МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____ 20__ р.

На правах рукопису
УДК 04.056:004.623(079.2)

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Система захисту інформації від нелегального копіювання

Виконавець: П.В. Гайдук
Керівник: к.т.н., доцент С.В. Єгоров
Нормоконтролер: к.т.н., доцент С.В. Єгоров

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Гайдука Павла Вікторовича

1. Тема: *Система захисту інформації від нелегального копіювання* затверджена наказом ректора від «26» квітня 2021 р. № 652/ст.
2. Термін виконання: з 10.05.2021 р. по 20.06.2021 р.
3. Вихідні дані: проаналізувати існуючі техніки атак та механізми захисту від нелегального копіювання; розробити програмний засіб, базуючись на проаналізованих методах захисту від нелегального копіювання; протестувати розроблений програмний засіб.
4. Зміст пояснювальної записки: аналіз існуючих методів, технологій та атак для отримання інформації за рахунок нелегального копіювання; розробка програмного засобу генерації та реєстрації ключа програмного продукту; тестування програмного засобу.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Ознайомлення з постановкою задачі та вивчення літератури.	10.05.21	*Виконано*
2.	Аналіз технік та механізмів нелегального копіювання інформації	11.05.21 – 12.05.21	*Виконано*
3.	Аналіз існуючих механізмів протидії нелегальному копіюванню інформації	14.05.21 – 16.05.21	*Виконано*
4.	Створення програмного засобу для генерації та реєстрації ключів програмного додатку	17.05.21 – 25.05.21	*Виконано*
5.	Розробка рекомендацій для захисту інформації від нелегального копіювання	26. 05.21 – 30.05.21	*Виконано*
6.	Загальне редагування та друг пояснювальної записки, графічного матеріалу	1.06.21 – 6.06.21	*Виконано*
7.	Проходження нормо-контролю, перепліт пояснювальної записки	7.06.21	*Виконано*
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	8.06.21	*Виконано*
9.	Отримання відгуку керівника, рецензії	10.06.21 – 11.06.21	*Виконано*

Здобувач вищої освіти П.В. Гайдук

(підпис, дата)

Керівник дипломної роботи С.В. Єгоров

(підпис, дата)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 61 сторінка, 1 таблиця, 24 джерела, 6 рисунків, 4 додатки.

Об'єкт дослідження – захист інформації від нелегального копіювання.

Предмет дослідження – існуючі спеціалізовані прилади та методи запобіганню нелегальному копіюванню інформації.

Мета роботи – створення програмного засобу забезпечення безпеки програмного продукту та розробка рекомендацій щодо уникнення нелегального копіювання.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, звітів експертів з інформаційної безпеки, міжнародних стандартів та їх порівняння, проведення дослідження.

Вивчено найпоширеніші методи реалізації атак на інформацію станом на 2021 рік. Детально розібрані та класифіковані більшість популярних типів атак за останні роки.

Проаналізовано методи пом'якшення атак з використанням спеціалізованого ПЗ, постачальників послуг захисту або фізичних приладів захисту та виокремлено найбільш ефективні з них.

Створено програмний засіб, який надає можливість генерувати та реєструвати код програмного продукту, базуючись на унікальному ID комп'ютера.

Розроблено рекомендації протидії атакам, поєднуючи, встановлені в результаті опрацювання дипломної роботи, методи боротьби з розподіленими атаками відмови в обслуговуванні.

Галузь використання – інформаційна безпека.

Ключові слова: піратство, захист інформації, водяна марка, конфіденціальність, кібербезпека, авторське право.

ЗМІСТ

ВСТУП

Актуальність теми. Щорічно різні компанії, що надають послуги в галузі забезпечення інформаційної безпеки і протидії кібератакам, фіксують збільшення кількості нелегального копіювання і його потужність. Періодичні повідомлення в засобах масової інформації про втрату інформації тих чи інших ресурсів в результаті різних атак, говорять про неефективність засобів протидії такого роду атак. На тлі зазначених вище атак до провідних IT-корпорацій, також збільшується кількість атак на малий та середній бізнес, котрий, до недавнього часу, не становив інтересу для зловмисників. Однак, в даний час, у зв'язку зі збільшенням їх важливості і затребуваності, втрата інформації може бути критичною. Разом з цим змінюються і мотиви, які рухають зловмисниками, якщо раніше серед причин виникнення механізмів нелегального копіювання можна було виділити прагнення до змін утиліт, намагань використати їх у своїх цілях і т.д., то сьогодні все частіше нелегальне копіювання є наслідком шантажу і способом вимагання грошей.

Об'єкт дослідження – захист програмного забезпечення від нелегального копіювання.

Предмет дослідження – технології та механізми захисту від нелегального копіювання.

Мета роботи – створення програмного засобу забезпечення безпеки програмного продукту та розробка рекомендацій щодо уникнення нелегального копіювання.

Завдання бакалаврської роботи:

- проаналізувати існуючі типи атак на втрату конфіденційності компаній в їх програмних додатках;
- виокремити основні технології та механізми захисту комп'ютерних мереж нелегального копіювання;
- дослідити можливість поєднання встановлених технологій та механізмів виявлення та протидії атакам у рекомендації, які дійсно допоможуть компаніям не втратити свої кошти та час.

- створити програмний засіб, який надає можливість генерувати та реєструвати ключі програмного продукту.

Методи дослідження – аналіз експлуатаційної документації, звітів світових конференцій та засад з інформаційної безпеки, теорія машинного навчання, теорія корпоративних інформаційних систем та проведення досліджень з використанням спеціалізованого ПЗ.

Практична цінність. Удосконалено атаки на програмні забезпечення, з наведенням прикладів. Створено програмний засіб для генерації та реєстрації ключів програмного додатку. Розроблено рекомендації по захисту інформації від нелегального копіювання, котрі включають згадані рішення захисту та поєднують їх в єдину модель захисту для найбільшої ефективності.

Галузь використання – інформаційна безпека.

Розділ 1. ТЕХНІКИ ТА МЕХАНІЗМИ НЕЛЕГАЛЬНОГО КОПІЮВАННЯ ІНФОРМАЦІЇ

1.1. Злом програмного забезпечення

Злом програмного забезпечення - це модифікація програмного забезпечення для видалення або вимкнення функцій, які особа, яка зламує програмне забезпечення, вважає небажаними, особливо функції захисту від копіювання (включаючи захист від маніпуляцій із програмним забезпеченням, серійний номер, апаратний ключ, перевірку дати та перевірку диска) або програмне забезпечення такі неприємності, як Smart Screen та рекламне ПЗ.

Злом програмного забезпечення відноситься до засобів досягнення доступу, наприклад, викраденого серійного номера або інструменту. Деякі з цих інструментів називаються Keygen, Patch або Loader. Keygen - це ручний генератор серійних номерів продукту, який часто пропонує можливість генерувати робочі серійні номери на своє ім’я. Patch - це невелика комп'ютерна програма, яка змінює машинний код іншої програми. Patch надає перевагу для зломщика, який не включає великий виконаний файл у випуск, коли змінюється лише кілька байтів. Loader змінює процес запуску програми і не знімає захист, а обходить його. Відомий приклад Loader’a - це тренер, який використовується для обману в іграх. Fairlight зазначав в одному зі своїх файлів .nfo, що подібні дірки в коді заборонені для випуску ігор Warez.

Поширення зламаних копій є незаконним у більшості країн. Були судові позови через програмне забезпечення для злому коду. Законним є використання зламаного програмного забезпечення за певних обставин. Навчальні ресурси для зворотної інженерії та злому програмного забезпечення є законними та доступними у формі програм Crackme.

Одним з основних факторів, що сприяють широко розповсюдженому характеру злому, є простота перевірки та модифікації, що забезпечується поточним налагодженням програмного забезпечення та засобів редагування. За допомогою редакторів та складних відлагоджувачів зломщики можуть сконцентруватися на конкретні частини програми та застосувати до них

модифікації. Однак захисні механізми намагаються протистояти цій формі піратства.

1.1.1. Методи

Найпоширенішим програмним зламом є модифікація двійкового файлу програми, щоб спричинити або запобігти виконання певної ключової гілки у виконанні програми. Це досягається шляхом зворотної інженерії скомпільованого програмного коду за допомогою налагоджувача, такого як SoftICE, x64dbg, OllyDbg, GDB або MacsBug, поки зловмисник програмного забезпечення не досягне підпрограми, що містить основний метод захисту програмного забезпечення (або шляхом дизасемблювання виконуваного файлу з програми). Потім двійковий файл модифікується за допомогою налагоджувача, шістнадцяткового редактора або монітора так, що замінює попередній код на інший або код NOP, в наслідок чого, модифікований код або завжди виконуватиме певну підпрограму, або пропускатиме її. Майже всі поширені програмні злами є різновидом цього типу. Розробники програмного забезпечення постійно розробляють такі методи, як затухання коду, шифрування та самомодифікуючий код, щоб ускладнити можливість сторонніх модифікацій. Навіть за умови вжиття цих заходів розробники зі зломом програмного забезпечення. Це пов’язано з тим, що професіонали дуже часто публічно публікують зламаний EXE або Retrium Installer для загальнодоступного завантаження, усуваючи необхідність для недосвідчених користувачів самостійно зламати програмне забезпечення.

Конкретним прикладом цієї техніки є злом програмного забезпечення, який видаляє термін придатності із обмеженого в часі пробного періоду програми. Такі зломи - це, як правило, програми, які змінюють виконувану програму, а іноді і .dll або .so. Подібні зломи доступні для програмного забезпечення, яке вимагає апаратного ключа. Компанія може також порушити

ліцензійне погодження на рахунок копіювання програм, які вони придбали законно, але мають ліцензію на певне обладнання, так що немає ризику простою через несправність обладнання (і, звичайно, не потрібно обмежуватись запуском програмного забезпечення лише для придбаного обладнання).

Іншим методом є використання спеціального програмного забезпечення, такого як CloneCD, для сканування на предмет використання захисту від копіювання в програмі. Після виявлення програмного забезпечення, що використовується для захисту програми, може бути використаний інший інструмент для зняття такого типу захисту з програмного забезпечення на CD або DVD. Це може дозволити іншій програмі, такі як Alcohol 120%, CloneDVD, Game Jackal або Daemon Tools, скопіювати захищене програмне забезпечення на жорсткий диск користувача. Популярні комерційні програми захисту від копіювання, які можна сканувати, включають SafeDisc та StarForce.

В інших випадках є можливість декомпілювати програму, щоб отримати доступ до вихідного коду або коду на рівні, вищому за машинний код. Це часто можливо з мовами скриптів та мовами, що використовують компіляцію JIT. Прикладом є злом (або налагодження) на платформі .NET, де можна розглянути можливість маніпулювання CIL для досягнення своїх потреб. Двійковий код на Java також працює подібним чином, коли існує проміжна мова перед компіляцією програми для запуску на машинному коді, що залежить від платформи.

Удосконалена зворотна інженерія таких засобів захисту, як SecuROM, SafeDisc, StarForce або Denuvo, вимагає злому чи багатьох зломщиків, щоб витратити набагато більше часу на вивчення захисту, врешті-решт знаходячи кожен недолік коду захисту, а потім кодуючи власні інструменти для "розгортання" захист автоматично від виконуваних (.EXE) та бібліотечних (.DLL) файлів.

В Інтернеті є ряд веб-сайтів, які дозволяють користувачам завантажувати зломи, створені групами warez, для популярних ігор та додатків (хоча існує загроза придбання шкідливого програмного забезпечення, яке іноді

поширюється на таких сайтах). Хоча ці зломи використовуються легальними покупцями програмного забезпечення, їх також можуть використовувати люди, які завантажили або іншим чином отримали несанкціоновані копії (часто через мережі Peer-to-peer).

Багато комерційних програм, які можна завантажити з Інтернету, мають пробний період (зазвичай 30 днів), і їх потрібно зареєструвати (тобто придбати) після закінчення пробного терміну, якщо користувач хоче продовжувати користуватися програмою. Для скидання пробного періоду записи реєстру та / або приховані файли, що містять інформацію про пробний період, змінюються та / або видаляються. З цією метою зломщики розробляють «Trial Reset» для певної програми, а іноді і для групи програм того самого виробника.

Метод скидання пробного періоду створений для уникнення обмежень програмного забезпечення протягом пробного періоду (наприклад, деякі функції доступні лише в зареєстрованій версії; зображення / відео / друковані копії, створені за допомогою програми, отримують водяний знак; програма працює лише 10 – 20 хвилин, а потім закривається автоматично). Деякі програми мають необмежений пробний період, але обмежені до моменту їх реєстрації.

1.2. Софтліфтинг

Софтліфтинг - поширений тип піратства програмного забезпечення, при якому встановлена або скопійована законно ліцензована програма порушує ліцензійну угоду. На відміну від комерційного піратства, метою софтліфтингу є надання програми багатьом користувачам, а не продаж копій для отримання прибутку. Софтліфтинг також відомий як піратство кінцевих користувачів.

Найпоширенішими прикладами софтліфтингу є:

- Надання програмного забезпечення більшій кількості корпоративних користувачів, ніж це передбачено ліцензійною угодою.
- Встановлення програмного забезпечення, ліцензованого для організації, на домашніх комп’ютерах.
- Спільний доступ до програмного забезпечення серед друзів.

Софтліфтинг є найпоширенішим видом піратства програмного забезпечення на підприємствах. За даними Асоціації програмної та інформаційної індустрії (SIIA), торгової асоціації для галузі програмного забезпечення та цифрового контенту, кожна четверта копія програмного забезпечення для бізнесу в США використовується незаконно.

1.3. Завантаження жорсткого диска

Завантаження жорсткого диска - це комерційне піратство програмного забезпечення. Компанії, які роблять різні збірки ПК, купують легальну копію програмного забезпечення, але потім відтворюють, копіюють або встановлюють програмне забезпечення на жорсткі диски комп’ютерів. Потім продається комп’ютер із жорстким диском, що містить попередньо встановлене програмне забезпечення. Цей метод піратства відомий як завантаження жорсткого диска.

Зазвичай комп'ютери продаються в магазинах перепродажу ПК. Для цього типу піратів часто використовують старі операційні системи. Торговельні посередники включатимуть додаткову плату за нелегітимне програмне забезпечення, хоча фактична плата не задокументована на квитанції або купюрі. Таким чином, клієнт не знає про будь-які піратські дії, але торговельний посередник все одно отримує прибуток від своєї первісної несанкціонованої покупки піратського програмного забезпечення.

Інші типи електронних підробок програмних засобів також є піратськими. І клієнт може не усвідомлювати цього, поки програмне забезпечення не містить

певної інформації або стане повністю непридатним для використання. Крім того, деякі перепродажні комп'ютерні організації сміливо завантажують піратське програмне забезпечення на комп'ютери замовника. У цих випадках клієнти знають, що отримують несанкціоноване програмне забезпечення. Наприклад, три філіппінські компанії з перепродажу комп'ютерів стали об'єктом розслідування, розпочатого Антипіратською групою Піліпінаса (PAPT). Розташовані в Манілі, компанії використовували завантаження піратського програмного забезпечення як маркетингову стратегію. Можливо, вони залучили клієнтів, але вони також залучили PAPT. Отримуючи фальшиве програмне забезпечення, покупці та продавці порушують законодавство про авторське право. Досвідчені споживачі повинні переконатись, що на придбаних комп’ютерах немає попередньо встановлених програм, особливо коли йдеться про торгівельних посередників.

1.4. Інтернет-піратство

Інтернет-піратство - це практика завантаження та розповсюдження вмісту, захищеного авторським правом, без дозволу, наприклад музики чи програмного забезпечення. Принцип піратства був вигаданий ще до створення Інтернету, але його популярність в Інтернеті зростає з кожним роком. Незважаючи на свою явну незаконність у багатьох розвинених країнах, Інтернет-піратство все ще широко практикується завдяки легкості, з якою це можна зробити.

1.4.1 Переваги

Інтернет-піратство призвело до вдосконалення технології обміну файлами, що покращило розподіл інформації в цілому. Крім того, піратські

громади, як правило, добре моделюють ринкові тенденції, оскільки члени цих спільнот зазвичай роблять подібні копії на широкий спектр людей. Піратство також може призвести до того, що підприємства розроблятимуть нові моделі, які краще відповідають поточному ринку. Вважається, що піратство в Інтернеті може допомогти запобігти інвестуванню бізнесу в непотрібні маркетингові кампанії. На додаток до допомоги бізнесу у проведенні досліджень, деякі організації можуть краще обслуговувати лише своїх найцінніших та законних споживачів або тих, хто купує законні копії своєї продукції. Оскільки, як очікується, піратські копії програмного забезпечення приваблюють клієнтів, чутливих до ціни, компаніям може бути не найкращим інтересом вступати в сторонні цінові війни зі своїми конкурентами або вкладати значні кошти в анти піратські кампанії, щоб завоювати цільових клієнтів.

Незважаючи на дискурс про цифрову загрозу піратства, було показано, що інновації та створення нових творів процвітають як ніколи в Інтернеті, незважаючи на дискурс про цифрові загрози. Піратство також принесло користь користувачам у країнах, де програми або недоступні, або затримуються. У випадку з програмою ABC Lost страх перед піратством в європейських та країнах Близького Сходу підштовхнув мережу до пришвидшення розповсюдження серед цих країн, в результаті чого стане доступним у цих країнах через 24/48 годин після виходу оригіналу.

Закони про захист авторських прав чіткі, а покарання суворе. Поширеність піратства перед цими потенційними покараннями пов’язана з тим, що люди не вважають піратство неприйнятним, не кажучи вже про незаконне, натомість розглядають його як етично прийнятне, оскільки основне здійснення піратства полягає в тому, що воно створює копію файлу, таким чином нічого матеріального не забирається у первісного власника. Крім того, незважаючи на масштабну сферу копіювання та обміну цифровим вмістом, споживачі, які займаються піратством, більше готові платити за легальний вміст, коли вміст є зручним для споживачів. Етичні та моральні схильності людини та судження, які вона використовує для прийняття рішень, можуть свідчити про

узгодженість різних етичних дилем, а також вказувати на їх ймовірність в піратському програмному забезпеченні.

І навпаки, ті самі особи зазначили, що поширеність піратства зумовлена нездатністю галузі задовольнити потреби споживачів. Багато хто називає причиною піратства незадовільну галузеву практику, таку як нав'язливе DRM у платному програмному забезпеченні та завищені медіа. Цифрове піратство створює значну загрозу для розвитку індустрії програмного забезпечення та зростання індустрії цифрових медіа, воно протягом останнього десятиліття викликало значний інтерес для дослідників та практиків. В контексті Індонезії моральна справедливість негативно вплинула на поведінку цифрового піратства. Тому зусилля щодо зменшення піратства були зосереджені на висвітленні важливості справедливості. Вивчення причин та наслідків цифрового піратства є одним із способів оцінки етики того, як наше суспільство споживає та поширює засоби масової інформації одне до одного. Широкі дослідження з вивчення цифрового піратства можуть допомогти краще зрозуміти психологію та етику цифрового світу. Одним із дослідницьких підходів, який створив теоретичну базу для вивчення програмного піратства, було розміщення незаконного копіювання програмного забезпечення в області етично прийнятих рішень, яке передбачає, що користувач повинен мати можливість визнати програмне піратство моральною проблемою. Людина, яка не може визнати моральну проблему, не зможе використовувати схеми прийняття моральних рішень. Є дані, що багато людей не сприймають програмне піратство як етичну проблему. Результати досліджень свідчать про те, що особиста мораль знижує цифрове піратство переважно на першій фазі, тоді як нейтралізація використовується людьми для підтримки своєї поведінки на інших етапах.

1.5. Підробка програмного забезпечення

Підробка програмного забезпечення - це незаконний спосіб копіювання програмних продуктів комп'ютера. Підроблене програмне забезпечення також порушує авторські права програмного забезпечення. Піратське програмне забезпечення може виглядати автентично і дуже схоже на оригінальний продукт. Копійоване програмне забезпечення може включати несанкціоноване копіювання та спільний доступ на DVD або CD і може бути зроблене за допомогою записувача компакт-дисків. Через недороге копійоване програмне забезпечення люди у всьому світі частіше купують піратський продукт, ніж оригінал.

Кілька поширених прикладів з підробкою програмного забезпечення:

- Розповсюдження з інших регіонів - копіювальні апарати можуть викрадати програмне забезпечення з різних регіонів і продавати їх деінде. Інший спосіб - покупець може дешево придбати підроблене програмне забезпечення, а потім продати його за вищу вартість, щоб отримати прибуток. Наприклад, подібні нелегальні операції можна знайти в країнах Азії;
- Веб-сайти аукціони - підроблене програмне забезпечення можна знайти на веб-сайтах аукціонах. Ціни зазвичай нижчі за оригінальний товар, який можна знайти в магазинах;
- Повідомлення електронної пошти зі спамом - люди отримують повідомлення про недороге програмне забезпечення. Повідомлення містять посилання на сайт, де покупець може придбати скопійоване програмне забезпечення. Однак інформація, яку покупець надає через сайт, швидше за все, стане жертвою крадіжки кредитної картки.

Багато країн мають закони про авторське право. Однак закони про авторське право не існують для інших країн. Продавець і покупець повинні надати ліцензійну угоду з кінцевим користувачем (EULA), щоб мати законне

право власності на програмне забезпечення. EULA - це контракт між користувачем та виробником програми чи програмного забезпечення. Він містить офіційну письмову інформацію, таку як обмеження та спосіб використання програмного забезпечення на законних підставах. Як правило, все програмне забезпечення має письмову заяву, яка забороняє користувачеві надавати програмне забезпечення іншому користувачеві. Основною метою ліцензійного договору є захист як виробника, так і користувача від незаконних дій або відповідальності. Більш коротка назва ліцензійного договору - це ліцензія користувача або програмного забезпечення.

Закон про Уніфікований закон про єдині інформаційні операції (UCITA) передбачає, що виробники програмного забезпечення мають право закривати будь-яку незаконну діяльність скопійованого програмного забезпечення. Цей закон був проблематичним, оскільки він не має достатньої інформації для висвітлення програмних операцій. Люди, які виступають проти UCITA, вважають, що це більше сприяє виробникам програмного забезпечення і обмежує захист покупців.

Поширення підробленого програмного забезпечення негативно впливає на світову економіку. Наприклад, люди, які беруть участь у діях з підробкою програмного забезпечення, можуть потрапити в біду, оскільки вони або продають, або купують щось незаконно. Інший приклад - програмні компанії мали б проблеми із створенням бізнесу, оскільки їм було б складно продати власне програмне забезпечення. Дослідження показало, якщо уникати підробленого програмного забезпечення, воно може створити більше можливостей для працевлаштування, а також економічне зростання.

Microsoft має інструмент для ідентифікації скопійованого програмного забезпечення. Користувач може перевірити справжність через веб-сайт справжнього програмного забезпечення Microsoft. Веб-сайт створений для запобігання незаконним копіюванням фальшивомонетниками та допомагає корпорації Майкрософт захищати свою інтелектуальну власність. Важливо купувати програмне забезпечення у чесних продавців. Завжди потрібно

переконуватися, що все програмне та апаратне забезпечення включено у комплект і що нічого не бракує.

1.6. Зворотна інженерія

Зворотна інженерія - це процес розкриття принципів, що лежать в основі апаратного чи програмного забезпечення, таких як його архітектура та внутрішня структура.

Зворотна інженерія застосовується в галузях обчислювальної техніки, машинобудування, проектування, електронної техніки, програмної інженерії, хімічної інженерії та системної біології

Очевидно, що якщо у вас є документація, весь процес стає набагато простішим. Але часто трапляється, що документації немає, і вам потрібно знайти інший спосіб дізнатись, як працює якийсь програмний продукт.

Існує багато застосувань зворотного проектування в галузі інформатики, зокрема:

- Дослідження протоколів мережевого зв'язку;
- Пошук алгоритмів, що використовуються в шкідливих програмах, таких як комп’ютерні віруси, троянські програми, програми-вимагателі тощо;
- Дослідження формату файлу, що використовується для зберігання будь-якого виду інформації, наприклад баз даних електронної пошти та образів дисків;
- Перевірка здатності власного програмного забезпечення протистояти зворотному проектуванню;
- Покращення сумісності програмного забезпечення з платформами та сторонніми програмами.

Законність зворотного проектування залежить від його призначення та способу використання програмного забезпечення. Усі цілі, згадані вище, є цілком законними, якщо ви отримали копію програмного забезпечення законно.

Але якщо ви маєте намір, наприклад, перепроектувати певну функцію додатку, а потім застосувати її в іншій програмі, ви, швидше за все, зіткнетеся з проблемами.

Що стосується юридичної документації, зворотне проектування часто заборонено ліцензійними угодами з кінцевими користувачами (EULA). Але Закон про захист авторських прав у цифрову епоху США вказує, що скасування програми назад є законним, якщо це зроблено для поліпшення сумісності з іншими продуктами.

Щоб запустити програмне забезпечення зворотної інженерії, вам потрібно:

- знання в галузі, де ви хочете застосувати зворотну інженерію;
- інструменти, які дозволять застосувати ваші знання при спробі розібрати програмне забезпечення.

Розглянемо загальний приклад, який не пов’язаний із програмним забезпеченням. Скажімо, у вас є годинник, і ви хочете з’ясувати, механічний він, кварцовий чи автоматичний.

Знання в цій галузі означає, що ви повинні знати, що існує три типи годинників. Крім того, ви повинні знати, що якщо є акумулятор, він знаходиться всередині годинника, і ви можете побачити його, якщо його відкрити. Ви також повинні мати базові знання про внутрішню структуру годинника, як виглядає акумулятор та які інструменти потрібні для відкриття корпусу годинника. Наявність інструментів для застосування ваших знань означає, що вам потрібно мати викрутку або інший спеціальний інструмент, який дасть вам можливість відкрити годинник.

Подібно до того, як зворотна інженерія годинників вимагає певного набору навичок та інструментів, програмне забезпечення для зворотної інженерії вимагає власних знань та інструментів для конкретної галузі. Розробники шкідливих програм часто використовують методи зворотної інженерії для пошуку вразливостей в операційній системі для подальшого створення комп’ютерного вірусу, який може використовувати системні

вразливості. Зворотна інженерія також використовується в криптоаналізі для пошуку вразливостей в шифрі заміщення, алгоритмі симетричного ключа або криптографії відкритого ключа.

Існують і інші способи зворотної інженерії:

- Взаємозв'язок. Зворотна інженерія може бути використана, коли системі потрібно взаємодіяти з іншою системою і як обидві системи повинні домовлятися, слід встановити. Такі вимоги, як правило, існують щодо сумісності;
- Військове чи комерційне шпигунство. Знання про останні дослідження ворога чи конкурента шляхом викрадення чи захоплення прототипу та його демонтажу може призвести до розробки подібного продукту або кращого протидії проти нього;
- Застарілість. Інтегральні схеми часто розробляються на власних системах і будуються на виробничих лініях, які застарівають лише за кілька років. Коли системи, що використовують ці деталі, більше не можуть підтримуватися, оскільки деталі більше не виготовляються, єдиним способом включити функціонал у нову технологію є зворотне проектування існуючого чіпа, а потім його перепроектування за допомогою новіших інструментів, використовуючи розуміння, яке було отримано у вигляді прикладу. Ще однією проблемою застарівання, яку можна вирішити за допомогою зворотного проектування, є необхідність підтримувати (технічне обслуговування та постачання для постійної роботи) існуючі застарілі пристрої, які більше не підтримуються їх оригінальним виробником обладнання. Проблема особливо критична у військових операціях;
- Аналіз безпеки продукції. Аналіз вивчає, як працює виріб, визначаючи технічні характеристики його компонентів та оцінюючи витрати, виявляючи потенційні порушення патенту. Також частиною аналізу безпеки продукції є отримання конфіденційних даних шляхом дизасемблювання та аналізу конструкції системного компонента. Іншим наміром може бути скасування захисту від копіювання або обхід обмежень доступу.

- Конкуренто-технічний інтелект. Існує для того, щоб розуміти насправді робить конкурент, а не те, що він говорить, що робить.
- Заощадження грошей. З’ясування того, що може зробити електроніка, може позбавити користувача придбання окремого продукту.
- Перепрофілювання. Застарілі об’єкти використовуються повторно, але корисно.
- Дизайн. Виробничі та дизайнерські компанії застосовували технологію Reverse Engineering для практично-виробничого процесу на основі ремесл. Компанії можуть працювати над "історичними" виробничими колекціями за допомогою 3D-сканування, 3D-моделювання та дизайну. У 2013 році італійські виробники Baldi та Savio Firmino спільно з Університетом Флоренції оптимізували свої інновації, дизайн та виробничі процеси.

1.7. Втручання

Втручання визначається як інша форма нападу на інтелектуальну власність. Фальсифікація часто поєднується з іншими формами піратства, як, наприклад, зворотна інженерія. Однак фальсифікація часто відбувається незалежно від форм атаки, і прикладом цього може бути зловмисник, який маніпулює програмним забезпеченням заради отримання фінансової вигоди. Втручання також можна визначити як небажану модифікацію програмного забезпечення, що включає як автоматизовані зміни, внесені вірусами, так і ручні зміни коду, внесені кінцевими користувачами.

Втручання може бути, наприклад, вилученням або зміною інформації, зашифрованої в програмному забезпеченні. Доступ та підробка такої інформації може призвести до значних фінансових втрат для власника інтелектуальної власності.

1.8. Апаратні методи для злому захисту від копіювання

Існує три типові апаратні методи для захисту від копіювання: аналіз слідів виконання програмного забезпечення, пристрої підміни чіпів модулів та емулятор машини.

При аналізі слідів зловмисник може збирати інформацію про цільове програмне забезпечення шляхом реєстрації та аналізу програмних слідів. Багато примітивних програм дуже передбачувані навіть у зашифрованому вигляді; наприклад, умовні гілки легко ідентифікувати та простежити навіть після шифрування. Крім того, технічно кваліфіковані користувачі часто можуть створювати власні пристрої трасування з дешевих та легко доступних компонентів.

Іншою формою апаратних прийомів для захисту від копіювання є пристрої підробки мод-чіпів. Мод-чіпи можна використовувати для запису, відтворення пам'яті та транзакцій шини. Крім того, їх можна використовувати для викрадення сигналу іншого пристрою.

Третя форма апаратних методів скасування захисту від копіювання - використання емуляторів. Якщо програмне забезпечення можна виконати на машинному емуляторі без дозволу, захист від копіювання програмного забезпечення може вважатися порушеним. Дуже складно боротися з цим видом скасування захисту від копіювання, не покладаючись на шифрування. Однак так можна запобігти емулятору машини порушити захист прав програмного забезпечення, захищаючи конфіденційність програмного забезпечення.

1.8.1. Аналіз слідів виконання програмного забезпечення

У програмній інженерії трасування передбачає спеціалізоване використання журналів для запису інформації про виконання програми. Ця інформація зазвичай використовується програмістами для налагодження, а

також, залежно від типу та деталізації інформації, що міститься в журналі трасування, досвідченими системними адміністраторами або персоналом технічної підтримки та засобами моніторингу програмного забезпечення для діагностики загальних проблем із програмним забезпеченням. Відстеження - це наскрізна проблема.

Не завжди існує чітка різниця між трасуванням та іншими формами ведення журналу, за винятком того, що термін трасування майже ніколи не застосовується до журналювання, що є функціональною вимогою програми (отже, виключаючи реєстрацію даних із зовнішнього джерела). Журнали, що фіксують використання програми (наприклад, журнал сервера) або події операційної системи, що в першу чергу цікавлять системного адміністратора, потрапляють у термінологічну сіру зону.

Труднощі у чіткому розрізненні журналу подій та відстеження програмного забезпечення виникають внаслідок того, що одні й ті самі технології використовуються для обох, а також тому, що багато критеріїв, що розрізняють ці дві події, є суцільними, а не дискретними. У таблиці 1.1 перелічені деякі важливі, але аж ніяк не точні та універсальні відмінності, які розробники використовують для вибору технологій для кожної мети та які спрямовують на окремий розвиток нових технологій у кожній галузі:

Таблиця. 1.1

Порівняння відмінностей запису дій та відстеження програмного забезпечення

Запис подій	Відстеження програмного забезпечення
Використовується переважно системними адміністраторами	Використовується переважно розробниками

Продовження табл. 1.1

Запис інформації "високого рівня" (наприклад, невдале встановлення програми)	Заносить інформацію про "низький рівень" (наприклад, виняток)
Не повинно бути занадто "галасливим" (містить багато повторюваних подій або інформація не корисна для цільової аудиторії)	Може бути галасливим
Формат вихідних даних, заснований на стандартах, часто є бажаним, іноді навіть необхідним	Мало обмежень на вихідний формат
Повідомлення журналу подій часто локалізуються	Локалізація рідко викликає занепокоєння
Додавання нових типів подій, а також нових повідомлень про події не обов'язково має бути гнучким	Додавання нових повідомлень про трасування повинно бути гнучким

Запис подій. Запис подій надає системним адміністраторам інформацію, корисну для діагностики та аудиту. Різні класи подій, які реєструватимуться, а також ті деталі, які з’являться у повідомленнях про події, часто розглядаються на початку циклу розробки. Багато технологій реєстрації подій дозволяють або навіть вимагають присвоєння кожному класу подій унікального «коду», який використовується програмним забезпеченням для реєстрації подій або окремим засобом перегляду (наприклад, Event Viewer) для форматування та виведення зручного для читання повідомлення. Це полегшує локалізацію і дозволяє системним адміністраторам простіше отримувати інформацію про проблеми, що виникають.

Оскільки реєстрація подій використовується для реєстрації інформації високого рівня (часто інформації про помилки), продуктивність реалізації журналювання часто менш важлива.

Особливе занепокоєння викликає запобігання тому, щоб повторювані події не записувались "занадто часто", здійснюється за допомогою регулювання подій.

1.8.2. Пристрої підміни чіпів модулів

Багато мікроконтролерів використовуються в сучасному обладнанні та електронних пристроях. Деякі з них використовуються аматорами для побудови невеликих пристроїв для розваги, інші використовуються невеликими компаніями в контрольному, вимірювальному чи іншому обладнанні, інші використовуються для серйозних застосувань військовими, службами безпеки, банками, медичними службами тощо. Кожен мікроконтролер виконує алгоритм або програма, завантажені в його пам’ять. Зазвичай цей алгоритм пишеться в Assembler (навіть якщо ви пишете програму на C, вона буде перекладена в Assembler під час компіляції); рідко алгоритм пишеться на Basic або Java.

Якщо писати програму для мікроконтролера, розробник зацікавлений у тому, щоб його робота була захищена від несанкціонованого доступу або копіювання, тому потрібно контролювати розподіл своїх пристроїв. З цією метою виробники мікроконтролерів розробили спеціальні функції, які за вибором дозволяють авторам програмного забезпечення перешкоджати завантаженню їхньої програми зі свого мікроконтролера, якщо її активовано. Ця функція існує для захисту від копіювання або блокування. Кожен мікроконтролер повинен бути запрограмований перед використанням. Існують різні методи, щоб це зробити залежно від виробника та типу мікроконтролера. Для цілей оцінки існують перепрограмовані версії мікроконтролерів, для виробництва в невеликих кількостях існують разові програмовані версії (OTP),

що дешевше перепрограмованої, а для великої кількості є заводські запрограмовані версії, які є дуже дешевими, але вам доведеться придбати не менше 1000 копій. Після того, як програма для мікроконтролера буде написана та успішно скомпільована, її слід завантажити у відповідну інтегральну схему мікроконтролера. Для цього вам доведеться використовувати спеціальний апаратний пристрій, який називається «програміст». Для більшості мікроконтролерів цей пристрій може бути дуже простим, дешевим і складатися з адаптера живлення, декількох транзисторів, декількох резисторів та роз'єму до RS232 або паралельного порту. Для інших мікроконтролерів доводиться використовувати спеціальні модулі програмістів, що розповсюджуються лише виробниками, але ці мікроконтролери не популярні. Звичайно, якщо ви хочете, щоб ваш пристрій працював належним чином роками (особливо для OTP-версій мікроконтролерів), було б краще використовувати промислові модулі програмістів, схвалені більшістю виробників. Ви можете знайти всю необхідну інформацію про ці пристрої на веб-сайтах виробників в Інтернеті.

Неінвазійні атаки. Найбільш широко використовувані неінвазійні атаки включають відтворення напруги живлення та тактового сигналу. Атаки під напругою та перенапругою можуть бути використані для вимкнення схеми захисту або змушення процесора до неправильної роботи. З цих причин деякі захисні процесори мають схему виявлення напруги, але, як правило, ця схема не реагує на перехідні процеси. Тож швидкі сигнали різного роду можуть скинути захист, не знищуючи захищену інформацію.

В деяких процесорах також можуть використовуватися перехідні процеси живлення та синхронізації, що впливають на декодування та виконання окремих інструкцій. Кожен транзистор і шляхи його з'єднання діють як елемент RC з характерною затримкою часу; максимальна корисна тактова частота процесора визначається максимальною затримкою серед його елементів. Аналогічно, кожен тригер має характерне часове вікно (кілька пікосекунд), протягом якого він відбирає свою вхідну напругу і відповідно змінює свій вихід. Це вікно може знаходитись де завгодно в межах зазначеного циклу

налаштування тригера, але цілком фіксовано для окремого пристрою за заданої напруги та температури. Отже, якщо ми застосуємо тактовий збій (тактовий імпульс набагато коротший за звичайний) або збій живлення (швидкий перехідний режим напруги живлення), це вплине лише на деякі транзистори в мікросхемі. Змінюючи параметри, центральний процесор може бути використаний для виконання ряду абсолютно різних неправильних інструкцій, іноді включаючи інструкції, які навіть не підтримуються кодом. Хоча ми не знаємо заздалегідь, який збій спричинить яку неправильну інструкцію в чіпі, провести системний пошук може бути досить просто.

Іншим можливим способом атаки є поточний аналіз. Використовуючи резистор 10 - 15 Ом в блоці живлення, ми можемо виміряти за допомогою аналого-цифрового перетворювача коливання струму, споживаного картою. Переважно, щоб запис проводився з щонайменше 12-бітовою роздільною здатністю, а частота дискретизації повинна бути цілим числом, кратним тактовій частоті карти.

Драйвери на шині адреси та даних часто складаються з десятка паралельних інверторів на біт, кожен з яких веде велике ємнісне навантаження. Вони викликають значне коротке замикання джерела живлення під час будь-якого переходу. Зміна однієї лінії шини від 0 до 1 або навпаки може сприяти порядку 0,5 - 1 мА загального струму в потрібний час після фронту тактової частоти, так що 12-бітового АЦП достатньо для оцінки кількості бітів шини, які змінюються за раз. Операції запису SRAM часто генерують найсильніші сигнали. За допомогою усереднення поточних вимірювань багатьох повторюваних однакових транзакцій ми можемо навіть ідентифікувати менші сигнали, які не передаються по шині. Такі сигнали, як стан розрядних бітів, представляють особливий інтерес, оскільки багато алгоритмів планування криптографічного ключа використовують операції зсуву, які виділяють окремі бітові ключі у прапорі перенесення. Навіть якщо зміни в розряді стану неможливо виміряти безпосередньо, вони часто спричиняють зміни в

послідовності команд або виконанні коду, що потім спричиняє чітку зміну споживання енергії.

Різні інструкції викликають різний рівень активності в декодері інструкцій та арифметичних одиницях і їх часто можна досить чітко розрізнити, так що частини алгоритмів можна реконструювати. Різні блоки процесора мають перехідні процеси в різний час щодо тактових частот і можуть бути розділені у високочастотних вимірах.

Інша можлива загроза захищеним пристроям - це збереження даних. Це здатність енергонезалежної пам'яті зберігати інформацію, що зберігається в ній протягом деякого періоду часу після відключення живлення. Статична оперативна пам'ять містила одну і ту ж клавішу протягом тривалого періоду часу і могла розкрити її при наступному увімкненні. Інший можливий спосіб - "заморозити" стан комірки пам'яті, застосувати до пристрою низьку температуру. У цьому випадку статична оперативна пам’ять може зберігати інформацію протягом декількох хвилин при -20ºС або навіть годин при нижчій температурі.

Інвазійні атаки. Незважаючи на складність інвазійних атак, деякі з них можна було зробити без використання дорогого лабораторного обладнання. Бюджетні зловмисники, швидше за все, отримають дешевше рішення на ринку секонд-хендів для напівпровідникового випробувального обладнання. З терпінням і майстерністю не повинно бути надто складно зібрати всі необхідні інструменти навіть на суму менше десяти тисяч доларів США, придбавши мікроскоп, який вже використовували, та використовуючи власноруч розроблені мікропозиціонери. Лазер не є важливим для перших результатів, оскільки вібрації в зондуючій голці також можуть використовуватися для пробивання отворів у пасивацію.

Інвазійні атаки починаються з видалення пакета чіпів. Пластик над чіпом можна зняти ножем. Епоксидна смола навколо стружки може бути видалена за допомогою димної азотної кислоти. Гаряча димляча азотна кислота розчиняє упаковку, не впливаючи на стружку. Процедуру бажано проводити в дуже

сухих умовах, оскільки присутність води може призвести до корозії оголених алюмінієвих з'єднань. Потім стружку промивають ацетоном в ультразвуковій ванні, а потім, додатково, короткою ванною в де іонізованій воді та ізопропанолі. Після цього мікросхему можна було вклеїти в тестовий пакет і скріпити вручну. Маючи достатній досвід можна буде видалити епоксидну смолу, не руйнуючи проводів, що їх скріплюють та контактів смарт-карт.

Після відкриття мікросхеми можна проводити зондування або модифікуючі атаки. Найважливішим інструментом для інвазійних атак є робоча станція з мікрозондацією. Його основним компонентом є спеціальний оптичний мікроскоп з робочою відстанню не менше 8 мм між поверхнею стружки та лінзою об'єктива. На стабільній платформі навколо гнізда для тестового пакету встановлюється кілька мікропозиціонерів, які дозволяють нам переміщати ручку зонда з точністю до субмікрометра по поверхні чіпа. На цьому місці ми встановлюємо голку зонда. Ці еластичні зондові волоски дозволяють нам встановити електричний контакт з вбудованими шинними лініями, не пошкоджуючи їх.

На упакованому чіпі алюмінієві з'єднувальні лінії верхнього шару все ще покриті пасиваційним шаром (зазвичай оксидом кремнію або нітридом), який захищає чіп від навколишнього середовища та міграції іонів. Крім цього, ми можемо також знайти шар поліаміду, який не був повністю видалений HNO3, але який можна розчинити етилендіаміном. Далі потрібно видалити шар пасивації, перш ніж зонди зможуть встановити контакт. Найзручніша техніка депасивації - використання лазерного різака. Ультрафіолетовий або зелений лазер встановлюється на порту камери мікроскопа і спрацьовує лазерні імпульси через мікроскоп на прямокутні ділянки мікросхеми з мікрометричною точністю. Ретельно дозовані лазерні спалахи видаляють плями шару пасивації. Отриманий отвір у шарі пасивації може бути зроблений настільки малим, що виставляється лише одна лінія шини. Це запобігає випадковому контакту з сусідніми лініями, а отвір також виконує роль стабілізатора положення зонда і робить його менш чутливим до вібрацій і перепадів температури.

Зазвичай непрактично читати інформацію, що зберігається на процесорі захисту, безпосередньо з кожної окремої комірки пам'яті, за винятком ПЗУ. Доступ до збережених даних повинен здійснюватися через шину пам'яті, де всі дані доступні в одному місці. Мікрозондація використовується для спостереження за всією шиною та запису значень у пам’яті під час їх отримання.

Важко одночасно спостерігати за всіма (зазвичай понад 20) даними та лініями шини. Щоб обійти цю проблему, можна використовувати різні методи. Наприклад, ми можемо повторити одну і ту ж операцію багато разів і використовувати лише два-чотири зонди для спостереження різних підмножин шинних ліній. Поки процесор виконує однакову послідовність доступу до пам'яті кожного разу, можна об'єднати записані сигнали підмножини шин в повну трасовану шини. Перекриття шинних ліній у різних записах допомагає їх синхронізувати до їх об’єднання.

Для того, щоб читати всі комірки пам'яті без допомоги програмного забезпечення картки, потрібно зловживати компонентом ЦП як лічильником адрес, щоб отримати доступ до всіх комірок пам'яті. Лічильник програм вже збільшується автоматично під час кожного циклу інструкцій і використовується для зчитування наступної адреси, що робить його ідеально придатним для використання в якості генератора послідовності адрес. Залишається лише заважати процесору виконувати інструкції про перехід, виклик або повернення, які могли б порушити лічильник програми в нормальній послідовності читання.

1.9. Висновки до першого розділу

Проаналізовано поширені технології та механізми більшості атак на конфіденційність інформації проти програмних забезпеченнях. Встановлено ефективність кожного окремого рішення та їх галузь застосування. Розглянуто

реалізацію розгортки окремих заходів захисту у мережі. Виокремлено позитивні та негативні аспекти роботи приладів, послуг.

Розділ 2. ІСНУЮЧІ МЕХАНІЗМИ ПРОТИДІЇ НЕЛЕГАЛЬНОМУ КОПІЮВАННЮ

2.1. Авторські права на програмне забезпечення

Авторське право на програмне забезпечення - це застосування законодавства про авторське право до машино-читаного програмного забезпечення. Хоча багато правових принципів та політичних дебатів щодо авторського права на програмне забезпечення мають близькі паралелі в інших сферах авторського права, існує низка відмінних питань, які виникають із програмним забезпеченням.

Авторські права на програмне забезпечення використовуються розробниками програмного забезпечення та компаніями-розробниками програмного забезпечення для запобігання несанкціонованого копіювання їх програмного забезпечення. Безкоштовні ліцензії з відкритим кодом також покладаються на законодавство про авторське право, щоб забезпечити виконання своїх умов. Наприклад, ліцензії copyleft покладають на ліцензіатів обов'язок ділитися своїми модифікаціями програми з користувачем або власником копії за певних обставин. Жодне таке мито не застосовувалося б, якби програмне забезпечення було у відкритому доступі.

Закон США про авторські права 1976 року загалом захищає оригінальні авторські твори, зафіксовані в будь-якому матеріальному виражальному середовищі, з якого вони можуть бути сприйняті, відтворені або іншим способом передані безпосередньо за допомогою машини чи пристрою. Закон США про авторські права захищає авторські твори, які належать до різних категорій. Ці категорії включають літературні твори (наприклад, книгу), музичні твори (наприклад, музичну партитуру), драматичні твори (наприклад,

виставу), кінофільми та інші аудіовізуальні твори (наприклад, фільм чи телевізійне шоу) та звукозаписи (наприклад, пісня). Поки твір є оригінальним, він потрапляє до однієї з прийнятних категорій і закріплюється на типі носія (папір, жорсткий диск, файл тощо), цей конкретний твір може мати право на захист авторських прав.

Однак сам факт того, що твір захищено авторським правом, не означає, що кожен елемент твору захищений. Швидше, Закон про авторське право встановлює певні обмеження щодо обсягу захисту авторських прав, доступних для оригінальних авторських творів. Одне з цих обмежень виражається у другому розділі Закону про авторське право, в якому зазначається, що будь-які ідеї, процедури, процеси, системи, методи роботи, концепції, принципи або відкриття, незалежно від того, в якій формі це описано, пояснено, проілюстровано , або втілені в такій роботі, не мають права на захист авторських прав.

У контексті програмного забезпечення програма, як правило, є реалізацією ідеї, яка виконується комп'ютером для вирішення конкретної проблеми. Сам вихідний код - це систематизований набір інструкцій для обробки комп’ютером для досягнення бажаного результату чи результату. Вихідний код - це набір інструкцій, який створить метод роботи, що використовується комп’ютером, створює процедури та процеси, які буде виконувати комп’ютер, а при виконанні комп’ютером створює систему для використання користувачем програми.

Закон про авторські права захищає комп’ютерні програми та визначає «комп’ютерну програму» як сукупність тверджень чи інструкцій, що використовуються прямо чи опосередковано в комп’ютері для досягнення певного результату. Оскільки Закон про авторське право визначає комп’ютерну програму як сукупність тверджень чи інструкцій, то для цілей авторського права набір заяв чи інструкцій, який є вихідним кодом програми, вважається літературним твором, що захищається авторським правом. Вихідний код можна вважати літературним твором, оскільки вихідний код виражається словами,

цифрами або іншими словесними чи цифровими символами чи знаками. Що стосується вимог до фіксованого середовища, то природа матеріальних об’єктів, таких як книги, періодичні видання, рукописи, фільми, стрічки чи диски, в яких втілено вихідний код, не має значення, якщо це відчутно. Оскільки вихідний код вважається літературним твором і може бути втілений у матеріальному носії, захист авторських прав може поширюватися і на комп'ютерні програми.

Хоча «комп’ютерні програми», визначені Законом про авторське право, можуть мати право на захист авторських прав, виняток щодо авторських прав щодо ідей та процедур, зазначених вище, все ще застосовується. Тоді виникає питання, чи комп’ютерна програма - це лише реалізація ідеї, яка, як ми знаємо, не підпадає під захист авторських прав, чи комп’ютерна програма є літературним твором, який підлягає захисту авторських прав? Оскільки Закон про авторське право не містить явних стандартів для відокремлення виразу комп’ютерної програми від її ідеї, потрібен спосіб відокремити вираз ідеї, захищеної авторським правом, від самої ідеї, що не захищається. Цей тест, який іноді називають дихотомією висловлення ідей, використовується для кращого розуміння того, які частини літературного твору, зокрема вихідний код, можуть мати право на захист авторських прав.

2.2. Патент на програмне забезпечення

Патент на програмне забезпечення - це право власності, яке захищає комп’ютерні програми або будь-яку роботу комп’ютера від комп’ютерних програм. Патент на програмне забезпечення вважається різновидом патента на корисність без справжнього юридичного визначення. Патенти на програмне забезпечення є предметом суперечок як у США, так і в усьому світі.

Патент на програмне забезпечення відрізняється від авторського права на програмне забезпечення. Обидва вони захищають товар, але авторське право

охоплює лише вираз ідеї. Наприклад, він може охоплювати лише точний написаний код програми. Патенти на програмне забезпечення та авторські права на програмне забезпечення є частиною законодавства про інтелектуальну власність.

Патент - це сукупність виключних прав, що надаються державою власнику патенту протягом обмеженого періоду часу, як правило, 20 років. Ці права надаються заявникам патентів в обмін на їх розкриття винаходів. Після видачі патенту в певній країні жодна особа не може робити, використовувати, продавати чи імпортувати / експортувати заявлений винахід у цій країні без дозволу власника патенту. Дозвіл, якщо він надається, зазвичай має форму ліцензії, умови якої встановлює власник патенту: він може бути безкоштовним, взамін за виплату роялті або одноразову плату.

Патенти мають територіальний характер. Щоб отримати патент, винахідники повинні подавати заявки на патенти в кожній країні, в якій вони хочуть патент. Наприклад, окремі заявки повинні подаватися в Японії, Китаї, США та Індії, якщо заявник бажає отримати патенти в цих країнах. Однак існують деякі регіональні відомства, такі як Європейське патентне відомство (ЄПВ), які діють як наднаціональні органи, що мають право видавати патенти, які потім можуть набути чинності в державах-членах, а також існує міжнародна процедура подання єдиного міжнародна заявка відповідно до Договору про патентну кооперацію (PCT), яка в подальшому може призвести до патентного захисту в більшості країн.

Ці різні країни та регіональні бюро мають різні стандарти щодо видачі патентів. Це особливо стосується програмного забезпечення або комп'ютерних винаходів, особливо там, де програмне забезпечення реалізує бізнес-метод.

Більшість країн встановлюють певні обмеження щодо патентування винаходів, що стосуються програмного забезпечення, але не існує єдиного юридичного визначення патенту на програмне забезпечення. Наприклад, патентне законодавство США виключає «абстрактні ідеї», і це було використано для відмови у деяких патентах, що стосуються програмного

забезпечення. У Європі «комп'ютерні програми як такі» виключаються з патентоспроможності, отже, політика Європейського патентного відомства полягає в тому, що програма для комп'ютера не є патентоспроможною, якщо вона не має потенціалу, щоб викликати «технічний ефект», який зараз розуміється як матеріальний ефект («перетворення природи»). Матеріальне право щодо патентоспроможності програмного забезпечення та винаходів, що реалізуються на комп’ютері, та судова практика, що тлумачить правові положення, відрізняються в різних юрисдикціях.

Патенти на програмне забезпечення за багатосторонніми договорами:

- патенти на програмне забезпечення відповідно до Угоди TRIPs;
- патенти на програмне забезпечення відповідно до Європейської патентної конвенції;
- комп’ютерні програми та Договір про патентну кооперацію.

Патенти на програмне забезпечення відповідно до національного законодавства:

- патенти на програмне забезпечення відповідно до патентного законодавства США;
- патенти на програмне забезпечення відповідно до патентного законодавства Великобританії.

Для США мета патентів викладена в конституційному пункті, який надає Конгресу повноваження «сприяти прогресу науки та корисних мистецтв, забезпечуючи на обмежений час авторам та винахідникам ексклюзивне право на їх відповідні праці та відкриття;» (Стаття I, Розділ 8, Пункт 8). Для Європи не існує подібного визначення. Зазвичай визнаються чотири теорії обґрунтування патентів, як це викладено, наприклад, Махлупом у 1958 р., які включають справедливість щодо винахідника та користь для суспільства шляхом винагородження винахідників. Розкриття інформації вимагається в обмін на ексклюзивне право, і розкриття може сприяти подальшому розвитку. Однак значення розкриття інформації не слід переоцінювати: інакше деякі

винаходи не можна було б тримати в таємниці, а патенти також забороняють використовувати незалежні винаходи.

Прагнучи знайти рівновагу, різні країни мають різну політику щодо того, де повинна знаходитися межа між патентоздатним та непатентоване програмним забезпеченням. В Європі під час дебатів було висунуто ряд різних пропозицій щодо встановлення межової лінії щодо запропонованої Директиви про патентоспроможність комп'ютерних винаходів, жодна з яких не була визнана прийнятною різними учасниками дебатів. Дві конкретні пропозиції щодо перешкоди, яку має передавати програмне забезпечення для патентування, включають:

- Комп'ютерна програма, яка використовує «керовані сили природи для досягнення передбачуваних результатів»;
- Комп'ютерна програма, що забезпечує «технічний ефект».

У США Бен Клеменс, запрошений науковий співробітник Інституту Брукінгса, запропонував надавати патенти лише на винаходи, що включають фізичний компонент, який сам по собі не є очевидним. Це ґрунтується на постанові судді Вільяма Ренквіста у справі Верховного суду США «Діамант проти Діра», яка заявила, що «... незначна діяльність після розпуску не перетворить принцип, що не підлягає патентуванню, на процес, що патентується». вважати програмне забезпечення, завантажене на ПК, абстрактним алгоритмом з очевидною діяльністю після розв'язання, тоді як новий дизайн схеми, що реалізує логіку, швидше за все, буде неочевидним фізичним пристроєм. Дотримання правила «незначної діяльності після розв'язання» згідно з рішенням судді Ренквіста також позбавить більшості патентів від ділових методів.

Відповідно до Європейської патентної конвенції (ЄПК), зокрема її статті 52, "програми для комп'ютерів" не розглядаються як винаходи з метою видачі європейських патентів, але це виключення з патентоспроможності застосовується лише в тій мірі, в якій європейська заявка на патент або європейський патент стосується комп'ютерної програми як такої. В результаті

цього часткового виключення, і незважаючи на те, що ЄРВ розглядає заявки на патенти в цій галузі набагато суворіше, ніж у порівнянні з їхніми американськими аналогами, це не означає, що всі винаходи, включаючи деяке програмне забезпечення, де-юре не підлягають патентуванню.

Патент та захист авторських прав становлять два різні засоби правового захисту, які можуть охоплювати один і той же предмет, наприклад, комп’ютерні програми, оскільки кожен із цих двох засобів захисту слугує своїм цілям. Програмні забезпечення захищені як літературні твори згідно Бернської конвенції. Це дозволяє творцеві заборонити іншому об'єкту копіювати програму, і, як правило, немає необхідності реєструвати код, щоб він захистив авторські права.

Патенти, з іншого боку, дають своїм власникам право перешкоджати іншим користуватися технологією, визначеною патентними заявками, навіть якщо ця технологія була розроблена самостійно і не було скопійовано програмне забезпечення чи програмний код. Насправді одне з останніх рішень ЄРО роз'яснює різницю, зазначаючи, що програмне забезпечення є патентоспроможним, оскільки воно в основному є лише технічним методом, виконуваним на комп'ютері, який слід відрізняти від самої програми для виконання методу, оскільки програма є просто вираз методу і, таким чином, захищена авторським правом.

Патенти охоплюють основні методології, втілені в певному програмному забезпеченні, або функції, які призначені в програмному забезпеченні, незалежно від конкретної мови чи коду, на якому написане програмне забезпечення. Авторське право запобігає прямому копіюванню деяких або всіх версій певного програмного забезпечення, але не заважає іншим авторам писати власні втілення основних методологій. Припускаючи, що набір даних відповідає певним критеріям, авторське право може також використовуватися для запобігання копіюванню заданому набору даних, дозволяючи автору зберігати вміст згаданого набору даних у комерційній таємниці.

Чи суперечить питання щодо того, як і принцип numerus clausus застосовуватиметься до законного гібридного програмного забезпечення, щоб забезпечити розумний баланс між майновими правами власників право власності та правами свободи професіоналів обчислювальної техніки та суспільства в цілому.

2.2.1. Програмне забезпечення з відкритим кодом

У спільноті вільного програмного забезпечення існує сильна неприязнь до патентів на програмне забезпечення. Значна частина цього була спричинена вільним програмним забезпеченням або проектами з відкритим кодом, що припиняються, коли власники патентів, що охоплюють аспекти проекту, вимагали ліцензійних платежів, які проект не міг сплатити, або не бажав сплачувати, або пропонували ліцензії на умовах, коли проект був не бажає прийняти або не може прийняти, оскільки це суперечить ліцензії на вільне програмне забезпечення, що використовується.

Кілька власників патентів запропонували патентні ліцензії на безоплатну винагороду для дуже невеликої частини своїх патентних портфелів. Такі дії викликали лише незначну реакцію спільнот вільного програмного забезпечення та програм з відкритим кодом з таких причин, як страх перед переконанням власника патенту або умови ліцензії настільки вузькі, що мало користі. Серед компаній, які зробили це, є Apple, IBM, Microsoft, Nokia, Novell, Red Hat та Sun (нині Oracle).

У 2005 році компанія Sun Microsystems оголосила, що надає портфель із 1600 патентів за допомогою патентної ліцензії загальної ліцензії на розробку та розповсюдження.

У 2006 році зобов'язання корпорації Майкрософт не подавати позив проти клієнтів Novell Linux, учасників openSUSE та розробників програмного забезпечення з вільним / відкритим кодом щодо патентів, а також пов'язана угода про співпрацю з Novell була сприйнята зневажливою заявою

Юридичного центру свободи програмного забезпечення, тоді як коментатори Фонду вільного програмного забезпечення заявили, що угода не буде відповідати GPLv3. Тим часом Microsoft досягла подібних домовленостей з Dell та Samsung через нібито порушення патентів операційної системи Linux. Microsoft також отримувала дохід від Android, укладаючи такі угоди, щоб не судитися з постачальниками Android.

2.3. Ліцензійна угода з кінцевим користувачем

Ліцензійна угода з кінцевим користувачем (EULA) - це ліцензія, яка надає користувачеві право певним чином використовувати програмний додаток. Ліцензійні договори призначені для забезпечення певних обмежень використання програмного забезпечення, таких як використання програмного забезпечення лише на одному комп'ютері. Укладаючи угоду, користувач отримує дозвіл на використання програмного забезпечення та користування ним.

Ліцензійний договір для завантаженого програмного забезпечення також називається обтіканням клацанням - на відміну від укладання. Це порівняння зроблено, оскільки старі EULA були у паперовій формі в упакованому продукті, до якого не було доступу, поки споживач не відкрив термоусадочну упаковку. Програмні компанії часто укладають спеціальні угоди з великим бізнесом та державними структурами, що включають контракти на підтримку та спеціально складені гарантії.

Завантаження програмного додатка, як правило, передбачає прочитання та згоду на користувацьку ліцензію перед тим, як дозволити її завантажувати. Користувач повинен погодитися з цим типом ліцензування перед встановленням відповідного програмного забезпечення, яке вважається інтелектуальною власністю постачальника програмного забезпечення. EULA

містить вимоги до користувачів програми, які обмежують, як часто і де вони будуть їх використовувати та на яких умовах.

Після відкриття програми встановлення програмного забезпечення EULA має бути цифровим підписом. В іншому випадку встановлення програмного забезпечення не може бути завершено.

Ліцензійні договори не є юридично обов'язковими договорами. Постачальник вимагає згоди клієнта з основними вимогами до використання перед встановленням програмного забезпечення. Коли споживач погоджується із зазначеними умовами ліцензійного договору, він фактично купує або орендує ліцензію у постачальника програмного забезпечення. Після цього споживач може продовжувати встановлюти продукт.

Застереженням ліцензійних договорів є те, що вони не захищають споживача, а лише власника авторських прав. Споживачі ніколи не повинні вважати, що їх права захищаються підписанням EULA. Фактично, постачальник програмного забезпечення володіє ліцензією, а також юридично володіє приватними даними користувача, введеними в програмне забезпечення. Постачальники програмного забезпечення можуть отримувати доступ до приватних споживчих даних у будь-який час, а також читати їх або ділитися ними як завгодно. Це, щонайменше, занепокоїло супротивників ліцензійних договорів. Таким чином, ліцензійні договори не розроблені для будь-якої гарантії. Переваги ліцензійних договорів безумовно дають перевагу власникам, а не на користувачам. Окрім цієї проблеми, яка часто залишається поза увагою, ліцензійні договори є вигідними для власників авторських прав, щоб запобігти копіюванню їх творів.

У суперечках подібного характеру в Сполучених Штатах справи часто апелюють, і різні окружні апеляційні суди іноді не погоджуються щодо цих положень. Це надає можливість Верховному суду США втрутитися, що зазвичай робиться обмеженим і обережним чином, мало заважаючи прецедентному чи постійному законодавству.

Ліцензійні угоди з кінцевими користувачами, як правило, тривалі та складаються на суто юридичній мові, що ускладнює пересічному користувачу надання інформованої згоди. Якщо компанія розробляє ліцензійну угоду з кінцевим користувачем таким чином, що навмисно відмовляє користувачів читати їх і використовує важку для розуміння мову, багато користувачів можуть не давати усвідомленої згоди.

2.3.1. Порівняння з ліцензіями на безкоштовне програмне забезпечення

Ліцензія на безкоштовне програмне забезпечення надає користувачам цього програмного права використовувати з будь-якою метою, модифікувати та розповсюджувати творчі роботи та програмне забезпечення, заборонене за замовчуванням авторським правом, і, як правило, не надається із запатентованим програмним забезпеченням. Ці ліцензії зазвичай містять відмову від гарантії, але ця функція не є унікальною для вільного програмного забезпечення. Ліцензії Copyleft також містять положення про додавання ключів, яке потрібно дотримуватися для копіювання або модифікації програмного забезпечення, яке вимагає від користувача надати вихідний код для твору та розповсюджувати їх модифікації за тією самою ліцензією (а іноді і сумісно); таким чином ефективно захищаючи похідні твори від втрати вихідних дозволів та використання у власних програмах.

На відміну від ліцензійних договорів, ліцензії на вільне програмне забезпечення не працюють як розширення договору чинного законодавства. Жодна домовленість між сторонами ніколи не укладається, оскільки ліцензія на авторське право - це просто декларація про дозволи на те, що в іншому випадку було б заборонено за замовчуванням згідно із законодавством про авторське право.

2.3.2. Ліцензії на упакування в термозбіжну плівку та упакування по клацанню миші

Термін «ліцензія на термозбіжну плівку» в розмовній формі відноситься до будь-якої ліцензійної угоди на програмне забезпечення, яка укладена в пакет програмного забезпечення і є недоступною для клієнта до покупки. Як правило, ліцензійна угода друкується на папері, що входить до комплекта програмного забезпечення. Він також може бути представлений користувачеві на екрані під час встановлення, і в цьому випадку ліцензію іноді називають ліцензією на обертання кліків. Неможливість замовника переглянути ліцензійну угоду перед придбанням програмного забезпечення змусила такі ліцензії порушувати юридичні виклики в деяких випадках.

Чи юридично обов’язкові ліцензії на термозбіжну плівку, різниться між юрисдикціями, хоча більшість юрисдикцій вважають, що такі ліцензії підлягають виконанню. Особливою проблемою є різниця у думках між двома американськими судами у справах Клочек проти шлюзу та Броуер проти шлюзу. Обидва випадки стосувались згорнутої ліцензійної документації, наданої Інтернет-постачальником комп'ютерної системи. Умови термозбіжної ліцензії не були надані під час придбання, а, навпаки, були включені до відвантаженого товару як друкований документ. Ліцензія вимагала від клієнта повернення товару протягом обмеженого періоду часу, якщо ліцензія не була погоджена. У місті Броуер апеляційний суд штату Нью-Йорк постановив, що умови документа, що містить термозбіжну ліцензію, підлягають виконанню, оскільки згода замовника була очевидною через неможливість повернення товару протягом 30 днів, визначених документом. Окружний суд штату Канзас у Клочеку вирішив, що договір купівлі-продажу був завершений на момент угоди, а додаткові умови, що передаються в документі, подібному до документа в Brower, не являли собою контракт, оскільки клієнт ніколи не погоджувався коли договір купівлі-продажу був завершений.

Крім того, у справі ProCD проти Зейденберга ліцензія визнана таким, що виконується, оскільки замовнику було необхідно погодитися з умовами угоди, натиснувши кнопку "Я згоден", щоб встановити програмне забезпечення. Однак у справі «Спехт проти Netscape Communications Corp.» ліцензіат зміг завантажити та встановити програмне забезпечення, не вимагаючи попереднього перегляду та позитивної згоди на умови угоди, тому ліцензія була визнана такою, що не підлягає виконанню.

Ліцензійні угоди Click-Wrap стосуються формування контрактів на веб-сайті. Поширеним прикладом цього є те, що користувач повинен точно дати згоду на умови ліцензування веб-сайту, натиснувши "так" у спливаючому вікні, щоб отримати доступ до функцій веб-сайту. Отже, це аналогічно ліцензіям на термозбіжну плівку, коли покупець мав на увазі згоду на умови ліцензування, спочатку видаляючи термозбіжну плівку програмного забезпечення, а потім використовувати саме програмне забезпечення. В обох типах аналізу основна увага приділяється діям кінцевого користувача та запитується, чи є явне чи неявне прийняття додаткових умов ліцензування.

2.4. Ключі програмного продукту

Ключ продукту, також відомий як програмний ключ, є специфічним програмним ключем для комп'ютерної програми. Він засвідчує, що копія програми є оригіналом.

Ключі продукту складаються з ряду цифр та / або букв. Цю послідовність, як правило, вводить користувач під час встановлення комп'ютерного програмного забезпечення, а потім передає ключ функції перевірки в програмі. Ця функція маніпулює послідовністю ключів відповідно до математичного алгоритму та намагається зіставити результати з набором допустимих рішень.

Генерація стандартних ключів, коли ключі продукту генеруються математично, не є повністю ефективною для припинення порушення

авторських прав програмного забезпечення, оскільки ці ключі можна розподіляти. Крім того, завдяки покращенню комунікації з ростом Інтернету стали більш поширеними атаки на ключі, такі як злом (усунення потреби в ключі) та генератори ключів продукту.

Через це видавці програмного забезпечення використовують додаткові методи активації продуктів, щоб перевірити, чи є ключі дійсними та безкомпромісними. Один із методів присвоює ключ продукту на основі унікальної особливості комп'ютерного обладнання покупця, яку неможливо так легко продублювати, оскільки це залежить від апаратного забезпечення користувача. Інший метод передбачає необхідність одноразової або періодичної перевірки ключа продукту за допомогою Інтернет-сервера (для ігор з онлайн-компонентом це робиться щоразу, коли користувач входить в систему). Сервер може деактивувати немодифіковане клієнтське програмне забезпечення, представляючи недійсні або порушені ключі. Модифіковані клієнти можуть обійти ці перевірки, але сервер все одно може заборонити цим клієнтам інформацію чи зв'язок.

Деякі з найефективніших захистів ключа продукту суперечать через незручності, суворе дотримання вимог, суворі покарання та, в деяких випадках, помилкові спрацьовування. Деякі ключі продукту використовують безкомпромісні цифрові процедури для забезпечення ліцензійної угоди.

Ключі продукту дещо незручні для кінцевих користувачів. Їх потрібно не тільки вводити щоразу, коли встановлюється програма, але й користувач повинен бути впевнений, що не втратить їх. Втрата ключа продукту зазвичай означає, що після видалення програмне забезпечення марне, якщо до видалення не використовується програма відновлення ключа (хоча не всі програми це підтримують).

Ключі продукту також представляють нові способи помилкового розповсюдження. Якщо товар поставляється з відсутніми або недійсними ключами, то сам продукт марний.

Є багато випадків постійних заборон, що застосовуються компаніями, виявляючи порушення користування. Інтернет-система зазвичай негайно вносить у чорний список обліковий запис, на якому запущено злом, або, в деяких випадках, шахрайство. Це призводить до постійної заборони. Гравці, які бажають продовжувати використовувати програмне забезпечення, повинні викупити його. Це неминуче призвело до критики щодо мотивації введення постійних заборон.

Особливо суперечливою є ситуація, яка виникає, коли ключі кількох продуктів пов'язані між собою. Якщо товари мають залежність від інших продуктів (як це має місце з пакетами розширень), зазвичай компанія забороняє всі пов'язані товари.

Наприклад, якщо підроблений ключ використовується з пакетом розширення, сервер може заборонити законні ключі від оригінальної гри. Подібним чином, із послугою Valve Steam усі придбані користувачем товари пов'язані в одному обліковому записі. Якщо цей обліковий запис буде заборонено, користувач втратить доступ до кожного продукту, пов’язаного з тим самим обліковим записом.

Ця "багатозаборона" є дуже суперечливою, оскільки забороняє користувачам продукти, які вони законно придбали та використовували.

2.5. Обфускація

При розробці програмного забезпечення обфускація - це навмисний акт створення вихідного або машинного коду, який людині важко зрозуміти. Подібно обфускації природною мовою, він може використовувати непотрібні обертальні вирази, щоб складати твердження. Програмісти можуть навмисно обфускувати код, щоб приховати його мету (безпека через обфускацію), його логіку або неявні значення, вбудовані в нього, перш за все, з метою запобігання фальсифікації, стримування зворотного проектування або навіть для створення

головоломки чи розваги для когось, хто читає початковий код. Це можна зробити вручну або за допомогою автоматизованого інструменту, останній є найкращим методом у промисловості.

На сьогоднішній день кіберзловмисники озброєні вражаючим асортиментом знань - від простих шкідливих програм до складних інструментів зворотного проектування. Дизасемблери, декомпілятори та інші інструменти дозволяють хакерам отримувати доступ та аналізувати вихідний код програми. За допомогою цієї інформації хакери можуть зловживати програмним забезпеченням різними способами: витягуючи конфіденційну інформацію, додаючи шкідливий код і навіть клонуючи програми.

Дослідження охоронної компанії Positive Technologies показало, що популярні мобільні банківські програми досить доступні до хакерських атак. Найбільш поширені проблеми з кібербезпекою пов’язані з тим, що назви класів і методів явно записані у вихідному коді, відсутність захисту від введення коду та відсутність обфускації коду.

Обфускація коду, зокрема, є перспективною практикою забезпечення програмного забезпечення. Якщо ви шукаєте способи захистити своє програмне забезпечення від вторгнень, для початку, можливо, ви захочете зміцнити його захист, заважаючи коду.

Методи обфускації дозволяють вам загартувати код програми, перетворюючи його, щоб приховати неявні значення та приховати логіку. Ці заходи ускладнюють несанкціоновану третю сторону зазирнути всередину вашого програмного забезпечення.

Більшість методів обфускації трансформують один із наступних аспектів коду:

- Дані. Зробіть елементи коду схожими на те, що вони є;
- Керування потоком. Потрібно робіти виконувану логіку недетермінованою, якщо програмне забезпечення декомпілюється;
- Структура макета. Форматування даних, перейменування ідентифікаторів та видалення коментарів коду.

Інструменти затухання працюють із вихідним кодом, машинним або двійковим кодом та байт-кодом. Щоб визначити, який тип коду найкраще затушувати, потрібно пам’ятати про обмеження кожного вибору.

Забруднюючи вихідний код, ви можете зіткнутися з проблемами при обробці та налагодженні обфускованого коду. Бінарне обфускування не тільки складніше, ніж інші два варіанти, але його потрібно застосовувати до кожної архітектури системи.

Хоча обфускування є однією з практик безпечного кодування, рекомендованої OWASP, воно все ще не настільки популярне серед багатьох розробників. Основна причина цього полягає в тому, що при надмірному використанні обфускованого коду може зашкодити продуктивності програмного забезпечення.

Але якщо ми говоримо про програму для мобільного банкінгу, важливість захисту даних банківського рахунку користувача є досить високою, щоб виправдати втрату додаткового часу та зусиль на збалансоване зміцнення коду.

Якщо ви хочете найбільш ефективно заплутати код, тут слід врахувати кілька речей:

- Перш ніж обфускувати ваш код, вам слід вирішити, які частини вашого коду можна заплутати. Уникайте заплутування критично важливого продуктивності коду. Також переконайтесь, що обфускований код жодним чином не впливає на функціональність програмного забезпечення;
- Зверніть увагу на розмір та ефективність заплутаного коду. При компіляції великого шматка коду час виконання може збільшитися до 1000 разів;
- Додайте непрозорі предикати до обфускованого коду, щоб неможливо було зрозуміти, куди піде виконання коду.
- Щоб створити шаруватий захист, поєднуйте кілька методів трансформації. Чим більше методів обфускації ви використовуєте, тим краще ваш код буде захищений;

- Використовуйте затухання коду лише як додатковий рівень безпеки, оскільки він не може замінити інші методи безпеки.

Забезпечення коду при правильній реалізації є ефективним рішенням для захисту вашого програмного забезпечення від несанкціонованого аналізу.

2.6. Програмне забезпечення, захищене від фальсифікації

Програмне забезпечення проти злому - це програмне забезпечення, яке ускладнює зловмиснику шанси на його модифікацію. Заходи, що беруть участь, можуть бути пасивними, такі як затухання, щоб ускладнити зворотну інженерію, або активні методи виявлення фальсифікації, мета яких зробити програму несправною або взагалі не працювати, якщо її змінити. По суті, це захист від втручання, реалізований у програмному домені. Він поділяє деякі аспекти, але також відрізняється від суміжних технологій, таких як захист від копіювання та надійне обладнання, хоча часто використовується в поєднанні з ними. Технологія захисту від фальсифікації зазвичай робить програмне забезпечення дещо більшим, а також впливає на продуктивність. Не існує надійно захищених програмних засобів захисту від фальсифікації; таким чином, поле - це гонка озброєнь між зловмисниками та програмними засобами протидії фальсифікаціям.

Втручання може бути зловмисним, щоб отримати контроль над деяким аспектом програмного забезпечення з несанкціонованою модифікацією, яка змінює програмний код та поведінку комп'ютера. Приклади включають встановлення руткітів та бекдорів, відключення контролю за безпекою, підривання автентифікації, введення шкідливого коду для крадіжки даних або

для досягнення вищих привілеїв користувача, зміни потоку керування та зв'язку, обхід ліцензійного коду з метою піратства програмного забезпечення, втручання в код для вилучення даних або алгоритмів. Програмні забезпечення вразливі до наслідків фальсифікації та змін коду протягом усього їх життєвого циклу - від розробки та розгортання до експлуатації та обслуговування.

Захист від несанкціонованого доступу може бути застосований як внутрішньо, так і зовні до програми, що захищається. Зовнішнє протидії фальсифікації зазвичай здійснюється шляхом моніторингу програмного забезпечення для виявлення фальсифікації. Цей тип захисту зазвичай виражається як сканер шкідливих програм та антивірусні програми. Внутрішнє запобігання фальсифікації використовується для перетворення програми у власну систему безпеки, і зазвичай це робиться з певним кодом у програмному забезпеченні, який виявлятиме фальсифікацію в міру її виникнення. Цей тип захисту від несанкціонованого доступу може мати форму перевірок цілісності виконання, таких як контрольні суми циклічного резервування, заходи проти налагодження, шифрування або затухання. Виконання у віртуальній машині стало звичним методом протидії втручанню, що застосовується в останні роки для комерційного програмного забезпечення; він використовується, наприклад, у StarForce та SecuROM. Деяке програмне забезпечення, що захищає від фальсифікації, використовує криптографічну скриньку, тому криптографічні ключі не розкриваються навіть тоді, коли криптографічні обчислення детально спостерігаються в налагоджувачі. Більш пізньою тенденцією досліджень є програмне забезпечення, захищене від фальсифікацій, яке має на меті виправити наслідки фальсифікації та дозволити програмі продовжувати працювати якби вона не була модифікована. Проста схема такого роду була використана у відеоіграх Diablo II, яка зберігала свої важливі дані гравця у двох копіях у різних місцях пам'яті, і якщо одна була модифікована зовні, гра використовувала нижнє значення.

2.7. Програмний водяний знак

Водяний знак програмного забезпечення передбачає вбудовування унікального ідентифікатора в частину програмного забезпечення для запобігання крадіжці програмного забезпечення. Водяні знаки не запобігають крадіжці, але натомість знеохочують злодіїв програмного забезпечення, надаючи засоби для ідентифікації власника програмного забезпечення та / або походження викраденого програмного забезпечення. Потім його можна витягти екстрактором або перевірити впізнавачем для підтвердження права власності на програмне забезпечення. Перший витягує оригінальний водяний знак, а другий лише підтверджує наявність водяного знаку. Алгоритм розпізнавання або вилучення водяних знаків також може бути класифікований як сліпий, якщо оригінальна програма та водяний знак недоступні, або повідомлений, коли доступна оригінальна програма та / або водяний знак.

Також можна вбудувати унікальний ідентифікатор клієнта в кожну копію розповсюджуваного програмного забезпечення, що дозволяє програмній компанії визначити особу, яка піратствувала програмним забезпеченням. Потрібно, щоб водяний знак був прихований, щоб його не вдалося виявити та видалити.

У більшості випадків водяний знак повинен бути надійним - тобто стійким до семантики, що зберігає трансформації (такі як оптимізація або обфускація). Однак у деяких випадках бажано, щоб водяний знак був крихким у тому сенсі, що якщо в програмному забезпеченні виконується семантика, що зберігає перетворення, водяний знак стає недійсним. Це корисно в контексті ліцензування програмного забезпечення, коли будь-які зміни в програмі можуть її відключити.

Методи нанесення водяних знаків широко використовуються в індустрії розваг для ідентифікації мультимедійних файлів, таких як аудіо- та відеофайли, і ця концепція поширилася на індустрію програмного забезпечення. Водяні

знаки не ставлять за мету зробити щоб програму було важко викрасти або нерозбірною, як обфускація, але це стримує крадіжки, оскільки злодії знають, що їх можна ідентифікувати.

Водяні знаки можна класифікувати як видимі, якщо реконгатор є загальновідомим, так і невидимі, де реконструктор або якийсь компонент (наприклад, ключ шифрування) не є загальнодоступним. Видимі водяні знаки можуть діяти як стримуючий фактор, але також можуть показати противнику розташування водяного знака, що полегшує завдання його видалення.

2.7.1. Труднощі програмного водяного знаку

Водяні знаки програмного забезпечення представляють кілька проблем із впровадженням, і багато з сучасних алгоритмів водяних знаків вразливі до атак. Програмне забезпечення з водяними знаками повинно відповідати наступним умовам:

- розмір програми не можна суттєво збільшувати;
- ефективність програми не повинна суттєво знижуватися;
- надійні водяні знаки повинні бути стійкими до семантики, зберігаючи трансформації (тендітні водяні знаки, за визначенням, не повинні бути);
- водяні знаки повинні бути достатньо добре прихованими, щоб уникнути їх видалення;
- водяні знаки повинні бути простими для отримання власником програмного забезпечення.

Можливо, найскладнішою проблемою для вирішення є збереження водяного знаку прихованим від зловмисників, одночасно дозволяючи власнику програмного забезпечення ефективно витягувати водяний знак при необхідності. Якщо водяний знак занадто легко витягти, то зловмисник також

зможе витягти водяний знак. Якщо водяний знак занадто добре прихований, тоді власник програмного забезпечення може не вдатися знайти водяний знак для його вилучення. Деякі інструменти водяних знаків використовують маркери для позначення місця збереженого водяного знака - це проблематично, оскільки створює ризик піддати водяний знак супернику.

Водяні знаки повинні бути стійкими до семантики, що зберігає трансформації, і в ідеалі має бути можливо розпізнати водяний знак із часткової програми. Семантика, що зберігає трансформації, за визначенням приводить до програм, які синтаксично відрізняються від оригіналу, але поведінка яких однакова. Здійснюючи такі перетворення, зловмисник може спробувати створити семантично еквівалентну програму із видаленим водяним знаком. Надлишковість та розпізнавання з порогом ймовірності можуть допомогти у вирішенні цих проблем.

Код водяного знаку повинен бути локально невідрізним від решти програми, щоб він був прихований від противників. Наприклад, уявіть собі водяний знак, який складається з фіктивного методу зі 100 змінними - такий тип методу, ймовірно, буде виділятися під час простого аналізу програмного забезпечення (наприклад, використання методів програмних метрик). Може бути важко програмувати генерацію коду, який неможливо розшифрувати із створеного людиною програмного коду, але статистичний аналіз оригінальної програми може допомогти у створенні відповідних водяних знаків.

В ідеалі водяні знаки програмного забезпечення повинні бути стійкими до атак декомпіляції-рекомпіляції, оскільки можлива декомпіляція Java.

Водяні знаки програмного забезпечення можуть бути ефективними для використання, при виконанні наступних вимог:

- вартість часу вбудовування;
- вартість виконання.
- вартість часу розпізнавання.

Типи водяних знаків. Чотири типи водяних знаків:

- марка авторства, що ідентифікує автора програмного забезпечення чи авторів. Ці водяні знаки, як правило, видимі та надійні.
- марка відбитків пальців, що ідентифікує канал розповсюдження, тобто особу, яка поширила програмне забезпечення. Водяні знаки, як правило, невидимі, надійні і складаються з унікального ідентифікатора, такого як контрольний номер клієнта.
- позначка перевірки, щоб переконатися, що програмне забезпечення є справжнім і незмінним, наприклад, як аплети Java з цифровим підписом. Ці водяні знаки повинні бути видимими для кінцевого користувача, щоб забезпечити перевірку, та неміцними, щоб забезпечити несанкціоноване використання програмного забезпечення.
- марка ліцензування, що використовується для автентифікації програмного забезпечення за ліцензійним ключем. Ключ повинен стати неефективним, якщо водяний знак пошкоджений, тому ліцензійні знаки повинні бути крихкими.

Висновки до другого розділу

Проведено аналіз існуючих технологій та механізмів протидії механізмів та систем нелегального копіювання інформації. Простежено функціональність та направленість кожного рішення відповідно до їх взаємодії та можливостей захисту різних рівні. Виокремлено позитивні та негативні моменти у роботі з кожним захисним механізмом. Розглянуто особливості реалізації захисних функцій, розгортки та/або позиціонування.

Розділ 3. СТВОРЕННЯ ПРОГРАМНОГО ЗАСОБУ ТА РОЗРОБКА РЕКОМЕНДАЦІЙ ДЛЯ ЗАХИСТУ ІНФОРМАЦІЇ ВІД НЕЛЕГАЛЬНОГО КОПІЮВАННЯ

Хоча більшість людей будуть чесними, платити за програмне забезпечення і не використовувати його не за призначенням, є люди, які будуть використовувати неліцензійні копії програмного забезпечення, крадуть програмне забезпечення з метою його перепродажу або реконструюють його, просто щоб сказати, що у них є.

Розробник втрачає незліченні години на планування, проектування та кодування програмного забезпечення для вирішення будь-якої проблеми. Найменше вам потрібно, щоб хтось вкрав вашу важку працю і нажився на ньому.

Інший, менш продуманий побічний ефект поганий безпеки програмного забезпечення - це потенційний збиток репутації програми.

За даними BSA, шкідливе ПЗ з неліцензійного ПЗ коштує близько 359 мільярдів доларів на рік. Якщо компанія визначить, що ваше програмне забезпечення, навіть якщо воно використовувалося неліцензійним, принесло шкідливе ПО, яке заподіяло шкоду, чи буде ця компанія звертатися до вас за своєю наступною покупкою програмного забезпечення? Можливо немає.

Навіть якщо розуміти, що програмне забезпечення було неліцензійним, в чому повністю винна компанія, все одно доведеться вжити заходів щодо усунення пошкоджень, щоб гарантувати, що шкідлива версія вашого програмного забезпечення не зачепить нікого.

Найкраще захистити своє програмне забезпечення до того, як воно буде поширене.

Захист комерційної таємниці і бренду має очевидні переваги. До них відносяться уникання непотрібних витрат, не кажучи вже про розчарування, пов'язаному з відкотом і оновленням програмного забезпечення з належним захистом. Безсумнівно, краще працювати над розробкою свого програмного забезпечення, ніж намагатися повернути його.

3.1. Програмний засіб для генерації та реєстрації ключа програмного продукту

Ліцензійний ключ програмного забезпечення (також відомий як ключ продукту) засвідчує оригінал копії програми. Зазвичай це унікальний рядок цифр і символів, і для активації одночасної роботи двох ідентичних ключів продукту часто потрібна онлайн-активація.

Однак програмним ключем може бути також USB або апаратний ключ, який фізично з'єднується з комп'ютером. Однією з небагатьох переваг використання цього типу апаратного ключа є те, що його можна використовувати без підключення до Інтернету.

Ліцензійний ключ програмного забезпечення - це шаблон цифр та / або букв, який надається уповноваженому покупцеві. Коли користувач вводить його під час встановлення програмного забезпечення, ключ розблоковує програмний продукт і робить його доступним для використання.

Створений неінтегрований програмний засіб надає можливість генерувати та реєструвати ключ програмного продукту, базуючись на

унікальному ID комп’ютера, що виключає можливість використання однакових ключів на різних ПК. Також програма надає можливість генерувати ключ програмного продукту в залежності від терміну дії.

Відкриваючи програмний додаток, користувачу надається можливість вибору трьох пунктів:

- Генерація – користувач переходить до вікна генерування ключа програмного продукту.
- Реєстрація – користувачу надається можливість ввести його ключ.
- Інформація – користувач отримує інформацію на рахунок його теперішньої версії програмного продукту.

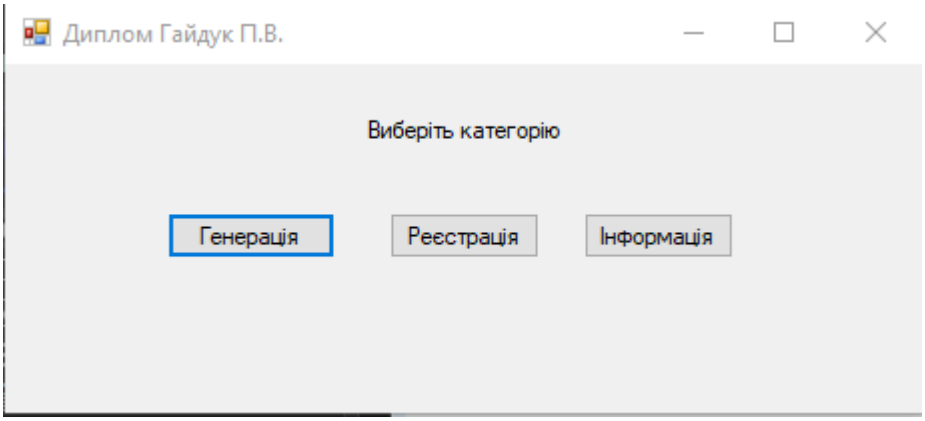

Рис. 3.1. Головна сторінка програмного додатку.

Натиснувши кнопку "Генерація", відкривається вікно генерації. В цьому вікні користувач може ознайомитися з його унікальним ID продукту, який базується на його особистому ID комп’ютера. Також надається можливість вибрати тип ліцензії програмного продукту, а саме: повна або обмежена. У випадку обмеженої ліцензії потрібно вказати кількість днів, протягом яких програма буде доступна для використання. Після натискання "Згенерувати", користувач отримує "Ключ продукту", який потрібно скопіювати для подальшого використання.

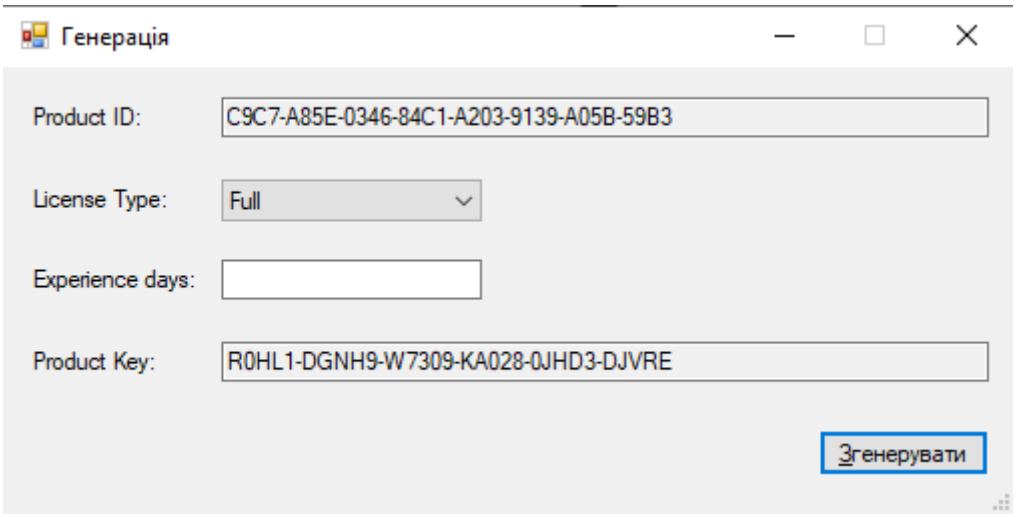

Рис. 3.2. Вікно генерації ключа програмного продукту у випадку повної версії.

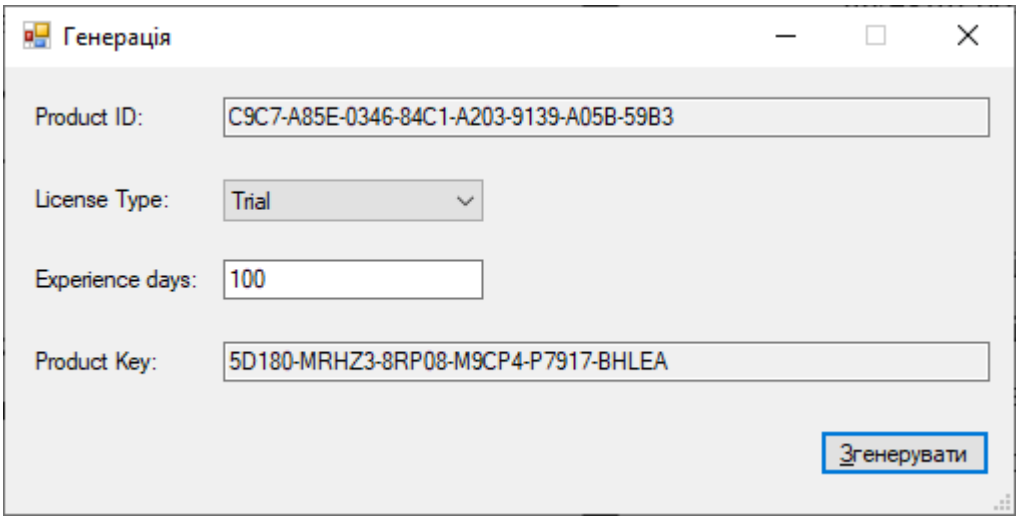

Рис. 3.3. Вікно генерації ключа програмного продукту у випадку обмеженої версії.

Повернувшись на головну сторінку і натиснувши "Реєстрація", користувача перенаправить на вікно реєстрації, на якому буде відображено його теперішній ID продукту, а також пустий рядок програмного ключа, в який потрібно вписати ключ, згенерований раніше. Натиснувши "ОК" у випадку вірного ключа, користувач буде сповіщений, що його ключ вірний і програма буде доступна для використання, а сам користувач автоматично повернеться на головне вікно. Якщо ключ не вірний, програма покаже повідомлення "Ключ продукту не вірний! ".

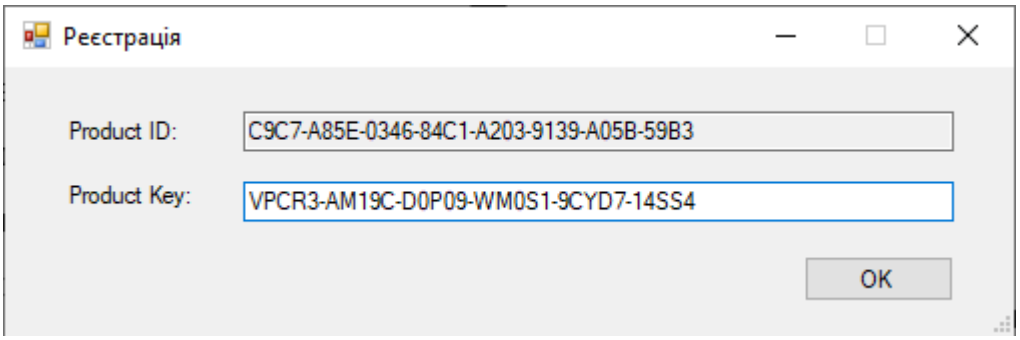

Рис. 3.4. Реєстрація ключа програмного продукту.

Повернувшись на головне, користувач зможе дізнатися інформацію, яка стосується його програмного засобу натиснувши "Інформація". В цьому вікні, він побачить назву продукту, ID продукту, ключ продукту, а також тип ліцензії, який він реєстрував.

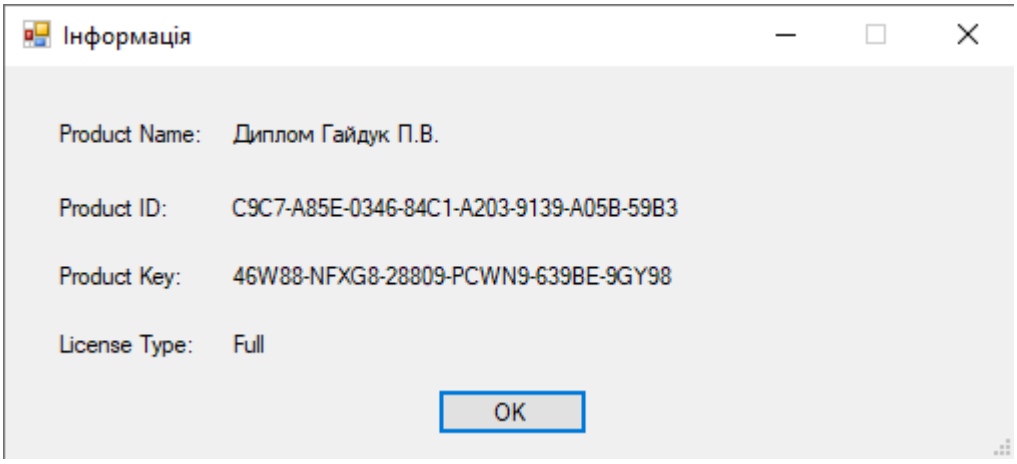

Рис. 3.5. Вікно інформації повної версії програмного продукту.

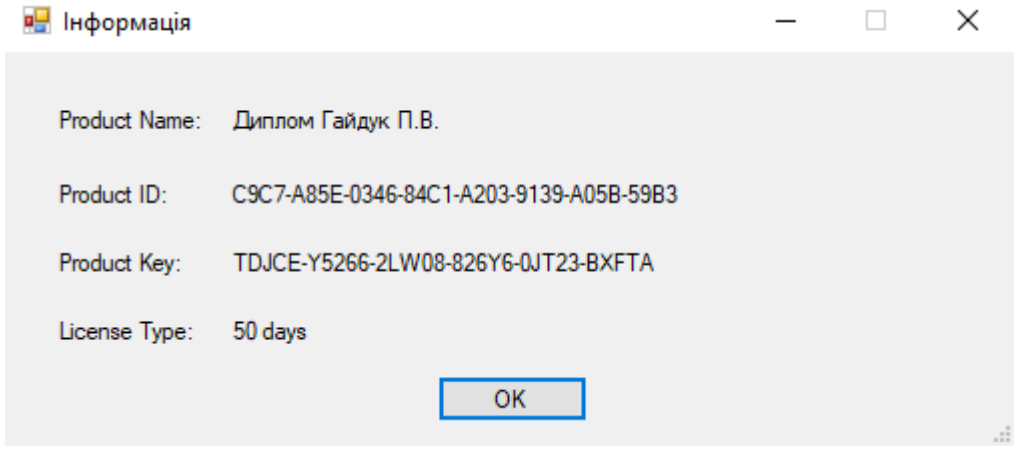

Рис. 3.6. Вікно інформації обмеженої версії програмного продукту.

3.2. Ключові елементи захисту

Захист від копіювання програмного забезпечення починається з стратегії ліцензування, де визначається, як отримати компенсацію за своє програмне забезпечення. Стратегії ліцензування включають:

- Безстроковий. Одноразовий платіж дає необмежений доступ до додатка;
- На основі функцій. Програмне забезпечення купується, а додаткові функції розблокуються з додатковими покупками;
- По часу. Програмне забезпечення орендується на певний термін і може бути продовжено після закінчення цього терміну;
- Багаторівневий. Версії програмного забезпечення пропонуються в залежності від кількості доступних функцій, наприклад Silver, Gold або Platinum.

Як тільки прийде етап стратегії ліцензування, буде підготовлено рішення, яке буде застосовувати цю стратегію і зробити її дуже важкою або майже неможливою для обходу, тому вам будуть платити правильну грошову оцінку для кожного користувача.

Стратегія захисту вашого програмного забезпечення від копіювання повинна враховувати, де ваше програмне забезпечення буде випущено. В області захисту програмного забезпечення від копіювання не існує універсального підходу. Потрібно шукати рішення, яке буде повністю задовольняти усі вимоги.

Також знадобиться рішення, що відбиває необхідний рівень безпеки. Зручна аналогія: не слід замикати велосипед в Форт-Нокс, щоб зберегти його в безпеці, і не слід використовувати велосипедний замок, щоб закріпити діамант Надії. Захист повинен відповідати вартості програмного забезпечення.

Види захисту. Зазвичай це недорогий варіант. Реалізація програмних рішень зазвичай здійснюється після компіляції програмного забезпечення. Для

захисту програми використовується програмна оболонка з деякими настройками конфігурації. Коли програмне забезпечення запускається кінцевими користувачами, воно не підключається до будь-яких ресурсів для захисту. Оскільки всі параметри ліцензування знаходяться на комп'ютері, на якому запущено програмне забезпечення, несанкціоноване використання досить легко. Рівень безпеки програмних рішень знаходиться в діапазоні від низького до середнього.

3.3. Програмне забезпечення з підключенням до Інтернету і активацією

Необхідне зворотне підключення до сервера ліцензування зазвичай призводить до високої вартості настройки, а також до регулярних платежів. Знову ж таки, для реалізації захисту використовується програмна оболонка, але оскільки потрібне підключення до Інтернету, варіанти, запропоновані програмною оболонкою, більш надійні. Розширені можливості пошуку дозволяють краще зрозуміти, де використовується програмне забезпечення, як воно використовується і чи використовується воно неавторизованим чином. Оскільки підключення до Інтернету потрібно завжди, деякі місця розташування можуть бути виключені. Рівень захисту знаходиться в діапазоні від помірного до сильного, оскільки параметри ліцензування залишаються віддаленими на безпечних серверах ліцензування.

Апаратний захист. Безпека дуже висока, тому що ліцензування міститься в апаратному USB-ключі безпеки, і підключення до Інтернету не потрібно. Вартість кожної ліцензії в розрахунку на кожну ліцензію невисока, і періодичні ліцензійні збори не стягуються. Реалізація може бути виконана з

використанням інтерфейсів прикладних програм (API) або програмної оболонки.

3.4. Переваги ключів безпеки

Апаратні захисні ключі USB - кращий вибір для ліцензування програмного забезпечення та забезпечення безпеки. Їх можна швидко і легко реалізувати за допомогою програмної оболонки. Для більшої гнучкості API-інтерфейси можуть бути інтегровані в ваше програмне забезпечення, а для максимальної безпеки можна використовувати як програмну оболонку, так і інтеграцію API.

Захисні ключі мають безліч переваг, починаючи з форм-фактора. Наявність фізичного ключа видаляє ліцензію з апаратного забезпечення комп'ютера. Із зовнішнім ключем, розташованим поза операційної системи, безпека досить висока.

Перенести ліцензію з одного комп'ютера на інший так само просто, як вийняти USB-накопичувач з одного комп'ютера і вставити його в інший. Захищений фізичний ключ електронного ключа не може бути прочитаний або скопійований.

Використовуючи USB-ключ, ви не зв'язуєтеся з сервером ліцензування, тому ви можете розгортати програмне забезпечення там, де підключення до Інтернету обмежено або заборонено. Це включає в себе місця з високим рівнем безпеки, такі як урядові установи і райони, де інтернет-трафік строго відстежується, обмежується або забороняється. Ключі позбавляють від необхідності надавати різні рішення для різних середовищ.

Інші переваги USB-ключів:

- Підвищена безпека вашої інтелектуальної власності (IP). Це можливо завдяки тісній інтеграції між прошивкою ключа і вашим програмним забезпеченням;

- Гнучкі варіанти ліцензування. Ваші можливості для створення і забезпечення дотримання ліцензій практично безмежні;
- Виключно проста реалізація. Першокласне рішення для електронного ключа може бути реалізовано протягом декількох хвилин, а не днів, тижнів або місяців;
- Мультиплатформенна підтримка. USB-ключі можуть підтримувати всі версії Windows, а також системи Mac і Linux.

Ресурс USB-ключа

Якщо вибирати цей варіант захисту, крім першокласного захисту, обраний вами постачальник електронного ключа для захисту програмного забезпечення від копіювання повинен надати:

- Легкість реалізації;
- Швидкість доставки виробничих ключів. Ніхто не хочете чекати, поки постачальник продасть більше програмного забезпечення;
- Гнучкість в реалізації та налаштування рішення відповідно до потреб;
- Ліцензування програмного забезпечення далеко не універсальний.
- Можна бути спокійним, знаючи, що ваш постачальник має перевірений послужний список і буде поруч, щоб підтримувати вас протягом усього життєвого циклу вашого застосування;
- Підтримка, яка допоможе впровадити рішення, а також постійна підтримка, коли прийде час оновити ваше програмне забезпечення;
- Не дозволяйте іншим отримувати прибуток від вашої важкої праці

Хоча рівень комп'ютерного піратства знизився на 2 відсотки за останні два роки, на нього як і раніше припадає 37 відсотків програмного забезпечення, встановленого на персональних комп'ютерах. Незначне зниження рівня піратства не означає, що можна нехтувати ліцензуванням програмного забезпечення і безпекою.

Пропозиції для конфіденційності програмного забезпечення:

1. Не розголошувати інформацію. Діліться ідеєю тільки з людьми, яким ви довіряєте, яким потрібно знати. Спочатку тільки партнери та потенційні члени команди, і тільки якщо їм довіряєте. Ніякого хвастощів, ніякого відкриття незнайомцям. Будьте обережні і з експертами. Ваш програмний бізнес буде залежати від комерційних секретів, а не від юридичних реєстрацій.

2. Потрібен адвокат. Багато що з того, що ви робите чи не робите, пов'язане з юридичною роботою, тому потрібен адвокат. Ви можете скоротити витрати, ознайомившись з проблемами перед зустріччю, щоб витрачати менше часу.

3. Переконатися, що код належить вам. Якщо хтось, крім вас, працює над кодом, запитайте свого адвоката, як переконатися, що він належить вам. Це пов'язано з продуктом роботи. Дотримуйтесь порад вашого адвоката з цього приводу. Не залишайте нічого на волю випадку або словесних обіцянок.

4. Авторські права на весь код. Програмне забезпечення складно запатентувати, але легко отримати авторське право. Авторське право поширюється на творчі роботи. Авторське право відносно дешево. Однак потрібно знати, що авторське право не захищає вас від копіювання за допомогою зворотного проектування, але, по крайній мері, захищає вас від людей, що крадуть ваш реальний код, тому що захищений тільки код, справжні слова; не ідея.

5. Розумно використовуйте угоди про нерозголошення. Можна попросити підписані угоди про нерозголошення у потенційних співзасновників і членів команди або інвесторів з друзів і сім'ї. Не покладайтеся на ці документи, не робіть їх великою проблемою, але використовуйте їх як можна частіше; і не чекайте, що їх підпишуть серйозні незалежні інвестори.

6. Патенти призначені для винаходів, алгоритмів і формул, а не ідей. Не можна запатентувати ідею. Якщо ідея програмного забезпечення залежить від розробленого алгоритму і раніше не використовувалася, можливо, у є шанс запатентувати його. Однак далеко не всі програмні продукти містять

алгоритми, які можна запатентувати. Найкраще для цього підійде патентний повірений, який має досвід роботи з програмним забезпеченням. Не об'єднуйте в пристрій тонну роботи і не витрачайте багато грошей без попередньої консультації з патентними експертами.

7. Товарні знаки призначені для торгівлі. Позначте логотип, рядки тегів, зображення і основні аргументи у вигляді цитат. Торгова марка теж відносно дешева.

8. Реєстрація доменного імені і юридичної особи. Ні те, ні інше не захищає програму, але обидва є хорошою ідеєю. Реєстрація юридичної особи в деякій мірі захищає фірмове найменування, якщо код був першим в світі. Це може коштувати всього 50 доларів. Реєстрація доменного імені, якщо є гарне, захищає для цього домену, але не наслідувачів зі схожими іменами.

9. Запишіть ідею і надішліть 10 копій рекомендованим листом. Це може захистити пізніше, коли виникнуть неакуратні проблеми, надавши юридичні докази того, в який день була ідея. Це дуже слабкий захист, але дешева і проста у використанні.

10. Інвестори приходять набагато пізніше. Не розмовляйте з інвесторами до тих пір, поки не досягнете успіху, маючи робочу версію, яку можна продемонструвати, досягнуті віхи і успіх. І остерігайтеся фальшивих інвесторів, які насправді є консультантами, які шукають нових клієнтів. Завжди перевіряйте веб-сліди інвесторів, перш ніж ділитися з ними чим-небудь. У законних інвесторів є сліди в мережі, і їх легко ідентифікувати.

Під час захисту програмного забезпечення, потрібно бути готовим до наступного:

1. Копіювання дозволено. Що стосується програмного забезпечення, це законно, якщо вони не крадуть код, а просто дублюють функціональність. З тих пір, як програмні продукти компаній вперше починають впливати на ринок, їх починають копіювати.

2. Інші почнуть копіювати ідею, концепції, веб-сайти, навіть упаковку і маркетингові повідомлення. І все це було законно.

3. Продукти весь час копіюються. Спробуйте знайти важливу категорію програмних продуктів, в якій бере участь тільки один учасник. Конкуренти з'являться, якщо ринок буде цікавим. Іноді конкуренція буває гірше, тому що це означає, що нікому не потрібно те, що продається.

4. Правовий захист зазвичай досить слабкий. Навіть патенти, які є кращим захистом, постійно копіюються або уникають обхідними шляхами. Реальна вартість патенту багато в чому залежить від того, наскільки добре він був написаний в першу чергу, а потім від того, наскільки уважно спостерігається за порушеннями.

5. Будуйте бар'єри для входу і постійно зміцнюйте їх. Єдиний реальний захист - бути швидшим і кращим конкурентів і продовжувати вдосконалюватися, щоб завжди бути попереду. Примусьте своїх клієнтів полюбити, зробіть їх лояльними. Слідкуйте за новими ринками і новими додатками в суміжних областях продукту або ринку.

3.5. Висновки до третього розділу

Виокремлено основні технології та механізми захисту інформації від нелегального копіювання у програмних забезпеченнях. Створено програмний засіб, який надає можливість реєструвати та генерувати ключ програмного продукту, базуючись на унікальному ID комп’ютера. Розроблено рекомендації по захисту інформації від нелегального копіювання, котрі включають згадані рішення захисту та поєднують їх в єдину модель захисту для найбільшої ефективності.

ВИСНОВКИ

В бакалаврській роботі були отримані наступні результати:

1. Досліджено основні технології та механізми виявлення атак на програмні забезпечення, втрату конфіденційності інформації з використанням спеціалізованих додатків та механізмів, фахівців в боротьбі з відповідним типом атак, академічних дослідженнях та опираючись на досвід, отриманий в результаті роботи та аналізу курсів, документації програмного забезпечення пов’язаного з виявленням або протидії нелегальному копіюванню інформації.

2. Проаналізовано базові рішення протидії нелегальному копіюванню інформації опираючись на авторитетні журнали та веб-ресурси з відповідною кваліфікацією в питаннях протидії атакам. Встановлено необхідність поєднання рішень для підвищення ефективності цільової стратегії захисту.

3. Створено програмний засіб, який надає можливість реєструвати та генерувати ключ програмного продукту, базуючись на унікальному ID комп’ютера. Доцільна до впровадження розроблена рекомендація захисту цільової інформації від нелегального копіювання різних рівнів захисту з використанням встановлених в бакалаврській роботі технологій та механізмів виявлення та протидії нелегальному копіюванню і їх поєднання для максимальної ефективності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тейєр, Кен. "How Does Reverse Engineering Work?". Процитовано 26 лютого 2018.

2. Вільяверде, Алехандро Ф .; Банга, Хуліо Р. (6 лютого 2014 р.). "Reverse engineering and identification in systems biology: strategies, perspectives and challenges". Журнал інтерфейсу Королівського товариства.

3. Чикофський, Е. & Cross, J.H., II (1990)."Reverse Engineering and Design Recovery: A Taxonomy". С. 13–17.

4. "The Tracing Book". Архів оригіналу 24.02.2009.

5. Роб О’Ніл (28 серпня 2013 р.)."New Zealand bans software patents". Процитовано 6 вересня 2013.

6. Закон про патенти, № 57 від 1978 р., Із змінами, внесеними 26 квітня 1978 р

7. Закон про патенти, № 57 від 1978 р., Із змінами, внесеними до розділу 25 (3), 26 квітня 1978 р

8. Поширені запитання - Корея, Європейське патентне відомство, архівовано з оригіналу 13 вересня 2009 р., Отримано 29 жовтня 2008 р

9. Ломбарді, Кендес (27 листопада 2006 р.), "Microsoft lost in translation", Блог новин, CNET, отримано 29 жовтня 2008 року

10. Чанг, Хой; Аталла, Михайло Дж. (2002). "Protecting Software Codes by Guards". Security and Privacy in Digital Rights Management. p. 160-175.

11. "ProduKey - Recover lost product key (CD-Key) of Windows/MS-Office/SQL Server". NirSoft. Retrieved 2021-02-09.

12. "Australian Pandora Tomorrow CD-Key Problems" Shack News

13. "Valve suspends 20,000 Steam accounts". GameSpot. Процитовано 15.05.2013.

14. "Obfuscation - Haskell Wiki". 16 лютого 2006 р. Випущено 3 березня 2020 р.

15. Монфор, Нік. "Obfuscated code". Джерело листопада 24,2017.

16. "JAPH - Just Another Perl Hacker". Perl Mongers. Перевірено 27 лютого 2015 року.

17. Каппаерт, Дж .; Преніл, Б. (2010). "A general model for hiding control flow". Матеріали десятого щорічного семінару ACM з управління цифровими правами - DRM '10.

18. "Keeping the Pirates at Bay". Гамасутра. Джерело 2013-12-24.

19. Чабоя, Давид (20 червня 2007). State of the Practice of Software Anti-Tamper. Управління технологій ініціативи із захисту від фальсифікації та захисту програмного забезпечення, Науково-дослідна лабораторія ВПС. Процитовано 24 грудня 2013.

20. Чой, Девід Ю .; Перес, Артуро (квітень 2007). "Online piracy, innovation, and legitimate business models". С.: 168–178.

21. "Definition of: Internet piracy". Енциклопедія журналу ПК. Перевірено 26 жовтня 2018.

22. Рош, Джером; Аллегрі; Вчені, Талліс; Філіпс; Палестріна; Манді (червень 1981). "Miserere". The Musical Times. С.: 412.

23. Гільйо, Ю .; Газет, А. (2009). "Semi-automatic binary protection tampering". Журнал з комп'ютерної вірусології. С.: 119–149.

24. Oorschot, P. C. (2003). "Revisiting Software Protection". Інформаційна безпека. Конспект лекцій з інформатики. С.: 1–13.

Додаток А

Фрагмент вихідного коду головної сторінки

```
namespace LicenseKey
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void GeneratBtn(object sender, EventArgs e)
      {
         using (frmGenerate frm = new frmGenerate())
         {
            frm.ShowDialog();
```

```
            }
        }

        private void RegBtn(object sender, EventArgs e)
        {
            using (Regfrm frm = new Regfrm())
            {
                frm.ShowDialog();
            }
        }

        private void AboutBtn(object sender, EventArgs e)
        {
            using (frmAbout frm = new frmAbout())
            {
                frm.ShowDialog();
            }
        }
    }
}
```

Додаток Б

Фрагмент вихідного коду генерації ключа програмного продукту

```
namespace LicenseKey
{
    public partial class frmGenerate : Form
    {
        public frmGenerate()
        {
            InitializeComponent();
        }

        private void Label1_Click(object sender, EventArgs e)
        {

        }
```

```
        private void GenerateBtn(object sender, EventArgs e)
        {
            KeyManager KeyMan = new KeyManager(ProdIDtxt.Text);
            KeyValuesClass KeyVal;
            string productkey = string.Empty;
            if (LicType.SelectedIndex == 0)
            {
                KeyVal = new KeyValuesClass()
                {
                    Type = LicenseType.FULL,
                    Header = Convert.ToByte(8),
                    Footer = Convert.ToByte(5),
                    ProductCode = (byte)ProductCode,
                    Edition = Edition.ENTERPRISE,
                    Version = 1
                };
                if (!KeyMan.GenerateKey(KeyVal, ref productkey))
                    ProdKeytxt.Text = "ERROR";
            }
            else
            {
                KeyVal = new KeyValuesClass()
                {
                    Type = LicenseType.TRIAL,
                    Header = Convert.ToByte(8),
                    Footer = Convert.ToByte(5),
                    ProductCode = (byte)ProductCode,
                    Edition = Edition.ENTERPRISE,
                    Version = 1,
                    Expiration =
DateTime.Now.Date.AddDays(Convert.ToInt32(txtExperience.Text))
                };
                if (!KeyMan.GenerateKey(KeyVal, ref productkey))
                    ProdKeytxt.Text = "ERROR";
            }
            ProdKeytxt.Text = productkey;
        }

        const int ProductCode = 1;

        private void Generatefrm(object sender, EventArgs e)
        {
            LicType.SelectedIndex = 0;
            ProdIDtxt.Text = ComputerInfo.GetComputerId();
```

```
        }
    }
}
```

Додаток В

Фрагмент вихідного коду реєстрації ключа програмного продукту

```
namespace LicenseKey
{
    public partial class Regfrm : Form
    {
        public Regfrm()
        {
            InitializeComponent();
        }

        const int ProductCode = 1;

        private void OKbtn(object sender, EventArgs e)
        {
            KeyManager KeyMan = new KeyManager(ProdIDtxt.Text);
            string prodkey = ProdKeytxt.Text;
            if(KeyMan.ValidKey(ref prodkey))
            {
                KeyValuesClass KeyVal = new KeyValuesClass();
                if (KeyMan.DisassembleKey(prodkey, ref KeyVal))
                {
                    LicenseInfo License = new LicenseInfo();
                    License.ProductKey = prodkey;
                    License.FullName = "Диплом Гайдук П.В.";
                    if (KeyVal.Type == LicenseType.TRIAL)
                    {
                        License.Day = KeyVal.Expiration.Day;
                        License.Month = KeyVal.Expiration.Month;
                        License.Year = KeyVal.Expiration.Year;
                    }
                    KeyMan.SaveSuretyFile(string.Format(@"{0}\Key.lic",
Application.StartupPath), License);
                    MessageBox.Show("Ваш продукт зареєстровано!", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                    this.Close();
                }
            }
```

```
            else
                MessageBox.Show("Ключ продукту не вірний!", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void FrmRegistration_Load(object sender, EventArgs e)
        {
            ProdIDtxt.Text = ComputerInfo.GetComputerId();
        }
    }
}
```

Додаток Г

Фрагмент вихідного коду інформації про програмний продукт

```
namespace LicenseKey
{

    public partial class frmAbout : Form
    {
        public frmAbout()
        {
            InitializeComponent();
        }

        private void OKbtn(object sender, EventArgs e)
        {
            this.Close();
        }

        const int ProdCode = 1;

        private void AboutFrm(object sender, EventArgs e)
        {
            ProdIDlbl.Text = ComputerInfo.GetComputerId();
            KeyManager KeyMan = new KeyManager(ProdIDlbl.Text);
            LicenseInfo License = new LicenseInfo();
            int value = KeyMan.LoadSuretyFile(string.Format(@"{0}\Key.lic",
Application.StartupPath), ref License);
            string prodk = License.ProductKey;
            if (KeyMan.ValidKey(ref prodk))
            {
                KeyValuesClass KeyVal = new KeyValuesClass();
```

```
            if (KeyMan.DisassembleKey(prodk, ref KeyVal))
            {
                ProdNamelbl.Text = "Диплом Гайдук П.В.";
                ProdKeylbl.Text = prodk;
                if (KeyVal.Type == LicenseType.TRIAL)
                    LicTypelbl.Text = string.Format("{0} days", (KeyVal.Expiration -
DateTime.Now.Date).Days);
                else
                    LicTypelbl.Text = "Full";
            }
        }
    }
}
```