

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.492.2

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«__» _____ 2021 р.

Магістерська дисертація

на здобуття ступеня магістра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення комп'ютерних систем»**

зі спеціальності 121 «Інженерія програмного забезпечення»

**на тему: «Методи та програмне забезпечення для розпізнавання
шкідливих файлів»**

Виконав (-ла):

студент (-ка) II курсу, групи ІТ-303мп

Гаврилова Надія Леонідівна _____

Керівник:

Старший викладач кафедри ІПТ, к.ф.-м.н.,

Вечерковська Анастасія Сергіївна _____

Рецензент:

доцент кафедри ІСТ, к.т.н., доц.,

Ткач Михайло Мартинович _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___» _____ 2021р.

**ЗАВДАННЯ
на магістерську дисертацію студенту
Гавриловій Надії Леонідівні**

1. Тема дисертації «Методи та програмне забезпечення для розпізнавання шкідливих файлів», науковий керівник дисертації Вечерковська Анастасія Сергіївна, к.ф.-м.н., затверджені наказом по університету від «25» жовтня 2021 р. № 3575-с
2. Термін подання студентом дисертації «6» грудня 2021 р.
3. Об'єкт дослідження – шкідливе(зловмисне) програмне забезпечення
4. Предмет дослідження – алгоритми та методи виявлення шкідливого (зловмисного) програмного забезпечення та файлів
5. Перелік завдань, які потрібно розробити – Проаналізувати різновиди та принципи роботи вірусів; Проаналізувати способи протидії вірусам через антивірусні програми; Провести порівняльний аналіз для виявлення переваг та недоліків існуючих антивірусних програм; Зробити висновки про проведений аналіз, обрати напрямок руху для створення власного антивірусного додатку; Створення програмного продукту, антивірусного програмного додатку; Тестування та оформлення інструкції щодо використання створеного програмного продукту.
6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 1 плакат
7. Орієнтовний перелік публікацій: 2 публікації.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Дослідження предметної області	14.09.2021	
2	Постановка та формулювання задач	14.09.2021	
3	Аналіз технологічного стеку для реалізації	20.09.2021	
4	Аналіз підходів до архітектури	01.10.2021	
5	Моделювання програмного забезпечення	10.10.2021	
6	Реалізація програмного забезпечення	15.10.2021	
7	Виконання експериментальних досліджень	30.10.2021	
8	Оформлення пояснювальної записки	05.11.2021	
9	Подання дисертації на попередній захист	22.11.2021	
10	Подання дисертації на захист	06.12.2021	

Студент

Надія ГАВРИЛОВА

Науковий керівник

Анастасія ВСЬОКОВСЬКА

РЕФЕРАТ

Розмір пояснювальної записки – 97 аркушів, містить 30 ілюстрацій, 14 таблиць, 2 додатки.

Актуальність теми. У роботі розглянуто проблему в області розробки програмного забезпечення з об'єктом антивірусного програмного забезпечення, показано основні особливості існуючих методів та програм виявлення зловмисних файлів та їх подальшого видалення, їх переваги та недоліки. Виявлено потребу в розробці програмного додатку із зручним інтерфейсом для користувача та не менш ефективною функціональною частиною захисту персонального технічного засобу.

Мета дослідження. Основною метою є забезпечення виявлення шкідливих файлів на персональних комп'ютерах користувачів, що можуть пошкодити\зруйнувати роботу системи.

Об'єкт дослідження: антивірусне програмне забезпечення.

Предмет дослідження: алгоритми та методи виявлення шкідливого (зловмисного) програмного забезпечення та файлів.

Для реалізації поставленої мети **сформульовані наступні завдання:**

- 1) проаналізувати різновиди та принципи роботи вірусів;
- 2) проаналізувати способи протидії вірусам через антивірусні програми;
- 3) провести порівняльний аналіз для виявлення переваг та недоліків існуючих антивірусних програм;
- 4) зробити висновки про проведений аналіз, обрати напрямок руху для створення власного антивірусного додатку;
- 5) створення програмного продукту, антивірусного програмного додатку;
- 6) тестування та оформлення інструкції щодо використання створеного програмного продукту.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано удосконалено метод створення програмного забезпечення, що буде нести в собі простий у використанні інтерфейс, та не менш ефективний сканер для аналізу файлової системи, що знаходяться на персональному комп'ютері (виділення деякої папки/окремого файлу чи сукупності всієї інформації збереженої у даному ПК) та виявлення вірусного/шкідливого носія.

Практичне значення отриманих результатів полягає в тому, що створено нове програмне забезпечення, що дозволяє захистити персональний комп'ютер від вірусних файлів чи програмних засобів, а також включає в себе досить простий користувацький інтерфейс. Переваги даного програмного продукту в тому, що інтерфейс виглядає сучасно, а система не перевантажена під час роботи даного додатку.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на V всеукраїнській науково-практичній конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2021) – м. Київ.

Публікації. Наукові положення дисертації опубліковані в:

Гаврилова Н.Л. Алгоритм виявлення вірусів – стратегія розпізнавання. Інженерія програмного забезпечення і передові інформаційні технології (SoftTech 2021): матеріали наук.-практ. конф. (Київ, 22–26 листопада 2021р.) Київ, 2021. С. 34-36.

Гаврилова Н. Л. Алгоритми розпізнавання вірусів. Експериментально-теоретичні дослідження в сучасній науці: матеріали 4 міжнар. наук.-практ. конф. (Кишинів, 4-5 Листопада 2021р.), Кишинів, 2021. С. 286-289

Ключові слова: ВІРУСИ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ОПТИМІЗАЦІЯ.

ABSTRACT

Explanatory note size – 97 pages, contains 30 illustrations, 14 tables, 2 applications.

Topicality. Examines the problem of software development with the object of malicious software, shows the main features of existing methods and programs for detecting malicious files and their further removal, their advantages and disadvantages. The need for developing a software add-on with a user-friendly interface for the user and no less effective functional part of the personal technical device protection was revealed.

The aim of the study. The main target is to ensure the detection of malicious files on users' personal computers, which can damage / destroy the system.

Object of research: malicious software.

Subject of research: algorithms and methods of detecting malicious software and files.

To achieve this goal, the **following tasks** were formulated:

- 1) analyze the varieties of viruses;
- 2) analyze ways to counteract viruses through antivirus programs;
- 3) perform a comparative analysis to identify advantages and disadvantages of existing antivirus programs;
- 4) make conclusions about the analysis, set a course for creation of a proprietary anti-virus add-on;
- 5) creation of a software product, antivirus software add-on;
- 6) testing and execution of the instructions for the use of the created software product.

The scientific novelty of the results of the master's dissertation is to improve the method of creating a software that has a simple user interface, and equally effective scanner for analyzing the file system located on a personal computer (identifying a single folder/file or an aggregate of all information stored on this PC) and detecting virus/spyware carryover.

The practical value of the obtained results is in the fact that the new software is created, which allows to protect the personal computer from virus files or software tools, and also includes a very simple user interface. The advantage of this software is that the interface looks good and the system is not overloaded while this add-on is running.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the Fifth All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems and Management Technologies" (ISTU- 2021) - Kyiv.

Publications. The scientific provisions of the dissertation published in:

Havrylova N. L. Virus detection algorithm – recognition strategy. Software engineering and advanced information technologies (SoftTech 2021): proceedings of the scientific and practical conference (Kyiv, November 22-26, 2021) Kyiv, 2021. P. 34-36.

Havrylova N. L. Virus recognition algorithms. Experimental-theoretical research in modern science: proceedings of the IV International scientific-practical conference (Kishinev, November 4-5, 2021), Kishinev, 2021, P. 286-289.

Keywords: VIRUS, SOFTWARE, OPTIMIZATION.

ЗМІСТ

ВСТУП	10
1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	14
1.1 Комп'ютерні віруси.....	14
1.2 Антивірусні програми	15
1.3 Огляд існуючих антивірусних програм	18
1.4 Не комерційні антивірусні програми	19
1.4.1 OpenAntiVirus	20
1.4.2 ClamAV.....	21
1.4.3 Moon Secure Antivirus	22
1.4.4 ClamWin	22
1.4.5 EG Free Antivirus	23
1.5 Комерційні антивірусні програми	23
1.5.1 AntiVir.....	25
1.5.2 NOD32	26
1.5.3 Dr.Web	27
1.5.4 Norton AntiVirus.....	28
1.5.5 Avast.....	29
Висновок до розділу 1	30
2 ВИДИ ШКІДЛИВОГО РПОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА СПОСОБИ ЇХ ВИЯВЛЕННЯ ТА УСУНЕННЯ.....	31
2.1 Класифікація комп'ютерних вірусів.....	31
2.1.1 Файлові віруси	31
2.1.2 Завантажувальні віруси	32

2.1.3	Макровіруси.....	34
2.1.4	Поліморфні віруси.....	36
2.1.5	Троянець.....	37
2.1.6	Черв'яки.....	38
2.2	Класифікація антивірусних програм	38
2.2.1	Сканери	38
2.2.2	CRC–сканери	39
2.2.3	Монітори	40
2.3	Антивірус та алгоритми розпізнавання вірусів.....	42
2.3.1	Стратегія розпізнавання «Byte Patterns».....	45
2.3.2	Стратегія розпізнавання «Hash Sums»	46
2.3.3	Стратегія розпізнавання «Heuristics»	48
2.4	Формати підписів.....	48
2.5	Математична модель алгоритмів хешування	50
2.5.1	MD5 hash-based signatures	50
2.5.2	SHA hash-based signatures.....	53
2.5.3	PE section based hash signatures	60
	Висновок до розділу 2	62
3	РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ ШКІДЛИВИХ ФАЙЛІВ	63
3.1	Архітектура додатку.....	63
3.2	.NET Framework та Windows Forms.....	66
	Висновок до розділу 3	72
4	ОГЛЯД РОЗРОБЛЕНОГО ДОДАТКУ.....	73

4.1	Загальний огляд можливостей	73
4.2	Огляд інтерфейсу користувача розробленого додатку	76
	Висновок до розділу 4	87
5	РОЗРОБКА СТАРТАП ПРОЕКТУ	88
5.1	Опис ідеї проекту	88
5.2	Технологічний аудит ідея проекту	90
5.3	Аналіз ринкових можливостей запуску стартап-проекту	91
	Висновок до розділу 5	96
	ВИСНОВКИ	97
	СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	98
	ДОДАТКИ	99
	ДОДАТОК А	100
	ДОДАТОК Б	102
	ДОДАТОК В	111

ВСТУП

Проблема, що розглядається у даній роботі полягає в наступному: комп'ютери стали справжніми помічниками людства і без них вже не може обійтися жодна організація, що надає послуги і не тільки. Чимало інформації проходить у процесі обробки запитів комп'ютером від користувача, мільйони даних надходять та виходять у мережу щосекунди, а саме тому особливо загострилася проблема захисту інформації.

Велика частка людей, не замислюючись встановлює собі програми чи скачує файли з інтернету, що досить часто призводить до зараження техніки зловмисними програмами, званих іншими словами вірусами. Віруси можуть змінювати роботу техніки та поступово руйнувати систему в цілому, що в більшості випадків призводить до зменшення швидкодії, а в гірших випадках до втрати інформації або ж навіть і повного виходу з ладу персонального технічного засобу.

Незважаючи на прийняті в багатьох країнах закони про боротьбу з комп'ютерними злочинами і розробку спеціальних програмних засобів захисту від вірусів, кількість нових програмних вірусів постійно росте, адже на кожную дію існує протидія. Дана ситуація вимагає від користувача персонального комп'ютера мінімального розуміння про природу походження вірусів, способи (види) зараження вірусами та протидії ним. Для цього і зробили антивірусні програми, що забезпечують спокійне та безперебійне користування комп'ютером, з мінімальним ризиком втрати інформації чи виходу його з ладу.

У літературі досить наполегливо пропагується, що позбутися від шкідливих файлів можна лише за допомогою складних (а досить часто ще й дорогих) антивірусних програм, що начебто тільки під їх захистом користувач може буди впевненим в цілковитій безпеці свого персонального комп'ютера. Проте, це не зовсім правдиво – більше глибоке знайомство з деякими особливостями внутрішньої будови та способами впровадження

комп'ютерних вірусів допоможе вчасно їх виявити та знайти їх місцезнаходження, навіть якщо під рукою не виявиться тої самої складної антивірусної програми.

Більшість антивірусних програм вимагають платних підписок чи ключів, що не являється доступним для всіх людей світу, а отже існує велика потреба у безкоштовній, проте не менш якісній програмі для захисту від вірусів.

Людство все більше покладається на техніку (а саме довіряє комп'ютеру тони інформації про себе, родичів та інших людей зі свого кола спілкування), а отже дана інформація потребує захисту від рук зловмисників. **Актуальним** є створення програмних продуктів, які будуть займатись виявленням та видаленням вірусних файлів, без додаткового навантаження на систему, а також надає можливість самостійно поповнювати бібліотеку вірусного ПО користувачу.

Метою роботи є забезпечення виявлення шкідливих файлів на персональних комп'ютера користувачів, що можуть пошкодити\зруйнувати роботу системи.

Для успішного досягнення мети необхідно виконати наступні **завдання**:

- 1) проаналізувати різновиди та принципи роботи вірусів;
- 2) проаналізувати способи протидії вірусам через антивірусні програми;
- 3) провести порівняльний аналіз для виявлення переваг та недоліків існуючих антивірусних програм;
- 4) зробити висновки про проведений аналіз, обрати напрямок руху для створення власного антивірусного додатку;
- 5) створення програмного продукту, антивірусного програмного додатку;

б) тестування та оформлення інструкції щодо використання створеного програмного продукту.

Об'єктом дослідження є антивірусне програмне забезпечення.

Предметом дослідження є алгоритми та методи виявлення шкідливого (зловмисного) програмного забезпечення та файлів.

Під час дослідження були використані наступні **методи**:

- 1) методи статичного аналізу;
- 2) методи захисту від вірусів (метод контролю цілісності, метод сканування підозрілих команд, метод відстеження поведінки програм);
- 3) методи формальної верифікації.

Що вже зроблено для вирішення проблеми.

На сьогоднішній день найбільш актуальним є створення системи розпізнавання вірусів, що надає можливість звичайним користувачам захистити свій персональний комп'ютер без вірусів досить легко.

Вже не перший рік існують алгоритми хешування, так як SHA1, SHA2, MD5 та багато інших, які можна використати для шифрування чи розшифрування інформації. За допомогою даних методів існують програми, що носять назву антивірусних і мають змогу розпізнати «посилання» заховані у файлах, а саме виявити шкідливі файли чи додатки, що можуть негативно вплинути на роботу здатність системи в цілому.

Використовуючи PE («Portable Executable» - дослівно з англійської портативний виконуваний файл) секції файлу можна розпізнати, що знаходить всередині файлу та розрізнити між собою вірусні та доброякісні файли, додатки, тощо.

Що планується зробити у даній роботі. На основі алгоритмів виявлення вірусних файлів буде створено програмне забезпечення, що надає можливість виявити шкідливі файли на видалити їх із системи, не перешкоджаючи та не руйнуючи систему безпосередньо. Використовуючи алгоритм хешування, що вимагає перетворення вхідних даних в бітовий рядок певної довжини, можна

порівняти створений хеш із PE секцією файлу та базою вірусів, завдяки чому визначити (просканувати) шкідливість даного файлу. У разі збігу між двома хешами (з файлу для сканування та базою даних) – програмне забезпечення створене в ході даної роботи поверне користувачу повідомлення про наявність небезпечного файлу та запропонує його позбутись.

Наукова новизна полягає в наступному – удосконалено метод створення програмного забезпечення, що буде нести в собі простий у використанні інтерфейс, та не менш ефективний сканер для аналізу файлової системи, що знаходяться на персональному комп'ютері (виділення деякої папки/окремого файлу чи сукупності всієї інформації збереженої у даному ПК) та виявлення вірусного/шкідливого носія. Для цього порівнюючи хеш відсканованого файлу з базою даних вірусів, алгоритм обробки відокремить файли, що порушують цілісність системи, від тих що не несуть в собі ризику.

Практична цінність полягає в наступному – створено нове програмне забезпечення, що дозволяє захистити персональний комп'ютер від вірусних файлів чи програмних засобів, а також включає в себе досить простий користувацький інтерфейс. Переваги даного програмного продукту в тому, що інтерфейс виглядає сучасно, а система не перевантажена під час роботи даного додатку.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Комп'ютерні віруси

Для початку почнемо з термінології та визначень, після чого перейдемо до класифікації та більш детального розгляду кожного виду вірусу. *Комп'ютерний вірус* – це певний фрагмент коду, що скритий у програмному додатку чи файлі, який має можливість повного самостійного розвитку (копіювання та розповсюдження в середині системи), що призводить до певних збоїв у роботі пристрою (сповільнення роботи, несподівані перезавантаження, безконтрольне мерехтіння тощо). Проведемо аналогію на прикладі людського здоров'я: коли людина переохолола в осені, а через певний час відчуває себе не такою жвавою та з'являються перші ознаки хвороби (чихання, болі у горлі, температура тощо), а саме вірус проявляє себе і намагається зруйнувати організм, так само відбувається і у комп'ютерному світі. У даному прикладі ліками для технічного пристрою буде виступати антивірус (система розпізнавання та видалення небажаних файлів), проте про це буде написано далі, у даному пункті необхідно розглянути різновиди «шкідників».

Процес під час якого вірус створює свою копію та розповсюджує її в іншу програму можна назвати *зараженням*, додаток чи будь-який інший об'єкт системи, що тримає в собі вірус – *зараженим*.

Кіберзлочинці, що крадуть конфіденційні дані, швидко розвиваються, а разом із ними й зростає різноманіття зловмисних паразитів, готових підірвати цілісність роботи системи. У всьому світі існують мільйони вірусів, проте можна виділити деякі з поширених типів та класифікувати їх у наступному вигляді:

- 1) файлові віруси (а саме мається на увазі, зловмисний файл, що розповсюджується та заражає сусідні із ним);

- 2) завантажувальні віруси (іншими словами місце поширення даного типу знаходиться у фізичних носіях, таких як жорсткі диски чи USB);
- 3) макровіруси (найчастіше зустрічаються у документах, та зберігаються як їх частина);
- 4) поліморфні віруси (даний тип має можливість змінювати оболонку, тобто код, кожного разу, коли потрапляє у нове середовище – систему);
- 5) троян, або відомий також назвою «троянський кінь» – представляє собою схований вірус, тобто шкідливе програмне забезпечення сховано від оболонкою іншого програмного застосунку;
- 6) черв'яки («Worms» - у перекладі з англійської) – вид зловмисних програм, що не потребують носіїв, а тому можуть потрапити у систему через мережу інтернет та розмножувати без іншої програми, тобто вони самостійні і не мають прикриття як трояни чи файлові віруси.

Для легшого розуміння природи дії та протидії вірусного забезпечення необхідно розглянути більш детальну інформацію щодо кожного з вище наведених «шкідників».

1.2 Антивірусні програми

Антивірусні програми є найбільш ефективними в захисниках комп'ютерів від вірусів, що підстерігають на кожному шляху. Проте, як не прикро визнавати, у світі не існує антивірусних програм, що можуть гарантувати ідеальний захист від вірусів.

На жаль, таких систем не може існувати, через те, що на будь-який дієвий алгоритм знайдеться протидія, на кожен з існуючих антивірусів завжди можна створити контр атакуючий вірус, що зробить неможливим для даного антивірусу своєчасне розпізнавання та видалення збудника спокою системи. З іншого боку, даний процес працює в обидві сторони, що означає наступне: на будь-який з існуючих алгоритмів вірусного

програмного забезпечення можна створити антивірусний алгоритм. Існування абсолютного вірусу чи антивірусу було доведено розрахунково, за допомогою математичних підрахунків та теорії кінцевих автоматів, автором якої являється Фред Кoen.

Кожну програму можна оцінити за набором показників якості, тому нижче буде наведено перелік позицій, що надають змогу описати та охарактеризувати загальні характеристики антивірусів (список надано у в порядку зменшення важливості показника).

Надійність та зручність використання, а саме відсутність переривань у роботі антивірусу, а також технічних проблем, що призводять до втрати розуміння з боку користувача, що необхідно для відновлення працездатності програми. Даний критерій є одним із найбільш значущих, через те, що навіть абсолютний (ідеальний для виявлення вірусів) антивірус (якщо б він міг існувати) може виявитися непотрібним, у разі якщо він не виконає процес повного сканування і десь по середині зависне, без ніякої можливості відновити чи продовжити процес після перезавантаження, тощо.

У разі коли антивірус потребує додаткових пізнань для вирішення тих чи інших проблем, може бути проігнорований, а отже і подальша його робота стає під питання. Якщо ж антивірус буде вимагати додаткових налаштувань, та буде надто настирливо пропонувати якісь додаткові налаштування, то, скоріш за все, дана програма залишиться непотрібною.

Якість виявлення вірусів також грає велику роль, оскільки головне призначення антивірусів забезпечити безпеку – а отже користувач має отримати стабільну роботу операційної системи без заторможувань чи неочікуваних повідомлень. Великою перевагою стане відсутність помилкових спрацьовувань з боку захисних програм, а також лікування вже вражених файлових чи програмних об'єктів системи.

Доступність для різних операційних систем є наступним, проте не менш цікавим пунктом у списку переваг, через те що програма, яка не розрахована на захист іншої ОС, може лише частково використовувати функції, що були закладені розробників, або ж іноді і зовсім не відкриватись на іншій ОС. "Нерідні" (або ще можна назвати розроблені не для певного роду ОС) антивіруси можуть виявляються згубними для технічного засобу та призвести до порушення цілісності системи.

Невідмінною перевагою стане перевірка файлів у реальному часі, тобто сканування відбуватиметься на всіх змінах стану операційної системи (завантаження файлів, збереження змін у документах, створення чи копіювання інформації тощо). Миттєва, а отже і примусова перевірка мають створювати захисний бар'єр для комп'ютера та даних, що збережені на ньому.

Переходячи до якості по швидкості роботи, необхідно розуміти, що майже всі сканування системи в цілому займуть певний час, проте це все одно не має тривати годинами. Необхідно також зазначити, що повільна робота антивірусу зовсім не гарантує знаходження більшої кількості вірусів. У різних системах знаходження шкідливого програмного забезпечення використовуються і різні алгоритми пошуку та виявлення зловмисних файлів, а отже і час обробки може відрізнитись.

Наявність додаткових функцій та функцій займає чи не останнє місце у списку показників якості антивірусу, оскільки дуже часто ці функції жодним чином не впливають на рівень корисності антивірусу. Однак деякі з цих додаткових функцій мають змогу полегшити життя користувача.

Найпопулярніші та не менш ефективні антивірусні програми являють собою саме сканери вірусів, а слідом можна зазначити CRC-сканери. Досить часто можна зустріти, ці сканери об'єднані в одну універсальну програму, що значно збільшує її потужність та ефективність. Також можна зазначити досить популярні види блокаторів та імунізаторів.

1.3 Огляд існуючих антивірусних програм

Сучасна робота персонального пристрою залежить від глобальних інформаційних мереж та без регулярної та кваліфікованої антивірусної підтримки, операційна система може частково або повністю відмовити, а отже необхідно забезпечити певний захисний бар'єр. Даною оболонкою для захисту комп'ютеру може виступати антивірус, тобто система виявлення та подальшої обробки зловмисного програмного забезпечення. На жаль, не кожен рядовий користувач може собі дозволити оплатити комерційні проекти по захисту технічних пристроїв, а отже необхідно знайти відкритий та безкоштовний аналог корпоративного.

Багато продвинутих користувачів мають натхнення поліпшувати продукт, яким звикли користуватись, а отже проекти відкритого коду можуть мати велику популярність серед такого роду зацікавлених людей. В межах даного огляду також буде надано і порівняно характеристики масштабних проектів, що створювались великою групою людей, та мають велику популярність серед користувачів (особливо серед корпоративних клієнтів), адже вони займають певне місце на ринку, та несуть у собі досить обширні функціональні можливості.

Сучасні програми, що протидіють розповсюдженню вірусів є багатофункціональними продуктами, що поєднують в собі як запобігаючи засоби профілактики, так й способи лікування вірусів та збереження даних. Виходячи з вищесказаного, метою дослідження є, з'ясувати які ж аналоги існують на ринку в сьогоднішні дні, які з них покривають потреби користувачів. Об'єктом дослідження буде вважатись сучасне антивірусне програмне забезпечення. Предметом дослідження буде аналіз ринку сучасного антивірусного програмного забезпечення та прийняття рішення, про подальшу розробку власного антивірусного додатку, що буде конкурентно спроможним на сучасному ринку.

1.4 Не комерційні антивірусні програми

Не комерційних проектів не так багато у світі, проте деякі з них знайшли свою аудиторію зацікавлених, що не звертаючи увагу на певні недосконалості надають перевагу їм замість проектів розроблених корпораціями зі світовим ім'ям.

Для початку наведемо коротку характеристику у табличному вигляді з основною інформацією щодо рішень, що будуть детально описані нижче.

Таблиця 1.1 – Порівняльна характеристика антивірусних не комерційних програмних продуктів

Назва антивірусної програми/ Характеристика	Open Anti Virus	ClamAV	Moon Secure Antivirus	ClamWin	EG Free Antiviruse
Сканер вірусних файлів	+	+	+	+	+
Наявність інтерфейсу	—	—	+	+	+
Наявність сучасного інтерфейсу	—	—	+	—	+
Ведення історій дій (лог файл)	+	+	+	+	+
Перевірка файлів, під час завантаження з інтернету	+	+	—	—	—
Захист поштової скриньки	—	+	—	+	—

Продовження таблиці 1.1

Назва антивірусної програми/ Характеристика	Open Anti Virus	ClamAV	Moon Secure Antivirus	ClamWin	EG Free Antiviruse
Наявність карантинного відсіку	—	+	—	+	+
Підтримка декількох мов інтерфейсу	—	—	—	—	—
Наявність документації для користувача	+	+	—	—	—

1.4.1 OpenAntiVirus

OpenAntiVirus – представляє собою один із проектів з відкритим кодом, що покриває собою потребу сканування системи для виявлення вірусів, що носить назву «Virus Hammer», що в перекладі з англійської дослівно – «вірусний молоток», проте це не єдина можливість, що надається бажаючим використати дану продукцію.

Доступні наступні функції, що захищають цілісність системи та особисті дані користувача:

1) «Virus Hammer», «Scanner Daemon» та «Pattern finder» – представляють собою сканер, що необхідно запускати користувачам для виявлення та знешкодження шкідливого програмного забезпечення. Дані набори інструментів відповідають за забезпечення захисту для персональної техніки, шляхом сканування (перевірки) файлової системи;

2) «Squid-vscan» – використовується для сканування трафіку користувача в цілому та дозволяє побачити вірус на досить ранньому етапі. Squid – це проксі-сервер для кешування інформації, що надходить з мережі Інтернету за підтримки протоколів HTTP, HTTPS, FTP тощо.

Підсумок по даній антивірусній програмі: OpenAntiVirus платформа для людей, які дійсно зацікавлені та захоплені дослідженнями антивірусів, комп'ютерною та мережевою безпекою, а отже мають потенціал до вивчення вже не один рік існуючої бібліотеки команд та подальшого вдосконалення даного продукту.

Інтерфейс платформи досить не зручний для звичайного користувача (консольний ввід інформації), а отже і змоги використовувати його регулярно з комфортом не має. Офіційний сайт виглядає старомодно, а саме тому не привертає до себе великої аудиторії.

1.4.2 ClamAV

ClamAV – це ще один антивірус з відкритим кодом початкового проекту, що можна знайти у мережі Інтернет. Головне призначення, як і двох раніше розглянутих додатків – це виявлення вірусів, троянських (тобто зловмисних) програм та інших видів шкідливих загроз.

Переходячи до основних особливостей можна зазначити наступне – дане програмне забезпечення включає у собі багатопотоковий сканер та набір інструментів з використанням командного рядка для сканування файлів на вимогу користувача чи для автоматичного запуску. Інтерфейс відсутній, тільки командний рядок для виконання операцій.

Підсумок по даній антивірусній програмі: досить широкий функціонал та база, що виявляє вірусні загрози, зловмисне програмне забезпечення та допомагає у виявленні потенційних неприємностей у мережі інтернет. Програма досить компактна та може бути використана у різних операційних системах, що надає можливість покрити великий спектр користувачів.

Недоліком даної програми буде те, що не існує користувацького інтерфейсу, а отже даний проект, згідно із початковим призначенням, вимагає від користувача глибших пізнань. Даний проект може бути використаний як

«база» для інших, а отже все бібліотеки та інформаційні бази, можуть бути легко завантажені з інтернет-ресурсів.

1.4.3 Moon Secure Antivirus

«**Moon Secure Antivirus**» – представляє собою проект з відкритим кодом, що призначений для виявлення вірусів, троянських програмного забезпечення та будь-якого шпигунського коду.

У даному додатку запропоновано деякі види стандартної роботи, такі як механізми сканування (виявлення) потенційної небезпеки з подальшим знешкодженням (видаленням), функція виявлення загроз у режимі реального часу.

Даний проект включає в себе наступні недоліки: доступний лише для Windows, велике використання ресурсів персональної техніки та необхідність оновлювати базу вірусів самостійно, що призводить до того, коли з метою більшої захищеності необхідно витратити час на додаткове налаштування.

1.4.4 ClamWin

ClamWin – це безкоштовний антивірусний інструмент, що придатний лише для однієї операційної системи, а саме Windows. Код є відкритим для будь-кого. Антивірус забезпечує досить гарні результати по виявленню вірусів та зловмисник програмник застосунків.

Доступний до налаштування та використання – планувальник запуску сканування, а саме можна обрати зручний день та час для користувача, коли відбудеться початок перевірки цілісності системи.

Програмний продукт надає перевагу доступу до додаткового компоненту тісно пов'язаного із поштовою скринькою Microsoft Outlook, що покриває своєчасне автоматичне видалення інфікованих «шкідниками» додатків у листах.

Про недоліки можна сказати наступне: інтерфейс досить застарілий, а у віці розвитку технологій це є досить суттєвим для більшості користувачів.

Даний антивірус може бути запущений та використаний лише під однією операційною системою, що не є гарним показником.

1.4.5 EG Free Antivirus

«EG Free Antiviruse» представляє собою безкоштовне антивірусне програмне забезпечення з відкритим вихідним кодом лише для Windows. Основними функціями цього антивірусного програмного забезпечення можна привести:

Програма безперервно сканує запущений фоновий процес для швидкого виявлення активних саме у цей момент часу вірусів.

Швидкісне сканування – інструмент часткового (швидкісного) сканування системи, у тих частинах, які найчастіше приховують зловмисні програми.

Повноцінне сканування – метод, що покриває всю систему, який знаходить та знешкоджує активні та неактивні віруси.

Карантинний відсік – використовується у разі, коли повністю видалити вражений файл не вдалось, він надходить у спеціальну зону пам'яті, щоб надалі ці віруси були неактивними і не могли погіршити роботу системи.

1.5 Комерційні антивірусні програми

У комерційних проєктів достатньо велика кількість, проте нижче будуть розглянуті лише деякі з них, що займають призові місця серед попиту у більшості користувачів.

Для початку наведемо коротку характеристику у табличному вигляді з основною інформацією щодо рішень, що будуть детально описані нижче.

Таблиця 1.2 – Порівняльна характеристика антивірусних комерційних програмних продуктів

Назва антивірусної програми/ Характеристика	AntiVir	NOD32	Dr. Web	Norton AntiVirus	Avast
Сканер вірусних файлів з можливістю їх подальшого видалення	+	+	+	+	+
Наявність сучасного інтерфейсу	+	+	+	+	+
Ведення історій дій (лог файл)	+	+	+	+	+
Перевірка файлів, під час завантаження з інтернету	+	+	—	—	—
Захист поштової скриньки	+	+	+	+	+
Підтримка декількох мов інтерфейсу	—	+	+	—	+
Наявність документації для користувача	+	+	+	+	+
Наявність функцій оптимізації ПК	—	—	—	+	+
Безкоштовна версія, що необмежена у часі	—	+	+	+	+
Персональні дані користувача захищені	+	+	+	+	—
Наявна корпоративна ліцензія, що розширює можливості додатку	+	+	+	+	+

1.5.1 AntiVir

AntiVir – це серія антивірусних програмних застосунків, які випускаються досить відомою корпорацією Avira. Програмні продукти серії «AntiVir Personal» представляють собою антивірусний програмний продукт (здебільшого безкоштовний), для персонального використання. Продукт являє собою програму-монітор для контролю операційної системи та програму-сканер, що робить перевірку цілісності системи.

AntiVir Premium – це платна версія того ж самого антивіруса, що має включає в себе ряд переваг над його безкоштовною версією, найбільш значні з яких:

- крім вірусів, виявляє рекламні та шпигунські програми;
- наявна додаткова можливість з перевірки поштової скриньки за протоколами POP3 і SMTP.

«AntiVir Premium Security Suite» – ще одна програма серії, що відрізняється від попередньо зазначених тим, що були додані Firewall (тобто захист в інтернеті) та анти-спам (захист від небажаного спаму).

AntiVir Workstation – призначений для захисту робочих станцій при використанні їх в бізнесі. Він має всі можливості PersonalEdition Premium і крім цього захищає комп'ютери в мережі.

Підсумок по даній антивірусній програмі: безкоштовна версія наявна, проте має дуже обмежені можливості, а отже для отримання більш якісного захисту – необхідно оплачувати Premium версію, на що не кожен користувач погодиться, а отже програма не є цікавою для звичайної людини, що користується ПК у домашніх цілях.

З точки зору корпоративного використання, а отже і використання преміум версії – програма несе у собі досить обширний набір функцій починаючи з захисту у режимі реального часу та закінчуючи захистом в

інтернеті, а отже для користувачів корпоративної мережі потенційних ризиків пошкодити систему персонального технічного засобу знижується.

1.5.2 NOD32

NOD32 – це досить обширний функціональний пакет антивірусного програмного забезпечення, що підтримується та покращується із року в рік словацькою фірмою під назвою Eset. NOD32 являє собою комплексне рішення по захисту операційної системи від небажаних втручань шляхом постійного контролю та аналізу кожного отриманого та відправленого байту інформації. Даний антивірусний додаток надає користувачу захист від вірусних файлів та програм (чи кодів програм схованих під маскою «доброякісних» програм.

Особливість та перевага даного продукту, що очевидна при оновленні додатку, а точніше його баз у тому, що фоново йде процес ряд серверів-дзеркал, які у свою чергу приводять до зниження навантаження на інтернет-канал, а отже і не заважають звичайній роботі користувача.

NOD32 включає в себе декілька видів сканерів вірусів, таких як: резидентний сканер; ручний сканер, який запускається для перевірки окремих файлів; IMON модуль (повністю англійською – «Internet MONitor») – надає можливість проконтролювати Інтернет-трафік (HTTP) та електронну пошту; DMON (повністю англійською – «Document MONitor»), що використовує досить відомий інтерфейс від компанії Microsoft, а саме API для безпосередньої точної перевірки документів.

Модуль, що надає захист пошти ESET від небажаних листів, виконує функцію фільтрування, а отже займається підвищенням захисту та безпечності відкриття того чи іншого листа відправленого по електронній пошті.

Підсумок по даній антивірусній програмі: безкоштовна версія наявна, проте вона буде явно піратською та підбір ключа стає все важчим. Версія з ліцензією надає великий спектр можливостей, проте не всі користувачі будуть

готові до використання «білого» програмного забезпечення, а отже будуть користуватись піратською версією.

NOD32 має схильність оманливого сприйняття файлу за вірус і тому захист доводиться вимикати, щоб отримати бажаний файл та мати змогу його відкрити. Також NOD32 може розпізнати деякі додатки, як віруси, хоча користувач встановив даний додаток досить давно і не оновлював його з давніх пір, і тільки з появою оновлення антивірусу – додаток розпізнається як вірус.

Здебільшого спектр можливостей досить широкий і антивірус навіть «перевиконує» поставлені цілі, намагаючись видалити «зайві» на його думку файли, проте це – недолік у межах корпоративного використання надає більший захист, а отже і недоліком для деякого кола користувачів не буде.

1.5.3 Dr.Web

Dr. Web – досить відома назва пакетного антивірусного програмного забезпечення, що носить в собі не менш відому мету – захист персонального комп'ютеру від різноманітних вірусних «шкідників», серед яких є черв'яки та файлові віруси, троянські програми та поліморфні віруси, макровіруси та схованих у документах «зловмисників», скрипт-віруси та шпигунське програмне забезпечення, клавіатурні шпигуни та програми-крадії паролів чи інших персональних даних, програми-люки та програми-жартівники, шкідливі скрипти (тобто код) та враженні файли. Дана програмна лінійка може надати захист й від досить набридливого в мережі спаму чи повідомлень, таких як фармінг чи фішинг повідомлень.

Досить великою перевагою антивірусу Dr. Web можна назвати наступне – це можливість установки на раніше вже заражений технічний засіб. Під час процесу установки виконується сканування не тільки пам'яті, а й файлів автозавантаження (слід зауважити, що безпосередньо перед скануванням

даний застосунок проводить оновлення вірусної бази, що буде досить актуальним при пошуку вірусів нових поколінь).

Розробники, що працювали над створенням такого антивірусного продукту були змушені відмовилися від «стандартного» процесу випуску оновлень вірусних баз по заздалегідь встановленому графіку, оскільки вірусні напливи точно не ведуться згідно графіку. Знаходження та подальше лікування складних поліморфних чи шифрованих вірусів можливе в даному програмному засобі

Кросплатформеність – використовується єдина вірусна база і єдине ядро антивірусного сканера, а отже даний додаток покриває досить широку аудиторію користувачів.

Підсумок по даній антивірусній програмі: досить широкий функціонал та база, що виявляє вірусні загрози. Програма досить компактна та кросплатформна, що надає можливість покрити великий спектр користувачів. Недоліком даної програми буде те, що оновлення не виходять часто, а отже ніяких додаткових можливостей чи оптимізацій алгоритмів роботи чекати не варто, вони якщо і будуть з'являться, то досить не регулярно.

1.5.4 Norton AntiVirus

Norton AntiVirus – це досить відомий і популярний антивірус. Відсоток розпізнавання вірусів дуже високий. У програмі використовується механізм, який дозволяє розпізнавати нові невідомі віруси.

Програма по боротьбі з вірусами видає детальну інформацію про виявлений вірус, а також надає вам можливість вибору: видаляти вірус або в автоматичному режимі, або більш обачно, за допомогою покрокової процедури, яка дозволяє побачити кожну з виконуваних в процесі видалення дій.

Недоліком даної програми є складність змін параметрів роботи програми, хоча базові (попередньо встановлені розробниками) налаштування

змінювати, практично не потрібно, проте для більшої захищеності краще витратити час на до налаштування.

1.5.5 Avast

Avast – це сімейство антивірусних додатків, що можуть бути використані у операційних системах від компанії Microsoft, а саме Windows чи у не менш відомих системах Linux. Даний продукт включає в себе дві основні версії для будь-яких користувачів: безкоштовна (проте лише для некомерційного використання) та платна (для будь-якого роду використання).

Можливості програми досить широкі, серед них – резидентний сканер вірусів; виконання перевірки системи на віруси під час завантаження заставки; перевірка технічного засобу на наявність вірусів під час запуску (процес відбувається до початку завантаження операційної системи); евристичний аналіз; блокування шкідливих само виконуваних кодів; автоматичне оновлення баз вірусів та самої програми; інтерфейс користувача включає в себе декілька мов.

Програма включає в себе досить широкий спектр можливостей, проте має як безкоштовну, так і платну версії, очевидно що, безкоштовний варіант досить урізаний у функціональних можливостях.

Про недоліки можна сказати наступне: великі потоки персональних даних надходять на сервери компанії, що створила даний антивірус, а отже є ризик розкриття персональних даних користувача, так Forbes з'ясував, що антивірус Avast експлуатував дані близько 400 мільйонів користувачів для власного збагачення з 2013 року. Генеральний директор компанії, що створила додаток, підтвердив продаж досить обширних масивів даних обличчям, що не працювали в компанії, проте повідомив, що інформація була анонімною тобто до завантаження на сервери вона була позбавлена персональної складової.

Висновок до розділу 1

Не беручи до уваги велику поширеність програм для боротьби з вірусами, вони продовжують «розмножуватись» та зростати у складності виявлення. Щоб протистояти ним, буде стояти потреба у створенні універсальних та від цього не менш якісних антивірусних програмних застосунків, які будуть включати в себе всі позитивні якості своїх попередників. Як не прикро визнавати, проте на даний момент немає такої антивірусної програми, яка б надавала захист від усіх різновидностей вірусів на повні 100%, проте можна максимально наблизитись до цього показника, а отже можна створити програму, що буде гарантувати максимально можливий рівень захисту. Після аналізу ринку аналогів антивірусів, можна зробити висновок, що існуючі програми створені людьми з натхненням роботи щось відкрито, а тому проектів з відкритим кодом досить багато на ринку, проте не у всіх доведена якість та ефективність, а деякі з них не мають сучасного (чи зовсім ніякого) інтерфейсу. Тому на далі у роботі потрібно розробити максимально зручний для звичайного користувача інтерфейс з використанням найновіших баз з інформацією про відомі види вірусів. Порівнювати комерційні проект з роботою, що робить одна людина не є раціональним, а отже серед існуючих на ринку програм – комерційних досить багато та з їх ефективністю можна легко погодитись, проте у основі даної роботи лежить створення програмного застосунку, що зможе конкурувати на ринку некомерційних проектів, а також не буде гіршим за них.

Згідно інформації наданої у розділі 1, можна побачити, що потреба у захисті від вірусів досить висока і більшість користувачів потребують дешевого, а точніше відкритого та безкоштовного, антивірусу, який буде захищати комп'ютер від шкідливого ПЗ та не буде видаляти ті чи інші файли, що не несли в собі загрози. Також антивірус має надавати змогу поповнювати базу регулярно, оскільки світ розвивається, а отже завжди необхідно мати під рукою актуальний набір (список) вірусних файлів чи кодів.

2 ВИДИ ШКІДЛИВОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА СПОСОБИ ЇХ ВИЯВЛЕННЯ ТА УСУНЕННЯ

2.1 Класифікація комп'ютерних вірусів

2.1.1 Файлові віруси

Вірус, що являє собою розповсюдження зараження через файли – це тип шкідливого програмного забезпечення, що змінює виконуваний системою файли (документи, фото, програми та інше) з метою надання невірної втрати даних чи погіршення працездатності того чи іншого додатку, документу, тощо.

Інфікований файлом вірус перезаписує частину чи весь повністю код, або ж вставляє частку зараженого коду у виконуваний файл. До даної групи належать віруси, які при своєму розмноженні можуть вражати будь-яку операційну систему, а також можуть «подорожувати» між різними технічними засобами.

Нерідко даний вид називають *паразитичним*, і ось чому так – в процесі роботи комп'ютер бачить вірус як частину програми, яку користувач намагалися запустити, а тому без зайвих зволікань надає йому ті ж права, що і оригінальній програмі. Саме надані системою права дозволяють вірусу самостійно створювати свої ж копії, встановлюватись у пам'яті або вносити зміни без зайвих заборон з боку ОС.

В роботі файлового вірусу можна описати декілька головних фаз дій, що виглядатимуть наступним чином:

- 1) фаза спокою, або іншими словами бездіяльності. На даному етапі «шкідник» перебуває у пасивному стані, ховаючись від операційної системи;
- 2) фаза розмноження (розповсюдження), що настає під час першого запуску ураженого програмного застосунку в якому і поселився вірус. У даній

фазі відбувається активація коду, що створює більше копій коду, що має в собі вірус для його подальшого розвитку;

3) тригерна фаза (іншими словами фаза спрацювання) – це процес переходу з другого етапу до четвертого, а саме до активних дій;

4) фаза дій, остання з усіх та передбачена для того, щоб вірус виконував своє призначення, а саме зловмисні дії, для яких він був розроблений. Такими діями можуть бути: впливаючі вікна з різним текстом чи картинками, пошкодження чи знищення інформації, видалення важливих для системи файлі, тощо.

2.1.2 Завантажувальні віруси

Завантажувальні віруси, або ж інакше їх називають вірусами завантажувального сектора – це різновид вірусу, що інфікує сектор завантаження гнучких дискових носіїв або ж основний сектор для запису у жорстких дисках. Заражений код, що потрапив на зовнішній носії інформації раніше, первинно запускається під час налаштування (чи оновлення) системи з інфікованого диска, та у процесі використання він заражає інші диски персонального комп'ютера.

Принцип дії даної категорії шкідливого забезпечення заснований на алгоритмах завантаження операційної системи комп'ютера, що викликається із MBR («Master Boot Record» - з англійської перекладається як основний завантажувальний запис). MBR являє собою невелику програму, що виконується при кожному запуску комп'ютера. Вона контролює послідовність завантаження та визначає, з якого саме розділу необхідно загрузити комп'ютер. MBR, як правило, знаходиться на першому із багаточисельних секторів жорсткого диска.

Видалення вірусів завантажувального сектора зазвичай дуже складне, оскільки більшість антивірусних програм не можуть очистити MBR під час

роботи ОС, а саме тому у переважній більшості випадків для видалення даного виду «шкідників» потрібні спеціальні завантажувальні антивірусні диски.

Вірус завантажувального сектора може спричинити різні проблеми пов'язані із пошуком чи відображенням даних, збережених безпосередньо у системі технічного засобу. У деяких випадках інформація може зникати із цілих розділів. Нерідко трапляються випадки, коли комп'ютер раптово стає нестабільним, тобто система працює з певними уповільненнями чи взагалі часто з'являються збої у повсякденному використанні.

Досить поширена проблематика вражених пристроїв в тому, що не вдається запустити або знайти жорсткий диск операційною системою, у таких випадках повідомлення про помилки (наприклад "Недійсний системний диск") з'являються постійно.

Алгоритм роботи завантажувального вірусу будуть описані далі. Більшість з вірусів даної категорії резидентні. Основним способом впровадження у пам'ять комп'ютера є завантаження з інфікованого диска чи флешки. Системний завантажувач (MBR) зчитує вміст першого сектора основного диска системи, з якого відбувається налаштування на запуск, поміщає зчитану інформацію в пам'ять і передає на неї (тобто на вірус) управління. Після успішної підміни системного виконання на «шкідника» починають виконуватися інструкції, серед яких можна перелічити наступні:

- зменшення обсягу вільної пам'яті технічного засобу, копіювання в місце, що звільнилося негативного коду і зчитування з диска продовження вірусного коду (якщо воно є);
- перехоплення необхідних векторів переривань системи, зчитування в пам'яті оригінального boot-сектора, і подальша передача контролю вірусу.

Наступними кроками цей різновид згубного коду поводитьсь так само, як резидентний файловий, а конкретніше – починає перехоплення звернень операційної системи до дисків та інфікує їх, а залежно від умов оригінального вірусного коду відбуваються деструктивні зміни чи виклик звукових та/або відео ефектів.

Існують також ще й нерезидентні віруси boot-сектора. При першому завантаженні вони уражують MBR жорсткого диску після чого передають управління оригінальному завантажувачу і на роботу комп'ютера більше не впливають.

2.1.3 Макровіруси

Макровіруси – представляють собою особливий вид вірусів, що відносяться до спектру програм написаних макромовою (програмна мова, що вбудована у системи обробки даних, такі як текстові редактори чи електронні таблиці) та тісно пов'язані із документами. Досить популярними та широко поширеними є макровіруси для таких програм від компанії Microsoft, як Word, Excel та Office.

Макровіруси часто можна зустріти вже вбудованими у документи чи не менш рідко вставляються як шкідливий код у програми для обробки текстів різного виду (незалежно від мови, якою вони написані). Також дана категорія руйнівного програмного забезпечення може надходити із документів, що були додані до електронних листів чи були завантажені за допомогою неперевіраних URL-адрес. Даний «шкідник» не проявляє себе до моменту первинного зараженого макросу, після чого він виконує ряд раніше прописаних команд.

Існують деякі тонкості для цього виду вірусів, оскільки є певні обмеження, що накладаються на конкретну систему, а саме для розповсюдження вірусу необхідна наявність вбудованої макромови, що перекриває наступні можливості:

- налаштований зв'язок між програмою (з використанням макромови) та деякого документа (іншими словами файлу з збереженою на ньому інформацією);
- повинно бути доступним перенесення (дублювання) макропрограми з одного файлу в інший;
- у системі повинні існувати самостійні та/або шаблонні макроси, що відповідають за підтримку управління макропрограми без допоміжного залучення до процесу з боку користувача.

Алгоритм роботи макровірусу заснованого на макросі Word розглянуто нижче.

Першим кроком у більшості випадків, при запуску документу Word відбувається перенесення з області локального коду до глобального деяких макросів документа, в ході чого задіяні команди дублювання при втручанні редактора макросів безпосередньо. Вірус, що був схований у макросі, задіює редактор та створює новий макрос, до якого додає власний свій код, після чого зберігає новостворений макрос в документі.

Наступним кроком Після закриття документу Word, відбувається автоматичне збереження глобальних макросів у DOT-файл (часто він носить ім'я NORMAL.DOT). Саме таким чином, під час безпосередньо наступного запуску Word-вірус підключиться (почне салю роботу) в той момент, як тільки WinWord підвантажить глобальні макроси.

Надалі вірус переназначає один або більше стандартних (наданих у Word) макросів та починає процес перехоплення команди по роботі із файлами. При кожному запуску даних команд вірус вражає файл, до якого відбувається виклик. Для того, щоб «шкідник» міг це зробити, відбувається конвертація (тобто зміна) файлу до формату Template та запис в файл своїх макросів, включаючи навіть автомакроси.

Саме таким чином, вірус, що перехоплює макрос «Зберегти файл як...», може почати зараження кожного DOS-файлу, що зберігається за допомогою перехопленого ним макросу. У разі, якщо перехоплений макрос «Відкриття файлу», то вірус має змогу записатись у будь-який файл у момент його зчитування з диску.

Алгоритм роботи макровірусу заснованого на макросі Excel дуже схожий до Word, проте головна відмінність полягає в тому, що при командах копіювання макросів збереження робляться не у NORMAL.DOT, а до файлів у STARTUP-каталозі самого Excel.

2.1.4 Поліморфні віруси

Поліморфні віруси являють собою складні інфіковані файли, що мають змогу створювати модифіковані версії самих себе, для уникнення виявлення, проте зберігають всі основні процедури після кожної інфекції. Для зміни свого фізичного складу кожного із файлів під час зараження, вони шифрують свої коди та щоразу вилучають різні ключі шифрування.

Як правило, поліморфні віруси покладаються на механізми мутацій (внесення змін), щоб оновлювати свої процедури дешифрування кожного разу, як тільки відбувається зараження технічного засобу. Саме через можливість постійних переінакшень всі традиційні рішення безпеки з деякою складністю можуть їх відловити, адже «шкідники» не використовують статичний код. Використання непростих механізмів апгрейду, що генерують велика кількість процедур дешифрування, ускладнює їх виявлення.

Поліморфні віруси досить часто поширюються через спам у електронній пошті, заражені сайти чи із використанням інших неперевіраних програм.

2.1.5 Троянець

Троянець чи «троянський кінь» – представляє собою приховане зловмисне програмне забезпечення, що на відміну від розглянутих раніше вірусів, не може запуститись самостійно, адже його код для початкового запуску вимагає відкриття оригінальної програми, так званої оболонки, що покриває шкідливий код.

Принцип дії даного виду зловмисного забезпечення можна легко зрозуміти із його назви, а саме спочатку троянця ховають у добросовісному застосунку чи файлі, щоб обдурити користувача. Після первинного запуску, трояни можуть змінюватись та «подорожувати» у файловій системі персонального комп'ютера та за допомогою мережі передавати особисті данні, що являє власне крадіжку інформації без згоди самого користувача.

Наведемо інформацію про деякі з підвидів троянців, такі як:

1) троян задньої двері («Backdoor» - дослівно з англійської задні двері) – з назви можна здогадатись, що даний підтип створює канал зв'язку за допомогою якою зловмисник отримує доступ до персонального технічного засобу та має повний контроль без згоди користувача. Після отримання даного доступу, можуть бути викрадені персональні дані чи підвантажено ще більше вірусного забезпечення;

2) троян-завантажувач – головною ціллю являється вже раніше уражений комп'ютер. Він встановлює, чи завантажує, все більше нових версій шкідливих програм;

3) ігровий троян, або іншими словами, той що живе у іграх – цвілевою аудиторією якого представляють собою геймери. Даний троянець краде інформацію про обліковий запис, що згодом може бути змінений від керівництвом зловмисника;

4) троянський банкір – це підвид, що націлений на будь-які фінансові дані користувача, найчастіше на банківські рахунки. Він покликаний викрасти

інформацію облікового запису, що не рідко включає дані банківської справи, кредитної картки чи оплати рахунків.

2.1.6 Черв'яки

Комп'ютерні хробаки дуже схожі на файлові віруси тим, що вони мають змогу до самостійного розповсюдження, тобто вони створюють функціональні копії самих себе і згодом заражають все більше операційну систему. Черв'як потрапляє у внутрішню систему технічного засобу через будь-яку слабкість у системі, після чого за допомогою функцій транспортування даних у системі, розвивається та поширюється без сторонньої допомоги. Деякі з даних видів «шкідників» використовують технології шифрування та очистки файлів, щоб завдати шкоди своїм цілям.

Більшість з відомих людству комп'ютерних черв'яків поширюються одним із наступних способів:

- файли, що надіслані як вкладення у листі електронної пошти;
- через посилання на веб-сайт, що надіслані через електронну пошту чи інші канали обміну повідомленнями між користувачами;
- деякі черв'яки можуть розповсюджуватись у вигляді мережевих пакетів, що проникають у пам'ять комп'ютера, після чого активується зловмисний код.

2.2 Класифікація антивірусних програм

2.2.1 Сканери

Описуючи принцип дії антивірусних програм типу сканеру, можна сказати, що основа лежить у особливостях для перевірки файлів та секторів, а також пам'яті системи, базуючись на пошуку всередині них вже відомих (тобто розповсюджених по світу, а отже і доданих до бази програми) та нових (тобто, раніше не існуючих чи не розповсюджених) вірусів. Для алгоритму знаходження вірусів програма користується масками. Під

маскою розуміють наступне – це послідовність коду, що являє собою особливий (специфічний) набір інформації для заданого вірусного файлу, програмного забезпечення, тощо. У разі, коли сам вірус не може бути представлений у вигляді маски чи довжина (кількість символів) маски не налічує необхідної довжини, то застосовуються й деякі інші методи. За приклад використання іншого підходу, можна вважати алгоритмічну мову, що у свою чергу виділяє більшість можливих варіантів коду, що безперечно можуть проявитись при ураженні системи від «шкідника», а саме вірус.

У сканерах також має популярність алгоритми евристичного сканування, що включають у себе аналіз деякої послідовності команд в заданому об'єкті, що підлягає перевірці, та набір деякої статистики на основі якої відбудеться прийняття рішення для кожного об'єкта окремо.

Повертаючись до переваг сканерів, можна однозначно привести універсальність, проте не слід забувати і про недоліки, які криються у розмірах антивірусних баз (що зростають та поповнюються все новими вірусами ледь не щодня), які програми мають досить часто оновлювати. Досить часто спостерігається відносно невелика швидкість пошуку вірусів, адже чим більша вибірка даних, тим більше триває пошук.

2.2.2 CRC–сканери

CRC–сканери несуть в собі наступний принцип дії, що працює від способу підрахунку контрольних сум (CRC–сум) для існуючих на жорсткому диску файлів чи системних секторів. Дані CRC–суми, після обчислень, заносяться до бази даних антивірусної програми, а разом із цим й інформація про довжину файлу, що було підраховано, дати останніх змін, тощо. Під час наступного ввімкнення CRC–сканери порівнюють дані, що збережені в базі даних, з щойно перерахованими показниками.

У разі коли інформація про будь-який файл у системі технічного засобу, записана раніше до бази, відрізняється від реальних значень, CRC–сканери відправляють сигнал назад та інформують про те, що обстежуваний файл був змінений або заражений вірусом.

CRC–сканери, які підтримують "антістелс"–алгоритми, являють собою досить потужну зброю проти вірусів різних видів, велика частка яких виявляється через короткий проміжок часу, після їх появи на пристрої. У даному виді антивірусів існує й невеликий недолік, що впливає на їх ефективність, оскільки CRC–сканери не надають змоги впіймати вірус у реальному часі (тобто «тут і зараз»), а видають результат через невеликий проміжок часу, то є ризик, що вірус вже почав змінювати роботу по комп'ютеру, а отже шкода вже завдана системі. Ще одним негативною відзнакою є те, що даний тип сканерів не має можливості розпізнати вірус в нових файлах, що не були відскановані раніше, оскільки інформація про них ще відсутня у базі.

Також слід зазначити, що з моментом розвитку шкідливого програмного забезпечення, даний недолік робить систему персонального комп'ютера більш вразливою, адже певні віруси здатні «стрибати» між новостворюваними файлами, а отже мають змогу сховатись від CRC–сканерів.

2.2.3 Монітори

Антивірусні монітори – це резидентні програми, що мають змогу перехопити вірусно-небезпечні ситуації та проінформувати користувача про погрозу, що насувається на його персональний комп'ютер. До таких погроз відносять: запити на відкриття для запису у файли, збереження в завантажувальні сектори чи MBR жорстких дисків, тобто всі запити, що більш характерні для «шкідників» операційної системи в моменти їхнього розповсюдження (дублювання).

До переваг даних моніторів можна сміливо включити здатність до виявлення та блокувати вірусів на дуже ранній етапах їх розвитку та розповсюдження. Як і у всіх інших видів, тут існують і недоліки, до яких можна віднести існування певних шляхів для обминання захисту від програм-монітора та великої кількості не точних опрацювань тих чи інших файлів.

Можна також вказати, один із напрямків антивірусних засобів для програм-моніторів, що самі по собі виконані у формі апаратних компонентів технічного засобу. Проте, такий напрямок не являє собою ідеальний спосіб захисту, адже його теж можна обійти. На жаль, навести можна ще один відносно негативний момент пов'язаний з проблемами у сумісності з різними стандартними налаштуваннями персональних комп'ютерів та вимушеної складності при встановленні даної апаратної частини.

Імунізатори, що представляють собою наступний підтип соніторних програм, та в свою чергу поділяються на дві категорії: імунізатори (мають змогу повідомити про зараження вірусом) та імунізатори (надають можливість блокувати зараження).

Перша категорія записується до кінця файлів та при кожному новому запуску файлу починають перевірку на зміну даних. Недоліком можна окреслити лише один, проте він дуже суттєвий і діє не на користь імунізатора – це відсутність відокремити зараження "стелс"-вірусом.

Друга категорія, а саме імунізаторів, створює захисний бар'єр для системи від шкоди вірусів певного виду. Файли, що зберігаються на жорстких дисках змінюються (оновлюються) таким чином, що навіть новий вірус сприймає їх за вже заражені, а отже навіть не намагається підмінити реальні дані своїм кодом.

Перевагою даного антивірусу являється те, що для захисту від резидентного «шкідника» в пам'ять пристрою зберігається програма, що

підміняє копію вірусу, та у процесі запуску зловмисник натикається на свою ж копію, а отже вважає, що система несе в собі зараження.

Не зважаючи на велику перевагу, такий тип імунізаторів не буде універсальним, через те, що не можна проімунізувати абсолютно усі файли від всіх відомих та невідомих вірусів.

2.3 Антивірус та алгоритми розпізнавання вірусів

Зайве повторювати щоразу, що антивірусу завжди доведеться стояти на варті цілісності та захисту операційної системи від зовнішніх впливів та загроз із боку «шкідників», а у ході аналізу того чи іншого файлу, ще й порівнювати аналізований файл із бібліотекою вже відомих людству вірусних програм чи кодів. За ті частки секунди (іноді і хвилини), коли користувач очікує запуску програми, антивірус виконує досить трудомістку роботу порівняння послідовностей коду. Звідси і виникають скарги на те, що використання антивірусного програмного забезпечення може сильно вповільнити роботу персонального технічного засобу. Звідси власне і зароджується інша безперервна боротьба розробників із створення антивірусного забезпечення з мінімальним впливом на системні ресурси комп'ютера.

Виявлення підписів (англійською «Signature detection») – це метод перевірки, під час якого антивірус ретельно перевіряє файли, які вносяться (надходять) в систему чи вже були раніше в ній, для аналізу більш ймовірних небезпек чи погроз із боку приховано у цих фалах кодів.

Антивірусні програми пропонуються користувачу вже з каталогом (базою) раніше виявлених і перевірених вірусів, та у ході роботи дане програмне забезпечення порівнює коди та шаблони у файлах (інших програмних додатках) чи веб-сторінках з унікальними бітами та шаблонами, що складають код вірусу. У разі, коли дані коди збігаються, файл відправляється антивірусом на карантин, що означає його переміщення в нове ізольоване місце, щоб він не заразив інші файли в системі.

Антивірусні програми надають змогу перевірити будь-яку потенційну шкідливу поведінку в операційній системі, прикладом таких дій можуть бути: ненадійні джерела, що ведуть записи реєстру або автоматичне виконання невідомої програми під час запуску системи.

Надалі буде розглянуто більш детально різноманітні методи для виявлення «Шкідників» системи, та більше розширено розказано певні тонкощі та особливості того чи іншого методу, проте для початку перерахуємо основні з них з коротким описом.

Визначення вірусів (англійською «Virus Definitions») по суті, являє собою перший метод, який традиційне антивірусне забезпечення використовує для знаходження вірусу. Програми виконують пошук підписів для виявлення шкідливих програм. Антивірусне забезпечення проводить аналіз та знаходить конкретний підпис файлу та зберігають його у своїй базі даних, після чого порівнюють загрози, і у разі виявлення збігів, захищають пристрої.

Евристичне виявлення (англійською «Heuristic-based detection») – це спосіб пошуку, що задіює алгоритм порівняння підпису вже відомих у світі вірусів на протигагу виявлення потенційної загрози. Антивірус, що використовує даний тип виявлення, також може віднайти віруси, які ще не були широко розповсюджені (тобто нові види загроз), проте є і недолік, який криється у наступному: антивірус може згенерувати хибні позитивні збіги, що вказують на незаражений файл як заражений.

Виявлення на основі поведінки (англійською «Behavior-based detection») полягає в наступному – у разі, якщо вищевказані методи виявлення, не знайшли «шкідників», то антивірус переходить у стан спостерігача та сканує поведінку програм, запущених на персональному комп'ютері. Програма-захисник виводить користувачу повідомлення з попередження, що певний додаток чи файл виконує «дивні» дії, що перелічені нижче:

- змінюються налаштування інших, не пов'язаних до використовуваного додатку програм чи файлів;
- десятки даних починають змінюватись чи видаляться без поважних на то причин (не спричинені діями користувача);
- відбувається спроба дистанційного підключення до даного персонального технічного засобу з раніше не відомих точок.

Виявлення у пісочниці (англійською «Sandbox Detection») – це метод аналізу операційної системи та всіх її файлів, при якому відбувається запуск програми у створеному чистому віртуальному середовищі та безпосередній запис послідовності виконання тих чи інших дій, щоб визначити, чи є програми, що піддаються аналізу шкідливими. У разі, коли програма вважається «безпечною» та проходить дану перевірку без ускладнень, вона починає виконуватись вже у реальному середовищі.

Цей метод не є розповсюдженим, у антивірусних рішеннях, що розповсюдженні у сьогоденні для звичайних користувачів та їх персональних комп'ютерів, оскільки він трудомісткий та досить повільний, проте у цілях збереження цілісності корпоративного використання можуть мати свою популярність.

Видобуток даних (англійською «Data Mining») працює на наступним принципом: перш за все з файлів вилучається ряд функцій (кодів), після чого використовуються алгоритми інтелектуального аналізу даних та машинного навчання для визначення поведінки файлу, для визначення потенціальної шкідливості файлу для системи.

Як можна зрозуміти, антивірусне програмне забезпечення може включати в себе широкий та різноманітний спектр методів аналізу та виявлення, включаючи такі як: пакетна обробка; аналіз у момент доступу; блокування, що засноване на поведінці, тощо.

Не звертаючи увагу на те, що основні механізми аналізу доволі часто опираються на складні концепції, подібні до байт-коду, різних інтерпретаторів

та інших представлень, фактична перевірка відповідності підписів зводиться до трьох основних стратегій: шаблонів байтів (англійською «byte patterns»), хеш-сум (англійською «hash sums») та евристики (англійською «Heuristics»).

2.3.1 Стратегія розпізнавання «Byte Patterns»

Досить відомою стратегією виявлення на основі підписів є використання шаблонів байтів. Дані підписи містять одну або кілька постійних послідовностей байтів, потенційно можуть бути розділені прогалинами різного розміру, які повинні відповідати вмісту файлу, щоб викликати виявлення. Знайдені підписи можуть бути порівняні за допомогою класичних методів обробки рядків. З цією метою набори рядків представлені у вигляді дерева, що служить кінцевим показником стану для визначення збігів. За допомогою представлення, з'являється змога моделювати знаки, діапазони та класи символів, забезпечуючи тим самим можливість, подібні до властивостей регулярних виразів.

Розглянемо приклад шаблонів байтів на основі безкоштовного проекту з відкритим кодом під назвою ClamAV. Сканер визначає простий формат для представлення шаблонів байтів, включаючи роз'єднання (aa | bb), прогалини {n-m}, символи підстановки «?» та класи символів.

На рисунку 2.1 (a) показана спрощена версія такого підпису для сімейства шкідливих програм Virut. Хоча ClamAV, як правило, може відповідати довільним даним, у наведеному прикладі підпис описує код x86, який відповідає простому порядку розшифрування.

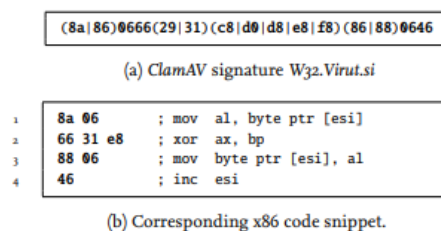


Рисунок 2.1 – Підпис ClamAV для шкідливого програмного забезпечення Virut (a) та відповідний фрагмент коду x86 (b)

Формально підписи на основі шаблонів є просто підмножиною регулярних виразів, однак для цілей виведення підписів ми обираємо інший офіційний опис, який повністю зосереджується на їх зовнішньому вигляді. Ми бачимо, що підписи на основі шаблонів даються послідовностями переходів над байтами, можливо розділеними прогалинами різного розміру.

Висловлюючи кожну диз'юнкцію як множину, що містить її альтернативи, підпис стає послідовністю наборів. Ми можемо описати це формально, визначивши символічний алфавіт $S = P(\{0, \dots, 255\})$, де P - множина потужності і відповідає всім можливим підмножинам значень байтів.

Крім того, ми використовуємо $*$ як ярлик для $\{0, \dots, 255\}$, що представляє нерелевантні байти. Кожне слово $w \in S^*$ відповідної мови тоді вже повністю виражає формат підпису. Однак лише ці слова не пояснюють різного розміру прогалин. Отже, ми введемо дві функції, $l: \{1, \dots, |w|\} \rightarrow \mathbb{N}$ та $h: \{1, \dots, |w|\} \rightarrow \mathbb{N}$, які призначають мінімальну та максимальну кількість повторів кожному із символів w . Тоді підпис на основі шаблону просто подається кортежем $s = (w, l, h)$.

2.3.2 Стратегія розпізнавання «Hash Sums»

Дана стратегія реалізується за допомогою відповідності на основі хеш-сум. З цією метою хеш-суми по повних файлах або частинах файлів обчислюються і зберігаються в базі підписів. Прості контрольні суми, такі як CRC32 або криптографічні хеш-функції, такі як MD5 або SHA1, часто використовуються через наявність швидких реалізацій як у програмному, так і в апаратному забезпеченні.

Хоча зіткнення хешу теоретично можуть призвести до хибно позитивних результатів, на практиці це не є проблемою в цій конкретній сфері застосування, що робить його привабливим вибором для багатьох розробників.

У порівнянні з шаблонами байтів, хеш-суми дозволяють зіставляти великі об'єми інформації у файлі з компактним підписом. Такий підхід, однак, не допускає використання символів підстановки або пропусків, а отже, не дає жодних засобів для зіставлення в основному схожих файлів з одним підписом. Як наслідок, індивідуальні підписи на основі хешу потрібні навіть для найменших варіантів відомих зразків шкідливого програмного забезпечення, а, отже, підписи на основі шаблонів можуть в кінцевому підсумку краще відповідати вимогам до місця.

Розглянемо приклад хеш-сум на основі безкоштовного проекту з відкритим кодом під назвою ClamAV. Сканер дозволяє визначати підписи на основі хешу або як хеш-суму повного файлу, або спеціально для файлів PE над окремими розділами.

Рисунок 2.2 ілюструє обидва типи хеш-сум для шкідливого програмного забезпечення Kido, де в першому випадку довжина відповідного регіону вказано у другому параметрі (162970), а в останньому випадку – у перший параметр (81920). Зауважте, що тип хеш-функції виводиться ClamAV на основі розмір хеш-суми або файлу бази даних, у якій вона зберігається.

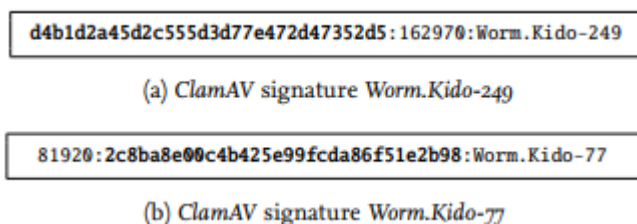


Рисунок 2.2 – Приклад двох підписів на основі хешу, що відповідають (а) повному файлу та (б) певному розділу PE

Концептуально підписи на основі хешу можна інтерпретувати як безперервні шаблони байтів. Зокрема, хеш-суму із заданим зміщенням та довжиною можна представити у вигляді безперервного байтового шаблону, якому передуює проміжок фіксованого розміру.

2.3.3 Стратегія розпізнавання «Heuristics»

Крім шаблонів байтів і хеш-сум, антивірусне програмне забезпечення часто використовує кілька додаткових евристик для виявлення загроз безпеці. Дані евристики включають перевірку інструкцій підрахунків, аналіз використання API, моделі виявлення n-грамів, а в деяких випадках навіть методи машинного навчання.

У порівнянні з підписами на основі шаблонів байтів і хеш-сум, ці підходи виявлення часто прив'язані до конкретного контексту виконання і можуть захоплювати складну семантику, таку як різні події, необхідні для запуску інфекції. Через цю складність евристика є менш придатною для створення шкідливих маркерів, і тому ми зосереджуємося лише на підписах на основі шаблонів.

2.4 Формати підписів

Як вже було зазначено раніше, антивірусні бази даних містять так звані вірусні підписи, що представляють собою безперервну послідовність байтів, спільних для певного зразка шкідливого програмного забезпечення. Згідного даного пояснення, можна зрозуміти, що він міститься всередині шкідливого програмного забезпечення або зараженого файлу чи додатку, проте не у всій системі, є також і не пошкоджені файли, що не мають збігу підпису з базою.

Один підпис може узгоджуватися з великою кількістю вірусів. Це дозволяє сканеру виявити абсолютно новий вірус, якого він ніколи раніше не бачив. Ця здатність зазвичай називається або евристикою, або загальним виявленням, про це вже було описано раніше, тому слід також розглянути й усі формати підписів, що досить часто зустрічаються серед розробників антивірусного програмного забезпечення.

Підписи на основі хешу (англійською «Hash-based signatures») – представляють собою найпростіший спосіб реалізації з використанням

контрольних сум хешу файл. Однак цей метод є досить обмеженим, оскільки покриває виявлення лише статичних шкідливих програм.

Існує декілька досить популярних підписів на основі хешу, з них можна навести наступні:

- «MD5 hash-based signatures» – в основі підпису лежить алгоритм хешування MD5;
- «SHA hash-based signatures» – в основі підпису лежить алгоритм хешування SHA, що налічує декілька різних підвидів;
- «PE section based hash signatures» – в основі підпису лежить аналіз PE секції виділеної з файлу, а не всього файлу в цілому;
- «Hash signatures with unknown size» – алгоритм, що працює на основі хешу, де розмір файлу не відомий, проте відомий розмір його хешу.

Підписи на основі тіла програмного забезпечення (англійською «Body-based signatures») – для збереження можна використовувати шістнадцятковий формат. У цьому визначення під шістнадцятковим підписом мається на увазі фрагмент тіла шкідливого забезпечення, перетворений на шістнадцятковий рядок, який можна додатково розширити за допомогою різних символів підстановки.

Підписи значків для файлів PE (англійською «Icon signatures for PE files») – підпис, що містить приблизний/нечіткий збіг значків (зображень), за допомогою якого можливо виявити шкідливі виконувані файли, що маскуються під певну форму зображень, документи тощо.

Підписи метаданих у файлах PE (англійською «Signatures for metadata in PE files») – надає змогу легко зіставити певну побудовану інформацію у файли PE (виконувані файли та бібліотеки динамічних посилань) з базою антивірусу та видати результат, щодо добро якості просканованого файлу.

2.5 Математична модель алгоритмів хешування

2.5.1 MD5 hash-based signatures

MD5 – з англійської розшифровується як «Message-Digest Algorithm 5th version» та перекладається наступним чином «п'ята версія алгоритму обробки повідомлень» – це п'ята версія алгоритму, розроблена Роном Рівестом для створення 128-розрядного дайджесту повідомлень. MD5 є доволі швидким алгоритмом, який приймає звичайний текст із 512-розрядних блоків, що далі поділяється на меншу кількість, а саме на 16 блоків, кожен із 32-розрядних, і створює 128-розрядний обробник повідомлень, що є набором з чотирьох блоків, кожен з 32 бітів.

Використання алгоритму MD5 – досить просте і зрозуміле, даний алгоритм був розроблений з ціллю поліпшення безпеки, оскільки він приймає введення будь-якого розміру і видає результат, як значення 128-бітового хешу.

Алгоритм MD5 був придуманий та створений для зберігання хешу пароля, а деякі файлові сервери також забезпечують заздалегідь обчислену контрольну суму файлу MD5, а зроблено це для того, щоб можна було порівняти з ним розраховану по цьому ж алгоритму контрольну суму завантаженого файлу. Операційні системи, що працюють на базі Unix часто включають у свої пакети розповсюдження розрахунку контрольної суми за алгоритмом MD5.

Надалі буде більш детально описано принцип дії розрахунків у MD5. Даний алгоритм працює у стані 128-розрядів, що розділеі на чотири 32-розрядні слова, тому для полегшення опису вони позначені наступними літерами A, B, C і D. Обробка блоку повідомлень складається з чотирьох подібних етапів, які називаються раундами; кожен раунд складається з 16 подібних одна одній операцій, заснованих на нелінійній функції F, модульному додаванні та обертанні вліво. Позначення будуть наступними:

M_i – виступає у ролі позначення 32-розрядного блоку введення повідомлення;

K_i – виступає у ролі позначення 32-бітної константи, що матиме різне значення для кожної операції;

$\lll_s$ – виступає у ролі позначення поворот лівого біта на s місця (s змінюється для кожної операції).

Рисунок 2.3 відображає повний цикл однієї операції в одному раунді. Існують чотири можливі виконувані функції (рисунок 2.4), та слід зазначити, що для кожного раунду лише один раз задіюється дана функція.

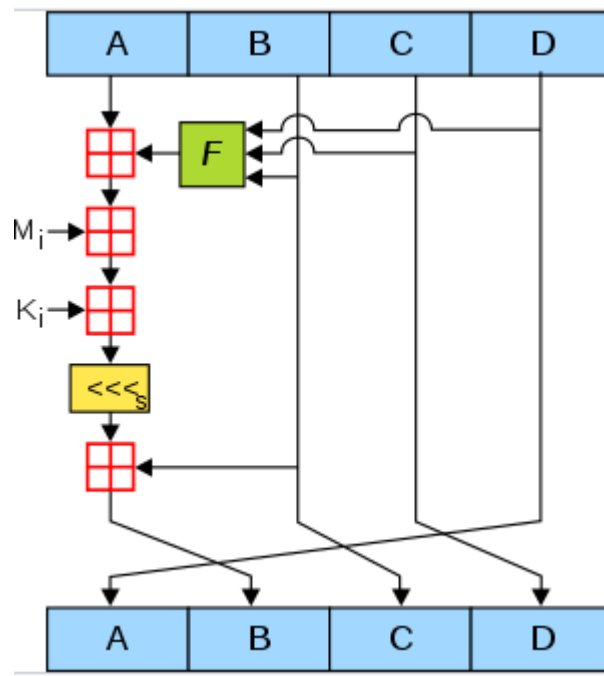


Рисунок 2.3 – Операції, що виконуються у кожному етапі (раунді) обробки повідомлення

Усі можливі функції, що використовуються під час обробки повідомлення виглядають наступним чином:

$$F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$$

$$G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$$

$$H(B, C, D) = B \oplus C \oplus D$$

$$I(B, C, D) = C \oplus (B \vee \neg D)$$

Символи $\oplus, \wedge, \vee, \neg$ позначають операції XOR, AND, OR та NOT відповідно.

Розглянемо детальніше процес обробки повідомлення у вигляді псевдокоду даного алгоритму (коментарі до певної частини коду будуть помічені зеленим кольором та починатимуться зі знаків //):

```
// Усі змінні є без знаковими 32-бітними та обертаються за модулем  $2^{32}$  під час підрахунків
var int s[64], K[64]
var int i
//Визначення таблиці констант :
for i from 0 to 63 do
    K[i] := floor( $2^{32} \times \text{abs}(\sin(i + 1))$ )
end for

// Ініціалізація змінних :
var int a0 := 0x67452301 // A
var int b0 := 0xefcdab89 // B
var int c0 := 0x98badcfe // C
var int d0 := 0x10325476 // D

// Підготовчий етап: додавання одного біту в кінець повідомлення
append "1" bit to message

// Підготовчий етап: дописування нульових бітів, поки довжина рядка не стане з 448 по модулю 512
append "0" bit until message length in bits  $\equiv 448 \pmod{512}$ 

// Допишуємо залишок від ділення початкової довжини повідомлення на  $2^{64}$ 
append original length in bits mod  $2^{64}$  to message

// Розподіляємо повідомлення на 512-бітні частини:
for each 512-bit chunk of padded message do
    break chunk into sixteen 32-bit words M[j],  $0 \leq j \leq 15$ 
    // Ініціалізація хешу:
    var int A := a0
    var int B := b0
    var int C := c0
    var int D := d0
    // Основний цикл:
    for i from 0 to 63 do
        var int F, g
        if  $0 \leq i \leq 15$  then
            F := (B and C) or ((not B) and D)
            g := i
        else if  $16 \leq i \leq 31$  then
            F := (D and B) or ((not D) and C)
            g :=  $(5 \times i + 1) \bmod 16$ 
        else if  $32 \leq i \leq 47$  then
            F := B xor C xor D
            g :=  $(3 \times i + 5) \bmod 16$ 
        else if  $48 \leq i \leq 63$  then
            F := C xor (B or (not D))
```

```

        g := (7*i) mod 16

        F := F + A + K[i] + M[g]  // M[g] має бути 32-бітним блоком
        A := D
        D := C
        C := B
        B := B + leftrotate(F, s[i])
    end for

    // додаємо результат даної частини до загального результату:
    a0 := a0 + A
    b0 := b0 + B
    c0 := c0 + C
    d0 := d0 + D
end for

// (Результат в форматі little-endian)
var char digest[16] := a0 append b0 append c0 append d0

```

Замість формулювання з оригінального RFC 1321, наведеного нижче, може бути використано для підвищення ефективності (корисно, якщо використовується мова асемблера - інакше компілятор, як правило, оптимізує наведений вище код. Оскільки кожне обчислення залежить від іншого у цих формулюваннях, це часто повільніше, ніж вищезгаданий метод, де `nand/i` можна паралелізувати):

```

( 0 ≤ i ≤ 15): F := D xor (B and (C xor D))
(16 ≤ i ≤ 31): F := C xor (D and (B xor C))

```

Даний алгоритм досить повільний, на зміну йому поступово приходить SHA алгоритм. SHA набагато безпечніший, ніж алгоритм MD5, і, крім того, він може бути реалізований у існуючій технології з перевищенням швидкості. В даний час на ринку з'являються нові алгоритми хешування, які мають на увазі підвищену безпеку даних, таких як SHA256 (який генерує 256 біт підпису тексту). Надалі розглянемо алгоритми SHA1 та SHA256.

2.5.2 SHA hash-based signatures

Алгоритм SHA (повністю англійською «Secure Hash Algorithm») – це алгоритм безпечного хешу, розроблений Національним інститутом стандартів і технологій разом з NSA, раніше випущений як Федеральний стандарт

обробки інформації, пізніше в 1995 році він був названий алгоритмом SHA, розробленим для модифікації MD4, іншими словами, ми можна сказати, що алгоритм SHA є модифікованою версією MD4.

SHA призначена для отримання вихідного повідомлення з урахуванням його дайджесту повідомлення та для пошуку повідомлення, що виробляє те саме повідомлення.

У сфері криптографії алгоритм SHA – це функція хешування у форматі кріпти, яка використовується для прийняття менших даних і створює рядок довжиною 160 біт, також відомий як 20-байтове значення хешу. Згенероване таким чином значення хешу відоме як дайджест повідомлення, який зазвичай відтворюється та створюється як шістнадцяткове число, яке має спеціально 40 цифр.

Надалі необхідно описати характеристики даного хешу та підкреслити певні ключові моменти алгоритму роботи. Криптографічні хеш-функції використовуються і використовуються для збереження та зберігання захищеної форми даних шляхом надання трьох різних видів характеристик, таких як опір попередньому зображенню, який також відомий як перший рівень стійкості зображення, другий рівень попереднього зображення опір і стійкість до зіткнення.

До різних типів алгоритмів SHA належать наведені нижче види та підвиди.

SHA0 – це ретронім, який застосовується до основної версії 160-річної або 20-байтової довготривалої хеш-функції, яка була опублікована в 1993 році з назвою алгоритму SHA. Його було вилучено дуже скоро після публікації через серйозну ваду, тому подальший розгляд не ефективним.

SHA1 – Це 160-розрядний або 20-байтовий хеш-механізм шифрування на основі функцій, досить схожих до алгоритму MD5. Даний алгоритм був придуманий та розроблений NSA (повністю «National Security Agency»), тобто Агентством національної безпеки, і повинен був бути частиною компонента

алгоритму цифрового підпису (DSA). Слабкі місця, пов'язані з криптографічними методами, були виявлені в SHA1 та пізніше стандарт шифрування був відкинутий і його не надто багато використовували.

SHA2 – формує сімейство з 2 однакових хеш-функцій, які складаються з блоків різного розміру, відомих як SHA-512 та SHA-256, які відрізняються головним чином розміром слова. Перший складається з діапазону значень слова 32 слова, тоді як другий складається з 64-розрядного значення слова. Укорочені версії цих значень включають SHA-224, SHA-384 та SHA-512 та SHA-224 або SHA-256.

SHA3 – це метод шифрування, який використовується переважно сьогодні, і який використовує хеш-функцію під назвою Кессак. Підтримувана довжина така ж, як у SHA-2. Однак більшість відмінностей полягає в тому, що ця структурно відрізняється, оскільки базується на широкому діапазоні генерації випадкових функцій, яка зазвичай підтримує всі випадкові перестановки і тим самим дозволяє вводити або поглинати, як її називають, будь-яку кількість представлених даних та виведення або стискання представлених даних. При цьому все це діє як псевдовипадкова функція для всіх наданих входів, що, отже, призводить до більшої гнучкості. Надалі буде описано алгоритм SHA у вигляді псевдокоду, для легшого розуміння принципу його дії.

Припустимо, що повідомлення 'abc' має бути закодовано за допомогою SHA алгоритму, при цьому повідомлення 'abc' у двійковому форматі має вигляд:

01100001 01100010 01100011

а в шістнадцятковому форматі вигляд:

616263

Першим кроком є ініціалізація п'яти випадкових рядків шістнадцяткових символів, які будуть служити частиною хеш-функції (показано у вигляді hex):

$H_0 = 67DE2A01$

$$\begin{aligned} H_1 &= BB03E28C \\ H_2 &= 011EF1DC \\ H_3 &= 9293E9E2 \\ H_4 &= CDEF23A9 \end{aligned}$$

Повідомлення потім додається, додаючи 1 та 0, поки повідомлення не стане 448 біт. Потім довжина повідомлення, представлена 64 бітами, додається до кінця, створюючи повідомлення довжиною 512 біт:

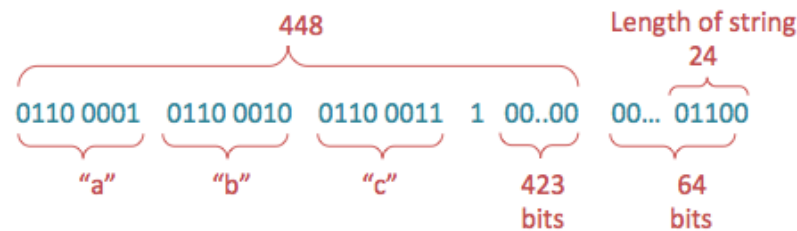


Рисунок 2.4 – Доповнення рядка "abc" в бітах, завершене довжиною рядка, яка становить 24 біта.

Отриманий вище вхід, M , потім поділяється на 512-розрядні фрагменти, і кожен фрагмент додатково ділиться на шістнадцять 32-розрядних слів, $W_0 \dots W_{15}$. У разі "abc" є лише один фрагмент, оскільки загальна кількість повідомлення становить менше 512 біт.

Для кожного фрагмента починаються 80 ітерацій, і, необхідних для хешування (80-це визначене число для SHA-1), далі йде виконання наступних кроків для кожного фрагмента M_n :

Для ітерацій з 16 по 79, де $\leq i \leq 79$, виконується наступна операція:
 $W(i) = S_i(W(i - 3) \oplus (W(i - 8) \oplus W(i - 14) \oplus W(i - 16))),$
де XOR або $\oplus$ представлено таким порівнянням входів x та y :

Таблиця 2.1 – Представлення порівняння виходів x та y

x	y	Output
0	0	0
1	0	1
0	1	1
1	1	0

Наприклад, коли i дорівнює 16, вибрані слова $W(13), W(8), W(2), W(0)$, та вихід - це нове слово, $W(16)$, тому, виконуючи XOR або $\oplus$, операція над цими словами дасть це (результати операцій обчислення будуть наведені нижче, у таблиці 2.2):

Таблиця 2.2 – Таблиця результатів виконання операцій XOR

$W(0)$	01100001 01100010 01100011 10000000
$W(2)$	00000000 00000000 00000000 00000000
$W(8)$	00000000 00000000 00000000 00000000
$W(13)$	00000000 00000000 00000000 00000000
	$\oplus$
$W(16)$	01100001 01100010 01100011 10000000

Наступним буде операція кругового зсуву $S_n(X)$ на слово X на n бітів, де n – ціле число між 0 та 32 визначене так:

$$S_n(X) = (X \ll n) \text{OR} (X \gg 32 - n),$$

де $(X \ll n)$ – операція зсуву ліворуч, отримана шляхом відкидання крайніх лівих n -бітів X та заповнення результату нулями n праворуч.

$(X \gg 32 - n)$ – операція зсуву вправо, отримана шляхом відкидання крайніх правих n -бітів X та заповнення результату нулями n зліва. Таким чином, $S_n(X)$ еквівалентно круговому зсуву X на n позицій, і в цьому випадку використовується круговий зсув вліво.

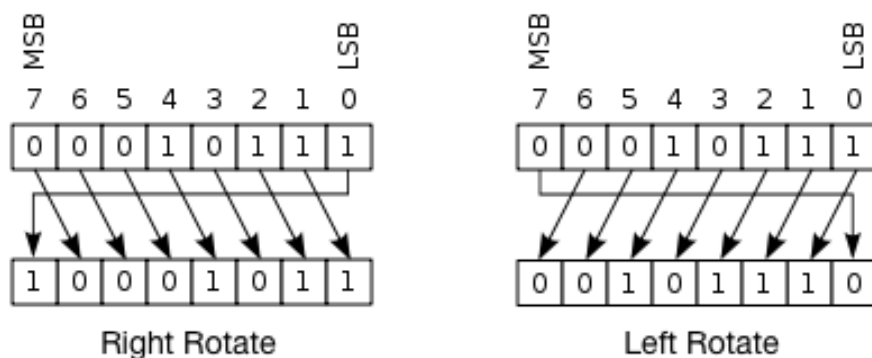


Рисунок 2.5 – Операція зсуву

Отже, зсув ліворуч $S_n(W(i))$, де $(W(i))$, де $W(i)$ дорівнює 10010, вийде 01001, оскільки крайній правий біт 0 зміщений до лівого боку рядка. Отже, $W(16)$ закінчиться

11000010 11000100 11000111 000000000

Тепер значення хешу, визначені на кроці 1, буде збережено у таких змінних:

$$A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$$

Для 80 ітерацій, де $0 \leq i \leq 79$, буде виконано обчислення:

$$TEMP = S_5(A) + f(i; B, C, D) = E + W(i) + k(i)$$

Детальніше про логічну функцію f та значення $K(i)$:

$$E = D$$

$$D = C$$

$$C = S^{30}(B)$$

$$B = A$$

$$A = TEMP$$

Збереження результату хешу фрагмента до загального значення хешу всіх фрагментів, відбувається як показано нижче, і далі буде перехід до виконання наступного фрагмента:

$$H_0 = H_0 + A$$

$$H_1 = H_1 + B$$

$$H_2 = H_2 + C$$

$$H_3 = H_3 + D$$

$$H_4 = H_4 + E$$

На завершальному етапі, коли всі частини були оброблені, вивід повідомлення представляється у вигляді 160-розрядного рядка, що складається з логічного оператора АБО, $\vee$ з 5 хешованих значень:

$$H = S(H_0) \vee S_{96}(H_1) \vee S_{64}(H_2) \vee S_{32}(H_3) \vee H_4$$

Отже, рядок "abc" стає представленим значенням хешу, подібним до:

a9993e364706816aba3e25717850c26c9cd0d89d

Наприклад, якщо рядок зміниться на "abcd", значення хешування буде різко відрізнятися, тому зломисники не зможуть сказати, що воно схоже на початкове повідомлення. Значення хешу для 'abcd' становить:

81fe8bfe87576c3ecb22426f8e57847382917acf

Функції, що використовуються в алгоритмі. Послідовність логічних функцій використовується в SHA-1, в залежності від значення i , де $0 \leq i \leq 79$, і від трьох 32-розрядних слів B , C і D для створення 32-розрядний вихід. Наступні рівняння описують логічні функції, де $\neg$ це логічне НЕ, $\vee$ - логічне АБО, $\wedge$ - логічне І, $\oplus$ - логічне XOR:

$$f(i; B, C, D) = (B \wedge C) \vee ((\neg B) \wedge D) \quad \text{for } 0 \leq i \leq 19$$

$$f(i; B, C, D) = B \oplus C \oplus D \quad \text{for } 20 \leq i \leq 39$$

$$f(i; B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \quad \text{for } 40 \leq i \leq 59$$

$$f(i; B, C, D) = B \oplus C \oplus D \quad \text{for } 60 \leq i \leq 79$$

Крім того, у формулах використовується послідовність незмінних слів, показана в шестнадцятковій формі нижче:

$$K(i) = 5A827999 \quad \text{where } 0 \leq i \leq 19$$

$$K(i) = 6ED9EBA1 \quad \text{where } 20 \leq i \leq 39$$

$$K(i) = 8F1BBCDC \quad \text{where } 40 \leq i \leq 59$$

$$K(i) = CA62C1D6 \quad \text{where } 60 \leq i \leq 79$$

Алгоритми SHA широко використовуються в протоколах безпеки та додатках, включаючи такі, як TLS, PGP, SSL, IPsec та S/MiME. Вони також знаходять своє місце у більшості криптоаналітичних прийомів та стандартів кодування, які в основному спрямовані на те, щоб бачити функціонування та роботу переважно всіх державних, а також приватних організацій та установ. Великі сьогоденні гіганти, такі як Google, Microsoft або Mozilla, почали рекомендувати використання SHA3 і припинити використання алгоритму SHA1.

Алгоритм SHA або захищеного хешування має на меті надати додатковий рівень безпеки зростаючим та масовим даним, з якими вам доводиться мати справу. Хакери та зловмисники продовжуватимуть знаходити вразливість у всіх нових формах хешування, які використовуються. Ми просто повинні переконатися, що ми достатньо оперативні, щоб бути більш безпечними, ніж дозволити нашим даним стати жертвою їх.

2.5.3 PE section based hash signatures

Контрольна, чи іноді називають ще перевірна, сума (англійською «checksum») – представляє собою певну послідовність цифр та букв, що будуть використані для перевірки файлів, документів, даних тощо. Щоб створити контрольну суму, здебільшого необхідно задіяти типові алгоритми перетворення, вони включають в себе наступні: MD5, SHA-1, SHA-256 та SHA-512.

Іншими, словами перевірочну суму можна описати, як це дуже простий алгоритм хешування, який дасть істотно різні результати навіть при невеликих змінах вхідних даних. Він працює шляхом обчислення функції хешування по вмісту файлу для отримання кінцевого результату. Тому файли з різним вмістом матимуть різні контрольні суми. Це не завжди так, оскільки можуть виникати хеш-колізії, коли два файли з різним вмістом мають однакову контрольну суму. Ймовірність зіткнення відносно низька, тому використання контрольної суми допомагає більшості часу, але не є гарантією.

Перейдемо більш детально до реалізацій контрольних сум у файлах PE, проте для початку, розглянемо більш детальний опис про даний формат файлів. Портативний виконуваний (PE) формат файлу - це формат файлів для виконуваних файлів, представлених у Windows NT. Він базується на специфікації COFF (Common Object File Format).

Формат файлу PE організований як лінійний потік даних. Він починається із заголовка MS-DOS, заглушки програми в реальному режимі та

підпису файлу PE. Одразу слідує заголовок файлу PE та додатковий заголовок. Крім того, з'являються всі заголовки розділів, а потім усі тіла розділів. Закривають файл ще кілька областей різної інформації, включаючи інформацію про переміщення, інформацію таблиці символів, інформацію про номер рядка та дані таблиці рядків. Все це легше засвоюється, якщо подивитися на це графічно, як показано на рисунку 2.6.

MS-DOS заголовок
MS-DOS заглушка
Підпис PE (PE Signature)
PE заголовок
Опціональний PE заголовок
Таблиця секцій
Дані

Рисунок 2.6 – Структура зображення портативного виконуваного файлу

Кожен виконуваний файл формату PE складається з безлічі взаємозалежних структур, що містять інформацію про сам файл, про те, що імпортується і експортується в його функціях, про всі переміщення елементів, ресурсів.

Кожен PE-файл складається з перерахованих вище елементів (рисунок 14), вони є обов'язковими. З самого верху знаходиться MS-DOS-заголовок – це спадщина ще з тих самих часів, коли відбувався перехід з DOS на Windows, що підтримує новий PE-формат.

Залишилося сказати про відмінності PE32 і PE64. Єдина відмінність в структурах цих форматів полягає в тому, що всі віртуальні адреси (VA) - це 64-розрядні числа (8-байтові). Всі відносні адреси (RVA) так і залишилися 32-розрядними.

Висновок до розділу 2

У даному розділі було розглянуто найпоширеніші види вірусів та антивірусних програм, для того щоб зрозуміти власне проти чого необхідно «боротись» та які саме методи мають допомогти у даному питанні.

За допомогою даного розділу, можна зрозуміти, що «загроза» персонального комп'ютера існує не тільки в інтернеті, у вигляді реклами чи спаму на електронній пошті, а і в інших додатках, що користувач використовує для власних потреб (таких як комп'ютерні ігри чи документи). Вірус може бути схований у середині будь-якого файлу та нести в собі різні види загроз, починаючи від зараження операційної системи та подальшого її руйнування, закінчуючи наданням контролю технічним засобом зловмисникам через віддалений робочий стіл та інше.

Згідно зробленого дослідження у розділі 2, можна також побачити, що антивірусні програми бувають різних видів та потенційний захист від них буває різним. У подальшій роботі необхідно буде розробити додаток, що покриє потребу користувача зберігати цілісність його технічного засобу та буде виявляти (а надалі і пропонувати видалити) той чи інший шкідливий файл/програму. Інтерфейс розробленого додатку має бути простим і зрозумілим у використанні, а оновлення бази має відбуватися регулярно.

З розглянутих алгоритмів для виявлення і розпізнавання вірусного програмного забезпечення досить ефективним є MD5 та SHA2, тому дані алгоритми будуть використані у створеному в даній роботі додатку.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ ШКІДЛИВИХ ФАЙЛІВ

3.1 Архітектура додатку

Важко знайти точне визначення архітектури програми. Можна стверджувати, що це підмножина архітектури програмного забезпечення, яка в першу чергу зосереджена на окремих програмах, на відміну, наприклад, від архітектури підприємства, яка охоплює все програмне забезпечення всередині компанії, включаючи взаємодію між різними додатками.

Архітектура програми відповідає за структуру програми високого рівня, яка служить орієнтиром або планом для всієї роботи з розробки цього додатка. Часто люди порівнюють це з архітектурою будівель. Однак між ними є дуже важлива різниця. На відміну від будівель, програмне забезпечення дуже змінюється і ніколи не «зроблено». Це відбивається на його архітектурі, яка також потребує постійної адаптації до мінливих вимог та подальшого розвитку.

Архітектура програми завжди повинна базуватися на вимогах, визначених зацікавленими сторонами (власниками бізнесу, користувачами тощо).

Важливо, щоб це включало не лише функціональні вимоги (тобто те, що потрібно робити додатку), але й нефункціональні вимоги. Останній стосується продуктивності програми, надійності, ремонтпридатності та інших аспектів, які безпосередньо не впливають на його функціональність, але впливають на те, як додаток сприймається його користувачами, розробниками та іншими зацікавленими сторонами.

Архітектурні рішення приймаються на різних рівнях. Для початку розберемо це більш детально за допомогою рисунку 3.1.

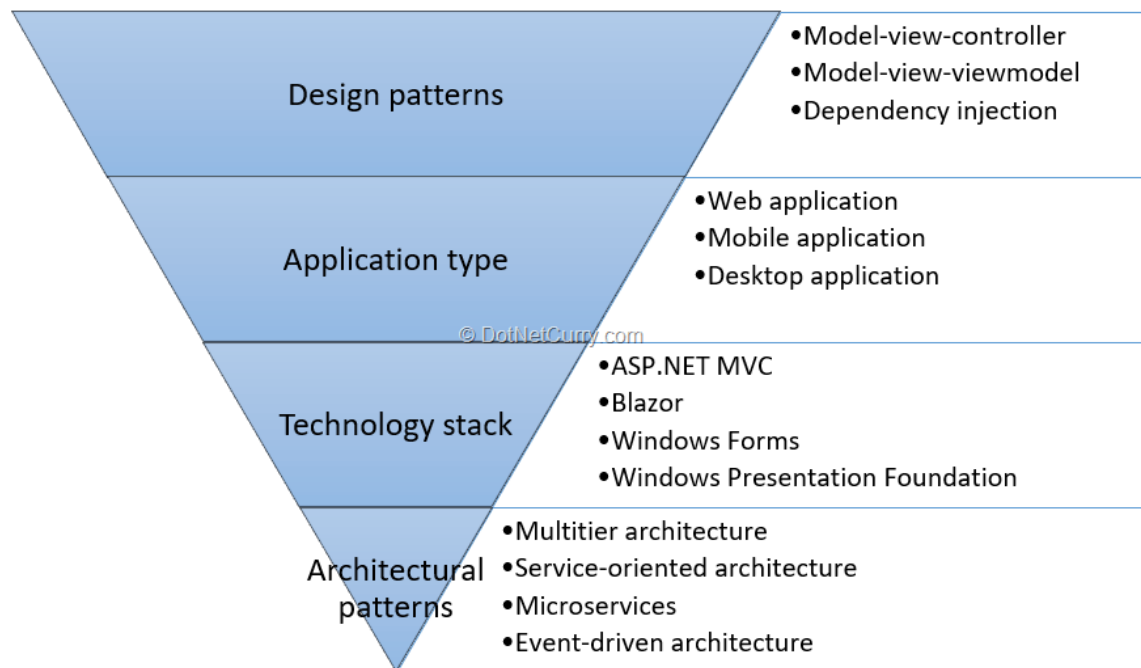


Рисунок 3.1 – Архітектурні рішення на різних рівнях

Архітектурні паттерни – виконують роль, подібну до шаблонів дизайну, тільки вони застосовні на більш високому рівні. Вони є перевіреними рішеннями багаторазового використання для поширених сценаріїв архітектури програмного забезпечення.

Ось приклади найвідоміших архітектурних паттернів.

У багаторівневій архітектурі (англійською – «Multitier Architecture») програми структуровані на кілька рівнів, які мають окремі обов’язки і зазвичай також фізично розділені. Ймовірно, найпоширенішим застосуванням цього архітектурного шаблону є трирівнева архітектура, яка складається з рівня доступу до даних (включаючи сховище даних), бізнес-рівня (реалізує бізнес-логіку) та рівня презентації (тобто інтерфейс користувача програми).

У сервіс-орієнтованій архітектурі (англійською – «SOA», що розшифровується як «Service-Oriented Architecture») компоненти програми реалізовані як незалежні автономні служби, які спілкуються за стандартизованим протоколом зв’язку.

Стандартизований контракт на обслуговування описує функціональні можливості кожної служби. Це дозволяє послабити зв’язок між службами та

високий рівень їх повторного використання. Коли вперше був представлений шаблон архітектури, найбільш часто використовуваним протоколом зв'язку був SOAP.

Мікросервіси (англійською – «Microservices») є більш сучасною підмножиною або еволюцією сервісно-орієнтованої архітектури (SOA). Як випливає з назви, вони, як правило, більш легкі, включаючи протоколи зв'язку, якими найчастіше є REST і gRPC. Кожна служба може використовувати будь-яку технологію, яка найкраще підходить для неї, але всі вони повинні бути незалежно розгорнуті з високим рівнем автоматизації.

В архітектурі, керованій подіями (англійською – «Event-driven Architecture»), ще більше акцентується на слабкому з'єднанні та асинхронному зв'язку між компонентами. Хоча це не обов'язкова вимога, система все одно може бути розподіленою, тобто вона може складатися з незалежних служб (або мікросервісів).

Основним відмінним фактором є те, що компоненти не зв'язуються за допомогою імперативних викликів. Замість цього вся комунікація відбувається за допомогою подій (або повідомлень), які публікуються компонентами, а потім прослуховуються іншими компонентами, які в цьому зацікавлені. Компонент, який розміщує подію, не отримує на нього прямої відповіді. Однак він може отримати непряму відповідь, прослухавши інше повідомлення. Це робить комунікацію більш асинхронною в порівнянні з іншими шаблонами.

У межах вибраного архітектурного шаблону високого рівня необхідно прийняти ще багато архітектурних рішень нижнього рівня, які ще більш детально визначають, як буде структурований код програми.

Існують додаткові шаблони, доступні для цих рішень, наприклад шаблон Model-View-Controller (MVC) чи View-Presenter-Model (MVP) для веб-програм та шаблон Model-View-Viewmodel (MVVM) для настільних програм,

проте дане використання не обмежене саме типом додатку, різні шаблони можна застосувати до різного роду програм .

Оскільки обсяг цих шаблонів стає ще меншим, ми іноді називаємо їх шаблонами (паттернами) дизайну замість архітектурних шаблонів. Прикладами таких є ін'єкція залежностей, синглтон та інші.

Для розроблення додатку було обрано наступні архітектурні шаблони (паттерни) згідно класифікаціях наведеної у рисунку 3.1:

- дизайнерський паттерн – View-Presenter-Model (MVP);
- тип застосунку – настільний Windows додаток (тобто застосунок для персонального комп'ютеру);
- стек технологій – .NET Framework з використання Windows Forms;
- архітектурний паттерн – багаторівнева архітектура.

3.2 .NET Framework та Windows Forms

Для розробки додатків Windows у платформі .NET доступні три основні додатки – Windows Forms, WPF та UWP.

Вони відрізняються не тільки інтерфейсом користувача, який можна створити за допомогою них, але й способом написання коду та тим, як створюються їхні інтерфейси.

Для розроблення додатку було обрано Windows Forms, оскільки дана технологія проста в освоєнні, а при наявності достатніх навичок створення дизайнерського інтерфейсу можна із меншими зусиллями створити досить зручний та простий для користувача інтерфейс.

Windows Forms використовує підхід швидкої та простої розробки. Програми складаються з кількох вікон, які називаються формами. За допомогою конструктора розробник може розміщувати елементи керування на формах і налаштовувати їх зовнішній вигляд і поведінку, змінюючи їх властивості в редакторі.

В результаті більшість програм Windows Forms мають дуже схожий вигляд, який часто називають сірим лінкором. Найкращий спосіб уникнути цього — використовувати спеціальні елементи керування від сторонніх розробників замість тих, які включені в структуру.

Оскільки результатом роботи конструктора є код, кожна форма має два окремих файли коду, щоб код, згенерований конструктором, не заважав написаному вручну коду. Часткові класи використовуються для того, щоб код з обох файлів компілювався в один клас.

Розповсюдження бізнес-логіки програми серед багатьох обробників подій у різних формах може ускладнити обслуговування програми, оскільки вона збільшується. Також складно писати для нього модульні тести, залишаючи тести інтерфейсу користувача єдиним варіантом для автоматичного тестування. Тести UI є більш крихкими та забирають більше часу на створення, ніж модульні тести.

Щоб уникнути цієї проблеми, можна використовувати шаблон проектування View-Presenter-Model (MVP). Цей підхід дозволяє перемістити більшу частину коду з форми (тобто з подання) до класу показчика, який відповідає за реакцію на події та оновлення представлення.

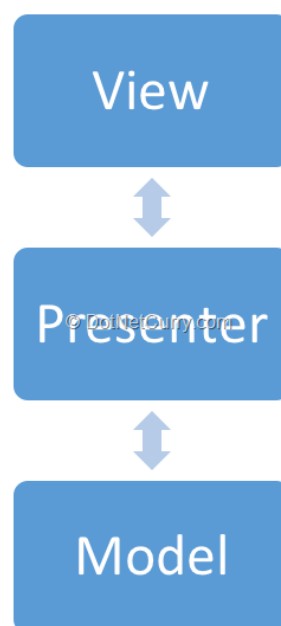


Рисунок 3.2 – Паттерн дизайнерської архітектури View-Presenter-Model

Повертаючись до додатку створеного у ході даної роботи, можна описати його архітектуру View-Presenter-Model у наступному вигляді:

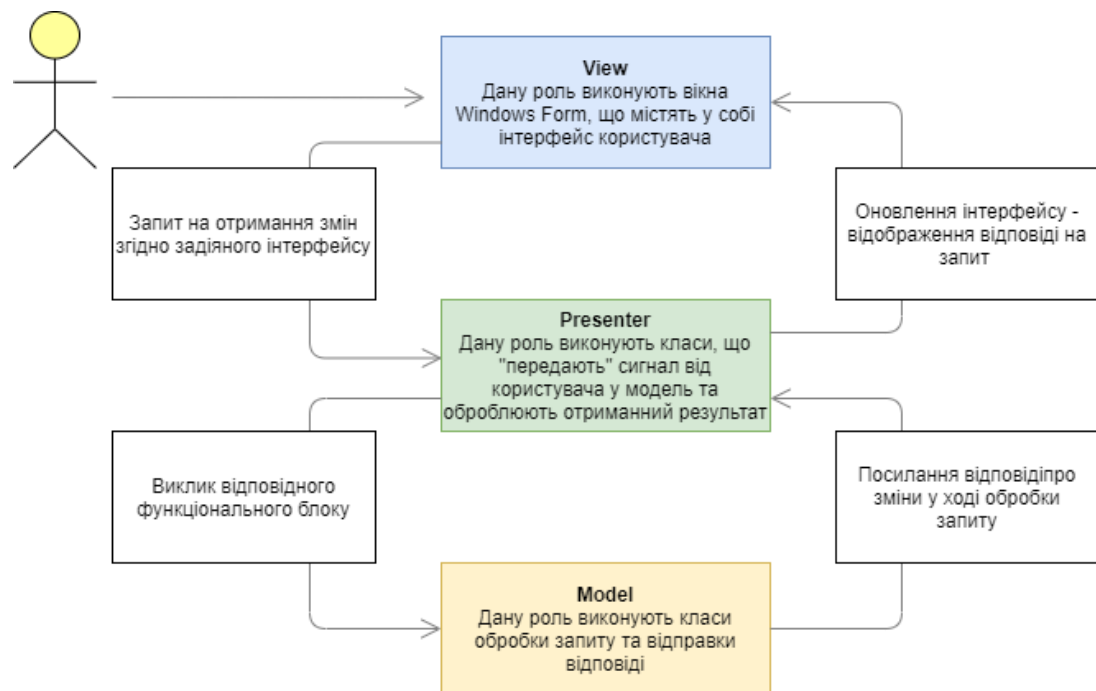


Рисунок 3.3 – Архітектура View-Presenter-Model для створеного у роботі додатку

У середовищі розробки Visual Studio для мови програмування C#, щоби була обрана для реалізації даного додатку було створено карту коду, яка допомагає швидше побачити (та зрозуміти), як написаний програмний код поєднується у єдине ціле, не читаючи при цьому окремі файли та рядки коду. За допомогою цих карт можна побачити організацію та взаємозв'язки у коді, включаючи його структуру та залежності. Переглянути дану мапу у скороченому та розгорнутому вигляді можна у додатку А.

Далі на рисунку 3.4 наведено загальну поведінку архітектури створеного у даній роботі додатку, що демонструє наявні модулі та їх взаємозв'язок між собою та базою даних. Як видно з рисунку, для реалізації буде використано інтерфейс користувача, деякі модулі обробки запитів з його боку, модуль обробки та видачі результату, а також оновлення інтерфейсу згідно виконаних раніше дій.

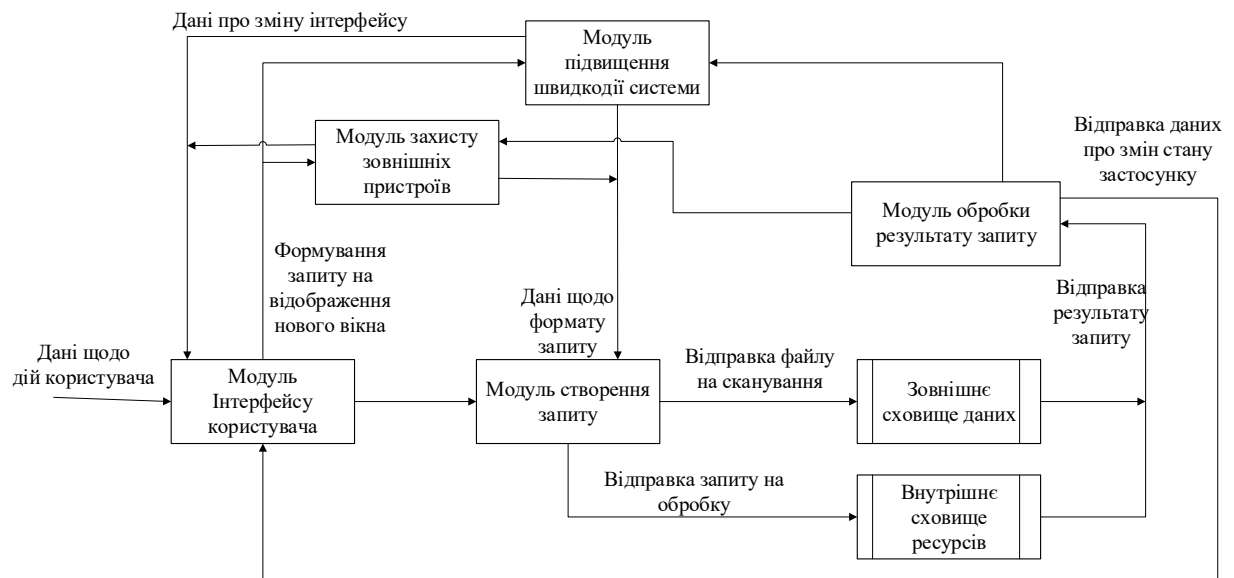


Рисунок 3.4 – Архітектура взаємодії програмних модулів

З ціллю легкості використання готового продукту, майже всі дії відбуватимуться у одному вікні, що посприятиме процесу розробки, проте полегшує дії безпосередньо користувачів. Взаємозв'язок отриманих у результаті розробки вікон можна виразити наступним чином:

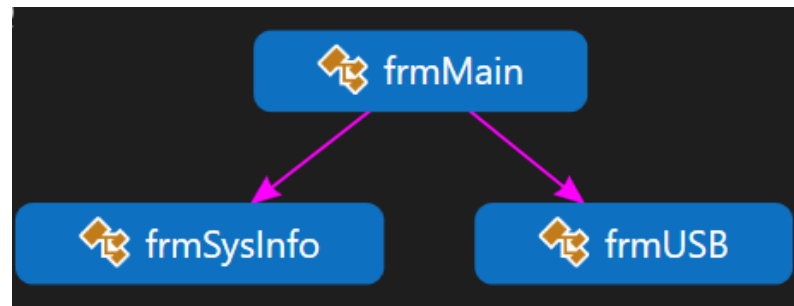


Рисунок 3.5 – Взаємозв'язок користувацьких вікон

Згідно отриманої архітектури, очевидно, що головна форма досить обширна та включає в себе багато елементів, що по чергово будуть показуватись чи приховуватись відповідно до дій користувача, тому надалі виникає необхідність навести блок-схеми опису головних процесів роботи (взаємодії) користувача та програмного продукту.

Для початку розглянемо блок схеми запуску сканування системи в цілому, коли запуск сканування відбувається безпосередньо від дій користувача:



Рисунок 3.6 – Блок схема ручного запуску сканування операційної системи

Надалі розглянемо схему опису процесу очистки системи від зайвого «сміття», що накопичується під час активної роботи системи, та незмінно буде рости, проте тільки після перезавантаження відбувається очистка операційної пам'яті технічного засобу, проте можна і за допомогою створеного додатку пришвидшити систему, закривши деякі фонові процеси та почистивши реєстр.



Рисунок 3.7 – Блок схема очистки операційної системи від файлів, що не використовуються

Надано далі блок схему опису процесу захисту зовнішніх носіїв, що були підключені до персонального технічного засобу від вірусних файлів. Оскільки при підключенні зовнішніх пристроїв через USB-порт, користувач не знає чи був він заражений чи ні – краще зробити швидку перевірку на найбільш очевидні віруси.

Слід зауважити, що передати та отримати вірус можна не тільки при підключенні флешки до власного комп'ютеру, а й шляхом підключення носія до «зараженого» засобу, а отже виникає необхідність не тільки робити сканування підключених пристроїв, а й захистити їх. У даній блок схемі наведено опис процесу, що підводить користувача до ввімкнення захисту на перевірку зовнішніх носіїв, а також їх захист.



Рисунок 3.8 – Блок схема ввімкнення захисту підключених зовнішніх носіїв від вірусів

Висновок до розділу 3

Для розробки додатку було використано мову с#, як одну з досить популярних у сьогоденні. Для реалізації інтерфейсу було використано .NET Windows Forms, як один із популярних способів створити простий та зручний у використанні користувацький інтерфейс. Згідно обраної архітектури View-Presenter-Model було реалізовано окремі класи, що відповідають за відображення, пересилання та обробку інформації відповідно до паттерну.

Багато рівність додатку була досягнута за допомогою відокремлення бізнес-логіки та інтерфейсу. У даному додатку було реалізовано обробник запитів користувача, а саме представник (Presenter), що згідно отриманого від інтерфейсу (View) сигналу (натискання кнопки) викликає відповідний бізнес-процес з моделі (Model) та отримуючи певний результат передає його назад до початкового місця, де користувач має змогу відслідкувати результати своїх дій.

4 ОГЛЯД РОЗРОБЛЕНОГО ДОДАТКУ

4.1 Загальний огляд можливостей

Розроблений програмний продукт надає змогу користувачу захистити свій персональний комп'ютер від вірусів. Даний продукт призначений для захисту операційної системи від різного виду вірусів та файлів, що можуть розповсюджуватись та сповільнювати, чи в деяких випадках навіть руйнувати цілісну роботу системи. У даному програмному засобі буде доступно наступний набір можливостей:

- 1) виявлення вірусів методом швидкого сканування системи;
- 2) виявлення вірусів метод повного сканування системи (займає більше часу, проте відбувається більше детальний аналіз файлової системи);
- 3) очищення непотрібних файлів та оптимізація системи;
- 4) користувач має змогу ввімкнути захист у режимі реального часу та сканування скачуваних файлів;
- 5) сканування усіх технічних засобів, що підключаються до комп'ютера через USB-порт та захистити свої USB пристрої.

Даний додаток несе у собі досить простий у використанні інтерфейс, проте на разі наявна тільки англійська оболонка, з часом це може бути виправлено за допомогою мовних пакетів. Надалі буде розглянута повна інструкція з поясненням кожної окремої функції даного програмного засобу із рисунками, для легшого розуміння.

Переваги даного додатку полягають й простоті його використання, інтерфейс досить простий та зрозумілий, дана програма не перевантажена багатьма «складними» для розуміння функціями, а також надає змогу захисту не лише персонального комп'ютера, а й деяких додатковий захист USB пристроїв. Також оптимізація системи є досить простою та зрозумілою у використанні, не кожний користувач має змогу зрозуміти як правильно почистити реєстр, та не кожен буде очищати корзину достатньо часто, щоб не

зберігати «тони» зайвої та забутої інформації. Серед недоліків даного додатку є прив'язка до однієї операційної системи, а саме Windows. Ще один не суттєвий недолік, що можна виправити з часом – це підтримка лише англійської мови, проте слова та інструкції написані максимально просто.

Для більш характерного порівняння наведемо коротку характеристику у табличному вигляді з основною інформацією щодо рішень, що були розглянуті як не комерційні аналоги та порівняємо їх з програмним продуктом, що було розроблено у даній роботі.

Таблиця 4.1 – Порівняльна характеристика антивірусних не комерційних програмних продуктів з розробленим власноруч програмним продуктом

Назва антивірусної програми/ Характеристика	Open Anti Virus	ClamAV	Moon Secure Antivirus	ClamWin	EG Free Antiviruse	Mr Shield
Сканер вірусних файлів	+	+	+	+	+	+
Наявність інтерфейсу	–	–	+	+	+	+
Наявність сучасного інтерфейсу	–	–	+	–	+	+
Перевірка файлів, під час завантаження з інтернету	+	+	–	–	–	+
Захист поштової скриньки	–	+	–	+	–	–

Продовження таблиці 4.1

Назва антивірусної програми/ Характеристика	Open Anti Virus	ClamAV	Moon Secure Antivirus	ClamWin	EG Free Antiviruse	Mr Shield
Наявність карантинного відсіку	—	+	—	+	+	—
Підтримка декількох мов інтерфейсу	—	—	—	—	—	—
Наявність документації для користувача	+	+	—	—	—	+
Наявність захисту зовнішніх пристроїв	—	—	—	—	—	+
Наявність оптимізації системи	—	—	+	—	+	+
Можливість переглянути системну інформацію	—	—	—	—	+	+

Переваги створеного продукту важко не помітити, адже наявність інтерфейсу у наші дні є ключовим фактором, проте і функціональні можливості значно ширші.

4.2 Огляд інтерфейсу користувача розробленого додатку

Під час запуску програмного застосунку, першим буде відкрито вікно:

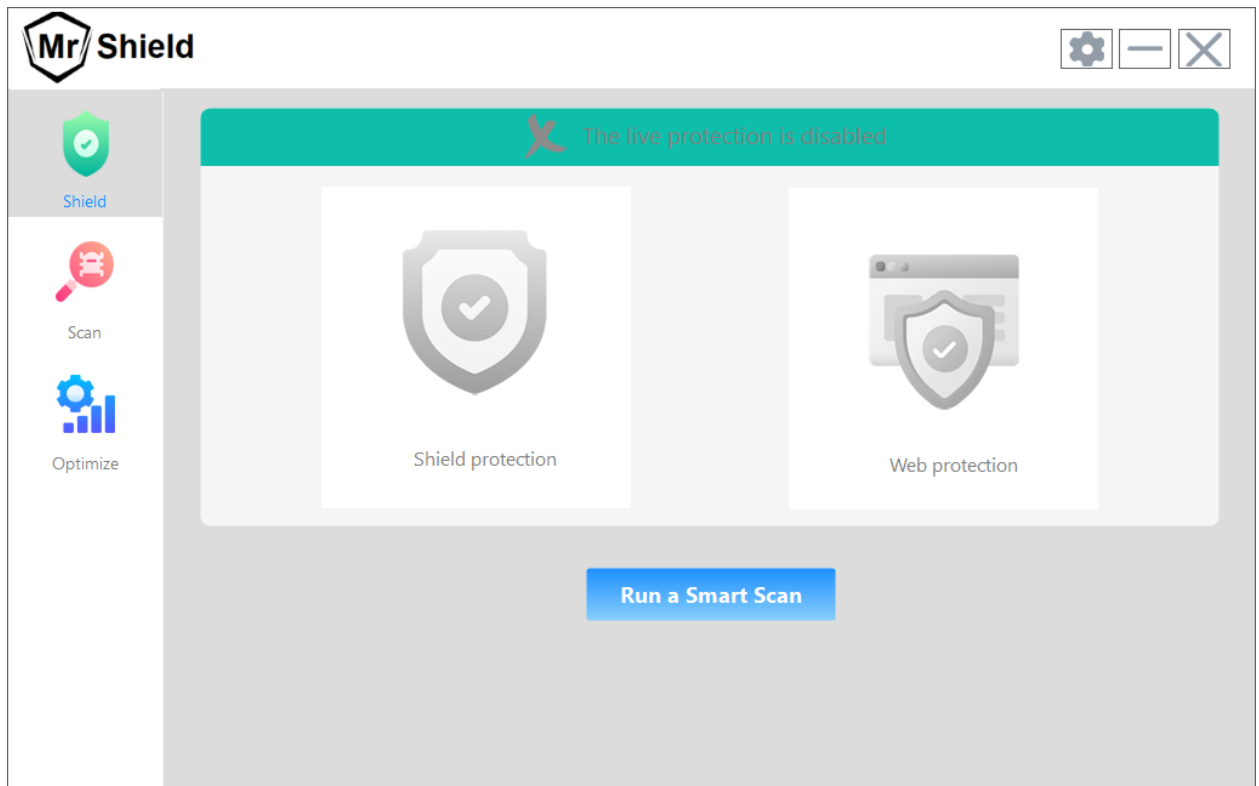


Рисунок 4.1 – Початкове вікно програмного застосунку з вимкненими функціями

На рисунку 4.1 можна побачити основні можливості даного додатку, що розподілені на 3 основні вкладки.

Перша Shield (з англійської «щит») – основне вікно застосунку, що відображає основні налаштування. По середині зверху відображено чи ввімкнено захист у режимі реального часу текстом, та великий знак «Shield protection» (з англійської «захист щитом») також є індикатором захисту. «Web protection» – індикатор захисту в інтернеті, тобто вказує користувачу чи ввімкнена перевірка завантажуваних файлів чи ні.

Друга вкладка Scan (з англійської «сканер») – вікно для роботи безпосередньо із можливостями сканування (буде розглянуто далі).

Остання вкладка Optimize (з англійської «оптимізація») – вікно для роботи із оптимізацією (пришвидшенням) операційної системи.

У верхньому правому кутку наявна кнопка налаштувань у вигляді шестерні, що відкриває меню з рисунку 4.2.

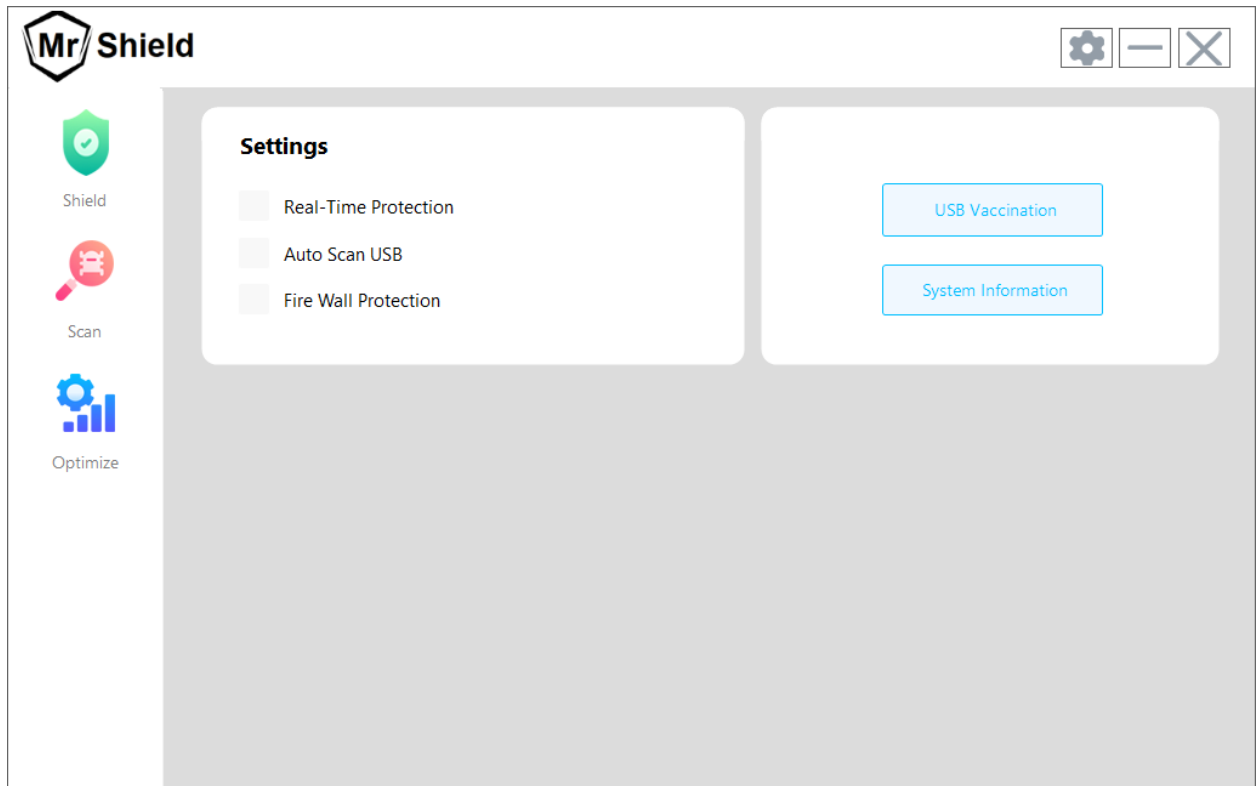


Рисунок 4.2 – Меню налаштувань

У даному меню доступні три види налаштувань:

- 1) «Real-Time Protection» (з англійської «захист у режимі реального часу») – режим, у якому сканування системи відбувається кожен раз, коли до неї надходить новий файл/документ/додаток тощо;
- 2) «Auto Scan USB» (з англійської «автосканування USB») – режим, у якому відбувається автоматичне сканування кожного девайсу, що підключений до персонального комп'ютеру через USB-порт;
- 3) «Fire Wall Protection» (з англійської «захисна стіна») – режим, що контролює файли завантажені з інтернету, та виявляє віруси на моменті їх завантаження.

Кнопка «USB Vaccination» (з англійської «Вакцинація USB») відкриває нове вікно, яке надасть можливість налаштувати додатковий захист від вірусів прихованих на зовнішніх носіях (таких як флешка), що підключаються до

персонального комп'ютеру через USB-порти. Віглядає дане вікно наступним чином:

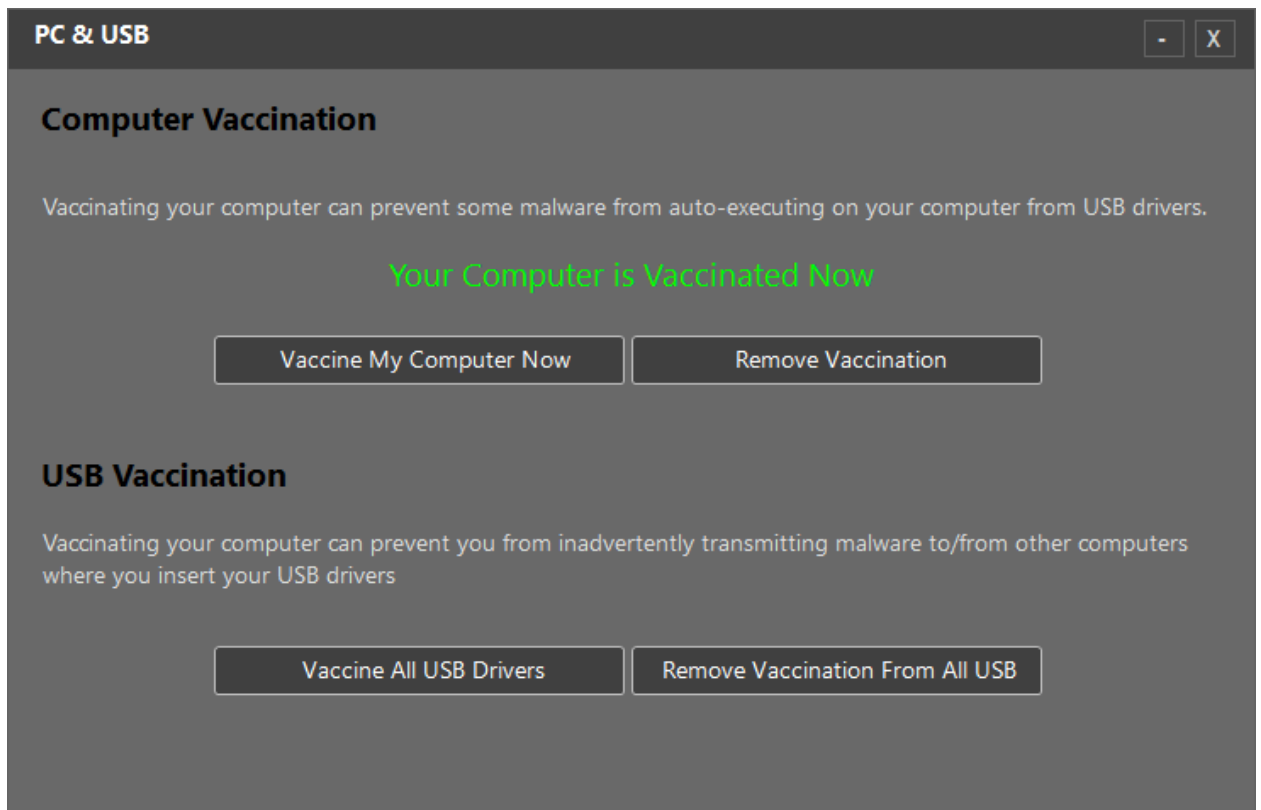


Рисунок 4.3 – Вікно для додаткового налаштування захисту ПК при підключенні через USB-порт

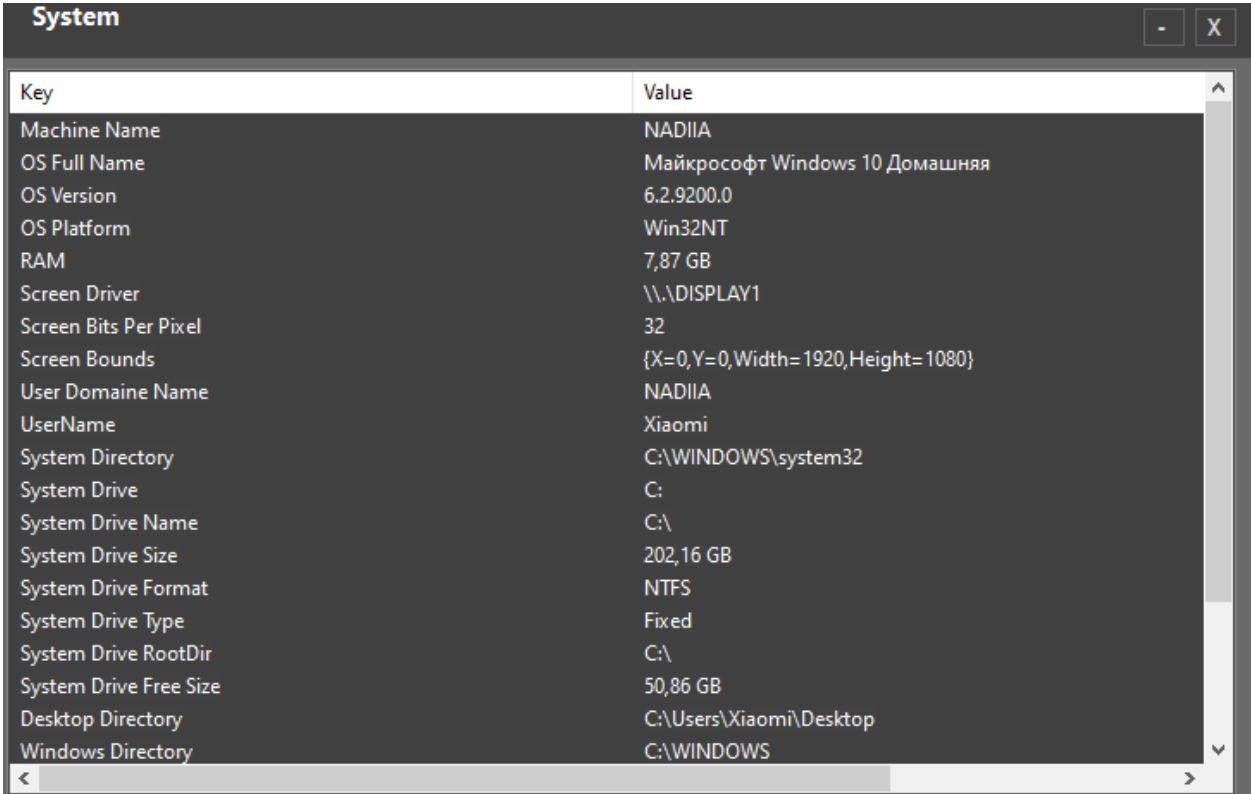
Кнопки з даного меню відповідають за наступні дії:

- 1) «Vaccinate My Computer Now» (з англійської «вакцінувати мій комп'ютер») – функція, що забезпечує додатковий захист від зовнішніх вірусів, що можуть заразити комп'ютер через флешку чи інший пристрій. Вакцинація може запобігти автоматичному виконанню деяких шкідливих програм з драйверів USB;
- 2) «Remove Vaccination» (з англійської «прибрати вакцинування») – функція, що вимикає захист вакцинування від зовнішніх пристроїв;
- 3) «Vaccinate All USB Drivers» (з англійської «вакцінувати всі USB драйвера») – функція, що призначена для захисту зовнішніх технічних засобів, таких як флешка чи портативний жорсткий диск. Дана методика може

запобігти ненавмисній передачі шкідливого програмного забезпечення на/з інших комп'ютерів, де використовуються пристрої через USB підключення;

4) «Remove Vaccination From All USB» (з англійської «прибрати вакцинування з усіх USB») – функція, що вимикає захист вакцинування у зовнішніх пристроїв.

Кнопка «System Information» (з англійської «Інформація про систему») відповідає за відображення інформації про операційну систему та загальна коротка інформація про операційну систему. Вікно виглядає наступним чином:



Key	Value
Machine Name	NADIIA
OS Full Name	Майкрософт Windows 10 Домашня
OS Version	6.2.9200.0
OS Platform	Win32NT
RAM	7,87 GB
Screen Driver	\\.\DISPLAY1
Screen Bits Per Pixel	32
Screen Bounds	{X=0,Y=0,Width=1920,Height=1080}
User Domain Name	NADIIA
UserName	Xiaomi
System Directory	C:\WINDOWS\system32
System Drive	C:
System Drive Name	C:\
System Drive Size	202,16 GB
System Drive Format	NTFS
System Drive Type	Fixed
System Drive RootDir	C:\
System Drive Free Size	50,86 GB
Desktop Directory	C:\Users\Xiaomi\Desktop
Windows Directory	C:\WINDOWS

Рисунок 4.4 – Вікно з інформацією про операційну систему та персональний комп'ютер

Коли усі налаштування відкриті початкове меню змінює свій вигляд на наступний – див. Рисунок 4.5, а в процесі змін налаштування буде з'являтися сисемне вікно (див. Рисунок 4.6), що описує зміни внесені у програмний застосунок.

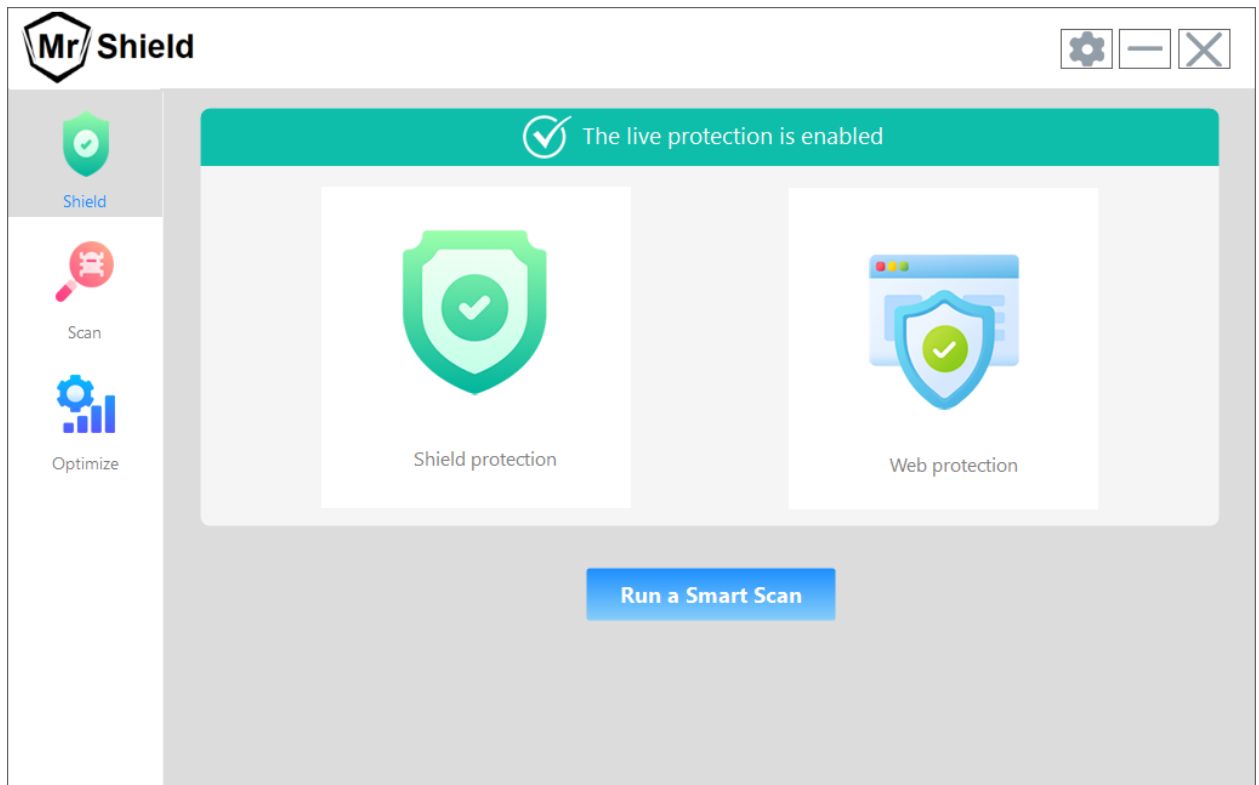


Рисунок 4.5 – Початкове вікно програмного застосунку з увімкненими функціями

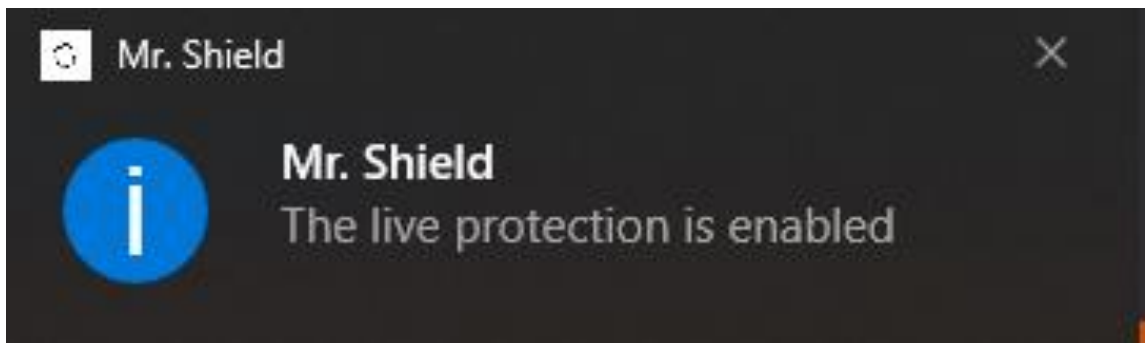


Рисунок 4.6 – Інформативне вікно щодо змін налаштувань

Наступна вкладка у боковому меню носить назву «Scan» (з англійської «Сканування») та призначена вона для відображення головного функціоналу даного програмного засобу, а саме сканування (пошуку) вірусів на персональному технічному засобі.

Розглянемо призначення кожної кнопки (див. рисунок 4.7):

1) «Run a Smart Scan» (з англійської «запустити розумне сканування») – функція, що робить порівняно швидке сканування та виявляє

більшу частину вірусів. Така сама кнопка є і на початковому вікні додатку (див. рисунок 4.1);

2) «Quick Scan» (з англійської «швидке сканування») – функція, що робить найшвидше сканування, проте воно може пропускати деякі файли, що не є рекомендованим, проте можна користуватись, коли проміжок часу досить обмежений (основні віруси будуть знайдені);

3) «Targeted Scan» (з англійської «цільове сканування») – функція, що робить цільове (тобто направлене на певну папку) сканування файлів. При використанні даного методу, буде необхідно явно вказати папку, яку треба перевірити. Вікно вибору папки можна побачити на рисунку 4.8;

4) «Full scan» (з англійської «повне сканування») – функція, що робить повноцінну перевірку усіх папок, файлів та програм операційної системи на наявність того чи іншого виду вірусів.

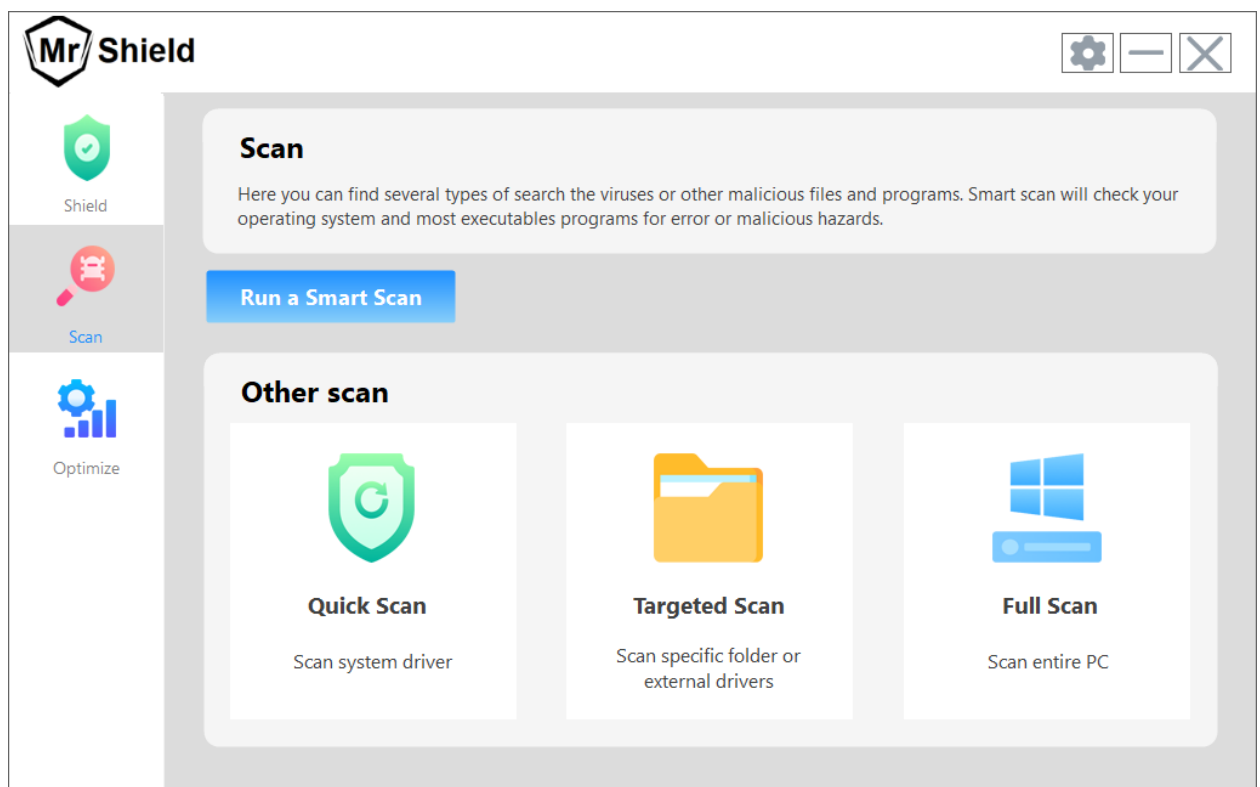


Рисунок 4.7 – Вікно сканування вірусів у програмному застосунку

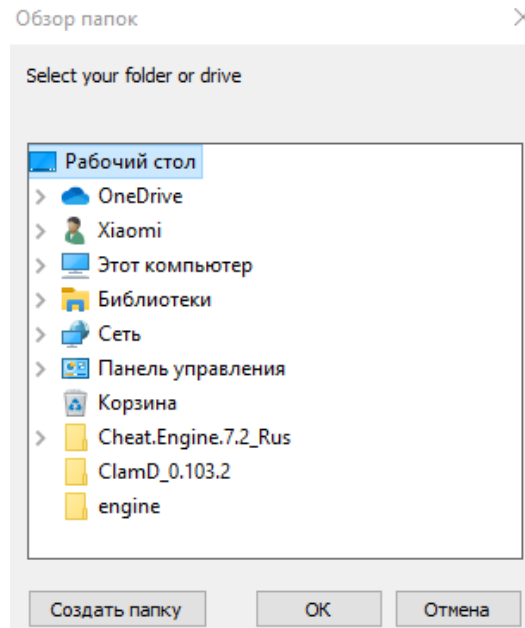


Рисунок 4.8 – Вікно вибору папки для цільового сканування

Після запуску сканування з'явиться вікно з відображенням індикатором виконання (прогресу) сканування, у відсотковому відношенні буде відображено кількість відсканованих файлів, відносно усіх існуючих у системі (див. рисунок 4.9).

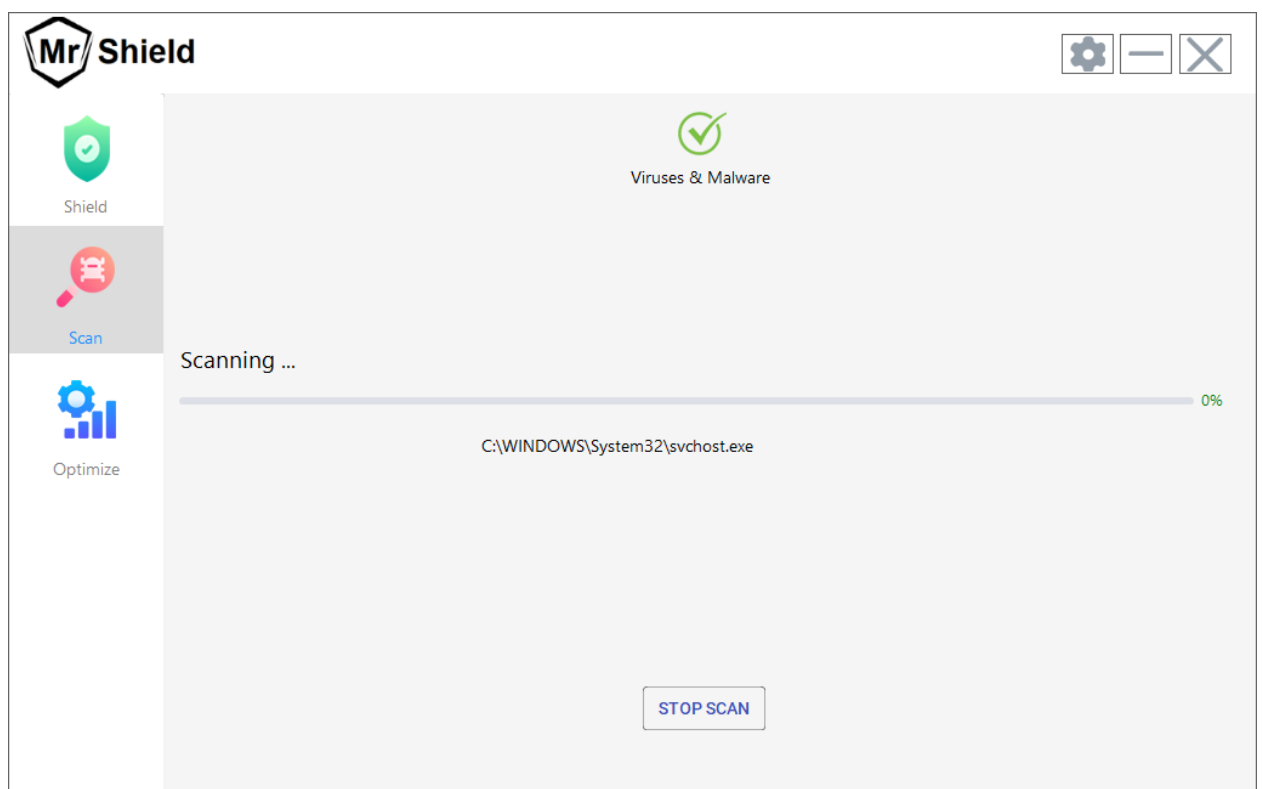


Рисунок 4.9 – Вікно індикатору виконання сканування

В будь-який момент часу можна перервати сканування, натиснувши відповідну кнопку «Stop scan» (з англійської «зупинити сканування»). Рисунок 4.10 відображає вікно, що з'являється після зупинки сканування. Після натискання кнопки «ОК» відбувається повернення на початкове вікно.

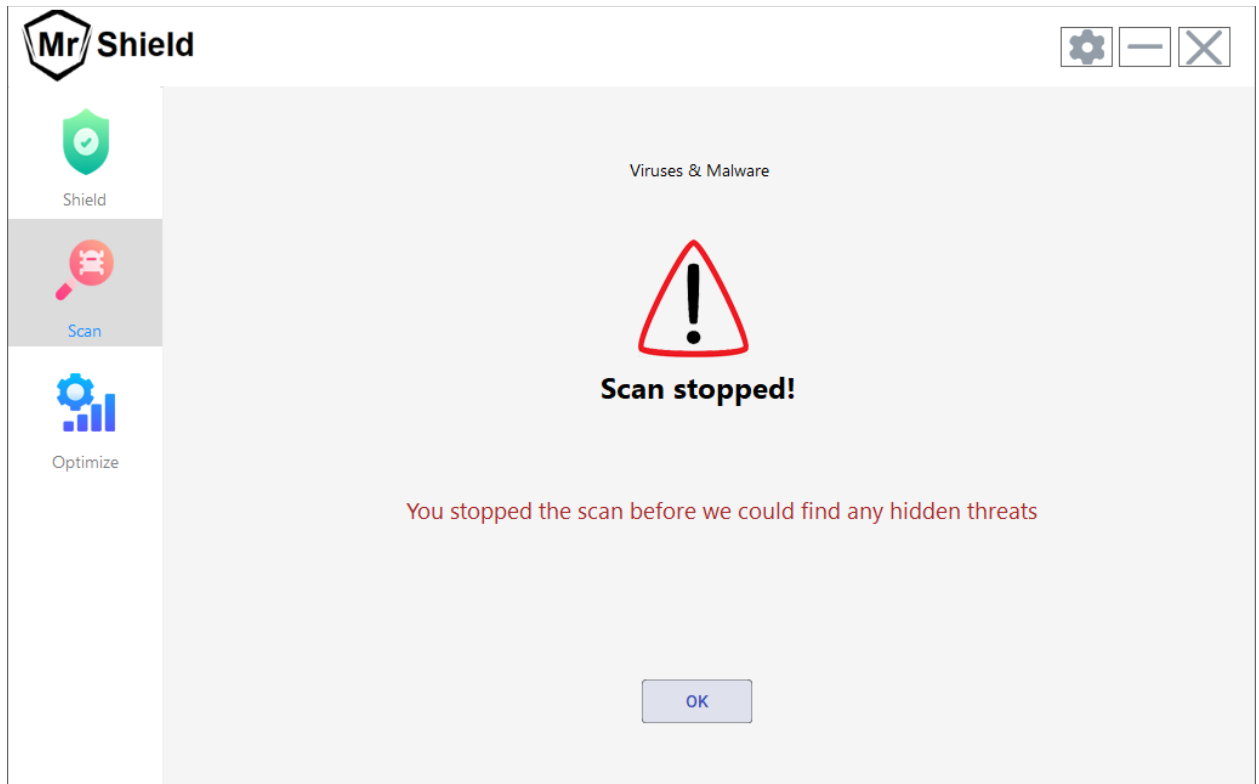


Рисунок 4.10 – Вікно зупинки виконання сканування

Коли сканування завершено та система повністю захищена, вікно про успішне завершення сканування буде містити повідомлення «Ваш комп'ютер тепер у безпеці» (англійською «Your computer is now safer») після чого можна натиснути кнопку «Done» (у перекладі з англійської «завершення») та повернутись до головного меню. У разі виявлення вірусів спочатку відображається вікно з кількістю знайдених вірусів (див. рисунок 4.12), після чого можна продовжити натиснувши кнопку «Next» та обрати опцію видалення вірусних файлів за допомогою кнопки «Delete» чи відмінити видалення кнопкою «Cancel» (див. рисунок 4.13). Після видалення вірусних файлів відкриється вікно з рисунку 4.13), що повідомить користувача про

успішне видалення вірусів. Після натискання кнопки «Done» відбудеться повернення до головного меню.

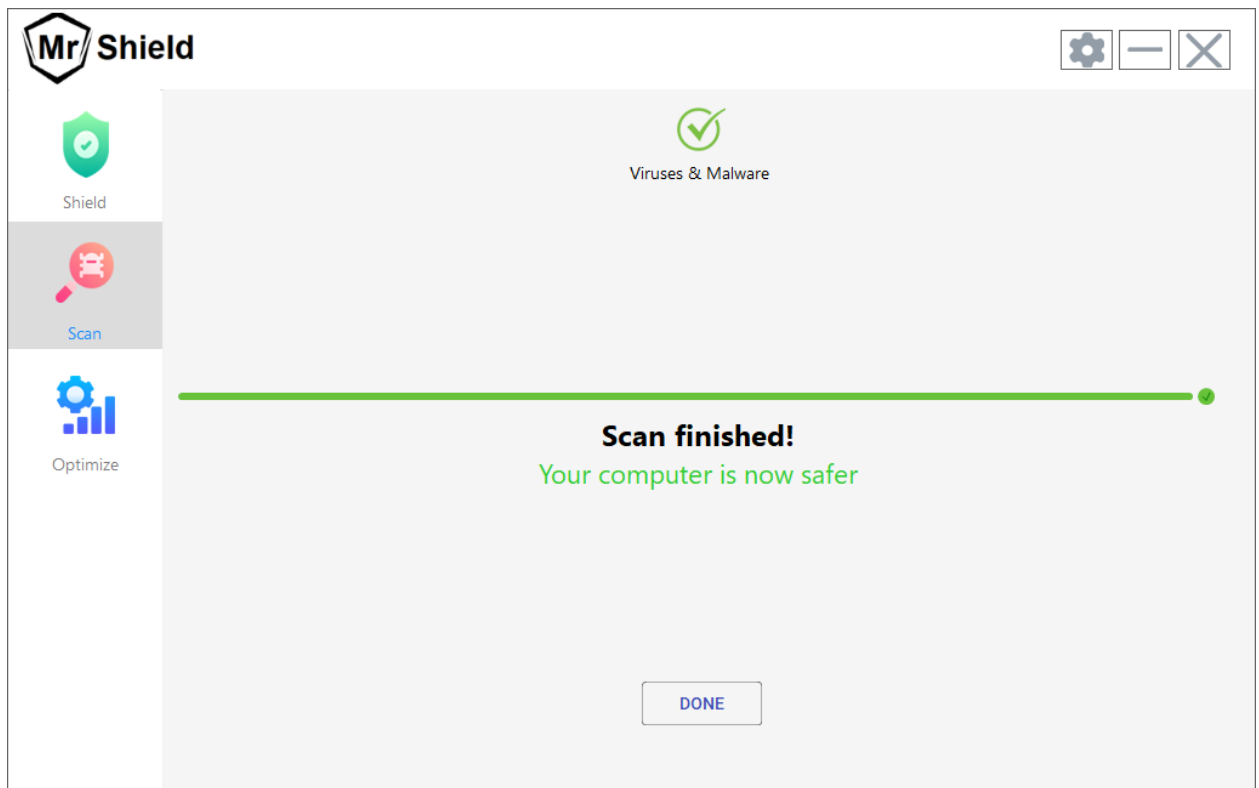


Рисунок 4.11 – Вікно з інформацією про відсутність вірусів

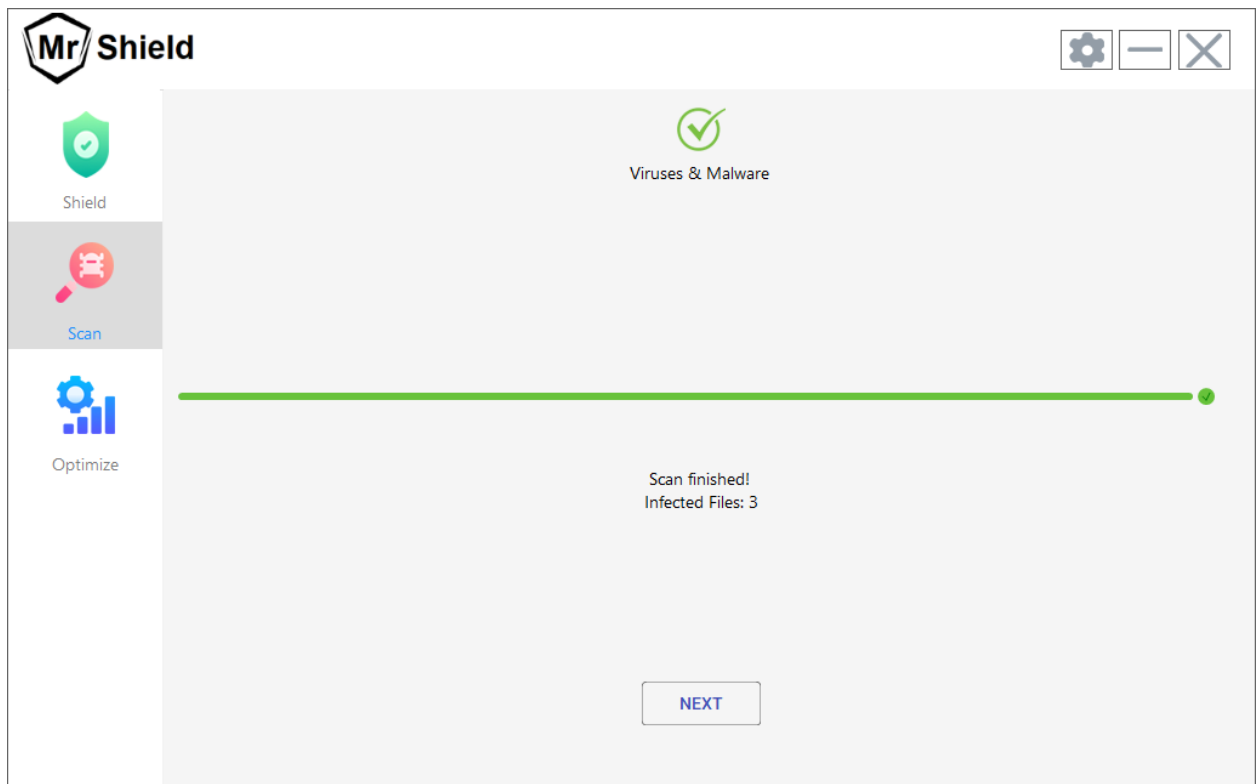


Рисунок 4.12 – Вікно інформацією про наявність вірусів

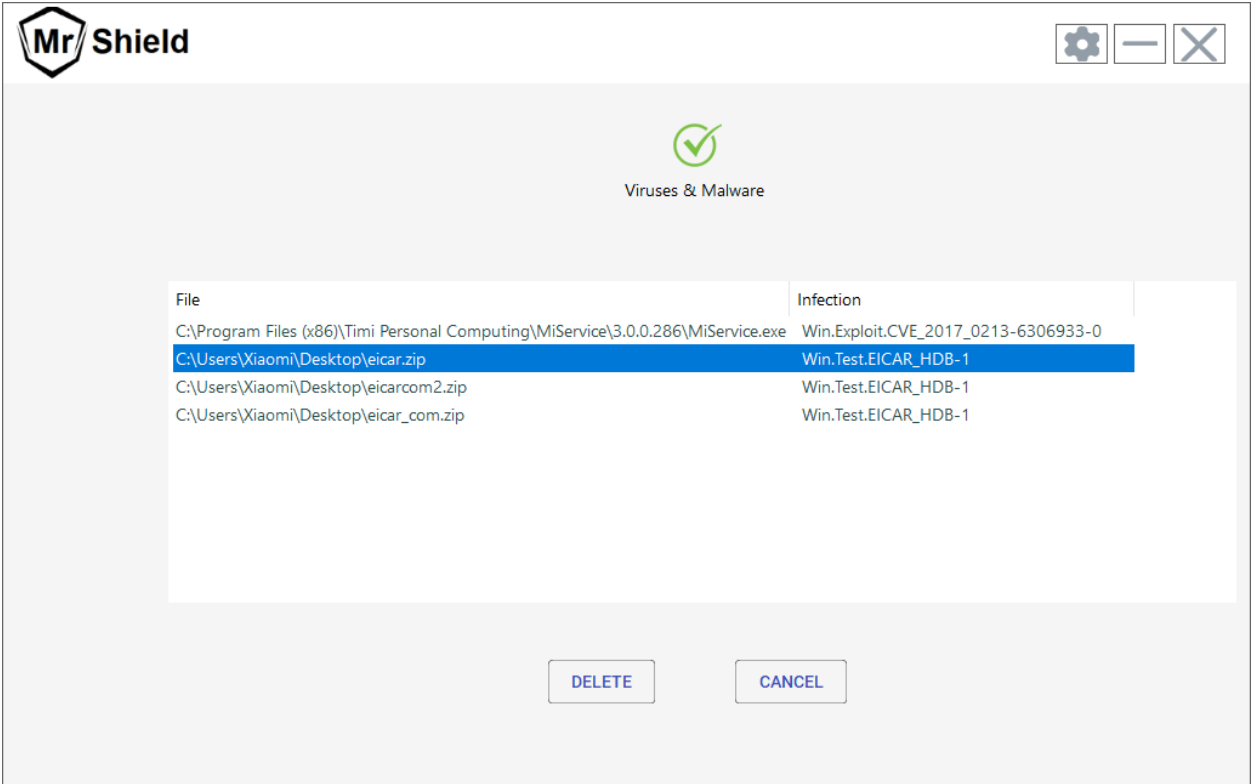


Рисунок 4.13 – Вікно з виявленими у процесі сканування вірусами та кнопкою для їх видалення чи відміни

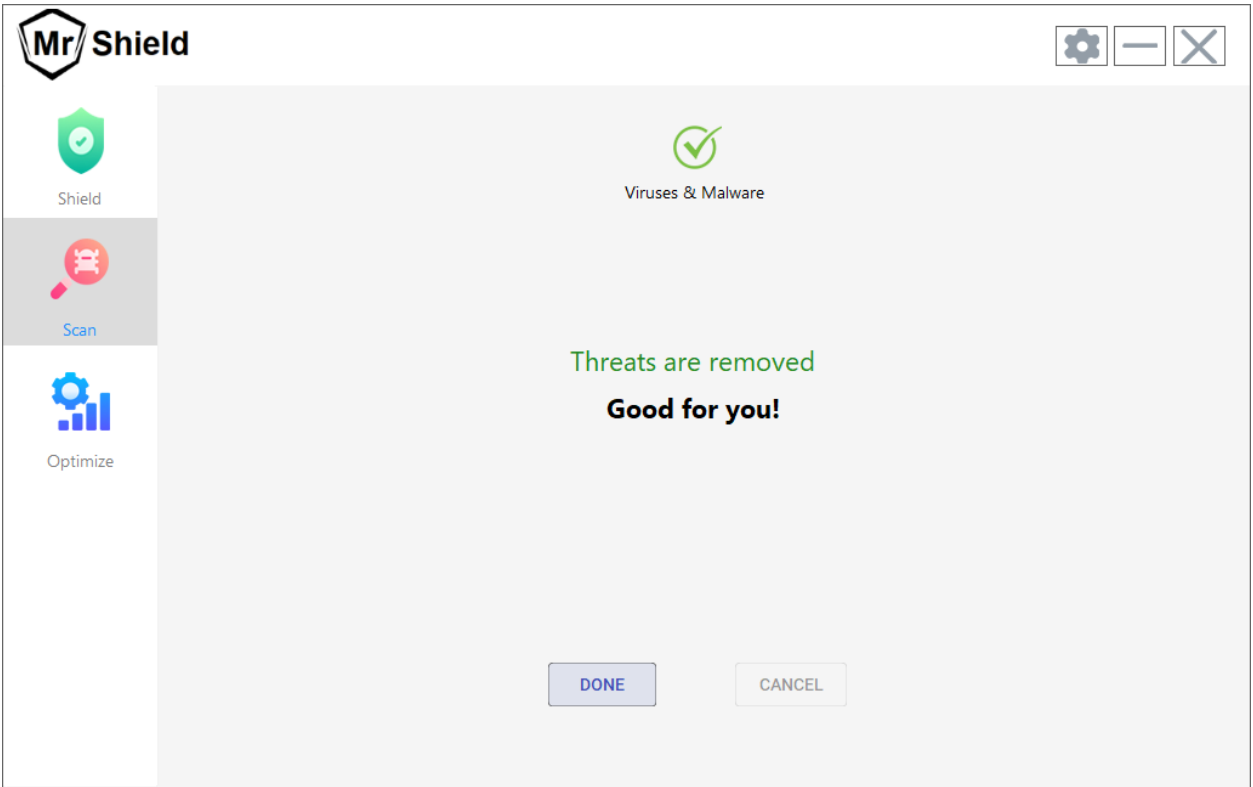


Рисунок 4.15 – Вікно з інформацією про видалення вірусів

Остання вкладка у боковому меню носить назву «Optimize» (з англійської «Оптимізація») та призначена вона для відображення можливостей покращення роботи операційної систем, а саме пришвидшення деяких процесів. Виглядає дане вікно наступним чином:

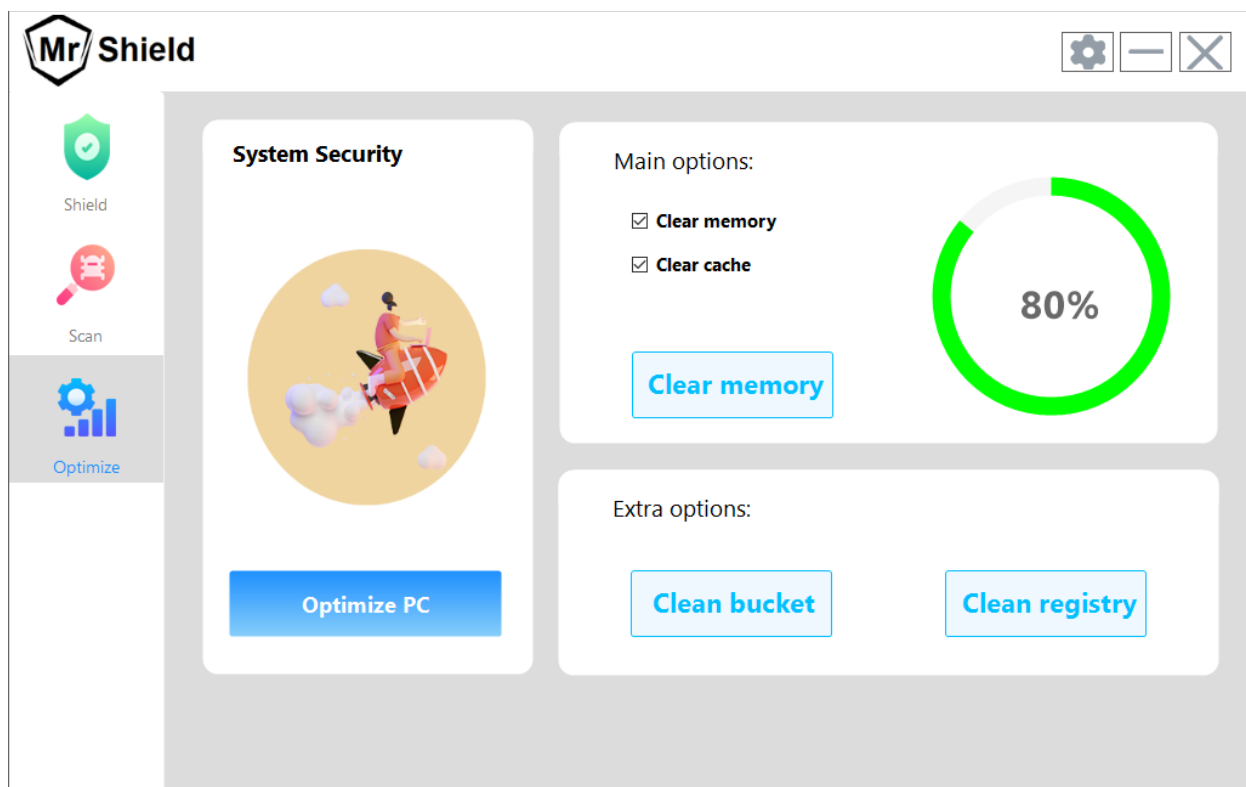


Рисунок 4.16 – Вікно з оптимізацією персонального комп'ютера
Доступні функціональні блоки (кнопки):

- 1) «Clean Memory» (з англійської «очистка пам'яті») – функція, що закриває рідковикористовувані процеси, що сповільнюють роботу системи;
- 2) «Clean Bucket» (з англійської «очистка корзини») – функція, що видаляє всі файли з корзини. Досить часто необхідна кнопка, адже не всі користувачі власноруч очищають корзину, а файли що потрапляють туди після видалення продовжують навантажувати систему та займати місце;
- 3) «Clean registry» (з англійської «очистка реєстру») – функція, що очищує реєстр. Не рекомендується робити це вручну, адже можна через незнання видалити щось необхідне операційній системі із реєстру, проте за

допомогою даної програми видалиться лише «сміття», що сповільнює роботу ПК;

4) «Optimize PC» (з англійської «оптимізація ПК») – функція, що виконує одразу всі три наведені раніше дії за одне натискання.

У разі успішного виконання оптимізації буде виведене відповідне інформативне вікно. Див. Рисунок 4.17 – пункти 1-3, що наведені вище відповідають повідомленнями з ліва на право.

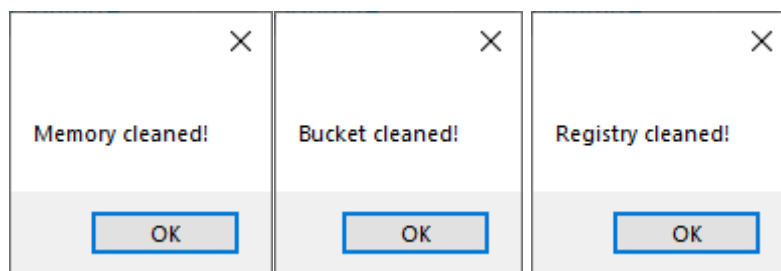


Рисунок 4.17 – Інформативні повідомлення про завершення процесів оптимізації

Висновок до розділу 4

Розроблений програмний продукт, що виходить на ринок мусить мати документацію, що пояснює можливості програмного продукту кінцевому користувачу, а отже постає питання, того щоб усі можливі форми його використання були чітко описані та за модливістю мали ілюстративне відображення (для зв'язки інтерфейсу та опису інструкцій).

Саме тому, у даному розділі було виконано опис повної користувацької інструкції щодо виконаних у додатку можливостей з наведенням усіх вікон програми (тобто інтерфейсу) в форматі рисунків (скрін шотів активної роботи застосунку). За допомогою інструкції користувач може повноцінно використовувати функціональні можливості додатку, а також у разі виникнення складностей має змогу переглянути «підказки» щодо виниклих запитань.

5 РОЗРОБКА СТАРТАП ПРОЕКТУ

5.1 Опис ідеї проекту

Надалі полягає необхідність розглянути та розписати більш детально деякі основні аспекти, пов'язані з ідеєю даного проекту, а саме більш детальні пояснення вигоди кінцевого користувача від безпосереднього використання розробленого додатку. Для опису було використано табличний формат, що описує зміст, напрямки застосування та вигоди користувача (див таблицю 5.1).

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди користувача
Розробка програмного забезпечення для розпізнавання шкідливих файлів зі зручним інтерфейсом. Додатково розроблена можливість захисту USB пристроїв та покращення швидкодії системи.	Виявлення шкідливих файлів у операційній системі та подальше їх видалення	Використання даного програмного продукту дозволяє провести перевірку операційної системи на наявність у ній шкідливих файлів.
	Підвищення швидкості роботи системи.	Можливість провести перевірку операційної системи та підвищити швидкість роботи.
	Захист зовнішніх носіїв від ураження.	Можливість провести перевірку зовнішнього пристрою та встановити на нього додатковий захист від ураження шкідливими файлами.

На сьогоднішній момент існує багато аналогів та конкурентних продуктів з функцією захисту персонального комп'ютер від вірусів (шкідливих файлів), проте порівнювати усі одразу буде не сильно інформативно, а отже для порівняння використаємо один комерційний та один не комерційний додатки. Наведемо порівняння у таблиці 5.2.

Таблиця 5.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			Слабкі (W), нейтральні(N) та сильні(S) сторони		
		Мій проект	NOD32	Open Anti Virus	W	N	S
1.	Сканер вірусних файлів	+	+	+			+
2.	Наявність сучасного інтерфейсу	+	+	-			+
3.	Ведення історій дій (лог файл)	+	+	+		+	
4.	Перевірка файлів, під час завантаження з інтернету	+	+	+		+	
5.	Наявність карантинного відсіку	+	+	+		+	
6.	Наявність документації для користувача	+	+	-			+
7.	Наявність функцій оптимізації ПК	+	+	-			

5.2 Технологічний аудит ідея проекту

У цьому пункті буде описано технології, що необхідні для реалізації розробленого програмного продукту та їх доступність.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

Ідея проекту	Технологія реалізації	Наявність технології	Доступність технології
Розробка програмного забезпечення для розпізнавання шкідливих файлів зі зручним інтерфейсом. Додатково розроблена можливість захисту USB пристроїв та покращення швидкодії системи.	Алгоритм виявлення шкідливих файлів (шляхом порівняння хешу файлу з базою даних)	Технологія наявна	Технологія доступна
	Алгоритм пришвидшення роботи системи	Технологія наявна	Технологія доступна
	Бібліотеки .NET Framework для створення додатку	Технологія наявна	Технологія доступна
	Використання Windows Forms для створення інтерфейсу	Технологія наявна	Технологія доступна

Після відповідного аналізу, результати якого наведені у таблиці 5.3 можна зробити висновок щодо проекту, що розробляється в рамках дослідження, а саме дана розробка не потребує додаткових зусиль на створення нових унікальних технологій, готовий проект буде покращувати існуючі рішення з використанням вже відомих технологій.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Метою даної частини роботи буде аналіз та визначення ринкових можливостей для етапу впровадження готового програмного продукту в реальне середовище, а також підрахунок та виявлення можливих ризиків (чи загроз) на етапі вводу в масову експлуатацію.

Таблиця 5.4 – Попередня характеристика ринку

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних конкурентів (серед не комерційних проектів), од	5
2.	Динаміка ринку (якісна оцінка)	Потреба зростає
3.	Наявність обмежень (зупиняючих факторів) для входу – необхідно задати характер факторів	Потреба відсутня
4.	Специфічні вимоги (потреби) до стандартизації та (чи) сертифікації	Потреба відсутня
5.	Середня норма рентабельності в галузі, %	Дані відсутні

За результатами аналізу наведеного у таблиці 5.5 впливає висновок про те, що згідно наданого оцінювання, ринок є сприятливим для успішного входження, оскільки немає додаткового навантаження на підготовку сертифікацій та ін.

Наступним пунктом є розряд можливих споживачів (зацікавлених у використанні програмного продукту облич'я), тому наведемо для порівняння таблицю 5.5.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Потреба у захисті персонального технічного засобу від шкідливих файлів	Фізичні особи та підприємства, які хочуть захистити власні технічні засоби від вірусних файлів.	Кількість користувачів, що будуть використовувати продукт.	Наявність зручного інтерфейсу та виявлення шкідливих файлів до моменту ураження ПК.
2.	Встановлення та можливість використання без сторонньої допомоги	Фізичні особи та підприємства, які хочуть захистити свій технічний засіб від вірусних файлів без додаткової сторонньої допомоги.	Кількість користувачів	Наявність інструкції з використання.

Необхідно також зазначити фактори, що потенційно можуть перешкоджати ринковому впровадженню проекту, тому їх буде відображено у таблиці 5.6., а більш детально розглянуто конкуренцію ринку за ступенем наведено у таблиці 5.7. За характером Нецінова

Таблиця 5.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкурентоспроможність	Виявлення на ринку нових конкурентних продуктів	Розширення функціоналу додатку (створення нових можливостей для користувача). Збільшення масштабів рекламної кампанії.
2.	Створення нових комп'ютерних вірусів, раніше не відомих у світі	Створення нових вірусів, що раніше були не відомі світу	Оновлення бази даних хешами нових вірусів. Розповсюдження оновленої бази користувачам з оновленою версією додатку.

Таблиця 5.7 – Ступеневий аналіз конкуренції на ринку

№ п/п	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1.	Вказати тип конкуренції: Чиста	Існує досить велика кількість додатків зі схожим функціоналом	Інтерфейс має бути максимально зручним та привабливим, завдання, що виконує програма – унікальними та повністю робочі.

Продовження таблиці 5.7

№ п/п	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
2.	За рівнем конкурентної боротьби: Всесвітній	Направленість додатку покриває потреби, що можуть знаходитись у різних країнах світу	Орієнтованість на потреби людини, не залежно від місця проживання
3.	Конкуренція за видами продукції: Товарно-видова	Існують програмні додатки, що включають в себе схожі можливості	Створення нових (унікальних) можливостей для користувачів
4.	За характером: Нецінова	Існують як комерційні (платні) так і не комерційні (безкоштовні) аналоги	Мають бути створені унікальні функції, що привернуть до себе аудиторію користувачів

Базовою стратегією було обрано стратегію спеціалізації. На випадок її провалу в подальшому буде використано стратегію диференціації. У наступній таблиці розглянуто стратегію конкурентної поведінки на ринку.

Таблиця 5.8 – Стратегія конкурентної поведінки на ринку

Проект є першопроходцем на ринку	Компанія буде шукати нових користувачів та (або) переманювати конкурентів	Компанія буде копіювати функціональні можливості конкурентів	Стратегія конкурентної поведінки
Ні	Нових	Так, проте надаючи перевагу власному розвитку та покращенню раніше відомих технологій	Нішева

Стратегія позиціонування продукту буде виглядати наступним чином:

- 1) вимоги до товару цільової аудиторії – Зручний інтерфейс. Використання функцій за власними потребам;
- 2) базова стратегія розвитку – Стратегія охоплення декількох сегментів ринку й розроблення для кожного з них окремого комплексу маркетингу;
- 3) ключові конкурентоспроможні позиції власного стартап-проекту – зручний та зрозумілий інтерфейс. Наявність інструкцій користувача;
- 4) вибір асоціацій, які сформують комплексну позицію проекту – зручність, надійність.

Висновок до розділу 5

В даному розділі було сформовано та описано ідею створення та подальшого розвитку стартап проекту. Було проведено дослідження можливості реалізації проекту з технічної та маркетингової точки зору, оцінено план дій план з урахуванням ризиків, обрано стратегії розвитку продукту на ринку. Так як прямих конкурентів, серед не комерційних проектів досить мало, реалізувати та розвивати в подальшому даний стартап проект доцільно.

ВИСНОВКИ

Після магістерської роботи було проведено аналіз сучасного програмного забезпечення щодо виявлення шкідливих (зловмисних) файлів, зроблена оцінка аналогів на комерційному та не комерційному ринку, що наявні у сьогоденні та мають здатність до виявлення та подальшого вилучення шкідливих файлів з операційної системи персонального комп'ютеру. Було наведено порівняльні характеристики та описано переваги та недоліки кожного з порівнюваних продуктів.

Було проаналізовано також методи (способи) виявлення, зчитування та порівняння зловмисних файлів. Було обрано мову програмування та необхідні технології для реалізації необхідного функціоналу майбутнього додатку.

Під час розробки додатку, було використано технології для простої (з точки розробника) та швидкої реалізації користувацького інтерфейсу, що не був перевантажений і несе у собі досить простий набір кнопок та вкладок для покриття заданих під час аналізу можливостей програмного продукту.

Дана робота включає в себе не тільки частину аналізу та розробки, а також й інструкцію користувача, що описує функціональні можливості додатку з точки зору використання.

Магістерська робота також включає в себе підрозділ стартапу, що надає змогу пояснити стратегію майбутнього розвитку програмного забезпечення. Оцінено можливі ризики, з урахуванням конкурентного ринку та можливі способи розвитку додатку у подальшому для підтримки його у сфері конкурентів.

Було наведено порівняння зацікавлених груп користувачів, що можуть у майбутньому використовувати розроблений продукт як у власних цілях, так і у межах корпоративної необхідності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1) Касперский Е.В. Компьютерные вирусы: что это такое и как с ними бороться, Москва: СК Пресс. 1998. 288 с.
- 2) Информатика. Базовый курс. 2-е издание / Под ред. С.В. Симоновича. СПб.: Питер, 2006. 640 с.
- 3) S. Jana and V. Shmatikov. Abusing File processing in Malware Detectors for Fun and Profit, San Francisco, CA, U.S.A, May, 2012, p. 22-50.
- 4) Shafiq, M. Zubair and Tabish, S. Momina and Mirza, Fauzan and Farooq, Muddassar. PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime, Saint-Malo, France, 2009, p. 121-141.
- 5) Компьютерная безопасность. Защита информации. URL: https://www.bsmu.by/downloads/kafedri/k_fiziki/2015-1/komp_bez.pdf (дата звернення: 20.10.2021)
- 6) MD5 Algorithm // Educba. URL: <https://www.educba.com/md5-algorithm/> (дата звернення: 27.10.2021)
- 7) SHA Algorithm // Educba. URL: <https://www.educba.com/sha-algorithm/> (дата звернення: 15.10.2021)
- 8) Checksum Algorithm // Ibm Documentation. URL: <https://www.ibm.com/docs/en/zos/2.4.0?topic=calculations-checksum-algorithm> (дата звернення: 01.11.2021)
- 9) Розроблення стартап-проєкту Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / за заг. ред. О.А. Гавриша. Київ: НТУУ «КПІ», 2016. 28 с.

ДОДАТКИ

ДОДАТОК А

Карта коду програми

ДОДАТОК Б
Лістинг програми

```

private void btnScan_Click(object sender, EventArgs e) // Quick Scan
{
    scanType = 1;
    loc_to_search = System.IO.Path.GetPathRoot(Environment.SystemDirectory);
    ShowPanel(pnlFormScan);
    this.BackgroundImage = null;
    this.BackColor = Color.WhiteSmoke;
    search = new Thread(new ThreadStart(ScanFolder));
    search.Start();
}

private void btnTargetedScan_Click(object sender, EventArgs e) // Folder Scan
{
    scanType = 2;
    folderBrowserDialog1.Description = "Select your folder or drive";
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        loc_to_search = folderBrowserDialog1.SelectedPath;
        ShowPanel(pnlFormScan);
        this.BackgroundImage = null;
        this.BackColor = Color.WhiteSmoke;
        search = new Thread(new ThreadStart(ScanFolder));
        search.Start();
    }
}

private void btnFullScan_Click(object sender, EventArgs e)
{
    scanType = 3;
    ShowPanel(pnlFormScan);
    this.BackgroundImage = null;
    this.BackColor = Color.WhiteSmoke;
    search = new Thread(new ThreadStart(ScanFolder));
    search.Start();
}

private void ScanFolder()
{
    ArrayList fileList = new ArrayList();
    // if it's a quick scan, first check all the running processes
    if (scanType == 1)
    {
        foreach (string proc in GetAllRunningProcesses())
        {
            if (System.IO.File.Exists(proc))
            {
                try
                {
                    var client = new MrShieldScanClient("localhost", 3310);
                    var scanResult = client.ScanFileOnServer(proc);
                    MrShieldScanResults scanfileType = client.GetFileType(scanResult.Result);
                    Invoke(new Action(delegate
                    {
                        try
                        {
                            curr_File.Text = proc;
                        }
                        catch { }
                    }));
                    switch (scanfileType)
                    {
                        case MrShieldScanResults.VirusDetected:
                            infected++;

```

```

        AddToResult(proc, scanResult.InfectedFiles.First().VirusName);
        break;
    }
}
catch (Exception ex)
{
    frmMain.PushLog(ex.Message);
}
}
}
Invoke(new Action(delegate
{
    try
    {
        curr_File.Text = "...";
    }
    catch { }
}));
}

if (scanType == 0)    // smart scan
{
    foreach (DriveInfo drive in DriveInfo.GetDrives())
    {
        try
        {
            fileList.AddRange(getFiles(drive.Name, smart_ext, SearchOption.TopDirectoryOnly));
        }
        catch { }
    }
    files = (string[])fileList.ToArray(typeof(string));
}
if (scanType == 1)    // quick scan
{
    files = getFiles(loc_to_search, wildcard, SearchOption.AllDirectories);
}
if (scanType == 2)    // targeted scan
{
    files = getFiles(loc_to_search, wildcard, SearchOption.AllDirectories);
}
if (scanType == 3)    // full scan
{
    foreach (DriveInfo drive in DriveInfo.GetDrives())
    {
        fileList.AddRange(getFiles(drive.Name, smart_ext, SearchOption.AllDirectories));
    }
    files = (string[])fileList.ToArray(typeof(string));
}
if (scanType == 4)    // usb scan
{
    files = getFiles(loc_to_search, wildcard, SearchOption.AllDirectories);
}
int total = files.Length;
int fileNum = 0;
foreach (string file in files)
{
    fileNum++;
    if (System.IO.File.Exists(file))
    {
        try

```

```

{
    var client = new MrShieldScanClient("localhost", 3310);
    var scanResult = client.ScanFileOnServer(file);
    MrShieldScanResults scanfileType = client.GetFileType(scanResult.Result);
    Invoke(new Action(delegate
    {
        try
        {
            curr_File.Text = file;
        }
        catch { }
    }));
    switch (scanfileType)
    {
        case MrShieldScanResults.VirusDetected:
            infected++;
            AddToResult(file, scanResult.InfectedFiles.First().VirusName);
            break;
    }
}
catch (Exception ex)
{
    frmMain.PushLog(ex.Message);
}
}
Invoke(new Action(delegate
{
    try
    {
        if (total != 0)
        {
            hopeProgressBar1.ValueNumber = (int)(fileNum * 100 / total);
        }
        else
        {
            hopeProgressBar1.ValueNumber = 0;
        }
    }
    catch { }
}));
}
files = null; // allow garbage collector to collect it
Invoke(new Action(delegate
{
    try
    {
        this.label41.Visible = false;
        if(infected == 0)
        {
            curr_File.Visible = false;
            label49.Visible = true;
            label50.Visible = true;
            btnStopScan.Text = "      DONE      ";
            flagScanStatus = 1;
        }
        else
        {
            curr_File.Location = new Point(315, 300);
            curr_File.Size = new Size(250, 60);
            curr_File.Text = "Scan finished!\n Infected Files: " + infected.ToString();

```

```

        btnStopScan.Text = "    NEXT    ";
        flagScanStatus = 2;
    }
}
catch { }
}));
}
private void btnDelete_Click(object sender, EventArgs e)
{
    if(flagDeleteStatus == 0)
    {
        try
        {
            string path = string.Empty;
            for (int i = 0; i < objectListView1.Items.Count; i++)
            {
                path = objectListView1.Items[i].SubItems[0].Text;
                try
                {
                    KillProcess(this.Handle, path); // try to kill the process before deleting it
                    System.IO.File.Delete(path);
                }
                catch { System.IO.File.Delete(path); }
            }
            objectListView1.Items.Clear();
            toolStrip1.Visible = true;
            btnDelete.Text = "    DONE    ";
            btnCancel.Enabled = false;
            label47.Visible = true;
            label48.Visible = true;
            objectListView1.Visible = false;
            flagDeleteStatus = 1;
        }
        catch { }
    }
    else
    {
        flagDeleteStatus = 0;
        toolStrip1.Visible = true;
        btnDelete.Text = "    DELETE    ";
        label47.Visible = false;
        label48.Visible = false;
        objectListView1.Visible = true;
        ShowPanel(pnlShield);
        StripButtonColor(toolStripButton1);
    }
}

private void btnCancel_Click(object sender, EventArgs e)
{
    toolStrip1.Visible = true;
    ShowPanel(pnlShield);
    StripButtonColor(toolStripButton1);
}

private void btnStopScan_Click(object sender, EventArgs e)
{
    switch(flagScanStatus)
    {
        case 0: // stop
            search.Abort();
    }
}

```

```

        hopeProgressBar1.Visible = false;
        label41.Visible = false;
        label42.Visible = false;
        label43.Visible = true;
        label44.Visible = true;
        curr_File.Text = " ";
        label35.Visible = true;
        btnStopScan.Text = "      OK      ";
        flagScanStatus = 3;
        break;
    case 1: // done
        FormatScan();
        btnStopScan.Text = "STOP SCAN";
        flagScanStatus = 0;
        toolStrip1.Visible = true;
        ShowPanel(pnlShield);
        StripButtonColor(toolStripButton1);
        scanType = -1;
        break;
    case 2: // next
        FormatScan();
        btnStopScan.Text = "STOP SCAN";
        flagScanStatus = 0;
        toolStrip1.Visible = false;
        ShowPanel(pnlDelete);
        btnCancel.Enabled = true;
        this.BackgroundImage = null;
        this.BackColor = Color.WhiteSmoke;
        infected = 0;
        break;
    case 3: // ok
        FormatScan();
        btnStopScan.Text = "STOP SCAN";
        flagScanStatus = 0;
        toolStrip1.Visible = true;
        ShowPanel(pnlShield);
        StripButtonColor(toolStripButton1);
        scanType = -1;
        break;
    }
}

private void btnSystemInfo_Click(object sender, EventArgs e)
{
    frmSysInfo dlg = new frmSysInfo();
    dlg.Show();
}

private async void btnOptimizePC_Click(object sender, EventArgs e)
{
    this.btnOptimizePC.Enabled = false;
    if (checkClearMemory.Checked)
    {
        await Optimize.ClearMemory(1);
    }
    if (checkClearCache.Checked)
    {
        await Optimize.ClearMemory(2);
    }
    this.btnOptimizePC.Enabled = true;
}

private async void btnClearMemory_Click(object sender, EventArgs e)

```

```

{
    this.btnClearMemory.Enabled = false;
    await Optimize.ClearMemory(1);
    MessageBox.Show("Memory cleaned!");
    this.btnClearMemory.Enabled = true;
}
public Optimize(System.Windows.Forms.ListView origin, CircularProgressBar.CircularProgressBar oLabel)
{
    try
    {
        Thread enumProcessThread = new Thread(EnumProcesses);
        pList = origin;
        pProcess = oLabel;
        enumProcessThread.IsBackground = true;
        _clearingMemory = false;
        _ramController = new RamController
        (
            RamClearingCompleted,
            5000,
            true,
            100
        );
        _ramController.EnableRamStatistics = true;
        _ramController.MaxUsageHistoryCount = 100;
        _ramController.SetProcessExceptionList(null);
        _ramController.EmptyWorkingSets = true;
        _ramController.ClearFileSystemCache = true;
        _ramController.ClearStandbyCache = true;
        _ramController.FillRam = true;
        _ramController.FillRamMaxRuns = 1;
        _ramController.InvokeGarbageCollector = true;
        _ramController.SetRamUpdateTimerInterval(5000);
        _ramController.AutoOptimizeTimed(false, 600000);
        _ramController.AutoOptimizePercentage = false;
        _ramController.SetAutoOptimizeThreshold(75);
        _ramController.ClearClipboard = true;
        enumProcessThread.Start();
    }
    catch (Exception ex)
    {
        frmMain.PushLog(ex.Message);
    }
}

public async static Task ClearMemory(int index)
{
    if (_clearingMemory) return;
    if (!(_ramController.EmptyWorkingSets && !_ramController.ClearFileSystemCache && !_ramController.ClearClipboard && !_ramController.FillRam)) return;
    frmMain.PushLog("Clearing RAM Memory");
    _clearingMemory = true;
    try
    {
        switch (index)
        {
            case 0:
                await _ramController.ClearMemory();
                break;
            case 1:
                await _ramController.ClearWorkingSets();

```

```

        break;
    case 2:
        await _ramController.ClearFileSystemCaches();
        break;
    case 3:
        await _ramController.FillRamData();
        break;
    }
}
catch (Exception ex)
{
    frmMain.PushLog(ex.Message);
}

_clearingMemory = false;

frmMain.PushLog("Done clearing RAM memory");
}
private void btnCleaner_Click(object sender, EventArgs e)
{
    this.btnCleaner.Enabled = false;
    try
    {
        CleanHelper.CleanTemporaries();
        CleanHelper.CleanMiniDumps();
        CleanHelper.CleanMediaPlayersCache();
        CleanHelper.CleanLogs();
        CleanHelper.CleanErrorReports();
        CleanHelper.EmptyRecycleBin();
    }
    catch (Exception ex)
    {
        frmMain.PushLog("MainForm.CleanPC" + ex.Message + ex.StackTrace);
    }
    MessageBox.Show("Bucket cleaned!");
    this.btnCleaner.Enabled = true;
}
internal static void EmptyFolder(string path)
{
    try
    {
        DirectoryInfo di = new DirectoryInfo(path);

        foreach (FileInfo file in di.GetFiles())
        {
            try
            {
                file.IsReadOnly = false;
                file.Delete();
            }
            catch { }
        }

        foreach (DirectoryInfo dir in di.GetDirectories())
        {
            try
            {
                dir.Delete(true);
            }
            catch { }
        }
    }
}

```

```

    }
}
catch { }
}

internal static void EmptyRecycleBin()
{
    SHEmptyRecycleBin(IntPtr.Zero, null, RecycleFlag.SHERB_NOSOUND |
RecycleFlag.SHERB_NOCONFIRMATION);
}
internal static void CleanTemporaries()
{
    EmptyFolder(TempFolder);
}
internal static void CleanMiniDumps()
{
    EmptyFolder(OSDriveWindows + "\\Minidump");
}
internal static void CleanErrorReports()
{
    EmptyFolder(ProfileAppDataLocal + "\\Microsoft\\Windows\\WER\\ReportArchive");
    EmptyFolder(ProfileAppDataLocal + "\\Microsoft\\Windows\\WER\\ReportQueue");
    EmptyFolder(ProfileAppDataLocal + "\\Microsoft\\Windows\\WER\\Temp");
    EmptyFolder(ProfileAppDataLocal + "\\Microsoft\\Windows\\WER\\ERC");
    EmptyFolder(ProgramData + "\\Microsoft\\Windows\\WER\\ReportArchive");
    EmptyFolder(ProgramData + "\\Microsoft\\Windows\\WER\\ReportQueue");
    EmptyFolder(ProgramData + "\\Microsoft\\Windows\\WER\\Temp");
    EmptyFolder(ProgramData + "\\Microsoft\\Windows\\WER\\ERC");
}
internal static void CleanMediaPlayersCache()
{
    EmptyFolder(ProfileAppDataLocal + "\\Microsoft\\Media Player");
    EmptyFolder(ProfileAppDataLocalLow + "\\Apple Computer\\QuickTime\\downloads");
    EmptyFolder(ProfileAppDataRoaming + "\\Macromedia");

    try
    {
        File.Delete(ProfileAppDataLocalLow + "\\Apple Computer\\QuickTime\\QTPlayerSession.xml");
    }
    catch { }
}
internal static void CleanLogs()
{
    EmptyFolder(System32Folder + "\\LogFiles");
    EmptyFolder(OSDrive + "\\inetpub\\logs\\LogFiles");
}

```

ДОДАТОК В

Результати перевірки роботи на співпадіння

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009368404

Дата проверки:
26.11.2021 13:40:27 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
26.11.2021 13:42:50 EET

ID пользователя:
76913

Название файла: IT-э03мп_Гаврилова_ПЗ

Количество страниц: 72 Количество слов: 14793 Количество символов: 115965 Размер файла: 138.13 KB ID файла: 1009389350

2.83%

Совпадения

Наибольшее совпадение: 0.78% с Интернет-источником (<https://uk.education-wiki.com/8428899-sha-algorithm>)

1.46% Источники из Интернета

7

Страница 74

1.85% Источники из Библиотеки

85

Страница 74

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы

8