Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
« » червня 2021р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

дипломної роботи
бакалавра
(назва освітнього рівня)

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)

спеціальність 125 Кібербезпека
(код і назва спеціальності)

освітня програма Кібербезпека
(назва освітньої програми)

на тему: «Захист авторських прав в програмних продуктах»

Виконавець: студент IV курсу, групи КБ-42

_____ **Стецюк Євгеній Олегович**
(підпис) (прізвище ім'я по-батькові)

	Прізвище, ініціали	**Підпис**
Керівник	Даков С. Ю.	

Нормоконтроль	Зюбіна Р. В.	

Київ 2021

Міністерство освіти і науки України

«Київський національний університет імені Тараса Шевченка»

Факультет інформаційних технологій

Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
завідувач кафедри кібербезпеки
та захисту інформації
_____ Н.В. Лукова-Чуйко
«10» жовтня 2020 р.

ЗАВДАННЯ
на виконання дипломної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студенту **КБ-42** (група) **Стецюку Євгенію Олеговичу** (прізвище ім'я по-батькові)

Тема дипломної роботи Захист авторських прав в програмних продуктах

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема дипломної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №2 від 08.10.2020 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Захист авторського права, симетричні шифри

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Необхідно ознайомитися з циклом розробки програмного забезпечення, вразливостями, обрати метод шифрування даних та розробити програмне забезпечення для захисту авторських прав в програмних продуктах.

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність Розроблено програмне забезпечення, яке реалізує механізм захисту авторських прав в програмних продуктах

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання:12 жовтня 2020 року

Завдання видав		С. Ю. Даков
	(підпис)	(ініціали, прізвище)
Завдання прийняла до виконання		Є. О. Стецюк
	(підпис)	(ініціали, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	25.01.2021 – 28.01.2020	*виконано*
2	Аналіз літератури	29.01.2020 – 11.02.2020	*виконано*
3	Обґрунтування вибору рішення	12.02.2020 – 15.02.2020	*виконано*
4	Концепція побудови програмних продуктів	16.02.2020 – 04.03.2020	*виконано*
5	Аналіз проблем інформаційної безпеки в програмних продуктах	05.03.2020 – 22.03.2020	*виконано*
6	Дослідження вразливостей та загроз	23.03.2020 – 08.04.2020	*виконано*
7	Вироблення механізму захисту і програмного забезпечення авторських прав в програмних продуктах	09.04.2020 – 10.05.2020	*виконано*
8	Оформлення пояснювальної записки	11.05.2020 – 08.06.2021	*виконано*
9	Підготовка до захисту дипломної роботи	09.06.2021– 21.06.2021	*виконано*

Завдання видав		С. Ю. Даков
	(підпис)	(ініціали, прізвище)
Завдання прийняв до виконання		Є. О. Стецюк
	(підпис)	(ініціали, прізвище)

Термін подання дипломної роботи до ЕК 08 червня 2021 року

РЕФЕРАТ

Дипломна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел, додатків, має 60 сторінок основного тексту. Список використаних джерел містить 16 найменування і займає 2 сторінки.

Метою даної роботи є розробка програмного забезпечення для захисту авторських прав в програмних продуктах.

У роботі проаналізовані основні методи захисту авторських прав в програмних продуктах і на результатах аналізу розроблений механізм захисту.

Проаналізована існуюча література з теорії життєвого циклу розробки програмного продукту і розроблене програмне забезпечення на базі механізму захисту.

Розроблена інструкція використання механізму захисту авторських прав в програмних продуктах, яка описує основні кроки, які треба зробити, щоб використовувати цей механізм.

Ключові слова: Захист авторських прав, життєвий цикл розробки програмного продукту, безпека програмних продуктів, блочний шифр AES.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AES	–	Advanced Encryption Standard
HTTP	–	Hypertext Transfer Protocol
URI	–	Uniform Resource Identifier
DLL	–	Dynamic link library
XAML	–	Extensible Application Markup Language
SQL	–	Structured Query Language
WPF	–	Windows Presentation Foundation
SDLC	–	Systems development life cycle
CMMI	–	Capability Maturity Model Integration

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тим, що розробники програмних продуктів вкладають багато ресурсів та ідей в свої програмні продукти, то захист авторських прав стає значною проблемою.

Великі компанії витрачають свої час та гроші на розробку програмних продуктів. На даний момент, бюджет деяких продуктів перевищує 200 мільйонів доларів. В приклад можна поставити будь-яку комп'ютерну гру від великої компанії, наприклад "GTA V". Бюджет даної гри становить 256 млн. доларів. Без будь-якого захисту, цей продукт можуть розповсюджувати вільно в інтернеті, в результаті чого компанія втратить великі кошти. Тому попит до захисту таких програмних продуктів збільшується з вражаючою швидкістю. А разом з тим і вартість.

Аналіз останніх досліджень та літератури. Вчені, які зробили вклад у вивчення захисту авторського права в програмних продуктах: Деян Бака, Рунар Моен, Ольгерд П'єчул, Алі Фаті Алі Савелі, Хулусі Ондер, Скотт В. Грем, Стівен Е. Міллс, Ярі Раман, Понт Джонсон, Бенджамін Лівши, Джеффрі К. Ейверi, Кріс Моррелл, Метью Данлоп, Тімоті Бюннемейер, Девід Реймонд.

Тому *метою роботи* є розробка програмного забезпечення для захисту авторських прав в програмних продуктах.

Для досягнення мети треба виконати наступні завдання:

- аналіз моделей та методологій побудови та життєвого циклу розробки програмного забезпечення;
- проаналізувати методи та проблематику захисту програмного забезпечення;
- розробити механізм захисту авторських прав в програмних продуктах ;
- на базі механізму захисту розробити програмне забезпечення.

Об'єктом дослідження в даній роботі є процес захисту авторського права в програмних продуктах.

Предметом дослідження в даній роботі є механізми захисту авторського права в програмних продуктах.

Методи дослідження дипломної роботи:

- аналіз літератури;
- аналіз документів;
- позначення проблеми захисту;
- моделювання механізму захисту;
- тестування механізму захисту;

РОЗДІЛ 1 ОПИС ПОБУДОВИ ПРОГРАМНИХ ПРОДУКТІВ

1.1 Життєвий цикл і етапи розробки програмного забезпечення

Життєвий цикл розробки програмного забезпечення (SDLC) - це процес, який використовується індустрією програмного забезпечення для проектування, розробки та тестування високоякісних програмних продуктів. SDLC має на меті виготовити високоякісне програмне забезпечення, яке відповідає або перевершує очікування споживачів, досягає завершення в межах часу та кошторису витрат. Зазвичай його поділяють на шість-вісім кроків: планування, вимоги, дизайн, побудова, документ, тест, розгортання, обслуговування. Деякі менеджери проектів будуть комбінувати, розділяти або опускати кроки, залежно від обсягу проекту. На рисунку 1.1 представлені основні компоненти, рекомендовані для всіх проектів з розробки програмного забезпечення.

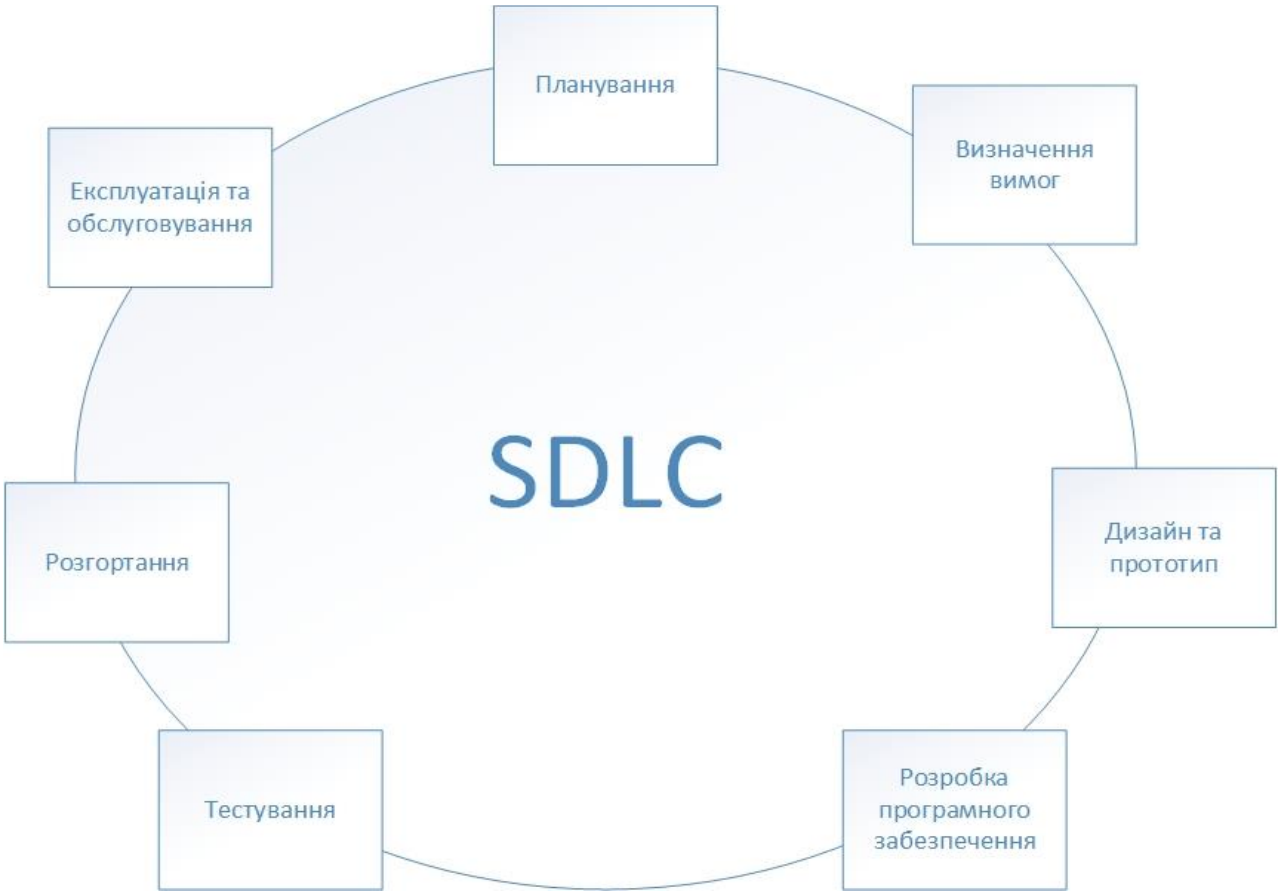

Рисунок 1.1 – Життєвий цикл розробки програмного забезпечення (SDLC)

ISO / IEC 12207 - це міжнародний стандарт для процесів життєвого циклу програмного забезпечення. Він прагне бути стандартом, який визначає всі завдання, необхідні для розробки та обслуговування програмного забезпечення [9].

SDLC - це також спосіб вимірювання та вдосконалення процесу розробки. Це дозволяє провести дрібнозернистий аналіз кожного етапу процесу, що у свою чергу, допомагає компаніям максимізувати ефективність на кожному етапі. Оскільки обчислювальна потужність зростає, це підвищує попит на програмне забезпечення та розробників. Компанії повинні скоротити витрати, швидше доставити програмне забезпечення та задовольнити або перевищити потреби своїх клієнтів. SDLC допомагає досягти цих цілей шляхом виявлення неефективності і занадто високих витрат та виправлення цих пунктів для оптимальної роботи.

Як працює життєвий цикл розробки програмного забезпечення?

Життєвий цикл розробки програмного забезпечення просто окреслює кожне завдання, необхідне для створення та розробки програмного додатку. Це допомагає зменшити втрату ресурсів на неважливі і не потрібні функції і компоненти та підвищити ефективність процесу розробки. Моніторинг розробки також гарантує, що проект залишатиметься на вірному шляху, і продовжуватиме бути реальною і виправданою інвестицією для компанії.

Багато компаній поділять кроки розробки на менші підрозділи для облегшення вигляду та розуміння цілого плану. Планування може бути розбито на дослідження технологій, маркетингові дослідження та аналіз витрат та вигод. Інші кроки можуть зливатися між собою. Фаза тестування може працювати одночасно з фазою розробки, оскільки розробники повинні виправляти помилки, що виникають під час тестування.

Сім основних фаз SDLC

Планування

На етапі планування керівники проекту оцінюють умови проекту. Це включає розрахунок трудових та матеріальних витрат, створення розкладу з цільовими завданнями та створення команд проекту та структури керівництва.

Планування може також включати відгуки зацікавлених сторін. Зацікавлені сторони - це всі, хто може отримати вигоду від програми. Можна спробувати отримати відгук від потенційних клієнтів, розробників, експертів з питань тематики та торгових представників.

Планування має чітко визначати обсяг і мету розробки програмного забезпечення. Він складає план курсу та положення команди для ефективного створення програмного забезпечення. Він також встановлює межі, які допомагають запобігти розширенню проекту або зміни курсу початкової мети.

Визначити вимоги

Визначення вимог вважається частиною планування для визначення того, що повинна робити програма, та її вимоги. Наприклад, програма в соціальних мережах потребує можливості зв'язку з другом. Інвентаризаційна програма може вимагати функції пошуку.

Вимоги також включають визначення ресурсів, необхідних для побудови проекту. Наприклад, команда може розробити програмне забезпечення для управління спеціальною виробничою машиною. Машина є вимогою в процесі.

Дизайн та прототип

Етап проектування моделює спосіб роботи програмного додатку. Деякі аспекти дизайну включають:

Архітектура - визначає мову програмування, галузеві практики, загальний дизайн та використання будь-яких компонентів чи готових шаблонів.

Інтерфейс користувача - визначає способи взаємодії клієнтів із програмним забезпеченням та реакцію програмного забезпечення на введення.

Платформи - визначає платформи, на яких буде працювати програмне забезпечення, такі як Apple, Android, версія Windows, Linux або навіть ігрові консолі.

Програмування - не просто мова програмування, а і методи вирішення проблем та виконання завдань у додатку.

Зв'язок - визначає методи, за допомогою яких програма може взаємодіяти з іншими активами, такими як центральний сервер або інші екземпляри програми

Безпека - визначає заходи, що вживаються для захисту програми, і може включати шифрування трафіку SSL, захист паролем та безпечне зберігання облікових даних користувача

Прототипування може бути частиною етапу проектування. Прототип схожий на одну з ранніх версій програмного забезпечення в ітеративній моделі розробки програмного забезпечення. Він демонструє основне уявлення про те, як виглядає та працює додаток. Цей "практичний" дизайн можна показати зацікавленим сторонам. Використати відгуки про вдосконалення програми. Змінити фазу прототипу на більш дешевше, ніж переписати код, щоб змінити фазу розробки.

Розробка програмного забезпечення

Це власне написання програми. Невеликий проект може бути написаний одним розробником, тоді як великий проект може бути розбитий на декілька компонентів і розроблятися декількома командами. На цьому етапі використовується програма контролю доступу або програма управління вихідним кодом. Ці системи допомагають розробникам відстежувати зміни коду в проекті. Вони також допомагають забезпечити сумісність між різними командними проектами та забезпечити досягнення цільових задач.

Процес програмування включає багато інших завдань. Багатьом розробникам потрібно попрацювати над навичками працювати в команді та підвищити рівень вже отриманих навичок. Такі завдання, як пошук та виправлення помилок та збоїв є критично важливими для розробки проекту. Деякі завдання часто затримують процес розробки, наприклад очікування результатів тестування або компіляція коду для запуску програми. SDLC може передбачити ці затримки, щоб розробники могли отримати інші завдання на час виконання попередніх.

Розробники програмного забезпечення цінують інструкції та пояснення до компонентів за які несуть відповідальність інші команди. Документація може бути формальним процесом, включаючи підключення посібника користувача для програми і також може бути неформальним процесом, як коментарі у вихідному коді, які пояснюють, чому розробник використовував певну процедуру. Навіть

компанії, які прагнуть створити легке та інтуїтивно зрозуміле програмне забезпечення, отримують вигоду від документації.

Документація може бути швидким оглядом основних функцій програми, які відображаються під час першого запуску. Це можуть бути відео-уроки для складних завдань. Письмова документація, така як посібники користувача, посібники з усунення несправностей та найчастіше задаваємі запитання, які допомагають користувачам вирішити проблеми або технічні питання.

Тестування

Дуже важливо протестувати програму, перш ніж зробити її доступною для користувачів. Значну частину тестування можна автоматизувати, як тестування безпеки. Інші випробування можна проводити лише в певному середовищі – в таких випадках розглядається можливість створення імітованого виробничого середовища для складних розгортань. Тестування повинно забезпечити правильну роботу кожної функції. Також слід протестувати різні частини програми для безперебійної роботи разом з перевіркою продуктивності, щоб зменшити будь-які зависання або затримки в обробці. Етап тестування допомагає зменшити кількість помилок та збоїв, з якими стикаються користувачі. Це призводить до кращого досвіду та задоволення користувачів при використанні програмного продукту.

Розгортання

На етапі розгортання додаток стає доступним для користувачів. Багато компаній хочуть, щоб фаза розгортання була повністю автоматизована. Таке розгортання може бути зроблено так же просто, як платіжний портал та посилання для завантаження на веб-сайті компанії. Розгортання програмного продукту також може бути зроблене у вигляді завантаження та встановлення програми на цільову платформу.

Розгортання також може бути складним і комплексним. Одним із прикладів є оновлення бази даних на рівні компанії до нещодавно розробленої програми. Оскільки в базі даних використовується кілька інших систем, інтеграція оновлення може зайняти більше часу та зусиль.

Експлуатація та обслуговування

На цьому цикл розробки майже закінчений. Додаток зроблено і використовується в полі. Однак фаза експлуатації та технічного обслуговування все ще важлива. На цьому етапі користувачі виявляють помилки, які не були знайдені під час тестування. Ці помилки потрібно вирішити, що може породити нові цикли розробки.

На додачу до виправлення помилок, ітеративна модель розробки містить в собі плани на додавання нових функцій і компонентів в майбутніх релізах. Для кожного нового релізу може початися новий цикл розробки.

1.2 Сучасні методології розроблення програмних систем

Пояснення моделей та методологій SDLC

Водоспад

Модель SDLC Водоспад - це класичний метод розробки. По мірі завершення кожної фази проект переходить на наступний крок. Це перевірена модель, і вона працює. Однією з переваг моделі водоспаду є те, що кожна фаза може бути оцінена на предмет безперервності та доцільності перед тим, як рухатись далі. Однак швидкість обмежена, оскільки одна фаза повинна закінчуватися до початку іншої.

AGILE

Модель AGILE була розроблена розробниками, щоб поставити потреби споживачів на перше місце. Цей метод сильно фокусується на досвіді користувача та введенні даних. Це вирішує більшу частину проблем старих додатків, які були незвичайними та громіздкими у використанні. Крім того, це робить програмне забезпечення чутливим до відгуків клієнтів. Agile прагне швидко випустити програмні цикли, реагуючи на мінливі ринки. Для цього потрібна сильна команда з відмінною комунікацією. Це також може призвести до того, що проект зверне з початкового курсу розробки, занадто сильно покладаючись на відгуки клієнтів.

Ітеративний

У моделі ітеративної розробки програмісти швидко створюють початкову базову версію програмного забезпечення. Потім вони розглядають і вдосконалюють

заявку невеликими кроками (або ітераціями). Цей підхід найчастіше використовується у дуже великих додатках. Він може швидко створити програму та її функціонал для задоволення бізнес-потреб. Однак цей процес може швидко перевищити допустимий обсяг використаних ресурсів [5].

DevOps

Модель безпеки DevOps включає операції людей, які використовують програмне забезпечення у цикл розробки. Як і Agile, це прагне покращити зручність використання та актуальність програм. Однією із суттєвих переваг цієї моделі є відгуки фактичних користувачів програмного забезпечення про етапи проектування та впровадження. Одним недоліком є те, що він вимагає активної співпраці та спілкування. Ці додаткові витрати можна компенсувати автоматизацією частин процесу розробки.

Інші моделі

Багато інших моделей SDLC є по суті варіантом цих основних процесів. Організації використовують виробничі процеси LEAN для розробки програмного забезпечення. Lean або ощадливе виробництво – це концепція менеджменту, що передбачає залучення у процес оптимізації бізнесу кожного співробітника і максимальну орієнтацію на споживача. V-подібна розробка - це тип водоспаду, який реалізує випробування, перевірку та підтвердження. Спіральний розвиток може вибирати моделі для кожного кроку в процесі розвитку.

Кращі практики розробки програмного забезпечення

На додаток до моделей та етапів розробки програмного забезпечення існує ще кілька корисних практик. Вони можуть застосовуватися до частини або до всього циклу розвитку.

Джерело контролю

Джерело контролю - це план безпеки для захисту вашого робочого коду. Реалізує контроль над джерелом, зберігаючи код в одному місці, з безпечним та зареєстрованим доступом. Це може бути фізичне місце, де файли зберігаються та переглядаються в одній кімнаті в будівлі. Це також може бути віртуальний простір,

де користувачі можуть входити за допомогою зашифрованого підключення до хмарного середовища розробки.

Додатки «Джерело контролю» включають систему управління змінами для відстеження роботи, виконаної окремими особами чи командами. Як і для будь-якого сховища, рекомендується використовувати резервну систему для запису прогресу розробки у випадку непередбачених подій [3].

Постійна інтеграція

Безперервна інтеграція виникла з того, чого не можна робити. Постійна інтеграція працює, щоб переконатися, що кожен компонент сумісний протягом усього циклу розробки. До неї різні команди будували власні проекти самостійно. Це створювало значні проблеми в кінці, коли розробники розгортали додаток. Постійна інтеграція гарантує, що всі команди використовують подібні мови програмування та бібліотеки, а також допомагає запобігати конфліктам та дублюванню роботи.

Системи управління SDLC

Система управління циклом розробки програмного забезпечення працює для контролю та управління кожним кроком циклу розробки. Системи управління додають прозорості кожній фазі та проекту в цілому. Вони також додають аналітику, відстеження помилок та системи управління роботою. Ці показники можна використовувати для покращення тих частин циклу, які не працюють ефективно.

Висновки за розділом 1

У даному розділі наведено процес розробки програмного забезпечення SDLC, який показує, що відбувається, і де саме процес розробки може покращитися. Як і багато бізнес-процесів, SDLC має на меті аналіз та вдосконалення процесу створення програмного забезпечення. Це створює масштабований вигляд проекту, від повсякденного кодування до управління датами виробництва.

Також описано ключові етапи та методи розробки програмного забезпечення, з котрих можна виділити сім основних етапів та три основні методи.

РОЗДІЛ 2 ОСНОВНІ ЗАГРОЗИ І МЕТОДИ ЇХ УСУНЕННЯ

2.1 Загрози для програмних продуктів

В інформаційній безпеці загрозою - джерелом небезпеки - часто є людина, яка має намір заподіяти шкоду, використовуючи одного або декількох зловмисних програмних агентів. Програмне забезпечення підпадає під дію двох загальних категорій загроз:

Загрози під час розробки (переважно інсайдерські загрози). Інженер-програміст може саботувати програмне забезпечення в будь-який момент його життєвого циклу шляхом навмисного виключення, включення або модифікації специфікації вимог, моделей загроз, проектної документації, вихідного коду, збірки та інтеграції, тесту випадки та результати випробувань, або інструкції та інструменти з установки та конфігурації. Практики безпечної розробки, описані в цій книзі, частково розроблені, щоб допомогти зменшити вплив програмного забезпечення на інсайдерські загрози в процесі його розробки.

Загрози під час роботи (як внутрішні, так і зовнішні загрози). Будь-яка програмна система, яка працює на підключеній до мережі платформі, швидше за все, матиме вразливі місця під час зловмисників. Атаки можуть скористатися загальновідомими, але не виправленими вразливими місцями, що призведе до пошкодження пам’яті, виконання довільних скриптів експлойта, віддаленого виконання коду та переповнення буфера. Недоліки програмного забезпечення можуть бути використані для встановлення шпигунського, рекламного та іншого шкідливого програмного забезпечення в системах користувачів, які можуть знаходитись у стані неактивності до тих пір, поки їх не буде запущено.

Слабкі місця, на які найімовірніше буде націлено, - це ті, що виявляються у зовнішніх інтерфейсах програмних компонентів, оскільки ці інтерфейси забезпечують зловмиснику прямий шлях зв’язку до вразливостей програмного забезпечення. Ряд добре відомих атак націлений на програмне забезпечення, яке

включає інтерфейси, протоколи, конструктивні особливості або помилки в розробці, які добре розуміються і широко розголошуються як приховування властивих недоліків. Це програмне забезпечення включає веб-програми (включаючи компоненти браузера та сервера), веб-служби, системи управління базами даних та операційні системи. Випадки зловживання (або зловживання) можуть допомогти менеджерам проектів та інженерам програмного забезпечення бачити своє програмне забезпечення з точки зору зловмисника, передбачаючи та визначаючи несподівану або ненормальну поведінку, за допомогою якої функція програмного забезпечення може бути ненавмисно використана або бути навмисно зловживаною.

Сьогодні більшість менеджерів проектів та IT, відповідальних за роботу системи, реагують на все більшу кількість Інтернет-атак, покладаючись на оперативний контроль на рівні операційної системи, мережі та бази даних або веб-сервера, не вдаючись безпосередньо вирішити проблему небезпеки програми програмне забезпечення рівня, яке скомпрометоване. Цей підхід має два критичні недоліки:

Безпека програми повністю залежить від надійності операційних захистів, які її оточують.

Багато програмних механізмів захисту (елементів керування) можуть бути легко налаштовані або застосовані неправильно. Крім того, вони з такою ж ймовірністю містять вразливі місця, якими можна скористатися, як і прикладне програмне забезпечення, яке вони (нібито) захищають.

Широкий розголос про буквально тисячі успішних атак на програмне забезпечення, доступне з Інтернету, просто полегшило роботу зловмиснику. Зловмисники можуть вивчати численні повідомлення про вразливі місця безпеки в широкому діапазоні комерційних програм та програм з відкритим кодом та отримувати доступ до загальнодоступних сценаріїв експлуатації. Більш досвідчені зловмисники часто розробляють (і діляться) складні, цілеспрямовані атаки, які використовують конкретні вразливості. Крім того, природа ризиків змінюється швидше, ніж програмне забезпечення може бути адаптоване для протидії цим ризикам, незалежно від процесу розробки програмного забезпечення та практики.

Щоб бути на 100 відсотків ефективними, захисники повинні передбачити всі можливі вразливості, тоді як зловмисникам потрібно знайти лише одну, щоб здійснити свою атаку.

Джерела незахищеності програмного забезпечення

Більшість комерційних програм та програм з відкритим кодом, систем проміжного програмного забезпечення та операційних систем надзвичайно великі та складні. У звичайному виконанні ці системи можуть переходити через величезну кількість різних станів. Ці характеристики особливо ускладнюють розробку та експлуатацію програмного забезпечення, яке є стабільно правильним, не кажучи вже про стабільно безпечне. Неминуча наявність загроз і ризиків для безпеки означає, що керівники проектів та інженери-програмісти повинні звертати увагу на безпеку програмного забезпечення, навіть якщо чіткі вимоги до нього не були враховані в специфікації програмного забезпечення.

Великого відсотка слабких місць у програмному забезпеченні можна було б уникнути, якщо б менеджери проектів та інженери-програмісти регулярно проходили навчання щодо того, як систематично та послідовно вирішувати ці слабкі сторони. На жаль, цього персоналу рідко вчать, як розробляти та розробляти безпечні програми та проводити перевірку якості для перевірки на небезпечні помилки кодування та використання поганих методів розробки. Вони, як правило, не розуміють, які практики ефективні при виявленні та усуненні несправностей та дефектів або при обробці вразливостей, коли зловмисники експлуатують програмне забезпечення. Вони часто не знайомі з наслідками для безпеки певних вимог до програмного забезпечення (або їх відсутністю). Подібним чином вони рідко дізнаються про наслідки для безпеки того, як програмне забезпечення розробляється, розробляється, розробляється, розгортається та працює. Відсутність цих знань означає, що вимоги до безпеки, швидше за все, будуть неадекватними, а отримане програмне забезпечення, ймовірно, буде відхилятися від визначених (і неуточнених) вимог безпеки. Крім того, ця відсутність знань заважає керівнику та інженеру розпізнати та зрозуміти, які помилки можуть виявитись як слабкі місця та вразливості програмного забезпечення, коли воно починає працювати [12].

Програмне забезпечення - особливо мережеве програмне забезпечення на рівні додатків - найчастіше скомпрометоване, використовуючи слабкі місця, які виникають із таких джерел:

Складності, неадекватність та / або зміни в моделі обробки програмного забезпечення (наприклад, веб-або сервісно-орієнтована модель архітектури).

Неправильні припущення інженера, включаючи припущення про можливості, результати та стани поведінки середовища виконання програмного забезпечення або про очікувані вхідні дані від зовнішніх сутностей (користувачів, програмні процеси).

Неправильна специфікація чи дизайн, або дефектне виконання

- Інтерфейси програмного забезпечення із зовнішніми об'єктами. Помилки розробки цього типу включають неадекватну (або неіснуючу) перевірку вводу, обробку помилок та обробку винятків.

- Компоненти середовища виконання програмного забезпечення (від рівня проміжного програмного забезпечення та рівня операційної системи до компонентів мікро-програмного та апаратного забезпечення).

Ненавмисна взаємодія між програмними компонентами, включаючи ті, що надаються третьою стороною.

Помилок не уникнути. Навіть якщо їх уникнути під час проектування та проектування вимог (наприклад, за допомогою використання формальних методів) та розробки (наприклад, шляхом всебічного огляду коду та великого тестування), уразливості все одно можуть бути внесені в програмне забезпечення під час його складання, інтеграції, розгортання та операції. Незалежно від того, наскільки точно дотримується життєвий цикл із підвищеною безпекою, доки програмне забезпечення продовжує збільшуватися в розмірах і складності, певна кількість помилок, що піддаються експлуатації, та інші слабкі місця, безумовно, існують.

2.2 Методи захисту програмних продуктів

Менеджери та інженери програмного забезпечення повинні розглядати всі несправності та недоліки програмного забезпечення як потенційно придатні для експлуатації. Зменшення слабких місць, що можна використати, починається зі специфікації вимог до захисту програмного забезпечення, а також з урахуванням вимог, які могли бути пропущені. Програмне забезпечення, яке включає вимоги до безпеки (наприклад, обмеження безпеки на поведінку процесів та обробку вхідних даних, а також стійкість до навмисних збоїв та терпимість до них), швидше за все, буде спроектовано, щоб залишатися надійним та захищеним в умовах атаки. Крім того, використання випадків зловживання / зловживання, що передбачають ненормальну та несподівану поведінку, може допомогти отримати краще розуміння того, як створити безпечне та надійне програмне забезпечення.

Розробка програмного забезпечення з самого початку з урахуванням безпеки є на порядок ефективнішою, ніж спроба перевірити, шляхом тестування та перевірки, надійність програмного забезпечення. Наприклад, спроба продемонструвати, що реалізована система ніколи не прийме небезпечного введення (тобто доведення негативного), неможлива. Однак ви можете довести, використовуючи такі підходи, як офіційні методи та абстрагування функцій, що програмне забезпечення, яке ви розробляєте, ніколи не прийме небезпечного введення. Крім того, простіше розробити та впровадити систему таким чином, щоб підпрограми перевірки вхідних даних перевіряли кожен вхід, який отримує програмне забезпечення, за набором заздалегідь визначених обмежень. Тестування функції перевірки вхідних даних, щоб продемонструвати, що вона послідовно викликається та правильно виконується кожного разу, коли вхід надходить у систему, потім включається в функціональне тестування системи.

Аналіз та моделювання можуть слугувати кращому захисту вашого програмного забезпечення від більш тонких, складних моделей атак, що включають зовнішньо вимушені послідовності взаємодій між компонентами або процесами, які ніколи не мали на меті взаємодіяти під час звичайного виконання програмного

забезпечення. Аналіз та моделювання можуть допомогти вам визначити, як посилити безпеку інтерфейсів програмного забезпечення із зовнішніми об'єктами та підвищити стійкість до всіх несправностей. Методи на підтримку аналізу та моделювання протягом кожної фази життєвого циклу, такі як моделі нападів, випадки зловживання та зловживання та аналіз архітектурного ризику, описані в наступних розділах цієї книги [8].

Якщо обмеження часу та ресурсів вашої організації-розробника перешкоджають застосуванню безпечної практики розробки до всієї системи програмного забезпечення, ви можете використовувати результати оцінки ризиків, керованої бізнесом, щоб визначити, яким компонентам програмного забезпечення слід надавати найбільший пріоритет.

Процес життєвого циклу, що підсилюється безпекою, повинен (принаймні певною мірою) компенсувати недоліки безпеки у вимогах програмного забезпечення шляхом додавання практик, керованих ризиками, та перевірки адекватності цих практик на всіх фазах життєвого циклу програмного забезпечення. На рисунку 1.2 зображено один приклад того, як включити безпеку в SDLC, використовуючи концепцію точок дотику. Найкращі методи програмного забезпечення (точки дотику, показані стрілками) застосовуються до набору програмних артефактів (квадратів), які створюються в процесі розробки програмного забезпечення. Метою цього конкретного підходу є те, що він є нейтральним до процесу, і, отже, може використовуватися з широким спектром процесів розробки програмного забезпечення (наприклад, водоспад, спритний, спіральний, інтеграція моделі зрілості можливостей [CMMI]).

Контроль безпеки протягом життєвого циклу програмного забезпечення не повинен обмежуватися вимогами, дизайном, кодом та етапами тестування. Важливо продовжувати проводити перевірку коду, тести безпеки, суворий контроль конфігурації та забезпечення якості під час розгортання та операцій, щоб гарантувати, що оновлення та виправлення не додають слабких місць безпеки або шкідливої логіки до виробничого програмного забезпечення.10. Додаткові міркування для керівників проектів, включаючи вплив вимог безпеки програмного

забезпечення на обсяг проекту, плани проектів, оцінку ресурсів, а також заходи щодо продукту та процесу.

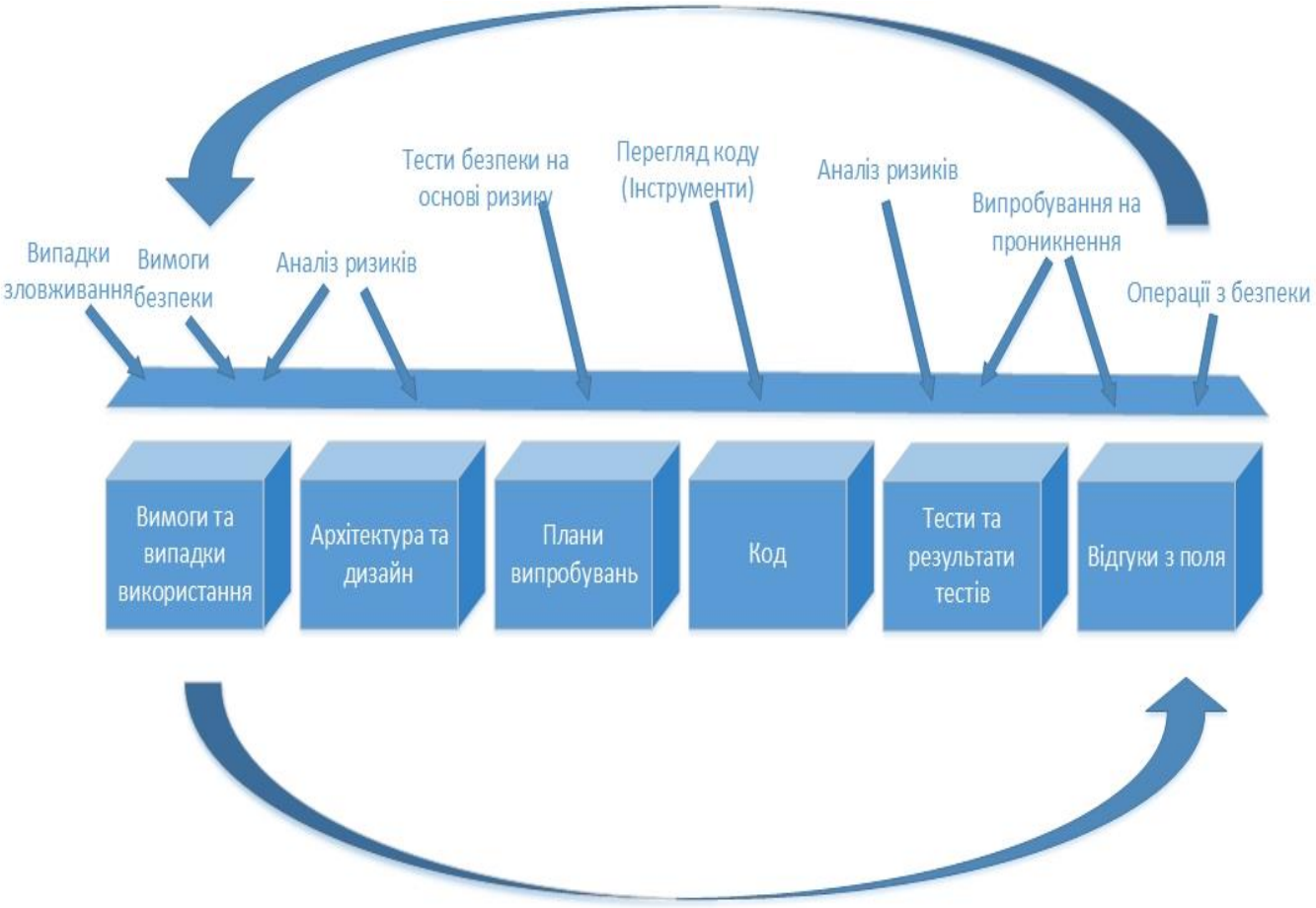

Рисунок 1.2 - Життєвий цикл розробки програмного забезпечення з визначеними точками безпеки

Ніколи не є гарною стратегією безпеки купувати найновіший інструмент безпеки та бути впевненим, що більше вашому проекту нічого не загрожує. Вам потрібно інвестувати в кілька інструментів, а також цілеспрямоване навчати розробників, налаштовувати та інтегрувати інструменти, перш ніж ви побачите віддачу від своїх інвестицій у безпеку.

Тому, перш ніж ви отримаєте інструмент, який вирішує лише невелику частину ваших ризиків безпеки, треба переконатися, що у вас є надійна стратегія безпеки програмного забезпечення, яка включає ці 10 практик щодо захисту програмного забезпечення.

1. Програмне забезпечення та система

Багато зловмисників використовують відомі вразливості, пов’язані зі старим або застарілим програмним забезпеченням. Щоб запобігти поширеним атакам, переконайтеся, що у всіх ваших системах є оновлені рішення. Регулярне оновлення рішень - одна з найефективніших практик захисту програмного забезпечення.

Звичайно, ви не можете постійно оновлювати програмне забезпечення, якщо не знаєте, чим користуєтесь. Сьогодні в середньому 70% - і часто більше 90% - програмних компонентів програм є відкритими. Вам потрібно вести інвентаризацію або специфікацію програмного забезпечення цих компонентів. Спеціальна специфікація допомагає вам переконатися, що ви виконуєте ліцензійні зобов’язання цих компонентів і залишаєтесь на вершині оновлень.

Складно створити специфікацію програмного забезпечення вручну, але інструмент аналізу складу програмного забезпечення автоматизує завдання та висвітлить ризики безпеки та ліцензування.

2. Навчання користувачів

Навчання працівників має бути частиною ДНК безпеки вашої організації. Маючи добре організовану та доглянуту навчальну програму з питань безпеки для ваших співробітників, ви зможете значно підвищити захист ваших даних та активів. Включіть навчання з обізнаності для всіх співробітників та безпечне навчання з кодування для розробників. Робіть це регулярно, а не раз на рік. І проводити симуляції, такі як фішинг-тести, щоб допомогти співробітникам виявляти та припиняти атаки соціальної інженерії.

3. Автоматизуйте завдання

Зловмисники використовують автоматизацію для виявлення відкритих портів, помилкових налаштувань безпеки тощо. Тож ви не можете захищати свої системи, використовуючи лише ручні методи. Натомість автоматизуйте щоденні завдання безпеки, такі як аналіз змін брандмауера та конфігурацій безпеки пристрою. Автоматизація таких завдань дозволяє вашим співробітникам служби безпеки зосередитись на більш стратегічних ініціативах щодо безпеки.

Ви також можете автоматизувати більшу частину тестування програмного забезпечення, якщо у вас є відповідні інструменти. Сюди входить, підтримка

специфікації програмного забезпечення, яка допоможе вам оновити компоненти програмного забезпечення з відкритим кодом та відповідати їх ліцензіям. За допомогою інструмента цих інструментів ви можете автоматизувати завдання, яке просто неможливо виконати вручну.

4. Застосовувати мінімальні права доступу

Переконайтеся, що користувачі та системи мають мінімальні права доступу, необхідні для виконання своїх робочих функцій. Застосування принципу найменших привілеїв значно зменшує поверхню атаки, усуваючи непотрібні права доступу, що може спричинити за собою різні компроміси.

Застосування принципу найменших привілеїв значно зменшує поверхню атаки.

Сюди входить уникнення "повзучих привілеїв", що трапляється, коли адміністратори не відкликають доступ до систем або ресурсів, які співробітник більше не потребує. Повзання привілеїв може статися, коли працівник переходить на нову роль, приймає нові процеси, виходить з організації або, насамперед, повинен отримати доступ лише до тимчасового або нижчого рівня.

5. Створіть надійний план реагування на інциденти

Скільки б ви не дотримувались найкращих практик безпеки програмного забезпечення, ви завжди зіткнетеся з можливістю порушення. Але якщо ви підготуєтесь, ви можете перешкодити зловмисникам досягти своєї місії, навіть якщо вони порушують вашу систему. Складіть надійний план реагування на аварії, щоб виявити атаку, а потім обмежити шкоду від неї.

Скільки б ви не дотримувались найкращих практик безпеки програмного забезпечення, ви завжди зіткнетеся з можливістю порушення.

6. Документуйте свою політику безпеки

Вести сховище знань, яке включає всебічно задокументовані політики безпеки програмного забезпечення. Політика безпеки дозволяє вашим співробітникам, включаючи адміністраторів мережі, персонал служби безпеки тощо, розуміти, які дії ви виконуєте і чому.

Крім того, недостатньо лише мати правила. Переконайтесь, що їх усі читають. Як мінімум, зробіть цю частину процесу вступу нових співробітників.

Політика безпеки дозволяє вашим співробітникам зрозуміти, яку діяльність ви виконуєте і чому.

7. Сегментуйте свою мережу

Сегментування вашої мережі - це застосування принципу найменших привілеїв. Правильна сегментація мережі обмежує рух зловмисників. Визначте, де зберігаються ваші критичні дані, та використовуйте відповідні засоби контролю, щоб обмежити трафік до цих сегментів мережі та з них.

8. Інтегруйте безпеку у свій SDLC

Інтегруйте діяльність із захисту програмного забезпечення у життєвий цикл розробки програмного забезпечення (SDLC) вашої організації від початку до кінця. Ці заходи повинні включати аналіз архітектурного ризику, статичне, динамічне та інтерактивне тестування безпеки додатків, тестування SCA та тестування. Вбудовування безпеки у ваш SDLC спочатку вимагає часу та зусиль. Але виправлення вразливостей на початку SDLC значно дешевше і набагато швидше, ніж очікування до кінця. Зрештою, це зменшує ваш ризик безпеки.

Інтегруйте діяльність із забезпечення програмного забезпечення у життєвий цикл розробки програмного забезпечення (SDLC) вашої організації від початку до кінця.

9. Моніторинг активності користувачів

Довіряйте, але перевіряйте. Моніторинг діяльності користувачів допомагає гарантувати, що користувачі дотримуються найкращих практик програмного забезпечення щодо безпеки. Це також дозволяє виявляти підозрілі дії, такі як зловживання привілеями та видавання себе за іншу особу.

10. Визначення ключових показників

Визначте ключові показники, які мають значення та стосуються вашої організації. Чітко визначені показники допоможуть вам з часом оцінити свою безпеку.

Фактом життя є те, що несправності, дефекти та інші слабкі сторони програмного забезпечення впливають на здатність програмного забезпечення безпечно функціонувати. Ці вразливі місця можна використовувати для порушення властивостей безпеки програмного забезпечення та змусити програмне забезпечення перейти в незахищений та придатний для використання стан. Боротьба з цією можливістю є особливо страшною проблемою, враховуючи всюдисущі зв'язки, вибуховий ріст і складність програмних систем[1].

Прийняття процесу розробки програмного забезпечення з підвищеною безпекою, який включає практику безпечної розробки, зменшить кількість несправностей та слабких місць розгортаного програмного забезпечення. Виправлення потенційних уразливостей якомога раніше у SDLC, головним чином шляхом прийняття процесів та практик, що покращують безпеку, є набагато економічнішим, ніж спроба діагностувати та виправити такі проблеми після запуску системи. Це просто має сенс.

Таким чином, цілі використання практик безпечного програмного забезпечення такі:

Несправності та інші слабкі місця, що використовуються, усуваються, наскільки можливо, добросовісними інженерами.

Швидше за все зменшується або усувається можливість того, що зловмисні інженери можуть навмисно імплантувати в програмне забезпечення несправності та слабкі місця, шкідливу логіку або махінації.

Програмне забезпечення є стійким до атак, стійким до атак та стійким до атак, наскільки це можливо та практично на підтримку виконання місії організації.

Щоб переконатися, що програмне забезпечення та системи відповідають вимогам безпеки протягом життєвого циклу розробки, перегляньте, виберіть та адаптуйте вказівки з цієї книги, веб-сайту BSI та джерел, цитованих у цій книзі, як частину звичайної діяльності з управління проектами.

2.3 Нормативно-правова база

Одним з об’єктів авторського права є комп’ютерна програма. У ст. 1 Закону України «Про авторське право і суміжні права» наводиться визначення комп’ютерної програми, відповідно до якого комп’ютерна програма – це набір інструкцій у вигляді слів, цифр, кодів, схем, символів чи у будь-якому іншому вигляді, виражених у формі, придатній для зчитування комп’ютером, які приводять його у дію для досягнення певної мети або результату (це поняття охоплює як операційну систему, так і прикладну програму, виражені у вихідному або об’єктному кодах).

Визначення є досить складним, оскільки містить багато технічних характеристик, які є не повною мірою зрозумілими для пересічної особи, в тому числі для суду, який розглядає спір. Варто додати, що комп’ютерна програма має свою візуалізацію, зокрема, це інтерфейс веб-сайту, мобільних додатків тощо. Комп’ютерні програми прирівнюються до літературних творів і мають таку ж правову охорону. Така охорона поширюється на комп’ютерні програми незалежно від способу чи форми їх вираження. Також про це йдеться і у міжнародно-правових актах. Так, у ст. 4 Договору ВОІВ про авторське право вказано, що комп’ютерні програми охороняються як літературні твори у розумінні ст. 2 Бернської конвенції незалежно від способу або форми їх вираження [4].

2.4 Проблематика захисту авторських прав в програмних продуктів

Що таке авторське право на програмне забезпечення?

Авторське право не захищає факти, ідеї, системи чи методи роботи, а захищає спосіб вираження цих речей. Ви можете окреслити свої ідеї в письмовій формі або на малюнках, але авторські права не можуть захистити саму ідею. Натомість він

захищає фіксовані, відчутні носії виразності, які можна відтворити, тобто остаточний письмовий чи художній твір.

Історично склалося, що комп’ютерні програми не захищались авторським правом, оскільки до 1974 року комп’ютерні програми не розглядалися як фіксовані матеріальні об’єкти. Однак у 1983 р. Традиційне законодавство про авторські права було розширено на машино-читане програмне забезпечення, а Закон про авторське право надав комп’ютерним програмам такий же статус авторського права, що і літературні твори. Незважаючи на те, що застосовуються багато однакові правові принципи та політики, існує низка різних питань, що виникають із авторським правом на програмне забезпечення.

Порушення авторських прав на програмне забезпечення

Коли ви запускаєте програму на комп’ютері, часто неможливо уникнути копіювання частини коду, оскільки зазвичай відбувається автоматичне копіювання програми, що відбувається в пам’яті комп’ютера, щоб забезпечити функціонування програмного забезпечення. Крім унікального програмного забезпечення, авторські права порушуються не лише шляхом безпосередньої копії оригінального твору, але й шляхом адаптації версій оригіналу.

Так, наприклад, якщо код (вихідний код або скомпільований код) переписано або іншим чином перетворено на іншу комп’ютерну мову, це також вважається порушенням законодавства про авторське право на програмне забезпечення, оскільки це "похідна" робота, і відповідна ліцензія для цього потрібно [4].

Авторські права на програмне забезпечення також можуть бути порушені, навіть не зробивши копії коду. Наприклад, використовуючи оригінальну комп’ютерну програму для “натхнення”, щоб створити той самий функціонал у новій програмі. Навіть якщо фактично не використовується жоден оригінальний код, авторські права на оригінальну програму в деяких випадках можуть бути порушені.

Авторське право на програмне забезпечення є складною сферою права, що розвивається, і, на відміну від інших художніх творів, копії програмного

забезпечення продаються з певними умовами, щоб підкреслити, що є прийнятним використанням.

Як захистити авторське право на програмне забезпечення?

Авторські права на програмне забезпечення переважно використовуються розробниками програмного забезпечення та власниками власного програмного забезпечення для запобігання несанкціонованому копіюванню їх програмного забезпечення. Власником авторських прав є, як правило, творець твору, видавець чи інший бізнес, якому призначено авторське право. Власники авторських прав регулярно застосовують правові та технологічні заходи для запобігання та покарання за порушення авторських прав (частіше називають піратством), коли твори, захищені законом про авторське право, використовуються без дозволу.

Для таких творів, як програмне забезпечення та веб-додатки, вихідний код насамперед є місцем авторського права, і повідомлення про авторські права слід вставити в заголовки всіх файлів вихідного коду, файлів довідки, посібників користувача та / або сторінок "про це програмне забезпечення", щоб зробити твердження про авторське право.

Там, де немає прямого копіювання коду, рядок за рядком, може бути важко довести, що копіювання насправді відбулося. Одним із способів спростити виявлення копіювання є включення зайвого коду або програмних компонентів до складу реального коду. Якщо передбачувана копія включає однакові надлишкові компоненти програми, навіть якщо вони не є рядковими копіями, це може дати дуже сильний висновок про те, що копіювання відбулося.

Незалежні продавці програмного забезпечення повинні бути дуже обережними щодо розкриття вихідного коду. Якщо хтось може самостійно створити з нуля те, що ви створили, просто переглянувши ваш вихідний код, за умови, що код істотно відрізняється, тоді авторські права на ваше програмне забезпечення не порушено. Модифікація вашого програмного забезпечення, захищеного авторським правом, для особистого користування також може вважатися прийнятною під застереженням про “добросовісне використання”, а також про порушення коду та зворотне проектування, коли для цього може бути вказана “законна причина”.

Однак врешті-решт будь-яке несанкціоноване використання програмного забезпечення вважається піратством або крадіжкою, визнаючи комерційну шкоду від порушення прав власника авторських прав.

Захист авторських прав на програмне забезпечення може бути важко забезпечити. Однак використання рішення щодо ліцензування на основі ідентифікаційних даних гарантує, що ви завжди будете знати, хто ваші кінцеві користувачі.

Висновки за розділом 2

У даному розділі наведено опис вразливостей захисту програмного забезпечення, показано основні ділянки розробки в яких частіше всього можна зустріти слабкі місця, вразливості. Описано головні методи захисту на таких ділянках, та виведено декілька рекомендацій по розробці програмного забезпечення, які підвищать рівень захищеності розробки.

Також було описано ключові особливості авторського права, а саме його визначення, яке місце займають програмні продукти в списку, що підлягає під захист авторського права, та способи його захисту в програмних продуктах.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАХИСТУ АВТОРСЬКИХ ПРАВ В ПРОГРАМНИХ ПРОДУКТАХ

3.1 Розробка механізму захисту

Для розробки програмного забезпечення для захисту авторських прав в програмних продуктах було вибрано програмну мову C# та WPF(Windows Presentation Foundation) графічну підсистему для рендерингу користувацького інтерфейсу. Мову було вибрано за наступними характеристиками: він добре інтегрується з Windows, це об'єктно-орієнтована мова програмування, крос-платформерність. При розробці механізму захисту авторських прав в програмних продуктах були визначені наступні характеристики: механізм буде мережевого типу, мати в собі симетричне шифрування та отримувати унікальний ідентифікатор з материнської плати.

Головна ціль такого механізму є захист авторських прав на якомога більший час, для того, щоб втратити найменшу кількість прибутку від програмного продукту, яке буде використовувати цей механізм. Основна логіка механізму захисту полягає в тому, що критичні для функціонування програмного забезпечення функції не будуть знаходитись в самому проекті програмного забезпечення, а зберігатись на сервері. Такий спосіб не дозволить користувачу мати на своєму пристрої всі файли і дані програмного забезпечення, а тому і не дозволить скопіювати його та розповсюдити в вільному доступі в глобальній мережі. При роботі програмного забезпечення всі файли які будуть потрібні для виконання критичних для програмного забезпечення функцій будуть завантажуватись в режимі онлайн і компілюватися відразу в проекті. Після виконання такої функції тимчасовий файл скомпільованої функції буде видалено. Видаляється такий файл відразу по завершенню виконання функції для того, щоб не було можливості мати одночасно більше однієї завантаженої функції. Всі файли які будуть завантажуватись з сервера будуть шифруватися унікальним ідентифікатором пристрою на якому встановлене програмне забезпечення. На

сервері знаходиться база даних користувачів в якій і знаходиться унікальний ідентифікатор, котрий отримується при реєстрації програмного забезпечення користувачем при встановлені. При реєстрації користувач надає наступні дані: логін, пароль, адреса електронної пошти, унікальний ключ, який користувач отримує при купівлі цього програмного забезпечення. Програмне забезпечення з таким механізмом захисту відправляє серійний номер материнської плати, який і буде унікальним ідентифікатором, та додаткову інформації материнської плати на кшталт імені виробник, тип і т.д.

При описі механізму було зазначено, що можна використовувати будь-яке симетричне шифрування. Для реалізації цього механізму в програмному забезпеченні був вибраний симетричний алгоритм блочного шифрування Advanced Encryption Standard (AES) або інша назва Rijndael. Цей алгоритм був вибраний через його легкість в реалізації та досить велику крипостійкість 2^{128}.

Механізм складається з двох частин (сторін) – програмного забезпечення і серверної частини. Кожна частина має свої кроки виконання для реалізації цього механізму (рисунок 3.1).

З сторони програмного забезпечення:

1) при реєстрації надіслати дані материнської плати;

2) при виконанні критичної функції для роботи програмного забезпечення зробити запит на сервер;

3) завантажити зашифрований файл;

4) розшифрувати файл унікальним ідентифікатором (серійним номером материнської плати);

5) скомпілювати цей файл в проект;

6) виконати функцію з скомпільованого файлу ;

7) видалити файл.

З сторони серверної частини:

1) при реєстрації перевірити ключ користувача, та додати в базу даних всю інформацію;

2) при запиті від програмного забезпечення вислати необхідний файл зашифрований унікальним ідентифікатором даного користувача.

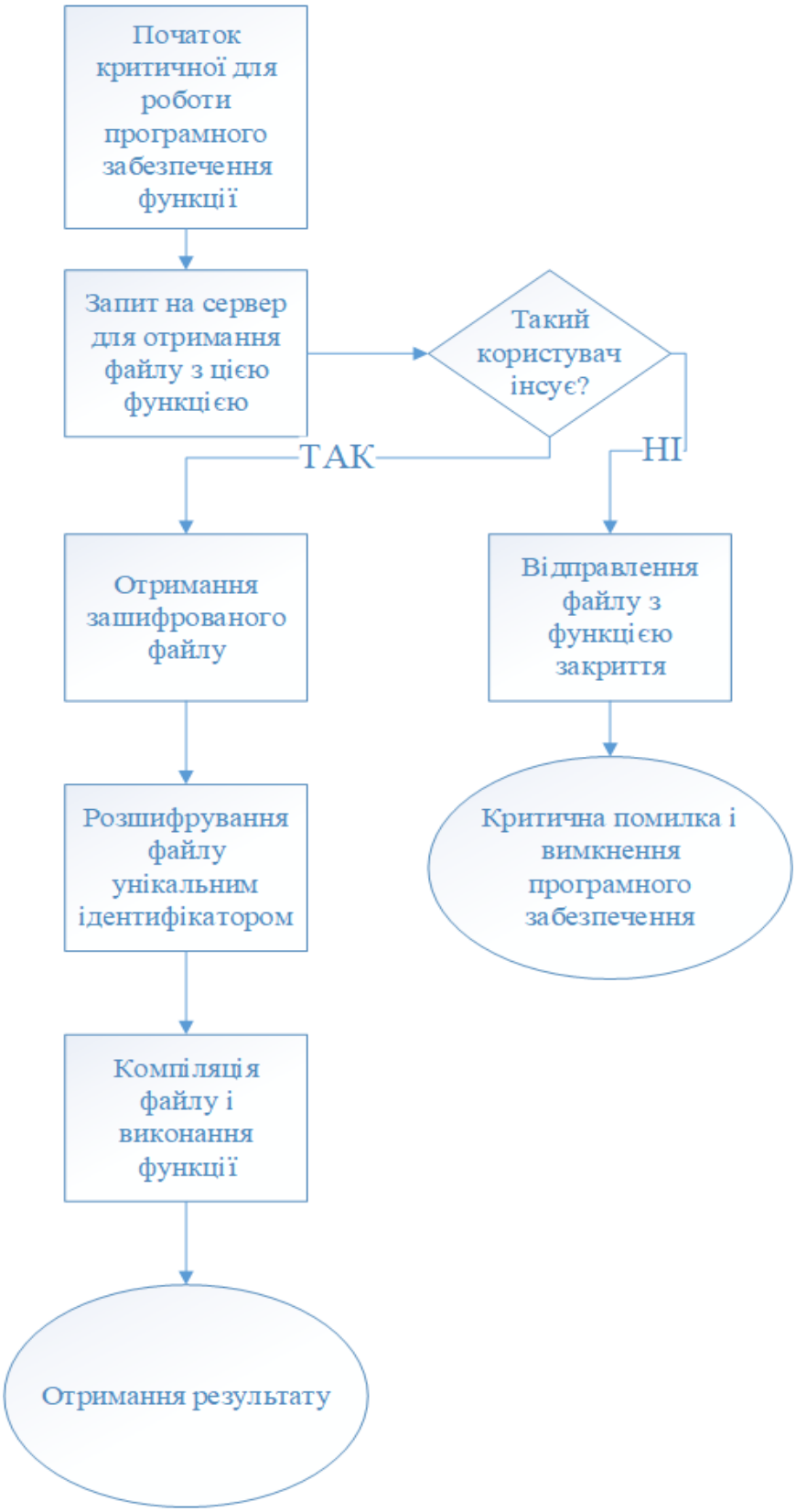

Рисунок 3.1 – Схема механізму захисту авторських прав в програмних продуктах

Переваги такого механізму.

1) Неможливо створити інший сервер, для реєстрації користувачів так, як серверна частина програмного забезпечення відповідає не тільки за реєстрацію користувачів, а і за надання файлів з потрібними функціями;.

2) Кожний користувач має свій особистий ідентифікатор, який зберігається на сервері і водночас зберігається на пристрої в вигляді унікального серійного номера материнської плати.

3) На пристрої не зберігається весь об'єм проекту і тому, скопіювати і дизасемблювати такий проект неможливо.

4) Програмне забезпечення підтримує мультиплатформність і тому його можна використовувати не тільки для десктопних додатків, а і для комп'ютерних ігор, мобільних додатків, додатків для ігрових девайсів в яких є можливість підключення до глобальної мережі.

5) Програмне забезпечення розроблене так, щоб не залежати від розміру проекту. Воно може вбудовуватися в проект вже на стадії розробки, або після того, як проект завершили. Через такий метод імплементації механізму захисту кожне програмне забезпечення буде захищене по різному. Чим більше функцій буде зберігатися на сервері, тим більше часу зловмисниками потребується для злому такого захисту.

Недоліки такого механізму.

1) Для роботи такого програмного забезпечення потрібне постійне підключення до глобальної мережі, так як без цього неможливо буде завантажити файли і програмне забезпечення не зможе функціонувати. Тому такий механізм захисту краще всього використовувати тільки в додатках, які по своїй структурі потребують підключення до глобальної мережі.

2) Більшість механізмів захисту, після розкриття інформації і алгоритму роботи, стають менш захищеними і механізм описаний в цій роботі не виключення. Якщо зловмисники дізнаються алгоритм цього механізму, то зможуть зрозуміти як можна отримати повний проект на одному пристрої. Але щоб це зробити, треба буде втратити багато часу, так як, для отримання цілого проекту треба використати всі

функції і завантажити всі файли проекту хоча б один раз. А для кожного програмного забезпечення різні функції знаходяться в різних місцях і файлах проекту.

3) Один із основних недоліків такого механізму захисту є те, що імплементація його в проект займає велику кількість часу. Для використання такого захисту є 2 способи. Перший спосіб закладається в тому, що треба найняти спеціаліста і разом з розробниками програмного забезпечення створити план вбудовування цього механізму в програмне забезпечення. Але не всі компанії захочуть відкривати вихідний код людині, яка не є співробітником цієї компанії. Другий спосіб як раз для таких компаній, він закладається в тому, щоб навчити розробників вбудовувати цей механізм захисту під час створення програмного забезпечення.

3.2 Побудова архітектури програмного забезпечення

При створенні програмного забезпечення був вибраний архітектурний стиль «зразок проектування». Зразок проектування – це шаблон рішення завдання, яке зустрічається з певною періодичністю. Такий архітектурний стиль можна використовувати всякий раз, коли виникає така задача, тому він і підходить для механізму описаного в цій роботі. Через те, що виклик функції відбувається на протязі всієї роботи з програмним забезпеченням, була потрібна архітектура, яка надавала таку можливість. При створенні архітектури були виділені основні характеристики:

1) ефективність системи – в першу чергу програмне забезпечення повинне вирішувати поставлену задачу по захисту авторських прав в програмних продуктах;

2) масштабованість системи – система повинна справлятися зі додатковими навантаженнями;

3) відкритість системи – система досить гнучка і не чинить опір змінам, так як кожна система змінюється з часом і до неї додаються нові функції;

4) модульність – система побудована таким чином і характер залежності такий, що можна використовувати будь-які компоненти окремо від інших;

5) мінімальна кількість повторень – застосування абстракцій і внесення великих частин повторюваних запитів в окремі методи;

6) можливість повторного використання – система повинна бути спроектована таким чином, щоб її фрагменти можна було використовувати в інших системах;

Після врахування архітектурного стилю, основних характеристик і особливості механізму була створена архітектура, яку можна побачити на рисунку 3.2.

Розроблене програмне забезпечення реалізує механізм захисту авторських прав та має дві частини: офлайн частина, яка встановлюється на пристрій та онлайн частина, яка знаходиться на сервері. В офлайн частині реалізована більша частина логіка додатка, окрім основних функцій, які знаходяться на сервері. Щоб не навантажувати офлайн частину системи, більшість перевірок робляться на сервері. Після перевірок, сервер лише надсилає відповідь “True” або “False”.

Кожний компонент програмного забезпечення був розроблений, як окремий модуль, який можна використати в будь-якому іншому додатку. Основна логіка механізму захисту починається запитом на сервер для отримання файлу с функцією для виконання і закінчується компіляцією файлу та видаленням його з системи.

Для роботи системи потрібно дві бази даних, перша знаходиться на серверній частині, в яку записується вся інформація про користувача, а саме: логін, пароль, унікальний ідентифікатор, ключ програмного забезпечення. Друга база даних знаходиться в офлайн частині у встановленому додатку. В неї записується інформація про всі файли та методи для компіляції.

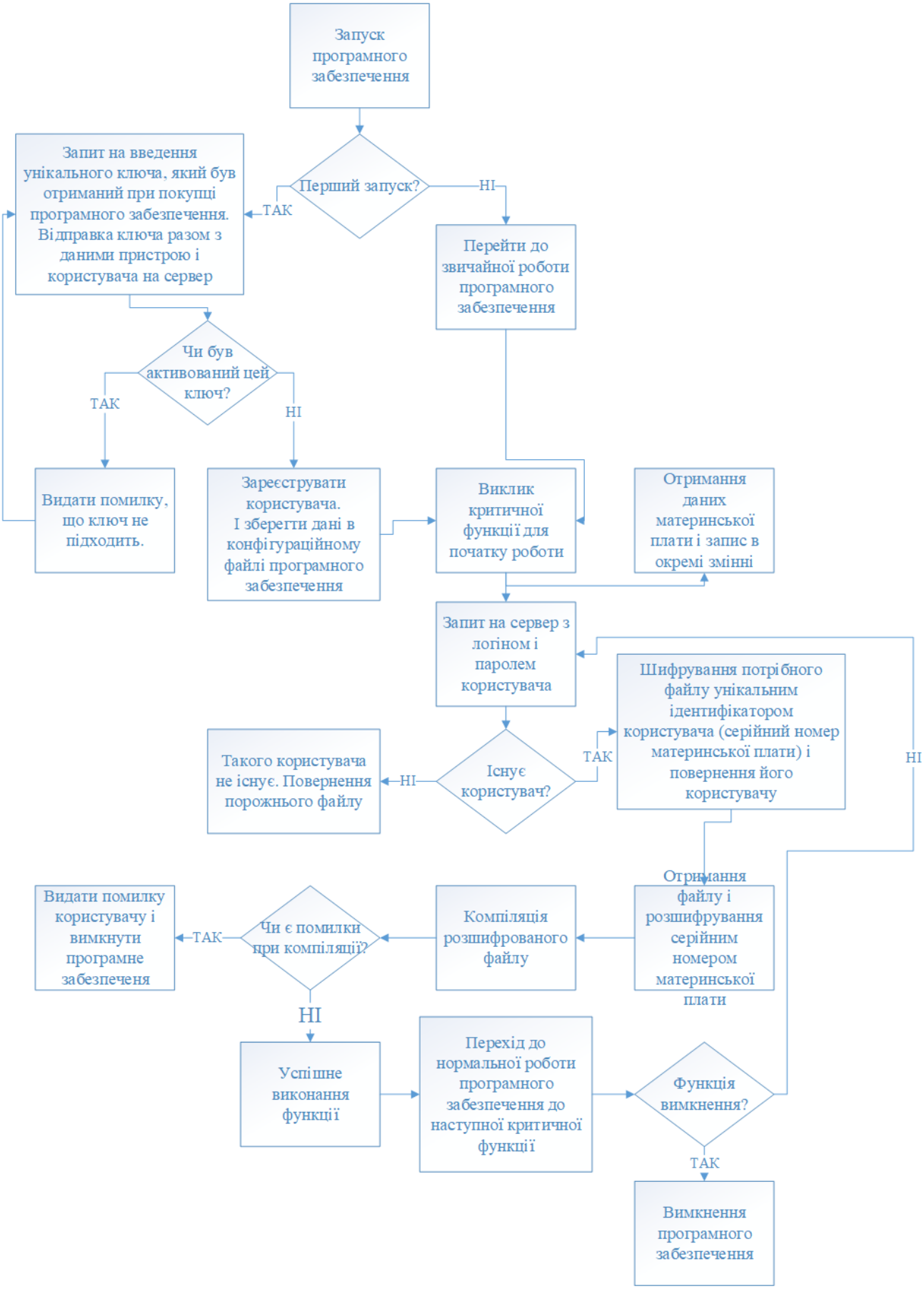

Рисунок 3.2 – Архітектура програмного забезпечення

3.3. Графічний інтерфейс роботи додатка

Розроблений додаток призначений для використання демонстрації роботи механізму та його модулей на пристроях з операційною системою Windows. Для роботи додатку треба завантажити ProjectOne.exe, а також встановити бібліотеку .NET Framework версії 3.5 і вище.

Якщо всі складові встановлені і завантажені, то запустити додаток можна за допомогою файлу ProjectOne.exe. Рисунок 3.3.

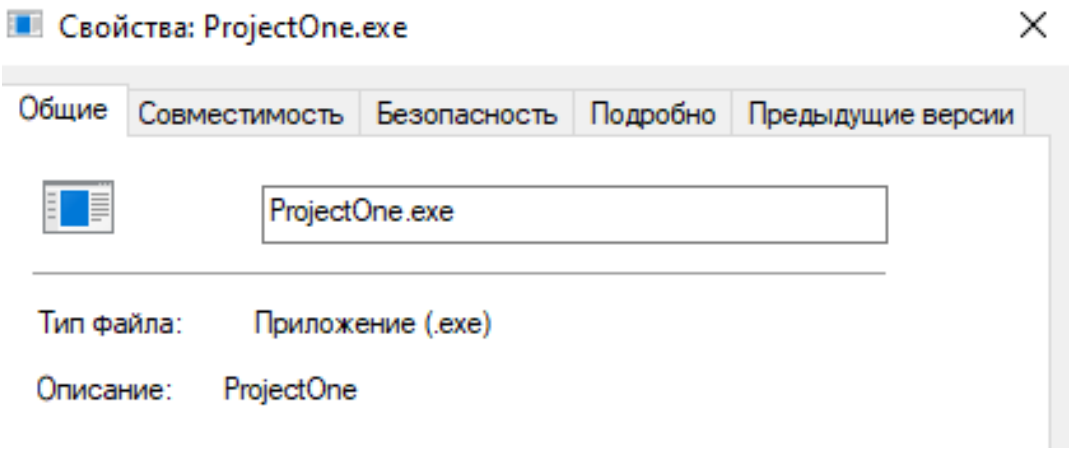

Рисунок 3.3 – Запускаючий файл програмного забезпечення

Після запуску цього файлу відкриється перша вкладка додатку «Шифрування». (Рисунок 3.4). При введені даних та ключа можна зашифрувати та розшифрувати будь-який текст за допомогою симетричного алгоритму блочного шифрування AES. На рисунку 3.4 та рисунку 3.5 можна побачити шифрування і розшифрування тексту “Test text” з ключем “amgjdk14d9t5n39g”.

Друга вкладка «SQL запити» показує роботу з базою даних, яку використовує основний механізм для отримання даних при компіляції файлу. (Рисунок 3.6 – 3.7) Є можливість отримати всі дані з бази даних, та зробити вибірку по окремим стовбцям з умовою WHERE чи без неї.

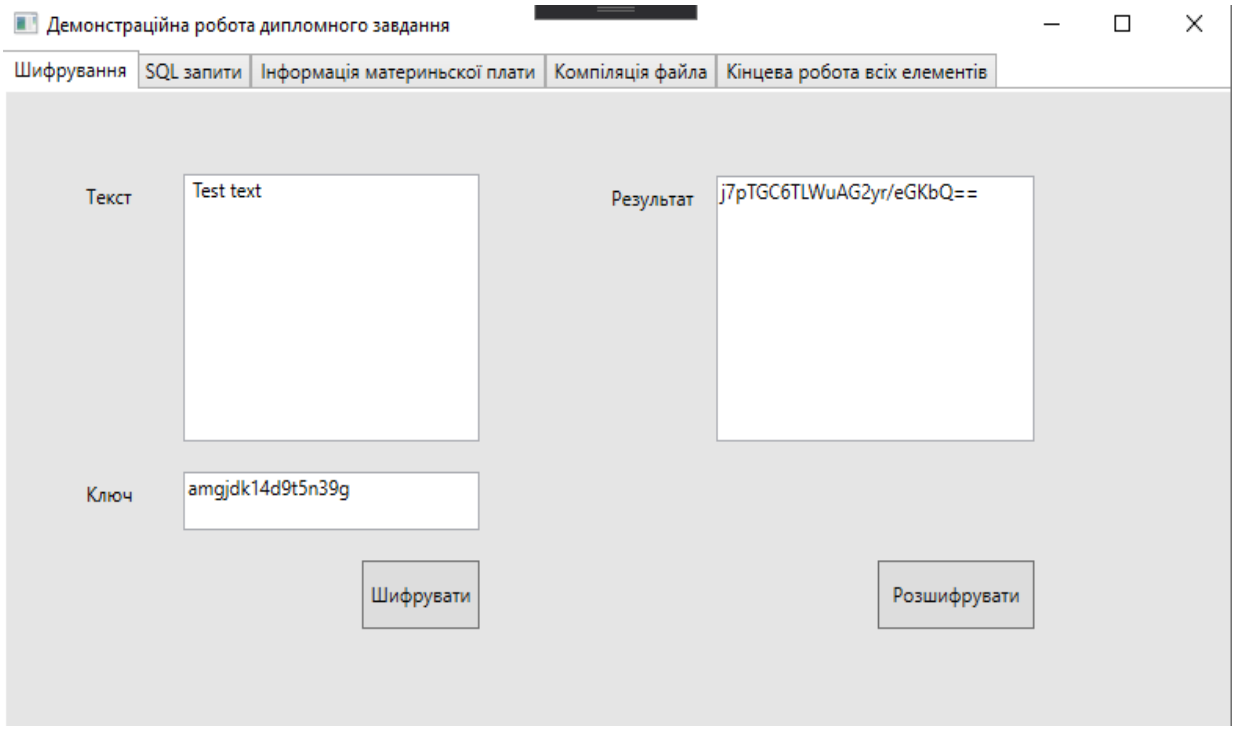

Рисунок 3.4 – Вкладка Шифрування

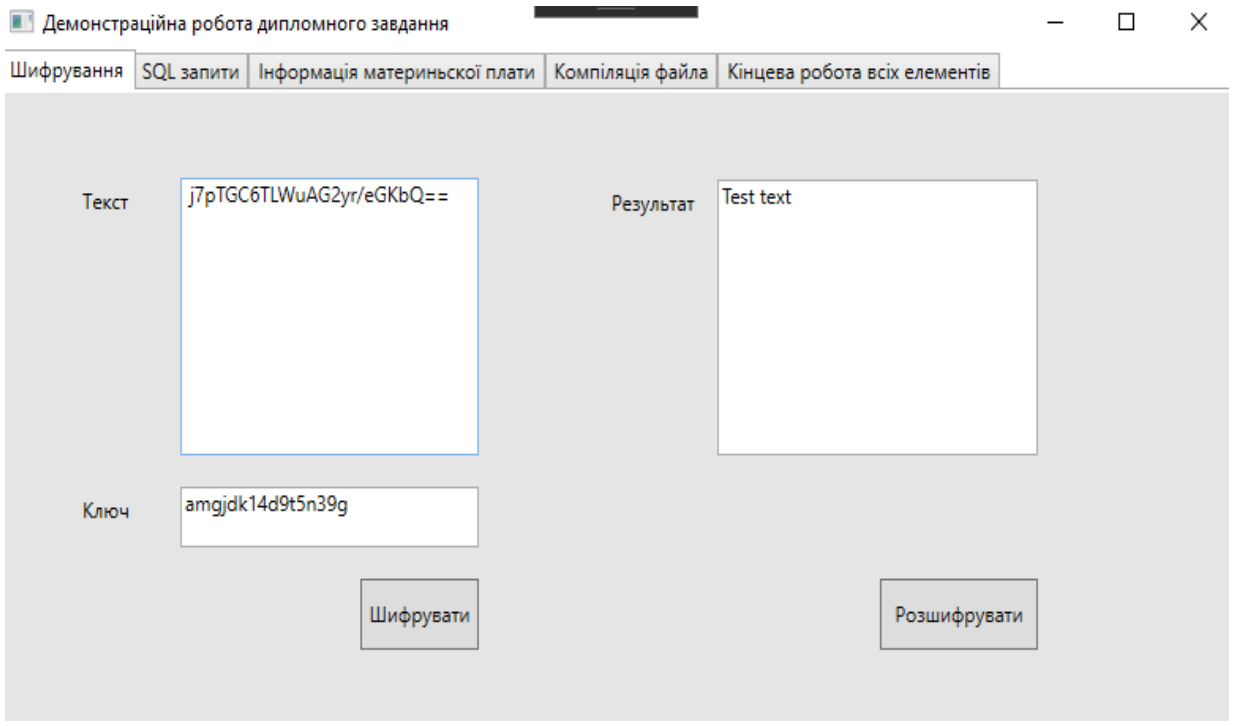

Рисунок 3.5 – Вкладка Шифрування

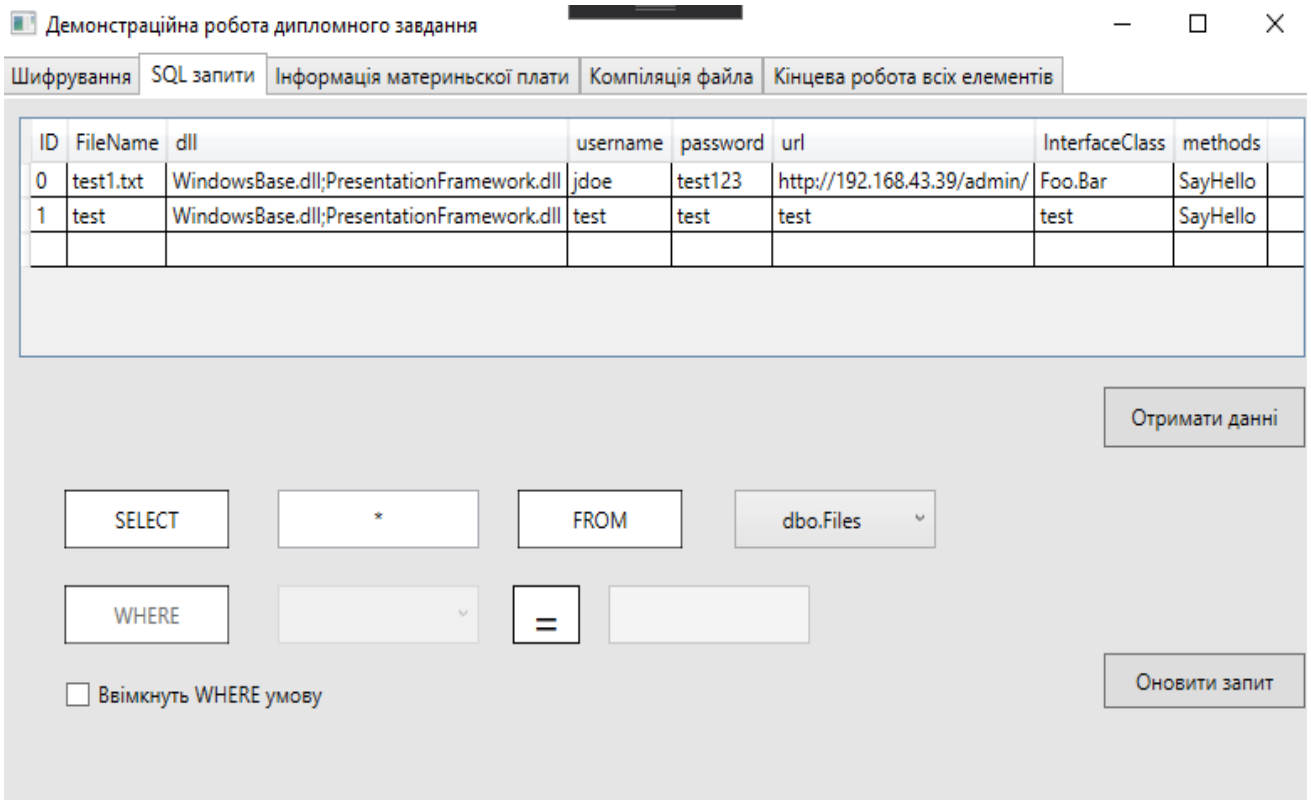

Рисунок 3.6 – Вкладка запитів

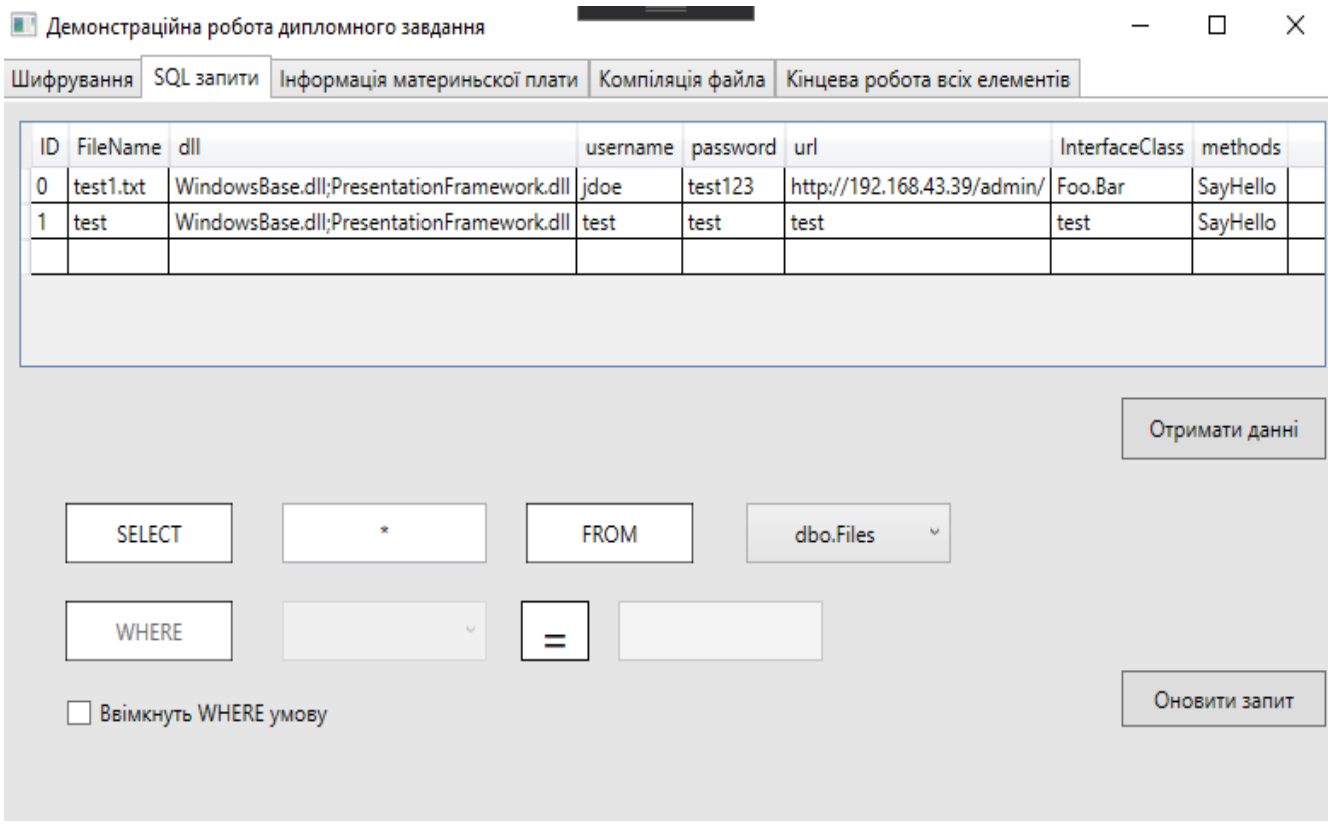

Рисунок 3.7 – Вкладка запитів

Третя вкладка «Інформація материнської плати» показує, що при натискання кнопки в режимі реального часу можна отримати дані про материнську плату такі як: стан материнської плати, компанія виробника, первинний тип шини, номер продукту, можливість заміни та унікальний серійний номер. (Рисунок 3.8)

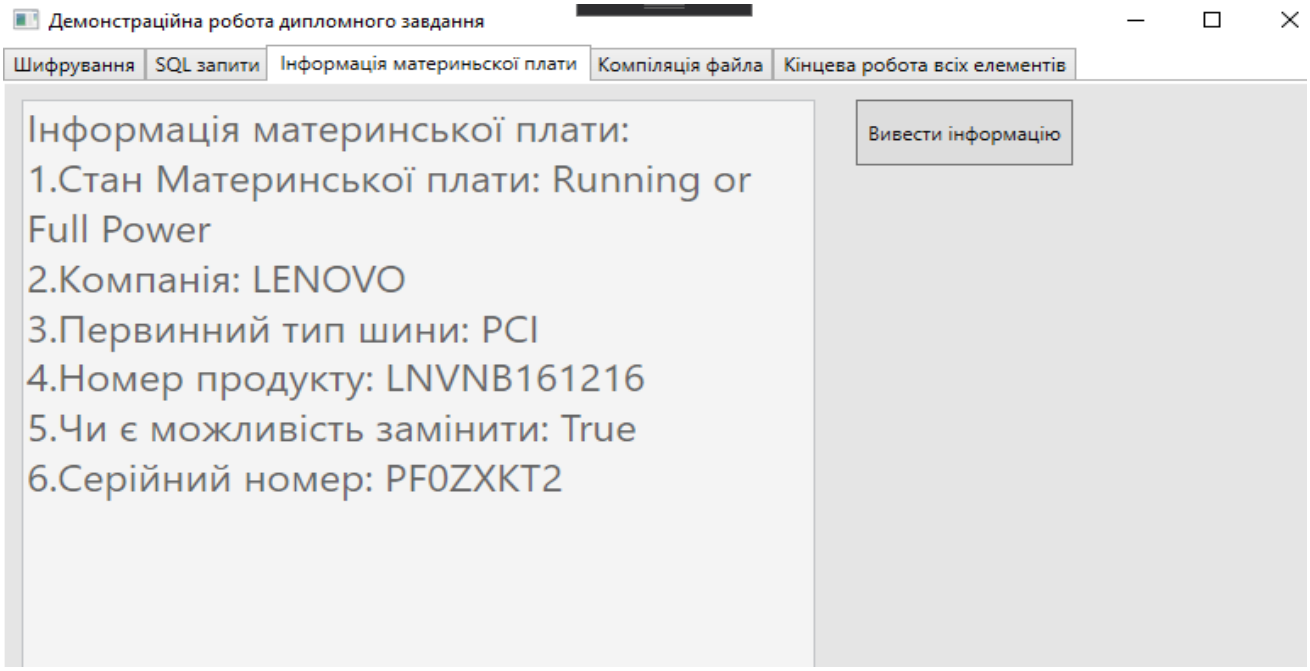

Рисунок 3.8 – Вкладка інформації материнської плати

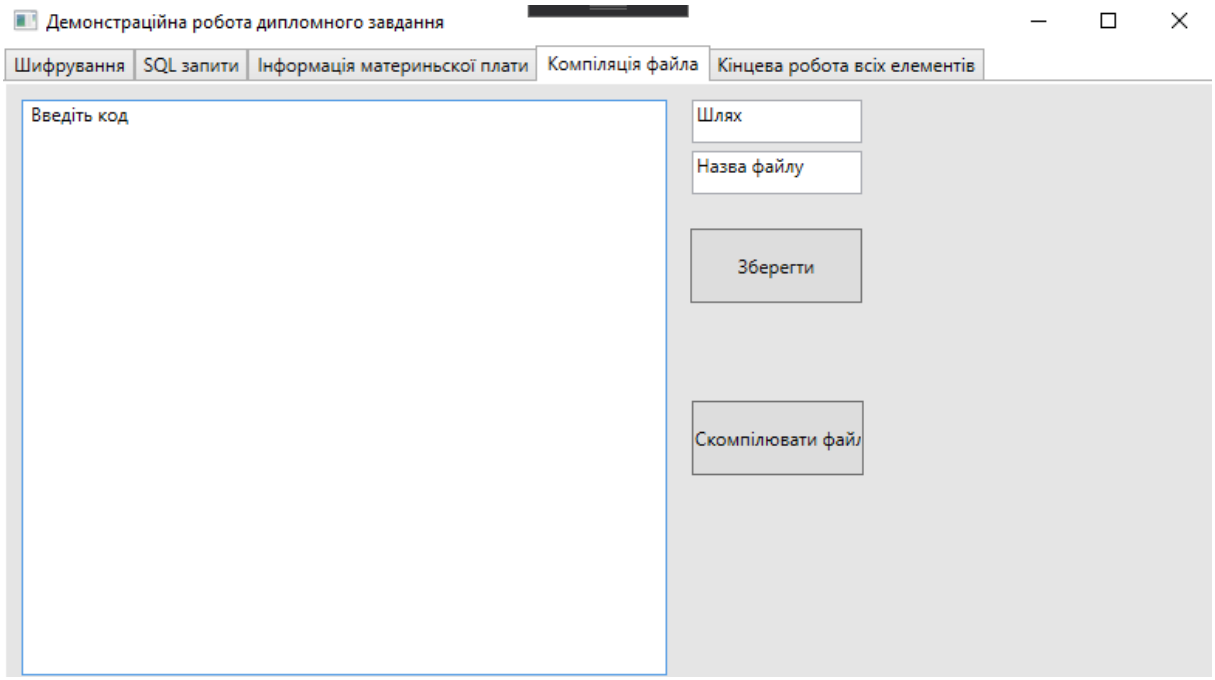

Рисунок 3.9 – Вкладка компіляції файлу

Четверта вкладка «Компіляція файлу» дозволяє написати файл та зберегти його, а потім вибрати цей файл, та скомпілювати (Рисунок 3.9 – 3.12). Під час компіляції вибраного файлу йде автоматичний запит на сервер і отримання файлу з іншою функцією, який компілюється в один час з вибраним файлом та виконується після нього.

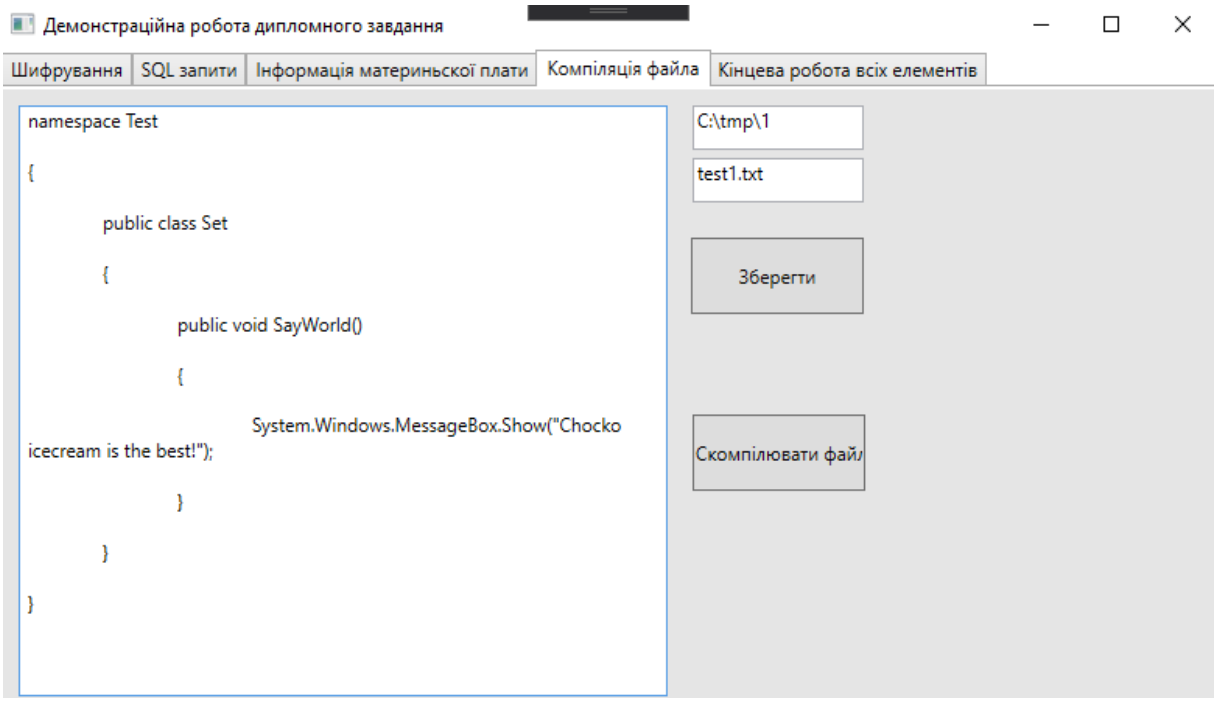

Рисунок 3.10 – Вкладка компіляції файлу

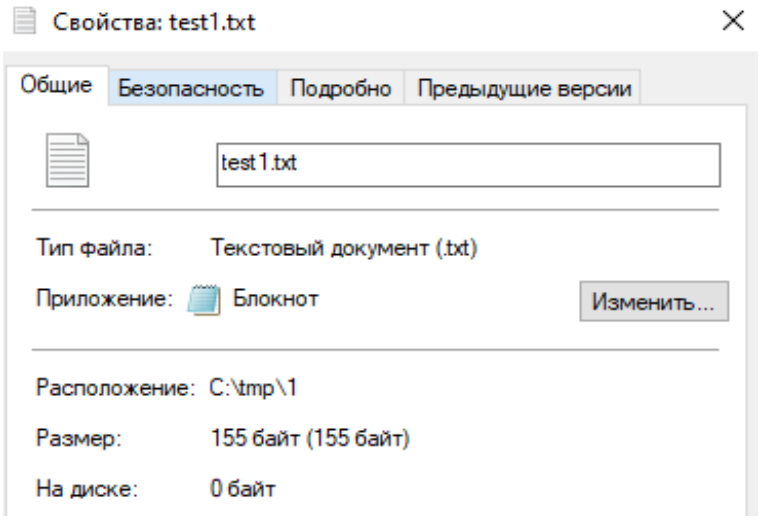

Рисунок 3.11 – Створений файл

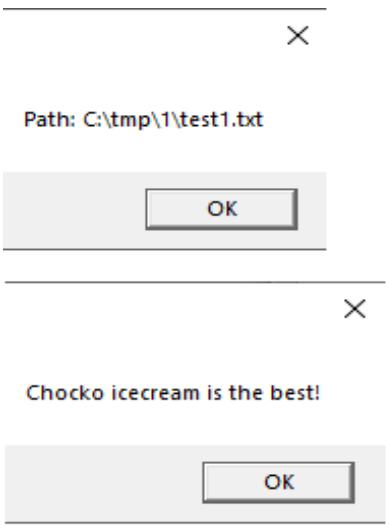

Рисунок 3.12 – Результат компіляції

П’ята вкладка «Кінцева робота всіх елементів» показує всі шаги які були виконані при компіляції завантажені і компіляції файлу (Рисунок 3.13). Два SQL запити де спочатку отримуються такі дані: логін користувача, його пароль, посилання на файл та назву простору імен з назвою класу і функції. В другому запиті було отримано назви DLL (Dynamic-link library), які потрібні для компіляції файлу. Зашифрований і розшифрований текст файлу, який було завантажено з серверу. Звіт про те, що під час компіляції не сталося помилок і метод успішно запустився і виконався.

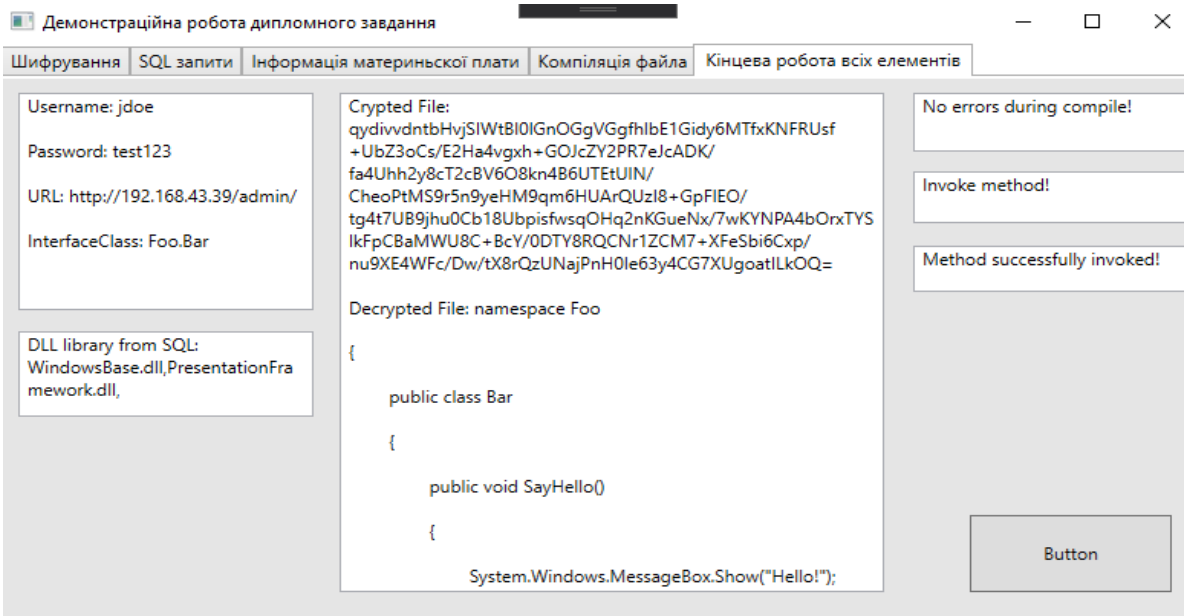

Рисунок 3.13 – Звіт про виконану роботу

Висновки за розділом 3

У даному розділі описані механізм захисту авторського права в програмних продуктах, , методика роботи користувача з інтерфейсом додатку. Також описані ключові елементи інтерфейсу, та архітектура програмного забезпечення з її характеристиками.

РОЗДІЛ 4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕНН ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1. Опис ключових класів програмного забезпечення

Діаграма класів — це діаграма призначена для представлення моделі статичної структури програмної системи. На діаграмі клас зображується прямокутником, розділеним на 2 частини: ім’я класу та операції. При розробці системи важливу роль відіграють як структура класу, так і відносини між ними. Базові типи зв’язків: асоціація, агрегація, композиція і узагальнення.

Класи розробленої програми і зв’язки між ними зображені на рисунку 4.1.

При програмуванні настільного додатку використана технологія WPF. В цій технології графічний інтерфейс з програмним кодом зв’язує мова розмітки XAML(Extensible Application Markup Language). Основні класи програми:

- MainWindow – основний клас з-за допомогою якого ми маємо змогу взаємодіяти з іншими класами. Віун має конструктор MainWindow() в якому викликається функція InitializeComponent(), що запускає процес ініціалізації всіх об’єктів графічного інтерфейсу і є точкою входу в програму. Вся логіка активних графічних елементів, таких як кнопка, список, таблиця, текстові блоки і т.д., зберігається в цьому класі. Кожна функція, яка взаємодіє з цими елементами, має структуру «void» і отримує два значення: object sender, RoutedEventArgs e.
- Config – клас конфігурації, який може використовуватися для загальних налаштувань додатка і зберігання даних. В ньому міститься змінна List<string> Report, в яку зберігається дані про виконання функцій класу FileCompiler.
- Crypt – клас, який автоматизує використання класу AES з офіціальних бібліотек C# .NET. Має дві функції Static string EncryptString() та static string DecryptString(), що отримують два значення: string key, string cipherText. В кожній функції йде перевірка ключа на його довжину. Допустимі значення - 128/192/256 біт.

- Files – клас, в якому оголошені тільки змінні. Цей клас потрібен, як один із компонентів таблиці при виводі значень з бази даних.
- MotherBoardInfo – клас, який відповідає за надання даних про материнську плату. Має дві головні змінні ManagementObjectSearcher baseboardSearcher та ManagementObjectSearcher motherboardSearcher, що відповідають за запити до root\\CIMV2, що містить класи для комп'ютерного обладнання та їх конфігурації.
- SQLHandler – клас, який відповідає за роботу з базою даних, а саме за отримання і конвертацію даних з одного типу в інший. Має два різних конструктора для різних випадків використання та шість функцій: int GetCountOfItems(), ObservableCollection<Files> GetCollectionOfData(), string GetStringValue(), List<string> GetStringList(), Int GetIntValue(), void UpdateSqlCommand(). Для зручності роботи з різними запитами була створена функція UpdateSqlCommand(), вона дозволяє зекономити оперативну пам'ять тим, що після використання змінної цього класу, ми можемо змінити команду, а не створювати нову змінну типу класу SQLHandler. Для виводу інформації в таблицю використовується функція ObservableCollection<Files> GetCollectionOfData().
- FileCompiler – другий основний клас, який збирає в собі всі ключові класи і реалізує логіку механізму захисту авторських прав. Має також два конструктора для різних випадків використання та сім функцій: string DemonstrateCompileMethod(), void DeleteFile(), MethodInfo CompileMethod(), void InvokeMethod(), void Clear(), string DownloadFileFromURL(), void GetData().

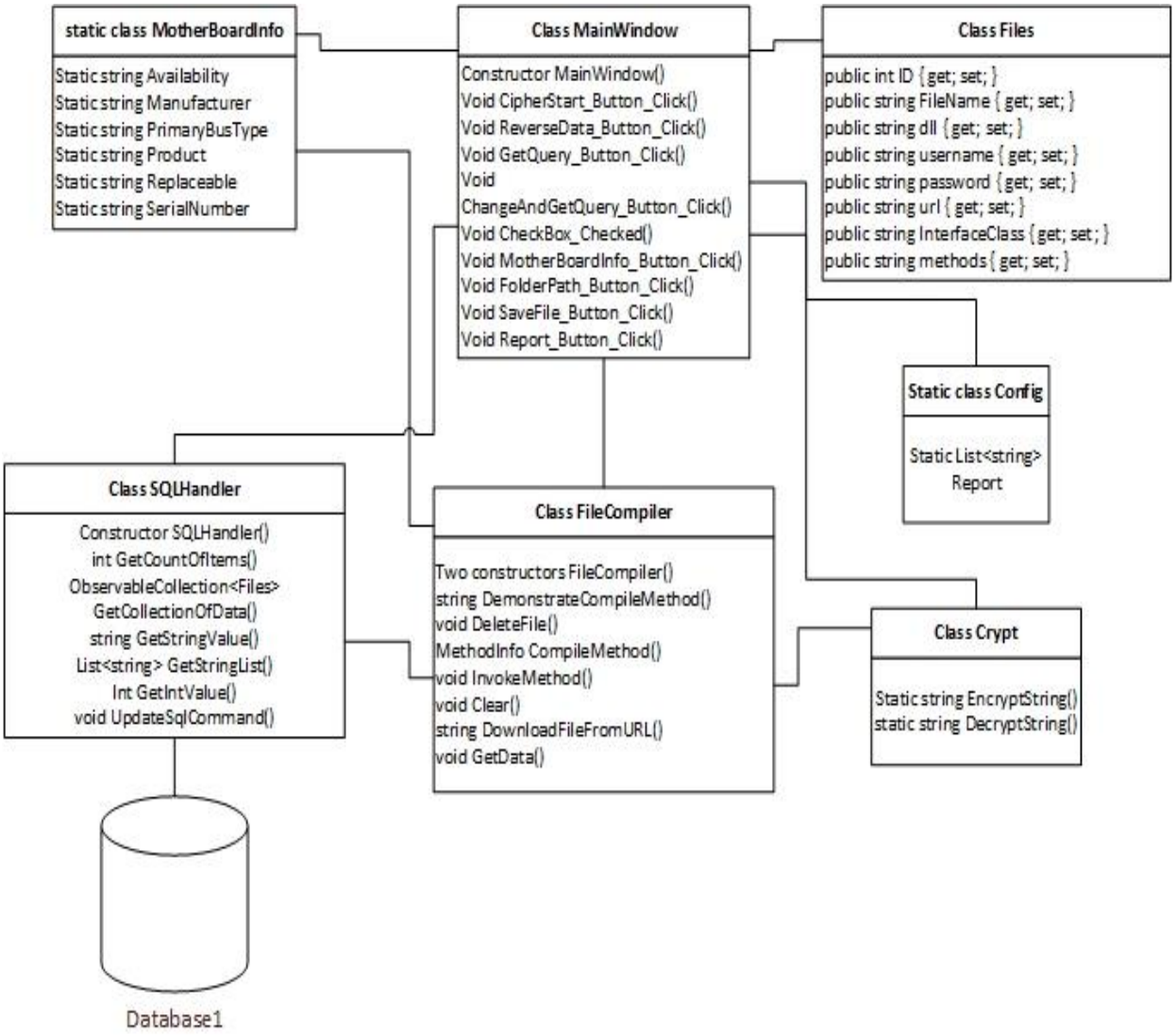

Рисунок 4.1 – Схема ключових класів

4.2 Тестування програмного продукту на помилки

Щоб протестувати програмне забезпечення з механізмом захисту авторського права, треба додати до оригінального проекту декілька додаткових строк коду, щоб отримувати інформацію на кожному етапі роботи, які зазвичай приховані від користувача.

Після додавання потрібних строк, треба переглянути місцезнаходження всіх файлів та їх зміст. Так, як файл з функцією знаходиться на окремому сервері, то треба перевірити його доступність і цілісність. На рисунку 4.2 зображений шлях до

файлу, а на рисунках 4.3 - 4.5 можна побачити зміст самого файлу. Функція виводить вікно з фразою “Hello!” на екран користувача.

Щоб перевірити підключення до серверу, треба дізнатись його адресу за допомогою команди ifconfig. На рисунку 4.5 зображений вивід функції і IP-адресу 192.168.43.39.

Після вводу цього адресу в браузері можна побачити стартову сторінку Apache на рисунку 4.7. При запиті, на сервері був ввімкнений wireshark і на рисунку 4.6 зображений трафік з HTTP-запитами.

Так як всі компоненти були перевірені і справно працювали під час тестування, можна запускати програмне забезпечення. Головна ціль тестування, перевірити спроможність програмного забезпечення завантажувати файл з серверу, та можливість компіляції цього файлу. string source = DownloadFileFromURL(username, password, @"c:\tmp\", fileName, url + fileName); - ця строка відповідає за завантаження файлу в директорію C:\tmp\. При завантажені файлу можна побачити запит “GET /admin/test1.txt” на рисунку 4.11 . На рисунку 4.12 зображений завантажений файл в цій директорії.

Коли завантаження завершилось, програмне забезпечення почало дешифрування і компіляцію файлу. На рисунку 4.10 зображено розшифрований зміст файлу. При компіляції створюється тимчасовий файл з іменем зображеним на рисунку 4.12.

Всі компоненти програмного забезпечення працюють без помилок.

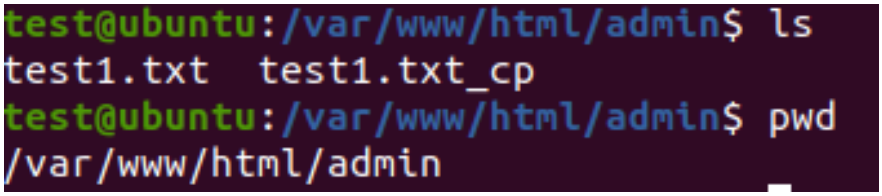

Рисунок 4.2 – Файл та шлях до нього

Рисунок 4.3 – Зашифрований файл

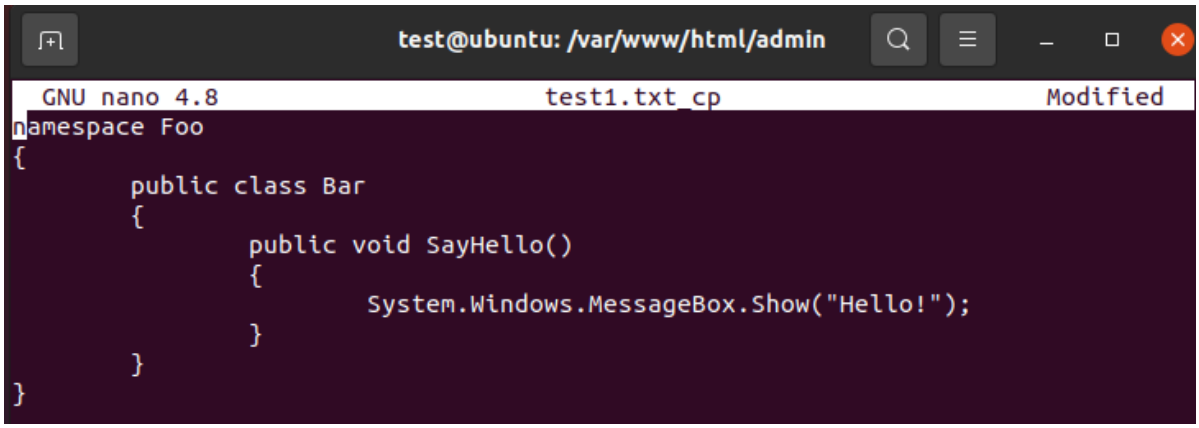

Рисунок 4.4 – Розшифрований файл

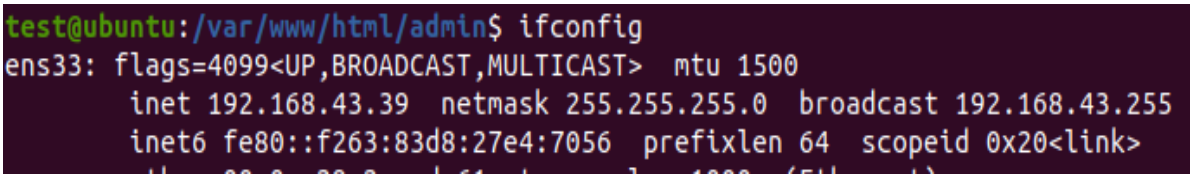

Рисунок 4.5 – IP-адреса серверу

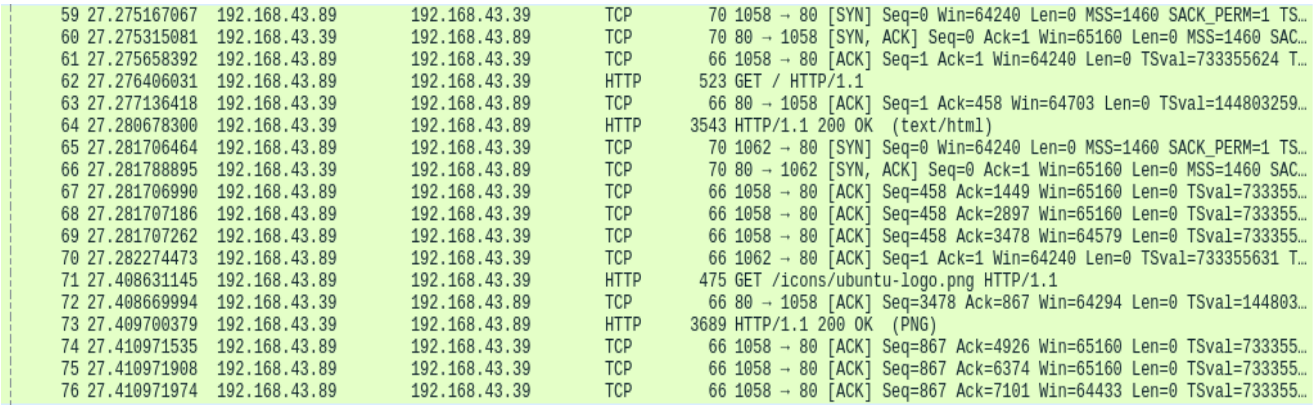

Рисунок 4.6 – Перехоплені пакети

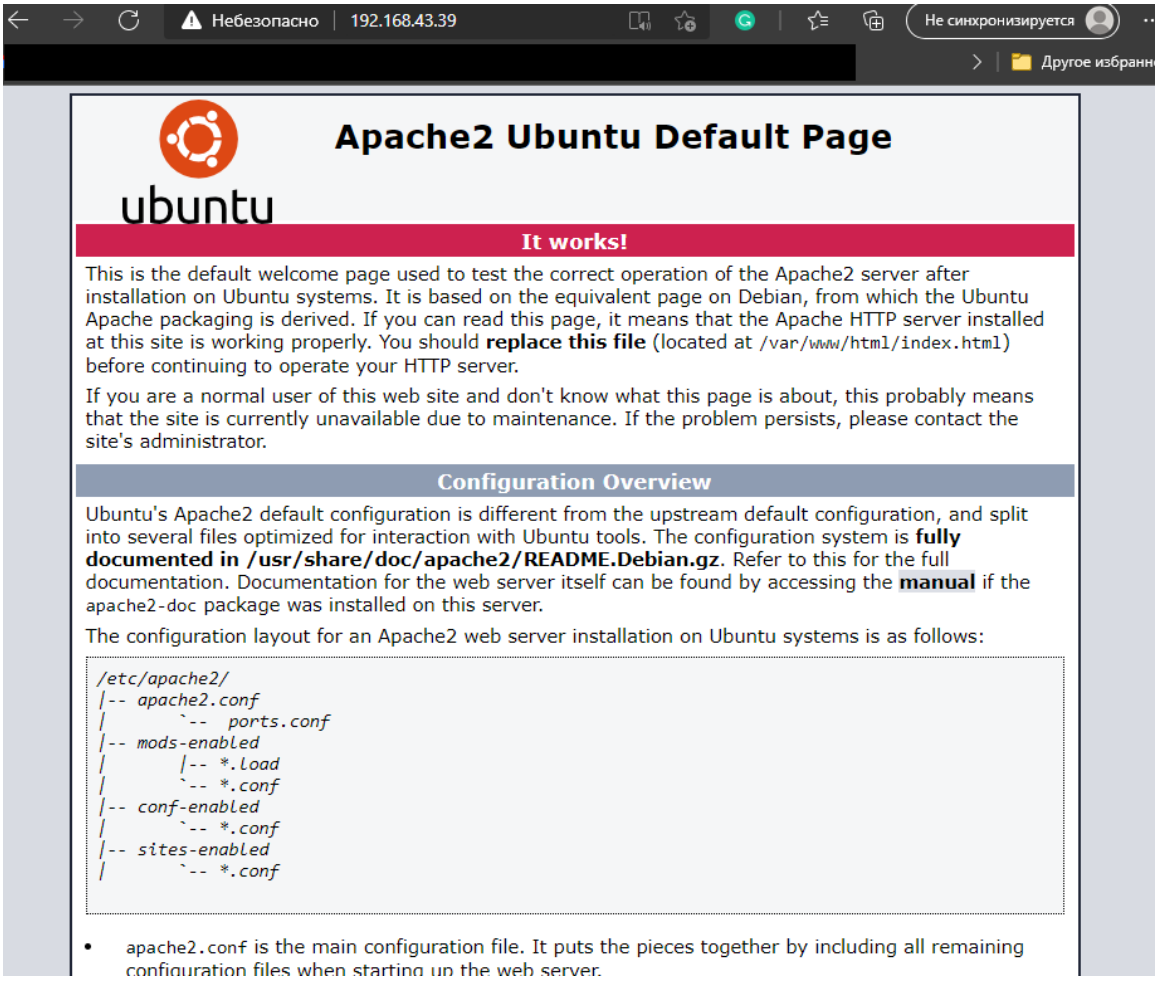

Рисунок 4.7 – Головна сторінка серверу

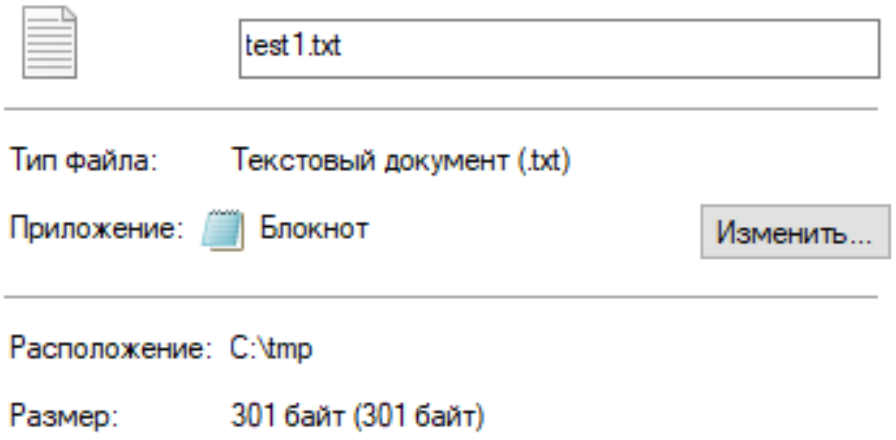

Рисунок 4.8 – Завантажений файл

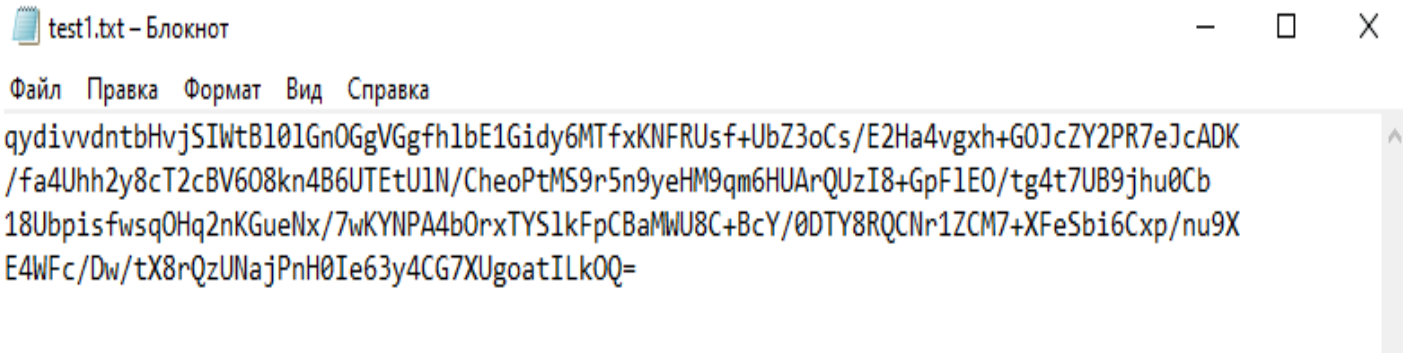

Рисунок 4.9 – Зміст завантаженого файлу

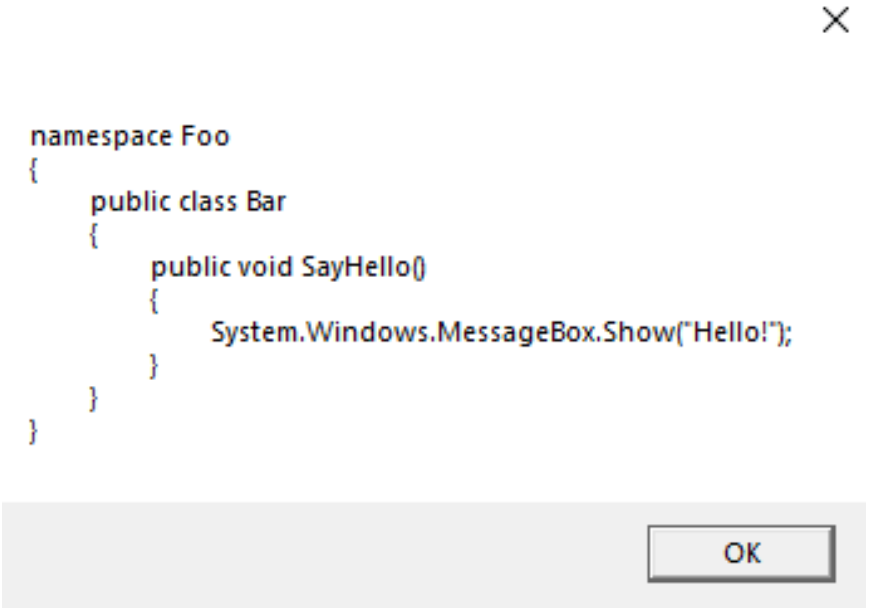

Рисунок 4.10 – Розшифрований зміст файлу

```
 70 18988 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T…
 70 80 → 18988 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
 66 18988 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0 TSval=735633987 …
144 GET /admin/test1.txt HTTP/1.1
 66 80 → 18988 [ACK] Seq=1 Ack=79 Win=65082 Len=0 TSval=145031088…
797 HTTP/1.1 401 Unauthorized  (text/html)
 66 18988 → 80 [ACK] Seq=79 Ack=732 Win=65160 Len=0 TSval=7356339…
159 GET /admin/test1.txt HTTP/1.1
 66 80 → 18988 [ACK] Seq=732 Ack=172 Win=64989 Len=0 TSval=145031…
620 HTTP/1.1 200 OK  (text/plain)
```

Рисунок 4.11 – Перехоплення пакетів з запитом на завантаження

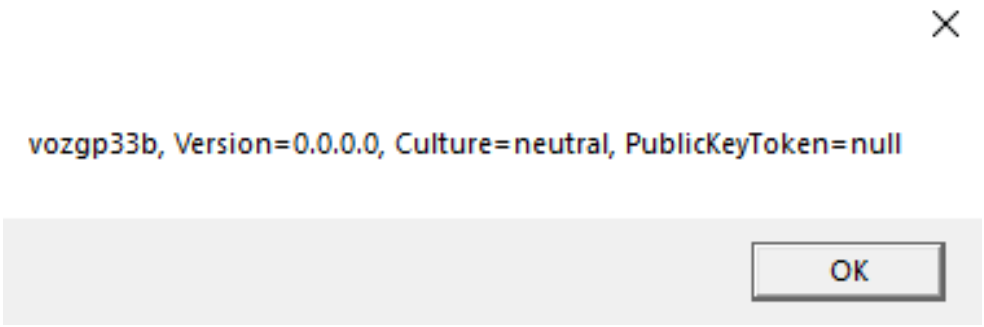

Рисунок 4.12 – Назва тимчасового файлу

4.3. Інструкція по використанню

Як вже було вказано раніше, цей механізм можна вбудовувати в програмне забезпечення, як на стадії розробки, так і на стадії готового проекту.

Додавання механізму захисту на стадії готового проекту.

1. Щоб додати цей механізм, треба визначити всі критичні для роботи проекту функції, та виділити з них ті, які викликаються не досить часто. Наприклад функції завантаження нового вікна при натисканні клавіші, візуалізація інтерфейсу чи текстури.

Не рекомендується відносити функції, які часто викликаються під час роботи програми, до критичних для роботи. Це може знизити продуктивність, збільшити навантаження на систему та зменшити пропускну можливість мережевої карти через очікування завантаження потрібної функції кожен раз та постійне надсилання та отримання запитів від серверу.

2. Після винесення критичних функцій в список треба виписати кожну функції і її дані в окремі класи.

Не рекомендується виносити всі функції в один клас, чи у вже існуючий клас в проекті, це може викликати помилки несумісності та повторюваності назв в просторі імен.

Якщо функція викликає системні чи користувацькі методи з проекту, треба вказувати весь шлях до цієї функції. Наприклад при виклику функції

MessageBox.Show(string object) треба вказувати до якої бібліотеки вона відноситься - System.Windows.MessageBox.Show(string object). Чи якщо треба викликати метод з проекту – ProjectOne.Crypt.EncryptString(). ProjectOne – це простір імен проекту, Crypt – клас, де EncryptString є метод, який потрібно викликати в скомпільованій функції.

3. Якщо база даних в проекті вже є, то створити окрему таблицю, якщо бази даних нема, то створити нову базу і в ній створити вже таблицю та обновити дані в SQLHandler. Додати в базу даних наступні дані: посилання на місцезнаходження файлу на сервері, IP-адреса чи доменна адреса сервера, назва файлу, назва функції, назва простору імен, назва класу, список DLL для роботи цієї функці.

4. До проекту додати класи FileCompiler, SQLHandler, MotherBoardInfo, Crypt.

5. На місці де потрібно було викликати функцію викликаємо нову функцію класу FileCompiler.CompileMethod в які передаємо назву файлу і функцію для виконання і зберігаємо результат в змінну.

6. Наступна функція яку треба викликати є FileCompiler.InvokeMethod(), якщо функція має якісь вхідні параметри, то треба в функцію InvokeMethod додати масив цих параметрів.

7. Якщо всі попередні шаги виконані правильно, то при запуску програмного забезпечення все повинно працювати, як і до встановлення захисту, але встановлений проект має важити менше і трохи більше навантажувати мережу. Якщо всі рекомендації були виконані правильно, то збільшене навантаження на мережу помітне не буде.

Додавання механізму захисту на стадії розробки.

1. Додати до проекту класи FileCompiler, SQLHandler, MotherBoardInfo, Crypt.

2. При розробці програми додавати критичні функції до окремого списку і вже відразу додавати їх в окремі файли.

3. На місці виклику заздалегідь писати функції виклику FileCompiler.CompileMethod і FileCompiler.InvokeMethod().

Рекомендується створити конфігураційний файл, в якому створити змінні для цих двох методів. Так, як змінювати один файл набагато легше ніж відкривати кожен клас окремо і в параметрах виклику змінювати значення власноруч [7].

4. Також як і в попередньому списку треба додавати інформацію до бази даних в проекті якщо вона вже існує. Наприкінці роботи, оновити дані в класі SQLHandler, та класі конфігурації, якщо такий є.

5. Якщо все зроблено правильно, то проект має функціонувати без труднощей.

Такий механізм захисту рекомендується додавати до програмного забезпечення на стадії розробки, так як не треба буде робити багато нових змін і шукати по всьому проекту критичні для його роботи функції. Тестувати цю систему набагато легше в стадії розробки ніж після закінчення проекту, так як на стадії розробки набагато менше зв'язків і легше знайти помилку.

4.4. Можливі модифікації програмного забезпечення

Механізм захисту авторських прав в програмних продуктах має можливість модифікації його компонентів, так як він побудован з різних модулей. Є можливість модифікувати чи замінити повністю один із модулей не вплинувши на роботу інших.

Перше й найлегше, що можна модифікувати в цьому механізмі це алгоритм шифрування. В механізмі описаному в цій роботі використовується симетричний алгоритм блочного шифрування AES. Цей алгоритм можна замінити тільки на інший симетричний алгоритм шифрування. Можна використовувати як потоків, так і блочні алгоритми шифрування. Наприклад Twofish, CAST5, RC4, Blowfish, TDES (3DES), Serpent, та IDEA. Ключ буде залишатись унікальний ідентифікатор пристрою у вигляді серійного номеру материнської плати. Такий номер можна модифікувати за певними правилами, щоб на серверній частині і на частині програмного забезпечення створювався один і той же ключ [6].

Якщо розглядати вибір унікального ідентифікатору, то можна використовувати не тільки серійний номер материнської плати, а й будь-який інший унікальний для пристрою номер чи набір символів.

Також можна замінити компонент завантаження даних з серверу. В механізмі, який розглядається в цій роботі, використаний клас WebClient. Такий клас є найпростішим способом завантаження будь-якого файлу з посилання. Він підтримує URI, що починаються з http, https ftp і file ідентифікаторів схем.

Такий спосіб можна замінити на більш новий і складний HttpClient, що являє собою інтерфейс API високого рівня, який поміщає в оболонку функціональні можливості нижнього рівня, доступні на кожній платформі, де вона і виконується. Він розроблен для відправки HTTP-запитів і отримання HTTP-відповідей від ресурсу, ідентифікованого за універсальним кодом ресурсу (URI). Клас HttpClient має можливості завантажувати великі об’єми даних за допомогою потокової передачі файлів. Також цей клієнт використовує потокобезпечні методи на відміну від WebClient та має унікальні методи роботи для багатьох платформ, як Windows/. NET Framework, Windows/Mono та інші.

При розробці механізму була використана база даних для зберігання інформації про функцію і файл в якому вона знаходиться. Не всі розробники додають в свої проекти бази даних, а створювати і додавати її до проекту лише, щоб зберігати там ці дані не завжди є оптимальним рішенням. Сам механізм не потребує саме бази даних, лише місця де можна зберігати потрібні дані для компіляції файлів. Тому використовувати файл типу xlsx, що є форматом файлів Excel. Можна використовувати навіть звичайний текстовий файл, але для цього потрібно буде змінити клас, чи замінити зовсім клас SQLHandler і такий спосіб буде більш складний через те, що до файлу прийдеться записувати всі дані за визначеною формою і пошук по такому файлу буде зробити більше складно з точки зору програмного коду [6].

В механізмі можна не лише модифікувати деякі компоненти, а і додавати нові, тому можна додати захист пам’яті додатку. В цьому випадку йде мова про контроль прав доступу до пам’яті на комп’ютері, а саме до оперативної пам’яті в яку

записуються значення змінних, функцій, класів, інтерфейсів і т.д. Захист пам’яті не закінчується одним контролем прав доступу, ще є можливість при записі значень змінних кожен раз шифрувати інформацію в них і кожен раз розшифровувати при читані.

Висновки за розділом 4

У даному розділі описані ключові класи програмного забезпечення, інструкція по використання, та можливі модифікації механізму захисту. Також було проведене тестування компонентів додатку та продемонстрована його робота. Всі компоненти були в звичайному режимі роботи. Ніяких інцидентів і проблем не з’явилося під час тестування. Офлайн і онлайн частини програмного забезпечення мали змогу обмінюватися файлами. Офлайн частина програмного забезпечення була протестована на помилки, та були додані спеціальні умови, щоб побачити, результати кожного кроку і перевірити їх коректність. Всі модифікації, які були запропоновані для покращення механізму були описанні і підтвердженні документацією класів та методів захисту.

ВИСНОВКИ

В ході виконання роботи розглянута проблема захисту авторських прав в програмних продуктах.

1. Проаналізовано методи та методології побудови програмних продуктів, їх життєвий цикл розробки, що має різну кількість шагів для різних програмних продуктів та методів розробки. Кожний метод побудови має свої переваги і недоліки. Взагалі життєвий цикл розробки доволі комплексний і має свої складнощі при його проектуванні.

2. Проаналізовано методи захисту програмного забезпечення, такі як мережеві та локальні, захист за допомогою електронних ключів, захист коду від аналізу і т.д.. Аналіз вивів, що використання прив’язки програмного забезпечення до параметрів пристроя, шифрування даних та захист програм шляхом перенесення їх в онлайн є найкращим рішенням для створення механізму захисту авторського права.

3. Розроблено механізм захисту авторських прав в програмних продуктах, на базі вже зазначених методів захисту, що має 4 основних етапи: шифрування, компіляція, завантаження з серверу, отримання унікальної інформації про систему.

4. На базі механізму захисту розроблене програмне забезпечення. Воно має дві бази даних в онлайн і офлайн частині, користувацький інтерфейс, який демонструє всі компоненти програмного забезпечення, як окремо, так і разом.

5. Проведено тестування всіх компонентів програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Software Security Engineering: A Guide for Project Managers // By Julia H. Allen, Sean Barnum, Robert J. Ellison, Gary R. McGraw, Nancy R. Mead Published May 1, - 2008 – P. 56 – 70 URL: Software Security Engineering: A Guide for Project Managers: Allen, Julia H., Barnum, Sean, Ellison, Robert J., McGraw, Gary, Mead, Nancy R.: 9780321509178: Amazon.com: Books

2. The Challenges and Solutions of Cybersecurity Among Malaysian Companies // Puteri Fadzline Tamyez, University Malaysia Pahang, Malaysia – 2013 URL: The Challenges and Solutions of Cybersecurity Among Malaysian Companies: Business & Management Book Chapter | IGI Global (igi-global.com)

3. Software Security Issues: Requirement Perspectives // Nikhat Parveen, Md. Rizwan Beg, M. H. Khan – 2008 P. 31

4. Software Copyright [Електронний ресурс]: – Режим доступу: https://www.10duke.com/resources/glossary/software-copyright/#:~:text=Software%20Copyright%20is%20the%20most,the%20death%20of%20the%20author.

5. Aplication of deception to software security // Faculty of Purdue University by Jeffrey K. Avery – 2014 P. 15

6. Improving the Security, Privacy, and Anonymity of a Client-Server Network through the Application of a Moving Target Defense // Chris Morrell, PhD, Electrical and Computer Engineering – May 2016 – P. 29

7. A dissertation submitted to the department if computer science and the committee on graduate studies // A dissertation by Benjamin Livshits – 2011 P. 39

8. Creating secure software Runar Moen Master's Thesis Master of Science in Information Security 30 ECTS // Department of Computer Science and Media Technology Gjøvik University College – 2013

9. Analysis and detection of security vulnerabilities in contemporary software // Pieczul, Olgierd c. 10 - 12

10. Improving Software Security Testing of Software Development Life Cycle (SDLC) // M.S.c In Software Engineering - Asia Pacific University - Ali Fathi Ali Sawehli
11. A security management system design a thesis submitted to the graduate school of social sciences // Middle east technical university – 2010 P. 21
12. Monitoring information systems to enforce computer security policies // by Scott W. Graham Stephen E. Mills – 2016
13. Developing Secure Software in an Agile Proces // Dejan Baca - Blekinge Institute of Technology Doctoral Dissertation Series No. 2012:05 School of Computing
14. What is the Software Development Life Cycle? [Електронний ресурс]: – Режим доступу: https://phoenixnap.com/blog/software-development-life-cycle#:~:text=What%20is%20the%20Software%20Development,%2C%20Test%2C%20Deploy%2C%20Maintain.
15. SDLC – Overview [Електронний ресурс]: – Режим доступу: https://www.tutorialspoint.com/sdlc/sdlc_overview.htm
16. Tackling the Challenges of Cyber Security - First edition // Author: Charles Brookson, Zeata Security Ltd. and Chairman ETSI TC CYBER December 2016
17. Метод захисту програмних продуктів від копіювання // матеріали IV Міжнар. Наук.-практ. Конф. “Проблеми кібербезпеки інформаційно-телекомунікаційних систем” 15-16 квітня 2021 року (PCSITS)

ДОДАТОК

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using Microsoft.CSharp;
using System.CodeDom.Compiler;
using System.Reflection;
using System.Net;
using System.Windows;

namespace ProjectOne
{
   class FileCompiler
   {
      #region Vars
      string fileName = "";
      SQLHandler sQLHandler;
      MethodInfo mi;
      object o;
      Dictionary<string, string> pathToFile;
      List<string> report;
      #endregion

      #region Accesses
      public string FileName
      {
         get
         {
            return fileName;
         }
         set
         {
            fileName = value;
         }
      }

      public List<string> Report
      {
         get { return report; }
      }
      #endregion

      #region Constructors

      public FileCompiler(string connectionString)
      {
         sQLHandler = new SQLHandler(null, connectionString);
         pathToFile = new Dictionary<string, string>();
         report = new List<string>();
      }

      public FileCompiler(string connectionString, string fileName)
```

```
{
    this.fileName = fileName;
    sQLHandler = new SQLHandler(null, connectionString);
    pathToFile = new Dictionary<string, string>();
    report = new List<string>();
}

#endregion

#region Methods

public string DemonstrateCompileMethod(string @path)
{
    string result = "";

    MethodInfo mi = null;
    Dictionary<string, string> providerOptions = new Dictionary<string, string>
        {
            {"CompilerVersion", "v3.5"}
        };
    CSharpCodeProvider provider = new CSharpCodeProvider(providerOptions);
    CompilerParameters compilerParams = new CompilerParameters
    {
        GenerateInMemory  = true,
        GenerateExecutable = false
    };
    //WindowsBase.dll;PresentationFramework.dll
    compilerParams.ReferencedAssemblies.Add("WindowsBase.dll");
    compilerParams.ReferencedAssemblies.Add("PresentationFramework.dll");
    MessageBox.Show("Path: " + @path);
    string source = String.Join("\n", File.ReadAllLines(@path));
    CompilerResults results = provider.CompileAssemblyFromSource(compilerParams, source);
    if (results.Errors.Count != 0)
        for (int i = 0; i < results.Errors.Count; i++)
        {
            MessageBox.Show(results.Errors[i].ToString());
        }
    object o = results.CompiledAssembly.CreateInstance("Test.Set");
    mi = o.GetType().GetMethod("SayWorld");
    mi.Invoke(o, null);
    return result;
}

public void DeleteFile(string fileName)
{
    //delete file
    if(pathToFile.ContainsKey(fileName))
    {
        pathToFile.Remove(fileName);
    }
}

public MethodInfo CompileMethod(string fileName, string method)
{
    string username, password, url, classInterface;

    GetData(out username, out password, out url, out classInterface, fileName);
```

```
    MethodInfo mi = null;
    //add sql for all this credentials
    /*string source = DownloadFileFromURL("jdoe", "test123", @"c:\tmp\",
       "test1.txt", "http://192.168.0.103/admin/test1.txt");*/
    string source = DownloadFileFromURL(username, password, @"c:\tmp\", fileName, url + fileName);
    MessageBox.Show(source);
    Dictionary<string, string> providerOptions = new Dictionary<string, string>
       {
          {"CompilerVersion", "v3.5"}
       };
    CSharpCodeProvider provider = new CSharpCodeProvider(providerOptions);
    CompilerParameters compilerParams = new CompilerParameters
    {
       GenerateInMemory  = true,
       GenerateExecutable  = false
    };

    compilerParams.TempFiles = new TempFileCollection(".", true);

    //Select dll string from database where filename = filename
    sQLHandler.Command = @"Select dll from dbo.Files where FileName = '" + fileName + "'";
    //sQLHandler.Command = "Select dll from dbo.Files where id=0";
    //sQLHandler.UpdateSqlCommand("@filename", fileName);
    sQLHandler.UpdateSqlCommand();
    string[] arrayOfdll = sQLHandler.GetStringValue().Split(';');
    string rep = "DLL library from SQL: ";
    for (int i = 0; i < arrayOfdll.Length; i++)
    {
       rep += arrayOfdll[i] + ",";
       //MessageBox.Show(arrayOfdll[i]);
       compilerParams.ReferencedAssemblies.Add(arrayOfdll[i]);
    }
    report.Add(rep);
    CompilerResults results = provider.CompileAssemblyFromSource(compilerParams, source);
    //MessageBox.Show(results.PathToAssembly.ToString());
    //MessageBox.Show(results.TempFiles.TempDir.ToString());
    //MessageBox.Show(results.CompiledAssembly.Location.ToString());
    MessageBox.Show(results.CompiledAssembly.FullName.ToString());
    //MessageBox.Show(compilerParams.TempFiles.TempDir.ToString());
    if (results.Errors.Count != 0)
    {
       for (int i = 0; i < results.Errors.Count; i++)
       {
          MessageBox.Show(results.Errors[i].ToString());
       }
    }else
    {
       report.Add("No  errors during compile!");
    }
    //add sql credentials
    object o = results.CompiledAssembly.CreateInstance(classInterface);
    mi = o.GetType().GetMethod(method);
    this.mi = mi;
    this.o = o;
    //mi.Invoke(o, null);
    pathToFile.Add(fileName, source);
    return mi;
}
```

```
    public void InvokeMethod(object[] parameters = null)
    {
        report.Add("Invoke method!");
        //MessageBox.Show("Method Invoke!");
        if(mi != null && o != null)
        {
            report.Add("Method successfully invoked!");
            //MessageBox.Show("Method Invoke2!");
            mi.Invoke(o, parameters);
            Config.Report = report;
        }
    }

    public void Clear()
    {
        mi = null;
        o = null;
    }

    string DownloadFileFromURL(string username, string password, string path, string filename, string url)
    {
        using (var client = new WebClient())
        {
            client.Credentials = new NetworkCredential(username, password);
            client.DownloadFile(url, @path + filename);
        }
        string cryptResult = String.Join("\n", File.ReadAllLines(@path + filename));
        //MessageBox.Show(cryptResult);
        string decryptResult = Crypt.DecryptString(MotherBoardInfo.SerialNumber + MotherBoardInfo.SerialNumber,
cryptResult);
        //MessageBox.Show(decryptResult);
        //return String.Join("\n", File.ReadAllLines(@path + filename));
        report.Add("Crypted File: " + cryptResult + "\n" + "Decrypted File: " + decryptResult);
        return decryptResult;
    }

    void GetData(out string username, out string password, out string url, out string classInterface, string fileName)
    {
        sQLHandler.Command = @"Select username from dbo.Files where FileName = '" + fileName + "'";
        sQLHandler.UpdateSqlCommand();
        username = sQLHandler.GetStringValue();
        //MessageBox.Show("Username: " + username);

        sQLHandler.Command = @"Select password from dbo.Files where FileName = '" + fileName + "'";
        sQLHandler.UpdateSqlCommand();
        password = sQLHandler.GetStringValue();
        //MessageBox.Show("Password: " + password);

        sQLHandler.Command = @"Select url from dbo.Files where FileName = '" + fileName + "'";
        sQLHandler.UpdateSqlCommand();
        url = sQLHandler.GetStringValue();
        //MessageBox.Show("URL: " + url);

        sQLHandler.Command = @"Select InterfaceClass from dbo.Files where FileName = '" + fileName + "'";
        sQLHandler.UpdateSqlCommand();
        classInterface = sQLHandler.GetStringValue();
        //MessageBox.Show("InterfaceClass: " + classInterface);
```

```
        report.Add("Username:  " + username + "\n" + "Password:" + password + "\n" + "URL:  " + url + "\n" + "InterfaceClass:"
+ classInterface);
      }

      #endregion
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO;
using Microsoft.CSharp;
using System.CodeDom.Compiler;
using System.Reflection;
using System.Data.SqlClient;
using System.Data;
using System.Collections.ObjectModel;
using System.Diagnostics;

namespace ProjectOne
{
   /// <summary>
   /// Логика взаимодействия для MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {

      #region Variables

      string connectionString = Properties.Settings.Default.Database1ConnectionString;

      #endregion

      public MainWindow()
      {
         InitializeComponent();
      }

      #region Button_Clicks

      #region Tab1

      private void CipherStart_Button_Click(object sender, RoutedEventArgs  e)
      {
```

```
        //MessageBox.Show(new StringBuilder().Insert(0, MotherBoardInfo.SerialNumber, 2).ToString());
        TextRange textRange = new TextRange(
            Input_Text_RichTextBox.Document.ContentStart,
            Input_Text_RichTextBox.Document.ContentEnd
        );
        string a = textRange.Text;
        string b = Input_Key_TextBox.Text;
        if(b.Length == 16)
        {
            string c = Crypt.EncryptString(b, a);
            Output_Result_TextBox.Text = c;
        }else
        {
            MessageBox.Show("Invalid Password!");
        }
        //string d = Crypt.DecryptString(b, c);
    }

    private void ReverseData_Button_Click(object sender, RoutedEventArgs e)
    {
        TextRange textRange = new TextRange(
            Input_Text_RichTextBox.Document.ContentStart,
            Input_Text_RichTextBox.Document.ContentEnd
        );
        textRange.Text = Output_Result_TextBox.Text;
        string a = textRange.Text;
        string b = Input_Key_TextBox.Text;
        if (b.Length == 16)
        {
            string c = Crypt.DecryptString(b, a);
            Output_Result_TextBox.Text = c;
        }
        else
        {
            MessageBox.Show("Invalid Password!");
        }
    }


    #region TestCode
    /*private void Button_Click(object sender, RoutedEventArgs e)
    {
        #region TestRegion

        /*using (var client = new WebClient())
        {
            client.Credentials = new NetworkCredential("jdoe", "test123");
            client.DownloadFile("http://192.168.0.103/admin/test1.txt", @"c:\tmp\" + "test1.txt");
        }

        testTXT.Text = File.ReadAllLines(@"c:\tmp\" + "test1.txt")[0];*/

    //debug SQLConnect
    //testTXT.Text = DownloadFileFromURL("jdoe", "test123", @"c:\tmp\", "test1.txt",
"http://192.168.0.103/admin/test1.txt");
    //SQLConnect();

    //SQLHandler sqlH = new SQLHandler("Select FileName from dbo.Files where id=1",
```

```
    //    Properties.Settings.Default.Database1ConnectionString);


    /*FileCompiler fileCompiler = new FileCompiler(Properties.Settings.Default.Database1ConnectionString);
    fileCompiler.CompileMethod("test1.txt", "SayHello");
    fileCompiler.InvokeMethod();
}

private void Button_Click1(object sender, RoutedEventArgs e)
{
    #region Crypt

    /*string a = "a \n bcsdf \n dsfsdf";
    string b = new StringBuilder().Insert(0, MotherBoardInfo.SerialNumber, 2).ToString();
    string c = Crypt.EncryptString(b, a);
    string d = Crypt.DecryptString(b, c);
    testTXT.Text = c;
    testTXT1.Text = d;

}*/

    #endregion

    #endregion

    #region Tab2

    private void GetQuery_Button_Click(object sender, RoutedEventArgs e)
    {
        SQLHandler sqlH = new SQLHandler("SELECT * FROM dbo.Files", connectionString);
        ObservableCollection<Files> custdata = sqlH.GetCollectionOfData();
        DB_DataGrid.ItemsSource = custdata;
        DB_DataGrid.DataContext = custdata;
    }

    private void ChangeAndGetQuery_Button_Click(object sender, RoutedEventArgs e)
    {
        string query = "";
        if(Select_TextBox.Text != "")
        {
            if(From_ComboBox.SelectedItem != null)
            {
                if(Where_CheckBox.IsChecked == true)
                {
                    if(Where_ComboBox.SelectedItem != null)
                    {
                        if(Where_TextBox.Text != "")
                        {
                            ComboBoxItem temp = From_ComboBox.SelectedItem as ComboBoxItem;
                            ComboBoxItem temp2 = Where_ComboBox.SelectedItem as ComboBoxItem;
                            query = "SELECT " + Select_TextBox.Text + " FROM " + temp.Content +
                                " WHERE " + temp2.Content + "='" + Where_TextBox.Text + "'";
                            //MessageBox.Show(query);
                            SQLHandler sqlH = new SQLHandler(query, connectionString);
                            DB_DataGrid.ItemsSource = null;
                            DB_DataGrid.Items.Clear();
                            DB_DataGrid.Items.Refresh();
                            DB_DataGrid.Columns.Clear();
```

```
string[] selectArr = Select_TextBox.Text.Split(',');
List<Files> files = new List<Files>();
//MessageBox.Show("1");
for (int i = 0; i < sqlH.GetStringList(0, sqlH.GetCountOfItems()).Count; i++)
{
   //MessageBox.Show("2");
   files.Add(new Files());
}
//MessageBox.Show("3");
DB_DataGrid.ItemsSource = files;
DB_DataGrid.DataContext = files;
DB_DataGrid.Columns.Clear();
int j = 0;
foreach (string a in selectArr)
{
   DataGridTextColumn textcol = new DataGridTextColumn();
   Binding b = new Binding(a);
   textcol.Binding = b;
   textcol.Header = a;
   DB_DataGrid.Columns.Add(textcol);
   List<string> tempList = new List<string>();
   int i = 0;
   switch (a)
   {
      case "Id":
         tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
         i = 0;
         foreach (Files file in files)
         {
            file.ID = Convert.ToInt32(tempList[i]);
            i++;
         }
         break;
      case "FileName":
         tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
         i = 0;
         foreach (Files file in files)
         {
            file.FileName = tempList[i];
            i++;
         }
         break;
      case "dll":
         tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
         i = 0;
         foreach (Files file in files)
         {
            file.dll = tempList[i];
            i++;
         }
         break;
      case "username":
         tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
         i = 0;
         foreach (Files file in files)
         {
            file.username = tempList[i];
            i++;
```

```
                }
                break;
            case "password":
                tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                i = 0;
                foreach (Files file in files)
                {
                    file.password = tempList[i];
                    i++;
                }
                break;
            case "url":
                tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                i = 0;
                foreach (Files file in files)
                {
                    file.url = tempList[i];
                    i++;
                }
                break;
            case "InterfaceClass":
                tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                i = 0;
                foreach (Files file in files)
                {
                    file.InterfaceClass = tempList[i];
                    i++;
                }
                break;
            case "methods":
                tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                i = 0;
                foreach (Files file in files)
                {
                    file.methods = tempList[i];
                    i++;
                }
                break;
        }
        DB_DataGrid.Items.Refresh();
        j++;
    }
  }
 }
}
else
{
  ComboBoxItem temp = From_ComboBox.SelectedItem as ComboBoxItem;
  query = "SELECT " + Select_TextBox.Text + " FROM " + temp.Content;
  SQLHandler sqlH = new SQLHandler(query, connectionString);
  DB_DataGrid.ItemsSource = null;
  DB_DataGrid.Items.Clear();
  DB_DataGrid.Items.Refresh();
  DB_DataGrid.Columns.Clear();
  string[] selectArr = Select_TextBox.Text.Split(',');
  List<Files> files = new List<Files>();
  for (int i = 0; i < sqlH.GetCountOfItems(); i++)
  {
```

```
    files.Add(new Files());
}
DB_DataGrid.ItemsSource = files;
DB_DataGrid.DataContext = files;
DB_DataGrid.Columns.Clear();
int j = 0;
foreach (string a in selectArr)
{
    DataGridTextColumn textcol = new DataGridTextColumn();
    Binding b = new Binding(a);
    textcol.Binding = b;
    textcol.Header  = a;
    DB_DataGrid.Columns.Add(textcol);
    List<string> tempList = new List<string>();
    int i = 0;
    switch (a)
    {
        case "Id":
            tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
            i = 0;
            foreach (Files file in files)
            {
                file.ID = Convert.ToInt32(tempList[i]);
                i++;
            }
            break;
        case "FileName":
            tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
            i = 0;
            foreach (Files file in files)
            {
                file.FileName = tempList[i];
                i++;
            }
            break;
        case "dll":
            tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
            i = 0;
            foreach (Files file in files)
            {
                file.dll = tempList[i];
                i++;
            }
            break;
        case "username":
            tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
            i = 0;
            foreach (Files file in files)
            {
                file.username = tempList[i];
                i++;
            }
            break;
        case "password":
            tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
            i = 0;
            foreach (Files file in files)
            {
```

```
                        file.password = tempList[i];
                        i++;
                    }
                    break;
                case "url":
                    tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                    i = 0;
                    foreach (Files file in files)
                    {
                        file.url = tempList[i];
                        i++;
                    }
                    break;
                case "InterfaceClass":
                    tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                    i = 0;
                    foreach (Files file in files)
                    {
                        file.InterfaceClass = tempList[i];
                        i++;
                    }
                    break;
                case "methods":
                    tempList = sqlH.GetStringList(j, sqlH.GetCountOfItems());
                    i = 0;
                    foreach (Files file in files)
                    {
                        file.methods = tempList[i];
                        i++;
                    }
                    break;
            }
            DB_DataGrid.Items.Refresh();
            j++;
        }
        //List<string> tempList = sqlH.GetStringList(0, sqlH.GetCountOfItems());
        //string[] ab = new string[1] { "test" };
        //string ab = "test";
        //DB_DataGrid.Items.Add(ab);
        /*DB_DataGrid.DataContext = new List<Files> { new Files()
        {
            FileName = "Test"
        } };*/
    }
  }
}

private void CheckBox_Checked(object sender, RoutedEventArgs e)
{
    Where_Label.IsEnabled = !Where_Label.IsEnabled;
    Where_TextBox.IsEnabled = !Where_TextBox.IsEnabled;
    Where_ComboBox.IsEnabled = !Where_ComboBox.IsEnabled;
}


#endregion
```

```
#region Tab3

private void MotherBoardInfo_Button_Click(object sender, RoutedEventArgs e)
{
  MotherBoardInfo_TextBox.Text = "Інформація материнської плати:\n1.Стан Материнської плати: " +
MotherBoardInfo.Availability
    + "\n2.Компанія: " + MotherBoardInfo.Manufacturer + "\n3.Первинний тип шини: " +
MotherBoardInfo.PrimaryBusType + "\n4.Номер продукту: "
    + MotherBoardInfo.Product + "\n5.Чи є можливість замінити: " + MotherBoardInfo.Replaceable +
"\n6.Серійний номер: " + MotherBoardInfo.SerialNumber;
}


#endregion

#region Tab4

private void FolderPath_Button_Click(object sender, RoutedEventArgs e)
{
  string fileName;

  using (var dialog = new System.Windows.Forms.OpenFileDialog())
  {
    System.Windows.Forms.DialogResult result = dialog.ShowDialog();
    fileName = dialog.FileName;
  }

  FileCompiler fileCompiler = new FileCompiler(connectionString);
  fileCompiler.DemonstrateCompileMethod(fileName);

  FileCompiler fileCompiler1 = new FileCompiler(Properties.Settings.Default.Database1ConnectionString);
  fileCompiler1.CompileMethod("test1.txt", "SayHello");
  fileCompiler1.InvokeMethod();
}

private void SaveFile_Button_Click(object sender, RoutedEventArgs e)
{
  string path = Path_TextBox.Text;
  TextRange textRange = new TextRange(
    MainText_TextBox.Document.ContentStart,
    MainText_TextBox.Document.ContentEnd
  );
  string text = textRange.Text;

  string fileName = FileName_TextBox.Text;
  using (StreamWriter outputFile = new StreamWriter(System.IO.Path.Combine(path, fileName)))
  {
    outputFile.WriteLine(text);
  }
}


#endregion

#region Tab5

private void Report_Button_Click(object sender, RoutedEventArgs e)
```

```
{
    TextRange report1 = new TextRange(
        Report1_RichTextBox.Document.ContentStart,
        Report1_RichTextBox.Document.ContentEnd
    );
    TextRange report2 = new TextRange(
        Report2_RichTextBox.Document.ContentStart,
        Report2_RichTextBox.Document.ContentEnd
    );
    TextRange report3 = new TextRange(
        Report3_RichTextBox.Document.ContentStart,
        Report3_RichTextBox.Document.ContentEnd
    );
    TextRange report4 = new TextRange(
        Report4_RichTextBox.Document.ContentStart,
        Report4_RichTextBox.Document.ContentEnd
    );
    TextRange report5 = new TextRange(
        Report5_RichTextBox.Document.ContentStart,
        Report5_RichTextBox.Document.ContentEnd
    );
    TextRange report6 = new TextRange(
        Report6_RichTextBox.Document.ContentStart,
        Report6_RichTextBox.Document.ContentEnd
    );

    string result = "";
    //int count = 0; 6 steps
    foreach(string a in Config.Report)
    {
        result += a + "\n";
        //MessageBox.Show(count.ToString());
        //count++;
    }
    report1.Text = Config.Report[0];
    MessageBox.Show("First step Confirmed!");
    report2.Text = Config.Report[1];
    MessageBox.Show("Second step Confirmed!");
    report3.Text = Config.Report[2];
    MessageBox.Show("Thrid step Confirmed!");
    report4.Text = Config.Report[3];
    MessageBox.Show("Fourth step Confirmed!");
    report5.Text = Config.Report[4];
    MessageBox.Show("Fifth step Confirmed!");
    report6.Text = Config.Report[5];
    MessageBox.Show("Sixth step Confirmed!");

    //MessageBox.Show(result);
}

#endregion

#endregion

#region Methods
```

```
        #region TestCode

        /*string DownloadFileFromURL(string username, string password, string path, string filename, string url )
        {
            using (var client = new WebClient())
            {
                client.Credentials = new NetworkCredential(username, password);
                client.DownloadFile(url, @path + filename);
            }
            //return File.ReadAllLines(@path + filename)[0];
            return String.Join("\n", File.ReadAllLines(@path + filename));
        }*/

        /*
        //username, password, path(don't forget about '@'), filename, url
        //void CompileFiles(string[] infoForDownload)
        void CompileFiles()
        {
            string source =
            @"
//using System;
//using System.Windows;
namespace Foo
{
    public class Bar
    {
        public void SayHello()
        {
            //System.Console.WriteLine(""Hello World"");
            System.Windows.MessageBox.Show(""Hello, you've done a great job!"");
            //MessageBox.Show(""Test"");
        }
    }
}
            ";

            //MessageBox.Show(""Test"");
            Dictionary<string, string> providerOptions = new Dictionary<string, string>
                {
                    {"CompilerVersion", "v3.5"}
                    //{"CompilerVersion", "v4.5"}
                };
            CSharpCodeProvider provider = new CSharpCodeProvider(providerOptions);

            CompilerParameters compilerParams = new CompilerParameters
            {
                GenerateInMemory  = true,
                GenerateExecutable = false
            };
            //compilerParams.ReferencedAssemblies.AddRange(dllFiles);
            //compilerParams.ReferencedAssemblies.Add("System.dll");
            compilerParams.ReferencedAssemblies.Add("WindowsBase.dll");
            compilerParams.ReferencedAssemblies.Add("PresentationFramework.dll");

            CompilerResults results = provider.CompileAssemblyFromSource(compilerParams, source);
```

```
        if (results.Errors.Count != 0)
          for (int i = 0; i < results.Errors.Count; i++)
          {
            MessageBox.Show(results.Errors[i].ToString());
          }


        object o = results.CompiledAssembly.CreateInstance("Foo.Bar");
        MethodInfo mi = o.GetType().GetMethod("SayHello");
        mi.Invoke(o, null);
      }*/

      /*void SQLConnect()
      {
        // Create the connection.
        using (SqlConnection connection = new
SqlConnection(Properties.Settings.Default.Database1ConnectionString))
        {
          // Create a SqlCommand, and identify it as a stored procedure.
          using (SqlCommand sqlCommand = new SqlCommand("Select FileName from dbo.Files where id=1",
connection))
          {
            //sqlCommand.CommandType = CommandType.StoredProcedure;

            // Add input parameter for the stored procedure and specify what to use as its value.
            //sqlCommand.Parameters.Add(new SqlParameter("@CustomerName", SqlDbType.NVarChar, 40));
            //sqlCommand.Parameters["@CustomerName"].Value = txtCustomerName.Text;

            // Add the output parameter.
            //sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
            //sqlCommand.Parameters["@CustomerID"].Direction = ParameterDirection.Output;

            try
            {
              connection.Open();
              MessageBox.Show("Connection open!");
              // Run the stored procedure.
              //sqlCommand.ExecuteNonQuery();
              SqlDataReader dataReader = sqlCommand.ExecuteReader();
              string output = "";
              while (dataReader.Read())
              {
                output = dataReader.GetValue(0).ToString();
              }
              MessageBox.Show(output);
              dataReader.Close();
              // Customer ID is an IDENTITY value from the database.
              //this.parsedCustomerID = (int)sqlCommand.Parameters["@CustomerID"].Value;

              // Put the Customer ID value into the read-only text box.
              //this.txtCustomerID.Text = Convert.ToString(parsedCustomerID);
              //string a = (string)sqlCommand.Parameters["@FileName"].Value;
              //testTXT1.Text = a;
            }
            catch
            {
              MessageBox.Show("Something went wrong");
            }
```

```
                finally
                {
                    sqlCommand.Dispose();
                    connection.Close();
                }
            }
        }
    }
    */

    #endregion

    #endregion


}
}
```

```
<Window x:Class="ProjectOne.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:ProjectOne"
mc:Ignorable="d"
Title="Демонстраційна робота дипломного завдання" Height="450" Width="800">
<Grid>
<TabControl HorizontalAlignment="Left" Height="419" VerticalAlignment="Top" Width="792">
<TabItem Header="Шифрування">
<Grid Background="#FFE5E5E5">
<Button Name="CipherStart_Button" Content="Шифрувати" HorizontalAlignment="Left"
Margin="227,280,0,0" VerticalAlignment="Top" Width="75" Height="41" Click="CipherStart_Button_Click"/>
<Label Content="Текст" HorizontalAlignment="Left" Margin="46,49,0,0" VerticalAlignment="Top"/>
<RichTextBox x:Name="Input_Text_RichTextBox" HorizontalAlignment="Left" Height="160"
Margin="113,49,0,0" VerticalAlignment="Top" Width="189">
<FlowDocument>
<Paragraph>
<Run Text="Введіть текст..."/>
</Paragraph>
</FlowDocument>
</RichTextBox>
<TextBox x:Name="Output_Result_TextBox" IsReadOnly="True" HorizontalAlignment="Left"
Margin="453,50,0,0" TextWrapping="Wrap" Text="Тут з’явиться результат шифрування" VerticalAlignment="Top"
Height="159" Width="203"/>
<TextBox x:Name="Input_Key_TextBox" HorizontalAlignment="Left" Height="35" Margin="113,227,0,0"
TextWrapping="Wrap" Text="Введіть ключ в 16 знаків" VerticalAlignment="Top" Width="189"/>
<Label Content="Ключ" HorizontalAlignment="Left" Margin="46,227,0,0" VerticalAlignment="Top"/>
<Label Content="Результат" HorizontalAlignment="Left" Margin="381,50,0,0"
VerticalAlignment="Top"/>
<Button x:Name="ReverseData_Button" Content="Розшифрувати" HorizontalAlignment="Left"
Margin="556,280,0,0" VerticalAlignment="Top" Width="100" Height="41" Click="ReverseData_Button_Click"/>
</Grid>
</TabItem>
<TabItem Header="SQL запити">
<Grid Background="#FFE5E5E5">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="10*"/>
<ColumnDefinition Width="121*"/>
```

```
</Grid.ColumnDefinitions>
<DataGrid x:Name="DB_DataGrid" ItemsSource="{Binding}" HorizontalAlignment="Left" Height="131"
Margin="10,10,0,0" VerticalAlignment="Top" Width="766" Grid.ColumnSpan="2"/>
<Button x:Name="GetQuery_Button" Content="Отримати данні" HorizontalAlignment="Left"
Margin="596,157,0,0" VerticalAlignment="Top" Width="120" Click="GetQuery_Button_Click" Height="33"
Grid.Column="1"/>
<CheckBox x:Name="Where_CheckBox" Content="Ввімкнуть WHERE умову" HorizontalAlignment="Left"
Margin="37,317,0,0" VerticalAlignment="Top" Checked="CheckBox_Checked" Grid.ColumnSpan="2"/>
<ComboBox Name="From_ComboBox" HorizontalAlignment="Left" Margin="376,213,0,0"
VerticalAlignment="Top" Width="120" Height="32" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" Grid.Column="1">
<ComboBoxItem>dbo.Files</ComboBoxItem>
</ComboBox>
<TextBox x:Name="Select_TextBox" HorizontalAlignment="Left" Height="32" Margin="104,213,0,0"
TextWrapping="Wrap" Text="*" VerticalAlignment="Top" Width="120" VerticalContentAlignment="Center"
Grid.Column="1" IsTabStop="False" HorizontalContentAlignment="Center"/>
<Button Name="ChangeAndGetQuery_Button" Content="Оновити запит" HorizontalAlignment="Left"
Margin="596,302,0,0" VerticalAlignment="Top" Width="120" Height="30"
Click="ChangeAndGetQuery_Button_Click" Grid.Column="1"/>
<Label Content="SELECT" VerticalContentAlignment="Center" HorizontalAlignment="Left"
Margin="37,213,0,0" VerticalAlignment="Top" Background="White" BorderBrush="Black" BorderThickness="0.8"
Width="98" Height="32" HorizontalContentAlignment="Center" Grid.ColumnSpan="2">
</Label>
<Label Content="FROM" VerticalContentAlignment="Center" HorizontalAlignment="Left"
Margin="247,213,0,0" VerticalAlignment="Top" Background="White" BorderBrush="Black" BorderThickness="0.8"
Width="98" Height="32" HorizontalContentAlignment="Center" Grid.Column="1"/>
<ComboBox x:Name="Where_ComboBox" HorizontalAlignment="Left" Margin="104,265,0,0"
VerticalAlignment="Top" Width="120" Height="32" HorizontalContentAlignment="Center"
VerticalContentAlignment="Center" IsEnabled="False" Grid.Column="1">
<ComboBoxItem>Id</ComboBoxItem>
<ComboBoxItem>FileName</ComboBoxItem>
<ComboBoxItem>dll</ComboBoxItem>
<ComboBoxItem>username</ComboBoxItem>
<ComboBoxItem>password</ComboBoxItem>
<ComboBoxItem>url</ComboBoxItem>
<ComboBoxItem>InterfaceClass</ComboBoxItem>
<ComboBoxItem>methods</ComboBoxItem>
</ComboBox>
<TextBox x:Name="Where_TextBox" HorizontalAlignment="Left" Height="32" Margin="301,265,0,0"
TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="120" VerticalContentAlignment="Center"
IsEnabled="False" Grid.Column="1" HorizontalContentAlignment="Center"/>
<Label Name="Where_Label" Content="WHERE" VerticalContentAlignment="Center"
HorizontalAlignment="Left" Margin="37,265,0,0" VerticalAlignment="Top" Background="White"
BorderBrush="Black" BorderThickness="0.8" Width="98" Height="32" HorizontalContentAlignment="Center"
IsEnabled="False" Grid.ColumnSpan="2"/>
<Label Content="=" HorizontalAlignment="Left" Margin="244,265,0,0" Background="White"
BorderBrush="Black" BorderThickness="0.8" Width="40" Height="32" FontSize="26" UseLayoutRounding="True"
ScrollViewer.CanContentScroll="True" ScrollViewer.VerticalScrollBarVisibility="Disabled" Padding="0"
VerticalAlignment="Top" HorizontalContentAlignment="Center" VerticalContentAlignment="Stretch"
Grid.Column="1"/>
</Grid>
</TabItem>
<TabItem Header="Інформація материньскої плати">
<Grid Background="#FFE5E5E5">
<Button x:Name="MotherBoardInfo_Button" Content="Вивести інформацію"
HorizontalAlignment="Left" Margin="502,10,0,0" VerticalAlignment="Top" Width="128" Height="42"
Click="MotherBoardInfo_Button_Click"/>
```

```
<TextBox x:Name="MotherBoardInfo_TextBox" HorizontalAlignment="Left" Height="371"
Margin="10,10,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="468" IsEnabled="False"
FontSize="24"/>
</Grid>
</TabItem>
<TabItem Header="Компіляція файла">
<Grid Background="#FFE5E5E5">
<RichTextBox x:Name="MainText_TextBox" HorizontalAlignment="Left" Height="371"
Margin="10,10,0,0" VerticalAlignment="Top" Width="417">
<FlowDocument>
<Paragraph>
<Run Text="Введіть код"/>
</Paragraph>
</FlowDocument>
</RichTextBox>
<TextBox x:Name="Path_TextBox" HorizontalAlignment="Left" Height="28" Margin="443,10,0,0"
TextWrapping="Wrap" Text="Шлях" VerticalAlignment="Top" Width="110"/>
<Button x:Name="FolderPath_Button" Content="Скомпілювати файл" HorizontalAlignment="Left"
Margin="443,204,0,0" VerticalAlignment="Top" Width="111" Height="48" Click="FolderPath_Button_Click"/>
<Button x:Name="SaveFile_Button" Content="Зберегти" HorizontalAlignment="Left"
Margin="442,93,0,0" VerticalAlignment="Top" Width="111" Height="48" Click="SaveFile_Button_Click"/>
<TextBox x:Name="FileName_TextBox" HorizontalAlignment="Left" Height="28" Margin="443,43,0,0"
TextWrapping="Wrap" Text="Назва файлу" VerticalAlignment="Top" Width="110"/>
</Grid>
</TabItem>
<TabItem Header="Кінцева робота всіх елементів">
<Grid Background="#FFE5E5E5">
<Button x:Name="Report_Button" Content="Button" HorizontalAlignment="Left" Margin="632,311,0,0"
VerticalAlignment="Top" Width="132" Height="55" Click="Report_Button_Click"/>
<RichTextBox x:Name="Report1_RichTextBox" HorizontalAlignment="Left" Height="155"
Margin="10,10,0,0" VerticalAlignment="Top" Width="193">
<FlowDocument>
<Paragraph>
<Run Text=""/>
</Paragraph>
</FlowDocument>
</RichTextBox>
<RichTextBox x:Name="Report2_RichTextBox" HorizontalAlignment="Left" Height="356"
Margin="220,10,0,0" VerticalAlignment="Top" Width="356">
<FlowDocument>
<Paragraph>
<Run Text=""/>
</Paragraph>
</FlowDocument>
</RichTextBox>
<RichTextBox x:Name="Report3_RichTextBox" HorizontalAlignment="Left" Height="61"
Margin="10,180,0,0" VerticalAlignment="Top" Width="193">
<FlowDocument>
<Paragraph>
<Run Text=""/>
</Paragraph>
</FlowDocument>
</RichTextBox>
<RichTextBox x:Name="Report4_RichTextBox" HorizontalAlignment="Left" Height="42"
Margin="595,10,0,0" VerticalAlignment="Top" Width="181">
<FlowDocument>
<Paragraph>
<Run Text=""/>
```

```
                    </Paragraph>
                </FlowDocument>
                </RichTextBox>
<RichTextBox x:Name="Report5_RichTextBox"  HorizontalAlignment="Left"  Height="37"
        Margin="595,66,0,0" VerticalAlignment="Top" Width="181">
                <FlowDocument>
                    <Paragraph>
                     <Run Text=""/>
                    </Paragraph>
                </FlowDocument>
                </RichTextBox>
<RichTextBox x:Name="Report6_RichTextBox"  HorizontalAlignment="Left"  Height="33"
        Margin="595,119,0,0" VerticalAlignment="Top" Width="181">
                <FlowDocument>
                    <Paragraph>
                     <Run Text=""/>
                    </Paragraph>
                </FlowDocument>
                </RichTextBox>
                    </Grid>
                </TabItem>
            </TabControl>

            </Grid>
        </Window>
```