожній папці залишаються інструкції щодо сплати викупу (наприклад, “!RECOVER_FILES!.txt” або “!READ_ME!.html”) з детальним описом процедури, яка зазвичай передбачає використання криптовалют (Bitcoin або Monero). У більш сучасних версіях реалізовано зворотний зв’язок через Tor-сайти з автоматичною генерацією унікального ідентифікатора жертви.

Окрім основної функції шифрування, Rector здійснює збір особистої інформації: зчитує автозаповнення браузерів, робить знімки активних вікон, отримує файли конфігурації криптогаманців і збирає системні дані (IP-адреси, імена користувачів, перелік встановленого ПЗ). Ця інформація передається на керуючий сервер через захищене HTTPS-з’єднання. Для ускладнення аналізу коду застосовуються різноманітні методи обфускації (цілеспрямоване заплутування вихідного коду), динамічне формування API-викликів та ручні антиемуляційні перевірки.

Загалом Trojan-Ransom.Win32.Rector ілюструє сучасні тенденції розвитку шкідливого програмного забезпечення та поєднує в собі багаторівневу криптографію, інструменти для приховування активності та розширені засоби збору даних.

Trojan-Ransom.Win32.Xorist - це шкідливе ПЗ являє собою модульне шкідливе програмне забезпечення типу ransomware, що належить до категорії відкритих конструкторів, які дозволяють зловмисникам легко адаптувати код під конкретні цілі. Його поширення зазвичай здійснюється через фішингові електронні листи з вкладеними ZIP або RAR-архівами, що містять експлойти чи скрипти для завантаження основного компонента вірусу.

Після проникнення у систему, Xorist виконує перевірку середовища виконання з метою виявлення віртуальних машин чи інструментів аналізу (дебагерів, емулюючих середовищ). За відсутності ознак дослідження вірус починає інтегруватися у легітимні системні процеси, наприклад, explorer.exe або svchost.exe, що дозволяє йому маскувати свою присутність і уникати швидкого виявлення.

Основна шкідлива функція Xorist полягає у шифруванні користувацьких файлів шляхом використання алгоритму XOR з додатковими криптографічними методами, що забезпечує достатній рівень блокування даних. Вірус сканує підключені носії, мережеві папки та локально синхронізовані хмарні сервіси, цілеспрямовано вибираючи файли певних розширень — документи, мультимедіа, бази даних, архіви — уникаючи системних файлів, аби не порушувати працездатність операційної системи. Після шифрування до кожного файлу додається унікальне розширення, а в кожній папці залишаються текстові файли з інструкціями щодо викупу (наприклад, README_FOR_DECRYPT.txt), де вказується сума (зазвичай $100–500 у Bitcoin) та контакти зловмисників.

Окремий модуль Xorist може здійснювати збір конфіденційної інформації: збережених паролів із браузерів, конфігурацій FTP-клієнтів, даних криптогаманців. Ця інформація передається на віддалений сервер через зашифроване з’єднання, що дає змогу не лише шантажувати жертву, а й реалізовувати викрадені дані на чорному ринку. Для протидії аналізу й антивірусному виявленню Xorist використовує різноманітні прийоми: динамічне генерування API-викликів, обфускацію (цілеспрямоване заплутування вихідного коду), регулярне оновлення конфігурації з серверу. Після завершення

шифрування програма також намагається вимкнути системні служби резервного копіювання та видалити точки відновлення, що практично унеможливлює відновлення даних без втручання зловмисників.

1.1.2 Огляд існуючих рішень запобігання зіткненню з ransomware

Для ефективного захисту комп'ютерних систем від програм-вимагачів та іншого шкідливого програмного забезпечення варто дотримуватися сучасних принципів кібергігієни та впроваджувати багаторівневий захист. Ось низка рекомендацій яких варто дотримуватися:

1. Регулярне оновлення операційної системи: кожен патч, який випускає виробник (наприклад, Microsoft через Windows Update або розробники Linux-дистрибутивів), закриває критичні вразливості, що часто використовуються зловмисниками для несанкціонованого доступу.
2. Оновлення прикладного програмного забезпечення, зокрема браузерів, офісних пакетів та систем керування базами даних, також є надзвичайно важливим. Ці продукти нерідко містять так звані нуль-денні вразливості (вразливість програмного забезпечення, про яку ще невідомо користувачам чи розробникам ПЗ й проти якої ще не було розроблено механізми запобігання ураження), які використовуються для drive-by download атак через шкідливі або підроблені вебсторінки.
3. Налаштування мережевого екрану (фаєрвола) повинно передбачати блокування непотрібних вхідних і вихідних з'єднань, з особливою увагою до портів, що потенційно використовуються для спілкування зі C&C-серверами. Використання рішень на зразок Windows Defender Firewall або pfSense дозволяє ефективно протидіяти трафіку, пов'язаному з бот-мережами.
4. Уникнення взаємодії із підозрілими електронними листами, особливо тими, що містять макроси або посилання на фішингові сайти, є критично важливим елементом захисту. Зловмисники часто імітують легальні сервіси, зокрема онлайн-банкінг чи поштові служби, щоб отримати доступ до систем користувачів.

5. Використання сучасних антивірусних рішень із функціями активного захисту, як-от Bitdefender із машинним навчанням, дозволяє виявляти не лише відомі загрози, а й нові, ще не додані до сигнатурних баз, завдяки аналізу поведінки процесів. Додатково рекомендується перевіряти файли перед інсталяцією за допомогою онлайн-сканерів типу VirusTotal.
6. Найважливішим компонентом забезпечення інформаційної безпеки залишається регулярне резервне копіювання даних за схемою 3-2-1: три копії на різних носіях, дві з яких локальні (наприклад, зовнішній SSD та NAS), а ще одна — у хмарному сховищі з підтримкою історії версій (наприклад, AWS S3 Versioning). Особливої уваги заслуговують незмінні резервні копії (immutable backups), які захищають дані від знищення або шифрування навіть у разі реалізації атаки програмою-вимагачем.
7. Додаткові заходи включають обмеження прав користувачів відповідно до принципу найменших привілеїв (PoLP), використання віртуалізованих середовищ для роботи з ризикованими додатками (наприклад, Sandboxie), а також застосування групових політик (GPO) для блокування виконання скриптів із тимчасових папок. Для критичних систем доцільно впроваджувати апаратні засоби захисту, такі як TPM-модулі для шифрування дисків або HSM (Hardware Security Module) для безпечного зберігання криптографічних ключів.

1.1.3 Розбір засобів протидії шифрувальникам

Для виявлення та усунення програм-вимагачів доцільно провести повне сканування системи із застосуванням актуального антивірусного програмного забезпечення. Серед найбільш ефективних засобів та застосунків для виявлення та боротьби із шкідливим ПЗ та відновленням даних можна виділити:

- Microsoft Security Essentials
- Microsoft Safety Scanner
- Windows Defender
- Acronis True Image
- Windows Server Backup

1.1.4 Ознайомлення з першими випадками зараження програмами шифрувальниками

Перші випадки зараження користувачів персональних комп'ютерів програмами-вимагачами були зафіксовані у травні 2005 року. Серед відомих представників цього типу шкідливого програмного забезпечення можна назвати TROJ.RANSOM.A, Archiveus, Krotten, Cryzip, MayArchive. Особливої уваги заслуговує вірус Gpcode та його різновиди: Gpcode.a, Gpcode.ac, Gpcode.ag, Gpcode.ak. Зокрема, останній з перелічених використовував для шифрування файлів алгоритм RSA із 1024-бітовим ключем, що значно ускладнювало процес дешифрування.

У липні 2023 року спеціалісти з комп'ютерної безпеки FortiGuard Labs опублікували результати дослідження, присвячені новому трояну-здирнику Big Head. Вказаний шкідливий софт імітує критичне оновлення операційної системи Windows, демонструючи фальшивий екран Windows Update. Поки користувач очікує завершення «оновлення», програма у фоновому режимі шифрує файли на пристрої; цей процес триває близько 30 секунд.

Існує також варіація цього шкідника, що використовує PowerShell-скрипт із назвою cry.ps1 для здійснення шифрування. Згідно з дослідженнями компанії Trend Micro, Big Head додатково перевіряє, чи працює система у віртуальному середовищі (зокрема VirtualBox або VMware), а також видаляє резервні копії, що значно ускладнює відновлення даних після атаки. Таким чином, цей шкідник становить серйозну загрозу для користувачів.

1.1.5 Огляд сучасних програмних моделей з боротьби з програмами шифрувальниками

Windows Server Backup — це, вбудований інструмент резервного копіювання та відновлення даних, який Microsoft інтегрувала у всі свої серверні операційні системи починаючи з версії 2008 (рис.1.1) Основна його мета — забезпечення збереження критично важливих даних, самої операційної системи, а також віртуальних машин. Це дозволяє захистити інформацію від апаратних збоїв, випадкового видалення чи навіть кіберзагроз, включно з ransomware. Інструмент

глибоко інтегрований з файловою системою NTFS та серверними ролями, тому користувач має змогу виконувати як повне резервне копіювання всього “заліза” (bare-metal recovery), так і точкове — наприклад, окремих папок, томів або системного стану (Active Directory, DHCP, DNS).

Технологічна основа Windows Server Backup — це Volume Shadow Copy Service (VSS), що дозволяє отримувати послідовні знімки даних навіть під час активної роботи сервера, тобто без потреби зупиняти будь-які сервіси. Користувач може налаштувати автоматичний розклад резервного копіювання або запускати процес вручну, використовуючи як графічний інтерфейс (wbadmin.msc), так і командний рядок (утиліта wbadmin). Підтримується зберігання резервних копій на різних носіях: локальних дисках, зовнішніх SSD чи HDD, мережевих папках (NAS, SMB) та навіть оптичних дисках. Водночас для повного відновлення системи рекомендовано використовувати локальні або підключені через USB/SATA пристрої.

Важливою особливістю є підтримка інкрементного резервного копіювання: після створення першої повної копії зберігаються лише змінені дані, що суттєво зменшує витрати місця та часу. Для збереження резервних копій використовується формат VHD (Virtual Hard Disk), сумісний з іншими Microsoft-продуктами, зокрема Hyper-V, що дає змогу швидко підключати резервну копію у вигляді віртуального диска для доступу до потрібних файлів.

Процедури відновлення можуть виконуватися на різних рівнях: від повного відновлення сервера до вибіркового повернення окремих файлів або папок через відповідний майстер. Для критичних сервісів, таких як Active Directory, реалізовано механізми versioning, які дають змогу відкотитися до певного стану на конкретний момент часу.

Водночас Windows Server Backup має свої обмеження: відсутня підтримка дедуплікації, кластерних середовищ (окрім базової роботи з CSV) та хмарних сховищ (за винятком ручного копіювання в Azure).

У складних інфраструктурах Microsoft радить використовувати System Center Data Protection Manager або Azure Backup, що надають ширший

функціонал, включно з політиками довгострокового зберігання, централізованим управлінням і підтримкою SQL/Exchange.

Для ефективного використання Windows Server Backup доцільно постійно перевіряти цілісність резервних копій (наприклад, пробним відновленням віртуальних машин), зберігати копії на ізольованих носіях для підвищення захисту від ransomware, а також комбінувати цей інструмент з іншими методами, наприклад, replication через DFS-R. Варто суворо обмежувати права доступу до резервних копій, щоб уникнути несанкціонованих дій або випадкового видалення, а у разі використання мережевих сховищ — застосовувати шифрування трафіку (наприклад, через IPsec) для захисту даних під час передачі.

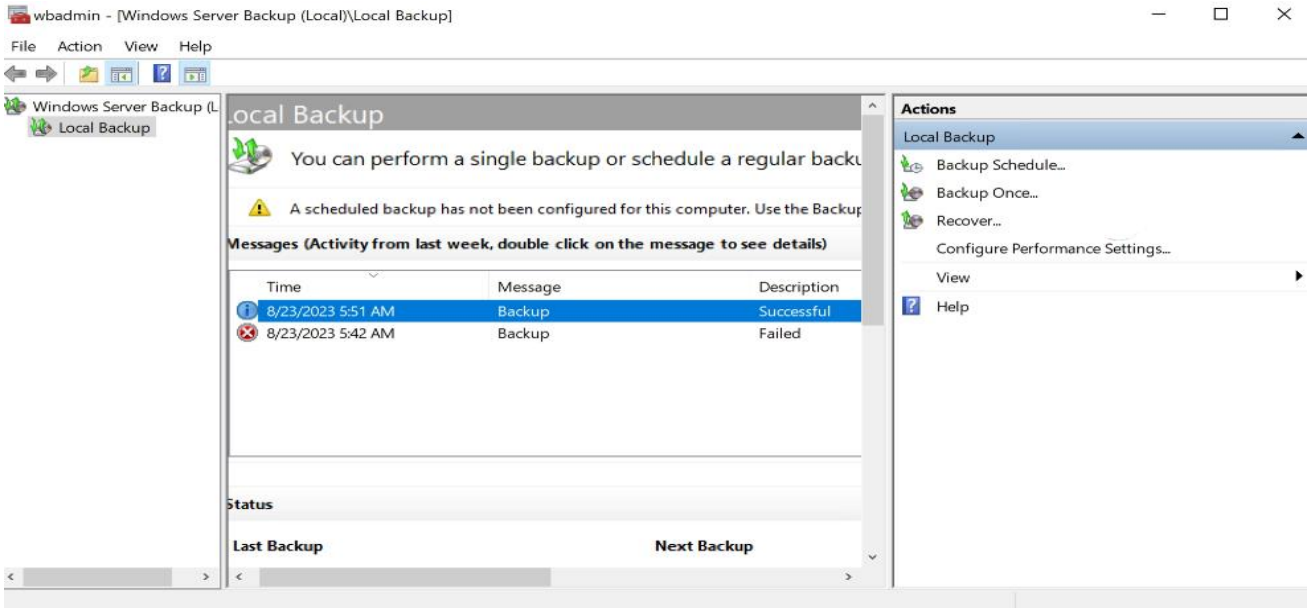


Рисунок 1.1 Функціонал створення резервних копій Windows Server Backup

Microsoft Defender Antivirus (минула назва - Windows Defender) є вбудованим антивірусним рішенням, інтегрованим у операційні системи Windows, починаючи з версій Windows 10 та Windows Server 2016 (рис.1.2).

Це програмне забезпечення забезпечує захист у реальному часі від різноманітних сучасних кіберзагроз, включаючи віруси, трояни, шпигунське ПЗ, руткіти та програм-вимагачів.

Ефективність захисту досягається завдяки тісній інтеграції з ядром операційної системи, що дозволяє виявляти та блокувати шкідливі дії без істотного впливу на продуктивність комп’ютера.

Архітектура Microsoft Defender Antivirus базується на багаторівневому захисті. Система поєднує класичні сигнатурні методи з сучасними технологіями, такими як штучний інтелект і машинне навчання.

Ключові компоненти включають Real-Time Protection, яка моніторить файлову систему, оперативну пам'ять, реєстр і мережеву активність; Cloud-Delivered Protection, що відправляє підозрілі файли на хмарний аналіз для виявлення нових загроз; Controlled Folder Access, який обмежує несанкціоновані зміни важливих даних (особливо актуально для протидії ransomware); а також Exploit Guard, що застосовує технології на кшталт ASLR і DEP для запобігання використанню вразливостей програмного забезпечення.

Оновлення сигнатурних баз відбувається автоматично через Windows Update, завдяки чому забезпечується актуальний рівень захисту без додаткових дій зі сторони користувача.

Для корпоративного сектору розроблено Microsoft Defender для кінцевих точок (Defender ATP), який надає розширені можливості централізованого керування, поведінкового аналізу (EDR) та інтеграції з SIEM-системами.

Інтерфейс Defender вирізняється мінімалізмом і доступний через Windows Security. Користувачі можуть переглядати статус захисту, результати останніх перевірок, запускати різні типи сканування (швидке, повне або вибіркове) та налаштовувати винятки для певних файлів або папок.

Додатково Defender інтегрується з Microsoft SmartScreen, який блокує запуск невідомих чи непідписаних програм, а також з Microsoft Defender Firewall для фільтрації мережевого трафіку.

На відміну від ранніх версій, сучасний Microsoft Defender Antivirus демонструє високу ефективність у незалежних тестах (зокрема, AV-Test, AV-Comparatives) — нерідко перевершуючи навіть платні антивірусні рішення у виявленні zero-day атак.

Водночас, для забезпечення максимальної безпеки рекомендується регулярно оновлювати операційну систему. Це дозволить закривати вразливості та захищати дані від потенційних загроз.

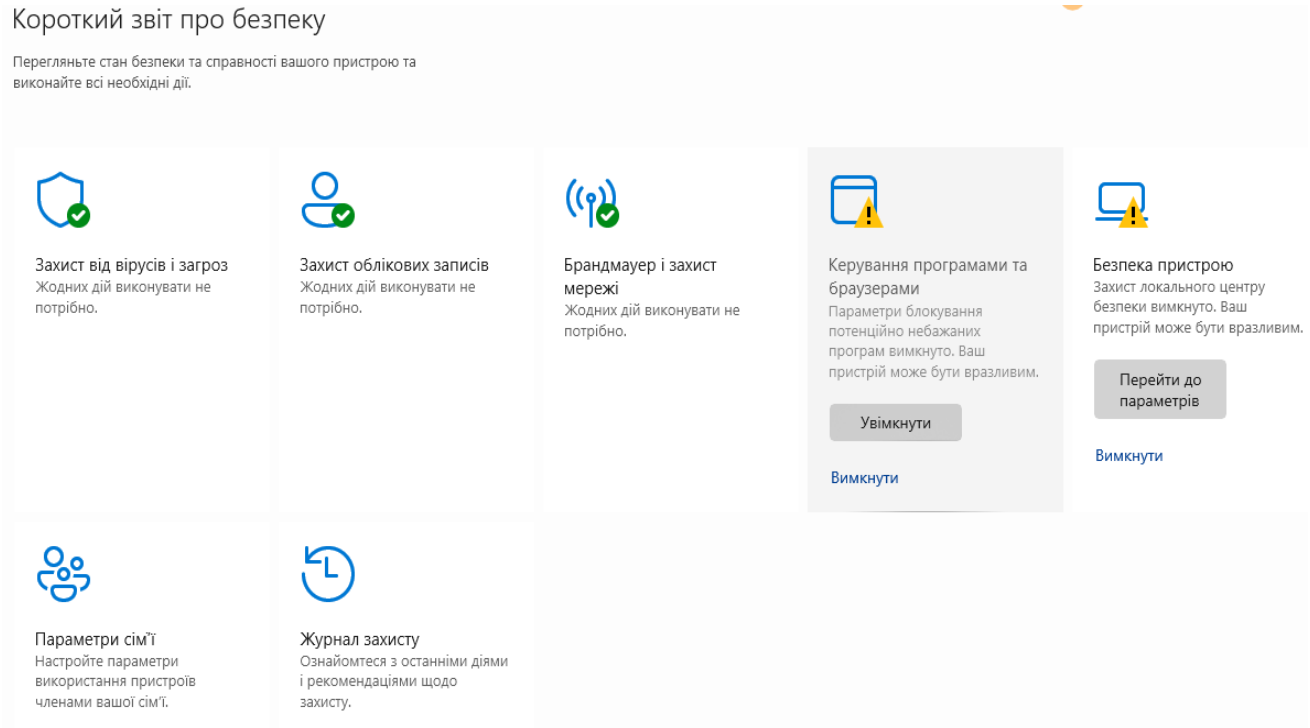


Рисунок 1.2 Огляд функціоналу та можливостей Microsoft Defender

Microsoft Safety Scanner - інструмент невідкладної допомоги для комп'ютерних систем, особливо у випадках, коли стандартні антивірусні засоби виявляються неефективними. Розроблений компанією Microsoft, цей продукт позиціонується як додатковий засіб захисту, а не як повноцінний антивірус. Його основна перевага полягає у можливості запуску без інсталяції, що є особливо корисним для вже інфікованих систем, де основний антивірус може бути заблокований або пошкоджений шкідливим програмним забезпеченням (рис.1.3).

Утиліта пропонує три режими сканування: швидке (аналіз критичних ділянок системи, таких як автозавантаження, активні процеси та ключові системні файли), повне (детальна перевірка всіх файлів, включаючи архіви та скрипти) і вибіркове (аналіз вказаних користувачем папок або розділів). Всі перевірки здійснюються на основі актуальних сигнатур, які завантажуються з інтернету під час кожного запуску програми.

Технологічно Microsoft Safety Scanner заснований на тому ж антивірусному рушії, що і Windows Defender, але має певні унікальні можливості. Зокрема, програма здатна виявляти шкідливі процеси, які маскуються під системні (наприклад, в explorer.exe), аналізувати порушені точки відновлення системи та знаходити файли, зашифровані програмами-вимагачами. Результати сканування

зберігаються у тимчасових логах у папці %TEMP%, де міститься детальна інформація про виявлені загрози, їхнє розташування та заходи, які були до них застосовані (видалення, ізоляція, або неможливість обробки).

Водночас слід враховувати, що Microsoft Safety Scanner не може замінити повноцінний антивірусний програмний комплекс. Вона не забезпечує захисту в реальному часі, не оновлюється автоматично та самовидаляється через 10 днів після завантаження.

Основне призначення утиліти — оперативна допомога у критичних ситуаціях, коли традиційні засоби захисту не спрацювали, або коли спостерігається аномальна поведінка системи (наприклад, значне уповільнення, неочікувані повідомлення про помилки, невідомі процеси у диспетчері задач). Згідно з рекомендаціями Microsoft, цей інструмент слід використовувати разом із Windows Defender та регулярними оновленнями системи безпеки.

По завершенні сканування програма надає детальний звіт із зазначенням кількості перевірених файлів, виявлених загроз і виконаних дій.

Якщо віруси не були знайдені, це не гарантує повної відсутності шкідливого програмного забезпечення, оскільки окремі види загроз можуть використовувати методи приховування, такі як поліморфізм або шифрування. У такому разі доцільно застосувати додаткові інструменти (наприклад, Malwarebytes Anti-Malware), або звернутися до фахівців з кібербезпеки. Запуск Microsoft Safety Scanner у безпечному режимі з підтримкою мережі є надзвичайно важливим етапом для ефективного виявлення складних загроз, що можуть перешкоджати скануванню у стандартному середовищі операційної системи.

У безпечному режимі (Safe Mode with Networking) система функціонує з мінімально необхідним набором драйверів та служб, водночас зберігається можливість доступу до мережі Інтернет для завантаження актуальних вірусних баз.

Ця процедура є особливо дієвою для ідентифікації руткітів, прихованих процесів і файлів, що маскуються під системні компоненти, а також для детального аналізу шкідливих служб.

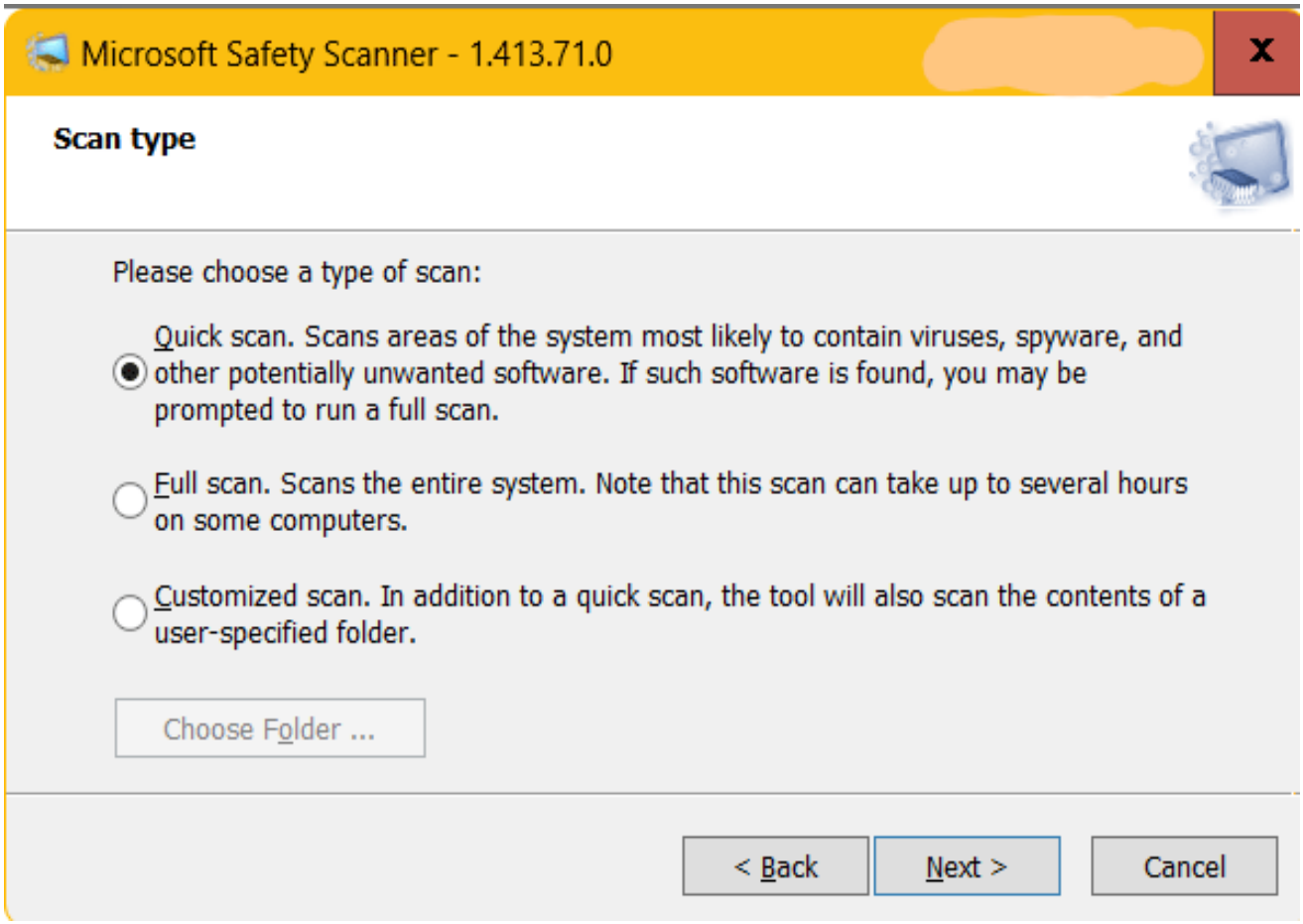


Рисунок 1.3 Демонстрація інтерфейсу з функціями Microsoft Safety Scanner

Microsoft Security Essentials (MSE) — це безкоштовне антивірусне програмне забезпечення, представлене корпорацією Microsoft у 2009 році. Основною аудиторією стали домашні користувачі та малі офіси; завданням MSE є виявлення та усунення різних типів шкідливого ПЗ, включно з вірусами, шпигунськими програмами та руткітами. Програма замінила Windows Live OneCare і була інтегрована у Windows Vista та Windows 7. Починаючи з новіших версій ОС, її функції повністю перейшли до “Захисника Windows” (рис.1.4).

У плані архітектури MSE реалізує постійний моніторинг файлової системи, реєстру та мережевих підключень з метою оперативного виявлення підозрілої активності (рис). Окремий модуль аналізує загрози на рівні руткітів, використовуючи низькорівневі методи сканування. Також впроваджено базовий поведінковий аналіз, що дозволяє розпізнавати нові шкідливі програми за характерним шаблонних дій. Сканування реалізовано багатопоточно, а також підтримується перевірка архівованих файлів.

Головна перевага MSE — глибока інтеграція із системними інтерфейсами Windows (зокрема, Win32 API та AMSI), що забезпечує мінімальне навантаження на ресурси.

Програма не містить додаткових інструментів на кшталт фаєрволу чи менеджера паролів, зосереджуючись виключно на базовому захисті. Водночас це обмежує її ефективність у порівнянні з сучасними платними антивірусами, особливо щодо виявлення складних загроз або zero-day експлойтів.

У незалежних тестах, наприклад, AV-Test, MSE демонструє нижчі показники детекції, що визнає і сам виробник, рекомендувавши комбінувати MSE з регулярними оновленнями ОС та обережністю користувачів.

Користувачі мають змогу налаштовувати розклад сканування, визначати дії щодо виявлених загроз (видалення, карантин, ігнорування), а також виключати окремі об’єкти зі списку сканування.

Інтерфейс програми мінімалістичний: основні відомості про захист, результати останнього сканування й запуск перевірки розміщені у головному вікні, а журнали подій ведуться у стандартному системному лог-файлі.

Microsoft Security Essentials (MSE), безперечно, являв собою важливий етап у розвитку систем безпеки Microsoft, навіть попри завершення офіційної підтримки для Windows 7 ще у 2020 році. Цей безкоштовний антивірус став першим широкодоступним продуктом компанії, який поєднував класичні методи виявлення зі спробами поведінкового аналізу.

Його архітектура – зокрема взаємодія з ядром Windows через Filtering Platform API та функції моніторингу файлових операцій у реальному часі – створила підґрунтя для подальшого розвитку Microsoft Defender.

Сучасний Defender значно розширив функціонал: він використовує хмарні технології, алгоритми машинного навчання, захист від програм-вимагачів й інтегрує EDR-систему (Endpoint Detection and Response), що на даний момент більше та краще відповідає сучасним вимогам до кібербезпеки, однак на ізольованих системах без доступу до інтернету MSE може послугувати аварійним рішенням у крайніх випадках.

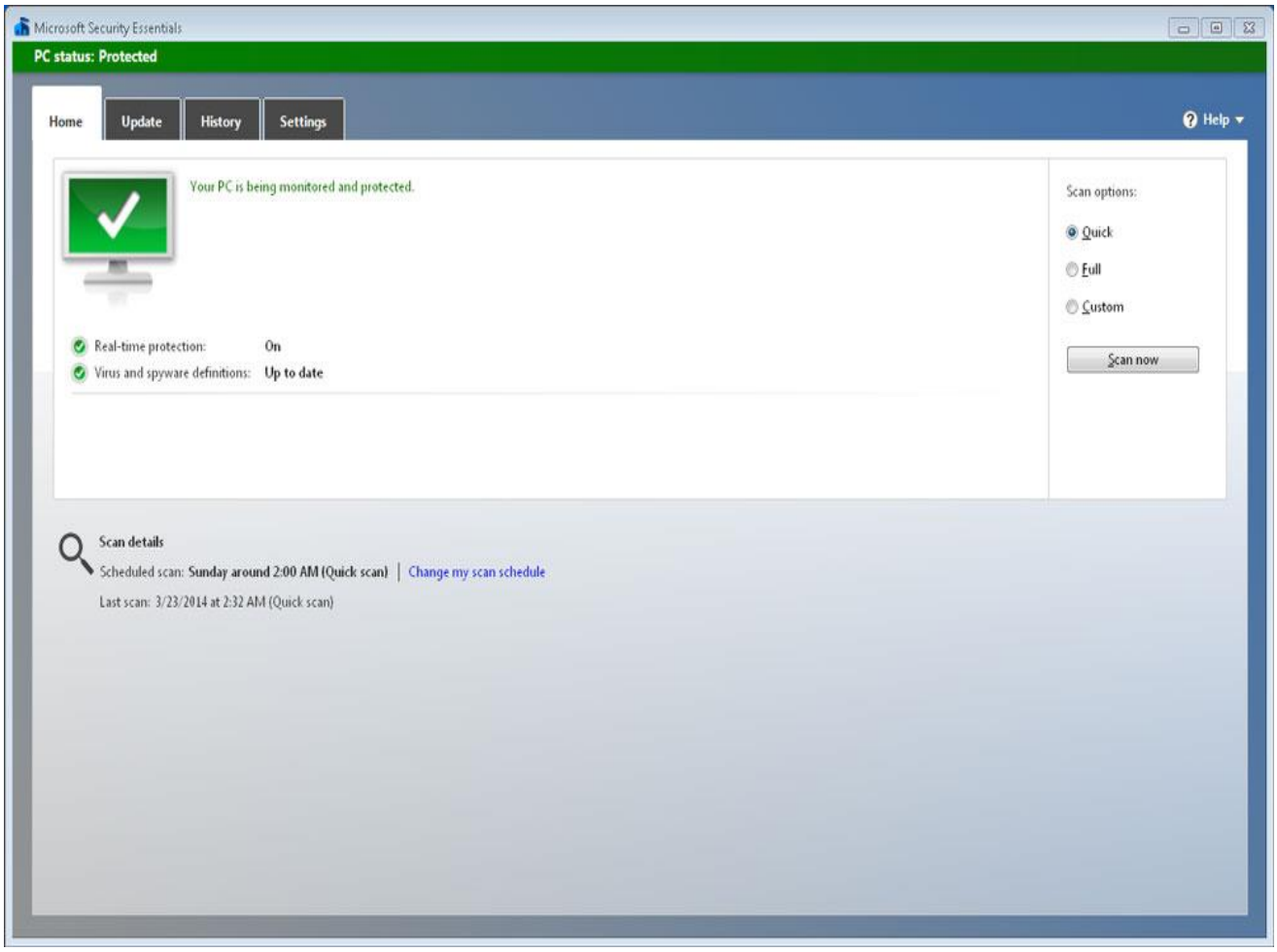


Рисунок 1.4 Демонстрація інтерфейсу з функціями Microsoft Security Essentials

Acronis True Image — це програмне забезпечення, яке орієнтоване на створення резервних копій, відновлення та перенесення даних операційних систем, включаючи Microsoft Windows, macOS, iOS і Android. Програма дозволяє створювати завантажувальні носії, що забезпечує можливість використання функціоналу навіть до запуску основної операційної системи.

Головна функція Acronis True Image полягає у створенні резервних копій даних із подальшою можливістю їх відновлення. Програмне забезпечення підтримує повне відновлення операційної системи разом із користувацькими файлами, а також перенесення розділів на нові носії. Передбачено два режими резервного копіювання: окремих файлів або цілих розділів (жорсткого диска), причому образ розділу є його точною посекторною копією. Дані можна зберігати на іншому розділі, зовнішньому носії або у хмарному сховищі. Програма сумісна з різними пристроями, зокрема комп’ютерами, планшетами та смартфонами (рис. 1.5).

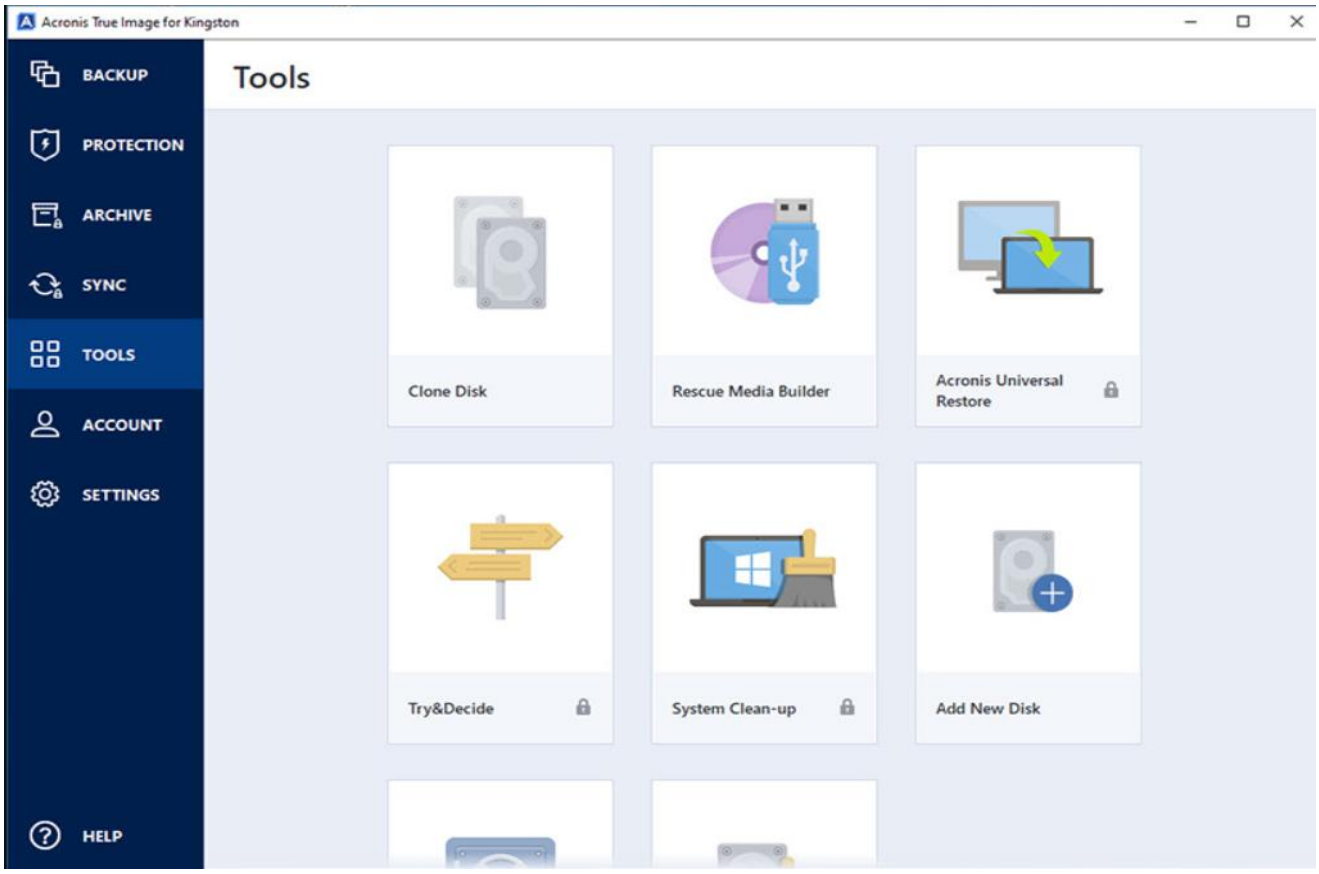


Рисунок 1.5 Інтерфейс із головними функціями Acronis True Image

Основні функції Acronis True Image можна окреслити наступним чином:

- Клонування диска — це процес створення повної копії усіх розділів одного диска на інший носій. Така операція дозволяє перенести як операційну систему, так і особисті файли на новий, часто більший диск, що особливо актуально при модернізації апаратного забезпечення.
- Створення так званого «рятівного» носія — передбачає запис програми на флеш-накопичувач або оптичний диск. Завдяки цьому можна запускати процес перегляду, копіювання та відновлення розділів ще до старту операційної системи, що є вкрай корисним у разі серйозних збоїв.
- Універсальний відновник — дозволяє створити резервну копію на зовнішньому носії з метою оперативного відновлення працездатності комп’ютера у випадку несправності ОС чи втрати важливих даних.
- Паралельний доступ — забезпечує можливість віддаленого підключення до цільового комп’ютера з інших пристроїв, що значно розширює варіанти управління та доступу до даних.

Acronis True Image представляє собою не просто програму для резервного копіювання, а повноцінний комплекс інструментів для оптимізації роботи з комп'ютером.

Серед його можливостей — збереження детальної інформації про комп'ютер та користувачів, налаштування запуску програми до старту операційної системи, підключення нового диску, повне та незворотне видалення даних з носія, перегляд і зміна розміру виділеної програмою області на диску, знищення історії дій користувача, а також обмеження потенційно небезпечної активності для захисту важливих файлів.

Програма підтримує широкий спектр файлових систем: NTFS, FAT32, ext2, ext3, ext4, ReiserFS, а також Linux Swap. Резервні копії зберігаються у форматі .tib, який сумісний із різними версіями програми, хоча гарантія коректної роботи файлів, створених у застарілих версіях, із новими редакціями відсутня.

У нинішню епоху стрімкої цифрової трансформації, коли кіберзагрози набувають усе більш витончених і непередбачуваних форм, питання забезпечення надійної кібербезпеки виходить на передній план. Особливо цінними стають інструменти, які не тільки ефективно протидіють потенційним ризикам, а й гарантують стабільний рівень захисту без зайвої складності для користувача.

Розглянуте програмне забезпечення є прикладом подібного рішення: воно гармонійно поєднує потужний захист із простотою використання, що робить його оптимальним вибором як для фахівців, так і для широкого кола користувачів.

Архітектура цього програмного продукту враховує всі основні аспекти кібербезпеки — від базового антивірусного захисту до протидії складним цілеспрямованим атакам. Завдяки впровадженню продуманих алгоритмів та регулярним оновленням система постійно випереджає потенційні загрози.

Саме це програмне забезпечення використовувалося на підприємстві, де я проходив переддипломну практику.

Завдяки ознайомленням з принципом дії цього ПЗ в мене з'являлися нові ідеї щодо створення програмної моделі відновлення даних після кібератаки.

1.2 Проектування програмної моделі відновлення після атаки ransomware

Створення моделі програми, що відновлює дані після кібератаки. Ця програма має стати розширеним рішенням для захисту від програм-шифрувальників та синхронізації файлів та відновлення у разі їх шифрування або пошкодження. Основною дією цієї програми є автоматичний моніторинг змін у визначеному диску (папці) , миттєвого створення бекапів (резервних копій) у вказаних місцях, зберігання усіх копій та автоматичного відновлення у разі пошкодження, несанкціоновану модифікацію та просто випадкового видалення файлів з подальшою заміною їх на неуражені копії (рис. 1.6). Більшість функціоналу програми базується на принципі автоматичного моніторингу файлової системи за допомогою бібліотеки watchdog, що дозволяє відстежувати всі зміни в режимі реального часу. Коли користувач створює, змінює або видаляє файл у підконтрольній папці, системою фіксується ті чи інша дія з файлами та ініціюються відповідні процедури синхронізації. Для кожного файлу обчислюється криптографічний хеш за допомогою алгоритму SHA-256, що порівнює відповідність хешів, і у разі виявлення іншого хешу у файлах – видалення та відновлення неуражених файлів з бекапів. Цей механізм є ключовим для виявлення фактів зловмисного втручання, наприклад вірусами-шифраторами .Ось частина коду, на якому вказано куди саме відбувається синхронізація всіх бекапів та метод їх хешування:

SOURCE_FOLDER = "C:\\server1" - вихідна папка, що моніториться.

DESTINATION_FOLDERS = ["E:\\server2", "F:\\server3"] - папки для резервних копій.

FILE_EXTENSIONS_TO_SYNC = [] - список розширень для синхронізації, якщо порожній - синхронізація усіх файлів

SMTP_SERVER = "smtp.gmail.com" - налаштування для відправки email через Gmail.

SMTP_PORT = 587 – один з портів, що використовується для відправки даних на email

EMAIL_LOGIN = "qqq643942@gmail.com"

EMAIL_PASSWORD = "usmd gpck qaqt qvoy"

RECIPIENT_EMAIL = "student1789@otfk.ukr.education"

MAX_EMAIL_RETRIES = 5 – максимальна кількість запитів за 1 цикл роботи коду

HASH_ALGORITHM = 'sha256' – використаний алгоритм хешування файлів

INTEGRITY_CHECK_INTERVAL = 3600 - інтервал перевірки цілісності файлів (1 година, тобто 3600 секунд)

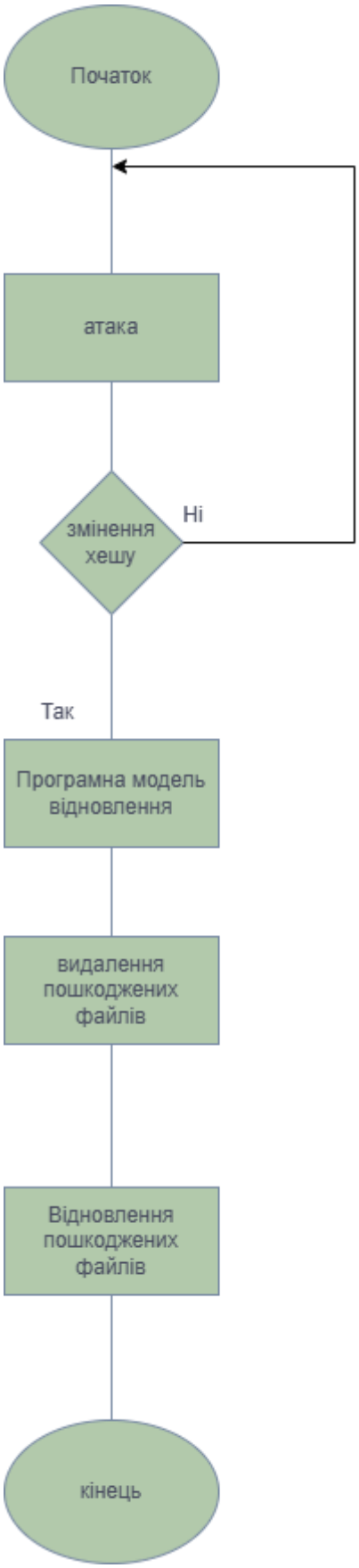


Рисунок 1.6 Загальна схема дії програмної моделі відновлення

Розробка програмної моделі почалася з вивчення вимог щодо системи резервного копіювання. В ході аналізу, було вирішено, що класичні підходи до

створення бекапів, що синхронізуються за розкладом, не забезпечують достатнього рівня захисту від існуючих загроз. В зв'язку з цим було прийнято рішення реалізувати систему, що комбінує переваги синхронізації у реальному часі з механізмами контролю цілісності даних. На етапі проектування я вирішив використовувати мову програмування Pyton через її кросплатформеність та наявність зручних бібліотек для роботи з файловою системою:

import time - для роботи з таймерами та затримками.

import shutil- для копіювання файлів.

import os - для роботи з файлами в операційній системі.

import smtplib - для надсилання email-повідомлень користувачу.

import hashlib - для роботи з хешами файлів

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.base import MIMEBase

from email import encoders

from watchdog.observers import Observer - для моніторингу змін у файловій системі.

from watchdog.events import FileSystemEventHandler

1.3 Впровадження синхронізації даних

Перший етап реалізації полягав у створенні базового механізму синхронізації між папками. Для цього було реалізовано клас SyncEventHandler, який працює з файлами із бібліотеки watchdog. Цей клас відповідає за обробку всіх подій, які відбуваються у файловій системі. На даному етапі була виконана основна робота з обробкою фільтрації тимчасових файлів, що створюються різними програмами (наприклад, текстовими редакторами) і не повинні синхронізуватись. Використання класу SyncEventHandler, що успадковується від FileSystemEventHandler та відокремлення (фільтрація) тимчасових файлів від синхронізації:

```
def __init__(self, source_folder, destination_folders, extensions_to_sync):
super().__init__()
```

self.source_folder = source_folder

self.destination_folders = destination_folders

self.extensions_to_sync = [ext.lower() for ext in extensions_to_sync]

self.last_processed_files = {} - словник для уникнення дублювання подій

self.file_hashes = {} - збереження хешів файлів

self.last_integrity_check = 0

self._initialize_hashes() - задіює хеші для всіх файлів

def _should_sync(self, file_path):

filename = os.path.basename(file_path)

temp_file_prefixes = ['~$', '~WR', '~ucm', '~', '.tmp'] - Ігнорування вказаних тимчасових файлів.

if any(filename.startswith(prefix) for prefix in temp_file_prefixes):

return False

if not self.extensions_to_sync:

return True

return any(file_path.lower().endswith(ext) for ext in self.extensions_to_sync)

print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] {file_path}")

1.4 Інтеграція хешування в роботу програмної моделі

На другому етапі розробки було впровадженно ключовий захисний механізм – систему перевірки цілісності файлів за допомогою криптографічної функції – хешування.

Ця функція є основною протидією ransomware-атакам, випадковим пошкодженням даних та несанкціонованим змінам.

Алгоритми безпечного хешування (SHA) — це ключовий елемент сучасної криптографії, який перетворює вхідні дані будь-якої довжини на унікальний цифровий відбиток фіксованого розміру. Такі хеш-функції відіграють важливу роль у забезпеченні цілісності даних, створенні цифрових підписів і багатьох інших аспектах інформаційної безпеки.

Історія розвитку SHA розпочалася у 1993 році з появою першої версії (SHA-0), яка використовувала 160-бітний хеш, але через виявлені криптографічні

вразливості швидко була замінена оновленим стандартом — SHA-1. Останній став широко застосовуваним і залишався актуальним протягом наступного десятиліття.

Згодом, у відповідь на нові криптоаналітичні виклики, у 2001 році було представлено сімейство SHA-2, що налічує кілька варіантів хешу різної довжини (224, 256, 384, 512 біт). Найбільш поширеною стала версія SHA-256, яка забезпечує високий рівень стійкості до криптографічних атак і широко використовується у сучасних системах безпеки, зокрема у блокчейн-технологіях та алгоритмах консенсусу Bitcoin.

У 2012 році, після масштабного конкурсу, NIST затвердив SHA-3 (Keccak) — принципово новий підхід до хешування, який базується на інших математичних засадах, але підтримує аналогічні розміри хешу, що значно спрощує інтеграцію у вже існуючі інфраструктури.

Алгоритми SHA забезпечують кілька фундаментальних властивостей: навіть незначна зміна вхідних даних змінює хеш кардинально; відновлення початкового повідомлення за хешем практично неможливе; алгоритми розроблені для уникнення колізій, тобто різні вхідні дані не повинні мати однаковий хеш.

Сфери застосування SHA надзвичайно широкі: перевірка цілісності файлів, цифрові підписи, захист паролів у базах даних, блокчейн-технології тощо. Перехід від SHA-1 до SHA-2, а згодом і до SHA-3 відображає постійний розвиток галузі у відповідь на зростаючу обчислювальну потужність та нові методи криптоаналізу. На сьогодні SHA-256 вважається одним із найнадійніших алгоритмів, тоді як SHA-1 є застарілим і не рекомендується для використання у нових системах.

1.4.1 Описання принципу роботи хешування

Кожен файл у контрольованій папці (SOURCE_FOLDER) та його резервні копії отримують цифровий відбиток – хеш, обчислений за алгоритмом SHA-256.

Цей хеш зберігається у внутрішньому словнику програми (file_hashes) під ключем, яким є відносний шлях до файлу. Приклад:

- Файл: C:\server1\file1.docx
- Ключ у словнику: file1.docx

- Хеш: a1b2c3… (64 символи)

1.4.2 Перевірка процесу обчислення та інтегрування хешів

1. Ініціалізація. При запуску програмна модель сканує всі файли у вихідній та резервних папках, обчислює їхні хеші й створює таблицю цілісності. Це гарантує, що система має еталонні значення для подальшого порівняння.

2. Моніторинг змін, під час роботи програма відстежує дії файлової системи:

- Створення файлу: новий хеш додається до словнику.

- Зміна файлу: обчислюється новий хеш та порівнюється зі збереженим

- Видалення файлу: хеш видаляється зі словника (якщо файл не вдалося відновити з резервної копії).

3. Реакція на розбіжності якщо хеш зміненого файлу не збігається з еталонним:

- Сприйняття системою файлу як ознаку пошкодження (наприклад, через шифрування)

- Автоматичний запуск процедури відновлення з найновішої резервної копії, де хеш файлу відповідає оригіналу.

- Якщо відновлення неможливе (всі копії уражені), файл видаляється для запобігання подальшому нанесення шкоди.

```
def _initialize_hashes(self):
all_folders = [self.source_folder] + self.destination_folders
for folder in all_folders:
rel_path = os.path.relpath(file_path, folder)
for root, _, files in os.walk(folder):
for file in files:
file_path = os.path.join(root, file)
if self._should_sync(file_path):
try:
file_hash = self._calculate_file_hash(file_path)
rel_path = os.path.relpath(file_path, folder)
if rel_path not in self.file_hashes: -
```

self.file_hashes[rel_path] = file_hash

except Exception as e:

1.4.3 Огляд зі сторони технічної складової

1. Періодична перевірка. Крім реакції на подію кожну годину (INTEGRITY_CHECK_INTERVAL = 3600) програма повністю сканує всі файли для виявлення аномалій, які могли бути припущені (якщо зловмисник тимчасово вимкнув моніторинг).

2. Оптимізація. Для великих файлів хеш обчислюється порційно (блоками по 64 КБ), щоб уникнути перенавантаження пам’яті.

3. Обробка помилок. Якщо файл тимчасово заблокований (наприклад відкритий у іншій програмі), система робить 5 спроб щоб його перевірити з інтервалом 0.5 секунд.

1.5 Впровадження системи сповіщень на електрону адресу

На третьому етапі розробки було реалізовано комплексний механізм сповіщення користувача про всі зміни у файловій системі. Ця функція перетворює програму з автономного інструменту синхронізації на повноцінну систему моніторингу із зворотнім зв’язком, що є дуже важливою складовою для користувачів та корпоративних середовищ.

1.5.1 Використання архітектури системи сповіщень

Система побудована на інтеграції з SMTP-сервером через бібліотеку smtplib – стандартної бібліотеки Python. Для формування складних MIME-повідомлень використовується модуль email. Частина з коду, що відповідає за синхронізацію даних на електрону адресу:

SMTP_SERVER = "smtp.gmail.com"

SMTP_PORT = 587

EMAIL_LOGIN = "qqq643942@gmail.com"

EMAIL_PASSWORD = "usmd gpck qaqt qvoy"

RECIPIENT_EMAIL = "student1789@otfk.ukr.education"

MAX_EMAIL_RETRIES = 5

Simple Mail Transfer Protocol (SMTP, Простий Протокол Пересилання Пошти) – це один із базових протоколів електронної пошти, фактично стандарт для передачі листів між поштовими серверами. Його роль у сучасних системах електронної комунікації – ключова: саме SMTP забезпечує доставку повідомлень, синхронізуючи відправника та одержувача через чітко визначений обмін командами та відповідями у текстовому форматі з мінімальними затримками (рис.1.7).

Принцип роботи протоколу доволі простий і побудований на ініціативі відправника. Клієнтська програма або поштовий сервер ініціює встановлення з'єднання з сервером-одержувачем, після чого відбувається поетапний обмін командами й відповідями. Кожна наступна дія клієнта можлива лише після отримання відповіді від сервера, що гарантує послідовність і контрольованість процесу доставки.

SMTP вирізняється тим, що не має вбудованих механізмів відкладеної доставки: якщо сервер одержувача недоступний у момент спроби передати лист, повідомлення не буде доставлено. Це обумовлює необхідність постійної готовності поштових серверів до прийому листів.

У типовій SMTP-інфраструктурі розрізняють таких поштових клієнтів: Mail User Agent (MUA) і поштові сервери: Mail Transfer Agent (MTA). Звичні програми на кшталт Outlook чи Thunderbird виступають у ролі клієнтів: вони надсилають листи на поштовий сервер, який вже бере на себе маршрутизацію та подальшу доставку.MUA відповідає за створення, редагування та відправку листів від імені користувача, тоді як MTA забезпечує передачу між серверами. Кожен етап доставки пошти базується на взаємодії цих двох компонентів. Ефективне функціонування всієї поштової системи базується на злагодженій взаємодії між MUA та MTA. Якщо клієнтський інтерфейс форматує повідомлення некоректно, це може призвести до порушення стандартів SMTP і, відповідно, до збоїв у доставці електронної пошти. Серверна складова, у свою чергу, відповідає за надійну передачу та обробку даних, і її стабільна робота є критично важливою для цілісності системи.

Рисунок 1.7 Загальна схема дії Simple Mail Transfer Protocol

Технічно протокол працює так: клієнт встановлює TCP-з’єднання із сервером (зазвичай через порт 25, 587 або 465), далі відбувається обмін службовими командами, ідентифікація сторін, вказання адреси відправника, одержувача(-ів) та передача тексту повідомлення.

У сучасних реалізаціях SMTP широко використовуються розширення для забезпечення безпеки: шифрування через STARTTLS, аутентифікація за допомогою логіна й пароля (SMTP AUTH), а також різноманітні засоби захисту від спаму. Ці механізми підвищують безпечність і конфіденційність електронного листування.

Важливо підкреслити, що всі команди та відповіді в SMTP передаються у зрозумілому текстовому вигляді, що значно полегшує діагностику й аналіз проблем у роботі протоколу.

Попри появу альтернативних технологій, SMTP залишається основним стандартом для передачі електронної пошти завдяки своїй простоті, універсальності й надійності. Він використовується як у невеликих персональних поштових сервісах, так і в масштабних корпоративних рішеннях.

Особливісті реалізації бібліотеки smtplib є:

1. Багатокомпонентні повідомлення (MIME):

- Текстова частина з описом події
- Прикріплення зміненого файлу
- Метадані (час події, шлях до файлу)

2. Механізм повторних спроб:

- При недоступності SMTP-сервера система робить до 5 спроб надіслати повідомлення
- Інтервал між спробами – 2 секунди
- Логування помилок для подальшого аналізу

1.5.2 Деталізований аналіз роботи системи сповіщень

Формування повідомлення користувачу та кожне сповіщення генерується класом _send_email_with_attechment і містить:

def _send_email_with_attachment(self, file_path, subject):

msg = MIMEMultipart()

msg["Від"] = EMAIL_LOGIN

msg["До"] = RECIPIENT_EMAIL

msg["Суб'єкт"] = f"File Sync: {subject}"

msg.attach(MIMEText(f"Файл {os.path.basename(file_path)} був {subject.lower()}."))

with open(file_path, "rb") as attachment:

part = MIMEBase("application", "octet-stream")

part.set_payload(attachment.read())

encoders.encode_base64(part)

part.add_header("ContentDisposition",f"attachment;

filename={os.path.basename(file_path)}")

msg.attach(part)

with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:

server.starttls()

server.login(EMAIL_LOGIN, EMAIL_PASSWORD)

server.send_message(msg)

1. Заголовки :
   - Відправник (назва програми + адреса)
   - Одержувач (вказаний в коді)
   - Тема з вказанням типу події та імені файлу
2. Тіло повідомлення:
   - Час події з точністю до секунди
   - Повний шлях до файлу
   - Тип операції (модифікація/створення/видалення)
   - Додаткова інформація про резервні копії
3. Прикріплення:
   - Для подій (модифікації/створення) – копія актуального файлу
   - Для видалення – остання доступна версія з архіву

Передача та обробка повідомлень у цій системі здійснюється відповідно до сучасних стандартів безпеки та оптимізації. На початку встановлюється захищене

TLS-з’єднання із SMTP-сервером, що забезпечує конфіденційність передачі даних. Відбувається обмін сертифікатами та погодження параметрів шифрування, що унеможливлює перехоплення інформації сторонніми особами.

Після встановлення захищеного каналу відбувається автентифікація користувача через SASL, де перевіряється коректність введених облікових даних. У разі невдалої авторизації система реєструє подію із зазначенням часу, IP-адреси та типу помилки.

Після успішної автентифікації розпочинається поетапна передача компонентів повідомлення: спершу передаються заголовки (відправник, одержувач, тема), далі — основний текст повідомлення, і на завершення — вкладення. На кожному етапі здійснюється перевірка цілісності й коректності даних. Після завершення передачі система ініціює закриття з’єднання із додатковою перевіркою статусу операції, що дозволяє підтвердити успішність або виявити можливі проблеми.

Система обробки помилок враховує різноманітні сценарії збоїв. У разі проблем із мережею (наприклад, тайм-аут чи недоступність сервера) здійснюються повторні спроби з поступовим збільшенням інтервалів. При помилках автентифікації фіксуються захешовані облікові дані та параметри сеансу. Для помилок із вкладеннями система генерує зрозумілі повідомлення для користувача та веде детальний журнал із технічною інформацією. У випадках перевищення квот користувачу надається інформація про ліміти та рекомендації щодо їх збільшення.

1.5.3 Розгляд технічних особливостей реалізації

Технічна реалізація охоплює низку оптимізаційних і безпекових механізмів. Для оптимізації мережевого трафіку використовується стиснення вкладень (наприклад, ZIP), що дозволяє зменшити обсяг переданих даних до 70%. Система обмежує максимальний розмір повідомлень, запобігаючи перевантаженню серверів.

Для масових змін застосовується пакетна відправка, яка зменшує навантаження на мережу.

Безпека системи забезпечується комплексом заходів: шифруванням з’єднання (STARTTLS), санацією вхідних даних для запобігання атакам, а також диференційованим доступом на основі ролей і прав користувачів.

Масштабованість реалізується завдяки сучасним архітектурним рішенням. Система використовує черги повідомлень (RabbitMQ або Apache Kafka) та багатопоточність для забезпечення стабільної роботи за високого навантаження, а також підтримує інтеграцію з корпоративними сервісами (Slack, Microsoft Teams) та стандартними API (REST, GraphQL) для спрощення інтеграції з іншими системами.

1.6 Впровадження системи моніторингу та сканування

На четвертому етапі розробки було впроваджено одні з найважливіших механізмів для підвищення надійності та ефективності роботи програми. Основна увага приділялася створенню системи фонового моніторингу, що забезпечує виявлення пошкоджень навіть у разі спроб обійти основні механізми захисту.

1.6.1 Реалізація періодичної перевірки та архітектура фонового сканування

1. Інтеграція з основним циклом програми:

- Перевірка запускається паралельно з основним потоком обробки подій
- Використовується модуль time для контролю інтервалів
- Реалізація сканування через метод _periodic_integrity_check

```
def _periodic_integrity_check(self):
"""Періодично перевіряє цілісність усіх файлів"""
current_time = time.time()
if current_time - self.last_integrity_check < INTEGRITY_CHECK_INTERVAL:
return
print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}]
```

2. Робота механізму планувальника подій:

- Час останньої перевірки зберігається у змінній last_integrity_check

```
self.last_integrity_check = current_time
```

print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Запуск періодичної перевірки цілісності...")

for root, _, files in os.walk(self.source_folder):

for file in files:

file_path = os.path.join(root, file)

if self._should_sync(file_path):

if not self._check_file_integrity(file_path):

print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Виявлено пошкодження файлу: {file_path}")

if not self._restore_file_from_backup(file_path):

try:

os.remove(file_path)

print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Видалено пошкоджений файл: {file_path}")

self._sync_deletion(file_path)

except Exception as e:

print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка видалення пошкодженого файлу {file_path}: {e}")

- Поточний час порівнюється зі значенням *INTEGRITY_CHECK_INTERVAL*
- При досягненні інтервалу ініціюється повне сканування

3. Робота алгоритму перевірки:

- Рекурсивний обхід всієї структури каталогів (os.walk)
- Перевірка на відповідність критеріям синхронізації
- Обчислення поточного хешу (_calculate_file_hash)
- Порівняння з еталонним значенням
- Логування результатів
- Моніторинг змін

1.6.2 Застосування методів перевірки моніторингу

Основною ідеєю роботи системи є використання інтелектуального кешування результатів перевірок: хеш-суми файлів зберігаються у оперативній

пам’яті, що забезпечує оперативний доступ до них. Такий підхід дозволяє суттєво підвищити швидкість процесу верифікації, оскільки система здійснює повне сканування лише тих файлів, які були змінені від моменту останньої перевірки. Для зменшення навантаження на підсистему введення-виведення реалізовано механізм інкрементного оновлення кешу, який аналізує лише модифіковані ділянки дисків, використовуючи технологію журналу змін файлів (file change journal).

Пріоритезація перевірки базується на динамічних алгоритмах, що враховують як час останньої модифікації, так і ступінь критичності файлів для функціонування системи. Першочергово перевіряються файли із розширеннями, характерними для офісних документів (.docx, .xlsx) або баз даних (.db), оскільки саме вони найчастіше стають об’єктами кібератак. Оптимальний порядок сканування визначається автоматично на основі статистики змін і налаштувань пріоритетів, заданих користувачем. Для забезпечення стабільної роботи під значним навантаженням застосовується механізм пакетної обробки: файли перевіряються групами по 100 об’єктів за ітерацію, із можливістю динамічного регулювання швидкості сканування відповідно до поточного навантаження на систему. Між групами перевірених файлів система робить короткі паузи, що дозволяє знизити навантаження на дискову підсистему й уникнути її перевантаження.

1.6.3 Аналіз працездатності системи та огляд можливих сценаріїв

Аналіз працездатності системи реалізовано досить ґрунтовно й передбачає багатоступеневий підхід до виявлення аномалій. У разі виявлення невідповідності контрольних сум файлів система негайно блокує підозрілий файл, ініціює процес відновлення даних із резервної копії та детально фіксує інцидент у журналі аудиту з описом характеру аномалії.

Щодо тимчасових файлів типу .tmp чи ~, для них впроваджено окремий режим: під час стандартного сканування вони ігноруються, проте за необхідності можуть бути включені до перевірки на запит. Окрему увагу було приділено сценаріям, коли файли перебувають у відкритому стані в інших програмах. Для таких

випадків реалізовано механізм повторних спроб доступу, причому інтервали між ними зростають експоненційно, що підвищує ймовірність успішної обробки без надмірного навантаження на систему.

1.6.4 Огляд технічних особливостей роботи у файловій системі

В системі робота з файловою структурою реалізована через низькорівневі API, що сприяє максимально ефективному доступу до даних. Вона коректно обробляє символічні та жорсткі посилання, а також мережеві шляхи (UNC), забезпечуючи стабільний рівень контролю незалежно від типу доступу до інформації.

Для обробки помилок застосовано п’ятирівневу класифікаційну систему, яка дозволяє точно визначити характер проблеми й обрати найбільш оптимальну стратегію відновлення. У разі критичних збоїв система автоматично переходить у безпечний режим функціонування, зберігаючи необхідні дані для подальшої діагностики.

Адаптивні механізми управління дають змогу системі динамічно коригувати власні параметри. Зокрема, інтервали між перевірками автоматично збільшуються при підвищеному навантаженні, а розподіл ресурсів CPU оптимізується в реальному часі. Для управління оперативною пам’яттю застосовується інтелектуальний алгоритм, який аналізує поточне навантаження й розподіляє ресурси відповідно до визначених пріоритетів виконання завдань.

1.6.5 Робота з можливим виникненням незвичних ситуацій

Обробка нештатних ситуацій у системі здійснюється через сукупність механізмів, що забезпечують відновлення після мережевих збоїв, запобігання конфліктам версій файлів та контроль за використанням ресурсів. У разі втрати зв’язку з сервером система автоматично переходить у автономний режим із фіксацією всіх змін для подальшої синхронізації. Для вирішення конфліктів версій передбачено механізм візуалізації ланцюжка змін, що дозволяє адміністратору оперативно ідентифікувати джерело розбіжностей. Система моніторингу постійно відстежує використання процесора, операції введення-

виведення та читання/запису на диску. При досягненні критичних рівнів автоматично застосовуються обмеження, що забезпечує стабільність і передбачуваність функціонування системи.

1.7 Тестування та аналіз роботи програмної моделі

Етап тестування став ключовим у перевірці надійності та стабільності роботи моделі програми, оскільки саме на цій стадії програма пройшла комплексну перевірку у різноманітних умовах експлуатації. Процес тестування був ретельно спланований і включав численні сценарії використання, що дозволили виявити потенційно уразливі місця системи і знайти оптимальні шляхи їх усунення (рис1.8).

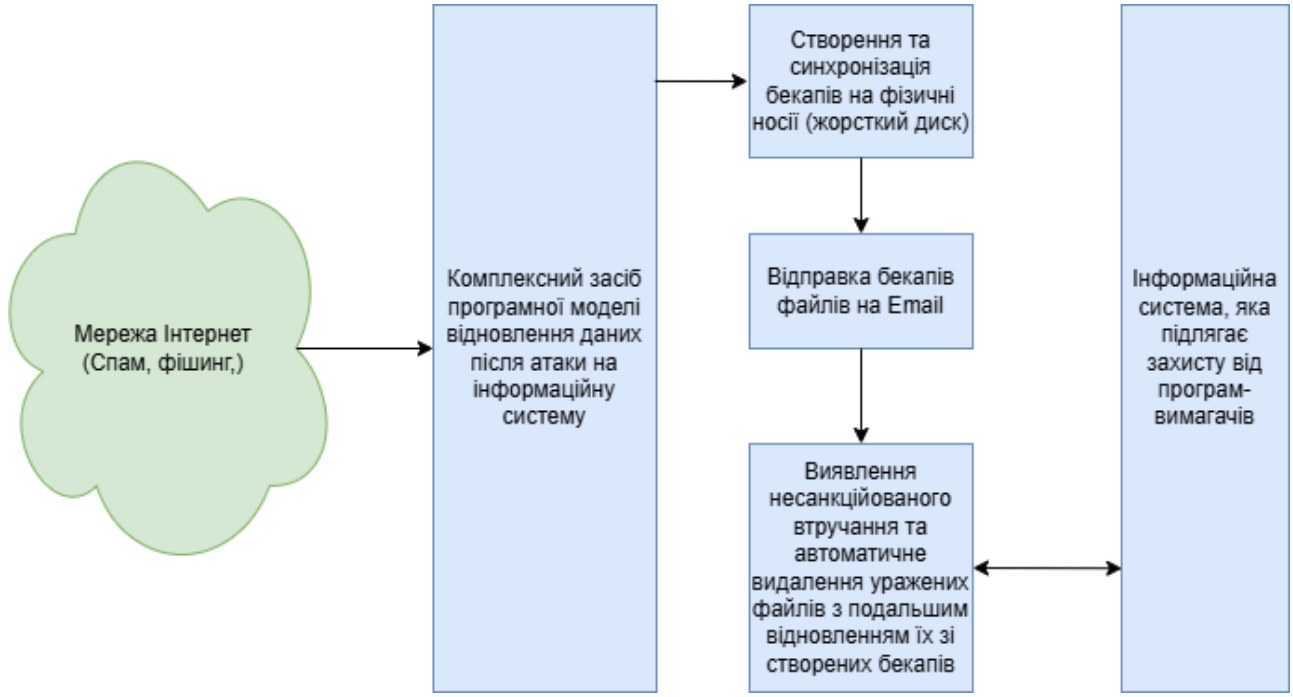


Рисунок 1.8 Схема відтворення послідовності дій програмної моделі відновлення після атаки на інформаційну систему

Особливу увагу під час тестування було приділено перевірці механізмів створення та обробки файлів. Було змоделювано різні ситуації надходження нових файлів у контрольовану папку.

Ці тести дозволили визначити оптимальні параметри для роботи системи чергування завдань і розподілу ресурсів, щоб уникнути перевантаження системи при інтенсивному використанні. Було встановлено, що програма здатна

ефективно обробляти до декількох тисяч файлів одночасно без значного впливу на продуктивність операційної системи.

Інтенсивне редагування вмісту файлів було невеликою частиною тестування. Для цього було розроблено сценарії, що імітували роботу з різними типами документів.

Тестування включало послідовне внесення змін, паралельне редагування одних і тих самих файлів. Це дозволило зробити оптимізацію алгоритмів синхронізації та зменшити ймовірність виникнення конфліктів версій при груповій роботі з документами.

Одним з найважливіших моментів тестування стала імітація атаки шифрувальника-вимагача. Для цього було розроблено спеціальне тестове середовище, яке точно відтворювало поведінку різних типів шкідливого програмного забезпечення.

Було використано сценарії поступового шифрування файлів, використовуючи алгоритми крипторгафії. У ході цих тестів було визначено, що система виявляє більшість спроб несанкціонованого доступу до файлів протягом перших секунд після початку атаки, що дозволило мінімізувати потенційні збитки. Однак деякі складні сценарії атак вимагали доопрацювання алгоритмів аналізу поведінки, що й було реалізовано на основі отриманих даних.

Перевірка стабільності програми при роботі з файлами різного обсягу та типу виявила необхідність оптимізації механізмів кешування та управління пам'яттю. Було встановлено, що при роботі з особливо великими файлами (понад 10 ГБ) система потребує додаткового часу для обчислення контрольних сум та виконання операцій синхронізації. Це призвело до впровадження прогресивного алгоритму обробки великих файлів, який розбиває їх на логічні блоки та обробляє їх послідовно, зменшуючи навантаження на апаратні ресурси.

Важливим результатом тестування стало виявлення необхідності вдосконалення системи обробки помилок. У процесі перевірки було задіяно різноманітні аварійні ситуації - від раптового відключення живлення до проблем з мережевим з'єднанням при роботі з віддаленими сховищами. Це дозволило

розробити надійні механізми відновлення після збоїв, які гарантують цілісність даних навіть у разі непередбачених обставин.

Система тепер здатна точно визначати стан синхронізації на момент збою та коректно продовжувати роботу після усунення проблеми, не створюючи конфліктів версій файлів.

Окремий цикл тестування був присвячений перевірці сумісності з різними версіями операційних систем та файлових систем. У ході тестування було визначено, що деякі особливості реалізації NTFS у нових версіях Windows можуть впливати на швидкість відстеження змін у великих каталогах.

Це спонукало до оптимізації алгоритмів моніторингу файлових подій, що значно підвищило продуктивність програми при роботі з великою кількістю файлів у єдиній структурі каталогів.

На основі результатів тестування було проведено доскональний аналіз продуктивності усіх компонентів системи. Це дозволило виявити певні неефективності в алгоритмах синхронізації. Вдосконалення цих алгоритмів значно зменшило час реакції системи на зміни та знизило загальне навантаження на процесор і дисковий підсистему.

Тестування також включало перевірку роботи програми в умовах обмежених апаратних ресурсів.

В ході роботи системи було знайдено рекомендовані мінімальні вимоги до системи, при яких програма зберігає свою функціональність без критичного зниження продуктивності основної роботи користувача.

Ці дані мають лише рекомендаційний характер щодо апаратного забезпечення для різних сценаріїв використання. А саме: процесор на 4 ядра та 6 (або 8) потоків та 8 ГБ оперативної пам’яті.

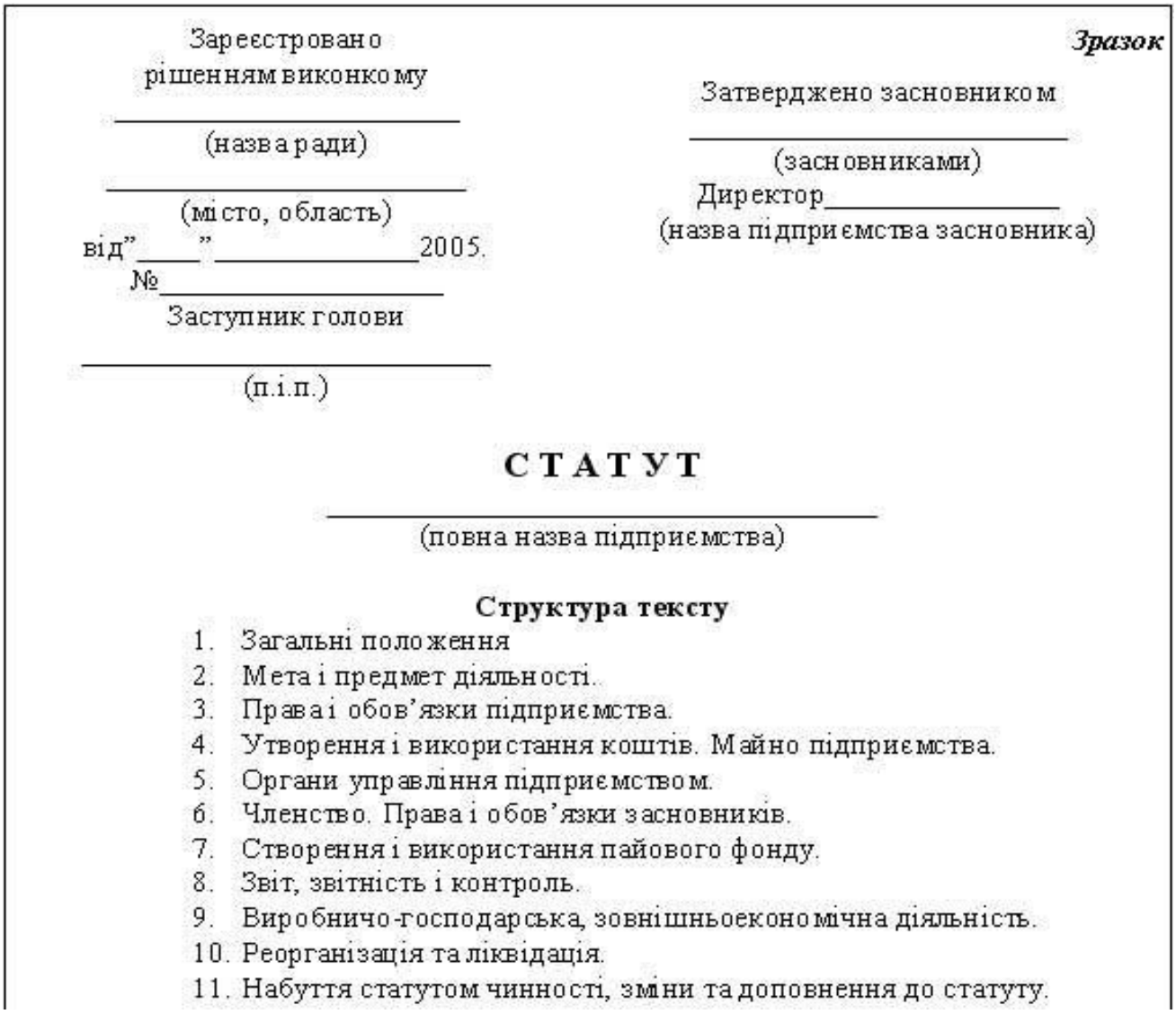
Після завершення всіх циклів тестування було проведено остаточну оптимізацію кодової бази, що дозволило скоротити час запуску програми та обсяг пам'яті, який вона використовує під час роботи. Одночасно було вдосконалено систему логування подій, що значно полегшило діагностику потенційних проблем у реальних умовах експлуатації.

1.7.1 Запуск роботи та перевірка працездатності програмної моделі

Щоб точно та наочно продемонструвати роботу мого дипломного проекту мені знадобиться використати іншу програму, що буде змінювати хеш файлів у вказаній папці та файловій системі. Ось демонстрація частини з коду програми, що змінює хеш файлу на інший:

```
def __init__(self, source_folder):
self.source_folder = source_folder
def _calculate_hash(self, file_path):
"""Знаходить поточний хеш файлу"""
hasher = hashlib.sha256()
with open(file_path, 'rb') as f:
while chunk := f.read(4096):
hasher.update(chunk)
return hasher.hexdigest()
def corrupt_random_bytes(self, file_path, num_bytes=10):
"""Замінює випадкові біти в файлі"""
file_size = os.path.getsize(file_path)
if file_size == 0:
return False
with open(file_path, 'r+b') as f:
for _ in range(num_bytes):
pos = random.randint(0, file_size - 1)
f.seek(pos)
f.write(bytes([random.randint(0, 255)]))
return True
if not all_files:
print("Немає файлів для змінення")
```

Для прикладу роботи програмної моделі у вихідній папці створюється новий файл з важливою інформацією та назвою «Квартальний звіт.docx» (рис.1.9), який відкривається без помилок й доступний до редагування.



Зареєстровано
рішенням виконкому
_____
(назва ради)
_____
(місто, область)
від"_____"_____2005.
№_____
Заступник голови
_____
(п.і.п.)

Зразок

Затверджено засновником
_____
(засновниками)
Директор_____
(назва підприємства засновника)

СТАТУТ

_____
(повна назва підприємства)

Структура тексту

1. Загальні положення
2. Мета і предмет діяльності.
3. Права і обов'язки підприємства.
4. Утворення і використання коштів. Майно підприємства.
5. Органи управління підприємством.
6. Членство. Права і обов'язки засновників.
7. Створення і використання пайового фонду.
8. Звіт, звітність і контроль.
9. Виробничо-господарська, зовнішньоекономічна діяльність.
10. Реорганізація та ліквідація.
11. Набуття статутом чинності, зміни та доповнення до статуту.

Рисунок 1.9 Приклад створення робочого файлу

Ініціація змінення хешу (за допомогою шифрувальника) цього файлу та повторна спроба скористатися інформацією й відкрити для його редагувати (рис. 1.10 та 1.11).

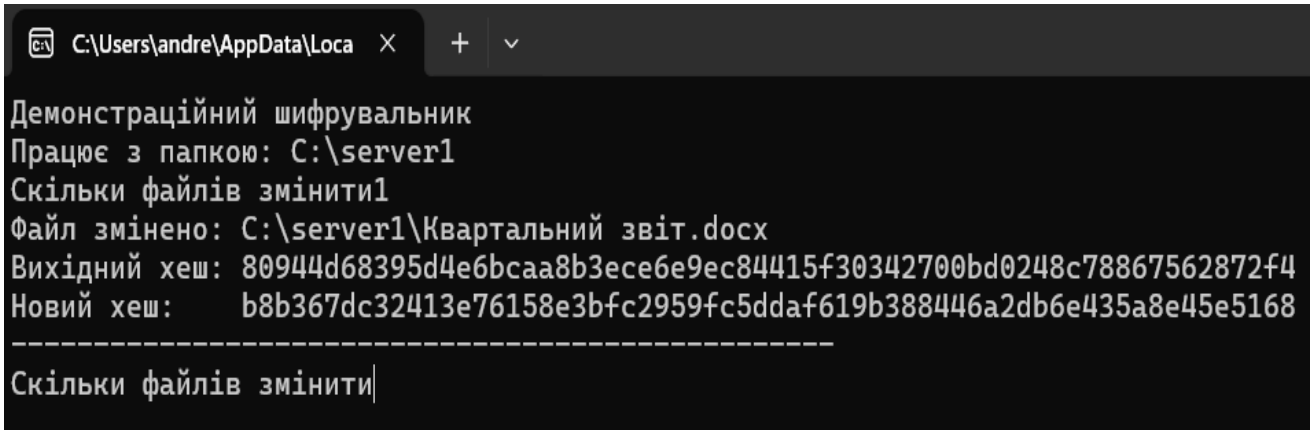


Рисунок 1.10 Демонстрація заміни хешу вказаного файлу на інший хеш

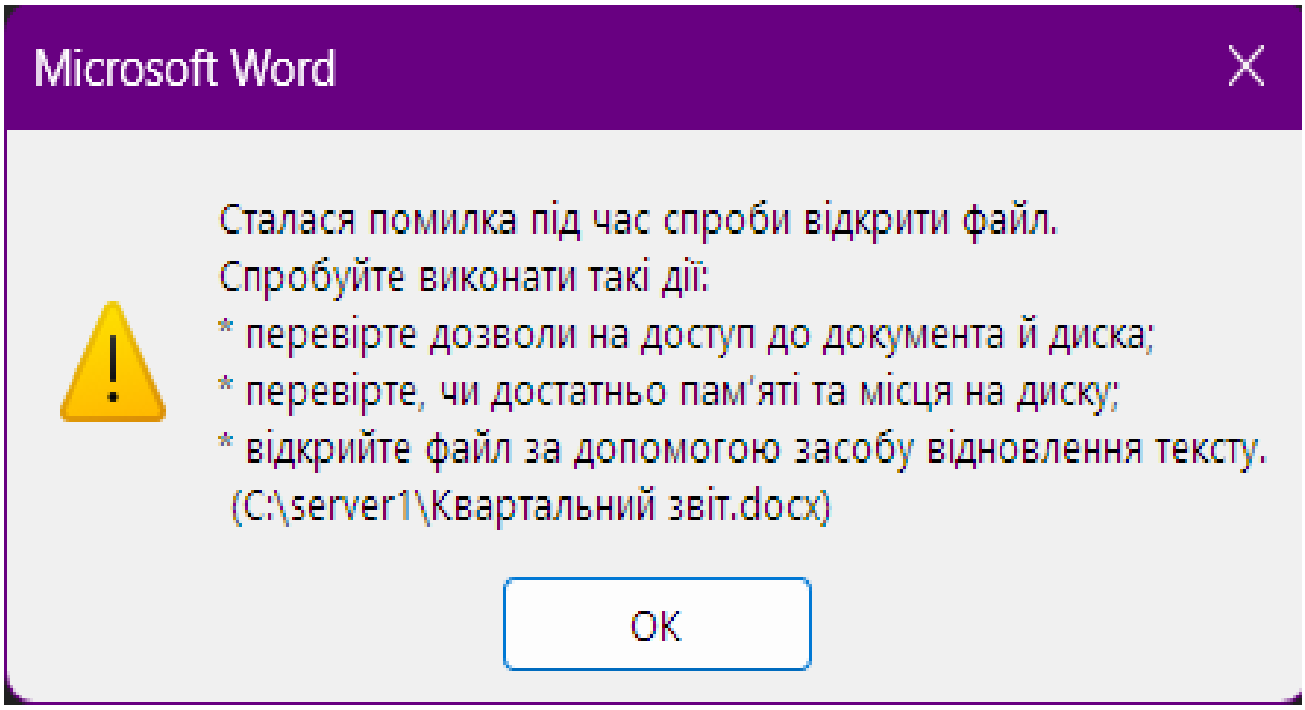


Рисунок 1.11 Демонстрація стану файлу, який був уражений

Запуск роботи моделі програми для синхронізації та відновлення даних; комплексне тестування її функціональності в умовах, що максимально наближені до реальних.

Система моніторингу ретельно аналізує вказану вихідну папку, перевіряючи наявність файлів та їх відповідність критеріям синхронізації. Під час цього процесу відбувається ініціалізація хеш-сум всіх існуючих у цій папці файлів, у даному випадку лише ‘Квартальний звіт.docx’ для демонстрації (рис.1.12). Після ураження система повністю видалила пошкоджений файл та відновила його з бекапів папки server2 та server3 (рис. 1.13).

Ініціалізація створення та надсилання програмною моделлю резервної копії на вказану електрону пошту за допомогою бібліотеки SMTP (рис.1.14)

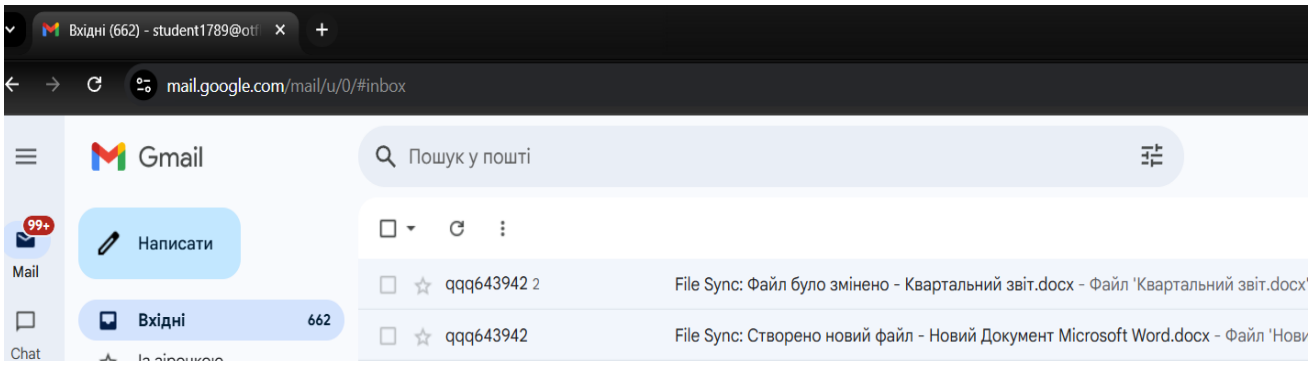


Рисунок 1.14 Синхронізація файлу на EMAIL

```
[2025-06-12 18:06:40] Ініціалізація хешів файлів...
[2025-06-12 18:06:40] Початок моніторингу...
Вихідна папка: C:\server1
Папки призначення: ['E:\\server2', 'F:\\server3']
Натисніть Ctrl+C щоб зупинити
[2025-06-12 18:06:40] Запуск періодичної перевірки цілісності...
[2025-06-12 18:06:48] Створено новий файл: C:\server1\Новий Документ Microsoft Word.docx
[2025-06-12 18:06:48] Помилка обчислення хешу C:\server1\Новий Документ Microsoft Word.docx: [Errno 13] Permission denie
d: 'C:\\server1\\Новий Документ Microsoft Word.docx'
[2025-06-12 18:06:48] Синхронізовано: Новий Документ Microsoft Word.docx → E:\server2\Новий Документ Microsoft Word.docx
[2025-06-12 18:06:48] Синхронізовано: Новий Документ Microsoft Word.docx → F:\server3\Новий Документ Microsoft Word.docx
[2025-06-12 18:06:51] Email надіслано: Створено новий файл - Новий Документ Microsoft Word.docx
[2025-06-12 18:07:50] Файл змінено: C:\server1\Квартальний звіт.docx
[2025-06-12 18:07:50] Синхронізовано: Квартальний звіт.docx → E:\server2\Квартальний звіт.docx
[2025-06-12 18:07:50] Синхронізовано: Квартальний звіт.docx → F:\server3\Квартальний звіт.docx
[2025-06-12 18:07:53] Email надіслано: Файл було змінено - Квартальний звіт.docx
[2025-06-12 18:07:53] Файл змінено: C:\server1\Квартальний звіт.docx
[2025-06-12 18:07:53] Синхронізовано: Квартальний звіт.docx → E:\server2\Квартальний звіт.docx
[2025-06-12 18:07:53] Синхронізовано: Квартальний звіт.docx → F:\server3\Квартальний звіт.docx
[2025-06-12 18:07:56] Email надіслано: Файл було змінено - Квартальний звіт.docx
[2025-06-12 18:07:56] Файл було видалено: C:\server1\~WRL0003.tmp
[2025-06-12 18:07:56] Файл було видалено: C:\server1\~$артальний звіт.docx
[2025-06-12 18:10:00] Файл змінено: C:\server1\Квартальний звіт.docx
[2025-06-12 18:10:00] Виявлено несанкціоновану зміну файлу: C:\server1\Квартальний звіт.docx
[2025-06-12 18:10:00] Файл відновлено з резервної копії: C:\server1\Квартальний звіт.docx
[2025-06-12 18:10:00] Синхронізовано: Квартальний звіт.docx → E:\server2\Квартальний звіт.docx
[2025-06-12 18:10:00] Синхронізовано: Квартальний звіт.docx → F:\server3\Квартальний звіт.docx
```

Рисунок 1.12 Демонстрування дії та моніторингу програмної моделі відновлення даних

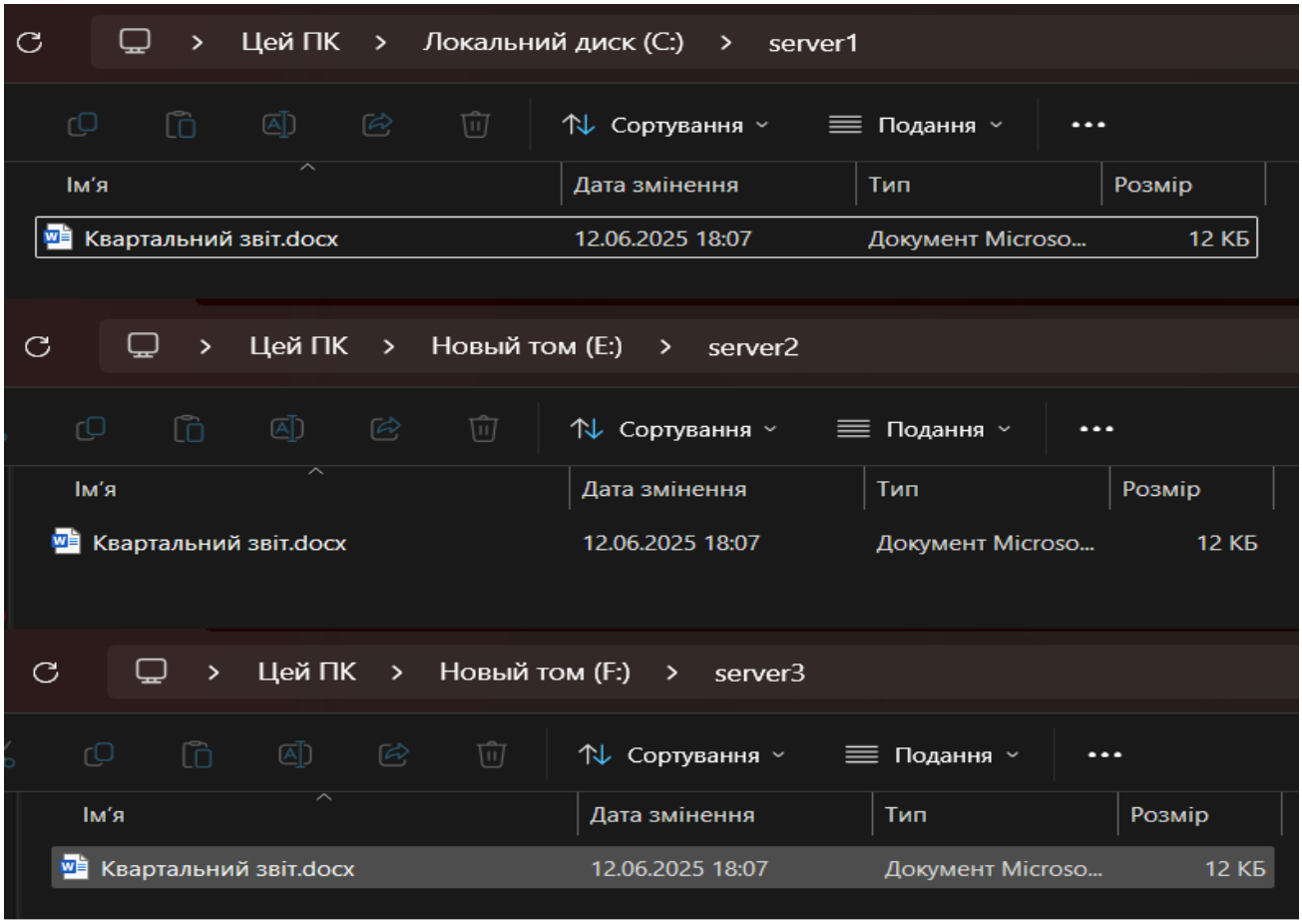


Рисунок 1.13 Збереження резервних копій до вказаних в коді шляхів.

1.8 Перспективи розвитку поекту

У цьому проєкті закладено значний потенціал для розвитку: з базового засобу синхронізації він може перетворитися на комплексну платформу для управління даними, якщо реалізувати низку покращень.

Питання безпеки є ключовим. Актуальним виглядає впровадження сучасних алгоритмів шифрування, таких як AES-256, для забезпечення надійного захисту даних під час передачі й зберігання.

Важливим етапом вдосконалення також може стати застосування двоетапної аутентифікації, що істотно знизить ризики несанкціонованого доступу до системних налаштувань.

Продуктивність системи також потребує оптимізації. Впровадження технології дельта-синхронізації, коли передаються лише змінені частини файлів замість цілих документів, дозволить значно скоротити час виконання операцій та зменшити навантаження на мережу.

Ще одним перспективним напрямом може стати є інтеграція із хмарними сервісами на кшталт Google Drive, Dropbox або AWS S3. Це дозволить створити гібридні рішення для зберігання даних — частково локально, частково у хмарі.

Також доцільно передбачити автоматичне архівування попередніх версій документів для відстеження історії змін.

Варто звернути увагу на можливість покращення інтерфейсу. Візуалізація процесу синхронізації у вигляді графіків чи діаграм зробить взаємодію із системою більш інтуїтивною. Автоматичні звіти на електронну пошту з інформацією про успішні операції й можливі проблеми (наприклад, нестачу пам’яті) значно підвищать рівень зручності.

Впровадження крос-платформенності є важливою умовою для розширення аудиторії користувачів. Додаткова підтримка Linux і macOS, а також впровадження веб-інтерфейсу чи мобільного додатку забезпечать гнучкість у використанні, що особливо буде актуальним рішенням для корпоративного сегменту. Загалом, комплексна реалізація цих удосконалень здатна вивести проєкт на якісно новий, сучасний рівень.

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

В даному дипломному проекті було розроблено програмний продукт (ПП) - програмну модель з відновлення даних після атаки на інформаційну систему.

Програмна модель була зпроєктована з урахуванням сучасних підходів до розробки, які забезпечують її ефективність й сприяють зручності використання.

Якість роботи відновлення даних після атаки на інформаційну систему вирізняється надійністю та точністю, відповідністю функціональним вимогам . У розділі було розгорнуто розглянено економічний аспект розробки: витрати зусиль, ресурсів, та часу необхідних для створення програми.

2.2 Розрахунок ціни програмного продукту нормативним методом

2.2.1 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного забезпечення залежить від низки чинників, зокрема рівня складності проєкту, обсягу необхідних робіт, трудомісткості окремих завдань і кваліфікації розробників. Для попередньої оцінки масштабів часто застосовують метод структурного порівняння — його суть полягає в аналізі аналогічних проєктів, що вже були реалізовані, з метою приблизного визначення необхідного обсягу коду. Даний підхід дозволяє отримати орієнтовні оцінки, втім, точність результатів значною мірою обумовлена адекватністю вибраних аналогів і коректністю проведеного порівняння.

Найбільш відповідний до функціоналу застосунку є . ПП автоматизованих розрахунків (1300-8600) машинних команд.На основі обраного аналогу, для якого відомий орієнтовний обсяг коду V_0 (2.3) в умовних машинних командах, далі робиться розрахунок трудомісткості розробки. Обсяг ПП становить 2.00 тис. умов. машинних команд із нормою часу 244 год/люд.

Враховуючи отримане значення, встановлюється норма часу на створення аналогічного програмного забезпечення. З урахуванням умов реалізації проєкту, це значення встановлюється відповідним коефіцієнтом (Кк = 0,7÷0,8). Таким чином, розрахункова трудомісткість становить: $T^a p = 244 \times 0,7 = 170,8$ год/люд.

Трудомісткість визначається окремо для кожного етапу розробки, спираючись на дані про трудовитрати для подібних програмних продуктів. При цьому враховується складність реалізації, рівень новизни проекту та ступінь використання типових програмних компонентів:

- Розробка технічного завдання $T_{ТЗ} = T^a p \times L_1 \times K_H$
- Розробка технічного проекту $T_{ТП} = T^a p \times L_2 \times K_H$
- Розробка робочого проекту $T_{РП} = T^a p \times L_3 \times K_H \times K_T$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.3.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.4.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.5.). Обрані відповідні варіанти виділено зеленимкольором.

Таблиця 2.3 Значення питомих коефіцієнтів трудомісткості стадії

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Таблиця 2.4 Значення поправочного коефіцієнта

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Оскільки розроблювана система є програмним забезпеченням, що має існуючі аналоги, для мого ПЗ встановлено код ступеня новизни “В”, а коефіцієнт $К_н$ = 0,7.

Відповідно до таблиці 2.4, маючи цей код новизни, можна перейти до визначення питомих коефіцієнтів трудомісткості за таблицею 2.3.

L_1=0,12

L_2= 0,11

L_3= 0,61

Таблиця 2.5 Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

У розробленому програмному продукті використовується від 40 до 60 відсотків існуючих функцій, це значить, що $К_т$= 0,7 Тепер потрібно розрахувати трудомісткість по кожному етапу окремо:

- Трудомісткість технічного завдання

$Т_{тз}=T^a p *L_1*К_н$= 170,8*0,12*0,7 = 14,3 (люд/годин)

- Трудомісткість розробки технічного проекту

$Т_{тп}=T^a p *L_2*К_н$ = 170,8*0,11*0,7 = 13,1 (люд/годин)

- Трудомісткість розробки робочого проекту

$Т_{рп}=T^a p *L_3*К_н*К_т$= 170,8*0,61*0,7 = 72,9 (люд/годин)

Для подальших розрахунків необхідно розрахувати кількість аркушів, витраченого на кожен етап. Технічне завдання: $N_{тз}$= 2 сторінки, технічний проєкт :$N_{тп}$= 50 сторінок, робочий проект – $N_{рп}$= 10 сторінок , пояснювальна записка $N_{пз}$= 20 сторінок – технічне завдання, розробка технічного проекту, розробка робочого проекту, пояснювальна записка відповідно. Розрахунок зведений у таблицю 2.6

Таблиця 2.6 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, години.		
1.ТЗ	$Т_{Ртз}$=14,3	$Т_{кк}=0,7*N_{тз}$= 0,7*2= 1,4	$Т_{нк}=0,15*N_{тз}$=0,15*2=0,3
2.Розробка ТП	$Т_{Ртп}$=13,1	$Т_{кк}=0,7*N_{тп}$= 0,7*50= 35	$Т_{нк}=0,15*N_{тп}$=0,15*50=7,5
3.Розробка РП	$Т_{Ррп}$=72,9	$Т_{кк}=0,7*N_{рп}$= 0,7*10= 7	$Т_{нк}=0,15*N_{рп}$=0,15*10=1,5

4.Розробка пояснювальної записки	$T_{рпз}=1,5*N_{пз}=$ 1,5*20 =30	$T_{кк}=0,7*N_{пз}=20*0,7=$ 14	$T_{нк}=0,15*N_{пз}=0,15*20=3$
Усього, в т.ч.:	$T_{пп}=200$		
- на розробку	$\Sigma T_р=130,3$		
- контроль керівника		$\Sigma T_{кк}=57,4$	
- нормоконтроль			$\Sigma T_{нк}=12,3$

На основі таблиці 2.6 розрахуємо тривалість розробки в роках:

$T_{пп}=T/(8.0*0.73*360)=$ 170,8/(8.0*0.73*360)= 0,081 (р), де:

8,0 – тривалість робочого дня;

0,73 – коефіцієнт перекладу в календарні дні;

2.2.2 Розрахунок ціни програмного продукту

У цьому розділі детально розглянуто розрахунок вартості програмного продукту, зокрема: основної заробітної плати, матеріальних витрат, витрат на машино-години та загальних витрат. Відомості щодо заробітних плат наведені у таблиці 2.7. Важливо зазначити, що з 1 січня 2025 року мінімальна місячна заробітна плата в Україні складає 8000 грн, а погодинна ставка — 46 грн.

Таблиця 2.7 Розрахунок основної заробітної плати виконавців.

Найменування робіт	Трудомісткість робіт, роб.години	Годинна тарифна ставка,грн..	Розрахунок, грн.
1.Розробка ПП	$\Sigma T_р=130,3$	46	5 993,8
2.Контроль керівника	$\Sigma T_{кк}=57,4$	53	3 042,2
3.Нормоконт-роль	$\Sigma T_{кк}=12,3$	50	615
Усього ($З_о$)			Σ Зо= 9 651

Виконаємо розрахунок матеріальних витрат, необхідних для створення програмного продукту. Підсумкові дані наведено в таблиці 2.8.

Таблиця 2.8 Розрахунок матеріальних витрат на розробку ПО

Найменування матеріальних витрат	Тип, модель	Кількість, шт	Ціна одиниці, грн.	Вартість, грн.
Папір А4		75	4	300
Транспортно – заготівельні Витрати 10%				$Втр\_з = 0{,}1 \times Вм1 =$ 0,1*300 = 30
Усього				$В_м = В_{м1} + В_{тр\_з} = 330$

Виходячи з отриманих даних щодо окремих статей витрат, було складено калькуляцію планової собівартості підприємства відповідно до форми. таблиця 2.9.

Таблиця 2.9 Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	330	$В_м$ (див. табл. 2.8)
2. Основна заробітна плата	9 651	$З_о$ (див. табл. 2.7)
3.Додаткова заробітна плата	965,1	$Зд = 0{,}1 \times Зо = 0{,}1*$ 9 651
4.Відрахування до єдиного фонду соціального внеску	2 335,52	$Вє.с.в. = 0{,}22 \times (Зо + Зд) = 0{,}22*(9\ 651+965{,}1)$
5. Накладні витрати	5 790,6	$Внак. = 0{,}6 \times Зо = 0{,}6*9651$
6. Повна собівартість	19 072,22	$С_{пов} = В_м + З_о + З_д + Вє.с.в. + Внак. = 330+$ 9 651+965,1+2 335,52+5 790,6

Розмір прибутку, що включається в ціну, визначається наступною формулою:

$П=(С_{пов}*Р)/100 = (19\ 072{,}22 *10)/100 = 1907{,}2$ грн

Де Р– плановий рівень рентабельності (10-15%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$Ц_о=С_{пов}+П= 19\ 072{,}22+1907{,}2= 20\ 979{,}42$ грн

Податок на додану вартість визначається по наступній формулі:

$ПДВ = 0.2 * Ц_о = 20\ 979{,}42*0{,}2 = 4\ 195{,}8$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$Ц_р=Ц_о+ПДВ= 20\ 979{,}42+20\ 979{,}42*0{,}2 = 25\ 175{,}3$ грн

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

У цьому розділі, присвяченому охороні праці, розглядаються ключові питання створення безпечних і комфортних умов для розробників програмного забезпечення. Хоча професійна діяльність у цій сфері, на перший погляд, не пов’язана з типовими виробничими ризиками, все ж таки існує низка важливих аспектів, ігнорування яких може негативно вплинути на здоров’я працівників.

Особлива увага приділяється ергономіці робочого місця: правильна організація простору, відповідне розміщення комп’ютера, якість освітлення та дотримання електробезпеки є обов’язковими для забезпечення безпеки персоналу. Тривала робота за комп’ютером може призводити до перевтоми очей, напруження опорно-рухового апарату, а також зниження загальної працездатності, тому важливо враховувати ці фактори під час планування робочого процесу та організації робочого середовища.

Нижче також розглядаються вимоги до мікроклімату в приміщенні, рівня шуму, освітлення, а також заходи пожежної й електричної безпеки. Дотримання цих стандартів є не лише вимогою законодавства та галузевих нормативів, а й необхідною умовою для збереження здоров’я працівників та підвищення ефективності їхньої роботи.

3.1 Опис шкідливих факторів, які впливають на розробника

Робота розробника програмного забезпечення супроводжується низкою несприятливих факторів, які можуть негативно впливати на здоров’я. Зокрема, тривале перебування перед монітором створює значне навантаження на органи зору: виникає втома очей, можливе тимчасове чи навіть стійке зниження гостроти зору.

Ще однією проблемою є статична поза під час роботи. Тривале сидіння за комп’ютером сприяє напруженню м’язів спини, шиї та рук, що з часом може призводити до розвитку захворювань опорно-рухового апарату. З метою профілактики рекомендується робити перерви та дотримуватись принципів ергономічної організації робочого місця.

Окрему увагу слід приділяти дотриманню правил електробезпеки, оскільки робота з комп’ютерною технікою завжди пов’язана з певними ризиками ураження електричним струмом у разі несправності обладнання. Крім цього, важливими є показники мікроклімату — температура, вологість і рівень шуму. Порушення цих параметрів може знижувати працездатність працівника та погіршувати його загальне самопочуття.

Таким чином, дотримання рекомендацій щодо захисту здоров’я, ергономіки та безпеки є необхідною умовою ефективної та безпечної діяльності програміста.

3.2 Вимоги з гігієни у приміщенні

3.2.1 Вимоги до приміщення

Відповідно до вищенаведеного тексту, для організації робочого місця розробника програмного забезпечення необхідно дотримуватись ряду основних вимог.

По-перше, площа приміщення має становити щонайменше 6 м² на одну особу, що забезпечує достатній простір для розміщення обладнання та комфортного пересування.

По-друге, висота стелі повинна бути не меншою за 3,2 метри, що сприяє належній циркуляції повітря та створює здорові умови для праці. Окрім того, обов’язковим є облаштування системи вентиляції для подачі свіжого повітря та видалення надлишкового тепла, яке продукується комп’ютерною технікою.

Дотримання цих санітарних та гігієнічних норм є необхідним для забезпечення безпеки та комфорту співробітників у процесі виконання ними професійних обов’язків.

3.2.2 Вимоги до рівня шуму в приміщенні

Під час роботи за комп’ютером шум не є основним шкідливим фактором, проте його рівень все одно впливає на самопочуття та працездатність користувача. Джерелами шуму можуть бути системний блок, вентилятори, принтери тощо.

Тривалий вплив підвищеного рівня шуму призводить до втоми, зниження концентрації уваги та продуктивності праці.

Згідно із санітарними нормами, допустимий рівень шуму на робочому місці користувача ПК не повинен перевищувати 50 дБ. Такий рівень вважається безпечним і комфортним для тривалої роботи.

З метою зменшення впливу шуму рекомендується використовувати комп'ютерне обладнання з низьким рівнем шуму, розташовувати його на певній відстані від робочого місця, а також застосовувати в приміщенні шумопоглинальні матеріали.

3.2.3 Вимоги до освітлення в приміщенні

Освітлення робочого місця є ключовим фактором для збереження зору та забезпечення оптимальних умов праці. Недостатній або надмірний рівень освітленості може спричинити втому очей, зниження працездатності та погіршення загального самопочуття.

Відповідно до встановлених норм, освітленість поверхні робочого столу для роботи за комп'ютером має становити не менше 300 люксів. Важливо, щоб освітлення було рівномірним, без різких тіней та відблисків на моніторі, що допомагає уникнути додаткового навантаження на зір.

Рекомендується поєднувати природне і штучне освітлення. При використанні штучних джерел світла доцільно обирати люмінесцентні або LED-світильники з нейтральною колірною температурою (приблизно 4000-5000 К). Робоче місце слід розташовувати таким чином, щоб уникати прямих сонячних променів на екран монітора, що сприяє підвищенню комфорту під час роботи.

3.2.4 Вимоги з електробезпеки

Робота розробника програмного забезпечення тісно пов'язана з постійним використанням комп'ютерної техніки, зокрема комп'ютерів, моніторів та периферійних пристроїв. У зв'язку з цим надзвичайно важливо дотримуватись правил електробезпеки для мінімізації ризиків ураження електричним струмом чи виникнення аварійних ситуацій.

Серед основних вимог виділяються наступні:

- Вся апаратура повинна підключатися лише до справних електромереж із обов’язковим заземленням.
- Використання подовжувачів та розеток допускається лише за умови їхньої цілісності й відсутності механічних пошкоджень.
- Прокладання кабелів у місцях, де вони можуть бути пошкоджені (наприклад, у проходах), не допускається.
- У разі виявлення несправностей у роботі обладнання його слід негайно вимкнути та повідомити відповідальну особу.
- У приміщенні обов’язково повинні бути засоби пожежогасіння, призначені для гасіння електрообладнання (наприклад, вогнегасник із вуглекислотою).
- Електрообладнання повинне бути технічно справним і проходити регулярні перевірки на відсутність пошкоджень. Кабелі, розетки й подовжувачі мають бути неушкодженими, без слідів оплавлення чи тріщин.
- Забороняється перевантаження електромережі, зокрема підключення великої кількості пристроїв до однієї розетки без відповідних розрахунків навантаження.

3.2.5 Вимоги до мікроклімату

Під час роботи за комп’ютером важливо забезпечити належний мікроклімат у приміщенні, оскільки саме він впливає на самопочуття та працездатність людини. Основні параметри, яких варто дотримуватися: температура повітря повинна знаходитись у межах від 18 до 24 °C, відносна вологість — у межах 40-60%, а швидкість руху повітря — не перевищувати 0,1 м/с.

Регулярне провітрювання приміщення є обов’язковим; у разі підвищеного тепловиділення або у спекотний період доцільно використовувати системи кондиціонування або вентиляції для підтримки оптимальних параметрів мікроклімату. Недотримання цих вимог може призвести до швидкої втоми, зниження концентрації уваги та загального дискомфорту під час роботи.

Важливо забезпечити ефективну вентиляцію для надходження свіжого повітря й видалення надлишкового тепла, що виділяється комп’ютерною технікою. Існує кілька типів вентиляції: природна (завдяки відкриванню вікон,

фрамуг, вентиляційних решіток), яка підходить для приміщень із невеликою кількістю обладнання; примусова (механічна), яка використовує вентилятори та витяжки для постійного повітрообміну незалежно від зовнішніх умов — рекомендована для офісів із великою кількістю комп’ютерів або серверного обладнання; та комбінована, що поєднує обидва підходи — наприклад, більшу частину часу використовується природна вентиляція, але за потреби вмикаються механічні системи.

3.3 Пожежна безпека

Організація робочого місця вимагає суворого дотримання правил пожежної безпеки. Попри те, що основна діяльність здійснюється за комп’ютером, експлуатація електрообладнання завжди пов’язана з певними ризиками займання, особливо у випадку несправностей або порушення правил користування.

Ключові вимоги пожежної безпеки включають наступне:

- Первинні засоби пожежогасіння повинні бути розміщені у легкодоступних місцях. Найдоцільніше використовувати вуглекислотні або порошкові вогнегасники, оскільки вони підходять для гасіння електроустановок під напругою.
- У приміщенні необхідно передбачити наявність пожежної сигналізації та чітко вказати шляхи евакуації. Персонал повинен бути ознайомлений із планом евакуації та порядком дій у разі пожежі.
- Використання легкозаймистих матеріалів поблизу комп’ютерної техніки категорично заборонено, зокрема це стосується паперових виробів чи горючих рідин.
- Керівники повинні регулярно проводити інструктажі з пожежної безпеки й перевіряти готовність приміщення до надзвичайних ситуацій.

ВИСНОВКИ

Представлена програмна модель являє собою цілісний інструмент для захисту даних із трьома ключовими функціями, а саме: механізм миттєвого резервного копіювання, постійне спостереження за цілісністю файлів, а також автоматичне відновлення інформації. Такий механізм дій забезпечує комплексний захист даних, де кожен елемент доповнює й підсилює інші у протидії із кіберзагрозам.

Основна перевага цього рішення - це саме універсальність та адаптивність до експлуатаційних умов. Для малого й середнього бізнесу та звичайних користувачів ця програмна модель може стати непоганою альтернативою дорогим корпоративним системам резервного копіювання, які часто вимагають великих витрат на ліцензійне програмне забезпечення, виділені сервери та послуги IT-фахівців для обслуговування. У порівнянні зі складними системами, розроблений інструмент не потребує спеціальних технічних знань для розгортання і експлуатації, що робить його доступним для використання звичайних користувачів із обмеженими технічними ресурсами.

Технологічну особливість рішення було зумовлена в необхідності поєднання інтерфейсу простоти з надійним функціоналом. Професійні системи резервного копіювання в більшості своїй мають складні налаштування та потребують попередньої підготовки для того, щоб їх ефективно застосувати на практиці. Ця ж програмна модель може запропонувати простий інтерфейс зі стійким принципом захисту інформації. Користувачу лише залишається налаштувати параметри моніторингу, вибрати критичні папки для моніторингу та обрати періодичність перевірок - не поглиблюючись у в технічні подробиці .

Крім того, забезпечення прозорості усіх дій з файлами: деталізоване логування та сповіщення користувача на електронну пошту обов’язково буде нагадувати всі події, пов’язані з файловою системою, що в свою чергу дасть нагоду реагувати на потенційні загрози за допомогою сповіщень для користувача.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Семенов А. П., Іваненко В. С. Мережева безпека та кіберзахист: монографія. — Харків: ХНЕУ, 2023.
2. Ковальчук Ю. П., Шевченко Л. І. Основи інформаційної безпеки: навч. посібник — Київ: НТУУ «КПІ», 2021.
3. Dan Boneh, Victor Shoup – A Graduate Course in Applied Cryptography – навч. посібник з криптографії
4. Laurens Van Houtven – Crypto 101 – навч. підручник з основ криптографії
5. Datatracker. RFC 5321 – SMTP Protocol [Електронний ресурс] - Режим доступу: https://datatracker.ietf.org/doc/html/rfc5321 Дата звернення 20.04.2025
6. Automate the Boring Stuff with Python [Електронний ресурс] - Режим доступу: https://automatetheboringstuff.com/ Дата звернення 21.04.2025
7. Wikipedia.org. Simple Mail Transfer Protocol [Електронний ресурс] -Режим доступу: https://uk.wikipedia.org/wiki/SMTP Дата звернення 21.05.2025
8. Wikipedia.org. Сімейство хешування SHA-2 [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/SHA-2 Дата звернення 23.05.2025
9. Wikipedia.org. Acronis True Image [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Acronis_True_Image Дата звернення 24.05.2025
10. Wikipedia.org. Програма-вимагач (Ransomware) [Електронний ресурс] - Режим доступу: https://uk.wikipedia.org/wiki/Ransomware#cite_note-7 Дата звернення 25.05.2025
11. GitHub.com (python-ebook, cryptography-book) [Електронний ресурс] - Режим доступу: https://github.com/search?q=free+book+python Дата звернення 25.04.2025

ДОДАТОК А. Лістинг коду програмної моделі відновлення

```
class SyncEventHandler(FileSystemEventHandler):
    def __init__(self, source_folder, destination_folders, extensions_to_sync):
        super().__init__()
        self.source_folder = source_folder
        self.destination_folders = destination_folders
        self.extensions_to_sync = [ext.lower() for ext in extensions_to_sync]
        self.last_processed_files = {}
        self.file_hashes = {}
        self.last_integrity_check = 0
        self._initialize_hashes()
    def _initialize_hashes(self):
        """Ініціалізує хеші для всіх файлів у вихідній та цільових папках"""
        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Ініціалізація хешів
файлів...")
        all_folders = [self.source_folder] + self.destination_folders
        for folder in all_folders:
            for root, _, files in os.walk(folder):
                for file in files:
                    file_path = os.path.join(root, file)
                    if self._should_sync(file_path):
                        try:
                            file_hash = self._calculate_file_hash(file_path)
                            rel_path = os.path.relpath(file_path, folder)
                            if rel_path not in self.file_hashes:
                                self.file_hashes[rel_path] = file_hash
                        except Exception as e:
                            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка
ініціалізації хешу для {file_path}: {e}")
    def _calculate_file_hash(self, file_path, block_size=65536):
        """Обчислює хеш файлу за вказаним алгоритмом"""
        try:
            hasher = hashlib.new(HASH_ALGORITHM)
            if not os.path.exists(file_path):
                raise FileNotFoundError(f"Файл не існує: {file_path}")
            with open(file_path, 'rb') as f:
                buf = f.read(block_size)
                while len(buf) > 0:
                    hasher.update(buf)
                    buf = f.read(block_size)
            return hasher.hexdigest()
        except FileNotFoundError:
            raise
        except Exception as e:
```

```
            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка обчислення хешу {file_path}: {e}")
            return None
    def _check_file_integrity(self, file_path):
        """Перевіряє, чи відповідає файл своєму відомому хешу"""
        rel_path = os.path.relpath(file_path, self.source_folder)
        if rel_path not in self.file_hashes:
            current_hash = self._calculate_file_hash(file_path)
            if current_hash:
                self.file_hashes[rel_path] = current_hash
            return True
        try:
            current_hash = self._calculate_file_hash(file_path)
            if current_hash is None:
                return False
            return current_hash == self.file_hashes[rel_path]
        except Exception as e:
            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка перевірки цілісності {file_path}: {e}")
            return False
    def _restore_file_from_backup(self, file_path):
        """Відновлює файл з резервної копії, якщо вона доступна"""
        rel_path = os.path.relpath(file_path, self.source_folder)
        original_hash = self.file_hashes.get(rel_path)
            if not original_hash:
            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Немає збереженого хешу для {file_path}")
            return False
        for backup_folder in self.destination_folders:
            backup_path = os.path.join(backup_folder, rel_path)
            if os.path.exists(backup_path):
                try:
                    backup_hash = self._calculate_file_hash(backup_path)
                    if backup_hash == original_hash:
                        os.makedirs(os.path.dirname(file_path), exist_ok=True)
                        shutil.copy2(backup_path, file_path)
                        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Файл відновлено з резервної копії: {file_path}")
                        return True
                except Exception as e:
                    print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка перевірки/відновлення резервної копії {backup_path}: {e}")
        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Не знайдено придатної резервної копії для {file_path}")
        return False
```

```
    def _periodic_integrity_check(self):
        """Періодично перевіряє цілісність усіх файлів"""
        current_time = time.time()
        if current_time - self.last_integrity_check < INTEGRITY_CHECK_INTERVAL:
            return
        self.last_integrity_check = current_time
        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Запуск періодичної
перевірки цілісності...")
        for root, _, files in os.walk(self.source_folder):
            for file in files:
                file_path = os.path.join(root, file)
                if self._should_sync(file_path):
                    if not self._check_file_integrity(file_path):
                        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Виявлено
пошкодження файлу: {file_path}")
                        if not self._restore_file_from_backup(file_path):
                            try:
                                os.remove(file_path)
                                print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Видалено
пошкоджений файл: {file_path}")
                                self._sync_deletion(file_path)
                            except Exception as e:
                                print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка
видалення пошкодженого файлу {file_path}: {e}")
    def _sync_deletion(self, source_path):
        """Синхронізує видалення файлу з резервними копіями"""
        if not os.path.exists(source_path):
            rel_path = os.path.relpath(source_path, self.source_folder)
            for dest_folder in self.destination_folders:
                dest_path = os.path.join(dest_folder, rel_path)
                if os.path.exists(dest_path):
                    try:
                        os.remove(dest_path)
                        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Видалено файл з
резервної копії: {dest_path}")
                    except Exception as e:
                        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка
видалення файлу з резервної копії {dest_path}: {e}")
  def on_modified(self, event):
        """Обробляє подію зміни файлу"""
        if not event.is_directory and self._should_sync(event.src_path):
            if not os.path.exists(event.src_path):
                return
            current_time = time.time()
            if (event.src_path in self.last_processed_files and
```

```
            current_time - self.last_processed_files[event.src_path] < 2):
            return
        self.last_processed_files[event.src_path] = current_time
        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Файл змінено:
{event.src_path}")
            try:
            if not self._check_file_integrity(event.src_path):
                print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Виявлено
несанкціоновану зміну файлу: {event.src_path}")
                if self._restore_file_from_backup(event.src_path):
    def _send_email_with_attachment(self, file_path, subject):
        """Надсилає email з прикріпленим файлом"""
        if not os.path.exists(file_path):
            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Файл для email не
знайдено: {file_path}")
            return
        for attempt in range(MAX_EMAIL_RETRIES):
            try:
                msg = MIMEMultipart()
                msg["From"] = EMAIL_LOGIN
                msg["To"] = RECIPIENT_EMAIL
                msg["Subject"] = f"File Sync: {subject} - {os.path.basename(file_path)}"
                body = f"Файл '{os.path.basename(file_path)}' був {subject.lower()} у
системі синхронізації."
                msg.attach(MIMEText(body, "plain"))
                with open(file_path, "rb") as attachment:
                    part = MIMEBase("application", "octet-stream")
                    part.set_payload(attachment.read())
                    encoders.encode_base64(part)
                    part.add_header(
                        "Content-Disposition",
                        f"attachment; filename= {os.path.basename(file_path)}",)
                    msg.attach(part)
                with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
                    server.starttls()
                    server.login(EMAIL_LOGIN, EMAIL_PASSWORD)
                    server.send_message(msg)
                    print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Email надіслано:
{subject} - {os.path.basename(file_path)}")
                break
            except smtplib.SMTPException as e:
                if attempt == MAX_EMAIL_RETRIES - 1:
                    print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Не вдалося
надіслати email після {MAX_EMAIL_RETRIES} спроб: {e}")
                else:
```

```
                time.sleep(2)
            except Exception as e:
                print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка email: {e}")
                break
if __name__ == "__main__":
    if not os.path.exists(SOURCE_FOLDER):
        print(f"Помилка: Вихідна папка '{SOURCE_FOLDER}' не існує.")
        exit()
    valid_destinations = []
    for folder in DESTINATION_FOLDERS:
        try:
            if not os.path.exists(folder):
                os.makedirs(folder)
                print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Створено папку
призначення: {folder}")
            valid_destinations.append(folder)
        except Exception as e:
            print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Помилка з папкою
призначення {folder}: {e}")
    if not valid_destinations:
        print("Немає дійсних папок призначення. Вихід.")
        exit()
    event_handler = SyncEventHandler(SOURCE_FOLDER, valid_destinations,
FILE_EXTENSIONS_TO_SYNC)
    observer = Observer()
    observer.schedule(event_handler, SOURCE_FOLDER, recursive=True)
    print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Початок моніторингу...")
    print(f"Вихідна папка: {SOURCE_FOLDER}")
    print(f"Папки призначення: {valid_destinations}")
    print("Натисніть Ctrl+C щоб зупинити")
    try:
        observer.start()
        while True:
            event_handler._periodic_integrity_check()
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
        print("\Зупинка моніторингу...")
    finally:
        observer.join()
        print(f"[{time.strftime('%Y-%m-%d %H:%M:%S')}] Моніторинг припинено.")
```

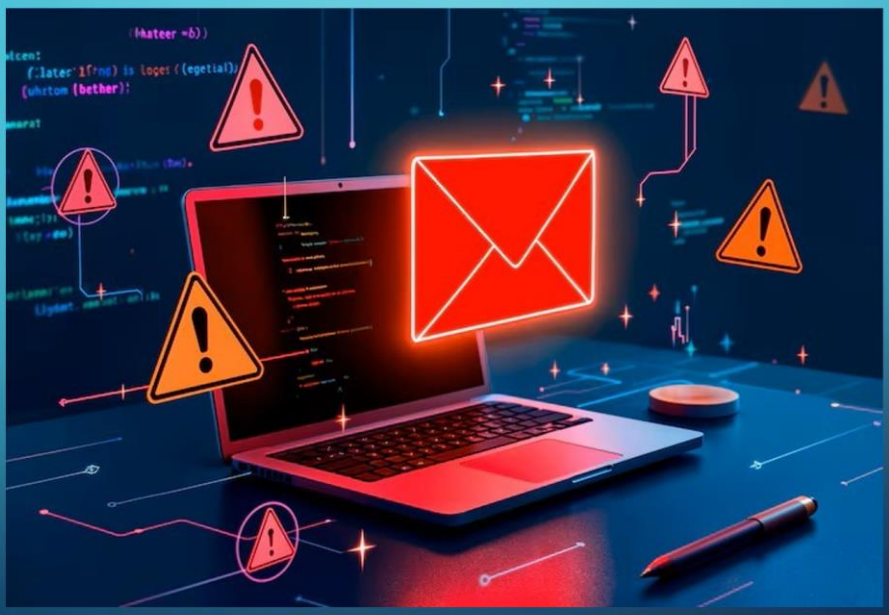
ДОДАТОК Б. Слайди мультимедійної презентації

РОЗРОБКА ПРОГРАМНОЇ МОДЕЛІ ВІДНОВЛЕННЯ ДАНИХ ПІСЛЯ АТАКИ НА ІНФОРМАЦІЙНУ СИСТЕМУ

ВИКОНАВ СТУДЕНТ ГРУПИ 4КБ-02

ТАТАРИНСЬКИЙ АНДРІЙ

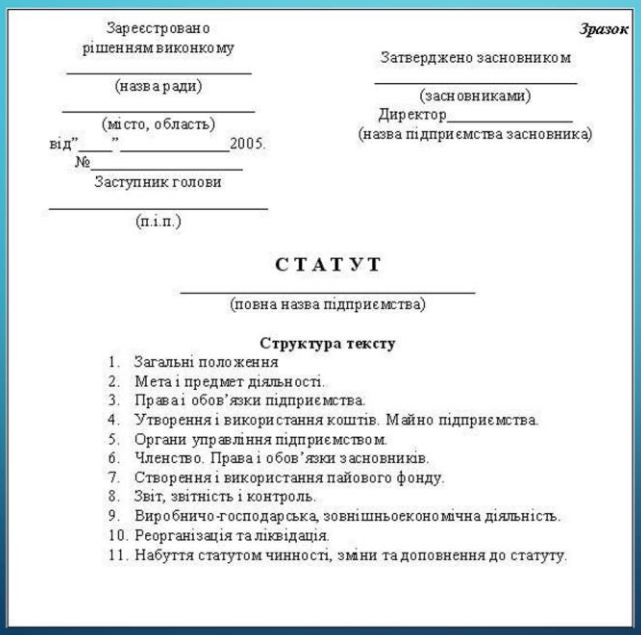
ВСТУП. АКТУАЛЬНІСТЬ ТЕМИ



ЗАГАЛЬНА МОДЕЛЬ ВЗАЄМОДІЇ ПРОГРАМНОЇ МОДЕЛІ З КОРИСТУВАЧЕМ ТА МЕРЕЖЕЮ

ЛОГІКА ДІЇ ПРОГРАМНОЇ МОДЕЛІ ВІДНОВЛЕННЯ

СТВОРЕННЯ ПІДГРУНТТЯ ДЛЯ РЕАЛІЗАЦІЇ РОБОТИ ПРОГРАМИ



Зареєстровано
рішенням виконкому
_____
(назва ради)
_____
(місто, область)
від"____"_____2005.
№_____
Заступник голови
_____
(п.і.п.)

Зразок

Затверджено засновником
_____
(засновниками)
Директор_____
(назва підприємства засновника)

СТАТУТ

_____
(повна назва підприємства)

Структура тексту

1. Загальні положення
2. Мета і предмет діяльності.
3. Права і обов'язки підприємства.
4. Утворення і використання коштів. Майно підприємства.
5. Органи управління підприємством.
6. Членство. Права і обов'язки засновників.
7. Створення і використання пайового фонду.
8. Звіт, звітність і контроль.
9. Виробничо-господарська, зовнішньоекономічна діяльність.
10. Реорганізація та ліквідація.
11. Набуття статутом чинності, зміни та доповнення до статуту.

Створення нового файлу для для прикладу роботи програми та подальші дії з ним

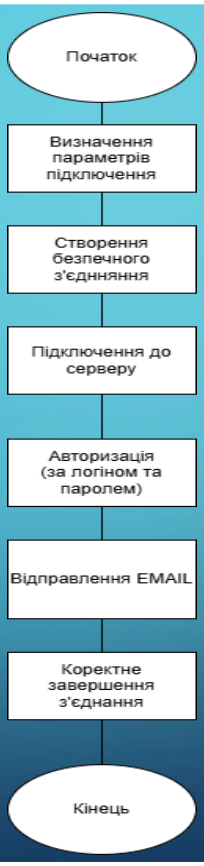


СХЕМА РОБОТИ БІБЛІОТЕКИ SMTP (СИНХРОНІЗАЦІЯ ДАНИХ НА EMAIL)

ЗАДІЯННЯ ПРОГРАМИ, ЩО ЗМІНЮЄ ХЕШІ ЦЬОГО ФАЙЛУ НА ВИПАДКОВІ (ШИФРУЄ ЙОГО)

шифрувальник.py - C:\Users\andre\Desktop\диплом\шифрувальник.py (3.13.3)

File Edit Format Run Options Window Help

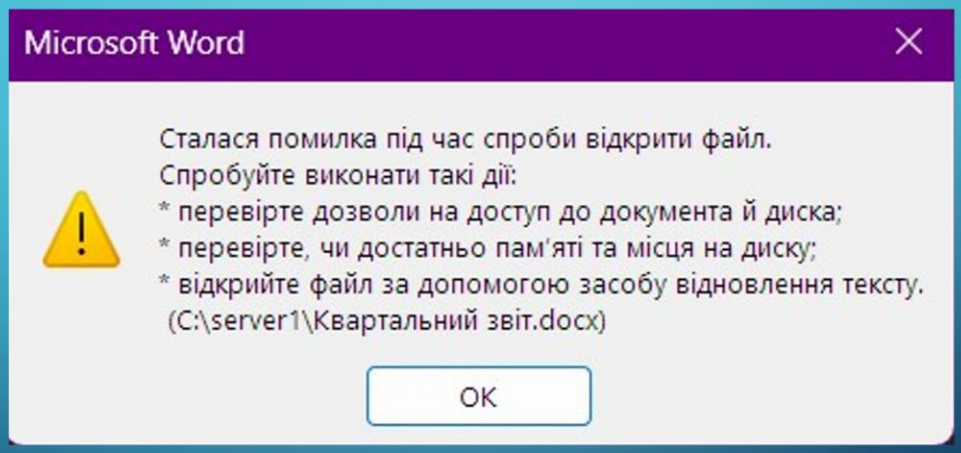
```python
import os
import hashlib
import random

class FileCorruptor:
    def __init__(self, source_folder):
        self.source_folder = source_folder

    def _calculate_hash(self, file_path):
        """Знаходить поточний хеш файлу"""
        hasher = hashlib.sha256()
        with open(file_path, 'rb') as f:
            while chunk := f.read(4096):
                hasher.update(chunk)
        return hasher.hexdigest()

    def corrupt_random_bytes(self, file_path, num_bytes=10):
        """
        Замінює випадкові біти в файлі
        """
        file_size = os.path.getsize(file_path)
        if file_size == 0:
            return False

        with open(file_path, 'r+b') as f:
            for _ in range(num_bytes):
                pos = random.randint(0, file_size - 1)
                f.seek(pos)
                f.write(bytes([random.randint(0, 255)]))
        return True
```

ЗАПУСК ШИФРУВАННЯ ФАЙЛУ

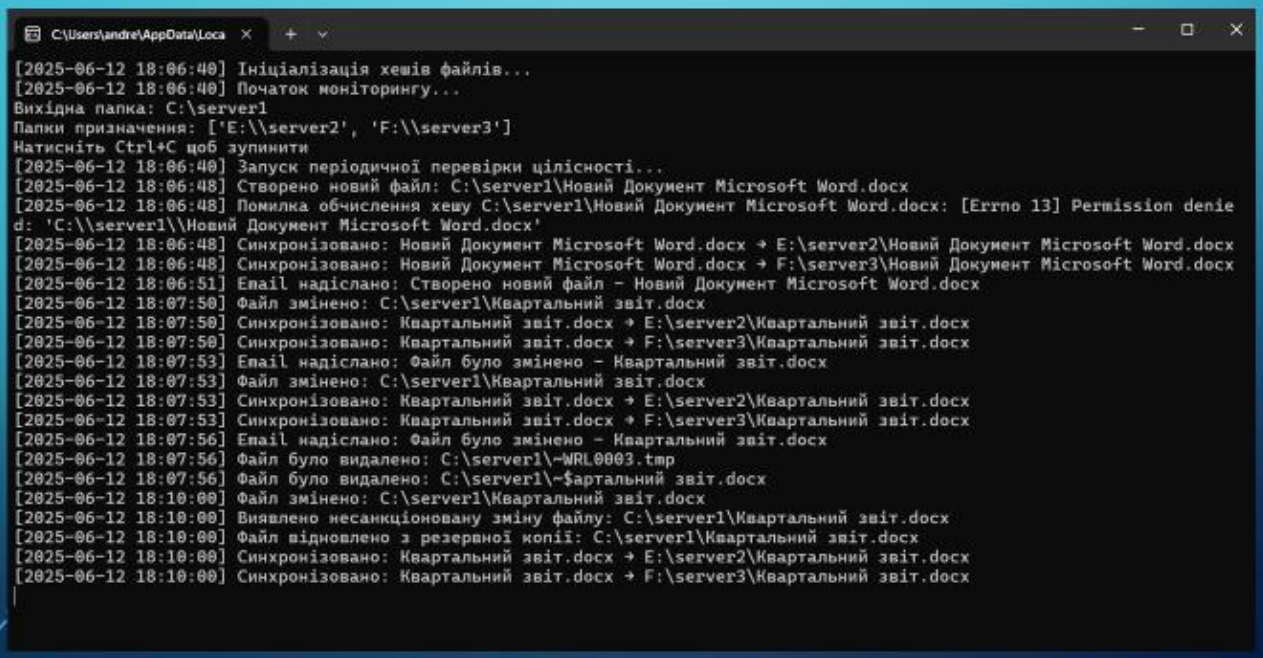
C:\Users\andre\AppData\Loca

```
Демонстраційний шифрувальник
Працює з папкою: C:\server1
Скільки файлів змінити1
Файл змінено: C:\server1\Квартальний звіт.docx
Вихідний хеш: 80944d68395d4e6bcaa8b3ece6e9ec84415f30342700bd0248c78867562872f4
Новий хеш:    b8b367dc32413e76158e3bfc2959fc5ddaf619b388446a2db6e435a8e45e5168
------------------------------------------------
Скільки файлів змінити
```

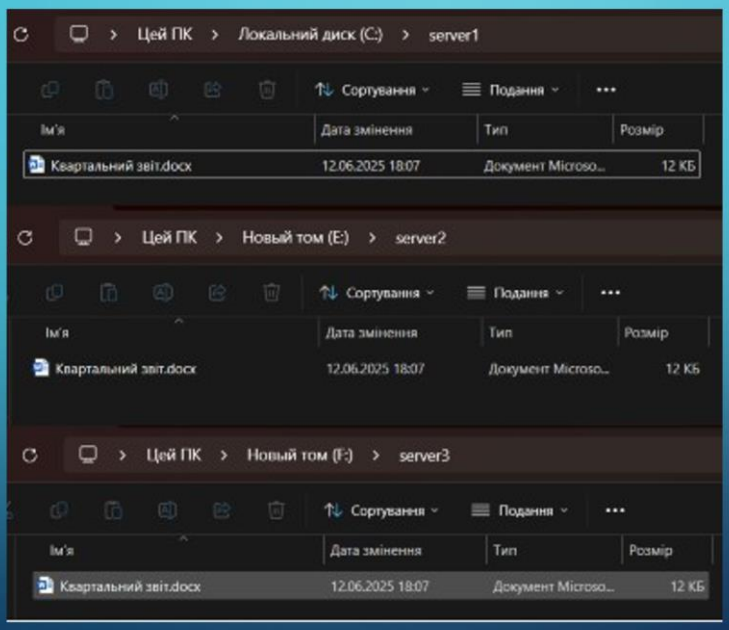
УНЕМОЖЛИВЛЕННЯ ВІДКРИТТЯ ДОКУМЕНТУ ПІСЛЯ ЗМІНЕННЯ ЙОГО ХЕШУ



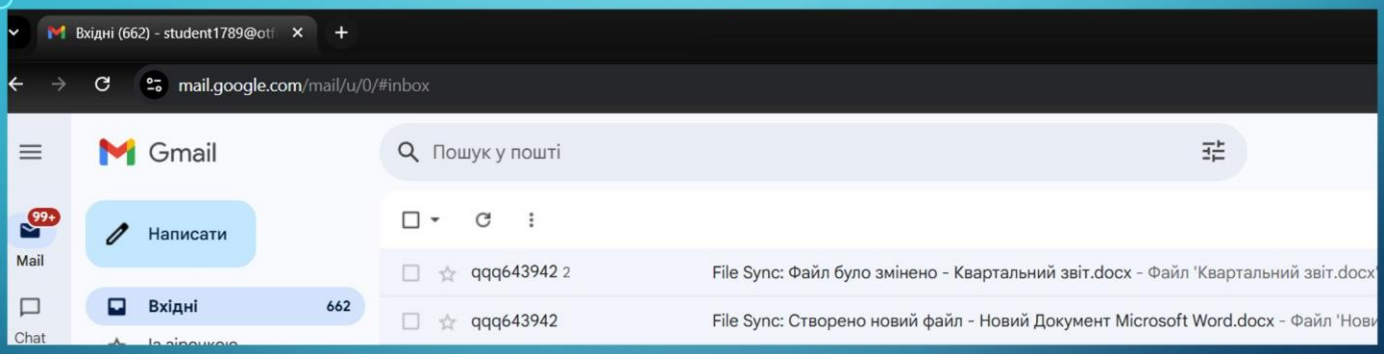
ЗАПУСК РОБОТИ ПРОГРАМНОЇ МОДЕЛІ ВІДНОВЛЕННЯ



СТВОРЕННЯ БЕКАПІВ ЦЬОГО ДОКУМЕНТУ. МОМЕНТАЛЬНЕ ВИДАЛЕННЯ УРАЖЕНОЇ КОПІЇ З ПОДАЛЬШИМ ВІДНОВЛЕННЯМ ІЗ РЕЗЕРВНИХ КОПІЙ



СИНХРОНІЗАЦІЯ ФАЙЛУ НА ЕЛЕКТРОНУ АДРЕСУ



МОЖЛИВОСТІ ТА ПЕРСПЕКТИВИ ПОДАЛЬШОГО РОЗВИТКУ ПРОГРАМНОЇ МОДЕЛІ ВІДНОВЛЕННЯ ДАНИХ ПІСЛЯ АТАКИ НА ІНФОРМАЦІЙНУ СИСТЕМУ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Татаринського Андрій Олексійовича

(прізвище, ім`я та по батькові)

Спеціальність *123 " Комп'ютерна інженерія "*

Освітня програма *«Безпека комп'ютерних систем і мереж»*

Керівник дипломного проекту (роботи) *Залапін Олексій Ігорович*

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка програмної моделі відновлення після атаки на інформаційну систему*

Обсяг розрахунково-пояснювальної записки *74* сторінок

Обсяг графічної (презентаційної) частини *13* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням створення моделі захисту за допомогою біометричної аутентифікації.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання проблематики та розробки моделі захисту SmartHome за допомогою біометричної аутентифікації, сформовано концепцію моделі згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів розробляємого програмного забезпечення. За допомогою відповідного програмного забезпечення реалізовані всі намічені роботи з процесом розробки.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи)

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint. Пояснювальна записка виконана задовільно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання – задовільна, академічного плагіату у роботі не виявлено.

г) перелік позитивних якостей дипломного проекту (роботи)

Детально описано процес виконання розробки програмної моделі;

Виконано проектування елементів програмного забезпечення з поясненнями на схемах та за допомогою коду;

Створено програмну модель відновлення, що працює в реальному часі й автоматично реагує на зміни

д) основні недоліки дипломного проекту (роботи)

Програмна модель працює лише в конкретному зазначеному алгоритмі; В якості інформаційної системи використовується вбудований функціонал ОС Windows; Відсутність графічного інтерфейсу; Обмежена масштабованість, модель орієнтована на окрему машину Недостатньо графічних ілюстрацій для демонстрації алгоритмів: деякі діаграми описані лише текстово.

Оцінка розрахункової частини *Добре*

Оцінка графічної частини *Задовільно*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

«21» 06 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Татаринського Андрія Олексійовича

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка програмної моделі відновлення даних після атаки на інформаційну систему*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 74 сторінки. У пояснювальній записці наведено етапи розробки програмного моделі для відновлення даних після атаки на інформаційну систему. Графічна частина складається з 13 слайдів мультимедійної презентації, які також містять схеми, діаграми та алгоритми, що передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проектом: *Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Татаринський А.О. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника, але з запізненням.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Татаринський А.О.під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою.*

Вважаю, що теоретична підготовка дипломника задовільна і він готовий до захисту дипломного проекту

г) вміння розв'язувати виробничі та конструкторські питання *Під час дипломного проектування здобувач освіти Воробйов С.С. мав змогу самостійно приймати окремі рішення з реалізації програмної частини для відновлення даних та показав вміння організовано працювати над поставленим завданням, розробляти структурні схеми та програмні зв'язки із застосуванням сучасних комп'ютерних програмних засобів, таких як Python IDE, а також готувати презентаційні та звітні матеріали.*

Оцінка розрахункової частини *Задовільно*

Оцінка графічної частини *Добре*

Загальна оцінка *Добре*

Прізвище, ім'я, по батькові керівника дипломного проекту *Залапін Олексій Ігорович*

Місце роботи і посада керівника дипломного проекту *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії.*

Підпис

«16» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Татаринський Андрій Олексійович
здобувач освіти гр. 4КБ-02, та

Залапін Олексій Ігорович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка програмної моделі відновлення даних після атаки на інфомаційну систему» (автор роботи – Татаринський А.О., керівник роботи – Залапін О.І.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Татаринський А.О. /

Керівник _____ / Залапін О.І. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Татаринського Андрія Олексійовича

на тему *Розробка програмної моделі відновлення даних після атаки на інформаційну систему*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з деякими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *23.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 22.06.2025 р. значення коефіцієнту подібності в роботі становить 14.68%, коефіцієнт цитування – 5.15%.*

(підпис) *23.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Татаринського А.О.* (П.І.Б.)

проведена «*23*» *червня* 2025 р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 6/22/2025
Дата редагування 6/22/2025

Документ прийнятий

Звіт подібності

метадані

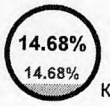
Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка програмної моделі відновлення даних після атаки на інформаційну систему

Автор
Татаринський Андрій Олексійович
Науковий керівник / Експерт
Залапін Олексій Ігорович

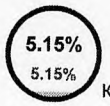
підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КЦ

25
Довжина фрази для коефіцієнта подібності 2

14080
Кількість слів

115887
Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	R	20
Інтервали	A→	0
Мікропробіли	¶	0
Білі знаки	R	0
Парафрази (SmartMarks)	a	102

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content	89 0.63 %
2	https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download	61 0.43 %
3	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	54 0.38 %
4	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	54 0.38 %
5	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	40 0.28 %

6	https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content	39 0.28 %
7	https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download	37 0.26 %
8	Програмне забезпечення для демонстрації вразливостей операційної системи Windows [redacted] 5/24/2023 National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")	33 0.23 %
9	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	33 0.23 %
10	https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download	32 0.23 %

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.90 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Програмне забезпечення для демонстрації вразливостей операційної системи Windows [redacted] 5/24/2023 **National University "Zaporizhzhia Polytechnic" (Кафедра "Програмні засоби")**	**96 (9) 0.68 %**
2	Система захищеного керування елементами комп'ютерної мережі [redacted] 3/16/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	18 (1) 0.13 %
3	«Вебзастосунок для роботи з інвестиційним портфелем» [redacted] 6/25/2024 The Kyiv Applied College of Tourism and Hospitality (The Kyiv Applied College of Tourism and Hospitality)	13 (1) 0.09 %

з Інтернету (13.78 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**338 (32) 2.40 %**
2	**https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download**	**311 (27) 2.21 %**
3	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/a141b658-5fa7-4f90-b0bd-7f0ccaed21e5/content**	**297 (13) 2.11 %**
4	https://github.com/shenxianmq/Auto_Symlink/commit/3960d680577fc4ee4af7899e2ef360c20fb14991.diff	198 (32) 1.41 %
5	https://blog.csdn.net/y131673/article/details/148447315	171 (19) 1.21 %
6	**https://card-file.ontu.edu.ua/bitstreams/035f6436-20b4-4ee6-8e99-bede670e308b/download**	**146 (7) 1.04 %**
7	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**78 (4) 0.55 %**
8	**https://card-file.ontu.edu.ua/bitstreams/6cf43324-8f08-4031-ba42-f80b18efbbc8/download**	**72 (2) 0.51 %**
9	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content**	**70 (2) 0.50 %**
10	**https://card-file.ontu.edu.ua/bitstreams/29489599-0581-4ce6-8890-c3b13d9f2e0e/download**	**57 (2) 0.40 %**
11	https://card-file.ontu.edu.ua/bitstreams/aed610a6-43ef-47e0-9066-e85c89456f3e/download	53 (5) 0.38 %

12	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/ead3fa83-2e3d-4cd7-bfbd-1d5ed04c1ce4/content**	**46 (2) 0.33 %**
13	https://qiita.com/mototoke/items/767f0b404bc389d997c8	18 (3) 0.13 %
14	https://card-file.ontu.edu.ua/bitstreams/bbed74c8-2ea7-44c5-8d00-0fe3fd9790ee/download	18 (1) 0.13 %
15	https://card-file.ontu.edu.ua/bitstreams/5240e379-7721-49f0-8ee8-27140b0b473a/download	13 (1) 0.09 %
16	https://www.sysdevlabs.com/uk/product.php?id=ufsxa5	13 (2) 0.09 %
17	https://ami.lnu.edu.ua/wp-content/uploads/2023/09/Zastosuvannia-kryptolohii-u-virtualniy-ekonomitsi.pdf	11 (1) 0.08 %
18	https://card-file.ontu.edu.ua/bitstreams/21173711-5b67-4b87-b17f-6302c25e7a31/download	9 (1) 0.06 %
19	https://card-file.ontu.edu.ua/bitstreams/34a6756b-592f-4b77-a805-183aa03a6a26/download	9 (1) 0.06 %
20	https://gist.github.com/Madhav-MKNC/c3d08b72456c1dee29854840cbdd1169	7 (1) 0.05 %
21	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	5 (1) 0.04 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних систем та мереж»
Група: 4 КБ- 02

Дипломний проєкт
здобувача освіти денної форми навчання КБ. 02.09.000.ДП

ТАТАРИНСЬКОГО
АНДРІЯ ОЛЕКСІЙОВИЧА

м. Одеса
2025 р. **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»**

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Безпека комп'ютерних **систем і мереж»** **Група:** 4 КБ-02

ПОЯСНЮВАЛЬНА **ЗАПИСКА до дипломного** проекту на **тему:**

Проектний матеріал складається з пояснювальної записки на _____ сторінках та графічного (презентаційного) матеріалу на _____ аркушах (слайдах) Дипломник (Татаринський А. О)
Керівник (Залапін О.І)

Консультанти:
з економічного розділу (Канський М. Ю.)
з розділу охорони праці та техніки безпеки (**Чорновол Н.І.**) з нормоконтролю (Петрашова **В.І.**) **старший** консультант (Кривченко Ю. В.) **До захисту допущений Голова циклової комісії** (**Кривченко Ю.В.**)
Завідувач відділення (Краснокутська К.Г.)

Захист «____»_____ 2025 р. **Протокол ЕК No**_____
Оцінка ЕК _____

Секретар ЕК _____