Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-24М
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Колесник Д.С.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Дрєєва Г.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Колесник Д.С. Дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Метою розробки є дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Об'єктом дослідження є процес протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Предметом дослідження є методи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Методи дослідження базуються на методах штучного інтелекту, методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, антивірусна система, штучний інтелект

ABSTRACT

Kolesnyk D.S. Research and software implementation of a system for counteracting detection by an antivirus system using artificial intelligence. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the second (master's) level of higher education, software has been developed, which is intended for a system for counteracting detection by an antivirus system using artificial intelligence.

The purpose of the development is the research and software implementation of a system for counteracting detection by an antivirus system using artificial intelligence.

The object of the research is the process of counteracting detection by an antivirus system using artificial intelligence.

The subject of the research is methods for counteracting detection by an antivirus system using artificial intelligence.

The research methods are based on artificial intelligence methods, information protection methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of a system for counteracting detection by an antivirus system using artificial intelligence.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer science, antivirus system, artificial intelligence

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КМ	–	комп’ютерна мережа
КСАЗ	–	комплексна система антивірусного захисту
МЕ	–	міжмережний екран
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ACL	–	Access Control List
FTP	–	File Transfer Protocol
http	–	HyperText Transfer Protocol
POP3	–	Post Office Protocol Version 3
SMTP	–	Simple Mail Transfer Protocol
VLAN	–	Virtual Local Area Network

ВСТУП

Актуальність теми. У даній роботі будуть запропоновані ефективні методи обходу статичного, динамічного й евристичного аналізу, використовуваного в новітніх антивірусних продуктах. Деякі техніки вже відомі навзагал, але є й додаткові трюки, які є ключовими при генерації недетектуємого вірусу. Обсяг файлу не менш важливий при використанні даних методів, і я постарався оптимізувати розмір настільки, наскільки можливо. У цьому документі також буде порушена тема внутрішнього пристрою антивірусів і операційної системи Windows.

Методи реалізації технік антидетекта залежать від типу вірусу. Всі методи, описувані в даній роботі, будуть працювати з усіма типами вірусів, однак основна увага приділена складеним корисним навантаженням для meterpreter, оскільки цей командний інтерпретатор уміє практично все, що й інші вірусні програми. Одержання сесії за допомогою meterpreter на віддаленій машині відкриває масу можливостей: розширення привілеїв, крадіжку облікових записів, міграцію між процесами, маніпуляцію реєстром і інші трюки пост-експлуатації.

Крім того, навколо meterpreter зібралося потужне й активне співтовариство, і цей інструмент популярний серед фахівців з безпеки.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– Дослідження системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– Програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Об'єктом дослідження є процес протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Предметом дослідження є методи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Методи дослідження базуються на методах штучного інтелекту, методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– Розроблено вітчизняний продукт протидії детектуванню антивірусною системою за допомогою штучного інтелекту, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2025 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для обходу статичного, динамічного й евристичного аналізу, використовуваного в новітніх антивірусних продуктах. При цьому в даній роботі буде реалізована й антивірусна складова.

Для того, що б зрозуміти, як буде працювати система введемо базову термінологію.

Детектування на базі сигнатур

За традицією антивірусні додатки виявляють вірусні програми в основному за допомогою сигнатур. Коли підозрілий файл попадає в антивірусну компанію, починається аналіз фахівцями, або системами динамічного аналізу. Як тільки вірус пізнаний, отримана сигнатура додається в базу даних антивірусу.

Статичний програмний аналіз

Статичний програмний аналіз додатка виконується без запуску програми. За допомогою подібного методу виконується аналіз вихідного коду або деяких форм об'єктного коду.

Динамічний програмний аналіз

Динамічний програмний аналіз виконується в процесі запуску додатків у середовищі з реальним або віртуальним процесором. Щоб динамічний аналіз був ефективним, цільове додаток повинне запускатися з достатньою кількістю тестових вхідних даних для ініціації роботи різних ділянок алгоритму.

Пісочниця

Пісочниця являє собою безпечне середовище для поділу запущених програм, що звичайно використовується для запуску нетестованого й недостовірного програмного коду з неперевірених джерел: виробників, користувачів або веб-сайтів для того, щоб уникнути шкоди для операційної системи.

Евристичний аналіз

Евристичний аналіз використовується багатьма антивірусними програмами для детектування раніше невідомих комп'ютерних вірусів, а також нових варіантів вірусів, що вже існують в «дикій природі». Даний вид аналізу, по суті, являє собою експертну систему, що визначає ступінь сприйнятливості системи стосовно конкретної погрози/ризику за допомогою різних правил прийняття рішень або вагових методів. Багатокритеріальний аналіз – один зі способів «зважування», що відрізняється від статичного аналізу, що опирається на доступну інформацію/статистику.

Ентропія

Ентропія являє собою випадкову послідовність, що збирається операційною системою або додатком для використання в криптографічних або інших цілях, де потрібні випадкові дані. Випадкова послідовність часто формується за допомогою апаратної платформи, наприклад, на базі руху миші або спеціальних генераторів випадкових чисел. Слабка ентропія може вплинути на продуктивність і безпеку.

1.2 Область застосування

Областю застосування є антивірусні системи. Розглянемо розповсюджені техніки, які дозволяють залишатися непомітним для більшості антивірусних продуктів.

Коли мова заходить про зменшення ймовірності детектування, перше, що приходить на розум: шифрувальники, пакувальники й обфускація коду. Ці інструменти й техніки усе ще дозволяють залишатися непомітним для більшості антивірусних продуктів, але оскільки кібербезпека розвивається семимильними кроками, більшість методів і програм, що існують у дикій природі, застаріли й не можуть зробити повністю недетектуємий вірус. З метою розуміння алгоритмів роботи я дам короткий опис для кожної типу цих технік і утиліт;

Обфускація

Обфускація коду пов'язана з перемішуванням вихідного тексту бінарного файлу без псування функцій. Цей метод утрудняє статичний аналіз і змінює геш-сигнатури бінарника. Обфускацію можна реалізувати за допомогою додавання декількох рядків сміттєвого коду або змінити послідовність виконання інструкцій програмним шляхом. Дана техніка дозволяє обійти пристойне число антивірусів, але кінцевий успіх залежить від глибини обфускації.

Пакувальники

Пакувальник стискає вихідний файл і поєднує стислі дані з кодом розпакування в один виконується файл, що. Коли стислий файл, що виконується, запускається, спочатку програма розпакування відтворить первісний код зі стислого файлу, після чого відбувається виконання. Коли антивіруси сканують упакований вірус, потрібно визначити алгоритм стиску й розпакувати файл, що виконується. Оскільки впаковані файли складніше проаналізувати, зловмисники зацікавлені в незасвіченому пакувальнику.

Криптори

Криптори призначені для шифровки бінарного файлу з метою утруднення аналізу або реверса-інжинірингу. У шифрувальника є дві частини: збирач і стаб. Збирач шифрує зазначений бінарний файл і поміщає усередину стаб, що є найважливішою частиною криптора. При запуску бінарного файлу спочатку стаб розшифровує первісну версію на згадку, а потім запускає розшифрований файл за допомогою методу «RunPE» (у більшості випадків).

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Бурхливий розвиток Інтернету не обійшов стороною всякого роду шахраїв і хакерів. Вони використовують шкідливе ПЗ для одержання доступу до особистої інформації користувача на його ПК або встановлення контролю за обчислювальною машиною.

Щоб ви могли орієнтуватися в сфері комп'ютерної безпеки, ми підготували рейтинг антивірусів 2025 року для Windows 11. У добірку включені як безкоштовні, так і комерційні програми.

ESET NOD32

Коли мова заходить про можливості антивірусів, про ESET згадують далеко не завжди, хоч захисні програми компанії встановлені на 100 млн. пристроях по усьому світі. Недооцінювати арсенал NOD32 Antivirus не варто, тому що в розпорядженні антивірусу є великий інструментарій для запобігання атак зловмисників на ваш комп'ютер. Вартість річної передплати на 2 комп'ютери клієнта становить 1590 гривень.

Варто відзначити швидку швидкість сканування локальних носіїв комп'ютера в спеціальному режимі. При цьому антивірус дбайливо витрачає ресурси ПК, не заважаючи іншим запущеним процесам. Геймерам сподобається спеціальний ігровий режим, що автоматично відключає всі спливаючі повідомлення. Єдиний мінус NOD32 Antivirus – слабкий захист від фішингу, програма все-таки пропускає велику кількість шкідливих сайтів.

Вивід: кращий антивірус для Windows 10, посідає перші місця в рейтингу порталу PCMag. Підходить для windows 7.

DR.Web

Останні версії антивірусів DR.Web уже котрий рік дають однозначна відповідь на питання який антивірус краще для Windows 10. Річна передплата на клієнт Dr. Web Security Space 11 коштує всього близько 950 гривень. За ці гроші користувач одержує файловий антивірус, Веб-антивірус, систему батьківського контролю й превентивний захист. Остання не допускає поширення шкідливих програм ще до їхнього запуску.

Остання версія Security Space 11 як і раніше не позбулася від стандартної «хвороби», від якої страждають багато антивірусних рішень – надмірного споживання оперативної пам'яті. Запущене сканування локальних носіїв помітно знижує швидкодія ПК.

Вивід: дешевий, але вимогливий до «заліза» антивірус. Розпізнає переважна більшість вірусів ще на ранній стадії.

Norton Security

Репутація компанії Norton говорить сама за себе, продукти розроблювача кілька років підряд займають провідні рядки в рейтингах авторитетних порталів. Річна передплата на останню версію антивірусу Norton Security Deluxe обійдеться покупцеві в 1205 гривень за 5 пристроїв.

Одним з головних козирів Norton Security перед конкурентами є наявність спеціальної системи інтелектуального навчання. Вона має багаторівневий принцип захисту, що не допускає до запуску шкідливе ПЗ. За версією журналу PCMag антивірус одержав 9.7 балів з 10 можливих, такі результати були виставлені після тесту програми на запобігання вірусних атак. Також варто відзначити високу швидкість Windows 10 із установленим Norton Security.

Вивід: практично бездоганний антивірус із відмінною продуктивністю. Входить у ТОП-10 продуктів для інформаційної безпеки за версією видання PCMag.

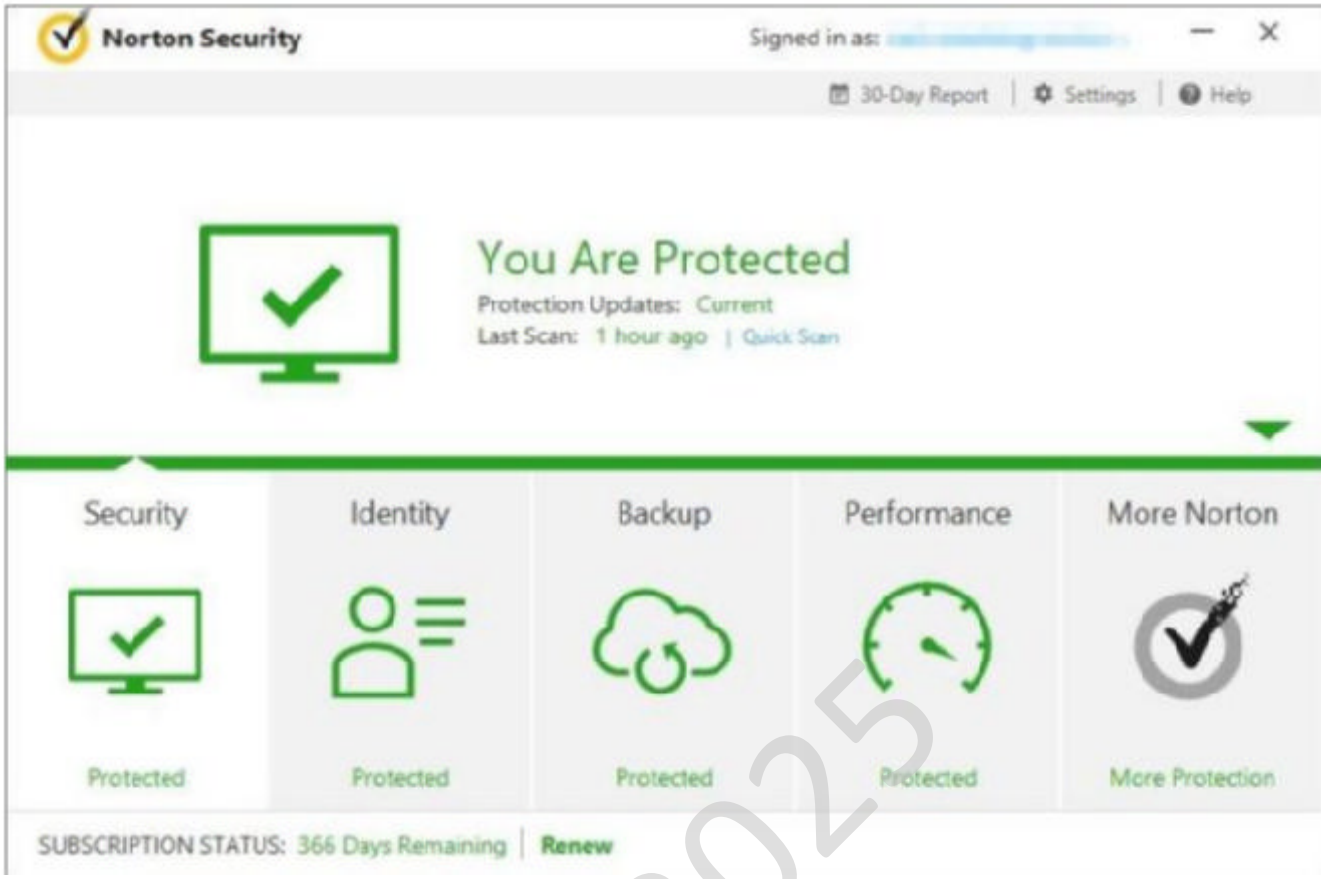

Рисунок 2.1 – Інтерфейс користувача Norton Security

Emsisoft Anti-Malware

Річна передплата на Emsisoft Anti-Malware 11.0 обійдеться користувачеві рівно в 999 гривень. Варто відзначити інтуїтивно зрозумілий дизайн антивірусу, всі зайві опції сховані в меню налаштувань. На домашній сторінці програми можна побачити чотири більших кнопки, які ведуть до інших вікон.

Клієнт демонструє відмінні результати в тесті PCMag, заблокувавши 98% шкідливих програм. Крім звичайного сканування, Anti-Malware також відслідковує підозрілу активність уже встановлених програм. При виявленні небажаних процесів клієнт відразу ставить користувача в популярність. А от з фішінгом справи погані, більшість атак підозрілих сайтів було зігноровано.

Вивід: недорогий антивірус із гарним захистом від шкідливого ПЗ. Єдиний мінус – непристосованість до фішингових сайтів.

AVAST Free

Існує думка, що безкоштовні антивіруси наділені урізаним функціоналом і нездатні забезпечити захист важливих даних. У випадку з AVAST Free всі саме навпаки, антивірус у деяких аспектах перевершує схожі рішення від конкурентів. Крім звичайного сканування локальних дисків AVAST Free також відслідковує інтернет трафік, зберігає паролі користувача, забезпечує зашифрований доступ до Мережі й перевіряє надійність Wi-Fi мереж.

Рисунок 2.2 – Інтерфейс користувача AVAST Free

Якщо ви все-таки зважитеся поставити платну версію іншого клієнта, AVAST Free автоматично перейде в сплячий режим. Антивірус не буде витрачати ресурси ПК і конфліктувати із іншою програмою. Проблемою ПЗ залишається слабкий захист від фішінгу, клієнт іноді не блокує потенційно небажані сайти.

Вивід: одна із кращих безкоштовних антивірусних програм, незважаючи на проблеми з фішінгом.

AVG AntiVirus Free

Програма управляється за допомогою єдиної централізованої утиліти AVG Zen. Її інтерфейс максимально простий і включає чотири основних меню: сам антивірус, оптимізація швидкодії ПК, захист VPN і функція Web Tuneup. Деякі з можливостей клієнта як і раніше відкриваються тільки після платної підписки.

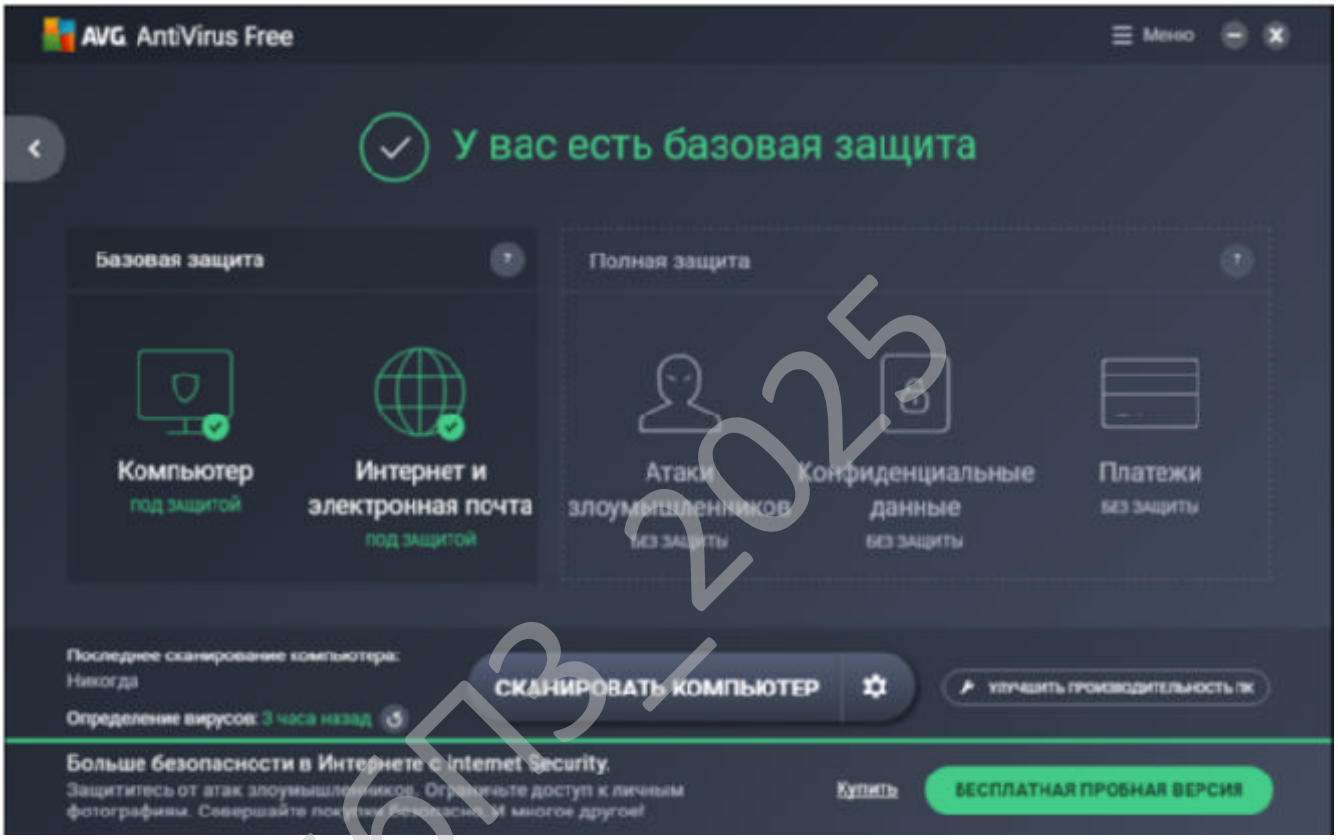

Рисунок 2.3 – Інтерфейс користувача AVG AntiVirus Free

Тести надійності дали AVG AntiVirus Free дуже високі бали. Програма блокує до 99% шкідливого ПЗ, що надходить на комп'ютер з різних джерел. Захист від фішінгу працює на середньому рівні, приблизно 15% небажаних сайтів все-таки відкриваються в браузері користувача. Окремо варто відзначити функцію Do Not Track. Вона блокує відстеження вашої діяльності (час відвідування сайтів, активність) сторонніми сервісами.

Вивід: безкоштовний і надійний вибір. Для розблокування всіх можливостей рекомендується оформити платну підписку.

BitDefender Antivirus Free Edition

Відсутність платної підписки не заважає цьому антивірусу надавати повний захист для користувача. У тесті на блокування програма набрало максимальну кількість балів із всіх можливих. Результати BitDefender Antivirus Free Edition виявилися навіть краще, ніж у деяких сервісів з досить дорогою платною підпискою.

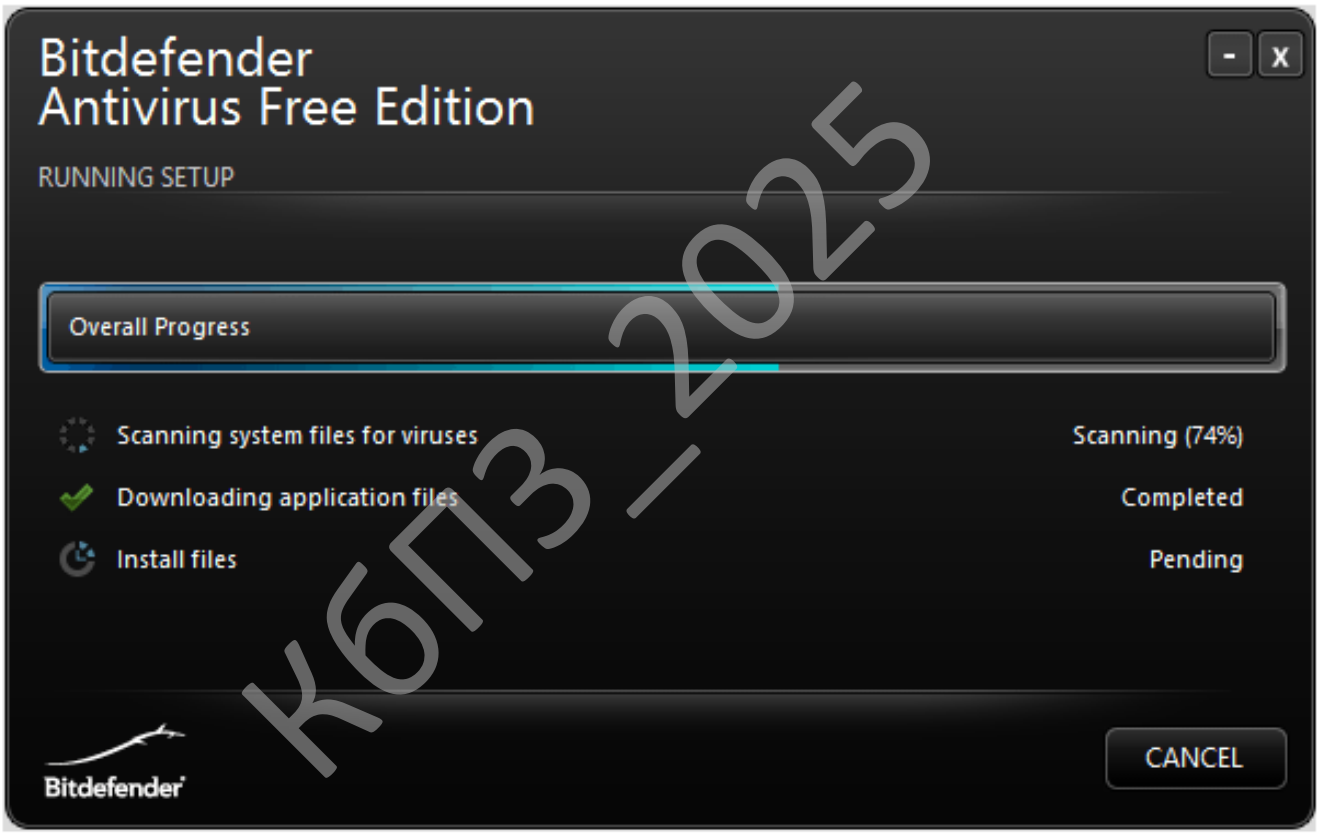

Рисунок 2.4 – Інтерфейс користувача BitDefender Antivirus Free Edition

Крім майже 100% блокування шкідливого ПЗ, антивірус забезпечує один з найвищих ступенів захищеності від фішінгу. Серед додаткових функцій – Bitdefender SafePay, спеціальний процес, що ізолює певну частину робочого

стола. Рекомендується використовувати його під час фінансових операцій через Інтернет.

Вивід: одна із кращих безкоштовних альтернатив для захисту від фішінгових атак. Єдиний мінус – практично повна відсутність всіх додаткових функцій без платної підписки.

Avira Free

Закриває рейтинг антивірусів розробка групи німецьких програмістів. Компанія Avira була заснована ще в 1986 році. З тих пор вона випускає на ринок комплексні рішення для інформаційної безпеки будь-якого рівня. Безкоштовна версія антивірусу відрізняється середнім рівнем захисту й дуже настирливою рекламою. У процесі роботи раз у раз з'являються нові повідомлення із закликом установити інші продукти компанії.

Ефективність Avira Free була доведена в лабораторних тестах, антивіруси для Windows 11 від багатьох інших конкурентів залишаються за набраними балами програми. Правда, за безпеку прийде заплатити більшою кількістю оперативної пам'яті, тому що клієнт дуже не ощадливо витрачає ресурси ПК. Розроблювачі обіцяють поліпшити оптимізацію антивірусу в 2026 році.

Вивід: якщо вам набридне настирлива реклама, замовте платну підписку. Антивірус від Avira залишається гарною альтернативою продуктам від більше відомих компаній.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки вебзастосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA,

Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність.

Python – скриптова мова програмування з досить простим синтаксисом. Для розуміння достатньо порівняти принципи написання найпростішої програми, яка виводить на екран текстове повідомлення. Саме тому мова програмування Python більш доступна для новачків, а професіонали встигли адаптувати її для вирішення великої кількості завдань. Це мультиплатформне рішення, тому знання Python дає можливість працювати у різних сферах: від розробки мобільних застосунків до ігрової індустрії та штучного інтелекту.

У мови програмування динамічна типізація: є можливість передавати до функцій будь-який тип даних без попереднього вказання. Інтерпретованість дозволяє знаходити помилки у коді ще до повної збірки у робочий застосунок. При цьому Python дуже чітко дає зрозуміти, де та через що виникла помилка.

Це мова об'єктноорієнтованого програмування (ООП). Програмне забезпечення на Python оформлене у вигляді моделей, які можуть бути зібраними у пакети. Тип та структуру кожного об’єкта можна запитати під час виконання програми. Для кожного з об’єктів можна отримати всю інформацію щодо його внутрішньої структури. Окрім того:

- у мови логічний синтаксис, завдяки чому вихідний код легко читати та розуміти;
- гнучкість та масштабованість Python дозволяє адаптувати високорівневу логіку та розширяти складні застосунки, як тільки виникне така необхідність;
- розробка на Python у більшості випадків проходить швидше, ніж на інших мовах програмування;
- Python – інтерпретована мова програмування. Це значить, що код можна написати у будь-якому текстовому файлі на будь-якій платформі, і потім успішно запустити;
- у Python – колосальна спільнота однодумців. Тож будь-які складнощі конкретних розробників вирішуються колективно.

Проте є декілька особливостей, які можна віднести до недоліків. Це повільність (ця мова програмування хоч і універсальна, проте повільніша за інші), велика кількість ресурсів, необхідних для роботи та «прив'язаність» до системних бібліотек.

Мова програмування Python використовується у наступних сферах:

1. Розробка програмних застосунків будь-якого напрямку.

2. Розробка серверної частини мобільних застосунків (найпопулярніший напрямок).

3. Ігри. Багато сучасних ігор для комп'ютерів (наприклад, World of Tanks) частково чи повністю написані на Python.

4. Вбудовані системи для різних пристроїв. Дуже часто Python використовують для написання внутрішніх платформ управління банкоматами.

5. Скрипти та плагіни до уже реалізованих програм для автоматизації процесів чи створення інших рішень.

6. Тестування (автоматизація цього процесу).

7. Машинне навчання. – основна мова для написання алгоритмів і аналітичних застосунків у сфері Machine Learning.

Бібліотеки Python

Різні бібліотеки Python використовують для виконання конкретних завдань. Наприклад, Matplotlib підходить для відображення даних у двовимірній та тривимірній графіці. Pandas підходить для зручної роботи з даними. NumPy дозволяє створювати масиви та керувати ними. Requests використовується для веброзробки. OpenCV-Python відкриває можливості для обробки зображень з метою оптимізації систем «машинного зору».

Найвідоміші фреймворки для мови програмування Python

Фреймворки Python допомагають створити зручне та функціональне середовище для розробки. У них міститься набір інструментів, модулів та бібліотек, корисних для виконання конкретних завдань. Це значно полегшує роботу: наприклад, дає змогу не витрачати час на розписування дій, які

повторюються, а використати релевантний інструмент. Тож є можливість позбутися рутинних процесів та сконцентруватися на логіці проєкту.

Серед найпопулярніших фреймворків для Python:

– Django – найстаріший та найвідоміший. Створений для реалізації великих інтерактивних проєктів;

– Pyramid – зручний у налаштуваннях, і дає можливість реалізувати складні нестандартні ідеї;

– Web2py – підходить в першу чергу для вебзастосунків і може використовуватись на будь-яких архітектурах.

Популярні Python IDE

IDE або інтегровані середовища розробки – це програмне забезпечення, яке надає розробникам необхідні інструменти для написання, редагування, тестування та налаштування коду. Для розробки на Python найчастіше використовують IDE PyCharm, IDLE, Spyder та Atom.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати

поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Перед початком вивчення ефективним методів, варто згадати пару слів про те, які є проблеми із широко відомими техніками й утилітами. На сьогоднішній день антивірусні компанії повністю оцінили всі ризики й найчастіше крім пошуку сигнатур і вірусного впливу шукають ознаки крипторів і пакувальників. У порівнянні з вірусами детектування крипторів і пакувальників простіше, оскільки останні мають однакові підозрілі алгоритми, на зразок розшифровки PE-файлу й запуску в пам'яті.

PE ін'єкція

Для того щоб зрозуміти схему запуску PE-образа в пам'яті, необхідно згадати про тім, як ОС Windows завантажує PE-файли. Звичайно при компіляції PE-файлу на адресу головного модуля встановлюється значення 0x00400000. Потім обробляються покажчики повних адрес і розраховуються інструкцій довгих переходів на базі адреси головного модуля. По закінченні компіляції в PE-файлі створюється секція таблиці релокації, що містить адреси інструкцій, що залежать від базової адреси образа, наприклад, покажчики повних адрес і інструкцій довгого переходу.

Під час виконання PE-образа операційна система перевіряє доступність кращого адресного простору. Якщо даний простір не доступно, перед стартом системний завантажник процесів повинен підбудувати абсолютні адреси в пам'яті. За допомогою секції релокації завантажник виправляє всі інструкції, що залежать від адреси, і запускає припинений процес. Весь цей механізм називається «Рандомізація розміщення адресного простору» (Address Space Layout Randomization; ASLR).

Для того щоб виконати PE-образ у пам'яті, криптору потрібно распарсити PE-заголовки й перемістити абсолютні адреси. Найчастіше там є підроблений системний завантажник, що відразу викликає підозру. Коли ми аналізуємо криптори, написані на З або мовах високого рівня, практично в кожному випадку спостерігаються API функції «NtUnmapViewOfSection» і «ZwUnmapViewOfSection», які попросту відключають подання секції від віртуального адресного простору процесу. Ці функції грають дуже важливу роль при використанні методу RunPE, що зустрічається практично в 90% крипторів.

Природно, антивірусні продукти не можуть позначати кожну програму як вірусну, котра використовує ці API-функції. Однак сам факт наявності даних функцій значить багато чого. Існує невеликий відсоток крипторів (в основному написаних на асемблері), які не використовують ці функції й виконують релокацію вручну. Якийсь час ці криптори працюють ефективно, але рано або пізно наступлять наслідку через спробу підробити системний завантажник. Інший серйозний мінус – при шифруванні PE-файлу серйозно збільшується ентропія. Якщо антивірусний сканер детектує нетиповий рівень ентропії, цілком можливо PE-файл буде позначений як підозрілий.

Ідеальний метод

Концепція шифрування вірусного коду має право на життя, але функцію дешифрування варто обробити гарною обфускацією. Коли справа доходить до виконання дешифрованого коду в пам'яті, можна обійтися без переміщення абсолютних адрес. Крім того, усередині вірусу потрібно передбачити алгоритм перевірки знаходження усередині пісочниці. Якщо перевірка виявила, що вірусний код аналізується антивірусом, функцію дешифрування запускати не слід. Замість шифрування всього PE-файлу, більш грамотно шифрувати шелл-код або тільки секція .text. У цьому випадку ентропія й розмір файлу залишаються на прийнятному рівні, і не міняються заголовки образа й секції.

Функція буде детектувати, чи аналізується вірус динамічно усередині чи пісочниці ні. Якщо функція виявляє ознаки антивірусного сканера, то або знову

викликається функція main, або вірус аварійно завершується. Якщо ж ознаки сканування відсутні, буде викликана функція для розшифровки шелл-коду.

Для збереження ентропії й розміру на прийнятному рівні я будуть обробляти шелл-код шифрувальником на базі алгоритму XOR з використанням мультибайтового ключа. Алгоритм XOR не є стандартом в індустрії, як наприклад RC4 або blowfish, але в нашім випадку сильне шифрування не потрібно. Антивірусні продукти не будуть займатися дешифруванням. Нам досить захисту від статичного аналізу рядків. Крім того, XOR-Шифрування набагато швидше й не використовує бібліотек, що впливає на підсумковий розмір вірусу.

Оскільки ми пишемо новий шматок вірусу, геш-сигнатура не буде відома антивірусним продуктам, і нам не потрібно турбуватися про виявлення за допомогою детектування, побудованого на базі сигнатур. Ми шифруємо шелл-код і обфускуємо функції анти-детектування/налагодження й алгоритм дешифрування, що цілком достатньо для обходу статичного/евристичного аналізу. Нам залишилося знайти спосіб обходу динамічного аналізу, що є найбільш важливою частиною функції «AV detect». Перед написанням даної частини необхідно розібратися, як працюють евристичні алгоритми антивірусних продуктів.

Евристичні движки

Евристичні движки засновані на статичному аналізі й механізмі правил. Головна мета таких движків – детектування вірусів нового покоління, які ще не відомі, за допомогою угруповання й оцінці погроз/ризиків в окремих фрагментів коду відповідно до визначених критеріїв. Навіть при скануванні найпростішої програми, що виводить на печатку «Привіт, світ», евристичний движок визначає рівень погрози й ризику. Якщо оцінка перевищує граничний рівень, файл позначається як вірусний. Евристичні движки – найбільш просунутий компонент антивірусних продуктів і використовують значну кількість правил і критеріїв. Оскільки антивірусні компанії не випускають документації, що описує

евристичні движки, відомі лише деякі критерії оцінки погроз/ризиків. Природно, тут може бути багато помилкових допущень і помилок.

От деякі відомі правила, використовувані при оцінці погроз:

Присутність циклу дешифрування.

– Читання поточного ім'я комп'ютера.

– Читання криптографічного машинного GUID.

– З'єднання з випадковими доменами.

– Читання дати установки Windows.

– Видалення файлів, що виконуються.

– Присутність потенційного IP-адреси в пам'яті бінарного файлу.

– Модифікація налаштувань проксі-сервера.

– Установка хуків/патчів до запущеного процесу.

– Інжектування в explorer.

– Інжектування у віддалений процес.

– Запит інформації про процес.

– Установка процесу в режим помилки для приховання повідомлення об помилки.

– Нетипова ентропія.

– Можлива перевірка на присутність антивірусного движка.

– Присутність функціонала для розширення привілеїв.

– Модифікація налаштувань політик додатків.

– Читання версії BIOS системи/відеокарти.

– Кінець PE-заголовка перебуває усередині нестандартної секції.

– Створення захищених областей пам'яті.

– Створення безлічі процесів.

– Спроба призупинити роботу на довгий час.

– Нестандартні секції.

– Читання Windows Product Id.

– Присутність функціонала для запуску/взаємодії із драйверами пристроїв.

– Присутність функціонала для блокування користувальницького уведення.

При написанні функції по детектуванню антивірусних продуктів і дешифрування шелл-коду, варто брати до уваги ці правила.

Дешифрування шелл-коду

Обфускація механізму дешифрування надзвичайно важлива, оскільки більшість евристичних движків у стані виявити цикли дешифрування усередині PE-файлів. Після значного збільшення випадків, коли програма вимагала викуп, деякі евристичні движки були майже повністю заточені під детектування процедур дешифрування. Після виявлення процедури дешифрування деякі сканери чекали, поки в регістрі ECX не виявиться значення 0, що в більшості випадків означає закінчення циклу. По досягненню закінчення циклу дешифрування виробляється повторний аналіз умісту файлу, що буде являти собою функцію дешифрування шелл-коду.

Показано цикл for, усередині якого виконуються логічні операції XOR між байтом шелл-коду й байтом ключа. Асемблерні блоки нагорі й унизу, по суті, не виконують ніяких операцій. Ці блоки «накривають» XOR-операцію випадковим набором байтів і переходів. Оскільки ми не користуємося просунуті механізми дешифрування, подібної обфускації буде досить для функції «Decrypt Shellcode».

3.2 Розробка структурної схеми

Детектування динамічного аналізу

У процесі написання механізму детектування пісочниці нам необхідно зробити обфускацію методів. Якщо евристичний движок виявить ознаки методів анти реверс-інжинірингу, різко зросте оцінка погрози з боку нашого вірусу.

Детектування відладника

Наш перший механізм детектування антивірусного продукту буде перевіряти присутність відладника усередині процесу. Для рішення даного завдання існує API-функція, що визначає, чи налагоджує процес відладником рівня користувача (user-mode debugger). Але ми не будемо використовувати дану функцію, оскільки більшість антивірусних продуктів моніторять вираження API-Викликів. Швидше за все, після детектування виклик буде розцінений як протидіючому реверсу-інжинірингу. Замість використання API-функції ми будемо шукати байт «BeingDebuged» в PEB-блоці.

Ділянка коду на рисунку вище витягає байт BeingDebuged з PEB-блоку. Якщо відладник є присутнім, перевірка повторюється заново, і так доти, поки в стеці не буде переповнення. Після переповнення спрацює виключення, і процес буде закритий. Цей шлях – найпростіший для виходу із програми. Ручна перевірка байта BeingDebuged дозволить обійти пристойну кількість антивірусних продуктів, але в деяких додатках передбачений подібний випадок, і ми повинні зробити обфускацію коду, щоб захиститися від статичного аналізу рядків.

Додавання переходів (інструкцій JZ) після кожної операції не вплине на логіку роботи функції, але додавання сміттєвих байтів між переходами дозволить обійти фільтри статичного аналізу рядків.

Завантаження підробленої бібліотеки

У цьому методі ми спробуємо завантажити неіснуючу бібліотеку під час виконання додатка. Звичайно коли ми намагаємося завантажити неіснуючу бібліотеку, HISTENCE повертає NULL, але деякі механізми динамічного аналізу, використовувані в антивірусних продуктах, допускають такі випадки й продовжують подальше дослідження потоку виконання додатка.

Використання функції GetTickCout

У цьому методі ми будемо опиратися на той факт, що час, що приділяється на сканування, обмежено. У більшості випадків антивірусні сканери

створені для кінцевого користувача. Відповідно, додаток повинне бути як можна більш дружелюбно й придатне для щоденного використання. Цей факт означає, що сканер не може витрачати занадто багато часу на сканування файлів, а повинен завершувати весь процес якнайшвидше. Спочатку розроблювачі вірусів використовували функцію «sleep» з метою дочекатися закінчення сканування, але зараз цей трюк практично не працює, оскільки кожний антивірусний продукт пропускає дану функцію. Ми будемо використовувати інший метод, показаний існує, що використовує API-функцію «GetThickCount». Дана функція витягає кількість мілісекунд, що пройшли з моменту запуску системи (до 49.7 днів). Ми будемо використовувати цю функцію для одержання часу, що пройшло з моменту запуску операційної системи, потім зупинимося на 1 секунду. Після функції sleep ми перевіримо, чи була ця ділянка пропущена, за допомогою порівняння двох значень, отриманих за допомогою GetTickCout.

Перевірка кількості ядер

Цей метод перевіряє кількість ядер процесора, що присутні у системі. Оскільки антивірусні продукти не можуть займати занадто багато ресурсів, ми можемо перевірити кількість ядер для того, щоб визначити, чи перебуваємо ми в пісочниці. Багато антивірусів не підтримують мульти-ядерну обробку й не можуть зарезервувати більше одного ядра для пісочниці.

Виділення великого обсягу пам'яті

Цей метод також експлуатує обмежений час, використовуваний при скануванні. Ми виділяємо близько 100 Мб пам'яті й заповнюємо даний простір порожніми байтами. Потім пам'ять звільняється.

Коли пам'ять, використовувана програмою, під час виконання починає рости, через якийсь час антивірусні сканери зупиняються, щоб не витрачати багато часу на аналіз файлу. Цей метод можна використовувати кілька разів. Дана техніка дуже стара й примітивна, але усе ще допомагає обійти пристойна кількість сканерів.

Маніпуляція прапором трасування

Як треба з назви, цей прапор використовується під час трасування програм. Якщо прапор трасування встановлений, кожна інструкція буде ініціювати виключення «SINGLE_STEP». Прапором трасування можна маніпулювати з метою протидії трасувальниками за допомогою коду, показаного існує:

Перевірка мьютексів

Цей метод через свою простоту виглядає дуже перспективним. Суть техніки полягає в перевірці присутності певного мьютекса в системі.

Якщо функція «CreateMutex» не поверне помилку ERROR_ALREADY_EXISTS, вірус запускається ще раз, оскільки більшість антивірусних продуктів не дозволяють програмам, які піддаються динамічному аналізу, запускати нові процеси або працювати з файлами поза пісочницею. Якщо за результатами перевірки мьютекса вертається помилка ERROR_ALREADY_EXISTS, можна запускати функцію дешифрування. Існують набагато більше креативні способи використання мьютексов з метою анти-детектування.

Правильні методи виконання шелл-коду

Починаючи з Windows Vista, компанія Microsoft впровадила технологію DEP (Data Execution Prevention; Запобігання виконання даних). Ця міра, спрямована на зміцнення безпеки, допомагає запобігти псуванню комп'ютера шляхом періодичного моніторингу. Моніторинг дозволяє підтримувати коректно використання пам'яті. Якщо механізм DEP виявляє випадок некоректного використання пам'яті вашого комп'ютера, програма закривається, і користувач сповіщається. Відповідно, ви не можете просто помістити трохи байт у символьний масив і виконати свою творчість. Вам потрібно виділити область пам'яті із прапорами на читання, запис і виконання за допомогою API-функцій.

Компанія Microsoft передбачила трохи API-функцій для резервування сторінок пам'яті. Більшість вірусів для резервування використовують функцію

«VirtualAlloc», що, як ви вже догадалися, полегшує завдання по детектуванню. Використання інших функцій для маніпуляції пам'яттю допоможе виконати той же самий трюк більш непомітно.

Далі я покаджу методи виконання шелл-коду за допомогою різних API-функцій.

HeapCreate/HeapAlloc:

ОС Windows також дозволяє створити купи з атрибутами на читання, запис і виконання.

LoadLibrary/GetProcAddress

Комбінація WINAPI-функцій LoadLibrary і GetProcAddress дозволяє використовувати інші API-функції. У цьому випадку не буде прямого виклику функцій, пов'язаних з маніпуляцією пам'яті, а сам вірус, імовірно, буде більше непомітний.

GetModuleHandle/GetProcAddress

Цей метод зовсім не використовує функцію LoadLibrary, а одержує оброблювач уже завантаженого модуля kernel32.dll за допомогою функції GetModuleHandle. Дана техніка для запуску шелл-коду, можливо, одна із самих непомітних.

Мультипоточність

PE-файли, що використовують кілька потоків, завжди складніше аналізувати, у тому числі й для антивірусних продуктів. За допомогою мультипоточного підходу ми можемо запускати шелл-код і одночасно продовжувати виконання функції «AV Detect».

У коді, показаному вище, виконання шелл-коду відбувається в окремому потоці. Паралельно виконанню шелл-коду в нескінченному циклі виконується функція для обходу антивірусу. Такий підхід дозволяє перевіряти присутність пісочниці й динамічного аналізу, що життєво важливо для обходу просунутих евристичних движків, які чекають запуску шелл-коду.

При компіляції вихідних текстів вірусів необхідно включити захист стека й видалити символи з метою утруднення реверс-інжинірингу й зменшення розміру кінцевого файлу, що виконується. Рекомендується компілювати в Embarcadero RAD Studio через присутність асемблерних вставок.

Якщо скористатися одночасно всіма методами, продемонстрованими в даній роботі, згенерований вірус зможе обійти 35 найбільш просунутих антивірусних продукту.

Впровадження бекдору в PE-файл

Далі, у даній роботі розповімо про декілька методів, використовуваних для розміщення вірусу в PE-файлі. Щоб зрозуміти суть матеріалу, що викладається, читачеві необхідно бути знайомим з асемблером для архітектури x86 і відладниками хоча б на середньому рівні й розуміти формат PE файлів. Цей документ був опублікований 8 грудня 2016 року на сайті pentest.blog, а також підготовлений в PDF форматі для офлайнового читання.

У цей час практично всі фахівці з безпеки, пентестери й аналітики вірусів мають справу з бекдорами на щоденній основі. Приміщення трояна в систему або в конкретну програму – найбільш популярний спосіб для підтримки постійного доступу до цільової машини. У більшості статей розповідається про методи для імплантування вірусів в 32 бітні PE файли, але оскільки PE-формат являє собою модифіковану версію формату Unix COFF (Common Object File Format; Загальний формат об'єктних файлів), логіка, закладена в основу цих технік, застосовна й до всіх інших типів файлів, що виконуються. Крім того, непомітність убудованого вірусу дуже важлива й прямо впливає на тривалість знаходження в системі. Методи, які будуть описані в даній роботі, спрямовані на зниження відсотка детектування настільки наскільки можливо. Перед подальшим читанням рекомендую ознайомитися з першою частиною, де розповідалося про обхід технік детектування, внутрішньому пристрої антивірусів і фундаментальних постулатів антидетектування.

Уведемо додаткову термінологію.

Навчальне проникнення

Існують групи, що складаються зі світлих хакерів (або білих капелюхів), які атакують цифрову інфраструктуру організації, як якби це робив реальний зловмисник, для того, щоб протестувати стійкість системи до зовнішніх погроз (по-іншому це називається пентест). Наприклад, компанія Microsoft регулярно проводить подібні кібер-навчання. Плюси від подібних заходів лежать на поверхні – у процесі можуть найтися проломи й проблеми, які можна заздалегідь усунути. Крім того, тут можуть виявитися шляхи влучення конфіденційної інформації назовні, схеми експлуатації й інші «недокументування» можливості системи.

Рандомізація розміщення адресного простору

ASLR являє собою техніку безпеки, спрямовану на захист від атак, пов'язаних з переповнення буфера. Щоб не дати зловмисникові коректно перейти на певну функцію усередині пам'яті, ASLR у випадковому порядку вишиковує позиції ключові областей інформації в адресному просторі процесу. Сюди ж включається базова адреса файлу, що виконується, і позиції стека, купи й бібліотек.

Code Cave (Печера в коді)

Code Cave являє собою шматок коду, що записується іншою програмою на згадку стороннього процесу. Цей код може бути виконаний за допомогою створення віддаленого потоку усередині цільового процесу. Code cave найчастіше є посиланням на секцію скриптових функцій коду, куди можна інжектувати будь-які інструкції. Наприклад, якщо в пам'яті скрипту 5 байт, і 3 байти використовуються, в 2 байти, що залишилися скрипту можна додати зовнішній код.

Контрольна сума

Контрольна сума являє собою невелику порцію інформації із блоку цифрових даних для виявлення помилок, які можуть з'явитися під час передачі або зберігання файлу. Звичайно за допомогою контрольної суми перевіряється

настановний файл після одержання із сервера. Між нами говорячи, контрольні суми звичайно використовуються для верифікації цілісності даних, але не враховують дійсність інформації.

Основні методи

Всі приклади з даної роботи будуть продемонстровані на базі файлу, що виконується, SSH-клієнта з ім'ям putty. Є кілька причин для вибору саме цього додатка як піддослідний зразок. Putty використовує безліч бібліотек і API-функцій. По-друге, впровадження вірусу в ssh-клієнт залучає менше уваги, оскільки програма вже виконує tcp-з'єднання, і, таким чином, буде простіше уникнути моніторингу з боку системи безпеки.

Код бекдору буде взятий із шелл-коду, використовуваного при зворотному TCP-з'єднанні й написаного під meterpreter. Головна мета – інжектувати шелл-код у цільовий PE-файл без псування функціональності додатка. Інжектований шелл-код буде запускатися в окремому потоці й буде постійно намагатися приєднатися до оброблювача. Друге завдання – під час виконання всіх цих маніпуляцій необхідно залишатися непомітним настільки, наскільки можливо.

Загальний підхід впровадження вірусу в PE-файл складається з 4 кроків:

– Знаходження доступного простору для коду бекдору.

– Перехоплення потоку виконання.

– Впровадження бекдору.

– Відновлення потоку виконання.

У кожному кроці є свої нюанси, які прямо впливають на стійкість, тривалість і непомітність убудованого вірусу.

Проблема з доступним простором

Знаходження доступного простору – перший крок до реалізації нашого завдання. Надзвичайно важливо вибрати правильне місце усередині PE-файлу для впровадження бекдору. Оцінка погрози з боку зараженого файлу сильно залежить від того, як ви вирішите це завдання. Тут існує два підходи.

Додавання нової секції

У порівнянні із другим підходом тут більше ймовірність виявлення. Хоча, з іншого боку, при додаванні нової секції в нас немає обмежень по просторі, і ми можемо впровадити бекдор будь-якого розміру.

За допомогою дизасемблера або редактори LordPE PE-файл можна розширити за допомогою додавання заголовка нової секції. Існує таблиця секцій файлу, що виконується, putty. За допомогою PE-редактора додана нова секція «NewSec» розміром 1000 байт.

Під час створення нової секції важливо встановити прапори на читання/запис/виконання, щоб запустити шелл-код, коли PE-образ проектується (map) на згадку.

Після додавання заголовка секції необхідно адаптувати розмір файлу, що робиться в шістнадцятковому редактору за допомогою додавання порожніх байтів розміром з нову секцію в кінець файлу.

Після додавання нової порожньої секції необхідно запустити виконується файл, що, і перевірити, є чи помилки. Якщо все пройшло гладко, нова секція готова до модифікації в відладнику.

Рішення проблеми доступного простору за допомогою додавання нової секції має деякі недоліки. Практично всі антивіруси пізнають нестандартні секції, і якщо, до того ж, там є повний набір прапорів на читання/запис/виконання, те ця ситуація виглядає ще більш підозріло.

Навіть якщо ми просто додамо нову секцію з повними правами без бекдору, деякі антивіруси вже позначають виконується файл, що, як вірусний.

Використання Code Cave

У другому методі, спрямованому на рішення проблеми доступного простору, використовуються code cave'и із цільового файлу, що виконується. Практично всі скомпільовані бінарні файли мають code cave'и, які можуть бути використані для впровадження вірусу. Code cave у порівнянні з новою секцією залучає набагато менше уваги, оскільки в цьому випадку застосовуються вже

існуючі звичайні секції. Додатковий і не менш важливий плюс полягає в тім, що після впровадження вірусу розмір PE-файлу не міняється. Однак ця техніка має свої недоліки.

Кількість і розмір code cave'ов варіюється від файлу до файлу, але в цілому загальний розмір менше, ніж при додаванні нової секції. При використанні code cave'ов код бекдору варто зменшувати настільки, наскільки можливо. Другий недолік – набір прапорів. Оскільки виконання буде перенаправлятися в code cave, у секції повинні бути права на виконання. У випадку з деякими шелл-кодами (які самі себе кодують або піддають обфускації) потрібно права на запис для того, щоб виконувати зміни усередині секції.

Використання декількох code cave'ов дозволяє перебороти обмеження із простором. Тут же додатковий плюс у тім, що вірус збирається з окремих шматків. Однак зміна привілеїв секції буде виглядати підозрілим. Існують просунуті методи для модифікації привілеїв в областей пам'яті під час виконання додатка з метою запобігання прямої зміни прапорів секції, але оскільки ці техніки вимагають спеціалізованого шелл-коду, кодування й парсингу таблиці IAT, ця тема буде порушена в наступних статтях.

За допомогою утиліти Cminer дуже легко підрахувати всі code cave'и бінарного файлу. Використовуємо команду ./Cminer putty.exe 300 для пошуку code cave'ов більше 300 байт.

У нашім випадку знайдено 5 гарних екземплярів для наступного використання. Стартова адреса задає адреса віртуальний пам'яті (virtual memory address, VMA) в code cave при завантаженні PE-файлу на згадку. Зсув файлу, вимірюваного в байтах, – адреса місцезнаходження потрібної області усередині PE-файлу.

За результатами пошуку з'ясувалося, що більшість областей перебувають усередині секції даних. Оскільки в цих секціях не встановлений прапор на виконання, будуть потрібні зміни. Розмір бекдору біля 400-500 байт, і області Cave 5 вистачить позаочі. Стартова адреса даної області потрібно зберегти, а

після зміни привілеїв секції перший етап впровадження вірусу можна вважати завершеним. Тепер потрібно перенаправляти виконання на нашу область.

Перехоплення потоку виконання

На цьому етапі потрібно перенаправляти потік виконання на код бекдору за допомогою модифікації потрібної інструкції у файлі, що виконується. Тут варто згадати важливу деталь щодо вибору інструкції для зміни. Всі бінарні інструкції мають розмір у байтах. Щоб перейти до адреси розташування бекдору буде потрібно довгий перехід (long jump) розміром, що використовує 5 або 6 байт. При зміні бінарного файлу інструкція, що буде патчиться, повинна бути того ж розміру, що й довгий перехід, інакше попередня й наступна інструкція будуть зіпсовані.

Дуже важливий вибір правильного місця для перенапрямку виконання, оскільки якщо перенапрямок буде здійснюватися прямо, неминуче детектування на етапі динамічного аналізу антивірусними продуктами.

Приховання усередині користувальницьких функцій

Перше, що спадає на думку, для обходу пісочниці й динамічного аналізу – відкладене виконання шелл-коду або використання детектора пісочниці, за результатами роботи якого виконується та або інша вітка алгоритму. З іншого боку, у більшості випадків через обмеження по розмірах ми не можемо додати зайві ділянки коду в PE-файл. Крім того, реалізація технік антидетектування на низькому рівні вимагає багато часу й сил.

Цей метод використовує функції, що вимагають дій користувача. Перенапрямок виконання усередині подібних функцій буде спрацьовувати тільки в тому випадку, якщо користувач працює в програмі. Якщо дана техніка буде реалізована коректно, успіх практично гарантований, і, до того ж, не буде збільшений розмір бекдору.

По натисканню кнопки «Open» із графічної оболонки запускається функція перевірки встановленого IP-адреси.

Якщо поле IP-адреси не порожнє, і значення коректне, запускається функція для з'єднання із зазначеним IP-адресою.

Якщо клієнт успішно створив ssh-сесію, з'явиться нове вікно для уведення ім'я користувача й пароля.

Саме в цьому місці відбудеться перенапрямок. Оскільки антивірусні продукти не настільки просунуті для аналізу такого роду механізмів, впроваджений бекдор, швидше за все, не буде виявлений за допомогою динамічного аналізу.

За допомогою нескладних методів реверс-інжинірингу, призначених для роботи з рядками й посиланнями на рядки, буде не складно знайти адресу функції з'єднання після того, як клієнт установив з'єднання із призначеним IP-адресою. Рядок «login as:», що з'являється в спливаючому вікні, допоможе нам знайти адресу функції з'єднання. У пошуках посилань на рядки нам допоможе IDA Pro.

Для знаходження рядка «login as:» в IDA Pro відкриваємо Views->Open Subviews->Strings on IDA

Після знаходження потрібного рядка двічі клікаємо і переходимо до місцезнаходження. Усередині секцій даних IDA знаходить всі перехресні посилання на рядки. Для виводу всіх перехресних посилань натискаємо комбінацію клавіш «Ctrl+X».

Існує посилання усередині функції, що виводить рядок «login as:»

Існує інструкція, що ми будемо змінювати. Після виконання коду бекдору, ця інструкція буде використовуватися знову.

Після зміни інструкції PUSH 467C7C на JMP 0x47A478 процес перенапрямку потоку виконання можна вважати завершеним. Важливо не забувати, що адреса наступної інструкції буде використовуватися як адреса повернення після виконання коду вірусу. Наступний крок – інжектування коду бекдору.

Інжектування коду бекдору

Перша думка, що спадає на думку при впровадженні бекдору, – збереження регістрів перед виконанням вірусного коду. Кожне значення усередині регістрів – надзвичайно важливо для виконання програми. Помістивши інструкції PUSHAD і PUSHFD на початку code cave, ми зможемо зберегти всі регістри й прапори регістрів усередині стека. Ці значення будуть повернуті після виконання коду вірусу, і програма продовжить виконання без яких-небудь проблем.

Як було згадано раніше, наш бекдор являє собою зворотний tcp-шелл-код під meterpreter, узятий із проекту metasploit. Однак буде потрібно деякі зміни усередині шелл-коду. Звичайно зворотний tcp-шелл-код намагається приєднатися до оброблювача деяка кількість разів, і якщо з'єднатися не вдалося, процес закривається за допомогою виклику API-функції ExitProcess.

Проблема полягає в тім, що якщо з'єднатися з оброблювачем не вийде, виконання клієнта putty буде зупинено. Після внесення невеликих змін після невдачі шелл-код буде знову намагатися з'єднатися з оброблювачем. До того ж, небагато зменшиться розмір шелл-коду.

Після внесення змін усередині асемблерного коду виконуємо компіляцію за допомогою команди nasm -f bin stager_reverse_tcp_nx.asm. Тепер зворотний tcp-шелл-код готовий до вживання, але поки ще не поміщений у правильне місце. Наша мета – здійснити виконання в окремому потоці, для створення якого буде потрібно окремий шелл-код, що виконує виклик API-функції CreateThread. Функція буде вказувати на первісний зворотний tcp-шелл-код.

Після приміщення байтів шелл-коду усередину файлу createthread.asm у шістнадцятковому форматі, як показано на рисунку вище, виконуємо компіляцію за допомогою команди nasm -f bin createthread.asm. Тепер шелл-код готовий для впровадження в code cave, але перед вставкою варто виконати кодування для того, щоб обійти статичний/сигнатурний аналіз антивірусу. Оскільки всі кодувальники із проекту metasploit відомі більшості антивірусів, рекомендується

використовувати нестандартне кодування. У даній роботі не буде розказано про створення нестандартних кодувальників, але можна обійтися сполученням декількох кодувальників із проекту metasploit. Після кожного акту кодування завантажуємо шелл-код у проект Virus Total в «сирому» форматі й перевіряємо результати перевірки. Комбінуємо кодувальники доти, поки шелл-код стане повністю непомітним.

Після правильного кодування шелл-код готовий до впровадження усередину code cave. Вибираємо інструкцію, яка треба за інструкцією PUSHFD, і натискаємо Ctrl+E в відладнику Immunity Debugger. Шелл-код буде вставлений у шістнадцятковому форматі.

Одержати закодований шелл-код у шістнадцятковому форматі можна двома способами: вивести на печатку за допомогою команди xxd -ps createthread або відкрити й скопіювати в шістнадцятковому редакторі. При копіюванні шістнадцяткових значень в відладник Immunity Debugger не забувайте про обмеження на копіюємі байти, які присутні при вставці коду. Необхідно запам'ятати останні два вставлених байти, і після натискання на кнопку OK потрібно зробити повторне копіювання наступних ділянок. Коли шелл-код повністю вставлений в code cave, операцію по впровадженню можна вважати завершеної.

Відновлення потоку виконання

Після створення потоку виконання бекдору необхідно відновити виконання програми. Тобто регістр EIP повинен указувати на функцію, що перенаправляла виконання на code cave. Однак перед переходом на цю функцію варто відновити раніше збережені регістри.

Помістивши інструкції POPFD і POPAD у кінець шелл-коду, ми відновимо усе раніше збережені регістри зі стека в тім же порядку. Після відновлення регістрів необхідно згадати ще про один нюанс. При перехопленні потоку виконання інструкція PUSH 467C7C була замінена на інструкцію JMP 0x47A478 для того, щоб перенаправляти виконання на code cave. При приміщенні інструкції PUSH 467C7C у кінець коду, перехоплена інструкція також відновиться. Тепер

прийшов час повернутися до функції, що перенаправляла виконання до code cave за допомогою вставки інструкції JMP 0x41CB73. Кінець результуючого коду повинен виглядати, як показано існує.

Потім виділяємо всі змінені й вставлені інструкції, натискаємо праву кнопку миші й копіюємо у виконується файл, що. Ця операція повинна бути пророблена з кожною модифікованою інструкцією. Коду всі інструкції скопійовані й збережені у файл, закриваємо відладник і тестуємо нашу творчість. Якщо виконання проходить без помилок, бекдор готовий до вживання.

У завершенні рекомендується змінити контрольну суму файлу, що вийшов, щоб зберегти автентичність і не викликати зайві підозри.

Якщо всі описані методи використані правильно, готовий бекдор буде повністю непомітним. На закінчення будуть наведені контрзаходи для захисту від описаних раніше технік. Це методи будуть корисні адміністраторам, аналітикам вірусів і розроблювачам антивірусів.

Контроль над привілеями секцій

Коли мова заходить про заражені файли, у першу чергу варто думати про детектуванні аномалій, пов'язаних із привілеями секцій. За замовчуванням компілятори ніколи не встановлюють повні права на секції, якщо тільки програміст не виставив спеціальні налаштування. Особливо не повинні мати привілеїв на виконання секції даних: .data i.rdata. Крім того, секції коду (наприклад, .text) не повинні мати прав на запис. Подібні аномалії, пов'язані зі зміною привілеїв, варто розглядати як підозріле поводження.

Присутність нестандартних секцій

Якщо програміст не вносить зміни в конфігурацію, компілятори звичайно створюють 5-6 стандартних типів секцій. В усі продукти, пов'язані з безпекою, повинен бути впроваджений механізм виявлення нестандартних і підозрілих секцій. Даний механізм повинен стежити за ентропією й вирівнюємо даних усередині секції. Якщо секція містить високу ентропію й нестандартно впорядковану інформацію, це ще одна підозріла ознака.

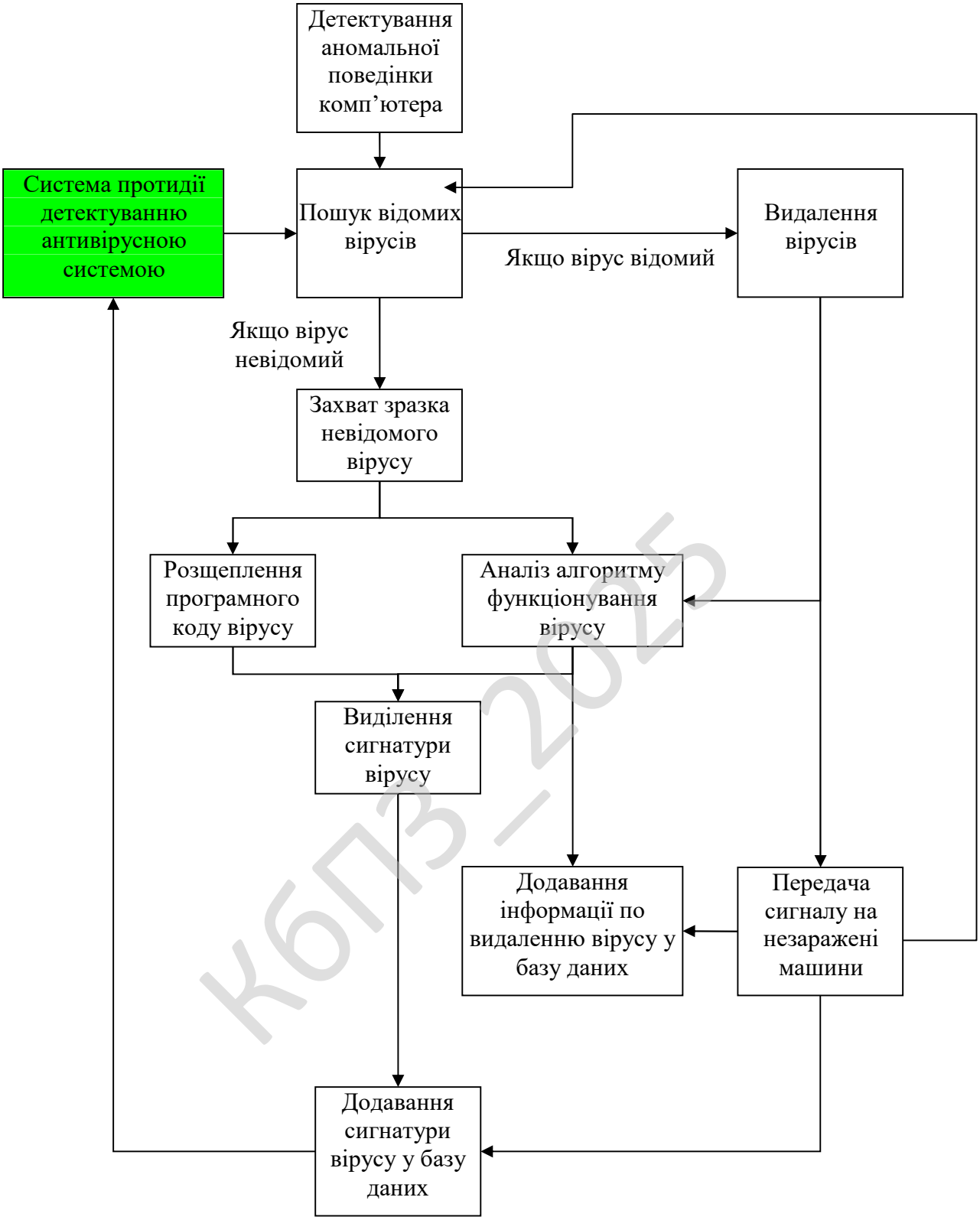

Рисунок 3.1 – Структурна схема системи

Перевірка сигнатур

Незважаючи на те, що ця техніка стара як світ, метод є дуже ефективним при перевірці файлів, що завантажуються з інтернету. Перевірка сигнатури sha1 – один з найбільш надійних способів уникнути зараження системи.

Перевірка контрольної суми файлу

Якщо є розходження між контрольною суму усередині заголовка образу й поточною контрольною сумою файлу, це означає, що файл був змінений. У системи безпеки варто впровадити механізм перевірки автентичності файлу за допомогою порівняння поточної контрольної суми й контрольної суми заголовка образу.

3.3 Розробка функціональної схеми

Функціональна система складається з двох великих блоків:

– Блок протидії детектуванню.

– Блок антивірусу.

Блок протидії детектуванню

Блок протидії детектуванню містить у собі наступні блоки:

– Обфускація (Дешифрування шелл-коду).

– Пакувальники (PE ін'єкція).

– Криптори (PE ін'єкція).

– Детектування динамічного аналізу.

– Контроль над привілеями секцій.

– Присутність нестандартних секцій.

– Перевірка сигнатур.

– Перевірка контрольної суми файлу.

Робота даних блоків описана в попередніх розділах.

Блок антивірусу

Крім блоку протидії антивірусу, у роботи розроблений й сам антивірус. Антивірус, що розроблений у результаті виконання дипломного проектування – захист вашого комп'ютера від шкідливих програм, що включає базові функції забезпечення безпеки вашого ПК.

Антивірус використовує новітні технології захисту, завдяки якому забезпечується безпека й стабільна роботу комп'ютера.

Основні функції антивірусу, що розроблений:

– Захист у режимі реального часу.

– Базовий захист при роботі в мережі Інтернет і з електронною поштою.

– Мінімальне завантаження комп'ютера.

– Інтуїтивно зрозумілий інтерфейс.

– Для повноцінного захисту комп'ютера крім антивірусу рекомендується використовувати міжмережний екран.

– Перевірка файлів, веб-сторінок, поштових і ICQ-повідомлень.

– Блокування посилань на заражені веб-сайти й сайти, що перехоплюють інформацію.

– Проактивний захист від невідомих погроз, заснована на аналізі поводження програм.

– Самозахист антивірусу, що розроблений попереджає погрозу вимикання з боку шкідливого ПЗ.

– Система миттєвого виявлення погроз, що моментально блокує нові шкідливі коди.

– Реалізовано модуль «Перевірка посилань», що попереджає про заражені або небезпечних веб-сайти.

– Проактивний захист нового покоління від невідомих погроз.

– Віртуальна клавіатура для безпечного введення логінів, паролів і номерів кредитних карт на веб-сторінках.

– Перевірка операційної системи й установлених програм на наявність уразливостей.

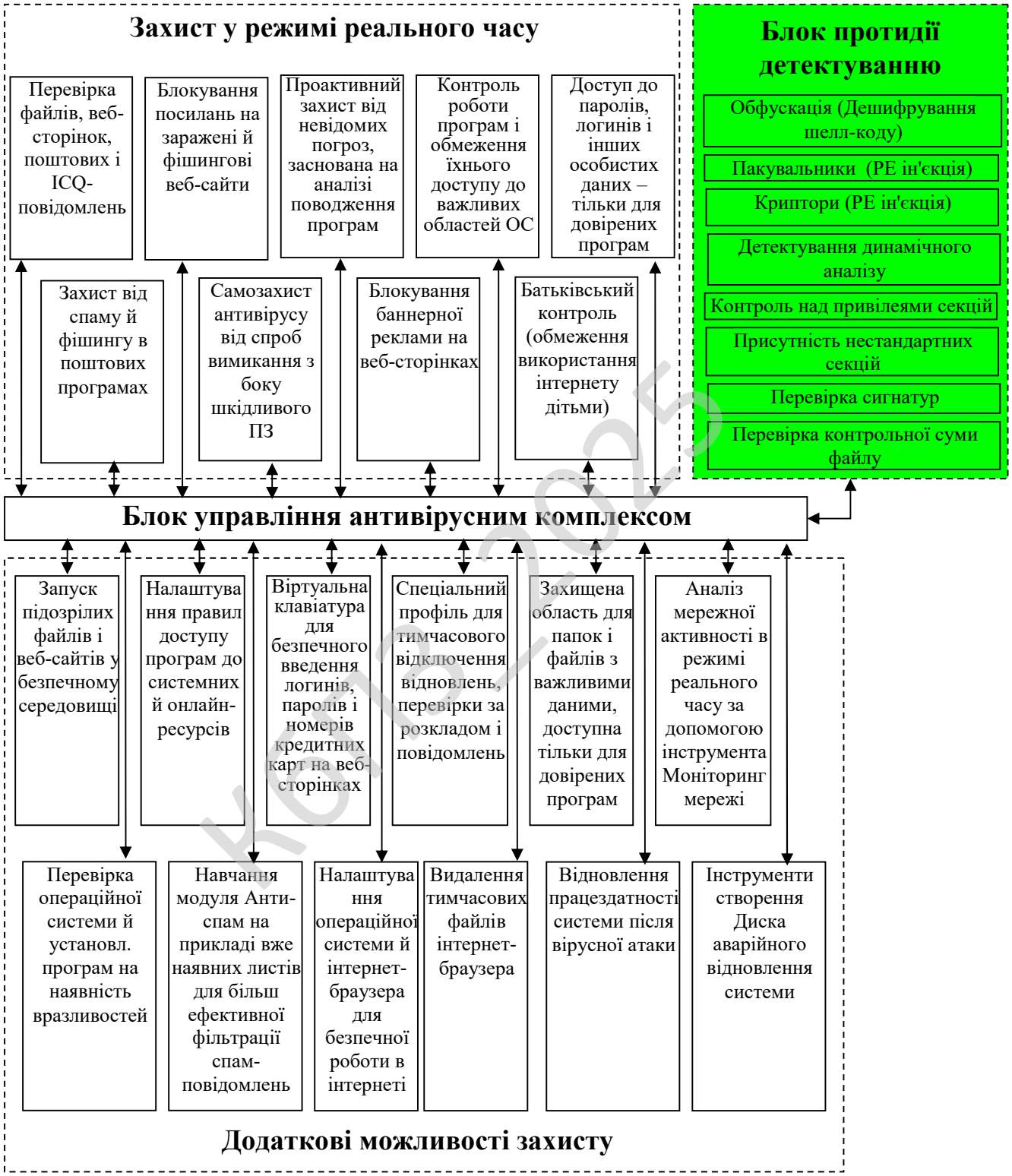

Рисунок 3.2 – Функціональна схема системи

– Налаштування операційної системи й інтернет-браузера для безпечної роботи в мережі Інтернет.

– Відновлення працездатності системи після вірусної атаки.

– Видалення тимчасових файлів інтернет-браузера.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

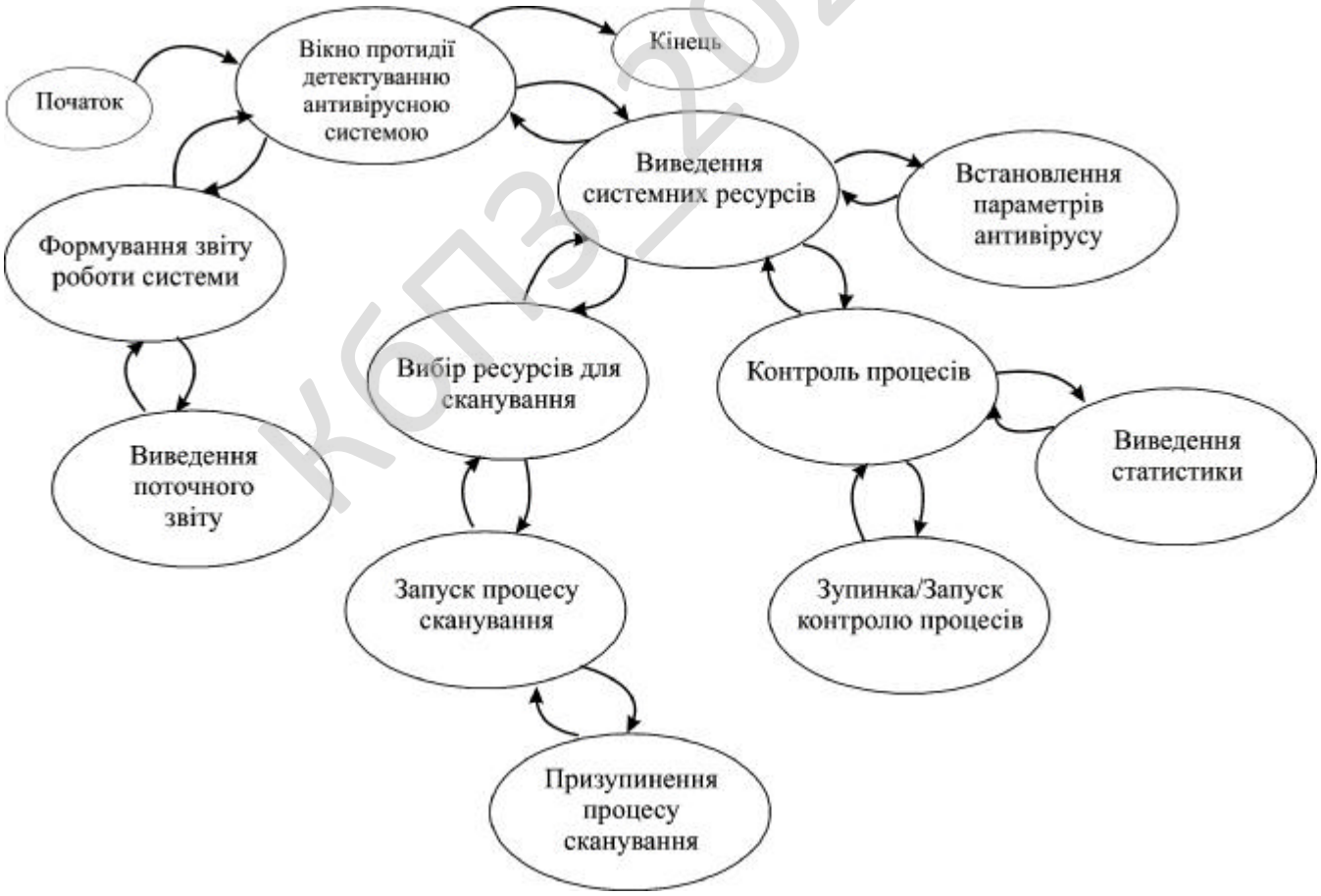

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над магістерською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine

написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або

підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання. показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю протидії детектуванню антивірусною системою, модулю обробки помилок програми і основному модулю.

Проекти в контрольованому середовищі (PRojects IN Controlled Environments – PRINCE) – це метод управління проектами. Метод включає в себе управління, контроль і організацію проекту. «PRINCE2» – це другий реліз зазначеного методу, що є зареєстрованим торговим знаком, державної торгово–промислової палати (Office of Government Commerce – OGC), незалежного підрозділу державного казначейства Сполученого Королівства.

PRINCE2 походить від більш раннього методу PROMPT та методу проектного управління PRINCE, який був розроблений у 1989 р. Центральним телекомунікаційним та комп'ютерним агентством (Central Computer and Telecommunications Agency – CCTA), як державний стандарт Великобританії з управління проектами у сфері інформаційних технологій (IT). Незабаром зазначений стандарт почали використовувати за межами IT сфери. У 1996 р. PRINCE2 був визнаний універсальним методом управління проектами. PRINCE2 став надзвичайно популярним і зараз де–факто є стандартом проектного управління в Великобританії.

Остання версія була випущена у 2009 р., як результат проекту оновлення Prince2:2009 державної торгово–промислової палати Великобританії.

PRINCE2:2009 Оновлення: З 2006 р. метод переглядався, а 16 Червня, 2009 р. був опублікований під назвою «Оновлення PRINCE2:2009». Назва «PRINCE2» (замість «PRINCE3» чи схожої) використано задля демонстрації збереження головних принципів методу.

Все ж таки, це найбільш суттєвий перегляд методу з 1996 р. з метою його адаптації до мінливого бізнес середовища, спрощення і «полегшення», а також кращої інтеграції з іншими методами державної торгово–промислової палати Великобританії (ITIL, P3O, P3M3, MSP, etc.). Головна різниця між релізом 2009 р. та більш ранніми релізами полягає в появі двох довідників замість одного. «Управління проектами, використовуючи PRINCE2» (Managing Projects Using PRINCE2) – оновлена методика управління, що призначена для менеджерів проектів, для яких управління проектами є щоденною діяльністю. «Стратегічне

управління проектами, використовуючи PRINCE2» (Directing Projects Using PRINCE2) – новий документ для Керівника проектів і Ради управління проектам, тобто спонсорів/замовників проектів.

Екзамен як з Теоретичної так і з Практичної частини буде базуватися на новому довіднику «Управління проектами» та не буде включати матеріали з довідника «Стратегічне управління проектами». Прохідний бал для Теоретичного екзамену залишиться не змінним. Прохідний бал для Практичного екзамену збільшиться з 50 % до 55 %. Тривалість екзамену з Практичної частини буде зменшено з 3 годин до 2,5 годин.

PRINCE2 – це структурований підхід до управління проектами, який спрямований на управління проектами в рамках чітко визначеної організаційної структури. PRINCE2 описує процедури координації людей та завдань в проекті, я к створювати/планувати проект та що робити, якщо проект потребує внесення змін через невідповідність фактичного стану виконання проекту плану його виконання. Кожний процес зазначено з ключовими вхідними та вихідними даними, а також з специфічними цілями та завданнями, які необхідно виконати, що загалом дає можливість контролю відхилень від плану.

Поділ методу на частини, що підлягають управлінню, забезпечує ефективний контроль ресурсів, завдяки чому можливо здійснювати контрольований та організований моніторинг проекту. PRINCE2 забезпечує єдину термінологію для всіх учасників проекту. Різноманітні ролі управлінців та зони відповідальності повністю описані та можуть бути адаптовані відповідно до складності проекту та компетенцій організації.

Помилки використання

Іноді PRINCE2 вважають не доцільним для дуже маленьких проектів через обсяг роботи, необхідної для створення та оновлення документів, протоколів та реєстрів. Досить часто це трапляється через нерозуміння, які частини PRINCE2 використовувати, адже PRINCE2 можливо повністю масштабувати.

Звичайно все це чудово, але сприйняття PRINCE2 погіршене тим, що для кожного ефективного проекту, який використовує методологію PRINCE2 існує певна кількість не ефективних проектів, які використовують, так звану, методологію PINO (PRINCE In Name Only – дослівно, PRINCE лише в назві).

Загальними проблемами є:

– Стадія початку проекту опускається/проштовхується і організація переходить відразу до створення Документу ініціювання проекту (Project initiation document – PID).

– Рада проекту не є ефективною – ескалації уходять в «чорну діру» тощо.

– Відсутній Опис продукту (Product description – PD), без якого управління якістю в PRINCE2 не працює.

– Припустимі відхилення не встановлюються, або встановлюються лише для часу виконання та вартості – чітко не визначено, в якій мірі повноваження передані виконавцям або коли необхідно використовувати ескалацію (піднімати рівень прийняття рішень).

– Проект фактично складається з однієї стадії (або «фази» використовуються, щоб уникнути Огляду виконання стадії завершення (End Stage Review)).

– Документи ініціювання проекту фактично копі–паст (copy–paste – копіювати – вставити) Документу ініціювання проекту з останнього проекту – їм не слідують і вони створюються тому що «потрібно».

– Низький рівень залучення «бізнесу», адже PRINCE2 вважається «технічним» методом – виходи (продукти) проекту можуть не призвести до бажаних результатів.

– Успіхи та невдачі використання PRINCE2 були однією з причин створення Моделі зрілості PRINCE2 (PRINCE2 Maturity Model – P2MM).

Модель зрілості PRINCE2. Модель зрілості PRINCE2 описує набір Сфер Ключових Процесів (Key Process Areas – KPAs), які необхідні для ефективного впровадження та використання PRINCE2 в організації. Довідник PRINCE2 описує

управління проектом і не включає опис процесів впровадження PRINCE2 в організації, в той час як P2MM описує саме процеси впровадження PRINCE2.

P2MM описує ключові практики, які додаються до процесів та складових PRINCE2 з метою забезпечення повторюваного використання методу (Рівень 2 KPAs), а також ключові практики необхідні для впровадження методу (Рівень 3 KPAs), як стандартного бізнес–процесу управління проектами, що включає: призначення власника (3.1), адаптацію методу (3.2), навчання (3.3), інтеграцію з іншими системами управління (3.4), механізм забезпечення якості (3.5).

Огляд методу

Блок–схема процесів PRINCE2. Стрілки означають потоки інформації. PRINCE2 – це процесуальний метод управління проектами на відміну від адаптивних методів, таких як Scrum. PRINCE2 2009 визначає 40 окремих активностей (завдань) і об'єднує їх у сім процесів.

Початок проекту.

На цьому етапі визначаються учасники проектної команди та готується короткий опис проекту (опис цілей проекту та комерційне обґрунтування проекту). Відповідно до загальних засад методу на цьому ж етапі планується наступний етап проекту. Після завершення зазначених робіт Рада проекту має погодити наступний етап проекту – ініціювання проекту.

Головні завдання включають: призначення керівника проекту та менеджера проекту, визначення та призначення команди управління проектом, створення короткого опису проекту, визначення методу управління проектом, планування наступного етапу проекту (ініціювання).

Ініціювання проекту

Цей етап базується на результатах виконання завдань етапу початку проекту. Короткий опис проекту розширюється до бізнес плану (Business case). Точки контролю якості проекту узгоджуються з точками контролю стану виконання проекту. Створюється план виконання наступного етапу проекту.

Результати етапу виносяться на погодження Ради проекту з метою погодження продовження роботи над проектом.

Головні завдання включають: планування показників якості, створення плану проекту, деталізація бізнес плану та ризиків, визначення точок контролю проекту, створення Документу ініціювання проекту (Project Initiation Document).

Стратегічне управління проектом

Цей процес визначає яким чином Рада проекту (що уособлює роль спонсору проекту) буде загалом контролювати проект. Рада проекту дозволяє/санкціонує етап ініціювання проекту, а також сам проект. Стратегічне управління також визначає яким чином Рада проекту має погоджувати план етапів, включаючи погодження будь-якого плану етапу, що може змінити існуючий план проекту у зв'язку з будь-якими змінами строків або ключових показників проекту.

Також визначається, яким чином Рада проекту може змінити план проекту, способи закриття проекту.

Головні завдання включають: засоби погодження ініціювання проекту, погодження проекту, погодження етапу або вилучення етапу проекту, підтвердження завершення проекту.

Контроль

PRINCE2 передбачає поділ проекту на етапи, які підлягають контролю. Загалом визначено яким чином пакети завдань призначаються та погоджуються. Також визначається яким чином відслідковується статус виконання завдань і яким чином такий статус доповідається Раді проекту.

Засоби для відслідковування та оцінювання виконання проекту пропонуються разом з способами застосування коригуючих дій.

Також закріплюються методи ескалації визначеного кола проблем Раді проекту.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

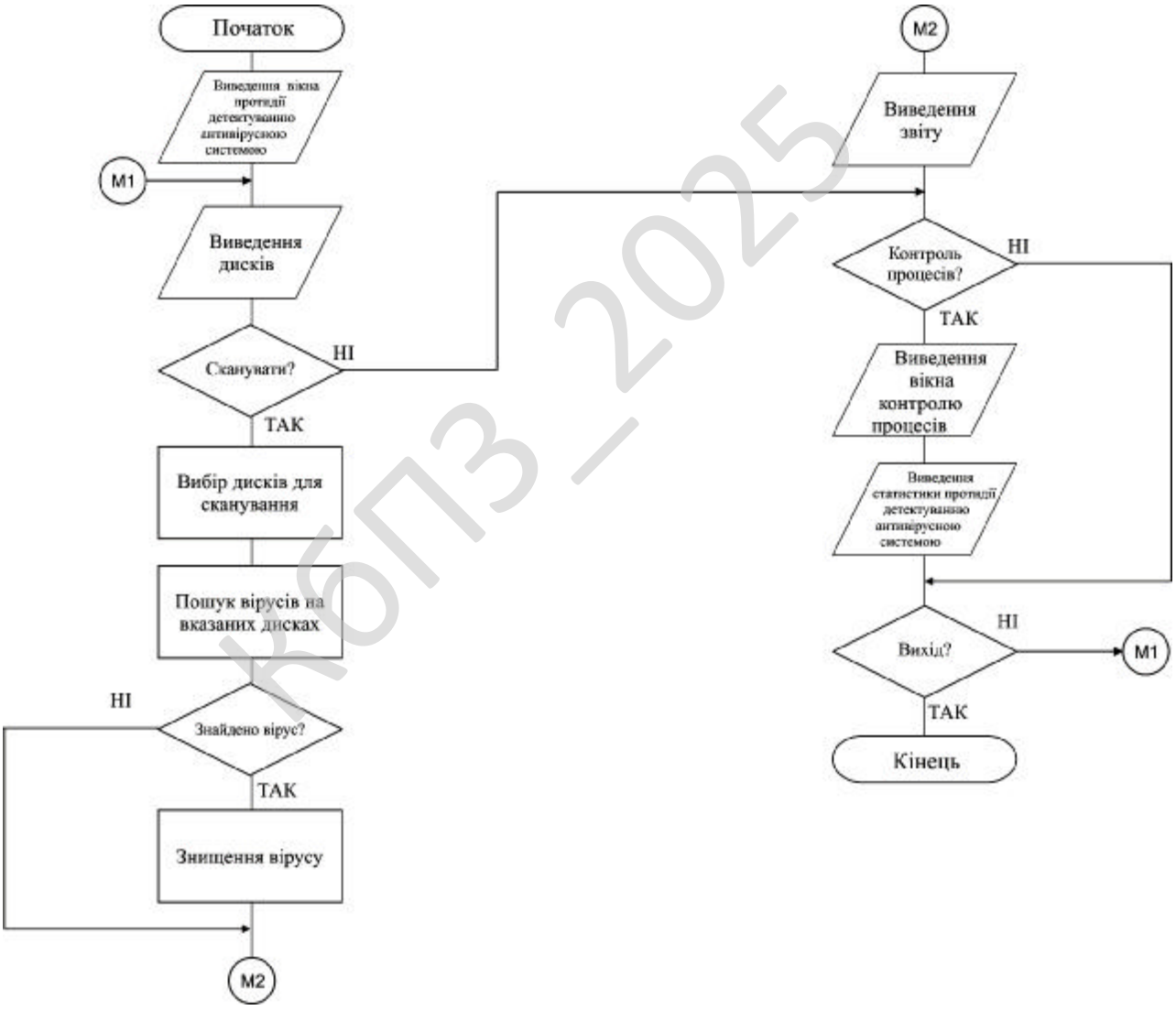

Рисунок 4.1 – Блок-схема основної програми

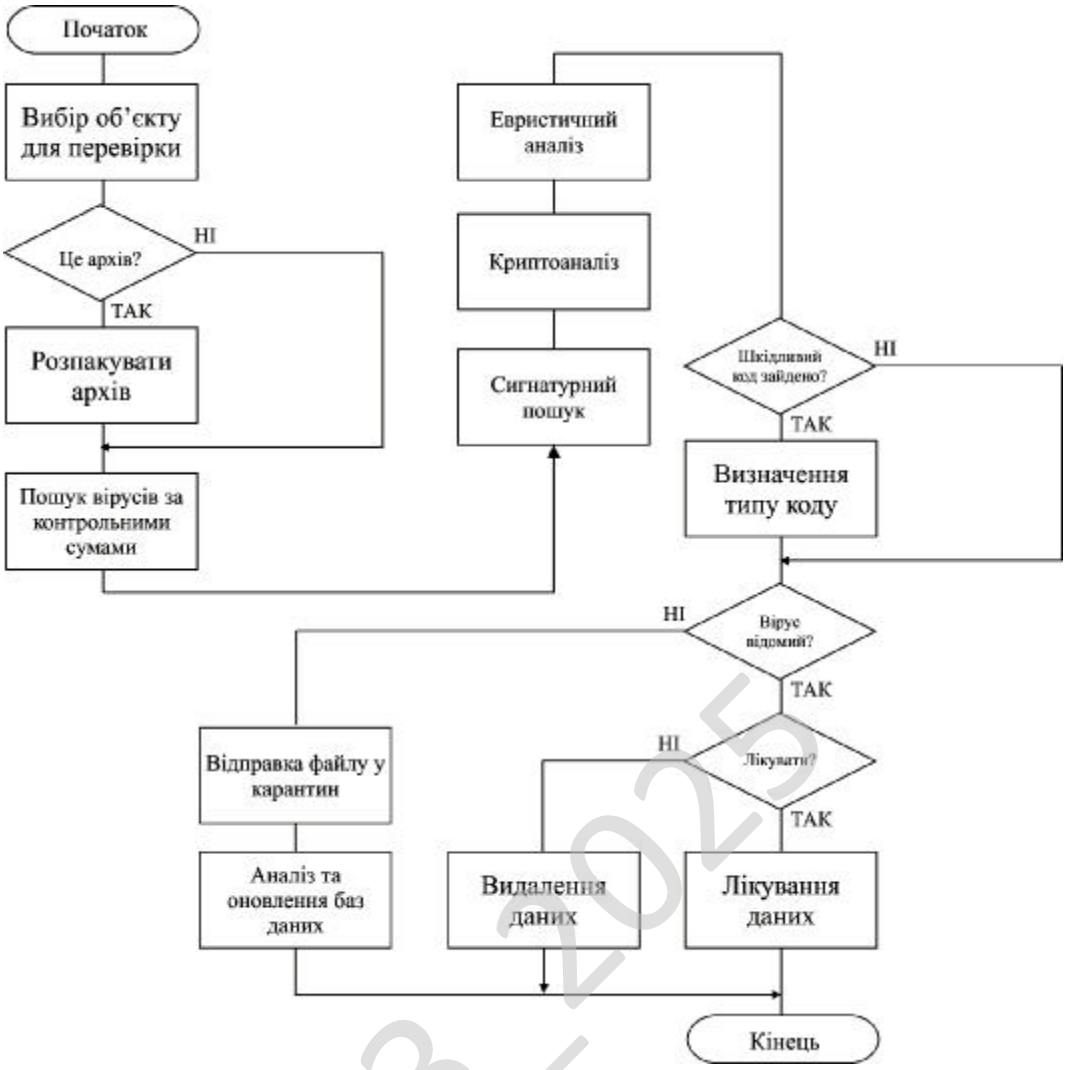

Рисунок 4.2 – Блок-схема роботи підпрограми

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Serpent – симетричний блочний алгоритм шифрування, розроблений Россом Андерсоном, Елі Біхамом та Ларсом Кнудсеном. Алгоритм був одним з фіналістів 2-го етапу конкурсу AES. Як і інші алгоритми, які брали участь у конкурсі AES, Serpent має розмір блоку 128 біт і можливі довжини ключа 128, 192 або 256 біт. Алгоритм являє собою 32-раундовий шифр на основі SP-мережі, і працює з блоком з чотирьох 32-бітових слів. Serpent був розроблений так, що всі

операції можуть бути виконані паралельно, використовуючи 32-а 1-бітних «потоків».

При розробці Serpent використовувався консервативніший підхід до безпеки, ніж у інших фіналістів AES, проектувальники шифру вважали, що 16 раундів достатньо, щоб протистояти відомим видам криптоаналізу, але збільшили число раундів до 32, щоб алгоритм міг краще протистояти ще не відомим методам криптоаналізу.

Шифр Serpent не запатентований і є громадським надбанням.

Алгоритм створювався під гаслом «криптографічний алгоритм 21 століття» для участі в конкурсі AES. При створенні нового алгоритму Serpent його автори дотримувалися консервативних поглядів на проектування, що підтверджується первісним рішенням про використання таблиць підстановки з відомого багато років раніше алгоритму шифрування DES, який протягом довгого часу вивчався провідними фахівцями в області криптографії та захисту інформації і чиї властивості і особливості були добре відомі науковому світу. Одночасно з цим до нового алгоритму міг бути застосований вичерпний аналіз, вже розроблений для DES. Не використовувалися нові, неперевірені і невипробувані технології при створенні шифру, який у разі прийняття був би використаний для захисту величезних масивів фінансових транзакцій та урядової інформації. Основною вимогою до учасників конкурсу AES було те, що алгоритм-претендент повинен бути швидшим, ніж 3DES, і надавати як мінімум такий же рівень безпеки: він повинен мати блок даних довжиною 128 біт і ключ завдовжки 256 біт. 16-раундовий Serpent був би таким же надійним, як 3DES, але в два рази швидшим. Однак, автори вирішили, що для більшої надійності варто збільшити кількість раундів до 32. Це зробило шифр таким же швидким, як DES, і набагато надійнішим, ніж 3DES.

Структура алгоритму

Алгоритм Serpent є SP-мережею, у котрій весь блок даних довжиною 128 біт на кожному раунді розбивається на 4 слова довжиною 32 біти. Всі значення,

що використовуються при шифруванні є бітовими потоками. Бітові індекси змінюють значення від 0 до 31 для 32-бітових слів, від 0 до 127 – для 128-бітових блоків та від 0 до 255 для 256-бітових ключів тощо. Для внутрішніх обчислень всі біти величин представлені в прямому порядку (little-endian).

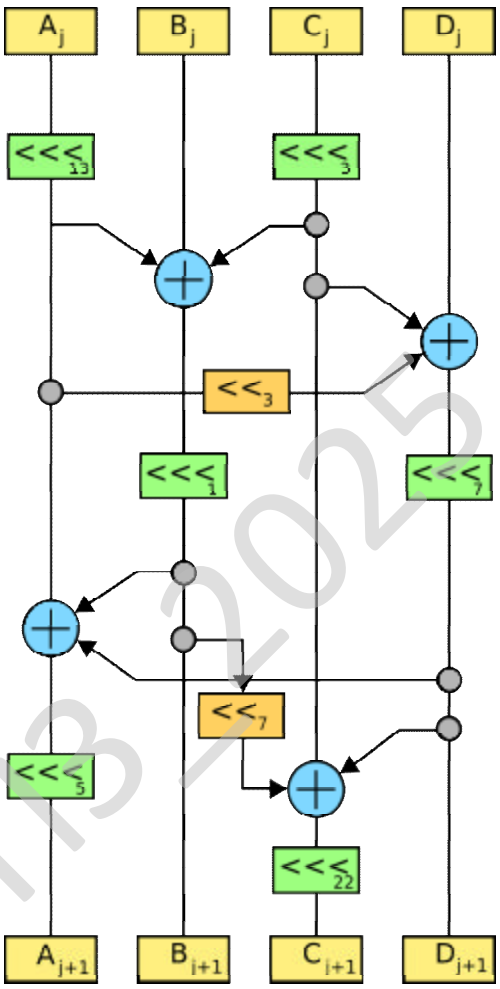

Рисунок 4.3 – Структура алгоритму Serpent

Serpent шифрує відкритий текст P довжиною 128 біт в шифротекст C довжиною таких же 128 біт за 32 раунд за допомогою 33 подключів K{0},...,K{32}, довжиною 128 біт. Довжина використовуваного ключа може приймати різні значення, але для конкретики зафіксуємо їх довжину в 128, 192 або 256 біт. Короткі ключі довжиною менше 256 біт доповнюються до повної довжини в 256 біт.

Шифрування складається з наступних основних кроків:

– Початкова перестановка.

– 32 раунд, кожен з яких складається з операції змішування з 128-бітовим ключем (побітове логічне виключаюче «або»), таблична заміна (S-box) і лінійне перетворення. В останньому раунді лінійне перетворення замінюється додатковим накладанням ключа.

– Кінцева перестановка.

Початкова і кінцева перестановки не мають будь-якої криптографічного значущості. Вони використовуються для спрощення оптимізованої реалізації алгоритму і підвищення обчислювальної ефективності.

Розширення ключа

Як і інші алгоритми, що брали участь в конкурсі AES, Serpent має можливі довжини ключа 128, 192 або 256 біт. «Неповний» ключ довжиною менше 256 біт доповнюється за наступним правилом: додається одиничний біт справа, за ним слід стільки нульових бітів, щоб довжина ключа стала дорівнює 256 бітам.

Початкова перестановка IP

Дана перестановка IP задається таблицею, де вказується позиція, на яку перейде відповідний біт (наприклад, біт 1 перейде на 32 позицію):

S-бокси (таблиці замін)

В алгоритмі Serpent таблиці замін є 4-бітовими перестановками з властивостями стійкості до диференціального криптоаналізу, до лінійного криптоаналізу і такою властивістю, що порядок вихідних біт, як функції вхідних повинен бути максимальний, тобто бути рівним 3.

Таблиця підстановки генерується з відомих і добре вивчених таблиць для алгоритму DES в ітераційному процесі, поки не будуть отримані бажані диференціальні й лінійні властивості. Таким чином, створюється 8 таблиць підстановки.

Лінійне перетворення LT

Лінійне перетворення LTзадається таблицею, де біти перераховані від 0 до 127 (наприклад, вихідний 2 біт утворений 2, 9, 15, 30, 76, 84, 126 битами, складеними за модулем 2) . В кожному рядку описується 4 вихідних біти, які разом складають вхідні дані на одну таблицю замін в наступному раунді. Варто зазначити, що даний набір являє собою таблицю IP(LT(FP(x))), де LT і є те лінійне перетворення.

Таблиця зворотного лінійного перетворення, яке використовується при розшифровці ILT.

Кінцева перестановка FP

Дана перестановка є зворотною до початкової, тобто $FP=IP^{-1}$ і задається наступною таблицею.

Ефективна реалізація алгоритму

Бажання авторів зробити алгоритм саме таким, яким він є стає зрозумілим при розгляді його ефективної низькорівневої реалізації.

Serpent був створений таким чином, щоб всі операції в процесі шифрування і розшифрування одного блоку могли бути виконані паралельно в 32 потоках. До того ж низькорівневий опис алгоритму набагато простішій, ніж стандартний опис. Ніяких початкових і кінцевих перестановок не потрібно.

Шифрування складається з 32 раундів. Відкритий текст є першими проміжними даними $B_0 = P$. Потім виконується 32 раунди, кожен і-й раунд складається з:

– Змішування з ключем. Проводиться побітове виключаюче «або» проміжних даних B_i з ключем довжиною 128 біт.

– Застосування таблиць підстановки. Вхідні дані довжиною 128 біт поділяються на 4 слова по 32 біта. Таблиця підстановки, реалізована послідовністю логічних операцій (як якщо це було б реалізовано апаратно), застосовується до цих 4 словам. В результаті виходить 4 вихідних слова. Таким

чином, центральний процесор виконує підстановку по 32 копій таблиці одночасно.

– Лінійне перетворення. 32-бітові слова перетворюються заданим порядком.

Першою причиною вибору такого лінійного перетворення є максимізація лавинного ефекту. Такі таблиці підстановки мають властивість, що зміна кожного вхідного біта призведе до зміни 2 вихідних бітів. Таким чином, кожен вхідний біт відкритого тексту вже через 3 раунди впливає на всі вихідні біти. Аналогічно кожен біт ключа впливає на результат шифрування.

Друга причина полягає в простоті перетворення. Воно може бути реалізоване на будь-якому сучасному процесорі з мінімальними витратами.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ протидії детектуванню антивірусною системою яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Розділу обрання групи: Загальне; Контроль процесів; Додатково; Фільтр сканування; Шляхи сканування; Параметри.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

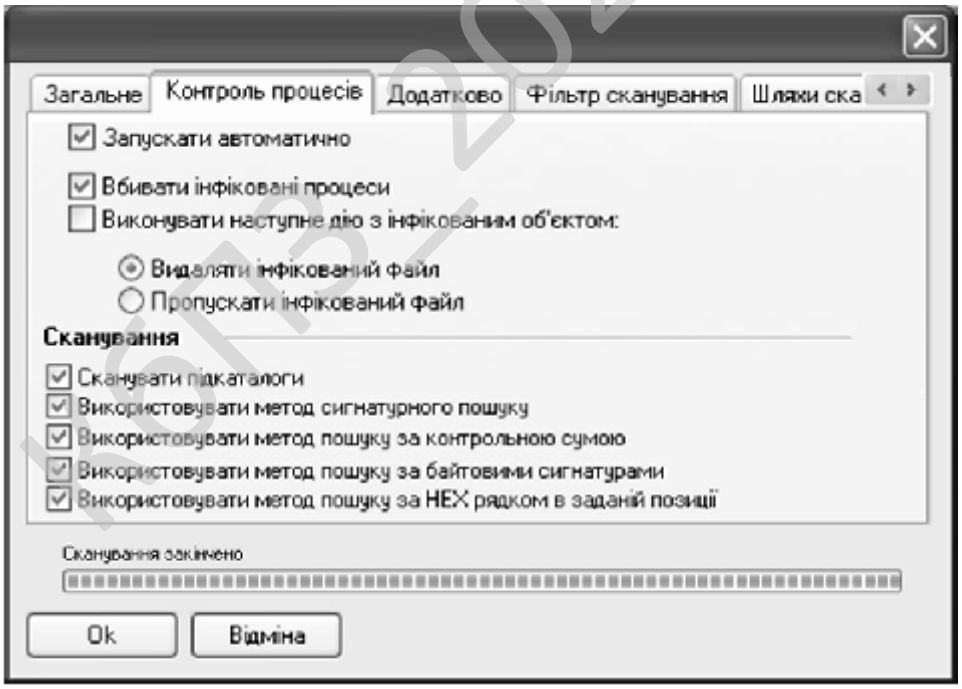

Рисунок 5.1 – Головне вікно ПЗ

Система призначена для обходу статичного, динамічного й евристичного аналізу, використовуваного в новітніх антивірусних продуктах. При цьому в даній роботі буде реалізована й антивірусна складова.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

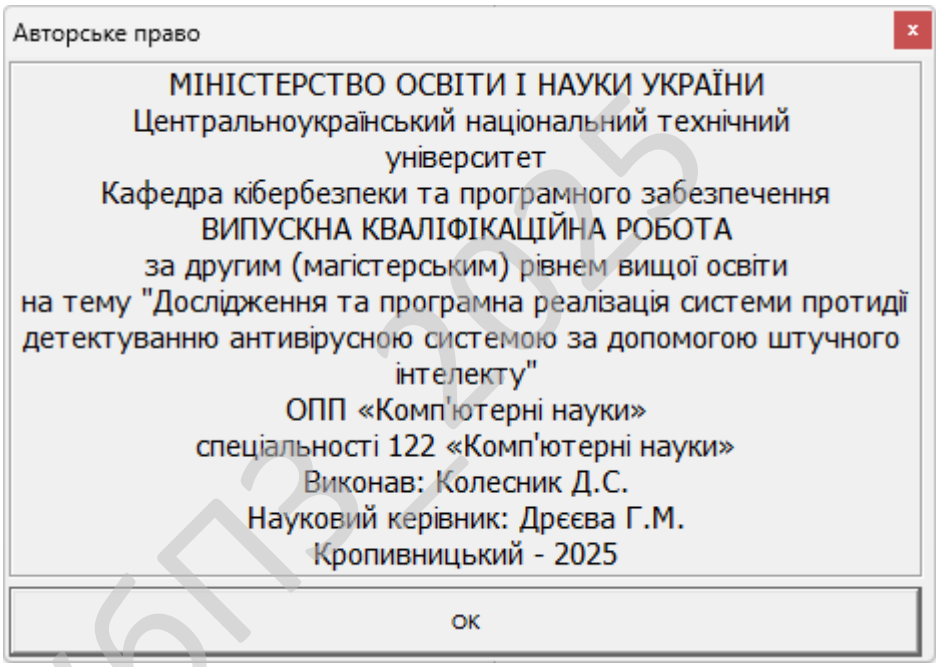

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати ІТ рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих

процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень. Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії. Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення. На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Метою розробки є дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Об'єктом дослідження є процес протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Предметом дослідження є методи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Методи дослідження базуються на методах штучного інтелекту, методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– Розроблено вітчизняний продукт протидії детектуванню антивірусною системою за допомогою штучного інтелекту, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження системи протидії детектуванню антивірусними програмами, що базується на штучному інтелекті, можуть бути цікавими передусім розробникам програмного забезпечення, які прагнуть мінімізувати ризики помилкових спрацьовувань антивірусів (false positive). Для компаній, що займаються створенням комерційного або наукового ПЗ, хибні блокування можуть означати втрату клієнтів, погіршення репутації бренду та збільшення витрат на технічну підтримку. Тому рішення, яке дозволяє прогнозувати ризики детектування ще до релізу продукту, має високу практичну цінність.

Крім того, проєкт може зацікавити компанії, що працюють у сфері кібербезпеки. Для них інтеграція такої системи відкриває можливість створення більш гнучких антивірусних модулів, здатних адаптуватися до складних шаблонів поведінки програм. Це важливо для підвищення точності аналізу файлів та зменшення кількості помилкових рішень.

Також результати дослідження можуть бути корисними для наукових установ та ІТ-факультетів університетів, які досліджують прикладне використання штучного інтелекту у сфері інформаційної безпеки. Для державних структур або лабораторій сертифікації ПЗ така система може стати інструментом попереднього аналізу безпечності продуктів перед їх офіційним схваленням чи розповсюдженням.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінку привабливості цього проєкту можна провести шляхом залучення експертів у сфері кіберзахисту, машинного навчання та розробки програмного забезпечення. Наприклад, група з десяти фахівців оцінює систему за кількома критеріями: інноваційність технології, економічна доцільність, простота інтеграції, потенціал комерціалізації та рівень ризиків. Кожен експерт виставляє оцінку за шкалою від 1 до 10, а потім ці значення усереднюються з урахуванням вагових коефіцієнтів, що відображають важливість кожного параметра.

Якщо за підсумками опитування інноваційність отримує середню оцінку 9, економічна вигода – 8, інтеграційна простота – 7, а потенціал ринку – 9, то з урахуванням вагових коефіцієнтів загальний індекс привабливості може сягати 8,5 із 10. Такий результат вказує на високу комерційну перспективу проєкту.

Цей метод дає змогу об’єктивно оцінити не лише технічну спроможність системи, а й ринкову зацікавленість, тобто наскільки інновація здатна привернути увагу бізнесу, розробників і споживачів. Отримані результати можуть бути основою для прийняття рішення про інвестування в подальшу розробку та впровадження продукту.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості впровадження системи доцільно використати метод витратного підходу в поєднанні з методом дисконтованих грошових потоків (DCF). Перший дозволяє оцінити всі прямі витрати, пов’язані з розробкою, навчанням моделей штучного інтелекту, тестуванням та впровадженням системи. Це надає чітке уявлення про обсяг необхідних інвестицій.

Метод дисконтованих потоків, своєю чергою, допомагає врахувати очікувані майбутні вигоди – наприклад, зниження кількості інцидентів

помилкового детектування, економію часу розробників і зменшення навантаження на службу підтримки. Застосування цього методу дозволяє побачити, як швидко проєкт може окупитися, а також розрахувати показники рентабельності та чистої теперішньої вартості (NPV).

Таке поєднання підходів є найбільш збалансованим, оскільки дозволяє врахувати як фактичні фінансові витрати, так і потенційні нематеріальні вигоди, зокрема підвищення конкурентоспроможності компанії та довіри користувачів до її продуктів.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Сучасні антивірусні системи базуються на сигнатурному аналізі та евристичних алгоритмах, проте зі зростанням складності програмного забезпечення виникає потреба у створенні технологій, що дозволяють програмам працювати стабільно, не викликаючи хибних спрацьовувань антивірусів.

Мета розробки – створити інтелектуальну систему на основі AI, яка аналізує структуру коду, дії процесів і поведінкові патерни, прогнозуючи ризик детектування ще до виконання програми. Це допоможе легітимним розробникам уникнути втрат від помилкового блокування програм антивірусами (false positive), зменшити кількість звернень до техпідтримки та підвищити репутацію продукту на ринку.

Підприємство, яке впроваджує таку систему, може зменшити непрямі фінансові втрати від “помилкових спрацьовувань”, що призводять до падіння продажів, негативних відгуків і зниження довіри користувачів. Вхідні дані зафіксовано в таблиці 7.1.

Таблиця 7.1 – Вихідні дані для розрахунку

Показник	До впровадження	Після впровадження	Економічний ефект
Кількість випадків хибного блокування продукту антивірусами, на рік	60	10	–50
Середня вартість втрат від одного інциденту (зменшення продажів, репутаційні ризики), грн	70 000	20 000	–50 000
Річні збитки від антивірусних детектів, грн	4 200 000	200 000	–4 000 000
Витрати на обслуговування клієнтів через хибні блокування, грн/рік	600 000	300 000	–300 000
Початкові інвестиції у розробку системи	—	2 500 000	—
Щорічні експлуатаційні витрати (оновлення моделей, техпідтримка)	—	400 000	—

Розрахунок економічного ефекту демонструє наступне: зменшення прямих збитків від детектування – 4 000 000 грн/рік, економія на обслуговуванні клієнтів – 300 000 грн/рік, зростання доходів за рахунок стабільності продукту (очікуване збільшення продажів завдяки покращенню репутації – +10%, що становить 1 000 000 грн/рік), сукупний річний економічний ефект – 5 300 000

грн/рік, чистий економічний ефект – 4 900 000 грн/рік, термін окупності ≈ 0,51 року (≈ 6 місяців), рентабельність інвестицій – 196 %.

Додаткові нефінансові переваги: покращення іміджу бренду, зменшення навантаження на технічну підтримку, автоматизація процесу тестування, підвищення якості програмного коду, конкурентна перевага.

Таким чином, система AI Anti-Detection є не лише інструментом зменшення витрат, а й інвестицією у довгострокову стабільність, реноме та технологічну перевагу підприємства.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Початковим етапом просування має стати створення демонстраційного прототипу системи, який би показував її здатність зменшувати кількість хибнопозитивних спрацьовувань на конкретних прикладах програмного забезпечення. Такий підхід дозволяє одразу продемонструвати практичну цінність рішення.

Далі варто організувати серію пілотних інтеграцій із компаніями, які вже стикаються з проблемою помилкових детектів. Це дозволить отримати реальні відгуки користувачів, зібрати статистику ефективності системи та вдосконалити алгоритми. Важливо забезпечити прозору систему зворотного зв’язку, щоб партнери бачили конкретні результати.

На етапі масштабування доцільно залучити профільні ЗМІ, виступи на конференціях і публікацію кейсів успіху. Просування має супроводжуватись акцентом на тому, що система не обходить антивірус, а сприяє підвищенню якості програмного коду, роблячи його безпечним і сумісним з існуючими безпековими політиками.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для підвищення ефективності збуту можна налагодити співпрацю з великими розробниками антивірусного програмного забезпечення та платформами сертифікації ПЗ. Якщо така система стане частиною процесу тестування програм перед релізом, це створить стабільний канал продажів через корпоративні клієнти. Також можна впровадити модель SaaS, яка дозволить клієнтам користуватися системою через вебінтерфейс без необхідності встановлення програмного забезпечення. Така стратегія значно спрощує вихід на міжнародний ринок і зменшує витрати на впровадження. Ще одним ефективним шляхом може бути створення партнерських програм із навчальними закладами або технічними хабами, які готують фахівців у галузі безпеки. Це сприятиме поширенню технології серед майбутніх розробників і формуватиме довгострокову впізнаваність бренду.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовим фактором успіху цього проєкту є якість алгоритмів штучного інтелекту та їхня здатність точно прогнозувати ймовірність помилкових детектувань без шкоди для безпеки продукту. Висока точність і мінімальна кількість хибних результатів створюють довіру з боку користувачів і розробників. Не менш важливим фактором є постійне оновлення системи на основі нових даних і трендів у кіберзахисті. Середовище безпеки змінюється щодня, тому система має бути здатна адаптуватися до нових антивірусних алгоритмів і загроз. Важливим чинником також є ефективна комунікація між розробниками і бізнесом. Якщо команда зможе продемонструвати не лише технічну перевагу, а й конкретну економічну вигоду, це значно підвищить шанси на впровадження. Успіх проєкту визначатиметься не тільки інноваційністю технології, а й її здатністю вирішити реальні проблеми галузі, забезпечуючи баланс між безпекою, продуктивністю та довірою користувачів.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці – система збереження життя і здоров'я працівників у процесі трудової діяльності, що включає правові, соціально-економічні, організаційні, технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ-компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ-компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з

охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2].

Постійне перебування за комп'ютером негативно впливає на органи зору та поставу. Тому вже після першого року роботи, багатьом фахівцям у сфері IT знайомі такі хвороби, як остеохондроз, синдром сухого ока, безсоння та інше.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (IT) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової діяльності. Їх праця стала більш інтенсивною, напруженою і вимагає значних витрат розумової, емоційної і фізичної енергії. Це призвело до необхідності у знаходженні комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку. Охорона здоров'я робітників, забезпечення безпеки умов праці, ліквідація та профілактика професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

8.2 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і

прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

8.3 Характеристика умов праці програміста

В приміщенні, в якому проводиться розробка і дослідження програмного продукту, відсутні умови, які можуть створювати підвищену або особливо підвищену небезпеку, тому воно відноситься до класу звичайних приміщень згідно Правил улаштування електроустановок (ПУЕ). Джерелом живлення є трифазна мережа напруги 380/220 В з глухо заземленою нейтралі, з частотою 50 Гц згідно За вибухопожежонебезпекою приміщення відноситься до класу В. В таблиці 8.1 наведена загальна характеристика приміщення щодо вибухопожежонебезпеки та важкістю робіт.

Температура повітря в приміщенні визначається температурою зовнішнього повітря і тепловою енергією, що виділяється всередині приміщення. Джерелами теплоти в даному приміщенні є люди, електроустаткування, а також освітлювальні прилади в темний час доби. Зовнішнім джерелом надлишкового тепла є сонячна радіація у світлий час доби. Робота, виконувана в даному приміщенні, відноситься до категорії I-а. Людиною в цьому випадку виділяється до 120 ккал теплової енергії за годину. Вологість повітря в приміщенні

визначається вологістю атмосферного і видихуваного людьми повітря, а також випарами з поверхні шкіри.

Таблиця 8.1 – Загальна характеристика приміщення щодо вибухопожежонебезпеки та важкістю робіт

Характеристика приміщень за вибухопожежною категорією та класом зони	Загальна характеристика приміщення	Категорія за важкістю робіт згідно ГН 3.3.5-8.6.6.1-2002
В – пожежонебезпечне клас П – II	Звичайне без ознак хімічного забруднення; нормальної вологості і за санітарними нормами	1а – до 139 Вт/м^2 1б – до 140-174 Вт/м^2 Клас умов праці – оптимальний

У приміщенні немає виділення шкідливих газів. Тому що в ньому не проводяться монтажні роботи, як пайка, зварювання чи інші, при яких виділяються шкідливі гази.

Для нормалізації параметрів повітряного середовища також періодично здійснюється провітрювання приміщення і вологе прибирання. У всьому будинку діє встановлена загально обмінна витяжна вентиляція.

Раціональне освітлення приміщення сприяє кращому виконанню виробничих завдань і забезпеченню комфорту при роботі. Для забезпечення нормального освітлення застосовуються природне, однобічне, бічне і штучне освітлення, а також комбіноване, нормуються згідно ДБН В.2.5-28-2006 Природне і штучне освітлення. За результатами виміру освітленості величина освітленості від системи загального штучного освітлення дорівнює 310 лк, що відповідає вимогам, які пред'являються до даного приміщення [1].

Основними джерелами шуму на робочих місцях, обладнаних відео дисплейними терміналами, є принтер, сканер факс і обладнання для

кондиціювання повітря, в самих відео дисплейних терміналів – вентилятори систем охолодження і трансформатори.

Згідно ДСанПіН 3.3.2.007-98 [2] допустимий еквівалентний рівень шуму для робочого місця програміста складає 50 дБА (акустичних децибел).

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга) [4, 10, 11].

8.5 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток 63х63х6 мм, (згідно з ДСТУ 2251-93 «Кутики сталеві гарячекатані рівнополичні. Сортамент») довжиною L=2 м, та горизонтальний електрод – металева полоса з перетином 60х5 мм. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – по контуру прямокутником (рис. 8.1).

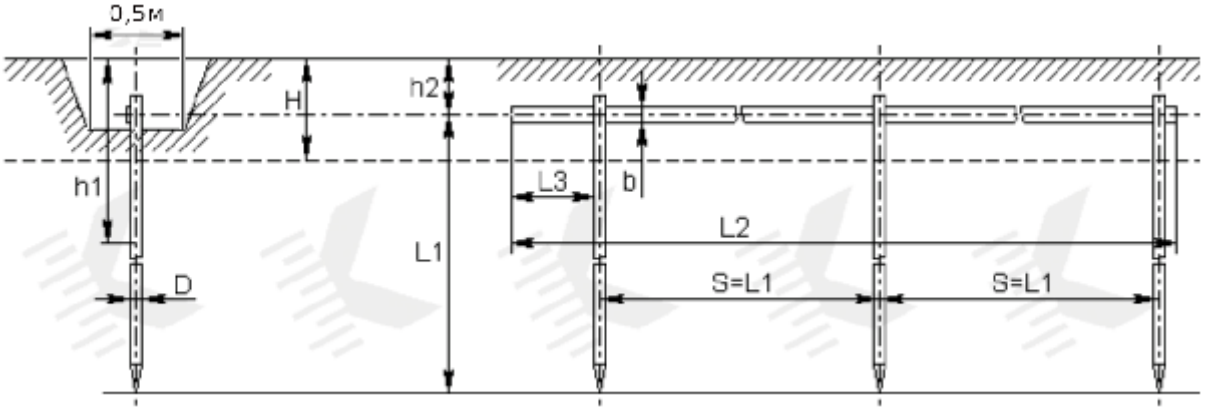

Рисунок 8.1 – Схема штучного заземлення

Розрахунок проведемо за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару ґрунту – чорнозем, нижнього шару ґрунту – глина (питомий опір ρ_2 = 40 Ом·м). Умовна товщина верхнього шару ґрунту: H=0,5 м. Відстань між вертикальними заземлювачами (електродами) A=2 м. Глибина закладення горизонтального контура заземлення t=0,6 м. Опір заземлювача, який нормується: $R_{ЗН}$ = 4 Ом. Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Виконаємо розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T = t + L/2 = 0,6 + 2/2 = 1,6 \text{ м.}$$

Розрахунковий питомий опір ґрунту (з урахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі ґрунту):

$$\rho = \psi \cdot \rho = 1,36 \cdot 40 = 54,5 \text{ Ом·м.}$$

де

ψ = 1,36 – табличне значення коефіцієнта сезонності для відповідної кліматичної зони у багатошаровому ґрунті [12];

ρ_2= 40 Ом·м – табличне значення питомого опору нижнього шару ґрунту (глина) [12].

Еквівалентний діаметр вертикального електрода (кутка) [12]:

$$D_в=0,95\cdot К = 0,95\cdot 63 = 59,85 \text{ мм} =0,0598 \text{ м}.$$

де К= 63 мм. – розмір металевого кутка (заданий).

Відношення A/L=2/2=1

Опір розтіканню електричного струму одного електроду вертикального заземлювача з урахуванням заглиблення заземлювача [12]:

$$R_О=0,366\cdot(\rho/L)\cdot[\lg(2L/D_в)+(1/2)\cdot\lg((4T+L)/(4T-L))]=$$

$$=0,366\cdot(54,5/2)\cdot[(\lg(2\cdot2/0,0598)+(1/2)\cdot\lg((4\cdot1,6+2)/(4\cdot1,6-2))] =19,6 \text{ Ом}.$$

Визначаємо коефіцієнт екранування вертикальних електродів Кев=0,62 при орієнтовній кількості вертикальних електродів, яке дорівнює 5 [12].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без врахування горизонтального заземлювача), при $R_{ЗН}$ = 4 Ом:

$$N=R_О /(К_{ев}\cdot R_{ЗН}) = 19,6/(0,62\cdot4) = 7,8 \approx 8 \text{ шт}.$$

Визначаємо довжину з'єднуючої полоси:

$$L_П =1,05\cdot A\cdot N= 1,05\cdot2\cdot8=16,6\approx 16 \text{ м}.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору ґрунту $K_П$ [12]:

$$R_П =0,366\cdot(\rho\cdot K_П/L_П)\cdot\lg(2\cdot L_П^{\,2}/(B\cdot t))=$$

$$=0,366\cdot(40\cdot5/16)\cdot\lg((2\cdot16^{\,2})/(0,06\cdot0,6)) =20,2 \text{ Ом}.$$

де $K_П$=5 – табличне значення кліматичного коефіцієнта питомого опору ґрунту для відповідної кліматичної зони для з'єднуючої полоси [12]:

B = 60 мм. = 0,06 м – ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [12]:

$$R =(R_О\cdot R_П)/(R_О\cdot\eta_П + N\cdot R_П\cdot К_{ев})=$$

$$=(19,6\cdot20,2)/(19,6\cdot0,6 + 8\cdot20,2\cdot0,62) =3,53 \text{ Ом}.$$

де $\eta_П$ = 0,6 – табличне значення коефіцієнта екранування з'єднуючої полоси [12].

Умова $R \le R_{ЗН}$ виконується ($3,53 \le 4$).

Остаточно отримали: кількість вертикальних електродів дорівнює 8 при R = 3,53 Ом.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– Досліджена система протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

– На основі отриманих результатів досліджень створена програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання протидії детектуванню антивірусною системою за допомогою штучного інтелекту.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Serpent.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Колесник Д.С. Дослідження та програмна реалізація системи протидії детектуванню антивірусною системою за допомогою штучного інтелекту // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

2. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.

3. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.

4. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.

5. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

6. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

7. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

8. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

9. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

10. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

11. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

12. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

13. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

14. Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед та ін. «Системи інформаційної зброї та технології інформаційної війни»: підручник / Петрик В.М., Присяжнюк М.М., Аль-Файюмі Халед, Жарков Я.М., Смірнов О.А, Буравченко К.О., Давидюк А.В., Кононович В.Г., Корчинский В.В., Кудирко В.М., Фесенко А.О.; за заг. ред. В.М. Петрика, М.М. Присяжнюка.– К.: Видавничий центр "Кафедра", 2025.– 320 с.

15. Усік, П.С., Смірнова, Т.В., Буравченко, К.О., Смірнов, О.А., Улічев, О.С., Смірнов, С.А. «Дослідження технологій забезпечення кібербезпеки банківських систем з використанням штучного інтелекту». *Кібербезпека: освіта, наука, техніка.* 2025. Том 1 № 29. С.704–716, 2025

16. Kuznetsov, O., Frontoni, E., Kuznetsova, K., Arnesano, M., Smirnov, O. «A secure biometric authentication architecture for blockchain-driven cyber-physical systems». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 193–224.

17. Kuznetsov, O., Atzeni, G., Arnesano, M., Randieri, C., Smirnov, O. «Secure IoT-based smart wheelchair system: From implementation to security enhancement strategy». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 225–257.

18. Kuznetsov, O., Smirnov, O., Kuznetsova, T., Shaikhanova, A., Svatowsky, I. «Privacy-utility trade-offs in IoT networks: A comparative analysis of differential privacy mechanisms for sensor data aggregation». *Security and Privacy of Cyber Physical Systems Emerging Trends Technologies and Applications*, 2025, pp. 589–622.

19. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

20. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

21. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

22. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

23. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

24. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

25. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

26. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

27. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

28. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

29. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

30. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

31. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

32. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

33. Смірнова Т.В., Гнатюк С.О., Бердибаєв Р.Ш., Сидоренко В.М., Жигаревич О.К., «Система корелювання подій та управління інцидентами кібербезпеки на об’єктах критичної інфраструктури». *Кібербезпека: освіта, наука, техніка*, №*3*(19), 2023, С. 176-196.

34. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *ІІ Міжнародна науково-практична Інтернет-конференція «Інновації та*

перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)» м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

35. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

36. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

37. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

38. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

39. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

40. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

41. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

42. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

43. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

44. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

45. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

46. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

47. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

48. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

49. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

50. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

51. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

52. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

53. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.