МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПОЛІСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Факультет інформаційних технологій, обліку та фінансів
Кафедра комп'ютерних технологій
і моделювання систем

Кваліфікаційна робота
на правах рукопису

Дмитрук Данило Вікторович

УДК 004.056.55:004.42

КВАЛІФІКАЦІЙНА РОБОТА

Інформаційна технологія захисту від програм-вимагачів

(тема роботи)

126 «Інформаційні системи та технології»

(шифр і назва спеціальності)

Подається на здобуття освітнього ступеня бакалавр

кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____Дмитрук Д. В.

(підпис, ініціали та прізвище здобувача вищої освіти)

Керівник роботи
Молодецька Катерина Валеріївна
(прізвище, ім'я, по батькові)
доктор технічних наук, професор
(науковий ступінь, вчене звання)

Житомир – 2025

Висновок кафедри _____

за результатами попереднього захисту: _____

Протокол засідання кафедри _____

№ _____від «_____» _____20____р.

Завідувач кафедри _____

_____ _____ _____

(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батькові)

«_____» _____20____р.

Результати захисту кваліфікаційної роботи

Здобувач вищої освіти _____захистив (ла)

(прізвище ,ім'я, по батькові)

кваліфікаційну роботу з оцінкою:

сума балів за 100-бальною шкалою _____

за шкалою ECTS _____

за національною шкалою _____

Секретар ЕК

_____ _____ _____

(науковий ступінь, вчене звання) (підпис) (прізвище, ім'я, по батьков

АНОТАЦІЯ

Дмитрук Д. В. Інформаційна технологія захисту від програм-вимагачів – Кваліфікаційна робота на правах рукопису.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 126 – Інформаційні системи та технології. – Поліський національний університет, Житомир, 2025.

У роботі розглянуто актуальну проблему зростання кількості атак програм-вимагачів. Було запропоновано підходи для розробки інформаційної технології, яка забезпечить відповідний рівень захисту від даного типу атак.

У першому розділі було проаналізовано поняття та класифікацію програм-вимагачів, їхній механізм дії, а також найбільш розповсюджені приклади шкідливого програмного забезпечення цього типу

Другий розділ присвячено архітектурі розробленої інформаційної технології захисту, яка включає кілька взаємодіючих компонентів, зокрема графічну програму, утиліту для сканування файлів за правилами YARA та фонову програму на C++ для моніторингу загроз. Розроблено UML-діаграми, контекстну модель, діаграми активності, сценарії використання та функціональну декомпозицію системи.

Реалізацію всіх компонентів інформаційної технології описано в третьому розділі. Наведено скріншоти інтерфейсу користувача, приклади коду, використані бібліотеки, фреймворки та сервіс API VirusTotal. Розроблено інструкцію для кінцевого користувача, що дозволяє легко налаштувати систему захисту відповідно до власних потреб

Ключові слова: програми-вимагачі, інформаційна безпека, захист даних, інформаційна технологія, кібератаки.

SUMMARY

Dmytruk D. V. Information Technology for Protection Against Ransomware – Qualification work on manuscript rights.

Qualification thesis for the degree of Bachelor in specialty 126 – Information Systems and Technologies. – Polissia National University, Zhytomyr, 2025.

This work addresses the urgent issue of the growing number of ransomware attacks. It proposes approaches for the development of an information technology system that provides an appropriate level of protection against this type of threat.

The first chapter analyzes the concept and classification of ransomware, their operational mechanisms, and the most widespread examples of this type of malicious software.

The second chapter is devoted to the architecture of the developed protection technology, which includes several interacting components: a graphical application, a file scanning utility using YARA rules, and a background C++ application for threat monitoring. UML diagrams, a context model, activity diagrams, usage scenarios, and the system's functional decomposition are provided.

The third chapter describes the implementation of all components of the information technology system. It includes user interface screenshots, code samples, libraries and frameworks used, and integration with the VirusTotal API. A user guide was developed to enable easy configuration of the system according to individual needs.

Keywords: ransomware, information security, data protection, information technology, cyberattacks.

ЗМІС

Т

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення;

ІТ – інформаційні технології;

ОС – операційна система;

GUI – Graphical User Interface;

RaaS – Ransomware-as-a-Service;

RAT – Remote Access Trojan;

URL –Uniform Resource Locator;

TTP – Tactics, Techniques and Procedures;

UML – Unified Modeling Language;

API – Application Programming Interface**;**

GDPR – General Data Protection Regulation;

WinAPI – Windows API;

IDEF0 – Icam DEFinition for Function Modeling;

IDEF3 – Integrated DEFinition for Process Description Capture Method

ВСТУП

Актуальність теми дослідження**.** Сучасне суспільство дедалі більше залежить від інформаційних систем, що робить їх потенційними цілями для кіберзагроз, зокрема шкідливого програмного забезпечення. Програми-вимагачі (ransomware) шифрують дані або блокують доступ до системи з метою вимагання викупу [1]. Такі атаки спричиняють значні втрати: від зупинки критичних сервісів до витоку конфіденційної інформації й прямих фінансових збитків. Часто витрати на простої в кілька разів перевищують суму викупу, що стимулює зловмисників до вдосконалення тактик і активного поширення подібного ПЗ.

Згідно з дослідженням компанії SpaceLift, у 2023 році середня сума викупу становила 1,85 млн доларів, а вартість повного відновлення інфраструктури — 2,73 млн доларів; у 2024 ці показники зросли до 2,74 млн доларів [2]. Крім того, за даними Ponemon Institute, понад 88% організацій зазнали принаймні однієї ransomware-атаки [3]. Це зумовлює необхідність розробки проактивних, доступних та ефективних рішень, здатних адаптуватись до нових викликів у сфері кібербезпеки.

Мета кваліфікаційної роботи полягає у створенні інформаційної технології захисту від програм-вимагачів, яка поєднує сигнатурний, поведінковий і проактивний підходи, забезпечує моніторинг системних процесів і надає користувачеві візуальні підказки щодо зміцнення кіберзахисту.

Предметом дослідження **є** процес проєктування, розроблення та впровадження інформаційної технології, орієнтованої на підвищення рівня кіберстійкості в середовищі ОС Windows, включаючи підтримку застарілих систем (зокрема Windows XP і Server 2003).

Об'єктом дослідження є процес проектування та реалізації інформаційної технології, орієнтованої на підвищення рівня кібербезпеки ОС Windows, з урахуванням підтримки застарілих версій [4].

Наукова новизна роботи полягає в інтеграції кількох підходів до захисту (сигнатурного, поведінкового, проактивного) в межах єдиної технології, адаптованої до широкого діапазону операційних систем, що супроводжується функціональністю візуального контролю безпеки та інтеграцією з сервісом VirusTotal і механізмом локального сканування за YARA-правилами.

Практичне значення отриманих результатів полягає у створенні прикладної, гнучкої та доступної технології, яка може бути використана як у домашньому, так і в корпоративному середовищі. Розроблене рішення здатне ефективно працювати на обмежених ресурсах, забезпечує базовий рівень захисту та дає змогу користувачеві своєчасно виявляти й мінімізувати потенційні ризики.

Результати дослідження були апробовані на всеукраїнській та міжнародній конференціях:

Дмитрук Д.В. Дослідження методів виявлення та нейтралізації аномальної активності в інформаційних системах. *Цифрові технології в управлінні та адмініструванні.*2025. С.88-90.

Дмитрук Д.В. Впровадження YARA-правил як інструменту блокування підозрілих файлів під час ransomware-атак. *The Future of Science: Emerging Research and Technological Innovations*. 2025. С.84-87.

РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ РОЗРОБЛЕННЯ ТЕХНОЛОГІЇ ЗАХИСТУ ВІД ПРОГРАМ-ВИМАГАЧІВ

1.1 Поняття та класифікація ransomware

На рисунку 1.1, наведеному в додатку А, на основі відкритих аналітичних даних Panda Security представлено найнебезпечніші типи кіберзагроз, які прогнозуються як ключові у 2025 році [5].

Згідно з даними Palo Alto Network [6], сучасні програми-вимагачі використовують складні методи шифрування, такі як AES-256, що ускладнює відновлення даних без відповідного ключа. Крім того, деякі з них застосовують тактику "подвійного вимагання", коли перед шифруванням дані викрадаються, і зловмисники погрожують їх оприлюдненням у разі несплати викупу.

Роблячи висновки зі статей CrowdStrike та Palo Alto Networks [7, 8], програми-вимагачі можна класифікувати за кількома основними ознаками. За способом блокування вони поділяються на шифрувальники (crypto-ransomware), які шифрують файли користувача з метою вимагання викупу за їх розшифрування, та блокувальники (locker-ransomware), що не шифрують файли, але блокують доступ до системи або інтерфейсу, позбавляючи користувача можливості працювати з комп’ютером.

З огляду на методи поширення, ransomware може проникати в систему через індивідуальний підхід — найчастіше за допомогою фішингових листів або шкідливих вебсайтів. Часто достатньо одного зараженого комп’ютера в корпоративній мережі, щоб атака розповсюдилася далі. Іншим поширеним способом є експлуатація відомих уразливостей у програмному забезпеченні, які дозволяють зловмисникам автоматично проникати в систему без участі користувача.

Також програми-вимагачі класифікують за рівнем таргетування. Масові атаки спрямовані на широку аудиторію без конкретної мети — зазвичай це автоматизовані кампанії, які розповсюджуються одночасно великою кількістю

каналів. У свою чергу, цільові атаки націлені на конкретні організації або осіб і часто супроводжуються ретельною розвідкою, складним проникненням та персоналізованим шифруванням даних для максимального тиску на жертву.

Шифрувальники (crypto-ransomware) — це найбільш поширений тип ransomware. Вони шифрують файли на пристрої користувача за допомогою криптографічних алгоритмів (наприклад AES, ChaCha20, RSA, poly1305) і вимагають викуп за ключ дешифрування.

З технічної точки зору, механізм роботи програм-вимагачів є таким: кожен файл у комп'ютерній системі шифрується за допомогою симетричного криптографічного алгоритму, найчастіше AES, ChaCha20 або Salsa20. Для кожного окремого файлу генерується унікальний симетричний ключ, який, у свою чергу, шифрується із застосуванням асиметричного алгоритму, такого як RSA або алгоритми на основі еліптичних кривих. Розширення зашифрованого файлу зазвичай змінюється на задане в програмі-вимагачі, а в самому файлі записані данні (як правило зашифрований ключ шифрування та розширення) для повернення в розшифрований вид. Також аби повідомити про викуп зазвичай в кожній директорії або лише на робочому столі створюється файл з інструкцією як отримати програму для дешифрування.

У наш час оператори програм-вимагачів дедалі частіше діють як «тіньові бізнеси», орієнтовані на максимізацію прибутку. Це проявляється у все більш «клієнтоорієнтованому» підході до вимагання викупу. В інструкції зазвичай вказані канали для зв'язку як-от електронна пошта, чати на веб-ресурсах «тіньового інтернету», Telegram-боти. Такий підхід створює ілюзію «надійності» угоди з кіберзлочинцями, що схиляє компанії до сплати викупу.

Крім того, все частіше застосовується стратегія «подвійного вимагання» (double extortion) [9]**:** оператори не лише шифрують файли, але й крадуть конфіденційні дані перед цим. Погроза оприлюднення або продажу викрадених даних слугує додатковим важелем тиску на потерпілу сторону. У ряді випадків зафіксовано публікацію конфіденційних корпоративних документів, внутрішнього листування та фінансової звітності на спеціалізованих веб-

ресурсах, так званих leak-sites, доступ до яких зазвичай здійснюється через мережу Tor.

Яскравим прикладом такої стратегії є група LockBit, яка на момент 2023 року була однією з найактивніших груп і опублікували дані понад 1700 компаній на власному leak-сайті, коли жертви відмовлялись платити викуп [10]. На рисунку 1.2 додатка А показана кількість жертв у 2023 році від найпоширеніших операторів.

Варто згадати також такі шифрувальники масового розповсюдження як WannaCry та Petya. Ці види шкідливого ПЗ в свою чергу полягались лише на вразливість EternalBlue [11] і за думкою експертів в кібербезпеці дана стратегія не принесла фінансового успіху їх творцям, хоча й кількість жертв була колосальною: один лише WannaCry зміг заразити понад 200 000 систем у 150 країнах [12]. На рисунку 1.3 додатка А зображено приклад зараження комп'ютера вірусом WannaCry.

Існують також такі вимагачі як блокувальними (locker-ransomware), вони блокують доступ до інтерфейсу користувача (GUI), не шифруючи файли. Найчастіше блокують екран або виводять фальшиві повідомлення від поліції або державних органів.

WinLock блокував робочий стіл Windows і показував фальшиві повідомлення про порушення закону, вимагав SMS-платежі для розблокування [13]. Він розповсюджувався іншим вірусом-вимагачем який імітував повідомлення про активацію Windows Product Activation.В 2010 році угрупування яке стояло за ним було затримано в Москві і загалом було зароблено більш ніж 16 мільйонів доларів на даному типі шкідливого ПЗ (див. рис. 1.4, Дод.А).

Police Ransomware відображав підроблені повідомлення від ФБР або поліції ЄС, звинувачуючи жертв у перегляді забороненого контенту. На рисунку з додатка К зображено приклад вікна з вимаганням викупу WinLock.

Вразливість EternalBlue свого часу стала ключовим чинником масового поширення шифрувальників. Її варіації, зокрема EternalRomance,

EternalSynergy та EternalChampion, експлуатували протокол SMB (Server Message Block) [14], що широко використовувався в критичній інфраструктурі. Хоча Microsoft оперативно усунула ці вразливості, важливість своєчасного оновлення ОС залишається надзвичайно актуальною.

Схожа більш нова вразливість SMBGhost [15] також надавала можливість атакувати мережу, але вже SMB версії 3. Вразливість була присутня в Windows 10 версії 1903 та 1909 і в Windows Server тих самих версій. Хоча практики масового зараження комп'ютерних мереж, як в атаках WannaCry, ця вразливість не набула, але вона й досі активно використовується на застарілих ОС операторами кібератак.

Найбільш актуальним методом зараження на сьогодні є цільові атаки (targeted attacks)**,** що дедалі частіше використовуються операторами програм-вимагачів для проникнення в інфраструктуру організацій. Найпоширенішими варіаціями таких атак є фішингові листи та фішингові веб-ресурси**.**

Найбільш поширеним методом зараження сьогодні залишаються цільові атаки (targeted attacks), зокрема фішингові листи та веб-ресурси. Такі листи імітують офіційну комунікацію, змушуючи жертву відкрити шкідливе вкладення або перейти за посиланням, що веде до інфікування системи (через infostealer, RAT тощо) або до підробленого сайту для збору облікових даних.

Сучасні браузери значно покращили захист від подібних завантажень, зокрема за допомогою технологій Safe Browsing, цифрових підписів і репутаційних фільтрів. Це призвело до зниження ефективності прямих шкідливих завантажень. У відповідь зловмисники почали використовувати більш витончені методи: наприклад, імітацію перевірки Captcha під час якої користувач несвідомо активує JavaScript-скрипт із вбудованим зловмисним кодом [16, 17]. Розповсюдження таких веб-ресурсів в наші дні в основному реалізується через вищезгадані фішингові листи, SEO-оптимізацію веб-сайтів, а також через зловмисні рекламні компанії. Такі підходи свідчать про постійне вдосконалення технік соціальної інженерії та гнучкість зловмисників у

боротьбі з сучасними методами захисту. На рисунку 1.5 додатка А зображено приклад атаки ClickFix за допомогою імітації перевірки Captcha.

1.2 Аналіз існуючих платформ

На ринку IT в галузі забезпечення кібербезпеки існує низка комерційних та відкритих рішень для боротьби з програмами-вимагачами. До найбільш відомих належать антивірусні продукти, які використовують евристичний аналіз та поведінкові алгоритми, інструменти резервного копіювання і сервіси багатомодульного аналізу файлів.

Розглянемо такі антивірусні продукти як Microsoft Windows Defender (Windows Security) та Bitdefender Total Security.

Microsoft Windows Defender є одним із найпоширеніших антивірусів завдяки вбудованості в ОС Windows. Він забезпечує захист у реальному часі, контроль доступу до папок, виявлення експлойтів, інтеграцію з SmartScreen тощо. До його переваг належать безкоштовність і невелике споживання ресурсів, однак функціональність щодо аналізу інцидентів обмежена без корпоративної EDR-версії.

Bitdefender Total Security демонструє високу ефективність у боротьбі з ransomware-загрозами. Цей антивірус має модуль Ransomware Remediation, який автоматично створює резервні копії критичних файлів перед потенційною атакою. Також присутні функції поведінкового аналізу, захисту від мережевих атак, а також інтеграція із sandbox-оточенням для перевірки підозрілих файлів. До основних переваг можна віднести потужну багаторівневу фільтрацію загроз, ефективність захисту та автоматичне реагування на інциденти. Недоліком є дещо складне налаштування в корпоративному середовищі та потреба в постійному оновленні системних компонентів для підтримки актуальності захисту.

Одним з ефективних рішень для резервного копіювання є Acronis Cyber Protect. Це багатофункціональний інструмент, який поєднує в собі резервне копіювання, кіберзахист та управління системами. Серед основних функцій можна відзначити створення образів дисків, резервне копіювання на локальні носії, хмару або мережеві ресурси, автоматичне створення резервних копій за розкладом, перевірку резервних копій на цілісність та можливість швидкого відновлення навіть на інше обладнання.

Серед рішень для резервного копіювання доцільно виділити Acronis Cyber Protect, який об’єднує резервне копіювання, кіберзахист і централізоване управління. Програма підтримує різні ОС, зокрема Windows, Linux і віртуальні середовища, а також включає модуль виявлення ransomware у режимі реального часу. Основним недоліком є потреба в ресурсах і висока вартість для індивідуального використання..

VirusTotal є одним із найвідоміших і широко використовуваних сервісів для багатомодульного сканування файлів та URL-адрес. Цей інструмент дозволяє користувачам безкоштовно перевіряти підозрілі файли чи веб-ресурси за допомогою десятків антивірусних рушіїв та систем виявлення загроз. Після завантаження файлу або вставлення посилання, VirusTotal запускає одночасне сканування за допомогою більш ніж 70 антивірусів. На рисунку 1.6 додатка А зображено результати сканування програми-шифрувальника Babyk на VirusTotal.

Проте більшість існуючих рішень або вимагають значних ресурсів системи**,** або не сумісні зі старими версіями ОС, такими як Windows XP чи Server 2003. Крім того, багато утиліт мають вузький функціонал і не забезпечують комплексного підходу, зокрема не поєднують моніторинг системи, перевірку наявності ознак шкідливого ПЗ та рекомендації для покращення захисту.

1.3 Огляд шкідливого програмного забезпечення

LockBit - це один із найактивніших представників сучасного ransomware-as-a-service (RaaS), який постійно модифікується. LockBit 3.0 ("LockBit Black") підтримує функції видалення тіньових копій, обходу антивірусного ПЗ, шифрування локальних та мережевих ресурсів, та вивантаження даних перед шифруванням [18, 19]. Оператори активно використовують подвійне вимагання — і шифрування даних, і погрози оприлюднити їх на спеціальних leak-сайтах.

На своєму веб-сайті група LockBit заявляє що має найшвидшу швидкість шифрування серед списку інших шифрувальників на ринку. На рисунку 1.7 додатка А зображено таблицю порівняння швидкості шифрувальників на сайті LockBit.

Розпочавши виконання шифрувальник LockBit видаляє важливі записи та резервні копіювання з пристрою аби унеможливити відновлення даних з боку жертв.

Команда «vssadmin delete shadows /all /quiet» - видаляє всі тіньові копії з системи.

Команда «bcdedit /set {default} recoveryenabled no» - не дає можливості користувачеві ввійти до recovery boot ОС Windows 10.

LockBit генерує SID і використовує його для додавання себе до списку автозавантаження за допомогою запису реєстру в «HKCU\Software\Microsoft\Windows\CurrentVersion\Run». На рисунку 1.8 додатка А зображено с створення запису реєстру LockBit.

Також дане ПЗ асоціює зображення кожного зашифрованого файлу з власним, має можливість перечислення кожного з хостів мережі за допомогою протоколу SMB.

Babuk (або Babyk) — це зразок програми-вимагача, що з’явився на початку 2021 року та був націлений переважно на корпоративні мережі. Babuk використовував модель Ransomware-as-a-Service (RaaS), де оператори платформи надавали її афілійованим учасникам, які здійснювали атаки. Відомий здебільшого завдяки використанню сильного шифрування (Elliptic Curve Cryptography), цей зразок був здатен обходити звичайні методи захисту та швидко поширювався в локальних мережах. У травні 2021 року сталося внутрішнє розділення в угрупованні, після чого вихідний код Babuk був опублікований у відкритому доступі на форумі xss.is [20]. На рисунку 1.9 додатка А зображено оригінальний пост на форумі xss.is від автора вимагача.

Цей витік став значною подією, адже код пізніше був використаний іншими кіберзлочинцями для створення похідних версій. Babuk також є одним з перших, хто активно атакував VMware ESXi-сервери, що стало новим вектором у сфері корпоративних атак. В додатку А візуально розглянуто код шифрувальника (див. рис.1.10, Дод.А) та наведено фрагмент коду C++ відповідального за використання еліптичної кривої в ransomware (див. рис.1.11, Дод.А).

Ці приклади демонструють різноманітність тактик, технік та процедур (TTP) програм-вимагачів. Сучасні зразки часто поєднують функціонал бекдорів, інфостілерів, експлуатаційне ПЗ для вразливостей та компонентів для автономного поширення. Вони здатні обходити традиційний захист і вимагають більш складних проактивних та поведінкових методів виявлення і реагування. Саме це зумовлює необхідність створення нових захисних технологій, адаптованих до змін в кіберпросторі.

1.4 Постановка завдання

В приведених прикладах вище було розглянуто велику кількість кібератак із застосуванням програм-вимагачів, які завдають значної шкоди як

приватним користувачам, так і організаціям, виникає об’єктивна потреба у створенні ефективної, доступної та проактивної інформаційної технології захисту. Більшість наявних рішень зосереджуються на реактивному виявленні загроз після проникнення системи, в той час як значна частина шкідливого ПЗ встигає заподіяти шкоду ще до моменту спрацювання класичних сигнатурних захистів. Актуальність розробки полягає також у тому, що велика частина існуючих захисних рішень має обмеження: високе споживання ресурсів, недоступність для старіших версій Windows, недостатня гнучкість або надмірна складність у використанні для пересічного користувача.

У межах даного проєкту необхідно вирішити низку завдань, серед яких — аналіз існуючих рішень у сфері протидії програмам-вимагачам з метою виявлення їхніх переваг та недоліків. Наступним етапом є формування архітектури нової інформаційної технології з обов’язковим урахуванням підтримки операційних систем Windows, включаючи застарілі версії, такі як Windows XP та Windows Server 2003. Передбачається реалізація десктопного застосунку на базі WinForms (.NET Framework 4.5), який забезпечуватиме візуалізацію рівня захисту системи та керування параметрами безпеки. Також необхідно розробити Python-утиліту для вибіркового сканування файлів за YARA-правилами, що дозволить виявляти потенційно шкідливі об’єкти за характерними шаблонами. Одним з ключових компонентів є низькорівневий агент безпеки, створений на мові C++, який працює у фоновому режимі, здійснює моніторинг директорії завантажень, перевіряє процеси на підозрілу активність, взаємодіє з API VirusTotal та, за необхідності, автоматично запускається при старті системи згідно з параметрами конфігураційного файлу settings.json. Крім того, важливо забезпечити простоту налаштування системи для кінцевого користувача та провести тестування працездатності розробленого рішення в експериментальному середовищі з подальшим аналізом його ефективності.

Таким чином, кінцевий продукт має на меті надати користувачеві інструмент не лише для виявлення та нейтралізації загроз програм-вимагачів, але й для проактивної оцінки кібергігієни системи, що дозволить знизити ймовірність успішного зараження.

Висновки до першого розділу

У першому розділі проаналізовано основні аспекти проблеми захисту від програм-вимагачів: класифікацію, механізми дії, методи поширення та сучасні тактики зловмисників, зокрема «подвійне вимагання». Розглянуто приклади відомих загроз (LockBit, Babuk, WannaCry тощо) та ключові вразливості (EternalBlue, SMBGhost), що сприяли масштабним атакам.

Також проведено огляд сучасних захисних рішень — антивірусів, резервного копіювання та сервісів аналізу файлів. Встановлено їх обмеження щодо ресурсомісткості, підтримки старих ОС та комплексності захисту.

На підставі цього сформульовано завдання створити адаптивну інформаційну технологію, здатну поєднувати кілька підходів до виявлення загроз і забезпечити практичну користь для кінцевого користувача.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ДЛЯ ЗАХИСТУ ВІД ПРОГРАМ-ВИМАГАЧІВ

1.1 Розроблення архітектури інформаційної технології

Розроблена інформаційна технологія захисту від програм-вимагачів складається з кількох взаємопов'язаних модулів, які виконують специфічні функції. Архітектура системи реалізована на основі принципів модульності, масштабованості та сумісності із старими версіями Windows, що забезпечується використанням .NET Framework 4.5 та C++. Архітектура містить три основні частини:
- Інтерфейс користувача — реалізований на C# WinForm. Відображає стан захисту системи, надає рекомендації, дозволяє змінювати параметри захисту та вибірково сканувати за допомогою python-утиліти.
- Фоновий агент — реалізований мовою C++. Здійснює проактивний захист: моніторинг процесів, сканування файлів, комунікацію з VirusTotal API.

На основі створеної UML-діаграми описано такі основні сценарії використання:

- Перегляд стану безпеки та рекомендацій;

- Керування конфігураціями;

- Схвалення або блокування підозрілих процесів;

- Перевірка стану системи та активних загроз;

- Моніторинг завантажених файлів і взаємодія з VirusTotal API.

На рисунку 1.1 додатка Б міститься зображення UML-діаграми розробленої ІС.

Актори System та VirusTotal API взаємодіють із системою автоматично. Основна логіка контролюється користувачем, але частина функцій виконується автономно у фоновому режимі.

Умовно, робота системи розбивається на такі активності:

- Початкове зчитування конфігурацій;

- Перевірка стану системи (антивірус, тіньові копії, оновлення тощо);

- Паралельний моніторинг процесів і файлів;

- Обробка інцидентів та зворотній зв’язок із користувачем.

Станова модель включає основні стани: Очікування, Моніторинг, Попередження, Реакція, Завершення.

1.2 Функціональне моделювання інформаційної технології

Функціональне моделювання є одним із ключових етапів проєктування інформаційної технології, оскільки дозволяє чітко визначити функціональні блоки, сценарії їх взаємодії та процеси, що забезпечують досягнення цілей системи. У цьому розділі представлено пояснення до контекстної діаграми, декомпозиції IDEF0, IDEF3, а також сценарії використання системи (діаграми UML типу Use Case).

На схемі наведеній в рисунку 1.2 додатка Б було змодельовано типову послідовність дій системи під час запуску, моніторингу, реакції на загрози та формування рекомендацій для користувача. IDEF3 дозволяє представити поведінку системи у часі, враховуючи паралельні й послідовні процеси, а також взаємозв’язки між модулями.

Згідно з діаграмою IDEF3, основна логіка функціонування системи включає кілька послідовних етапів. На початковому етапі запуск системи супроводжується зчитуванням базової конфігурації (2), після чого виконується

ініціалізація основних модулів (3) та відображення графічного інтерфейсу користувача (4).

Далі реалізується механізм постійного моніторингу системи, який передбачає перевірку появи нових файлів (7) та запуску нових процесів (5), а також отримання атрибутів активних процесів (6) із подальшим порівнянням результатів із сервісом VirusTotal (9). Окремо здійснюється перевірка наявності активного антивірусного захисту (16), актуальності оновлень (15), стану резервного копіювання (18), функціонування шифрування дисків (17) та загального стану системи (14).

У разі виявлення потенційної загрози система може реагувати відповідно: дозволяється додавання підозрілих об'єктів до білого списку (11), видалення шкідливих компонентів (12), а також фіксація всіх виконаних дій у логах (13). Завершальним етапом є формування рекомендацій, яке базується на оцінці поточного стану системи та виявлених ризиків (19).

Для опису функціональної структури інформаційної технології захисту від програм-вимагачів була побудована контекстна діаграма IDEF0. Основна функція верхнього рівня — це «Захист системи від ransomware-атак». На цій діаграмі зазначено, що до системи надходить низка вхідних даних: поточний стан системи, результати попередніх перевірок, файли (зокрема потенційно шкідливі виконувані файли), а також конфігураційні параметри, що задаються користувачем у файлі «settings.json».

Забезпечення виконання основної функції відбувається за участі кількох ключових механізмів. Це, зокрема, агент захисту (UAgent), модуль перевірки процесів із реалізацією хукінгу, модуль рекомендацій, графічний інтерфейс користувача, API сервісу VirusTotal, а також вбудовані механізми захисту, що надаються операційною системою Windows. Суттєвий вплив на логіку роботи системи мають політики безпеки, які діють на рівні системи.

Робота системи обмежується чинними стандартами і політиками. Серед них — вимоги GDPR, стандарти ISO 27001, встановлені правила моніторингу, технічні обмеження самої системи, стратегія резервного копіювання та політика реагування на загрози. Всі ці фактори визначають межі та напрямки функціонування інформаційної технології.

В результаті виконання функцій система навдена в рисунку 1.3 додатка Б генерує вихідні дані, до яких належать попередження про виявлені загрози, рекомендації з підвищення рівня захисту, звіти про виконані перевірки, лог-файли з записами подій, оновлені налаштування безпеки, а також зібрані або змінені дані з системного реєстру.

Висновки до другого розділу

Другий розділ присвячено проєктуванню інформаційної технології захисту від програм-вимагачів, що поєднує сигнатурний, поведінковий і хмарний аналіз загроз. Архітектура системи враховує сумісність зі старими версіями Windows, модульність та зручність для користувача.

За допомогою UML-моделей сформовано структурну модель взаємодії між інтерфейсом, службами моніторингу та зовнішніми сервісами (VirusTotal API), що формалізує сценарії використання та забезпечує узгоджену роботу системи в реальному часі.

Функціональне моделювання за IDEF0 і IDEF3 дозволило детально проаналізувати логіку процесів, обробку подій, обмін даними і взаємодію з користувачем. Побудовано контекстну діаграму та декомпозицію ключових процесів: перевірка стану, сканування, генерація рекомендацій і реагування на загрози.

Таким чином, на основі розробленої архітектури сформовано комплексну інформаційну технологію, яка здатна виявляти шкідливу активність, оцінювати

рівень безпеки середовища та надавати кінцевому користувачеві релевантні поради для запобігання зараженню системи програмами-вимагачами. Отримані результати закладають основу для реалізації прототипу системи, що буде представлено у наступному розділі. Враховуючи методи виявлення аномальної активності в ІС було також написано публікацію [21].

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

1.1 Розроблення інтерфейсу користувача

При розробці інтерфейсу ключову увагу було приділено головному вікну звідки і відбувається керування іншими компонентами інформаційної системи. На рисунку 3.1 додатка В зображено головне вікно USecure в IDE Visual Studio.

При натисканні на підкомпоненти «Поради», «Проактивний захист», «Просканувати файл» та «Про ПЗ» компоненту Windows Forms «ToolStripMenuItem» відкриваються нові вікна з відповідним функціоналом. Відбувається це за допомогою стандартного методу ініціалізації об’єкту форми та виконаня методу Show() ініціалізованого об’єкту. На рисунку 3.2 додатка В зображено реалізацію відкриття нових вікон в USecure.

При завантаженні форми (метод Form2_Load()) спочатку виконується перевірка на предмет впровадження порад в систему, а потім перевірка на те скільки компонентів з налаштувань проактивного захисту було активовано. На основі цих даних формується висновок в відсотках. Важливе зауваження: ніяка система не може бути на 100% захищена, тому висновок про захищеність системи, який формується в змінній SystemProtectionPercentage завжди буде менший за 100. На рисунку 3.3 додатка В зображено частину коду перевірки порад та налаштувань проактивного захисту в методі Form2_Load(). А на рисунку 3.4 додатка В зображено частину коду відповідального за формування висновку і присвоювання його в елементи форми label2 та progressBar1.

Також для дизайну вікна було обрано відповідне зображення Icon, назву вікна обрано як «USecure». Для отримання фіксованого по розмірам, розташованого в центрі вікна було обрано відповідні параметри «MinimizeBox», «MaximizeBox», «FormBorderStyle» та «StartPosition» (для центрування вікна на екрані). Надалі відповідні параметри будуть використані

в інших вікнах з точки зору їх оптимальності для користувацького досвіду (див. рис. 3.5, Дод.В).

З суб'єктивної точки зору було створено також вікно з етичною порадою (див. рис. 3.6, Дод.В). Дане вікно можливо вимкнути натиснувши на checkBox1. При вході в інформаційну систему перевіряється відповідний змінений параметр в settins.json («helpzsu»).

При натисканні на кнопку button1 («Допомогти ЗСУ») викликається метод button1_Click(), що відкриває в браузері веб-сайт «https://savelife.in.ua/» за допомогою методу Process.Start().

При натисканні на кнопку button2 («Продовжити») відкривається головне вікно Form2 раніше описаним методом з ініціалізацією об'єкта Form2 і при активності checkBox1 записує параметр «helpzsu» до файлу налаштувань.

Форма порад складається з елементу dataGridView1 та label1, label2, label3.

При ініціалізації елементу dataGridView1 (на більш слабких пристроях можливо необхідно буде дочекатись кінця ініціалізації) за допомогою методів IsLatestUpdatesInstalled(), IsAntivirusInstalled(), IsFirewallInstalled(), IsBitLockerUsed() висновки до відповідних порад щодо встановлення останніх оновлень, встановлення антивірусу, встановлення фаєрволу, використання BitLocker («Виконано» або «Не виконано»). GetLastBackupShadowCopy() в свою чергу відповідає за пораду на рахунок зроблених тіньових копій: виводить інформацію щодо останньої тіньової копії або виводить повідомлення «Тіньових копій не знайдено» якщо ПЗ їх не вдалось знайти (див. рис. 3.7, Дод.В). В свою чергу label1, label2, label3 дають незалежні текстові поради, які не перевіряються та фіксуються програмним методом. На рисунку 3.8 додатка В зображено запущене вікно порад.

Вікно проактивного захисту складається з заголовку (label1), декількох checkbox та кнопки button1 («Примінити налаштування») відповідальних за редагування налаштувань даного типу захисту (див. рис. 3.9, Дод.В).

При завантаженні цієї форми (метод ProactiveProtectionForm_Load()) перевіряється наявність файлу налаштувань і при його наявності відповідні налаштування відображаються як активні чи неактивні відміткою в відповідальному за пункт налаштувань checkbox (див. рис. 3.10, Дод.В).

При натисканні на button1 («Примінити налаштування») відповідні налаштування зчитуються з checkbox і записуються в файл налаштувань.

Вікно вибіркової перевірки файлів було додано на більш пізніх етапах розробки як інструмент для гнучкішого сканування виконуваних файлів. Воно складається з кнопки для обрання файлу (button1), відображення шляху до обраного файлу (label1), кнопки для початку виконання перевірки (button3), label2 для заголовку «Результат», richTextBox1 для відображення результатів сканування python-утилітою та кнопки (button2) для копіювання результату до буферу обміну. (див. рис. 3.11, Дод.В)

При натисканні на button1 відкривається діалогове вікно для вибору виконуваного файлу і після обрання текст label1 змінюється на шлях (див. рис. 3.12, Дод.В).

При натисканні на button3 за допомогою об’єкту Process виконується python-скрипт pyyaradetector і вивід з даного інструменту записується в richTextBox1.

Вікно «Про ПЗ» в свою чергу складається лише з тексту з інформацією про програмне забезпечення (див. рис. 3.13, Дод.В).

3.2 Технічна реалізація інструментів для проактивного та сигнатурного захисту

Модуль UAgent відповідає за проактивний та сигнатурний захист інформаційної системи, він не має графічного інтерфейсу і працює в фоновому режимі. Написаний на C++ та складається з виконуваного файла UAgent.exe та динамічних бібліотек які знаходяться на стадії POC (proof-of-concept): agent32.dll, agent64.dll.

На початку за допомогою класу fstream бібліотеки std зчитується зміст файлу налаштувань settings.json (див. рис. 3.14, Дод.В). Потім за допомогою класу regex бібліотеки std виконується пошук параметрів і в глобальні змінні заносяться дані про параметри аби згодом вжити необхідних користувачеві дій (див. рис. 3.15, Дод.В).

Якщо активний параметр «agent_autostart», то виконується функція AddSelfToAutostart. В ній отримується повний шлях виконуваного модуля UAgent.exe за допомогою WinAPI функції GetModuleFileNameW [22] та створюється ключ «UAgent» зі значенням повного шляху в реєстрі зі шляхом «HKCU\ Software\Microsoft\Windows\CurrentVersion\Run» за допомогою функцій RegOpenKeyExW (для отримання дескриптора ключа), RegSetValueExW (для створення ключа зі значенням) та RegCloseKey (для закриття дескриптора ключа). На рисунку 3.16 додатка В зображено код функції AddSelfToStart.

При активності параметру «virustotal_downloads_send» створюється окремий поток для функції VTCheckDownloads (перевірка завантажених файлів на сервісі VirusTotal). Якщо ще не було проведено ін’єкції динамічної бібліотеки UAgent, то вона проводиться.

У функції VTCheckDownloads ми отримуємо шлях до директорії завантажень, а потім за допомогою FindFirstFile / FindNextFile / FindClose [23] виконуємо ітерацію по файлам та директоріям. Кожен файл відправляється через API VirusTotal за допомогою бібліотеки curl [24], при цьому до полів форми HTTP POST-запиту додається поле для API ключа користувача (параметр «apikey») та поле для самого файлу (параметр «file»).

Якщо встановлений параметр «agent_background_check» (перевірка підозрілих файлів та процесів в фоновому режимі), то виконується ін’єкція динамічної бібліотеки, оскільки даний функціонал не залежить від виконуваного файлу UAgent.

Параметр «agent_processes_access_check» відповідає за перевірку підозрілих процесів у системі. Потокова функція SuspiciousProcessesCheck

нескінченно кожні пів секунди (500 мілісекунд) виконує ітерацію по процесам і перевіряє їх відкриті дескриптори за допомогою іншої функції – LoopForProcesses (див. рис. 3.17, Дод.В).

Всередині функції LoopForProcesses виконується отримання snapshot за допомогою WinAPI функції CreateToolhelp32Snapshot [25] процесів на момент виконання коду і його ітерація за допомогою функцій Process32First та Process32Next [26, 27]. При цьому процеси які раніше проходили перевірку або мають назву зі списку дозволених пропускаються, в інакшому випадку – створюється окремий поток для сканування процесу задля збільшення швидкості сканування. Код даного потоку знаходиться в функції CheckOpenedFilesOfProcess. Всередині цієї функції за допомогою NtQuerySystemInformation отримується загальний список дескрипторів і якщо дескриптор належить до обраного для сканування процесу дублікується його дескриптор та отримується шлях до відкритого файлу аби згодом перевірити чи не містить він слів із списку зазначеного в коді SuspiciousProcessesCheck.cpp (змінна bannedSubStrings): якщо містить, то користувачу запропонується усунути загрозу (див. рис. 3.18, Дод.В).

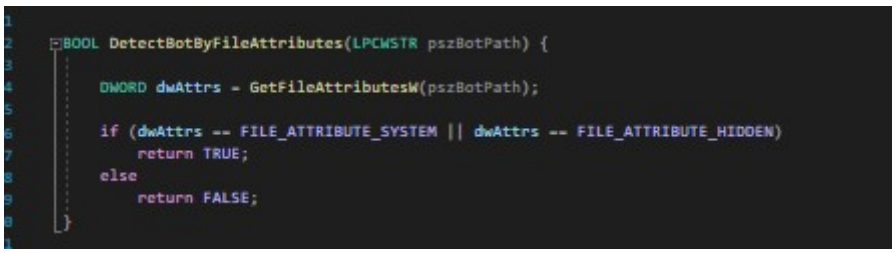
Загалом принцип роботи чітко розглядається в точці входу WinMain в програмі UAgent зображено на рисунку 3.19 додатка В.

Окремої уваги вартує реалізація динамічної бібліотеки UAgent (agent32.dll та agent64.dll). Існує можливість побудувати 32 та 64 бітні версії цього елементу ПЗ.

На початку роботи проходить вищеописане отримання параметрів, хоча в даному випадку отримуються лише два параметри необхідні для роботи ПЗ: «agent_background_check» та «virustotal_downloads_send». В відповідності до їх активності створюються два потоки за які відповідають функції BackgroundRoutine для параметру «agent_background_check» та HookingRoutine для «virustotal_downloads_send» відповідно (див. рис. 3.20 Дод.В).

Всередині функції потоку BackgroundRoutine в ще одному окремому потоці (в планах додати більше механізмів в кожному окремому потоці)

викликається один з механізмів протидії malware віддаленого доступу функцією BotKillerThread. В ній вищезгаданим методом ітерації процесів перевіряється чи не має виконуваний файл процесу підозрілий шлях (а саме CSIDL_APPDATA) і чи при цьому процес не має назву системного файлу (наприклад dllhost.exe чи explorer.exe) або чи не має при цьому виконуваний файл процесу підозрілі параметри (системний чи прихований). Якщо ж процес не проходить перевірку на підозрілість то користувачеві ІС пропонується усунути загрозу. На рисунку 3.21 додатка В зображено код функції перевірки назви процесу.



```
BOOL DetectBotByFileAttributes(LPCWSTR pszBotPath) {

    DWORD dwAttrs = GetFileAttributesW(pszBotPath);

    if (dwAttrs == FILE_ATTRIBUTE_SYSTEM || dwAttrs == FILE_ATTRIBUTE_HIDDEN)
        return TRUE;
    else
        return FALSE;
}
```

Рисунок 3.22 - Код функції перевірки атрибуту виконуваного файлу процесу за допомогою WinAPI функції GetFileAttributesW

Оскільки динамічна бібліотека знаходиться на стадії POC з погляду на великий об’єм розробки комплексного продукту ІС, то функція відповідальна за сканування сервісом VirusTotal (HookingRoutine), раніше реалізована в виконуваній програмі UAgent, планується бути реалізована в наступних релізах дещо видозміненим чином і в динамічній бібліотеці. Планується перехоплення функції створення процесів CreateProcess і аналіз кожного виконуваного файлу за допомогою сервіса VirusTotal..

Одним з найновіших компонентів на момент написання став python-скрипт для вибіркового сканування за правилами YARA. На цю тему було написано публікацію в конференції The Future of Science: Emerging Research and Technological Innovations [28].

Отримує як аргумент шлях до виконуваного файлу і у відповідь виводить результат сканування обраними правилами YARA. При цьому правила рекурсивно шукаються у обраних директоріях зі списку rule_dirs, що дозволяє

додавати власні правила, і скануються за допомогою бібліотеки yara [29]. В додатку Д міститься повний код описаного скрипта.

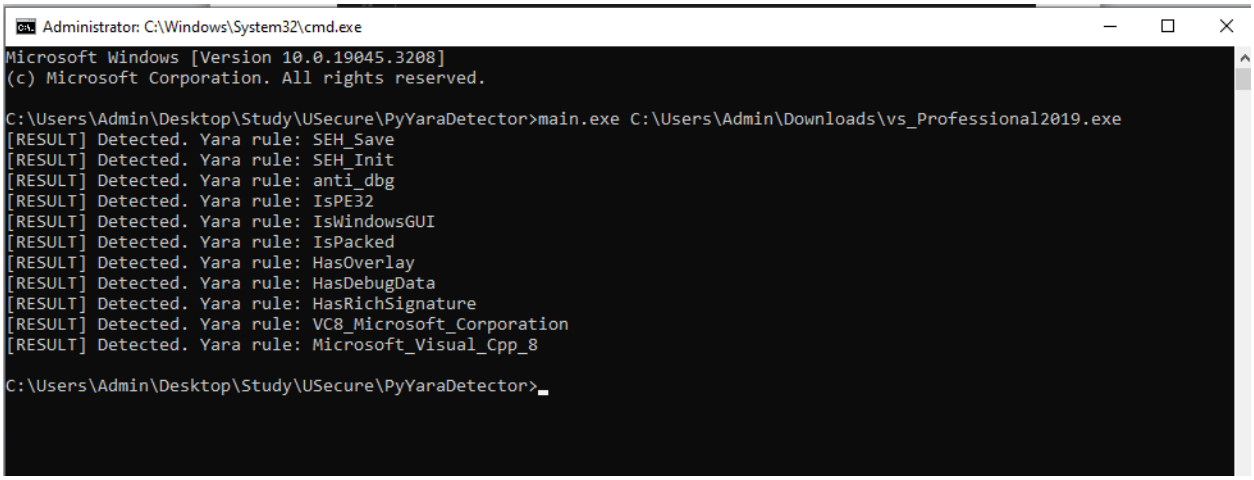


Рисунок.3.23 - Результат роботи скрипта

3.3 Інструкція для користувача

Розроблена інформаційна технологія захисту від програм-вимагачів є досить простою та гнучкою у використанні, хоча для максимальної ясності бажано ознайомитись з інструкцією для користувача.

Після встановлення шляхом розархівування архіву останнього релізу необхідно розпочати з запуску WinForms-застосунку USecure.exe.

На головному екрані користувач може ознайомитись з прогрес-баром рівня безпеки, який відображає умовну оцінку стану системи (в процентах) та обрати в стрічці меню зверху вікно з відповідним функціоналом.

При необхідності перевірити стан захисту ОС і вжити заходів відповідно до сформованих порад необхідно натиснути на «Поради» в верхній стрічці меню і в відкритому вікні будуть відображені поради та їх статуси.

Якщо існують сумніви щодо певного виконуваного файлу і існує необхідність його просканувати необхідно натиснути відповідно на

«Просканувати файл» і у відкритому вікні обрати файл для сканування та натиснути на кнопку «Перевірити»; за необхідності можна скопіювати результат сканування до буферу обміну за допомогою кнопки «Копіювати».

Аби редагувати активні налаштування проактивного захисту, які при старті будуть відключені необхідно натиснути на «Проактивний захист» у стрічці меню зверху головного вікна і відповідно до потреб активувати налаштування в checkbox та підтвердити натисканням кнопки «Примінити налаштування». Після цього, при необхідності тогочасної роботи модуля проактивного захисту UAgent, необхідно запустити виконуваний файл UAgent.exe. Модуль можна запустити і пізніше якщо нагальної необхідності не існує і додати його до запуску при старті системи (checkbox «Додання USecure Agent в автозавантаження»).

Аби ознайомитись з короткою інформацією щодо розробленої системи необхідно натиснути на пункт стрічкового меню «Про ПЗ» в головному вікні і у відкритому вікні буде представлена інформація.

Висновки до третього розділу

У третьому розділі було реалізовано функціональну інформаційну технологію захисту від програм-вимагачів, яка включає графічний застосунок USecure (на базі WinForms .NET Framework 4.5), Python-утиліту для сканування файлів за YARA-правилами, фоновий агент UAgent на C++ для моніторингу директорій, перевірки процесів і взаємодії з VirusTotal, а також систему конфігурації через файл settings.json. Додатково розроблено інструкцію користувача, що забезпечує простоту експлуатації. Система забезпечує сигнатурний і проактивний захист, візуалізацію рівня безпеки, а також підтримує застарілі ОС, що підвищує її універсальність і практичну цінність.

ВИСНОВКИ

У цій кваліфікаційній роботі було розроблено інформаційну технологію захисту від програм-вимагачів, яка поєднує сучасні підходи до кібербезпеки з практичними вимогами щодо зручності, гнучкості та сумісності з різними версіями операційних систем, зокрема застарілими.

Перший розділ присвячено аналізу загроз ransomware: охарактеризовано їхню класифікацію, методи дії та вектори поширення, а також наведено приклади реальних інцидентів. Встановлено, що наявні засоби захисту мають обмеження, які знижують їхню ефективність у нових умовах.

У другому розділі здійснено проєктування архітектури інформаційної технології. За допомогою UML-діаграм, моделей IDEF0 та IDEF3 визначено взаємодію між графічним інтерфейсом, фоновими модулями та зовнішніми сервісами, такими як VirusTotal.

У третьому розділі було реалізовано кожен із модулів проєкту, а саме десктопна WinForms-програма USecure, фоновий агент на C++ (UAgent), Python-утиліта та гнучкий конфігураційний механізм через settings.json.

Крім того, було створено інструкцію користувача, яка дає змогу експлуатувати систему без необхідності глибоких технічних знань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Програма-вимагач - що таке вірус-вимагач, як його видалити *ESET* : веб-сайт. URL: https://www.eset.com/ua/support/information/entsyklopediya-zahroz/prohrama-vymahach

2. 50+ Ransomware Statistics for 2025 *Spacelift* : веб-сайт. URL: https://spacelift.io/blog/ransomware-statistics

3. Study Reveals 88% of Companies Experienced a Ransomware Attack Last Year *HIPAA Journal* : веб-сайт. URL: https://www.hipaajournal.com/cost-of-a-ransomare-attack-study-2024/

4. DanyloDmytruk/USecure *GitHub* : веб-сайт. URL: https://github.com/DanyloDmytruk/USecure

5. 9 Key Cybersecurity Trends to Know in 2025 *Panda Security Mediacenter* : веб-сайт. URL: https://www.pandasecurity.com/en/mediacenter/cybersecurity-trends/

6. What Is Ransomware? *Palo Alto Networks* : веб-сайт. URL: https://www.paloaltonetworks.com/cyberpedia/what-is-ransomware

7. 5 Most Common Types of Ransomware *CrowdStrike* : веб-сайт. URL: https://www.crowdstrike.com/en-us/cybersecurity-101/ransomware/types-of-ransomware/

8. What Are the Most Common Types of Ransomware? *Palo Alto Networks* : веб-сайт. URL: https://www.paloaltonetworks.com/cyberpedia/what-are-the-most-common-types-of-ransomware

9. What Is Double Extortion Ransomware? *Zscaler* : веб-сайт. URL: https://www.zscaler.com/resources/security-terms-glossary/what-is-double-extortion-ransomware

10. Understanding Ransomware Threat Actors: LockBit *CISA* : веб-сайт. URL: https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-165a

11. Експлойт EternalBlue атакує з новими силами - дослідження ESET *ESET* : веб-сайт. URL: https://www.eset.com/ua/about/newsroom/press-releases/malware/uyazvimost-v-sluzhbakh-udalennogo-rabochego-stola-novaya-opasnost-dlya-organizatsiy-wannacryptor/

12. Вірус WannaCry заблокував 200 тис. комп'ютерів у 150 країнах *Портал новин LB.ua* : веб-сайт. URL: https://lb.ua/world/2017/05/14/366220_virus_wannacry_zablokiroval_200_tis.html

13. WinLock Ransomware *KnowBe4* : веб-сайт. URL: https://www.knowbe4.com/ransomware-knowledgebase/winlock

14. IBM Documentation *IBM* : веб-сайт. URL: https://www.ibm.com/docs/en/aix/7.2.0?topic=management-smb-protocol

15. SMBGhost – Analysis of CVE-2020-0796 *Trellix* : веб-сайт. URL: https://www.trellix.com/blogs/research/smbghost-analysis-of-cve-2020-0796/

16. Fake Docusign, Gitcode Sites Spread NetSupport RAT via Multi-Stage PowerShell Attack *The Hacker News* : веб-сайт. URL: https://thehackernews.com/2025/06/fake-docusign-gitcode-sites-spread.html

17. Microsoft Warns of ClickFix Phishing Campaign Targeting Hospitality Sector via Fake Booking[.]com Emails *The Hacker News* : веб-сайт. URL: https://thehackernews.com/2025/03/microsoft-warns-of-clickfix-phishing.html

18. Malware Evolution - Analyzing LockBit 2.0 *Cynet* : веб-сайт. URL: https://www.cynet.com/attack-techniques-hands-on/malware-evolution-analyzing-lockbit-2-0/

19. LockBit Analysis *Truesec* : веб-сайт. URL: https://www.truesec.com/hub/blog/lockbit-analysis

20. Babuk ransomware's full source code leaked on hacker forum *BleepingComputer* : веб-сайт. URL: https://www.bleepingcomputer.com/news/security/babuk-ransomwares-full-source-code-leaked-on-hacker-forum/

21. Дмитрук Д.В. Дослідження методів виявлення та нейтралізації аномальної активності в інформаційних системах. *Цифрові технології в управлінні та адмініструванні*.2024. С.88-90

22. GetModuleFileNameW function (libloaderapi.h) *MSDN* : веб-сайт. URL: https://learn.microsoft.com/en-us/windows/win32/api/libloaderapi/nf-libloaderapi-getmodulefilenamew

23. Listing the Files in a Directory *MSDN* : веб-сайт. URL: https://learn.microsoft.com/en-us/windows/win32/fileio/listing-the-files-in-a-directory

24. curl *curl* : веб-сайт. URL: https://curl.se/

25. CreateToolhelp32Snapshot function (tlhelp32.h) *MSDN* : веб-сайт. URL: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-createtoolhelp32snapshot

26. Process32First function (tlhelp32.h) *MSDN* : веб-сайт. URL: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-process32first

27. Process32Next function (tlhelp32.h) *MSDN* : веб-сайт. URL: https://learn.microsoft.com/en-us/windows/win32/api/tlhelp32/nf-tlhelp32-process32next

28. Дмитрук Д.В. Впровадження YARA-правил як інструменту блокування підозрілих файлів під час ransomware-атак. *The Future of Science: Emerging Research and Technological Innovations*. 2025. С.84-87

29. Using YARA from Python *yara 4.4.0 documentation* : веб-сайт. URL: https://yara.readthedocs.io/en/stable/yarapython.html

ДОДАТКИ

ДОДАТОК А

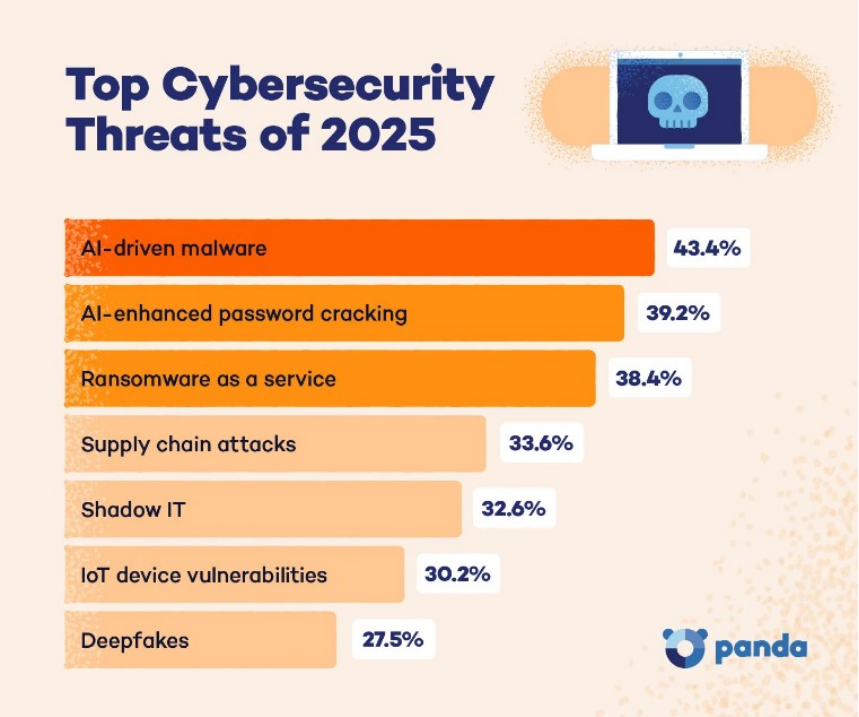


Рисунок 1.1 - Найнебезпечніші види кіберзагроз за 2025 рік

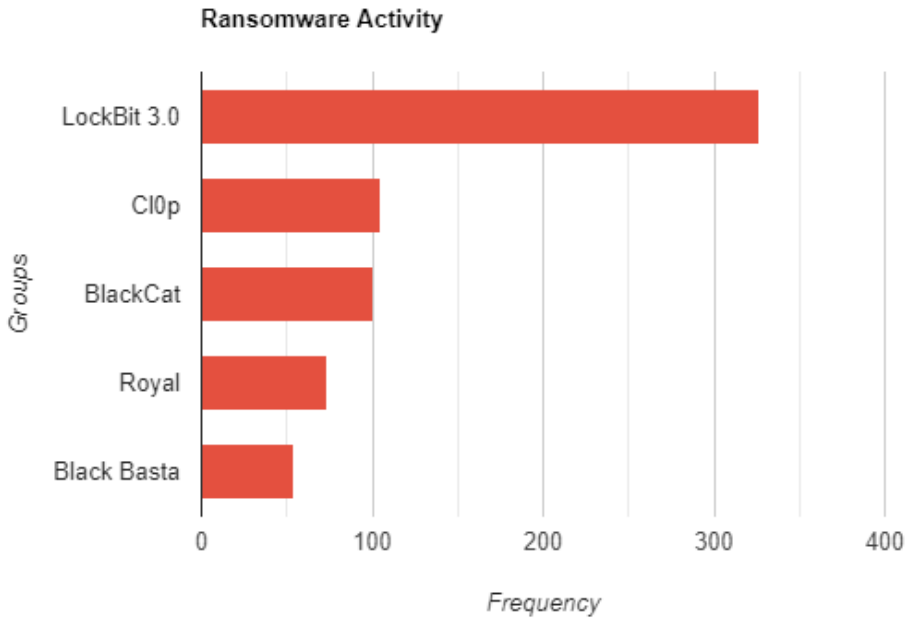


Рисунок 1.2 - Кількість жертв у 2023 році

Рисунок 1.3

Рисунок 1.4

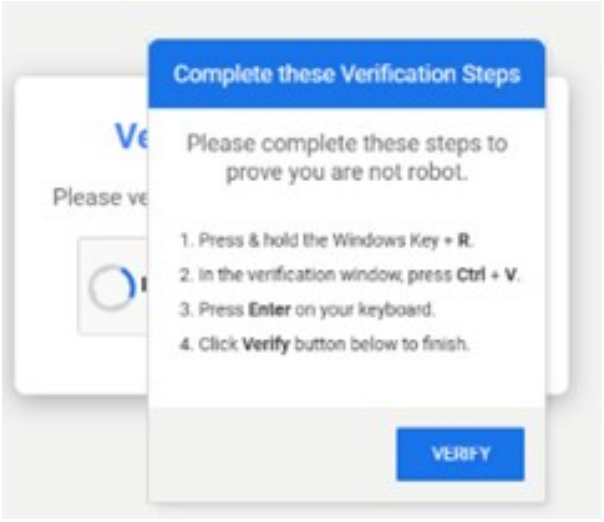


Рисунок 1.5

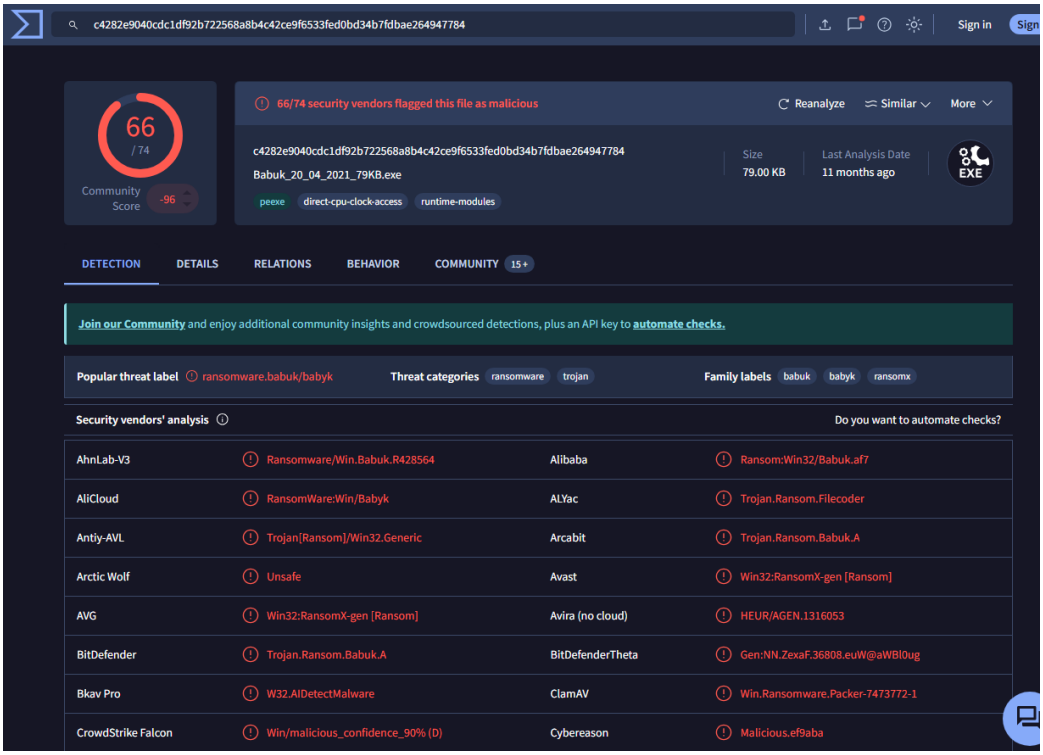


Рисунок 1.6

Encryption speed comparative table for some ransomware - 02.08.2021 (added BlackMatter)

PC for testing: Windows Server 2016 x64 \ 8 core Xeon E5-2680@2.40GHz \ 16 GB RAM \ SSD

Name of the ransomware	Date of a sample	Speed in megabytes per second	Time spent for encryption of 100 GB	Time spent for encryption of 10 TB	Self spread	Size sample in KB	The number of the encrypted files (All file in a system 257472)
LOCKBIT 2.0	5 Jun, 2021	373 MB/s	4M 28S	7H 26M 40S	Yes	855 KB	109964
LOCKBIT	14 Feb, 2021	266 MB/s	6M 16S	10H 26M 40S	Yes	146 KB	110029
Cuba	8 Mar, 2020	185 MB/s	9M	15H	No	1130 KB	110468
BlackMatter	2 Aug, 2021	185 MB/s	9M	15H	No	67 KB	111018
Babuk	20 Apr, 2021	166 MB/s	10M	16H 40M	Yes	79 KB	109969
Sodinokibi	4 Jul, 2019	151 MB/s	11M	18H 20M	No	253 KB	95490
Ragnar	11 Feb, 2020	151 MB/s	11M	18H 20M	No	40 KB	110651
NetWalker	19 Oct, 2020	151 MB/s	11M	18H 20M	No	902 KB	109892
MAKOP	27 Oct, 2020	138 MB/s	12M	20H	No	115 KB	111002
RansomEXX	14 Dec,2020	138 MB/s	12M	20H	No	156 KB	109700
Pysa	8 Apr, 2021	128 MB/s	13M	21H 40M	No	500 KB	108430
Avaddon	9 Jun, 2020	119 MB/s	14M	23H 20M	No	1054 KB	109952
Thanos	23 Mar, 2021	119 MB/s	14M	23H 20M	No	91 KB	81081

Рисунок 1.7

Class:	Registry
Operation:	RegSetValue
Result:	SUCCESS
Path:	HKCU\Software\Microsoft\Windows\CurrentVersion\Run\{D568D8EB-2424-4974-C3B9-C312099CCE75}
Duration:	0.0000837

Type:	REG_SZ
Length:	72
Data:	"C:\Users\John\Desktop\Lockbit.exe"

Рисунок 1.8

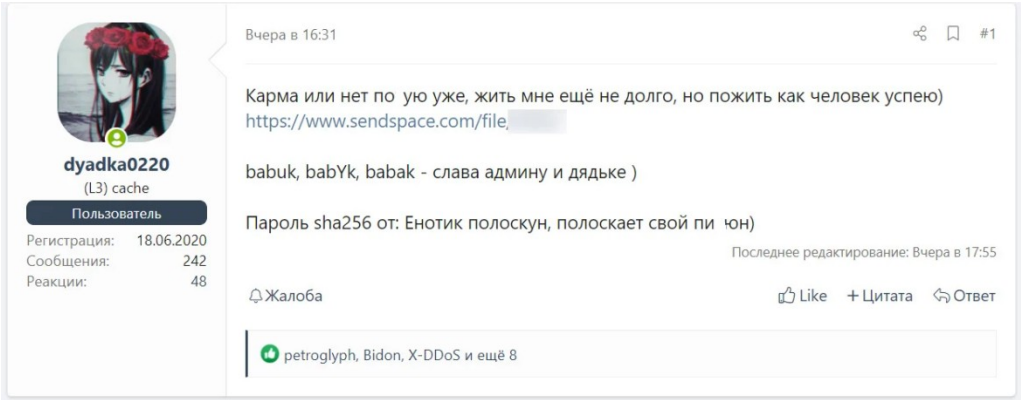


Рисунок 1.9

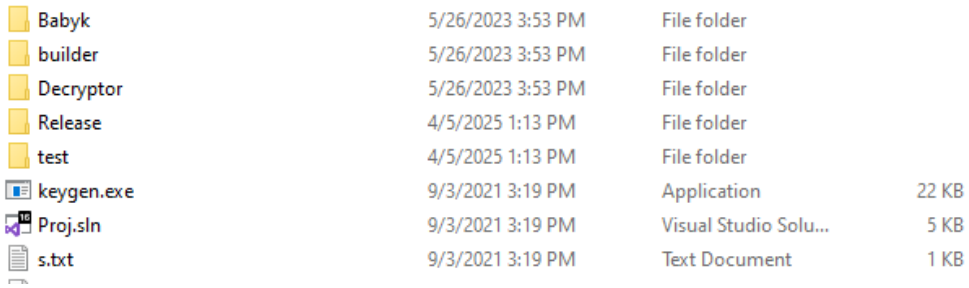


Рисунок 1.10 - Код шифрувальника Babyk

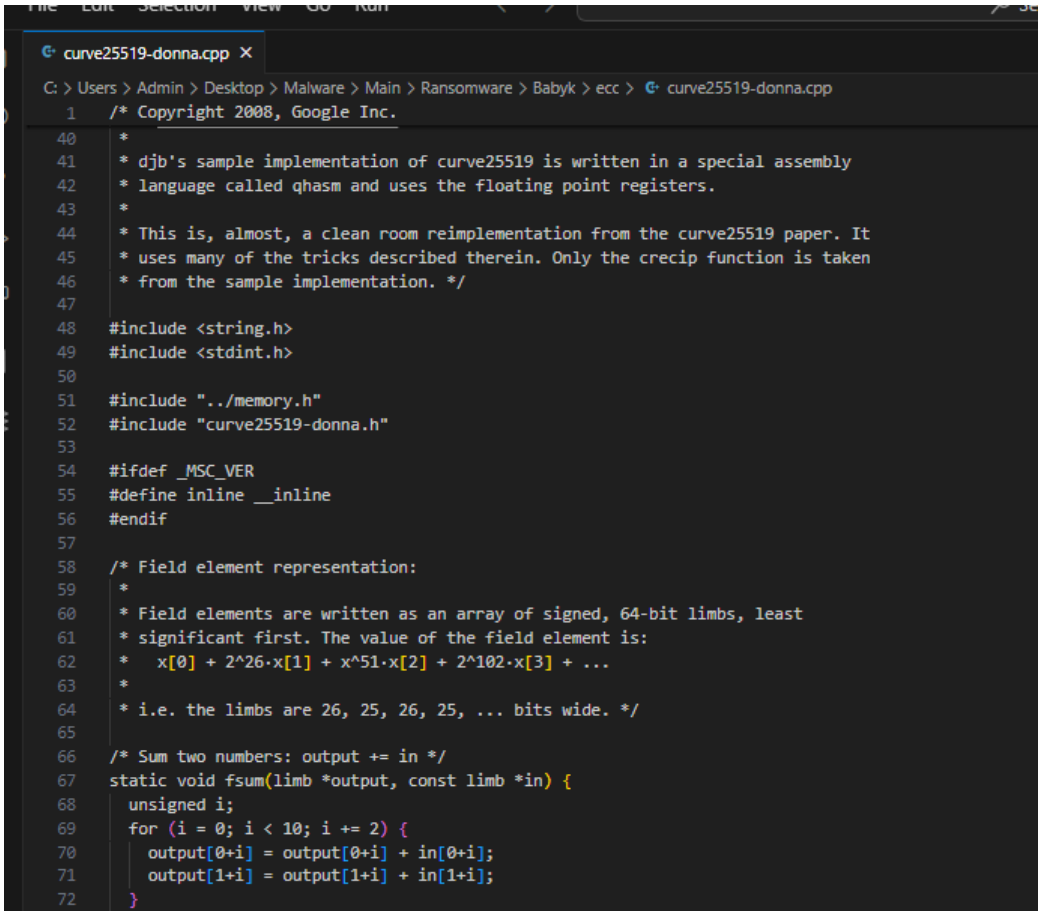


Рисунок 1.11 - Фрагмент коду відповідальний за використання еліптичної кривої в Babyk

ДОДАТОК Б

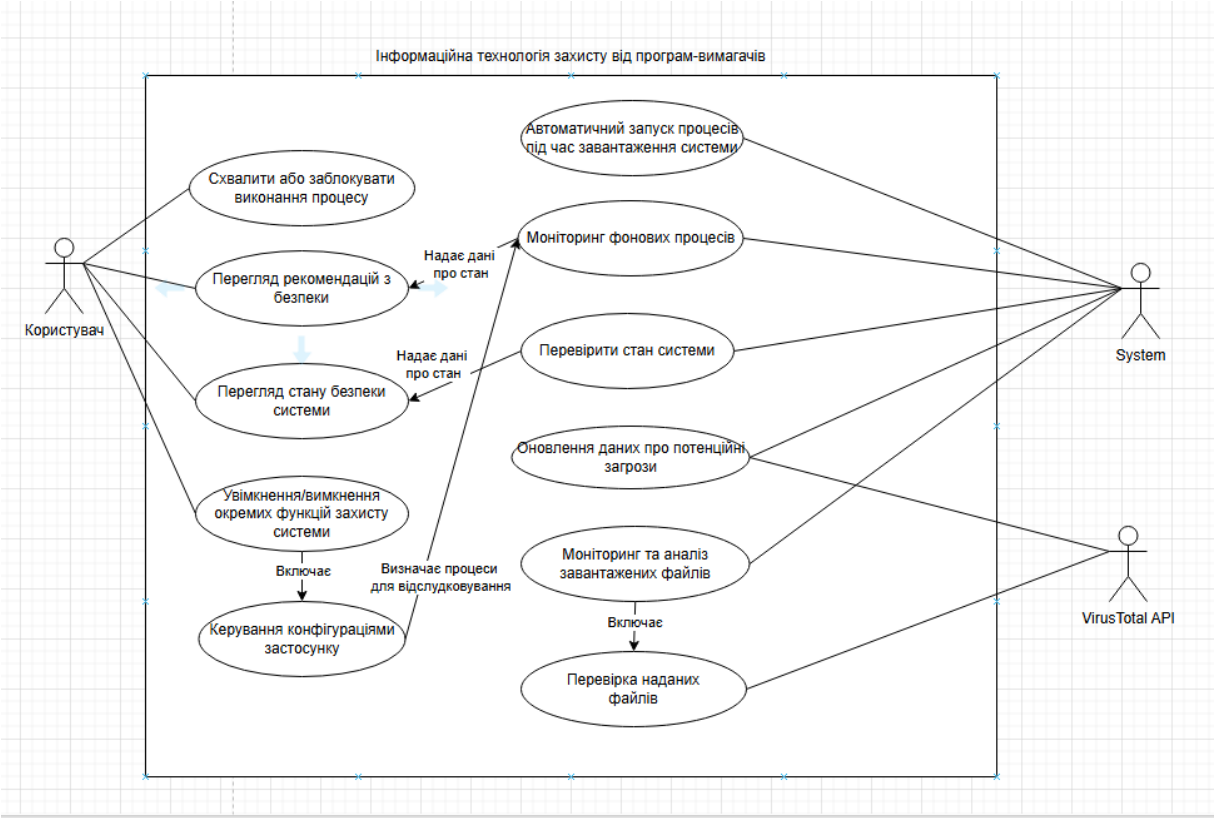


Рисунок 1.1

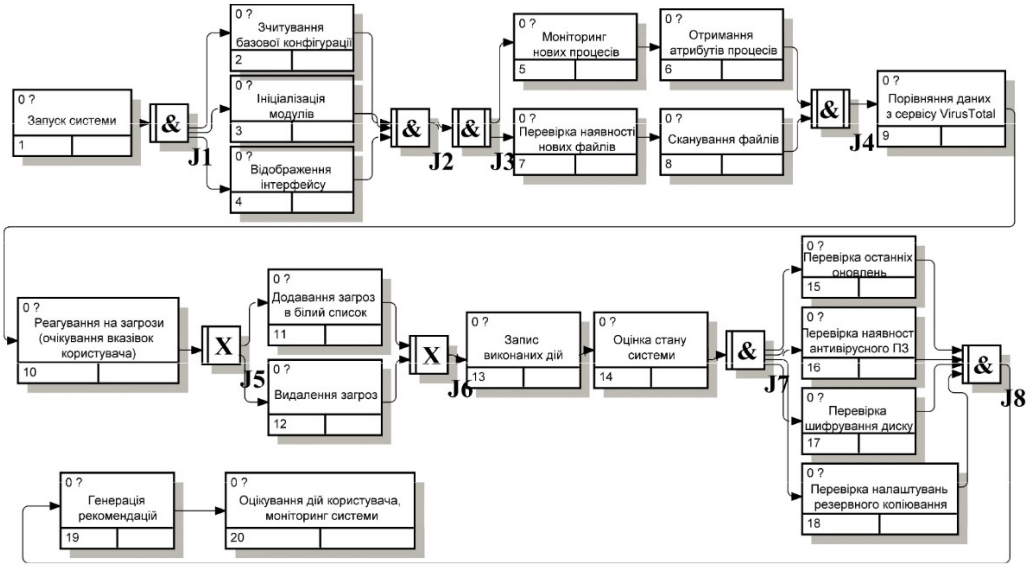


Рисунок 1.2 - IDEF3 діаграма USecure

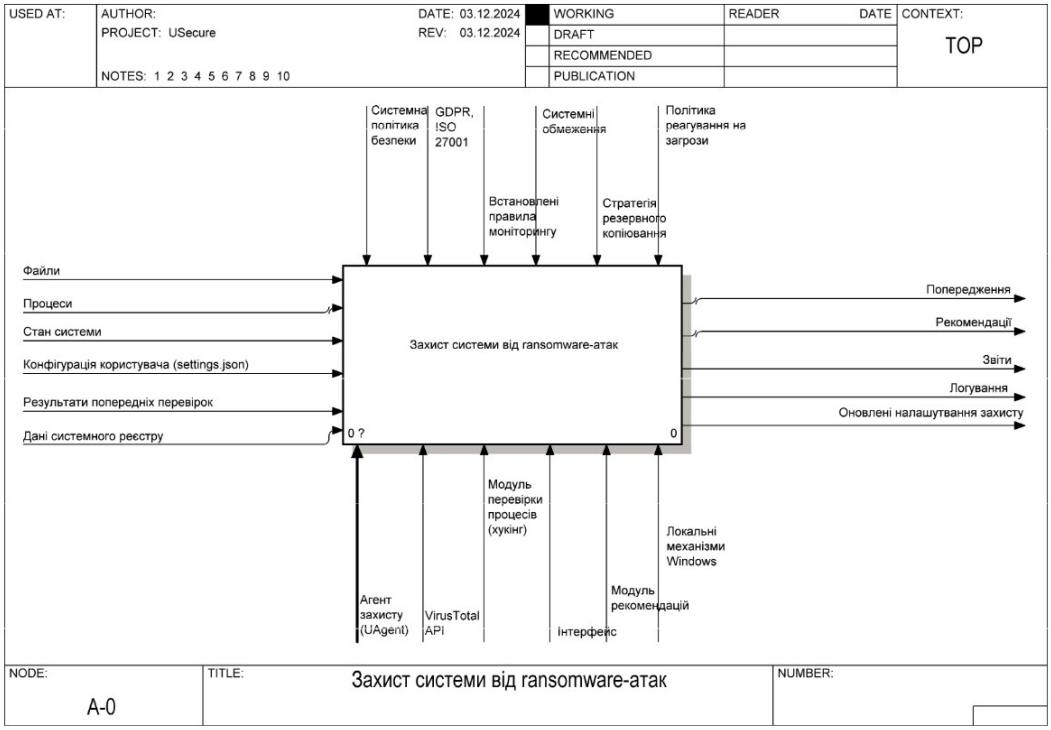


Рисунок 1.3 - IDEF0 діаграма USecure

ДОДАТОК В

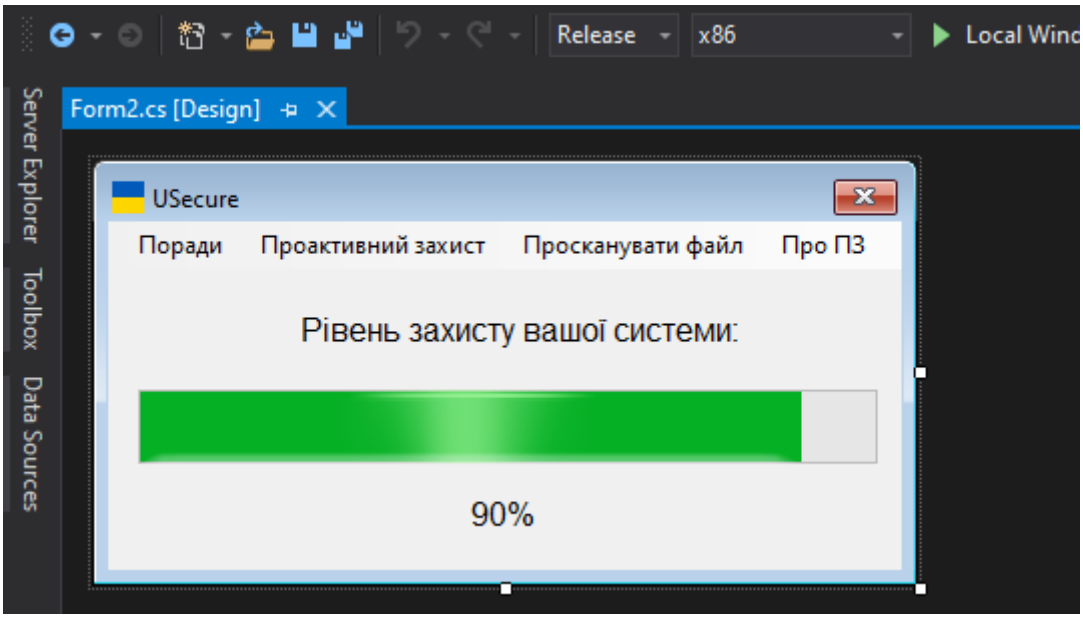
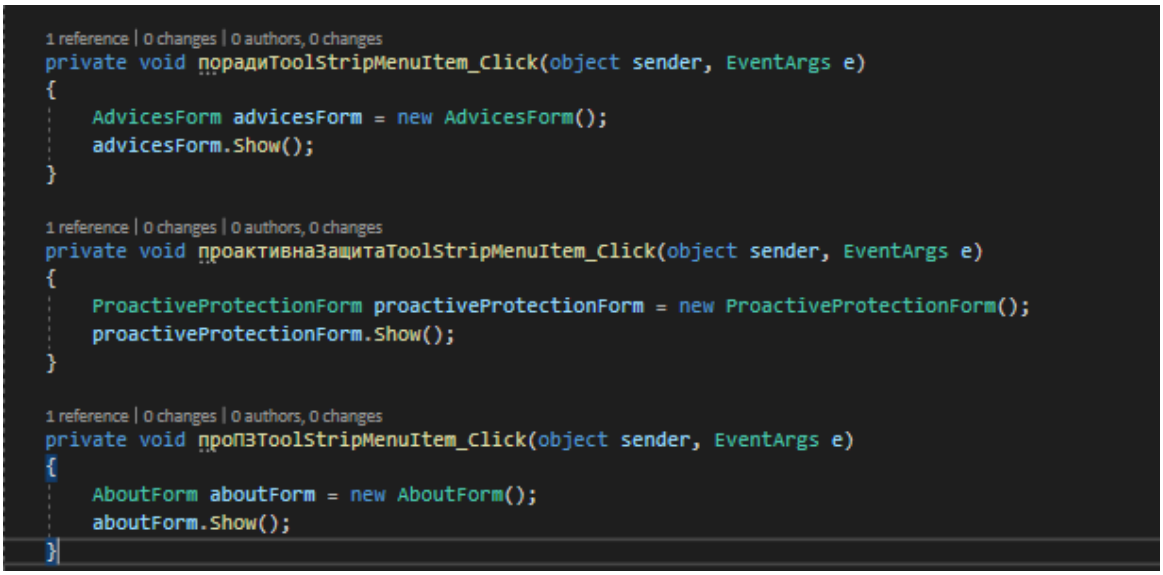
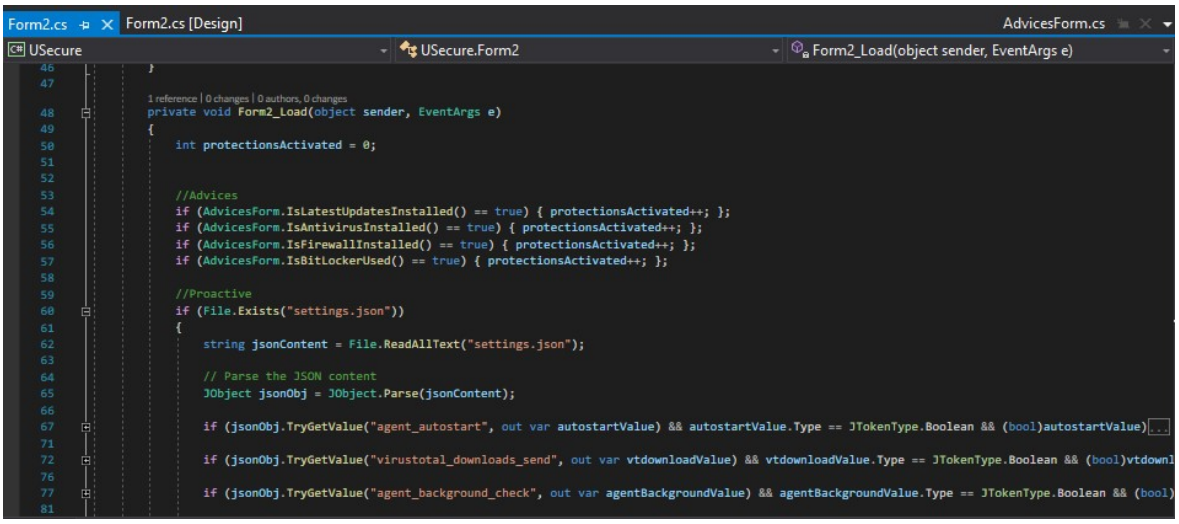


Рисунок 3.1



```
1 reference | 0 changes | 0 authors, 0 changes
private void порадиToolStripMenuItem_Click(object sender, EventArgs e)
{
    AdvicesForm advicesForm = new AdvicesForm();
    advicesForm.Show();
}

1 reference | 0 changes | 0 authors, 0 changes
private void проактивнаЗащитаToolStripMenuItem_Click(object sender, EventArgs e)
{
    ProactiveProtectionForm proactiveProtectionForm = new ProactiveProtectionForm();
    proactiveProtectionForm.Show();
}

1 reference | 0 changes | 0 authors, 0 changes
private void проПЗToolStripMenuItem_Click(object sender, EventArgs e)
{
    AboutForm aboutForm = new AboutForm();
    aboutForm.Show();
}
```

Рисунок 3.2



```
Form2.cs | Form2.cs [Design] | AdvicesForm.cs
USecure | USecure.Form2 | Form2_Load(object sender, EventArgs e)

        }

        1 reference | 0 changes | 0 authors, 0 changes
        private void Form2_Load(object sender, EventArgs e)
        {
            int protectionsActivated = 0;


            //Advices
            if (AdvicesForm.IsLatestUpdatesInstalled() == true) { protectionsActivated++; };
            if (AdvicesForm.IsAntivirusInstalled() == true) { protectionsActivated++; };
            if (AdvicesForm.IsFirewallInstalled() == true) { protectionsActivated++; };
            if (AdvicesForm.IsBitLockerUsed() == true) { protectionsActivated++; };

            //Proactive
            if (File.Exists("settings.json"))
            {
                string jsonContent = File.ReadAllText("settings.json");

                // Parse the JSON content
                JObject jsonObj = JObject.Parse(jsonContent);

                if (jsonObj.TryGetValue("agent_autostart", out var autostartValue) && autostartValue.Type == JTokenType.Boolean && (bool)autostartValue)...

                if (jsonObj.TryGetValue("virustotal_downloads_send", out var vtdownloadValue) && vtdownloadValue.Type == JTokenType.Boolean && (bool)vtdownl

                if (jsonObj.TryGetValue("agent_background_check", out var agentBackgroundValue) && agentBackgroundValue.Type == JTokenType.Boolean && (bool)
```

Рисунок 3.3

```
    //Summary:
    int SystemProtectionPercentage = (int)Math.Round((((float)protectionsActivated / 8.0) * 100));

    if(SystemProtectionPercentage == 100)
    {
        SystemProtectionPercentage--; //There's no absolutely protected system
    }

    label2.Text = SystemProtectionPercentage.ToString()+"%";
    progressBar1.Value = SystemProtectionPercentage;


}
```

Рисунок 3.4

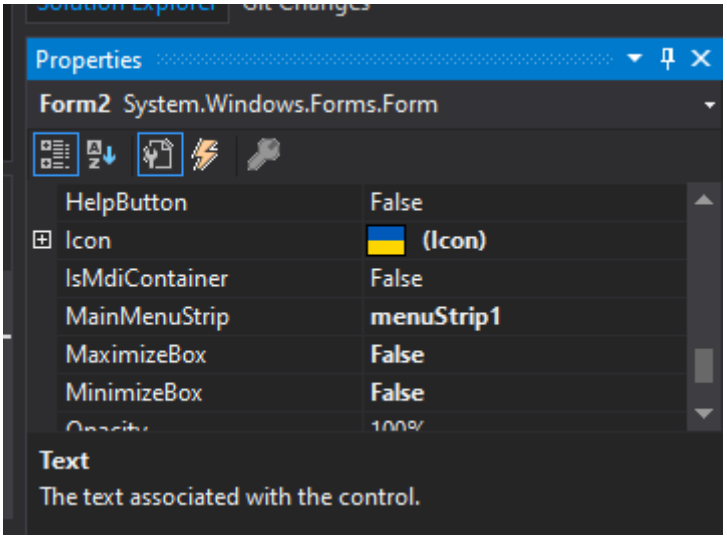


Рисунок 3.5 - Вікно параметрів головного вікна Form2

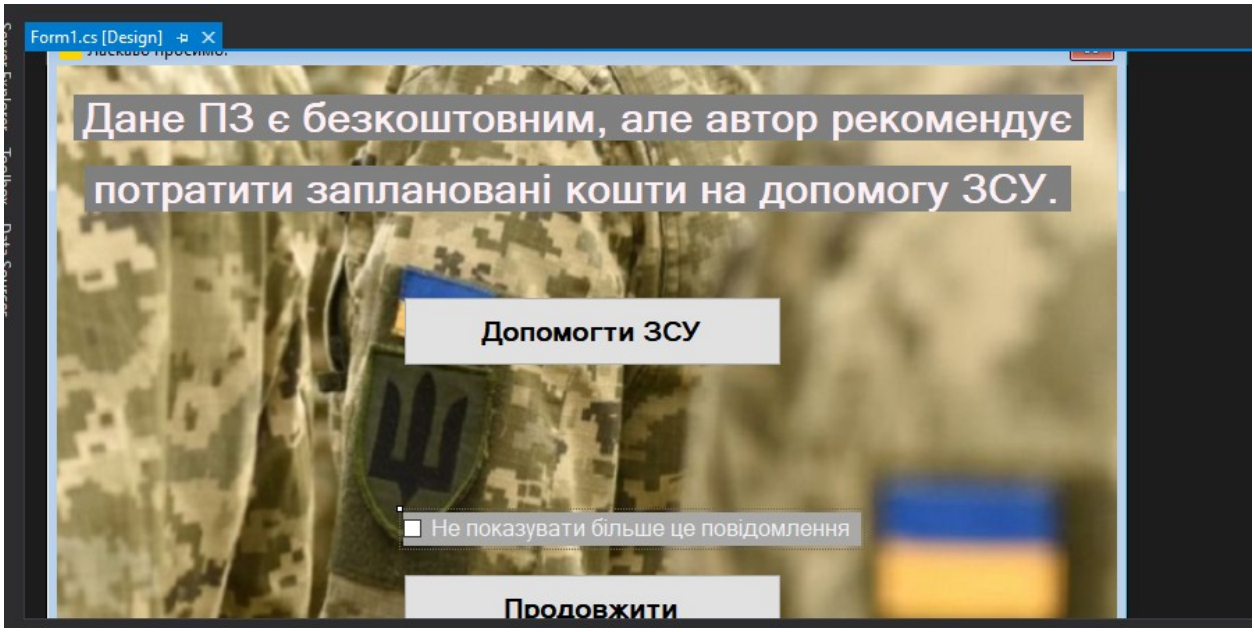


Рисунок 3.6

```
1 reference | 0 changes | 0 authors, 0 changes
private void InitializeDataGridView()
{
    // Create columns
    DataGridViewTextBoxColumn column1 = new DataGridViewTextBoxColumn();
    column1.HeaderText = "Порада";
    column1.Name = "AdviceColumn";

    DataGridViewTextBoxColumn column2 = new DataGridViewTextBoxColumn();
    column2.HeaderText = "Статус";
    column2.Name = "StatusColumn";

    // Add columns to DataGridView
    dataGridView1.Columns.Add(column1);
    dataGridView1.Columns.Add(column2);

    // Add sample data
    dataGridView1.Rows.Add("Встановлення останніх оновлень системи", (IsLatestUpdatesInstalled()) ? "Виконано ✓" : "Не виконано ✗");
    dataGridView1.Rows.Add("Встановлення антивірусного ПЗ", (IsAntivirusInstalled()) ? "Виконано ✓" : "Не виконано ✗");
    dataGridView1.Rows.Add("Використання фаєрвола", (IsFirewallInstalled()) ? "Виконано ✓" : "Не виконано ✗");
    dataGridView1.Rows.Add("Шифрування диску (BitLocker)",(IsBitLockerUsed()) ? "Виконано ✓" : "Не виконано ✗");
    dataGridView1.Rows.Add("Резервне копіювання (остання тіньова копія)", GetLastBackupShadowCopy());

    // Set DataGridView properties
    dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dataGridView1.AllowUserToAddRows = false; // Optional: Disable the extra "new row" at the bottom
}
```

Рисунок 3.7 - Реалізація ініціалізації елементу dataGridView1

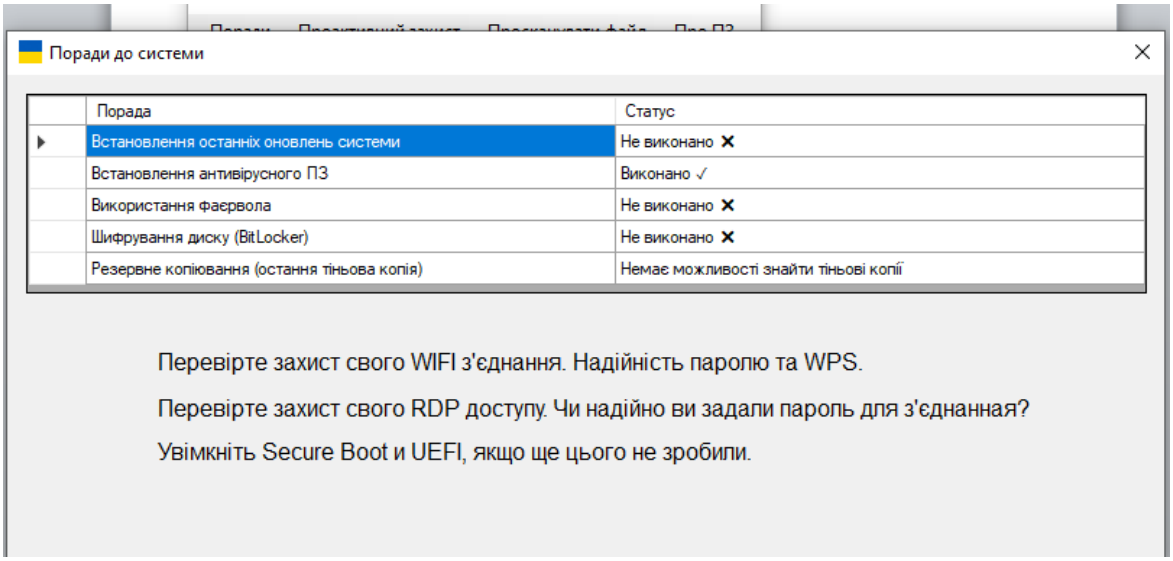


Рисунок 3.8 - Запущене вікно порад

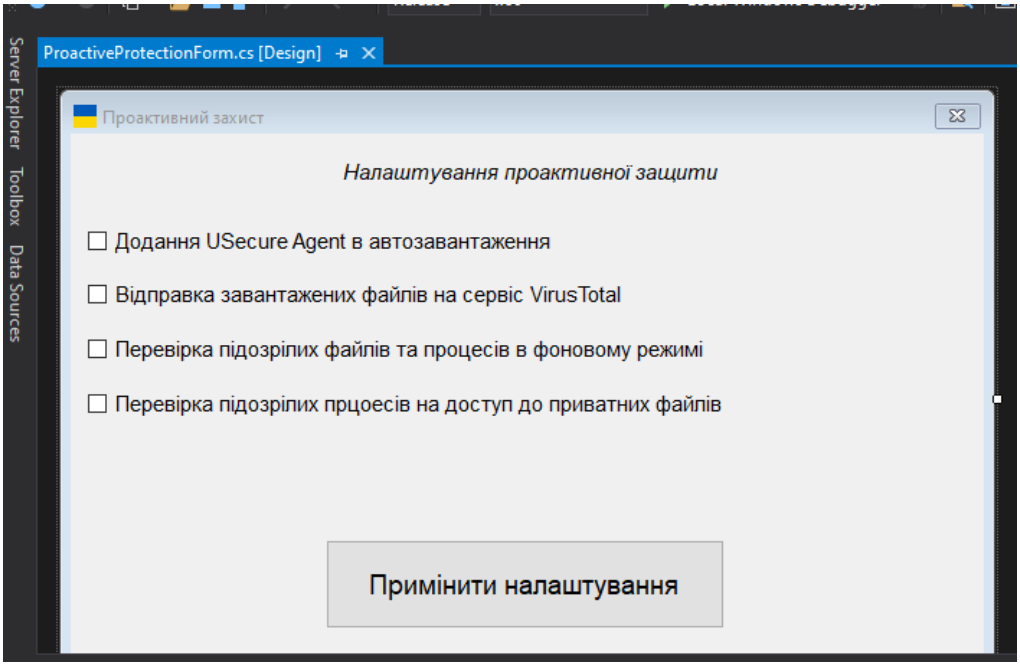


Рисунок 3.9 - Вікно проактивного захисту ProactiveProtectionForm

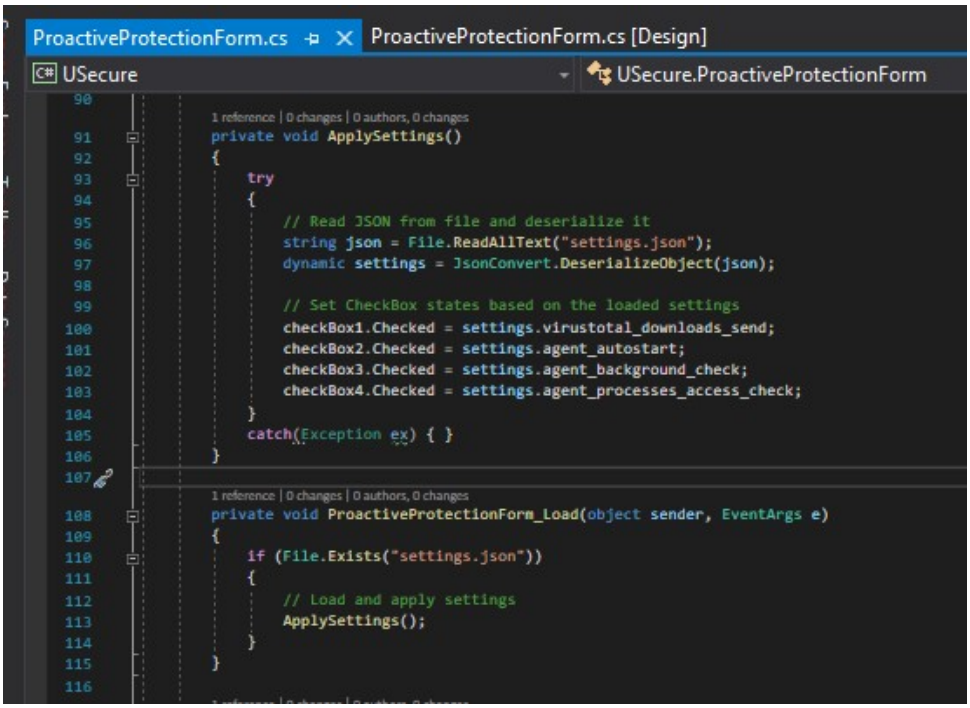


Рисунок 3.10 - Код відповідальний за відображення активних налаштувань

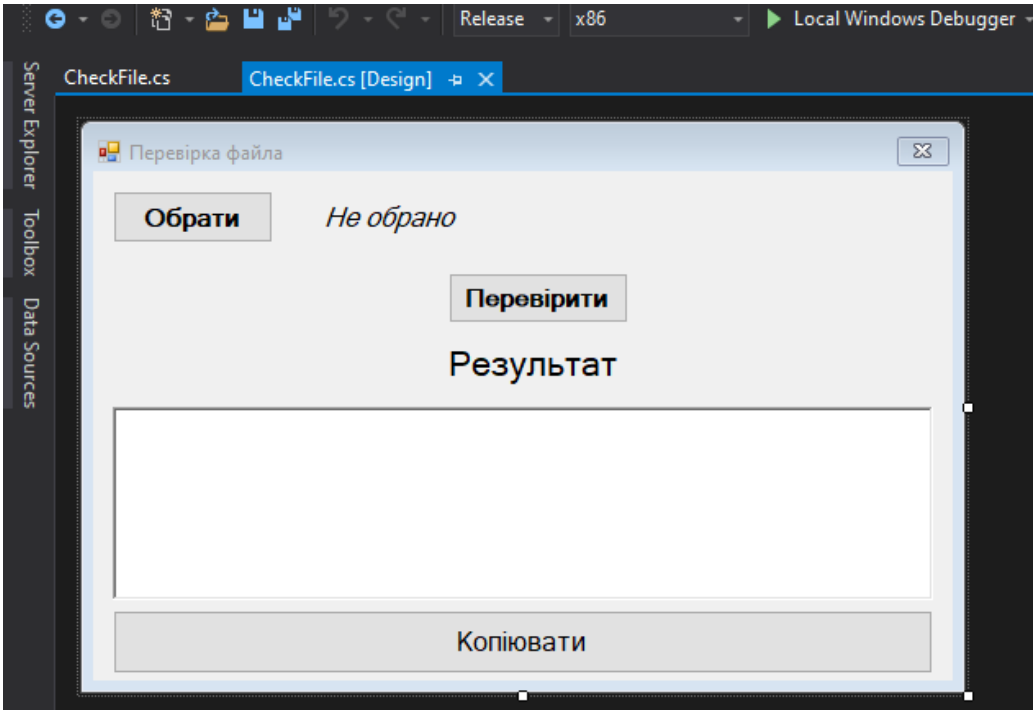


Рисунок 3.11 - Вигляд вікна вибіркового сканування

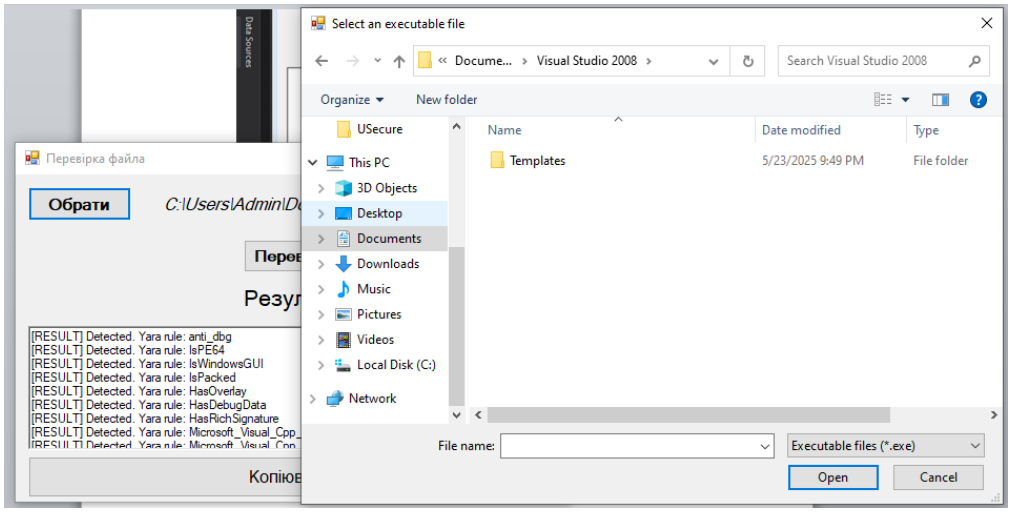


Рисунок 3.12 - Вигляд вікна вибіркового сканування з вибором виконуваного файлу через діалогове вікно та результатами сканування

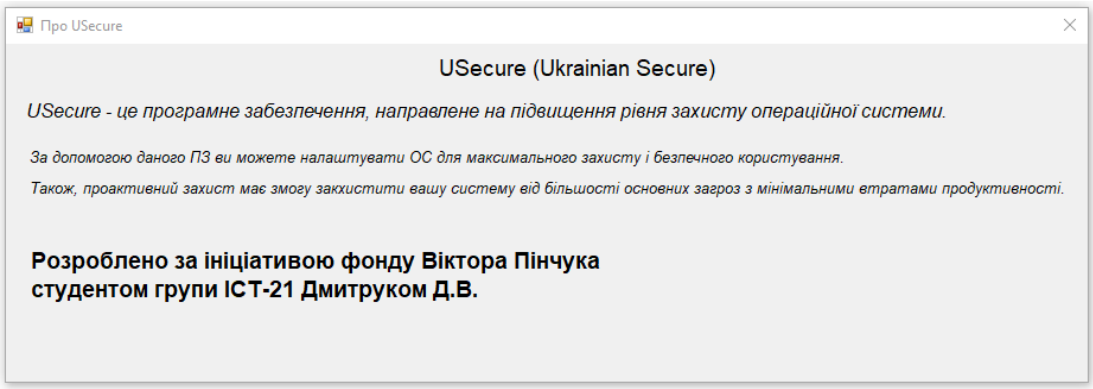


Рисунок 3.13 - Вигляд вікна «Про ПЗ»

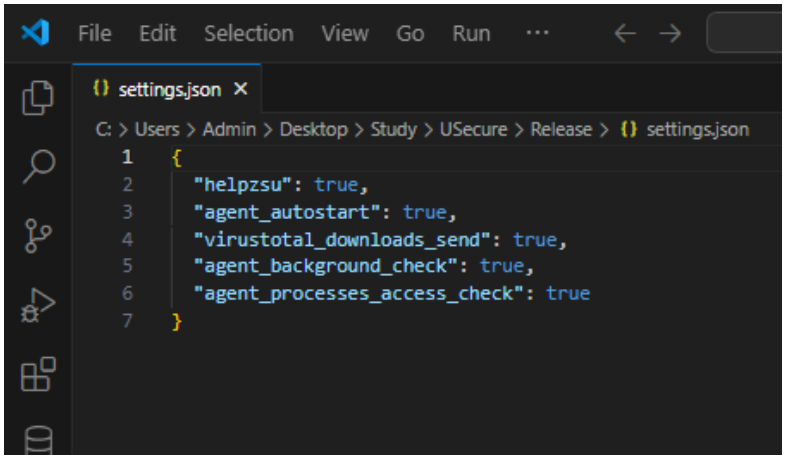
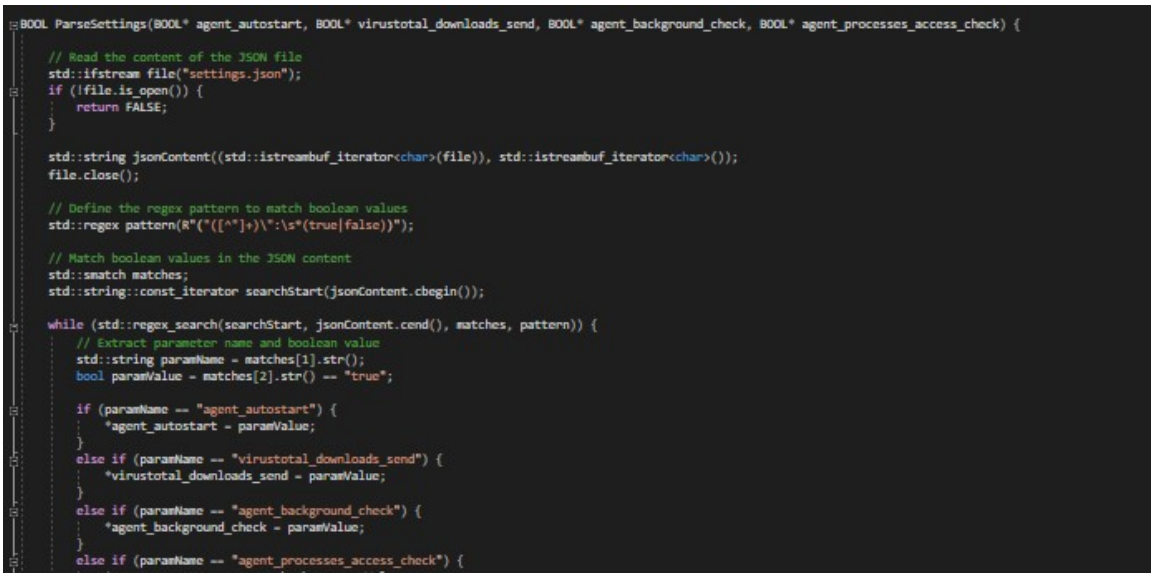
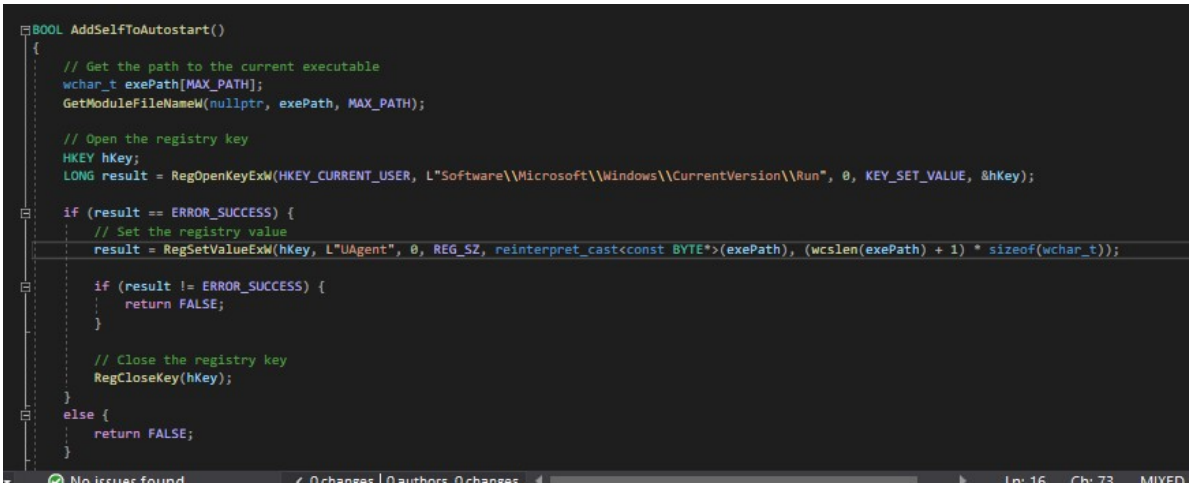


Рисунок 3.14 - Зміст файлу налаштувань



```
BOOL ParseSettings(BOOL* agent_autostart, BOOL* virustotal_downloads_send, BOOL* agent_background_check, BOOL* agent_processes_access_check) {

    // Read the content of the JSON file
    std::ifstream file("settings.json");
    if (!file.is_open()) {
        return FALSE;
    }

    std::string jsonContent((std::istreambuf_iterator<char>(file)), std::istreambuf_iterator<char>());
    file.close();

    // Define the regex pattern to match boolean values
    std::regex pattern(R"("([^"]+)\":\s*(true|false))");

    // Match boolean values in the JSON content
    std::smatch matches;
    std::string::const_iterator searchStart(jsonContent.cbegin());

    while (std::regex_search(searchStart, jsonContent.cend(), matches, pattern)) {
        // Extract parameter name and boolean value
        std::string paramName = matches[1].str();
        bool paramValue = matches[2].str() == "true";

        if (paramName == "agent_autostart") {
            *agent_autostart = paramValue;
        }
        else if (paramName == "virustotal_downloads_send") {
            *virustotal_downloads_send = paramValue;
        }
        else if (paramName == "agent_background_check") {
            *agent_background_check = paramValue;
        }
        else if (paramName == "agent_processes_access_check") {
```

Рисунок 3.15 - Код функції ParseSettings, яка відповідає за зчитування параметрів



```
BOOL AddSelfToAutostart()
{
    // Get the path to the current executable
    wchar_t exePath[MAX_PATH];
    GetModuleFileNameW(nullptr, exePath, MAX_PATH);

    // Open the registry key
    HKEY hKey;
    LONG result = RegOpenKeyExW(HKEY_CURRENT_USER, L"Software\\Microsoft\\Windows\\CurrentVersion\\Run", 0, KEY_SET_VALUE, &hKey);

    if (result == ERROR_SUCCESS) {
        // Set the registry value
        result = RegSetValueExW(hKey, L"UAgent", 0, REG_SZ, reinterpret_cast<const BYTE*>(exePath), (wcslen(exePath) + 1) * sizeof(wchar_t));

        if (result != ERROR_SUCCESS) {
            return FALSE;
        }

        // Close the registry key
        RegCloseKey(hKey);
    }
    else {
        return FALSE;
    }
```

Рисунок 3.16

```
DWORD WINAPI SuspiciousProcessesCheck(LPVOID lpParam) {
    allowedPIDs = (DWORD*)malloc(sizeof(DWORD)*4096);
    memset(allowedPIDs, 0, sizeof(DWORD) * 4096);

    while (true)
    {
        try {
            //Check process
            LoopForProcesses();
            Sleep(500); // 0.5 sec of delay
        }
        catch (...) {

        }
    }

    free(allowedPIDs);

    return TRUE;
}
```

Рисунок 3.17 - Код функції SuspiciousProcessesCheck

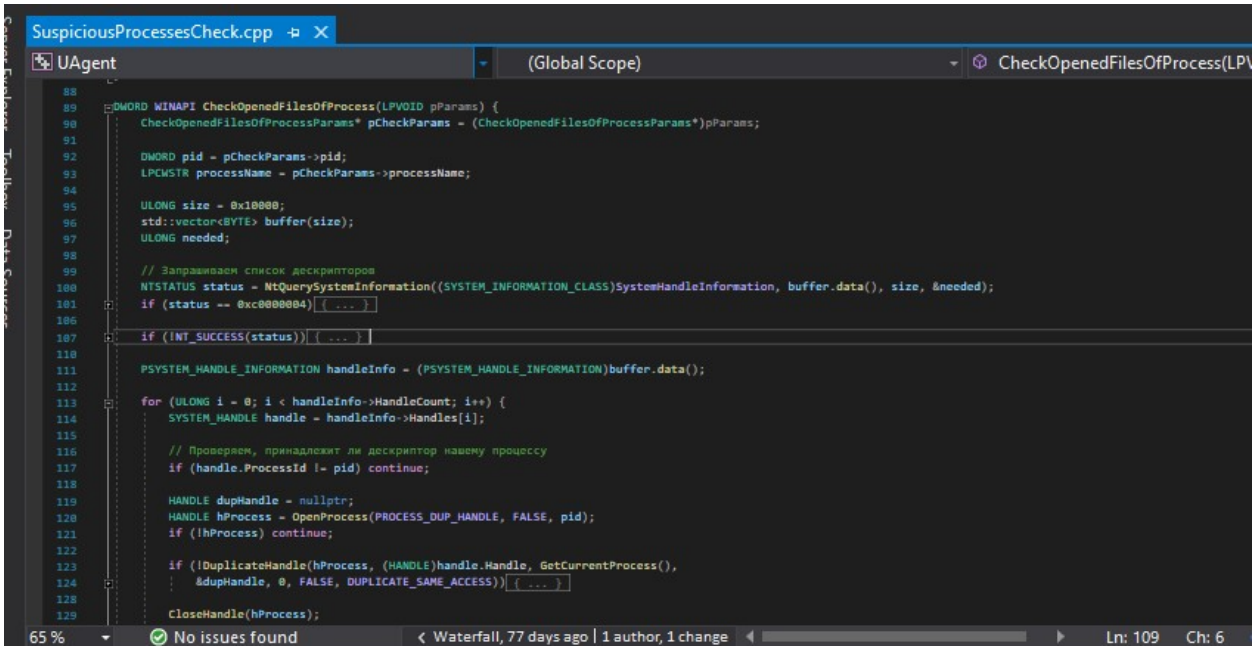


Рисунок 3.18 - Фрагмент коду функції потоку CheckOpenedFilesOfProcess

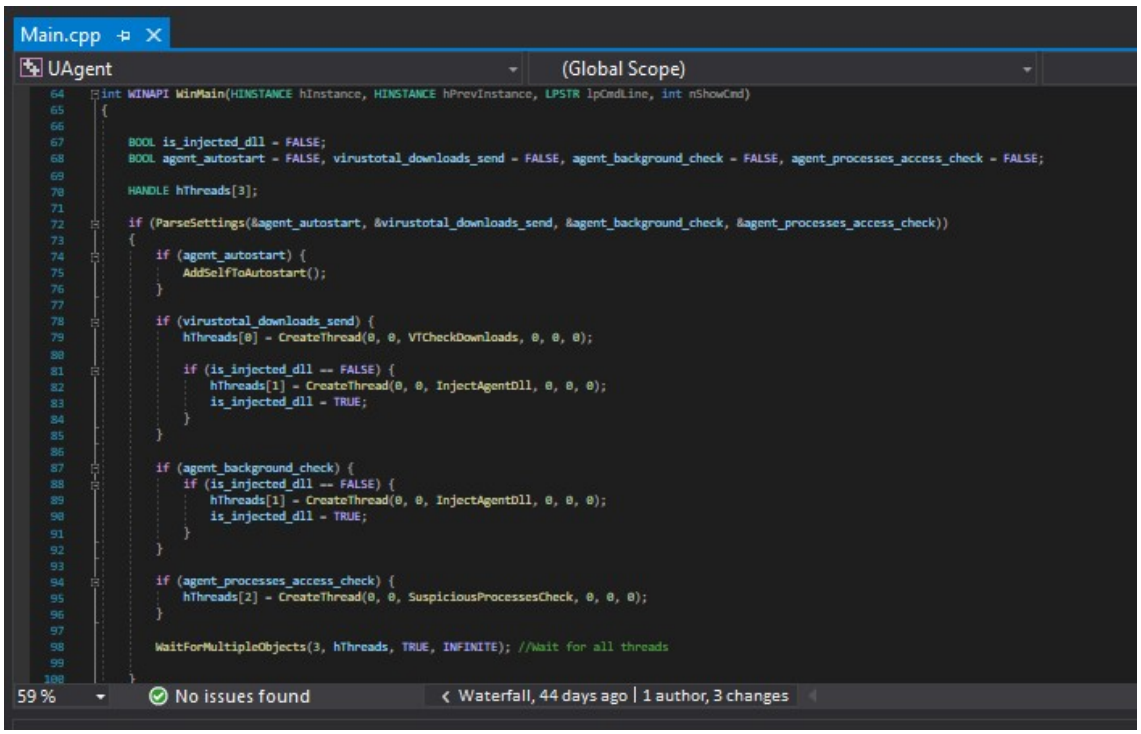


Рисунок 3.19 - Точка входу WinMain ПЗ UAgent інформаційної системи USecure

```
VOID MainRoutine()
{

    HANDLE hThreads[2];

    LPSTR SettingsFilePath = (LPSTR)malloc(MAX_PATH);
    GetSettingsFile(&SettingsFilePath);

    BOOL agent_autostart = FALSE, virustotal_downloads_send = FALSE, agent_background_check = FALSE, agent_processes_access_check = FALSE;

    if (ParseSettings(SettingsFilePath, &agent_autostart, &virustotal_downloads_send, &agent_background_check, &agent_processes_access_check))
    {
        if(agent_background_check)
        {
            hThreads[0] = CreateThread(0, 0, BackgroundRoutine, 0, 0, 0);
        }

        if (virustotal_downloads_send)
        {
            hThreads[1] = CreateThread(0, 0, HookingRoutine, 0, 0, 0);
        }
    }

    WaitForMultipleObjects(2, hThreads, TRUE, INFINITE); //Wait for all threads

    free(SettingsFilePath);
}
```

Рисунок 3.20 - Код функції MainRoutine, яка викликається після під’єднання dll

```
BOOL DetectBotByProcEntry(PROCESSENTRY32 pEntry) {

    if (!IsWindowVisible(FindWndByPID(pEntry.th32ProcessID))) {

        LPCWSTR SuspiciousProcesses[]{
            L"runtimebroker.exe",
            L"dllhost.exe",
            L"wscript.exe",
            L"svchost.exe",
            L"services.exe",
            L"smss.exe",
            L"lsass.exe",
            L"csrss.exe",
            L"lms.exe",
            L"wininet.exe",
            L"system.exe",
            L"update.exe",
            L"explorer.exe",
            L"regasm.exe"
        };

        for (int i = 0; i < 14; i++)
        {
            if (lstrcmpW(pEntry.szExeFile, SuspiciousProcesses[i]) == ERROR_SUCCESS) {
                return TRUE;
            }
        }

    }
```

Рисунок 3.21

ДОДАТОК Г

PE Yara rules checker

Rules repository: https://github.com/Yara-Rules/rules/tree/master

```
import yara

import os

import sys

import glob


def load_yara_rules_from_dirs(directories):

    rule_files = []

    for dir_path in directories:

                        rule_files.extend(glob.glob(os.path.join(dir_path,    '*.yar'))    +
glob.glob(os.path.join(dir_path, '*.yara')))


    rules_dict = {}

    for idx, rule_file in enumerate(rule_files):

        try:

            rules_dict[f"rule_{idx}"] = rule_file

        except Exception as e:

            print(f"Failed to include {rule_file}: {e}")

    return yara.compile(filepaths=rules_dict)
```

```
def main():
    if len(sys.argv) != 2:
        print("Usage: python scan.py path_to_exe")
        sys.exit(1)

    exe_path = sys.argv[1]
    if not os.path.isfile(exe_path):
        print(f"File not found: {exe_path}")
        sys.exit(1)

    rule_dirs = [
        "rules/antidebug_antivm",
        "rules/cve_rules",
        "rules/malware",
        "rules/packers"
    ]

    try:
        try:
```

```
            rules = load_yara_rules_from_dirs(rule_dirs)
        except Exception as e:
            print(f"[ERROR LOAD] Can not load yara rules {e}")
            return

        matches = rules.match(exe_path)
        if matches:
            for match in matches:
                print(f"[RESULT] Detected. Yara rule: {match.rule}")
        else:
            print("No detection.")
    except yara.Error as e:
        print(f"YARA error: {e}")

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