МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КІБЕРБЕЗПЕКИ

ВИПУСКНА РОБОТА

на тему:

«Захист програмних продуктів від несанкціонованого копіювання та поширення»

Завідувач
випускаючої кафедри **Любчак В. О.**

Керівник роботи **Лаврик Т. В.**

Студента групи КБ – 81 **Сущок І.О.**

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра кібербезпеки

Затверджую_____
Зав. кафедрою Любчак В.О.
“_____” _____2022 р.

ЗАВДАННЯ

Завдання до бакалаврської роботи

Студента четвертого курсу, групи КБ-81 спеціальності «Кібербезпека» денної форми навчання Сущка Івана Олександровича.

Тема: «Захист програмних продуктів від несанкціонованого копіювання та поширення»

Затверджена наказом по СумДУ
№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналітичний огляд ліцензійного та неліцензійного програмного забезпечення, загроз від використання неліцензійного програмного забезпечення; 2) огляд можливих засобів захисту програмного забезпечення; 3) проєктування та опис моделі захисту програмного забезпечення; 4) вибір програмних засобів та опис програмної реалізації додатку; 5) тестування реалізованого додатку.

Дата видачі завдання “_____” _____ 2022г.
Керівник бакалаврської роботи _____ Лаврик Т.В.
Завдання приняла до виконання _____ Сущок І.О.

РЕФЕРАТ

Записка: 43 стор., 11 рис., 0 табл., 1 додаток, 28 джерел.

Об'єкт дослідження – захист програмного забезпечення від несанкціонованого копіювання та поширення.

Мета роботи – розроблення програмного рішення, що дозволить автоматизувати процес захисту функціонуючих додатків від несанкціонованого доступу, копіювання та поширення.

Метод дослідження — метод аналітичного огляду, метод порівняння, метод моделювання, методи розробки, що базуються на мові програмування C#.

Результати роботи – проаналізовано характерні особливості ліцензійного та неліцензійного програмного забезпечення, а також можливі загрози від використання неліцензійного програмного забезпечення; проведено огляд можливих засобів захисту програмного забезпечення; розроблено модель захисту програмного забезпечення від несанкціонованого копіювання та поширення; здійснено програмну реалізацію додатку, що забезпечуватиме відповідний захист програмного забезпечення.

ЗАХИСТ КОДУ, ОБФУСКАЦІЯ, ЛІЦЕНЗІЙНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЛІЦЕНЗІЙНИЙ КЛЮЧ, ЗАХИСТ ВІД ПІРАТСТВА, НЕСАНКЦІОНОВАНЕ ВИКОРИСТАННЯ, НЕСАНКЦІОНОВАНЕ КОПІЮВАННЯ, НЕСАНКЦІОНОВАНЕ ПОШИРЕННЯ

ЗМІСТ

ВСТУП

Використання комп'ютера неминуче пов'язано з роботою з деяким програмним забезпеченням, таким як, операційні системи, офісні додатки, інші необхідні для роботи додатки, а також ігрові програми. Тож користувачі комп'ютерів поділяються на декілька категорій, серед яких є ті, що використовують програмний продукт, який знаходиться у вільному доступі, або який вони свідомо придбали. Але зустрічаються й ті, які користуються піратським програмним забезпеченням, зламаними версіями ліцензійних додатків. Таким чином, розробники оригінальних додатків зазнають збитків, порушується їх право на інтелектуальну власність, і як наслідок, робота, на яку було витрачено багато часу залишається не оплаченою належним чином. Це може також завдати збитків і по репутації розробників, адже у деякому піратському програмному забезпеченні (ПЗ) може бути вбудоване шкідливе програмне забезпечення. У такому випадку користувач, у першу чергу, буде звинувачувати розробника, а не тих хто зламав це ПЗ, і вбудував туди ці віруси.

Нажаль переважна кількість користувачів, а також малий та середній бізнес, обирають неліцензійне програмне забезпечення заради економії коштів, і як наслідок, це може призводити до ризику втрати деяких конфіденційних даних, особистих даних користувачів та даних банківської сфери. Таким чином, спроби зекономити і не витрачати кошти на придбання ліцензійного ПЗ, в решті решт призводить до виникнення ризику для безпеки, який може призвести до значних фінансових втрат, що обійдуться власникам набагато дорожче, ніж передбачалося [1].

Згідно з дослідженням BSA Global Software Survey, яке випустила Асоціація розробників програмного забезпечення, станом на 2018 рік, в Україні приблизно 80% встановленого на комп'ютерах ПЗ не має необхідної ліцензії. У цьому дослідженні також наводяться дані щодо обсягу та вартості неліцензійного ПЗ, встановленого на персональні комп'ютери. Так, за

даними дослідження, в Україні вартість всього ПЗ, встановленого без ліцензії за 2017 рік становить $108 млн.

Також BSA відзначають, що використання неліцензійного ПЗ сприяє зростанню кібератак та пов'язано з фінансовими втратами. Неліцензійне ПЗ, заражене вірусами, задає збитків компаніям в усьому світі у розмірі $359 млрд. за рік [2].

Отже, виходячи з описаних проблем, набуває актуальності питання захисту програмного забезпечення від несанкціонованих розробниками дій з боку користувачів та потенційних хакерів.

Метою даної випускної роботи є розробка програми, за допомогою якої розробники ПЗ зможуть зберегти інтелектуальну власність на свої програмні продукти. Принцип роботи програми полягає у вбудовуванні перевірки ліцензії у готові додатки, що зможе забезпечити захист від несанкціонованого використання. Також програма застосовує обфускацію програмного коду додатка зі збереженням його функціоналу з метою приведення його до стану ускладнюючого аналіз, розуміння принципів його роботи та модифікації при можливій декомпіляції. Таким чином забезпечується захист від несанкціонованого поширення та копіювання, ускладнюючи роботу для потенційних зловмисників.

РОЗДІЛ 1. АНАЛІЗ ЛІЦЕНЗІЙНОГО ТА НЕЛІЦЕНЗІЙНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕНННЯ

1.1. Програмне забезпечення: загальна характеристика та класифікація

Програмне забезпечення (ПЗ) – загальне поняття, що вказує на набір певних інструкцій, у вигляді програмного код, який повідомляє комп'ютеру як і що саме необхідно зробити для виконання поставленої задачі. Він не залежить від апаратного забезпечення і робить комп'ютери програмованими.

Прикладом програмного забезпечення є будь-яка програма або код, що працює на комп'ютері. Це можуть бути комерційні програми, ігри, операційні системи або навіть шкідливе програмне забезпечення, таке як віруси, програми шпигуни тощо [3].

На сьогоднішній день існує велика кількість типів програмного забезпечення, серед яких виділяють декілька основних.

Системне програмне забезпечення – це загальна категорія програмного забезпечення, яка слугує базовою платформою для запуску додатків і дозволяє функціонувати апаратному забезпеченню. Виділяють також декілька рівнів системного ПЗ:

- Операційна система (ОС) – це комплекс взаємопов'язаних програм, призначених для управління ресурсами комп'ютера та організації взаємодії з користувачем. Без ОС комп'ютер являє собою набір апаратних компонентів, які самостійно не здатні виконувати які-небудь дії. Використання ж ОС дозволяє виконувати основні функції, надає графічний інтерфейс, який дозволяє користувачу взаємодіяти з комп'ютером, та надає платформу, на якій можуть виконуватися програмні додатки. У більшості обчислювальних систем ОС є основою, найважливішою, а іноді й єдиною частиною системного програмного забезпечення [4].
- Прошивка (вбудоване програмне забезпечення) – це вміст енергозалежної пам'яті будь-якого обчислювального пристрою. За допомогою прошивки пристрій, на який вона встановлена, отримує інструкції

як себе поводити та як взаємодіяти з іншими пристроями. Прошивка є напівпостійним ПЗ, вона може оновлюватися та зберігатися коли на пристрій не подається живлення [5].

- Драйвер – це комп'ютерне програмне забезпечення, за допомогою якого операційна система отримує доступ до апаратного забезпечення деякого пристрою. Зазвичай операційні системи містять у собі драйвери для ключових компонентів апаратного забезпечення, без яких система не зможе працювати. Проте для деяких пристроїв необхідні спеціальні драйвера, які надає виробник. Майже кожен компонент комп'ютера включаючи відеокарту, клавіатуру та мишу, має свої власні драйвери [6].
- Утиліта – це службова допоміжна комп'ютерна програма у складі загального програмного забезпечення для виконання спеціалізованих типових задач, пов'язаних з роботою обладнання та операційної системи. Зазвичай це невеликі програми, які часто поставляються разом з ОС. Прикладами утиліт є програмне забезпечення для очищення жорсткого диску, інструменти архівації файлів, антивірусне ПЗ тощо. Загалом утилітами вважають програми, які допомагають зробити комп'ютер краще [7].

Прикладне програмне забезпечення (програма або додаток) – це програма, призначена для виконання певних задач та розрахована на безпосередню взаємодію з користувачем. До програмних додатків належать комп'ютерні програми, написані для користувачів для задання комп'ютеру певної роботи [8].

Існує велике різноманіття програмних додатків. Деякі з найбільш поширених використовуються для роботи, наприклад, для організації таблиць, як текстові редактори, для управління даними. Прикладом є пакет Microsoft Office. До них також відносяться мультимедійні додатки, ігри та розважальні програми, а також веб-браузери.

Програмне забезпечення для програмування. Розробкою програмного забезпечення переважно займаються програмісти, інженери-програмісти та розробники програмного забезпечення. І для такої розробки вони також користуються деякими інструментами:

- Компілятори – це програми, які перетворюють код, написаний фахівцями на високорівневих мовах програмування, у низкорівневу форму машинного коду, яка може інтерпретуватися комп'ютерним обладнанням. Компілятори дозволяють розробникам писати більш складні програми на обраній мові програмування.
- Відлагоджувач – це комп'ютерна програма, яка використовується для тестування та відлагодження програмного коду. З його допомогою розробники здійснюють пошук та видалення помилок у своїх додатках [9].
- Компонувальник (редактор зв'язків, лінкер) – це програма, яка бере вихідні дані компілятора, зазвичай це декілька окремих файлів, та об'єднує їх у один виконуваний файл, який користувач може запускати окремо, без необхідності запуску у середовищі програмування [10].
- Шкідливе ПЗ – це програмне забезпечення, що перешкоджає коректній роботі комп'ютера, збирає конфіденційну інформацію користувачів або виконує інші небажані дії у комп'ютерній системі. На сьогоднішній день до такого програмного забезпечення відносяться: віруси (комп'ютерна програма здатна до самопоширення і здатна завдавати певної шкоди ОС), комп'ютерні хробаки (програми, які відтворюють самі себе, задля розповсюдження на інші комп'ютери), трояни (шкідлива програма, яка вводить користувачів в оману відносно своїх істинних мотивів) та програми-шпигуни (програма, ціль якої збір інформації про користувача та організація її відправки іншій особі). Для протистояння шкідливому ПЗ були розроблені антивірусні та антишпигунські утиліти , які виявляють та видаляють загрози з комп'ютера [11, 12, 13, 14, 15].

Для використання програмного забезпечення його спершу необхідно деяким чином отримати. Різні програми, можуть також і по різному розповсюджуватися. В залежності від методики поширення, обраної розробниками, виділяються різні типи програмного забезпечення.

Комерційне програмне забезпечення – це програмне забезпечення, розроблене з метою продажів його копій або у комерційних цілях для отримання прибутку. Комерційною програма вважається програма, якщо вона була створена під час деякої підприємницької діяльності. Такі програми може придбати користувач і отримати її фізичну або цифрову копію. Проте у такому випадку користувач не стає власником отриманого програмного забезпечення, оскільки це залишається інтелектуальною власністю розробника і підлягає під захист авторських прав. Процес придбання ПЗ означає те, що користувач отримає від розробника ліцензію, що надає права на використання продукту [16].

Саме на захист комерційного ПЗ і націлена дана випускна робота, задля забезпечення збереження за автором його прав на інтелектуальної власності і обмеження дій від користувачів, які сприяють їх порушенню.

Програмне забезпечення з відкритим початковим кодом – це комп'ютерне програмне забезпечення, що було випущено за ліцензією, у якій правовласник надає користувачам усі права на ПЗ. Користувач може вільно використовувати таку програму, вносити свої зміни у код та розповсюджувати її кому завгодно і для будь-яких цілей. Саме завдяки такому підходу, коли користувачі можуть самі вільно дослідити код програми, якою вони користуються, зростає довіра суспільності до такого ПЗ [17].

Безкоштовне програмне забезпечення – це програми, які скоріше за все є забезпеченням із закритим програмним кодом, як у прикладі комерційного ПЗ, але поширюються безкоштовно для кінцевих користувачів. Права на такі програми можуть суттєво відрізнятися між собою. Кожна окрема організація або безпосередньо автори-розробники можуть розробити власні ліцензійні

угоди для такого типу поширення ПЗ. Наприклад, деякі постачальники можуть дозволяти проводити модифікації та поширення ПЗ, у той час як інші надають права лише на безоплатне використання. Переважна більшість безкоштовного програмного забезпечення не надає доступ користувачам до початкового коду, на відміну від відкритого ПЗ.

Модель безкоштовного ПЗ дозволяє розробникам набагато легше поширювати свої програми серед користувачів, оскільки надається можливість використання деякого функціоналу додатків абсолютно безкоштовно.

Розробники ПЗ можуть звернутися до такого типу поширення своєї продукції задля швидкого охоплення цільової аудиторії або самореклами шляхом надання програм з обмеженим функціоналом. У такому випадку для отримання повного спектру послуг користувачам буде надана можливість придбати більш повну комерційну версію. Також у деяке безкоштовне ПЗ можуть вбудовувати рекламні банери з метою отримання деякого доходу його авторами. Таке ПЗ інколи називається рекламним [18]

Умовно безкоштовне програмне забезпечення – це тип закритого ПЗ із закритим початковим кодом, що надається користувачам для пробного використання. Таке ПЗ частіше за все постачається безкоштовно або за відносно невелику оплату. Воно також має обмежений функціонал або обмежений термін безкоштовного використання [19].

При використанні такого типу ПЗ розробники надають користувачам можливість ознайомитись та протестувати функціонал своїх додатків, і якщо в них виникає бажання, надається можливість придбання повної версії без обмежень. Інакше кажучи, умовно безкоштовне ПЗ це версія комерційного ПЗ з деякими обмеженнями.

1.2. Неліцензійне програмне забезпечення та загрози його використання

На сьогоднішній день одним із популярних способів розповсюдження комерційного програмного забезпечення є піратство.

Програмне піратство – це тип інтернет-піратства, що полягає у несанкціонованому та незаконному копіюванні програмного забезпечення, захищеного авторським правом.

Зазвичай розповсюджуються саме платні комерційні програми задля обходу процесу придбання за кошти. Оригінальні програми часто можуть модифікуватись, з них може видалятися частина перевірки ліцензії. Зловмисники також можуть і залишати щось своє у таких додатках, і скоріше за все це буде якесь приховане шкідливе ПЗ, майнери тощо [20].

Програмне піратство поділяється на чотири основні напрями:

- створення копій для резервного копіювання;
- створення копії та передача її іншим особам безкоштовно;
- перепродаж або надання оренди оригінально програмного забезпечення;
- створення копії з подальшим перепродажем.

Створення та розповсюдження піратського програмного забезпечення є незаконним у більшості країн. Піратство порушує авторські права розробників комерційного ПЗ і позбавляє їх потенційного доходу від продажу оригінальних копій додатків.

Згідно з діючим законодавством України передбачено три види відповідальності за використання неліцензійного програмного забезпечення:

- Адміністративна відповідальність

За незаконне використання, привласнення авторства або інше умисне порушення прав на об’єкт права інтелектуальної власності, під яке підпадає право на програмне забезпечення, передбачена відповідальність у вигляді штрафу від десяти до двохсот неоподаткованих мінімумів [21].

- Кримінальна відповідальність

Незаконне відтворення та розповсюдження комп’ютерних програм, або інше умисне порушення авторського права та суміжних прав, або фінансування таких дій, якщо це завдало значної матеріальної шкоди, -

караються штрафом у розмірі від двохсот до тисячі неоподаткованих мінімумів, або виправні роботи чи позбавлення волі на строк до двох років [22].

- Цивільна відповідальність

Дії, що можуть привести до порушення авторських прав на інтелектуальну власність, це опублікування, відтворення, ввезення та вивезення на митну територію без відома правовласника, свідомий обхід технічних засобів захисту, підроблення або зміна інформації. У судовому порядку правовласники можуть вимагати відшкодування моральної шкоди, збитків, стягнення з порушників доходу [23].

Отже, до відповідальності за використання піратського програмного забезпечення можуть притягнутися не лише особи, що його створюють або поширюють, а й ті користувачі, які свідомо отримують та інсталюють такі додатки для власного використання.

Проте не тільки з боку законодавства можуть виникати ризики від піратського ПЗ. Використання такого програмного забезпечення збільшує ризики атаки шкідливим ПЗ. З огляду на те, що до піратства для економії коштів можуть вдаватися не лише користувачі, але й малий та середній бізнес, ризики для них стають набагато вагомішими.

Найпоширенішим ризиком, який може виникнути, є крадіжка даних з пристроїв користувачів, що встановили такі програми. Таким чином, можуть бути викриті дані щодо банківських карт, особистої або конфіденційної інформації та комерційних таємниць. Втрата такого роду інформації може призвести до значних збитків.

Також у піратське ПЗ можуть бути інстальовані майнери, які використовують ресурси пристроїв для отримання піратами криптовалюти. Цей процес може відбуватися автоматично і без відома користувачів. Майнери змушують робити пристрої жертв у режимі максимальної продуктивності для виконання своїх цілей. У такому режимі запуск

додаткових потоків задач, що ініціюють користувачі, може суттєво гальмувати систему. Також тривала робота на межі своїх можливостей рано чи пізно призводить до виходу з ладу фізичних компонентів комп'ютера і виведення його з ладу. Найчастіше пошкоджуються такі комплектуючі, як відеокарта, процесор, оперативна пам'ять та система охолодження. Ці компоненти можуть коштувати чимало і це може завдати збитків для бюджету користувачів.

Ще один вид шкідливого ПЗ, що може інсталюватись разом з піратськими додатками, це програми-вимагачі. Такий вид шкідливого ПЗ блокує пристрій або шифрує файли на ньому. Шахраї вимагають від своїх жертв грошових переказів за розблокування доступу до своїх девайсів, проте в деяких випадках навіть це не гарантує того, що файли вдасться повернути. Якщо така програма потрапить у корпоративну мережу, це може призвести до втрати комерційних даних, що може викликати значні фінансові витрати.

З огляду на усі наведені приклади ризиків та загроз, можна зробити висновок, що використання піратського ПЗ призводить до значного збільшення шансів стати жертвою атаки з боку шкідливого ПЗ, що майже в будь-якому випадку призводить до збитків, які перевищують вартість придбання ліцензійного ПЗ. Отже, використання перевіреного оригінального програмного забезпечення набагато безпечніше і в кінцевому випадку обійдеться дешевше.

Проте загрози виникають не лише для користувачів піратського ПЗ. Збитків також зазнають й компанії-розробники. Вони розробляють своє програмне забезпечення з метою отримання прибутку від продажу ліцензійних копій. При розповсюдженні їхніх програмних продуктів нелегально і безкоштовно отримати повну очікувану суму прибутку компанії не вдасться. Для малих і нових компаній, що тільки виходять на ринок програмного забезпечення, така ситуація може призвести до недоотримання фінансування і врешті-решт – до банкрутства.

1.3. Постановка задачі

Аналізуючи усі загрози та ризики, що несуть за собою використання піратських програм, стає очевидним, що необхідно звести до мінімуму можливості їх виникнення. Цього можна досягти шляхом реалізації відповідних засобів захисту, при удосконаленні підходів до їх розробки. Більш масштабні та відомі компанії мають достатньо фінансів та можливостей для реалізації деяких власних прийомів для зменшення вірогідності зламу своїх програмних продуктів. Проте, невеличкі компанії або навіть самостійні розробники не завжди можуть дозволити собі такі витрати. Саме для вирішення такої проблеми планується розробити додаток, який зможе забезпечити реалізацію захисту для вже готового програмного продукту. Додаток має в автоматичному режимі вбудовувати перевірку та підтвердження ліцензії, яку має надавати розробник після підтвердження користувачем придбання копії додатку. Також проводиться обфускація програмного коду з метою ускладнення його аналізу та розуміння принципів роботи для забезпечення захисту від копіювання та подальшого поширення.

Метою випускної роботи є розроблення програмного додатку, який зможе допомогти розробникам комерційного програмного забезпечення захиститися від незаконних дій над об’єктом своєї інтелектуальної власності. Реалізовані засоби мають забезпечити захист від несанкціонованого використання, копіювання та поширення програм, розроблених у комерційних цілях.

Розроблений програмний додаток має забезпечити неуступний функціонал:

1) представлення зрозумілого та зручного інтерфейсу;

2) реалізація можливості налаштування деяких параметрів засобів захисту зі сторони розробника;

3) автоматичне вбудовування підтвердження та перевірки ліцензійного ключа програми;

4) застосування алгоритмів обфускації для захисту обраного додатку.

РОЗДІЛ 2. ХАРАКТЕРИСТИКА ЗАСОБІВ ЗАХИСТУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Загальна характеристика систем захисту

Для захисту програмного забезпечення необхідно реалізовувати деякий комплекс заходів. Для комерційного ПЗ, доцільно організувати захист від несанкціонованого використання, розповсюдження та модифікації програм.

Системи захисту ПЗ поділяються на декілька основних напрямів за методами встановлення, механізмами захисту та принципами функціонування.

За методами встановлення виділяють наступні типи систем захисту:

- Встановлювані на скомпільовані модулі ПЗ. Вони дозволяють легко захистити вже готовий програмний продукт. Однак стійкість до злому таких систем невелика. Щоб запустити додаток без такого захисту, необхідно лише визначити точку завершення роботи елементів захисту і передачі управління захищеному додатку, після чого примусово зберегти її у незахищеному вигляді.
- Системи, що вбудовуються у первинний код. Такі системи не дуже зручні з боку розробника, оскільки виникає необхідність витрачати час та кошти для навчання персоналу роботі з програмним інтерфейсом систем захисту. Також ускладнюється і процес тестування ПЗ, оскільки при вбудовуванні безпосередньо у код можуть виникати проблеми його роботи, пов’язані з системою захисту або механізмами, що вони використовують. Проте такі системи більш стійкі до атак ніж попередні, оскільки визначити точку завершення роботи системи захисту стає набагато складніше.
- Комбіновані системи. Використання обох попередніх систем у комбінованому вигляді у значній мірі підвищує захист ПЗ. Проаналізувати та деактивувати такі захисні модулі стає максимально важко.

За механізмами захисту, системи поділяються на:

- Системи, що використовують складні логічні механізми. Вони використовують різноманітні методи та прийоми, спрямовані на ускладнення аналізу програмного коду та самих алгоритмів захисту. Однак, якщо алгоритм роботи такої системи буде викрито, модифікування процедур перевірки може призвести до обходу захисту.
- Системи, що використовують криптографічні алгоритми. Вони є більш стійкими в порівнянні з попередніми. Для обходу такого захисту, зловмисникам необхідно визначити ключ для дешифрування ПЗ. Для підбору такого ключа, можуть знадобитися значні обчислювальні потужності, та час. Навіть якщо підбір ключа вдасться через деякий час, інформація яка їм зашифрована може вже втратити свою актуальність.
- Комбінована система, що включає в себе дві попередні системи, що дозволяє досягти більш високого рівня захисту [24].

Для реалізації захисту ПЗ використовуються різноманітні методи та алгоритми, серед яких виділяють кілько основних:

1. Алгоритми заплутування. Вони використовують спотворення реальної кількості параметрів вбудовування методів та процедур, які насправді нічого не виконують, використовуються хаотичні виклики та переходи у різні частини коду тощо.
2. Алгоритми мутації. Принцип роботи полягає у створенні таблиці операндів – синонімів, і їхня заміна відповідно до визначеної схеми, наприклад, при кожному старті програми, або у випадковому порядку.
3. Алгоритми компресії даних. Програма, до якої використано цей алгоритм, запаковується, після чого розпаковується по мірі виконання.
4. Алгоритми шифрування даних. Програма шифрується, а потім дешифрується по мірі виконання. Можлива повне або часткове дешифрування.

5. Методи ускладнення дизасемблювання. Використовуються різноманітні прийому, що ускладнюють перетворення машинного коду у текст програми на мові асемблер.

6. Емуляція процесорів та операційних систем. При використанні такої системи створюється віртуальний процесор або операційна система та програма перекладач із системи команд реального пристрою у систему команд згенерованого процесора або ОС. Це сильно ускладнює процес дослідження алгоритмів роботи ПЗ, оскільки програма може виконуватися лише за допомогою емулятора.

Нестандартні методи роботи з апаратним забезпеченням. Модулі системи захисту звертаються до апаратного забезпечення пристрою в обхід процедури операційної системи, використовуючи маловідомі або недокументовані його властивості [25].

2.2. Основні способи захисту програмного забезпечення

Захист своїх авторських прав та розроблених додатків від піратства є головною метою розробників комерційного програмного забезпечення. Для цього вони можуть використовувати деякі із запропонованих способів захисту або їх комбінувати для посилення рівня захищеності.

З основних напрямів реалізації захисту виділяються наступні:

- Авторські права, патенти угоди з кінцевим користувачем

Використовуючи такий спосіб захисту, створюється правова основа для бізнесу з метою захисту своєї інтелектуальної власності. Угода з кінцевим користувачем визначає межі, у яких вони повинні використовувати свою копію програми. Процес придбання такої копії не означає, що користувач стає власником програми, і тому він повинен підпорядковуватися угоді, у якій прописані допустимі дії. У більшості угод прописані обмеження щодо незаконного копіювання продукту, розповсюдження безкоштовно тощо.

Проте такі угоди часто порушуються користувачами, саме тому були розроблені інші способи захисту.

- Використання підтвердження ключів програмної копії продукту

Ключ програмного продукту – це спеціально розроблений унікальний ключ, необхідний для підтвердження ліцензії цієї програми. Володіння таким ключем підтверджує те, що програма була придбана без порушення авторських прав та з джерел, що мають право ні її розповсюдження. Такий ключ використовується для процедури активації продукту. Під час цього процесу перевіряється наявність відповідного ліцензійного ключа, що вводить користувач з базою даних розробників, що дозволяє перевірити його дійсність. Програмне забезпечення не зможе запуститися без підтвердження ліцензійного ключа. Використання такої схеми забезпечує зниження рівня піратства серед кінцевих користувачів. Використання прив'язки ключа до апаратних засобів пристрою є одним з найнадійніших типів захисту, оскільки він прив'язується до індивідуальних параметрів комп'ютера. Зазвичай використовуються серійний номер процесора, жорсткого диску, версії BIOS тощо.

- Обфускація початкового коду програми

Це спосіб заплутування коду з методу ускладнення аналізу алгоритмів його роботи. Можуть використовуватися складні алгоритми, що дозволяють змінити програмний код, додати нових функцій з неоднозначним функціоналом, але не впливаючи на роботу програми, щоб заплутати код зробити його хаотичним та нечитабельним. Однак такий алгоритм не забезпечує повний захист, адже зловмисники при достатній кількості часу можуть все ж таки розібрати як працює код та відтворити копію. Саме тому обфускацію часто використовують у поєднанні з іншими способами захисту. Зазвичай її поєднують із захистом від несанкціонованого доступу.

- Програмне забезпечення з захистом від несанкціонованого доступу

Такий тип захисту призначений для запобігання модифікацій та зворотного проектування. При виявленні зміни початкового коду програма почне працювати з помилками або взагалі вийде з ладу. У такому випадку втручання у програмний код з метою обходу алгоритмів захисту і створенні своїх копій з боку зловмисника ні до чого не призведе.

- Водяні знаки у програмному забезпеченні

Власники програмного забезпечення можуть вбудовувати прихований водяний знак у початковий код. За необхідності він зможе підтвердити права власності або походження програми при її вилученні. Така відстежуваність допоможе запобігти створенню нелегальних копій, оскільки права власності справжніх авторів можна буде легко підтвердити [26].

2.3. Модель захисту програмного забезпечення

З метою реалізації поставленої задачі, було обрано деякі засоби програмного захисту, що описано вище.

При використанні розробленого програмного додатку користувач має підтвердити ключ програмної копії продукту. Його можна отримати на сайті розробника або на електронну адресу користувача після підтвердження оплати ліцензійної версії програми. Перший запуск завантаженої програми ініціює запуск підтвердження ліценції у спеціальному вікні. Після внесення користувачем свого індивідуального ключа відбувається зв'язок з базою даних розробника з метою підтвердження того, що такий ключ існує і належить автентифікованому користувачу. Також цьому ключу присвоюється у відповідність унікальний індикатор комп'ютера, з якого надходить запит.

Якщо відбувається підтвердження власника ключа, на комп'ютері створюється файл ліцензії з занесенням деяких даних. Цей файл перевіряється при кожному наступному старті програми, і якщо буде

виявлено якісь зміни у ньому або запит певного ключа з іншого пристрою, програма повідомить.

За допомогою певних алгоритмів, програмний код додатку буде модифіковано. У налаштування програми можна буде впливати на деякі параметри модифікації з метою більш індивідуального налаштування обфускації. Такі дії зможуть заплутати аналіз алгоритмів роботи програми. Також при використанні коду програм у форматі dll забезпечується більш високий рівень захисту.

Обрані системи повинні забезпечити достатній рівень захисту програмного забезпечення в автоматичному режимі.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Опис програмної реалізації

Для програмної реалізації поставленої задачі обрано мову програмування C#. Для реалізації графічного інтерфейсу використовується Windows Form. Програмування відбувається у середовищі розробки Visual Studio 2022.

Розроблений додаток може бути використаний розробниками для захисту вже функціонуючих програмах.

Під час запуску додатку відбувається запуск першої форми (рис. 3.1.), на якій наведено посилання для ознайомлення з ліцензійною угодою використання, з якою користувач повинен погодитися для використання програми. Якщо усі положення угоди задовольняють користувача, він може продовжити роботу, натиснувши відповідну кнопку. У іншому випадку, можна припинити роботу, закривши вікно програми.

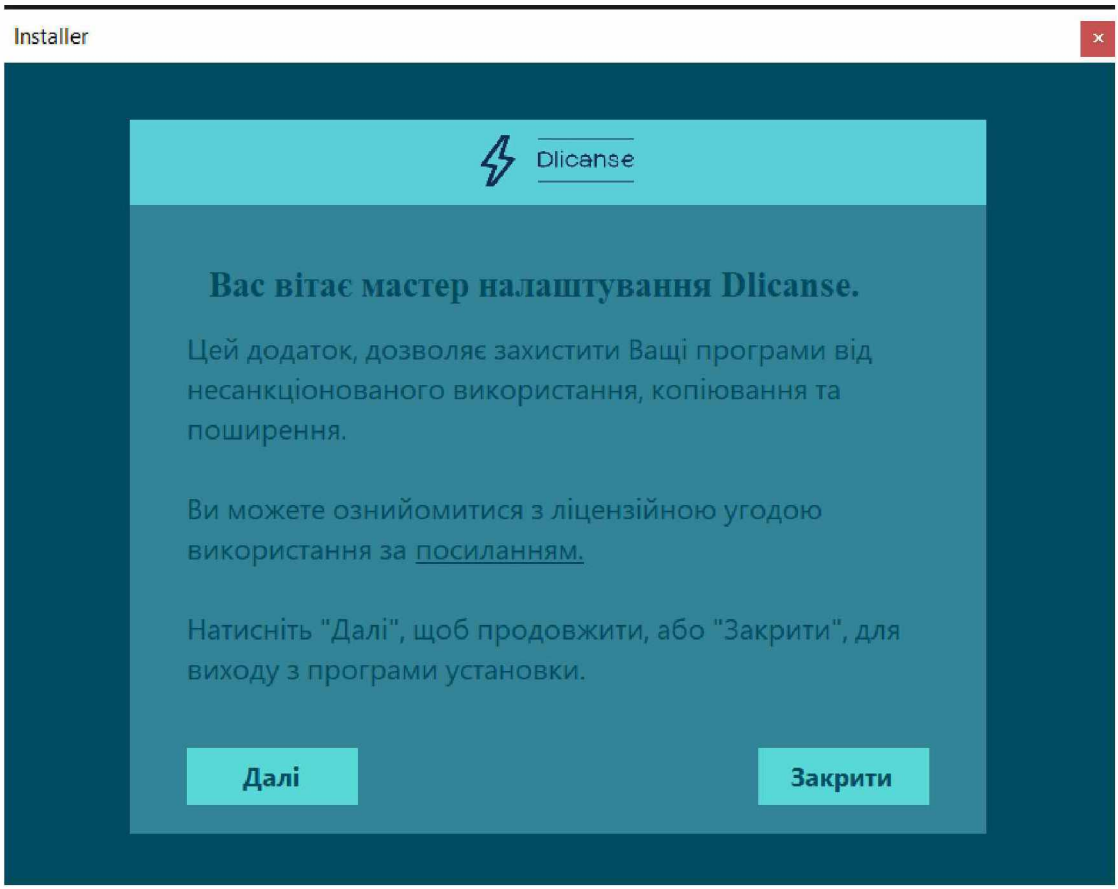

Рисунок 3.1 – Стартова форма програми

На наступній формі (рис 3.2.) пропонується обрати шлях розташування файлів, що потребують захисту. Після цього можна вибрати елементи захисту коду та запустити їх на виконання.

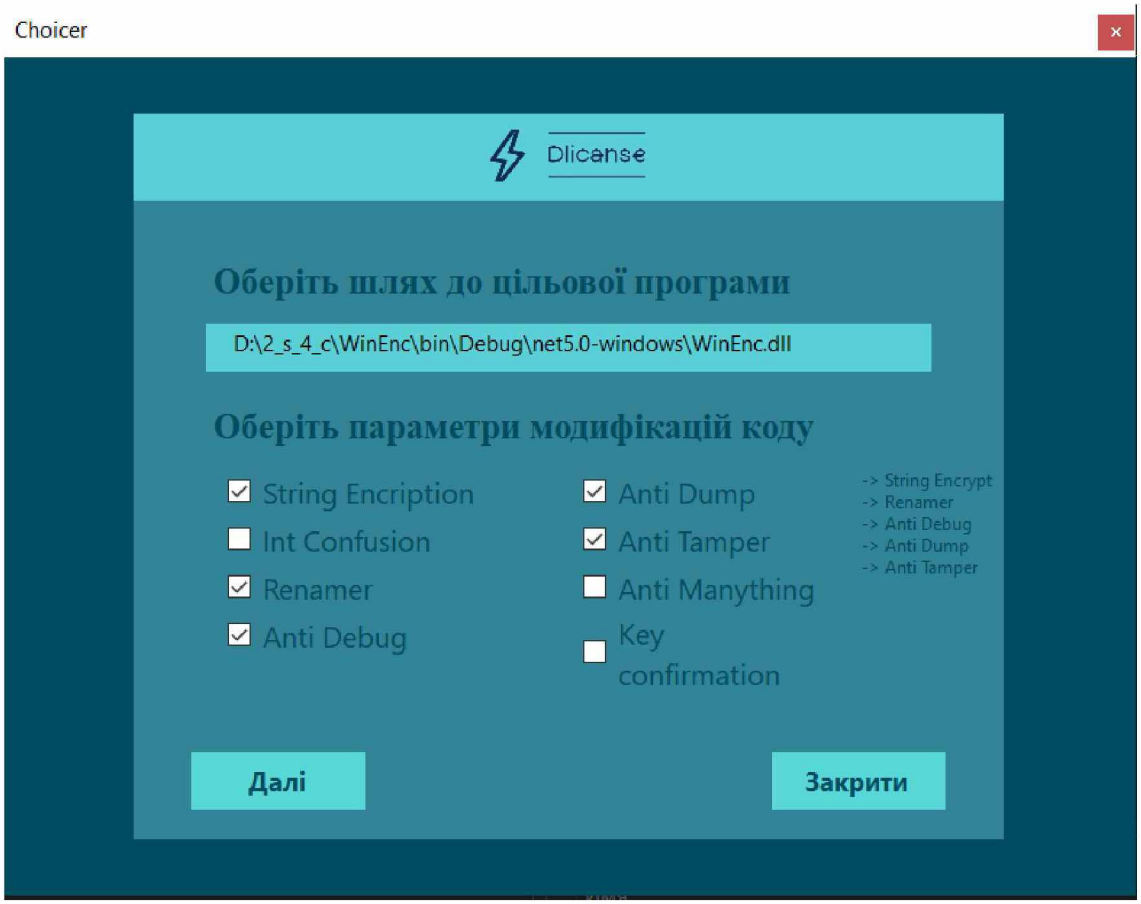

Рисунок 3.2 – Вікно вибору параметрів захисту

Наведені пункти надають можливість обфускації коду за деякими методами та перевірки ліцензійного ключа. Обрані пункти позначають у лівій частині вікна для більш зручного орієнтування. Коли усі необхідні пункти обрано, при натисканні відповідної кнопки “Далі” запускається процес реалізації захисту в залежності від вибору на попередньому етапі. Після завершення роботи, виводиться відповідне повідомлення, що інформує користувача про це.

При розробці способів обфускації коду для реалізації захисту було використано деякі ідеї з репозиторію на платформі GitHub [27]. У зазначеному проєкті були наведені приклади роботи різноманітних підходів

захисту коду. Окрім обфускації було розроблено перевірку ліцензійного ключа програми. Основні елементи коду програми наведено у Додатку А.

Серед можливих способів захисту наведені наступні:

1) Шифрування рядкових змінних у проекті.
2) Внутрішня перестановка методів, та елементів програми.
3) Перейменування усіх змінних для ускладнення розуміння алгоритмів роботи.
4) Засоби протидії відлагодженню коду.
5) Протидія аналізу дампу пам'яті програми.
6) Захист від несанкціонованих змін.
7) Реалізація загального захисту за декількома напрямками.
8) Вбудовування перевірки ліцензійного ключа програми.

Перші сім пунктів націлені саме на захист коду від можливого аналізу, для протидії несанкціонованого копіювання та можливого подальшого розповсюдження.

Останній пункт націлено саме на захист від несанкціонованого використання. У цьому пункті відбувається вбудовування перевірки ліцензійного ключа, без якого програма не зможе бути запущена. Передбачається, що поширювати свою, вже захищену за допомогою цього додатку, програму власники будуть через відповідні сервіси, де користувачі зможуть зареєструватися за допомогою, наприклад, email адреси та придбати права на використання програмного продукту. На електронну адресу буде надіслано відповідний ліцензійний ключ, що пов'язується у базі даних з поштою користувача. Після встановлення та запуску програми з боку користувача буде запропоновано ввести дані, що підтверджують права на можливість використання програми (рис 3.3.). Необхідно ввести email адресу та ліцензійний ключ, що надійшов на цю адресу. Якщо користувач не отримав ліцензії або виникли якісь проблеми, він може натиснути на відповідний текст «У мене не має ліцензійного ключа», що допоможе відкрити у браузері сайт підтримки власників програми.

Уведені дані порівнюються з базою даних (рис. 3.4.), у якій зберігаються згенерована пара ліцензійного ключа та певні секретні слова, що необхідні на наступних етапах та за наявності відповідні електронні адреси, зчитані фізичні ідентифікатори.

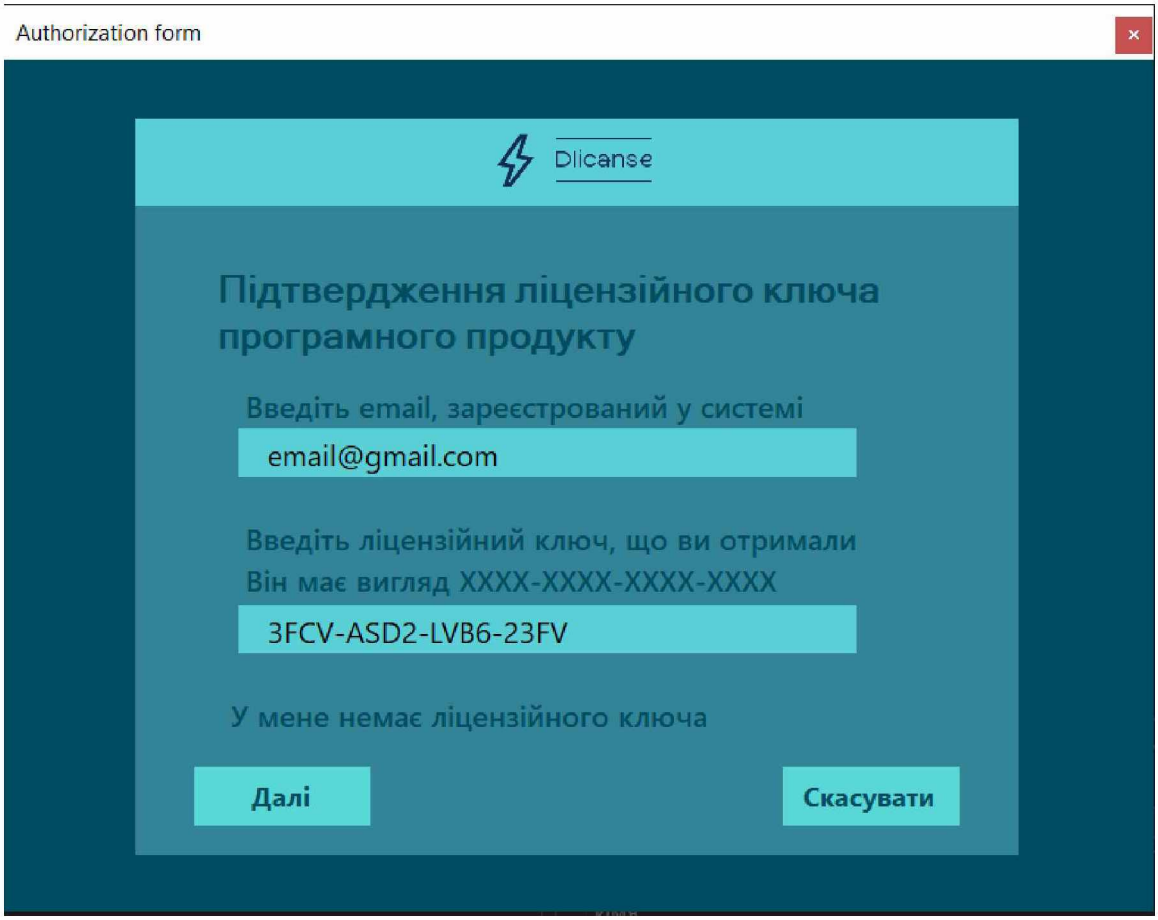

Рисунок 3.3 – Стартове вікно перевірки ліцензії

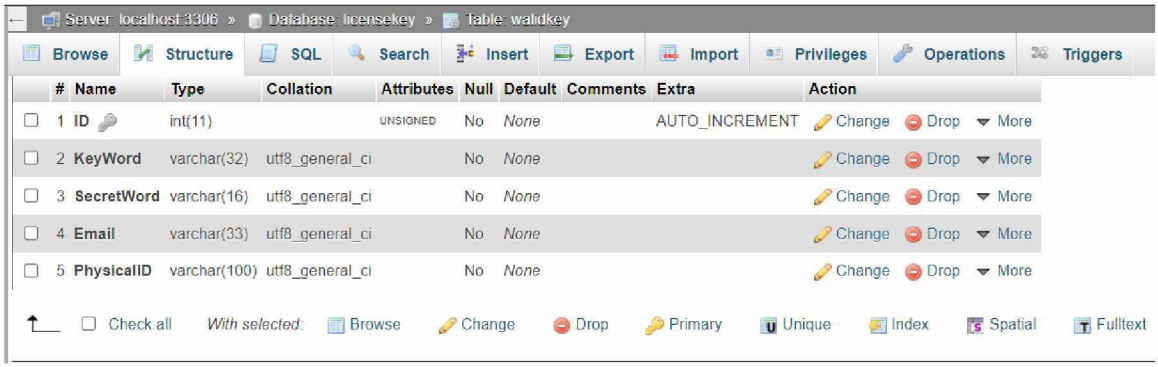

Рисунок 3.4 – Структура бази даних

Якщо дані, що увів користувач збігаються з базою даних, здійснюється читання серійних номерів деякого фізичного обладнання пристрою, на якому запущена програма. Це необхідно для того, щоб запобігти використанню програми на сторонніх пристроях та запобігти використанню програми декількома користувачами з одного придбаного ключа. Фізичний ідентифікатор заноситься у базу даних до відповідної комірки, після чого додаток отримує секретне слово із запису в базі, необхідне для подальших дій програми.

На наступному етапі відбувається створення ліцензійного файлу «license.xml» (рис 3.5) за місцем локального розташування програми. У файл заносяться ті дані, що увів користувач, його фізичний ідентифікатор та сигнатура, що складається з хеш-суми фізичного ID та секретного ключа, що було отримано з бази даних.

Рисунок 3.5 – Вміст файлу «license.xml»

При подальших стартах програми відбуватиметься перевірка вмісту файлу ліцензії без необхідності внесення повторних даних та зіставлення з даними у базі даних задля забезпечення того, що конкретна копія програми активована та може працювати лише на визначеному комп’ютері.

Якщо при старті програми буде зафіксовано зміни у файлі ліцензії або відмінності фізичних компонентів системи, відбудеться запуск відповідного вікна з повідомленням про це (рис 3.6.). Якщо зміна або пошкодження файлу відбулося випадково, а також заміна комплектуючих комп’ютера ініційована

користувачем, буде запропоновано перейти на сторінку підтримки розробника, де можна буде підтвердити особистість користувача та права на використання ліцензійної копії програми, і ввести зміни щодо фізичних ідентифікаторів пристрою або повторно отримати коректний файл ліцензії.

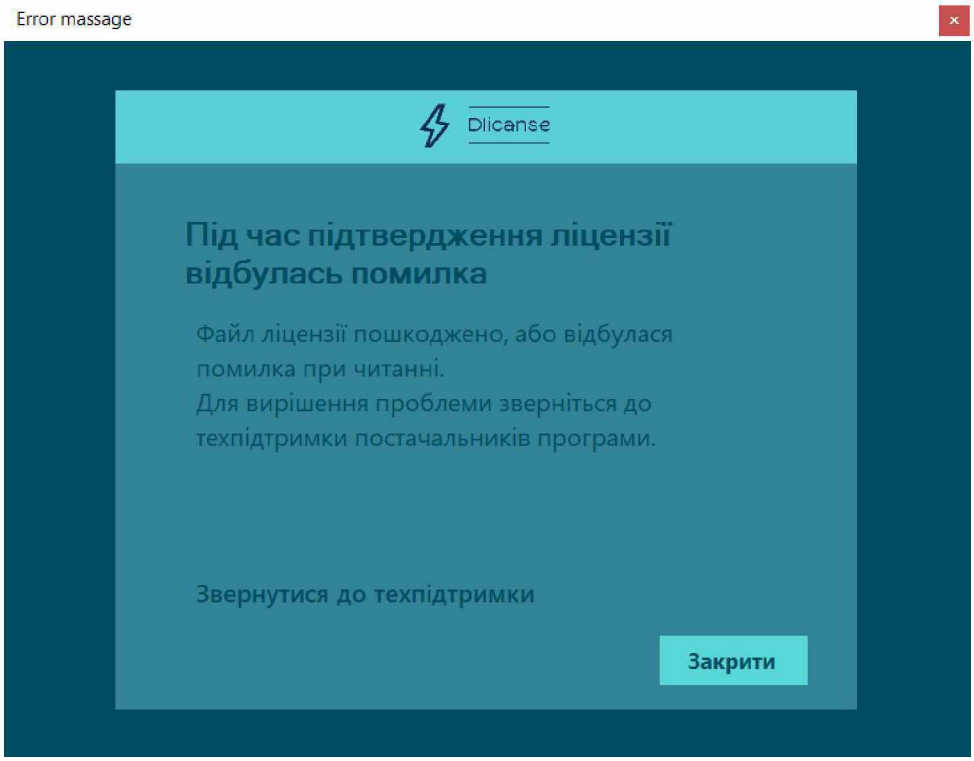

Рисунок 3.6 – Вікно повідомлення про помилку

Після завершення попередніх етапів буде виведено повідомлення з відповідним текстом про успішно пройдений процес авторизації у системі (рис. 3.7.).

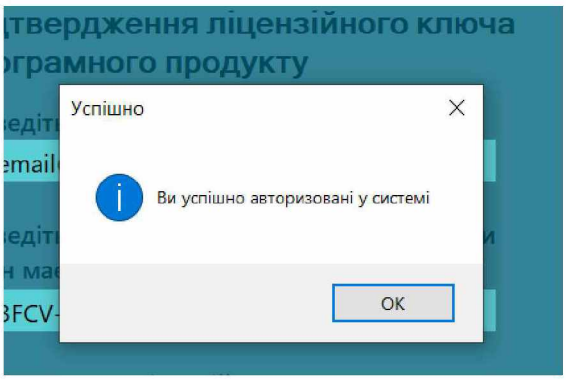

Рисунок 3.7 – Повідомлення про успішне виконання

3.2. Тестування та приклад роботи

Для тестування роботи додатку було створено просту Windows Form програму на мові C# (рис 3.8.), функціонал якої полягає у шифруванні та дешифруванні тексту за допомогою шифру Віженера.

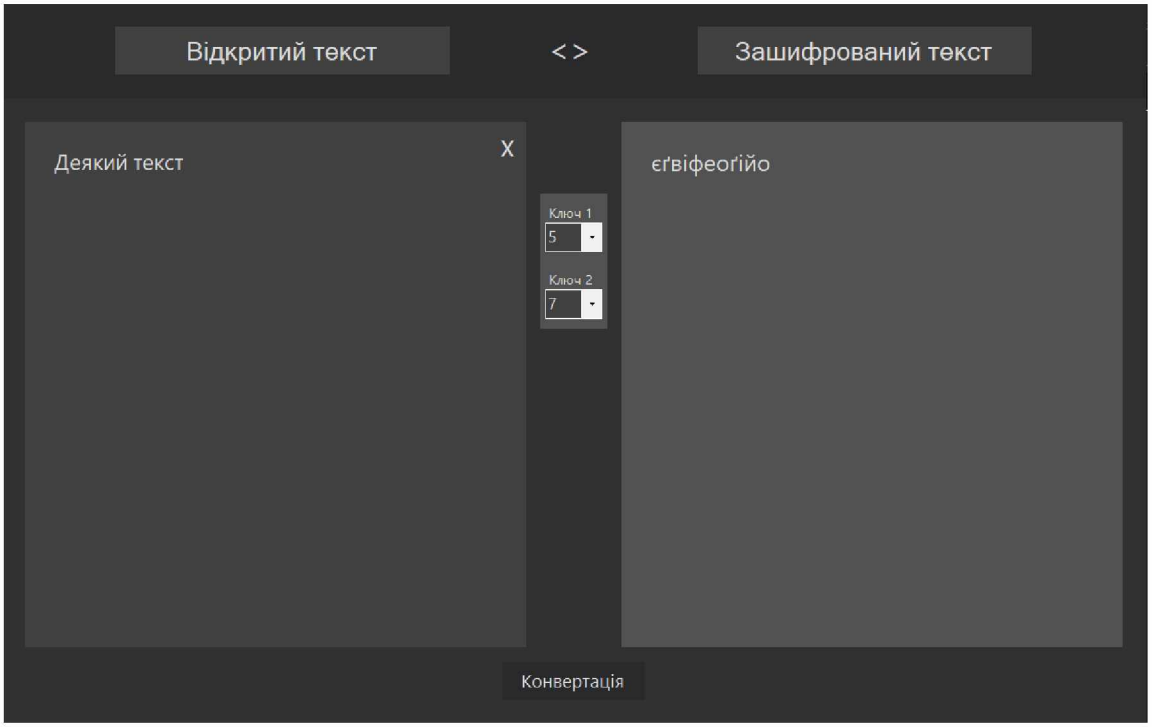

Рисунок 3.8 – Приклад програми, яку необхідно захистити

На даному етапі програму може запустити та використовувати кожен, хто її встановить. Також за допомогою деяких програм декомпіляції можна отримати доступ до відкритого вихідного коду програми. Наприклад, використовуючи програму JetBrains dotPeek можна продивитися усі класи та код, з якого складається обрана програма (рис. 3.9.). Можна переглянути алгоритми роботи, методи, які використовуються, та внести корективи. Для «піратів» у такому випадку не складно буде відтворити копію такої програми або вирізати елементи якогось можливого захисту і розповсюджувати її для своїх потреб. У такому вигляді розповсюджувати комерційне програмне забезпечення дуже небезпечно, тому його необхідно захистити.

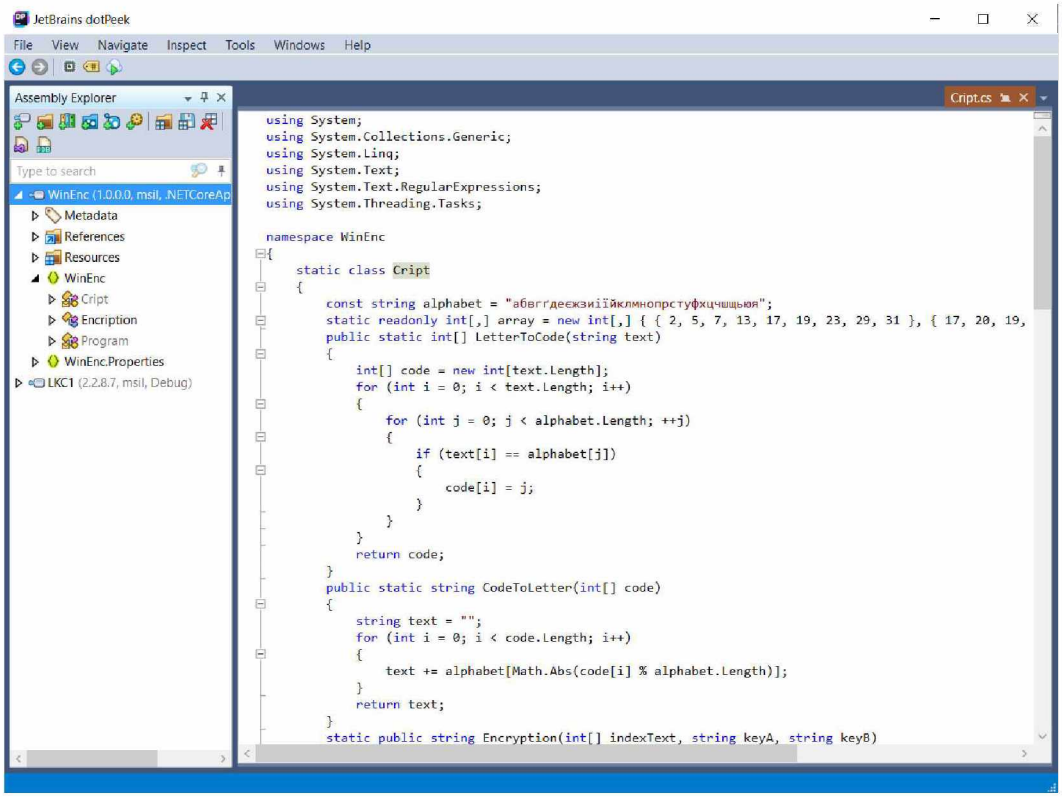

Рисунок 3.9 – Приклад декомпіляції файлів незахищеної програми

Отже, використовуючи цей додаток для шифрування, забезпечимо йому захист за допомогою розробленого підходу. Для початку запускаємо розроблений додаток для створення захисту та обираємо шлях до файлів, що необхідно захистити (рис. 3.10). Після чого обираємо необхідні пункти захисту та запускаємо їх на виконання. Обрані елементи захисту дозволять забезпечити перейменування усіх класів, методів та змінних, переставлять їх місцями, зашифрують рядкові змінні, забезпечать захист від відлагодження, від аналізу дампу пам’яті та внесення змін у код програми. Буде також вбудована перевірка ліцензійного ключа. Усе це зможе забезпечити достатній рівень захищеності обраної програми.

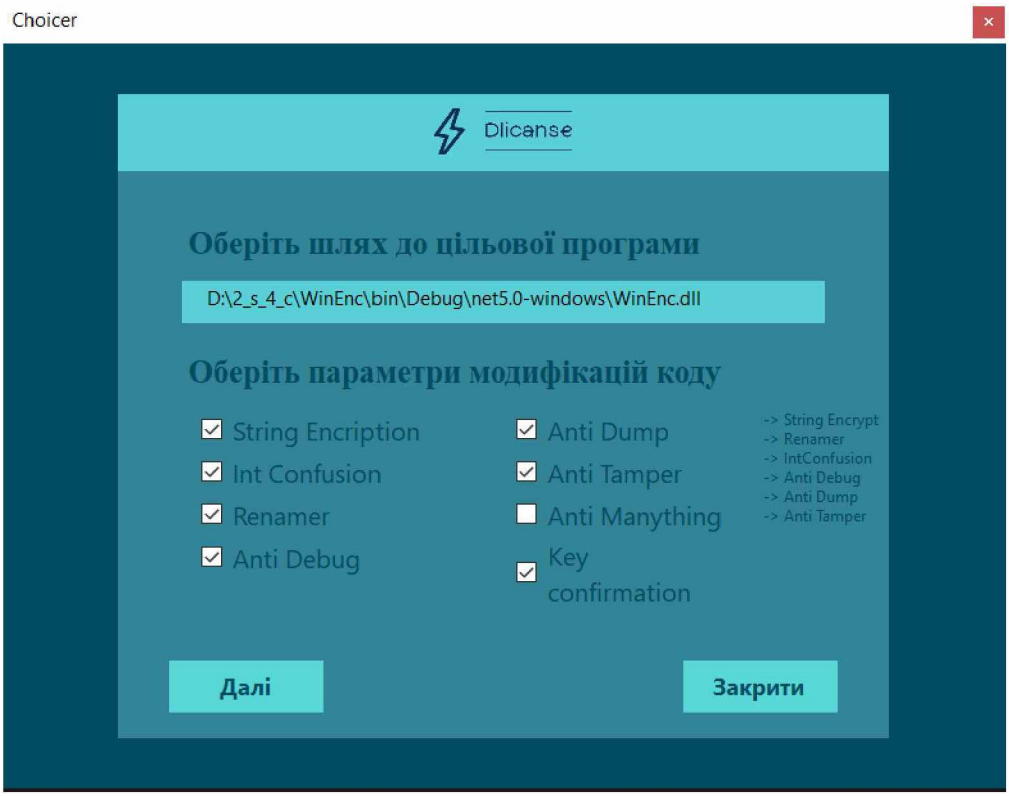

Рисунок 3.10. – Етап вибору методів захисту програми

Отже, після модифікації коду ми отримуємо захищену версію програми для шифрування. Для перевірки цього факту знову використаємо декомпілятор і результат його роботи наведено на рис. 3.11.

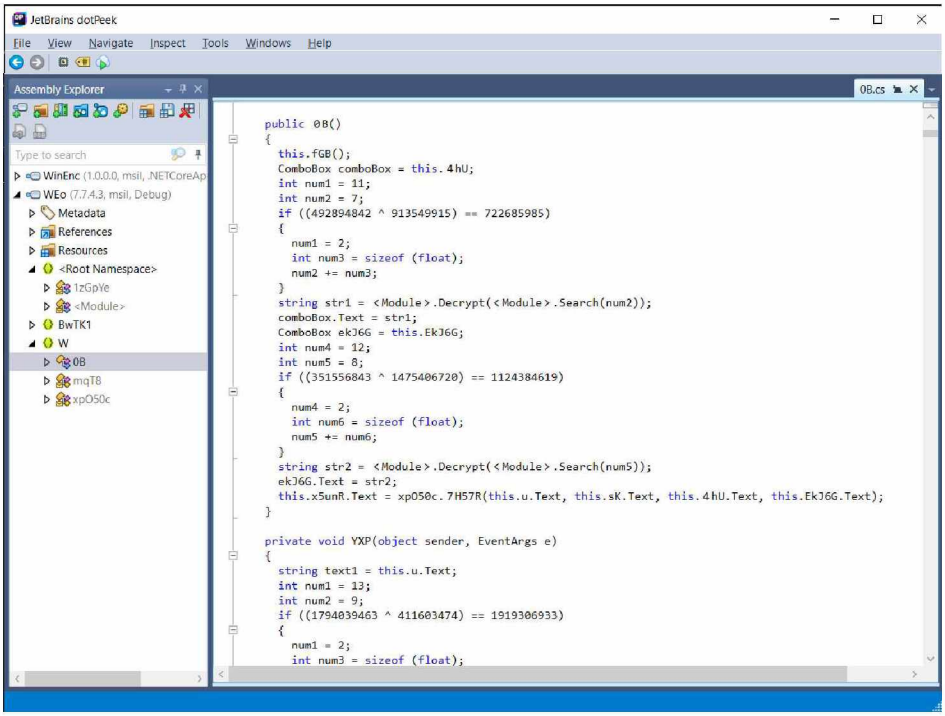

Рисунок 3.11. – Приклад декомпіляції файлів захищеної програми

Як можна переконатися з результатів роботи декомпілятора при порівнянні рис. 3.9 та рис. 3.11 у коді відбулися значні зміни, що значно ускладнює можливість аналізу його роботи. Крім цього, відбулося перейменування змінних та внесення методів, що заплутують програмний код. Також підтвердження ліцензії було успішно вбудовано у додаток для шифрування. При старті програми відбувається запит на введення email адреси, приклад якого було наведено на рис. 3.3. Створення ліцензійного файлу також відбувається без проблем.

Отже, можна констатувати, що захист обраної програми для шифрування реалізовано успішно, оскільки були внесені необхідні зміни для обфускації коду та вбудовано перевірку ліцензії.

ВИСНОВКИ

Використання комп'ютера ставить користувачів перед вибором між ліцензійним або піратським програмним забезпеченням. Нажаль, на сьогоднішній день велика кількість звичайних користувачів, а інколи й малий та середній бізнес вдаються до використання неліцензійного ПЗ. Такий вибір може наражати на небезпеку атак з боку шкідливих програм. Разом з піратським додатком можуть бути встановлені віруси, комп'ютерні хробаки, шпигунські програми, майнери тощо. Такі атаки також можуть призвести до втрати важливих даних та великих збитків. Ще більших збитків можуть понести розробники програмного забезпечення, адже вся робота, що може вестися декілька років і вимагати фінансових вкладів, врешті-решт може не окупитися. Піратство позбавляє розробників частини коштів, що вони могли б заробити на продажі копій своїх програм.

Отже, виходячи з описаної проблематики, що створює піратське програмне забезпечення, постає необхідність реалізації деяких систем захисту програмного забезпечення від несанкціонованого доступу, копіювання та поширення.

Проте, не всі розробники мають достатньо часу та фінансування для реалізації своїх засобів захисту для реалізованого ПЗ.

Виходячи з цього, було розроблено програму, за допомогою якої можна вбудувати у готові програми механізми захисту. Використання такої програми допоможе автоматизувати процес захисту програмного забезпечення з боку розробників для подальшого розповсюдження вже більш захищеної версії своїх додатків.

Реалізоване програмне забезпечення дозволяє додати перевірку та підтвердження ліцензійної копії продукту. При активації такого ключа йому привласнюється відповідний ідентифікатор комп'ютера, на якому відбувається активація. Після авторизації відбувається створення локального ліцензійного файлу, за допомогою якого і відбуватиметься подальша перевірка прав на використання захищеної програми. Прив'язка до фізичного

індикатора допоможе впевнитись у тому, що програма може бути запущена лише з конкретного пристрою, що належить авторизованому користувачу.

Також додаток забезпечує можливість обфускації програмного коду початкової програми з метою ускладнення процесу аналізу алгоритмів його роботи.

Отже, такий додаток може допомогти розробникам, які не можуть реалізувати власний захист. Його можна використовувати для забезпечення середнього рівня безпеки. Цього буде достатньо для власників програм та невеликих компаній, що прагнуть захистити свою інтелектуальну власність.

У подальшому можливе вдосконалення цього додатку для розширення його функціональних можливостей. Розробка можливості адаптації під різноманітні мови програмування, не обмежуючись лише C#. Можливо також передбачити розроблення мобільного аналогу цієї програми для забезпечення захисту мобільних додатків. Крім цього, вбачаємо ще можливість перенести додаток на іншу платформу для розширення переліку систем, на якій він зможе коректно працювати.

СПИСОК ЛІТЕРАТУРИ

1. Ризики використання неліцензійного програмного забезпечення [Електронний ресурс] // Офіційний сайт компанії N-Space. – Режим доступу: https://nspace.ua/info/riziki-vikoristannya-nelitsenzijnogo-programnogo-zabezpechennya/

2. 80% всего ПО в Украине – нелицензионное [Електронний ресурс] // Український онлайн-журнал AIN.UA. – Режим доступу: https://ain.ua/ru/2018/09/19/80-vsego-po-v-ukraine-nelicenzionnoe/

3. What is software? A guide to all of the different types of programs and applications that tell computers what to do [Електронний ресурс] // Business Insider – Режим доступу: https://www.businessinsider.com/what-is-software

4. Операційна система [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki

5. Вбудоване програмне забезпечення [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki

6. Device driver [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Device_driver

7. Utility [Електронний ресурс] // The Tech Terms Computer Dictionary. – Режим доступу: https://techterms.com/definition/utility

8. Інформатика та комп'ютерна техніка [Електронний ресурс] // Київський професійно-педагогічний коледж імені Антона Макаренка. – Режим доступу: https://kppk.com.ua/ELLIB/ebook/Gorbenko/IKT/3.

9. Налагоджувач [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki

10. Компонувальник [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki

11. Malware [Електронний ресурс] // The Tech Terms Computer Dictionary. – Режим доступу: https://web.archive.org/web/20181122022325/https://techterms.com/definition/malware

12. Комп'ютерний вірус [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://uk.wikipedia.org/wiki

13. Computer worm [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Computer_worm

14. Trojan horse (computing) [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Trojan_horse_(computing)

15. Spyware [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Spyware

16. Commercial software [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki.

17. Open-source software [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Open-source_software

18. Freeware [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Freeware

19. Shareware [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: https://en.wikipedia.org/wiki/Shareware

20. Software piracy [Електронний ресурс] // Вільна енциклопедія Вікіпедія. – Режим доступу: http://69.63.68.22/archive/doc/wikipedia/wikipedia-terodump-0.1/tero-dump/wikipedia/so/Software_piracy.html

21. Кодекс України про адміністративні правопорушення [Електронний ресурс] // Офіційний портал Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/80731-10#Text

22. Кримінальний кодекс України [Електронний ресурс] // Офіційний портал Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2341-14#Text

23. Закон України «Про авторське право і суміжні права» [Електронний ресурс] // Офіційний портал Верховної Ради України. – Режим доступу: https://zakon.rada.gov.ua/laws/show/3792-12#Text

24. Канафеев, Р. И. Системы защиты программного обеспечения / Р. И. Канафеев. — Текст : непосредственный // Молодой ученый. — 2019. — № 21 (259). — С. 34-36. — URL: https://moluch.ru/archive/259/59688

25. Оцінка ефективності систем захисту програмного забезпечення [Електронний ресурс] // Сайт руху по захисту прав користувачів програмного забезпечення. – Режим доступу: https://web.archive.org/web/20190125183712/http://consumer.stormway.ru/sps_eval.htm

26. How to stop software piracy [Електронний ресурс] // Red Points official website. – Режим доступу: https://www.redpoints.com/blog/how-to-stop-software-piracy/?nab=1&utm_referrer=https%3A%2F%2Fwww.google.com%2F

27. MindLated [Електронний ресурс] // Веб сервіс для хостингу IT-проектів GitHub. - Режим доступу: https://github.com/Sato-Isolated/MindLated

28. DLicanse [Електронний ресурс] // Веб сервіс для хостингу IT-проектів GitHub. - Режим доступу: https://github.com/GrindDevol/DLicanse

ДОДАТОК А

Основні елементи програмної реалізації

Ознайомитися з повним кодом програми можна за посиланняя на репозиторії GitHub [28].

Клас Renamer, призначений для перейменування змінних у проекті:

```
using dnlib.DotNet;
using dnlib.DotNet.Emit;
using System;
using System.Collections.Generic;
using System.Linq;

namespace DInstaller.Protection.Renamer
{
   public static class RenamerPhase
   {
      private static readonly Random Random = new();

      private static string RandomString(int length, string chars)
      {
         return new(Enumerable.Repeat(chars, length)
           .Select(s => s[Random.Next(s.Length)]).ToArray());
      }

      public enum RenameMode
      {
         Ascii,
         Key,
         Normal
      }

      private static readonly string[] NormalNameStrings = {
         "HasPermission", "HasPermissions", "GetPermissions", "GetOpenWindows",
"EnumWindows", "GetWindowText", "GetWindowTextLength", "IsWindowVisible",
"GetShellWindow", "Awake", "FixedUpdate", "add_OnRockedInitialized",
"remove_OnRockedInitialized", "Awake", "Initialize", "Translate", "Reload",
"<Initialize>b__13_0", "Initialize", "FixedUpdate", "Start", "checkTimerRestart",
"QueueOnMainThread", "QueueOnMainThread", "RunAsync", "RunAction", "Awake",
"FixedUpdate", "IsUri", "GetTypes", "GetTypesFromParentClass",
"GetTypesFromParentClass", "GetTypesFromInterface", "GetTypesFromInterface",
"get_Timeout", "set_Timeout", "GetWebRequest", "get_SteamID64", "set_SteamID64",
"get_SteamID", "set_SteamID", "get_OnlineState", "set_OnlineState", "get_StateMessage",
"set_StateMessage", "get_PrivacyState", "set_PrivacyState", "get_VisibilityState",
"set_VisibilityState", "get_AvatarIcon", "set_AvatarIcon", "get_AvatarMedium",
"set_AvatarMedium", "get_AvatarFull", "set_AvatarFull", "get_IsVacBanned",
"set_IsVacBanned", "get_TradeBanState", "set_TradeBanState", "get_IsLimitedAccount",
"set_IsLimitedAccount", "get_CustomURL", "set_CustomURL", "get_MemberSince",
```

```
"set_MemberSince", "get_HoursPlayedLastTwoWeeks", "set_HoursPlayedLastTwoWeeks",
"get_Headline", "set_Headline", "get_Location", "set_Location", "get_RealName",
"set_RealName", "get_Summary", "set_Summary", "get_MostPlayedGames",
"set_MostPlayedGames", "get_Groups", "set_Groups", "Reload", "ParseString",
"ParseDateTime", "ParseDouble", "ParseUInt16", "ParseUInt32", "ParseUInt64", "ParseBool",
"ParseUri", "IsValidCSteamID", "LoadDefaults", "LoadDefaults", "get_Clients", "Awake",
"handleConnection", "FixedUpdate", "Broadcast", "OnDestroy", "Read", "Send",
"<Awake>b__8_0", "get_InstanceID", "set_InstanceID", "get_ConnectedTime",
"set_ConnectedTime", "Send", "Read", "Close", "get_Address", "get_Instance", "set_Instance",
"Save", "Load", "Unload", "Load", "Save", "Load", "get_Configuration", "LoadPlugin",
"<.ctor>b__3_0", "<LoadPlugin>b__4_0", "add_OnPluginUnloading",
"remove_OnPluginUnloading", "add_OnPluginLoading", "remove_OnPluginLoading",
"get_Translations", "get_State", "get_Assembly", "set_Assembly", "get_Directory",
"set_Directory", "get_Name", "set_Name", "get_DefaultTranslations", "IsDependencyLoaded",
"ExecuteDependencyCode", "Translate", "ReloadPlugin", "LoadPlugin", "UnloadPlugin",
"OnEnable", "OnDisable", "Load", "Unload", "TryAddComponent", "TryRemoveComponent",
"add_OnPluginsLoaded", "remove_OnPluginsLoaded", "get_Plugins", "GetPlugins",
"GetPlugin", "GetPlugin", "Awake", "Start", "GetMainTypeFromAssembly", "loadPlugins",
"unloadPlugins", "Reload", "GetAssembliesFromDirectory", "LoadAssembliesFromDirectory",
"<Awake>b__12_0", "GetGroupsByIds", "GetParentGroups", "HasPermission", "GetGroup",
"RemovePlayerFromGroup", "AddPlayerToGroup", "DeleteGroup", "SaveGroup", "AddGroup",
"GetGroups", "GetPermissions", "GetPermissions", "<GetGroups>b__11_3", "Start",
"FixedUpdate", "Reload", "HasPermission", "GetGroups", "GetPermissions", "GetPermissions",
"AddPlayerToGroup", "RemovePlayerFromGroup", "GetGroup", "SaveGroup", "AddGroup",
"DeleteGroup", "DeleteGroup", "<FixedUpdate>b__4_0", "Enqueue", "_Logger_DoWork",
"processLog", "Log", "Log", "var_dump", "LogWarning", "LogError", "LogError", "Log",
"LogException", "ProcessInternalLog", "logRCON", "writeToConsole", "ProcessLog",
"ExternalLog", "Invoke", "_invoke", "TryInvoke", "get_Aliases", "get_AllowedCaller",
"get_Help", "get_Name", "get_Permissions", "get_Syntax", "Execute", "get_Aliases",
"get_AllowedCaller", "get_Help", "get_Name", "get_Permissions", "get_Syntax", "Execute",
"get_Aliases", "get_AllowedCaller", "get_Help", "get_Name", "get_Permissions", "get_Syntax",
"Execute", "get_Name", "set_Name", "get_Name", "set_Name", "get_Name", "get_Help",
"get_Syntax", "get_AllowedCaller", "get_Commands", "set_Commands",
"add_OnExecuteCommand", "remove_OnExecuteCommand", "Reload", "Awake",
"checkCommandMappings", "checkDuplicateCommandMappings",
"Plugins_OnPluginsLoaded", "GetCommand", "GetCommand", "getCommandIdentity",
"getCommandType", "Register", "Register", "Register", "DeregisterFromAssembly",
"GetCooldown", "SetCooldown", "Execute", "RegisterFromAssembly"
    };

    private const string Ascii =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

    private static string GetRandomName()
    {
        return NormalNameStrings[Random.Next(NormalNameStrings.Length)];
    }

    public static string GenerateString(RenameMode mode)
    {
        return mode switch
```

```
            {
                RenameMode.Ascii => RandomString(Random.Next(1, 7), Ascii),
                RenameMode.Key => RandomString(16, Ascii),
                RenameMode.Normal => GetRandomName(),
                _ => throw new ArgumentOutOfRangeException(nameof(mode), mode, null)
            };
        }

        private static readonly Dictionary<string, string> Names = new();

        public static ModuleDefMD ExecuteClassRenaming(ModuleDefMD module)
        {
            foreach (var type in module.GetTypes())
            {
                if (type.IsGlobalModuleType) continue;

                if (type.Name == "GeneratedInternalTypeHelper" || type.Name == "Resources" ||
type.Name == "Settings")
                    continue;
                if (Names.TryGetValue(type.Name, out var nameValue))
                    type.Name = nameValue;
                else
                {
                    var newName = GenerateString(RenameMode.Ascii);

                    Names.Add(type.Name, newName);
                    type.Name = newName;
                }
            }
            return ApplyChangesToResourcesClasses(module);
        }

        private static ModuleDefMD ApplyChangesToResourcesClasses(ModuleDefMD module)
        {
            var moduleToRename = module;

            foreach (var resource in moduleToRename.Resources)
            {
                foreach (var item in Names)
                {
                    if (resource.Name.Contains(item.Key))
                    {
                        resource.Name = resource.Name.Replace(item.Key, item.Value);
                    }
                }
            }

            foreach (var type in moduleToRename.GetTypes())
            {
                foreach (var property in type.Properties)
                {
```

```
                if (property.Name != "ResourceManager")
                    continue;

                var instr = property.GetMethod.Body.Instructions;

                foreach (var t in instr)
                {
                    if (t.OpCode == OpCodes.Ldstr)
                    {
                        foreach (var (key, value) in Names.Where(item =>
t.Operand.ToString()!.Contains(item.Key)))
                        {
                            t.Operand = t.Operand?.ToString()?.Replace(key, value);
                        }
                    }
                }
            }
        }
        return moduleToRename;
    }

    public static ModuleDefMD ExecuteFieldRenaming(ModuleDefMD module)
    {
        foreach (var type in module.GetTypes())
        {
            if (type.IsGlobalModuleType) continue;

            foreach (var field in type.Fields)
            {
                if (Names.TryGetValue(field.Name, out var nameValue))
                    field.Name = nameValue;
                else
                {
                    var newName = GenerateString(RenameMode.Ascii);

                    Names.Add(field.Name, newName);
                    field.Name = newName;
                }
            }
        }

        return ApplyChangesToResourcesField(module);
    }

    private static ModuleDefMD ApplyChangesToResourcesField(ModuleDefMD module)
    {
        foreach (var type in module.GetTypes())
        {
            if (type.IsGlobalModuleType) continue;

            foreach (var method in type.Methods)
```

```
            {
                if (method.Name != "InitializeComponent")
                    continue;
                var instr = method.Body.Instructions;

                for (var i = 0; i < instr.Count - 3; i++)
                {
                    if (instr[i].OpCode == OpCodes.Ldstr)
                    {
                        foreach (var item in Names.Where(item => item.Key ==
instr[i].Operand.ToString()))
                        {
                            instr[i].Operand = item.Value;
                        }
                    }
                }
            }
        }
        return module;
    }

    public static ModuleDefMD ExecuteMethodRenaming(ModuleDefMD module)
    {
        foreach (var type in module.GetTypes())
        {
            if (type.IsGlobalModuleType) continue;

            if (type.Name == "GeneratedInternalTypeHelper")
                continue;

            foreach (var method in type.Methods)
            {
                if (!method.HasBody) continue;

                if (method.Name == ".ctor" || method.Name == ".cctor")
                    continue;

                method.Name = GenerateString(RenameMode.Ascii);
            }
        }
        return module;
    }

    public static void ExecuteModuleRenaming(ModuleDefMD mod)
    {
        foreach (var module in mod.Assembly.Modules)
        {
            var isWpf = false;
            foreach (var asmRef in module.GetAssemblyRefs())
                if (asmRef.Name == "WindowsBase" || asmRef.Name == "PresentationCore" ||
asmRef.Name == "PresentationFramework" || asmRef.Name == "System.Xaml")
```

```
                isWpf = true;
            if (!isWpf)
            {
                module.Name = GenerateString(RenameMode.Ascii);

                module.Assembly.CustomAttributes.Clear();
                module.Mvid = Guid.NewGuid();
                module.Assembly.Name = GenerateString(RenameMode.Ascii);
                module.Assembly.Version = new Version(Random.Next(1, 9), Random.Next(1, 9),
Random.Next(1, 9), Random.Next(1, 9));
            }
        }
    }

    public static ModuleDefMD ExecuteNamespaceRenaming(ModuleDefMD module)
    {
        foreach (var type in module.GetTypes())
        {
            if (type.IsGlobalModuleType) continue;

            if (type.Namespace == "")
                continue;

            if (Names.TryGetValue(type.Namespace, out var nameValue))
                type.Namespace = nameValue;
            else
            {
                var newName = GenerateString(RenameMode.Ascii);

                Names.Add(type.Namespace, newName);
                type.Namespace = newName;
            }
        }

        return ApplyChangesToResourcesNamespace(module);
    }

    private static ModuleDefMD ApplyChangesToResourcesNamespace(ModuleDefMD
module)
    {
        foreach (var resource in module.Resources)
        {
            foreach (var item in Names.Where(item => resource.Name.Contains(item.Key)))
            {
                resource.Name = resource.Name.Replace(item.Key, item.Value);
            }
        }

        foreach (var type in module.GetTypes())
        {
            foreach (var property in type.Properties)
```

```
            {
                if (property.Name != "ResourceManager")
                    continue;

                var instr = property.GetMethod.Body.Instructions;

                foreach (var t in instr)
                {
                    if (t.OpCode == OpCodes.Ldstr)
                    {
                        foreach (var (key, value) in Names.Where(item =>
t.ToString().Contains(item.Key)))
                        {
                            t.Operand = t.Operand?.ToString()?.Replace(key, value);
                        }
                    }
                }
            }
        }
        return module;
    }

    public static ModuleDefMD ExecutePropertiesRenaming(ModuleDefMD module)
    {
        foreach (var type in module.GetTypes())
        {
            if (type.IsGlobalModuleType) continue;

            foreach (var property in type.Properties)
            {
                property.Name = GenerateString(RenameMode.Ascii);
            }
        }
        return module;
    }
  }
}
```

Клас Generator, який формує фізичний ідентифікатор пристрою користувача, та генерує файл ліцензії, необхідний для перевірки ліцензійного ключа додатку:

```
using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Management;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Xml;

namespace DLicense
{
   static partial class Generator
   {
      private static string GetBaseBoardID()
      {
         ManagementClass mc = new ManagementClass("Win32_BaseBoard");
         ManagementObjectCollection moc = mc.GetInstances();
         string strID = null;
         foreach (ManagementObject mo in moc)
         {
            strID = mo.Properties["SerialNumber"].Value.ToString();
            break;
         }
         return strID;
      }
      private static string GetProcessorID()
      {
         ManagementClass mc = new ManagementClass("Win32_Processor");
         ManagementObjectCollection moc = mc.GetInstances();
         string strID = null;
         foreach (ManagementObject mo in moc)
         {
            strID = mo.Properties["ProcessorId"].Value.ToString();
            break;
         }
         return strID;
      }
      private static string GetBiosID()
      {
         ManagementClass mc = new ManagementClass("Win32_PhysicalMedia");
         ManagementObjectCollection moc = mc.GetInstances();
         string strID = null;
         foreach (ManagementObject mo in moc)
         {
            strID = mo.Properties["SerialNumber"].Value.ToString();
            break;
```

```
            }
            return strID;
        }
        public static string GetAllPhysicalID()
        {
            return (GetBiosID() + "/" + GetBaseBoardID() + "/" + GetProcessorID()).Trim();
        }
        public static void GenerateLicense(string Email, string Key)
        {
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(@"<license>
            <LicenseKey></LicenseKey>
            <Email></Email>
            <PhysicalID></PhysicalID>
            <Signature></Signature>
            </license>");
            doc.ChildNodes[0].SelectSingleNode(@"/license/LicenseKey", null).InnerText = Key;
            doc.ChildNodes[0].SelectSingleNode(@"/license/Email", null).InnerText = Email;
            doc.ChildNodes[0].SelectSingleNode(@"/license/PhysicalID", null).InnerText =
GetAllPhysicalID();
            doc.ChildNodes[0].SelectSingleNode(@"/license/Signature", null).InnerText =
GetHexString();
            doc.Save(System.IO.Path.Combine("license.xml"));
        }

        private static string GetHexString()
        {
            MD5 md5 = new MD5CryptoServiceProvider();
            byte[] data = System.Text.Encoding.UTF8.GetBytes(GetAllPhysicalID() + GetSecret());
            byte[] hash = md5.ComputeHash(data);
            return Convert.ToBase64String(hash);
        }

        public static string GetSecret()
        {
            DataBace db = new DataBace();
            db.openConnection();
            MySqlCommand commandGetSecretWord = new MySqlCommand("SELECT
`SecretWord` FROM `walidkey` WHERE `PhysicalID` = @physicalId", db.getConnection());
            commandGetSecretWord.Parameters.Add("@physicalId",
MySqlDbType.VarChar).Value = Generator.GetAllPhysicalID();
            string secretWord = commandGetSecretWord.ExecuteReader().ToString();
            db.closeConnection();
            return secretWord;
        }
    }
}
```