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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Лісовий Д.С.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Лісовому Дмитру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Лісовий Д.С.
(прізвище та ініціали)

АНОТАЦІЯ

Лісовий Д.С. Програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

Метою розробки є програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

Результат роботи – програмна реалізація системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

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

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

Програма може використовуватися на мобільному пристрої під керуванням ОС Android.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, шкідливе програмне забезпечення, ОС Android

ABSTRACT

Lisovyi D.S. Software for the cybersecurity system against malicious software in the OS Android. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the cybersecurity system against malicious software in the OS Android.

The purpose of the development is the software for the cybersecurity system against malicious software in the OS Android.

The result of the work is the software implementation of the cybersecurity system against malicious software in the OS Android.

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

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

The program can be used on a mobile device running the OS Android.

The program is developed in the Python environment.

Keywords: cybersecurity, malware, OS Android

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Ідея установки програмного забезпечення для безпеки операційній системі Android для смартфону або планшета може здатися зайвою, але є багато вагомих причин, щоб розглянути подібну можливість.

Будь-яка людина, включаючи шахраїв, може представити свій додаток у магазині контенту Google Play без попереднього схвалення, і надалі це добро може бути завантажено й установлено широким колом користувачів, а також самими різними джерелами. Це означає, що користувачі Android піддаються більшому ризику, ніж користувачі мобільних платформ iOS або Windows Phone, додатки для яких належним чином попередньо вивчаються й перевіряються з метою майбутньої публікації.

Проте, навіть якщо ви обережні при установці різного роду додатків на власний мобільний пристрій, перегляді веб-сайтів або інших речей, існують досить сильні мотиви для установки антивірусних програм. Дане програмне забезпечення звичайно містить у собі функції захисту від крадіжки, наприклад. Всупереч назві, вони не будуть запобігати самому процесу злодійства, але вони можуть дозволити вам заблокувати або стерти дані з украденого смартфону або планшета. Якщо ви навіть просто забули свого мобільного супутника в кіно або ресторані, протиугінні особливості можна використовувати для знаходження пристрою. Менш безпечною є функція резервного копіювання користувальницьких даних. Деякі користувачі просто створюють резервну копію контактів і іншої особистої інформації, коли Android автоматично робить це при вході в аккаунт Google, і більше не турбуються про можливу крадіжку або втрату.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

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

– Огляд існуючих систем від шкідливого програмного забезпечення у ОС Android.

– Дослідження системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

– Програмна реалізація системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі від шкідливого програмного забезпечення у ОС Android.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сьогодні заразити смартфон або планшет під керуванням операційної системи Android досить просто, навіть якщо дотримувати рекомендацій і встановлювати додатки тільки з довірених джерел.

Наприкінці квітня експерти компанії Lookout Mobile Security виявили на Google Play тридцять дві програми, що містять інфіковану бібліотеку. Шкідливий компонент був інтегрований під видом стандартної рекламної функції в ігри, словники й утиліти вже після того, як вони успішно пройшли перевірку в Google Play.

Усього заражені програми були завантажені біля дев'яти мільйонів разів. Крім традиційних рекламних повідомлень користувачі одержували свідомо помилкові попередження про мнимі погрози й критичні відновлення. При клиці на такому баннері відбувався перенапрямок на заражений або фішинговий веб-сайт.

Приховано від користувача троянський компонент підключався до контрольованих зловмисниками віддалених серверів і передавали своїм творцям зібрану інформацію, включаючи номер мобільного телефону і його IMEI. Додатково завантажувався троян сімейства AlphaSMS, що відправляв SMS на платні номери.

Квітневий скандал одержав продовження. Недавно фахівцями компанії Webroot на Google Play був виявлений черговий троян, розповсюджуваний під видом програми керування шрифтами. У всіх випадках потерпілі або не користувалися антивірусами, або останні виявилися неефективні.

1.2 Область застосування

Незадовго до інциденту дослідники з Північно-Західного університету (штат Іллінойс) з'ясували, що багато антивірусів для ОС Android легко одурачити.

У своїй роботі автори дослідження використовували різні техніки модифікації додатків, що утрудняють аналіз їхнього коду, але не приводять до втрати функціональності.

Абсолютно всі антивіруси, що брали участь у тестуванні, для мобільних пристроїв перестали визначати відомі їм шкідливі компоненти після глибокої модифікації останніх. Обфускація коду включала шифрування, перенапрямок викликів і вставку сміттєвих фрагментів. Така обробка вимагає досить високої кваліфікації й нечасто зустрічається в реальному житті. Провал антивірусів у цьому випадку очікуємо, але неприємно дивує інше.

Відомі погрози в експерименті переставали детектуватися навіть після таких тривіальних процедур, як перейменування настановного пакета й зміна полів метаданих. У випадку з рут-експлойтом DroidDream і SMS-трояном FakePlayer складні методи часто виявлялися зайвими. Антивіруси Lookout, Trend Micro і Zoner «засліплювалися» при будь-якій техніці обману.

Причина такої похмурої картини у тому, що ці й багато інших антивірусів обмежуються винятково сигнатурним аналізом. Якщо немає точного збігу із записом у базі даних, то файл зізнається «чистим».

За результатами останніх тестів, багато захисних програм виявилися практично марними. Реалізація AegisLab Antivirus Free, VIRUSfighter FREE Antivirus і Ikarus Mobile Security така, що вони пропускають більшість погроз, створюючи помилкове почуття захищеності.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Avast Mobile Security & Antivirus

Avast Mobile Security & Antivirus пропонує абсолютний шквал зручних інструментів безпеки й функцію резервного копіювання для Android-смартфонів і Android-планшетів. Хочете використовувати повноцінну версію – будьте готові розщедритися й викласти певну суму за доступ до преміум-функціоналу, що особливо нічим не примітний крім основного сканування на наявність вірусів.

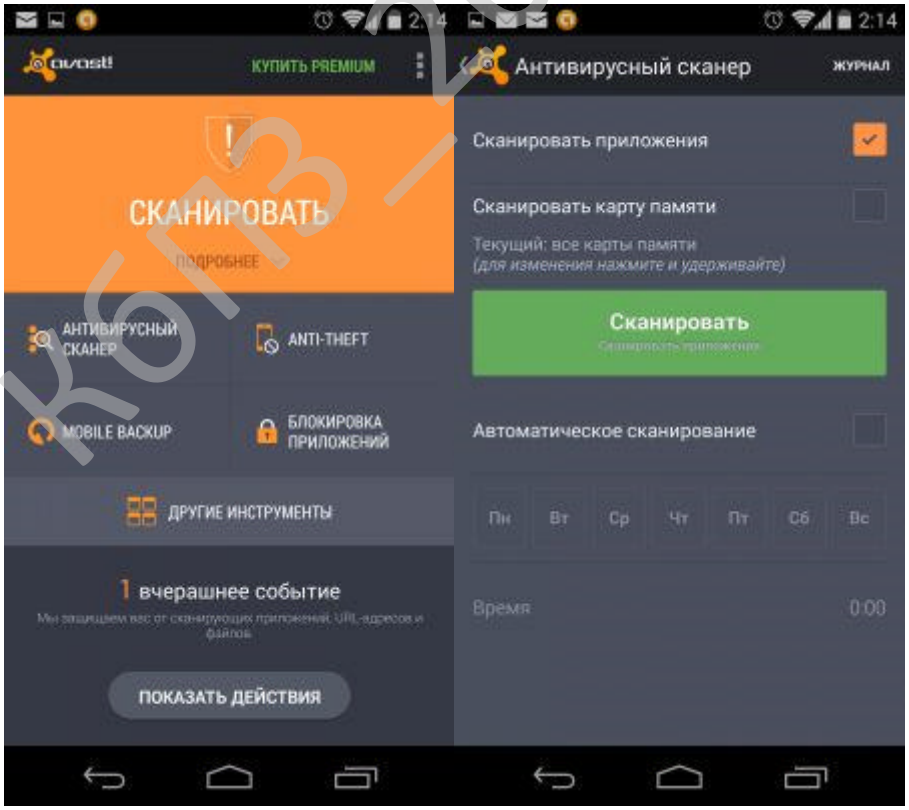

Рисунок 2.1 – Інтерфейс користувача Avast Mobile Security & Antivirus

У засоби проти «викрадення» вашого мобільного пристрою, які встановлюються як окремий додаток, входять невидимка – те, що злодії не зможуть видалити, і це першокласна річ. Використовуючи розроблену веб-консоль або команди текстових повідомлень із іншого смартфону ви можете потенційно піймати злодія на місці злочину таємно стежачи за ним за допомогою фронтальної або основної камери, або таємно записуючи звук – все це згодом можна буде завантажити на персональний комп'ютер, обробити, а потім піймати злочинця. Крім того, є опція, що включить таку сирену й навіть прочитає «уголос» повідомлення: «Цей телефон був загублений або украдений». Зловмисник при цьому не зможе заткнути вашого улюбленця або просто зменшити гучність – йому доведеться насолоджуватися тільки косими поглядами перехожих або наручниками поліції.

Крім цього, смартфон або планшет можна віддалено знайти на карті, заблокувати або знищити. Інші відомі характеристики містять у собі блокування додатків, що змушує користувача ввести PIN-код, щоб відкрити певний додаток: зручно для перешкоди доступу до Google Play або веб-браузера, якщо ви регулярно даєте пристрій дітям, щоб ті грали в ігри, наприклад. Убудований брандмауер дозволяє заборонити використання убудованими або сторонніми додатками мереж мобільного зв'язку (3G, 4G LTE), бездротового інтерфейсу Wi-Fi або роумінгу даних – у цьому випадку необхідні ROOT-права, які, думаю, багато з людей не будуть використовувати, адже зламувати без певного досвіду не рекомендується.

Блокування виклику дозволяє заборонити дзвінки або текстові повідомлення з обраних абонентів із графіком за вашим вибором. У ході тестування ці функції відмінно працювали. Проте, було б непогано, принаймні, одержувати повідомлення про те, що виклик був заблокований.

Окремий додаток Mobile Backup синхронізує копії ваших фотографій, музики, відео, додатків і інших файлів в обліковий запис Google Диска. З огляду на, що операційна система Android з коробки пропонує сервіси Google, додатком

Mobile Backup від Avast навряд чи будуть користуватися, особливо тому, що резервну копію даних можна здійснити стандартними засобами мобільної ОС від Google.

Можливо, сама слабка частина служби Avast є, по іронії долі, антивірусним захистом. Як би додаток не перевіряв наявність вірусів, вдається завантажувати й зберігати шкідливі файли – інші додатки при цьому виявляють їх. Також Avast Mobile Security & Antivirus не забороняє відкривати веб-сайти, присланні за допомогою SMS-фішингу. У всякому разі, додаток здатний забезпечити гідним функціоналом – за окрему ж плату ви одержите ще більше.

Trend Micro Mobile Security & Antivirus

Trend Micro – давнє й відоме ім'я й бренд в охоронному бізнесі. Як же не мати додаток для глобального масштабу популярної операційної системи Android?

Давайте почнемо з позитивних сторін. Сканування мобільного пристрою на наявність шкідливих файлів можна тільки привітати – працює добре. Trend Micro Mobile Security & Antivirus також є єдиним додатком у сьогоднішньому топі, що забороняє мені відвідувати веб-сайти, пов'язані з фішинговими атаками, що нагромадилися в папці спама в електронній пошті Gmail.

Сканування й блокування тестового вірусу пройшло не настільки успішно, наскільки хотілося б, але при повному скануванні програма знаходить віруси – це радує. Що стосується повідомлень, то в додатка від Trend Micro у цьому плані все чітко – повідомлення відображається щораз, коли відбулося сканування встановленого додатка або файлу. Єдиний мінус – ці попередження захламляють панель повідомлень Android через деякий час, але це незначно.

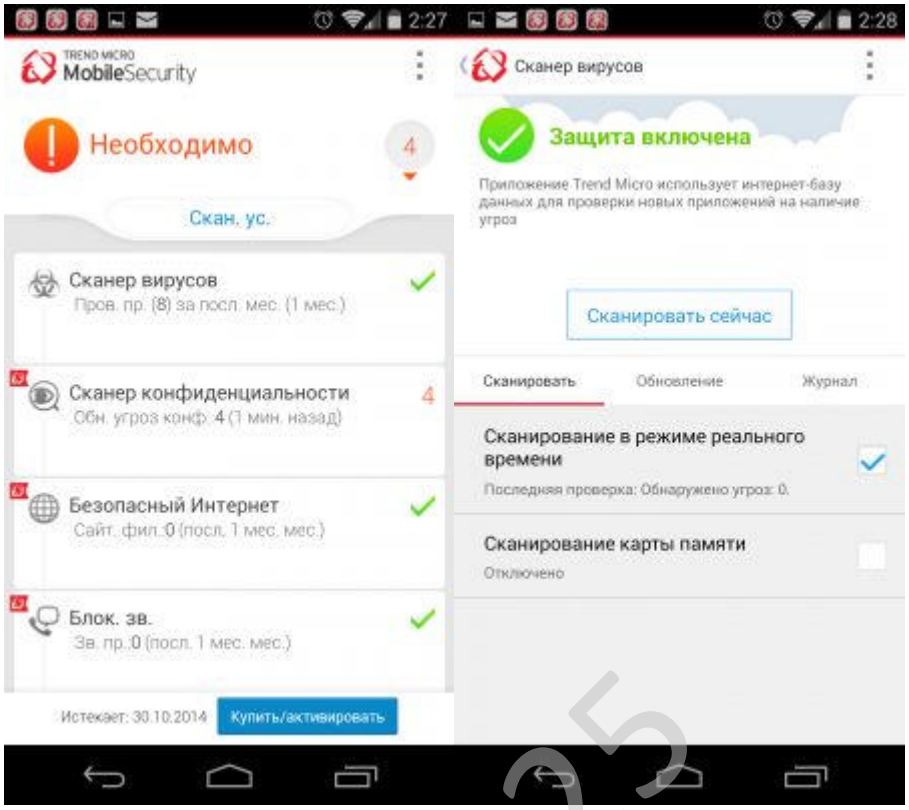

Рисунок 2.2 – Інтерфейс користувача Trend Micro Mobile Security & Antivirus

Функції відносно протиугіності (я буду це так називати, більше звично) досить бідні в порівнянні з конкурентами, описаними вище й нижче – вони можуть бути активовані тільки через веб-портал, тобто немає SMS-керування, що можна зустріти в аналогічному програмному забезпеченні від інших компаній. Є тільки чотири варіанти: пошук, блокування, тривога й видалення даних – у загальному-то, все стандартно. Функція блокування працює добре без звукового сигналу тривоги, однак краще з ним, тому що в цьому є зміст. Незважаючи на той самий корисний зміст, «лемент» загубленого пристрою являє собою лише коротку мелодію. Такий сигнал тривоги навряд чи почуєш навіть із сусідньої кімнати.

Функція видалення надає можливість видалити тільки особисті дані або здійснити повне скидання налаштувань, причому остання – це більше кращий метод, що працює майже миттєво через з'єднання 3G.

В Trend Micro відмінна програма для батьківського контролю мобільних додатків, але вона сміховинно проста в плані обходу цієї системи. Найпростіший спосіб – відкриття браузера Google Chrome у режимі інкогніто буде досить, щоб «зломати» батьківський контроль, даючи нам безперешкодний доступ до порнографічних сайтів, соціальним мережам Facebook, Твіттер, ВКонтактеі й іншим веб-сайтам, які, як правило, найсуворішим образом заблоковані батьками.

Дзвінки й текстові повідомлення можуть бути заблоковані й відправлені як чорний список, так і в білий (тільки певні абоненти). Крім того, є різні варіанти, як звертатися з небажаними дзвінками, починаючи від голосової пошти. Однак, дзвінки заблокованих абонентів будуть змушувати мобільний пристрій репетувати і якийсь час повідомляти вас про це. Як не крути, але ви, захотівши, щоб вас не тривожили, однаково будете приділяти увагу тим самим заблокованим абонентам, тому що смартфон не заспокоїться доти, поки не побачить, що ви в курсі всіх справ.

Резервне копіювання даних можна зробити за допомогою окремого додатка. Тут я мало що скажу, відзначивши тільки, що краще використовувати стандартні засоби операційної системи Android для подібних дій – вони й більше зручні, і пропонують більше можливостей.

IKARUS Mobile Security

IKARUS не є одним з найбільш відомих брендів в області безпеки, але програмне забезпечення в особі антивірусних додатків для операційної системи Android виразно гідно уваги. У магазині додатків Google Play у цього бренда не так багато позитивних оцінок, як у конкурентних рішень. IKARUS Mobile Security спокійно працює на Android-смартфонах і Android-планшетах – благо присутня якісна сумісність і підтримка. Після завантаження й першого запуску додаток відразу почне сканувати вашого електронного помічника на наявність у ньому шкідливих файлів і вірусів – якщо нічого не буде знайдено, то ви зможете далі використовувати IKARUS Mobile Security, а якщо виявиться зло, тоді вам доведеться його викорінити. Коли я завантажив тестовий вірус, програма виявила

щось підозріле й вивела на екран мого смартфону повідомлення, хотів би я видалити шкідливий файл.

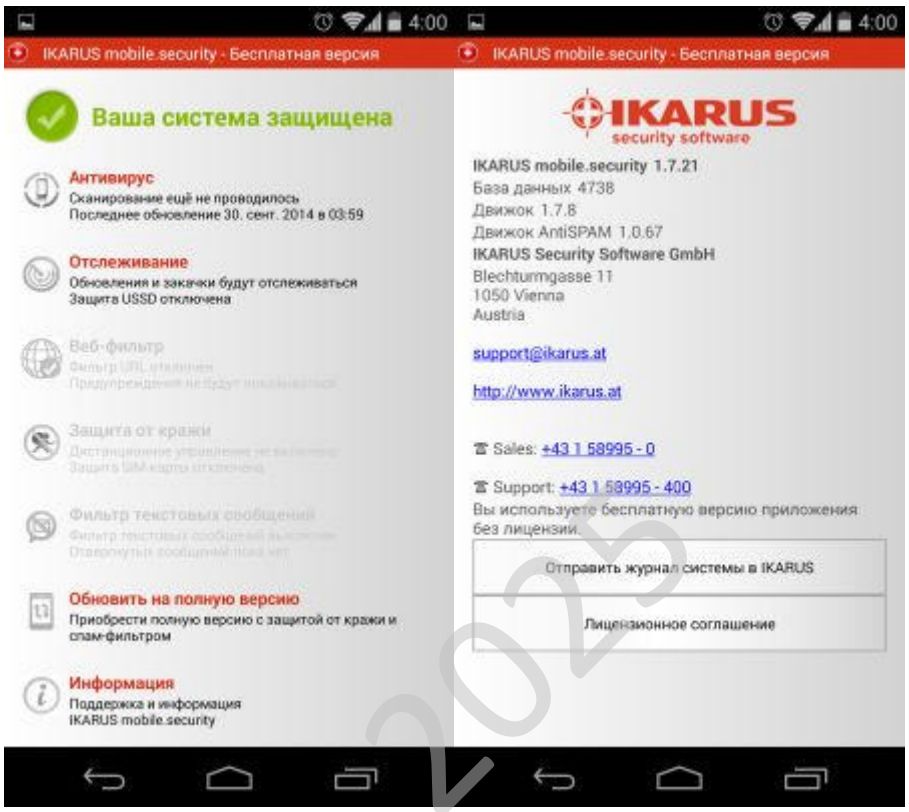

Рисунок 2.3 – Інтерфейс користувача IKARUS Mobile Security

Сам додаток не назвеш вимогливим і енергоємним, він впливає на системні ресурси й автономну роботу мобільного пристрою, маючи при цьому, відповідно, набагато менше можливостей, ніж такі конкуренти, як Avast. Описи вірусів можуть бути оновлені до двох разів у день, щоб переконатися, що ви захищені від новітніх погроз.

Налаштування виробляється швидко й без суєти: IKARUS Mobile Security володіють трьома основними екранами, де можна налаштувати текстові повідомлення, чорні списки, сканування сайтів і віддалені функції безпеки. На жаль, фактичне здійснення цих віддалених функцій безпеки містить у собі безліч турбот.

Ви можете віддалено заблокувати смартфон, відправивши текстове повідомлення з вашим паролем з іншого пристрою. Коли ви заблокуєте пристрій, смартфон дзвонить неодноразово. Що погано, так ця якість цієї функції: при першому розблокуванні смартфону паролем, пристрій продовжував ревіти й довелося перезавантажити його, щоб воно прийняло пароль.

Функція віддаленого стирання даних є набагато менш хворобливою. Вимикання пристрою, перезавантаження й повернення до заводських налаштувань, видалення всіх фотографій, додатків і інших даних у процесі. Підсумок такий: функцій в IKARUS Mobile Security менше, але вони більше якісні в порівнянні з аналогічними продуктами.

Norton Security & Antivirus

Norton – одне з найгучніших імен в охоронному бізнесі поряд з Trend Mobile. Давайте проникнемо в глибини цього антивірусного програмного забезпечення й спробуємо розібратися, які функції може надати користувачеві додаток Norton Mobile Security і який рівень захисту утвору компанії Norton.

Norton Mobile Security уміє, як і багато аналогів, сканувати веб-браузер на знаходження підозрілої активності, хоча я нічого не міг зробити, щоб мій Nexus 5 задзвонив по повній: завантаження тестового вірусу, відкриття посилань із фішингових листів, усілякі сумнівні оголошення на файлообмінних сайтах жодного разу не спровокували Norton втрутитися.

Додаток не пропонує стільки протиугінних можливостей, скільки може запропонувати Avast, і також є деякі тривожні моменти в них. Видалення всього й вся із пристрою – одна із преміум-класу протиугінних функцій Norton Mobile Security, можна активувати через SMS або веб-консоль Нортону – неефективна. Вона не виконує повне скидання налаштувань до заводського стану, а лише очищає контакти, файли й інші персональні дані. Доступ до Gmail і Google Play відкритий, тому дозволяє потенційно украсти злодіям все бажане.

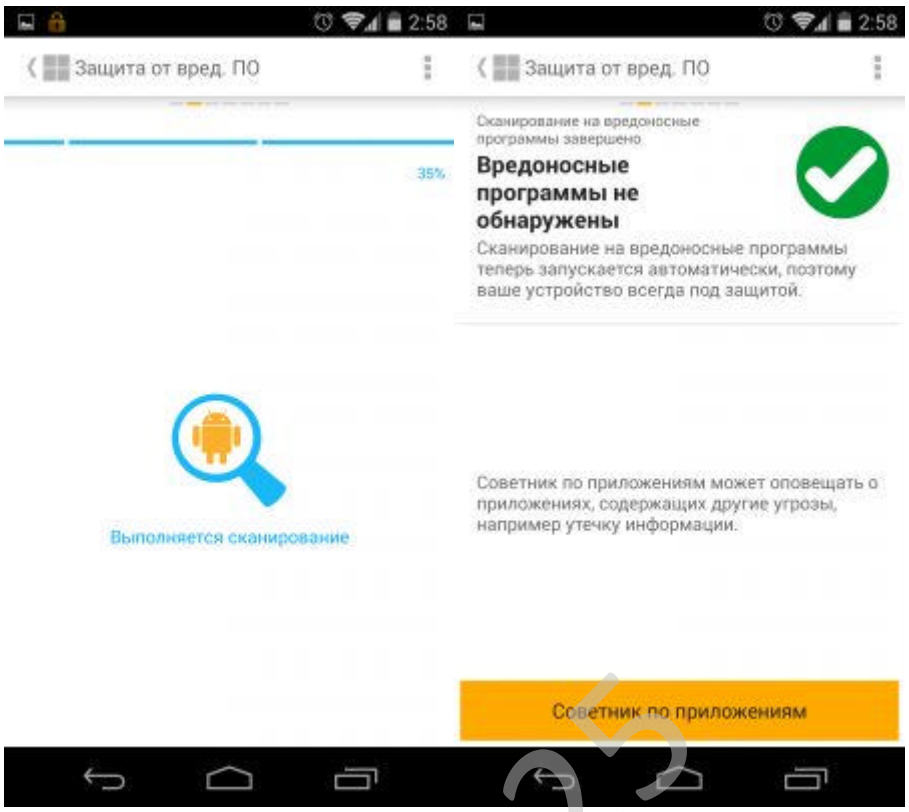

Рисунок 2.4 – Інтерфейс користувача Norton Security & Antivirus

Блокування смартфону або планшета більше ефективні. Функція блокування не відправляє яких-небудь аудіосигналів. Ще можна ввести користувальницьке повідомлення на екран блокування – можливо, ви напишете про гарну винагороду за безпечне повернення, і це приведе вас до мети.

Відстеження місця розташування користувача виявилося небагато норовливим у ході різних тестів. Із включеною функцією Wi-Fi була дана адреса найближчого сусіда, що пробачно, але з мобільним зв'язком третього покоління виявилося, що я живу в 500 метрах від своєї квартири – у цьому мало користі. Коли я спробував використовувати функцію «Sneak Peek», що охоплює можливості камери смартфону, я одержав попередження, що повідомляє мені про те, що ця функція не доступна в моїй країні «у зв'язку із законами таємності».

Блокування викликів у зародковому стані в порівнянні з функціоналом Avast. Ви можете вибрати, які номери телефонів заблокувати або конкретного контакту, а також текстові повідомлення, але немає докладного розкладу, що

незручно. Резервне копіювання рівною мірою примітивно, в Norton Mobile Security це тільки резервне копіювання телефонної книги, тобто контактів, тобто, що є одним зі стандартних засобів операційної системи Android уже протягом багатьох років її існування.

Підводячи підсумки, скажу наступне: Norton Mobile Security простий у використанні, досить продуктивний, безперечно, корисний і пропонує вищеописаний функціонал зовсім безкоштовно. Це додаток гідний займати більше активну позицію, але протиугінні функції відносно слабкі. У всякому разі, ви не витратите ні копійки, тому не варто сприймати все в багнети.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Python – це об’єктноорієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктноорієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному

навчанні, інженерії даних, веброзробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв'язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам'яті. Ця мова програмування використовує великий обсяг пам'яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам'яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам'ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів,

розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов’язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веброзробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

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

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

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

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Антивірус для Android виявляє зловмисне програмне забезпечення та запобігає зараженню вашого пристрою Android. Хоча ті самі комп'ютерні віруси, які загрожують ноутбукам і настільним комп'ютерам, не є загрозою для телефонів і планшетів Android, багато інших типів зловмисного програмного забезпечення є загрозою.

Без антивірусного програмного забезпечення ваш пристрій не матиме захисту від онлайн-злочинців і розширених онлайн-загроз. Зловмисне програмне забезпечення можна використовувати для шантажу, викрадення вашої особистої інформації, грошей і навіть вашої особи. Оскільки пристрої Android більш уразливі до зловмисного програмного забезпечення, ніж iPhone, злочинці часто націлюються саме на них.

Оскільки загрози для Android зростають і стають все більш серйозними, пристрої Android не слід використовувати без антивірусного програмного забезпечення. Антивірус автоматично сканує всі програми та файли, які ви завантажуєте на свій мобільний пристрій. Він захищає вас від троянів, шпигунського програмного забезпечення та всіх інших загроз зловмисного програмного забезпечення, перш ніж вони можуть завдати вам шкоди. Виберіть той, який оптимізований для смартфонів і планшетів Android.

Чи потрібен антивірус для пристроїв Android?

Так само, як настільні комп'ютери з ОС Windows, смартфони з Android також сприйнятливі до вірусів. Мобільні пристрої часто підключаються до зовнішніх мереж, наприклад загальнодоступних Wi-Fi, що наражає їх на різні онлайн-загрози. Деякі віруси спеціально розроблені для націлювання на мобільні пристрої, і їх дуже важко виявити.

Трояни – поширений тип вірусу, який можна маскувати під законні мобільні програми. Наприклад, банківський троян Android виглядає так само, як справжня банківська програма, але може обманом змусити вас надати доступ до вашого онлайн-банківського рахунку. Завантажуйте нові програми лише з офіційного магазину додатків і використовуйте преміальний антивірусний захист, щоб уникнути троянів.

Вбудований захист від вірусів у телефонах Android може забезпечити найпростіші функції безпеки, але цього далеко недостатньо для захисту від сучасних вірусів. Антивірус Google Play Protect на пристроях Android сканує всі програми, встановлені з Google Play Store, і запобігає встановленню шкідливих програм. Однак у вбудованому антивірусі смартфона Android відсутні деякі додаткові функції, необхідні для повної безпеки в Інтернеті, наприклад VPN або менеджер паролів.

Платний або безкоштовний антивірус для Android

Подібно до вбудованого антивірусу Android, безкоштовний антивірус, завантажений із магазину додатків, може забезпечити базовий захист. Оплачуючи преміум-антивірус, ви отримуєте доступ до кількох розширених функцій безпеки, яких немає у безкоштовних програмах.

Антивірус містить усі функції повного мобільного пакету безпеки для ваших пристроїв Android.

– Розширені функції безпеки, які виявляють і видаляють шкідливе програмне забезпечення, яке може заразити ваш пристрій. Антивірус преміум-класу також запобігає потраплянню нових вірусів на ваш телефон.

– Сповіщення про порушення даних у реальному часі, які інформують вас, якщо будь-яка ваша особиста інформація була скомпрометована внаслідок порушення даних. Якщо ваша адреса електронної пошти, номер кредитної картки чи інша інформація витікає, злочинці можуть використовувати її для крадіжки особистих даних або шахрайства.

– Часті оновлення, щоб гарантувати, що ваш антивірус завжди

використовує найновішу версію. Оновлений антивірус покращує ваш веб-захист у разі виявлення нових вірусів.

– Різноманітні функції, яких не вистачає безкоштовному антивірусу, як-от менеджер паролів, VPN та інструменти для захисту ваших грошей і особистих даних в Інтернеті.

– Захист кількох пристроїв одночасно лише за одну підписку. Безкоштовні версії зазвичай охоплюють лише один пристрій одночасно.

– Підтримка клієнтів і професійна допомога експертів з кібербезпеки. Розширене антивірусне програмне забезпечення захищає як ваш телефон Android, так і настільний комп'ютер за допомогою одного облікового запису.

– Краща продуктивність, ніж безкоштовне антивірусне програмне забезпечення. Погано оптимізоване антивірусне програмне забезпечення може сповільнювати роботу телефону Android і розряджати акумулятор пристрою швидше, ніж зазвичай.

Покращте мобільну безпеку на своєму пристрої Android

Мобільна безпека передбачає всі заходи, які використовуються для захисту вашого телефону або планшета Android. Оптимальний мобільний захист захищає ваші пристрої Android від різних форм кіберзлочинності, як-от несанкціонований доступ, злом і шпигунство. Ось кілька порад щодо покращення безпеки мобільного зв'язку та захисту себе в Інтернеті:

1. Використовуйте антивірус для Android. Використання преміум-антивірусу є ключем до повної безпеки в Інтернеті.

2. Оновіть програми та операційні системи. Не забувайте часто оновлювати програми та операційні системи, щоб уникнути нових вірусів. Онлайн-злочинці можуть скористатися помилками та вразливими місцями в старих версіях програмного забезпечення. Завжди оновлюйте програми з офіційного Google App Store, щоб уникнути зловмисного програмного забезпечення. F-Secure Total регулярно оновлюється, щоб відловлювати нові складні онлайн-загрози.

3. Надійні паролі. Завжди використовуйте унікальні та надійні паролі для своїх облікових записів. Створюйте нові паролі за допомогою генератора паролів, щоб уникнути крадіжки особистих даних, і зберігайте свої облікові дані за допомогою менеджера паролів.

4. Не відкривайте підозрілі файли. Якщо ви отримали повідомлення від невідомого або підозрілого відправника, не відкривайте його. Ніколи не натискайте на посилання та не завантажуйте файл, якщо ви не впевнені, що він безпечний. Навіть звичайні електронні листи можуть бути спробами фішингу або містити приховані віруси.

5. Двофакторна аутентифікація. Увімкніть двофакторну автентифікацію у своїх облікових записах, щоб отримати додатковий рівень захисту для свого пристрою Android. Двофакторна автентифікація передбачає підтвердження вашої особи за допомогою додаткового методу на додаток до імені користувача та пароля.

3.2 Розробка структурної схеми

Структурна схема яка зображена на рисунку 3.1 відображає структурні блоки розробленої системи. Розглянемо частково детально елементи структурної схеми.

Фаєрвол

Основна роль фаєрвола – контролювати доступ до ПК із боку зовнішньої мережі, тобто вхідний трафік і, навпаки, контролювати доступ із ПК у мережу, тобто вихідний трафік.

Фільтрація мережного трафіку може відбуватися на декількох рівнях. Більшість фаєрволів, включених у комплекти безпеки для ПК мають набір правил принаймні для двох рівнів – нижній інтернет рівень, контрольований IP правилами й верхній рівень додатка.

Говорячи про верхній рівень, фаєрвол містить набір правил для того щоб допустити а бо заборонити доступ конкретного додатка в мережу. Такі терміни, як мережні правила (Network Rules), експертні правила (Expert Rules) або налаштування правил IP (IP Rule Setting) використовуються на нижньому рівні правил. На верхньому рівні ми зустрічаємося з термінами Контроль програм (Program Control) або правила додатків (Application Rules).

Поведінковий контроль

Компоненти поведінкового контролю (Behavior Control) здійснюють моніторинг дій всіх додатків у системі й блокують дії, які загрожують безпеці системи і її користувачів. Поведінковий аналіз містить базу даних наборів правил, які визначають, які дії повинні бути дозволені або заблоковані для кожної програми. Система захисту виконує контроль і припиняє роботу програм, які можуть виконати потенційно небезпечну дію. Якщо існує правило, що визначає конкретну ситуацію, воно використовується для того щоб або щоб дозволити, або заблокувати дію. Якщо яка-небудь дія вирішена заблокувати, виконання програмного потоку модифікується таким чином, щоб дія не виконувалася й всі параметри додатка міняються для того щоб гарантувати безпека; якщо дія програми дозволена набором правил, його виконання відбувається без змін з боку захисту. Іноді не існує певного правила для дії програми в базі даних. У таких випадках, залежно від налаштувань компонентів поведінкового контролю, користувачеві або пропонується прийняти рішення, або дія виконується або блокується в автоматичному режимі на підставі інформації евристичного аналізу.

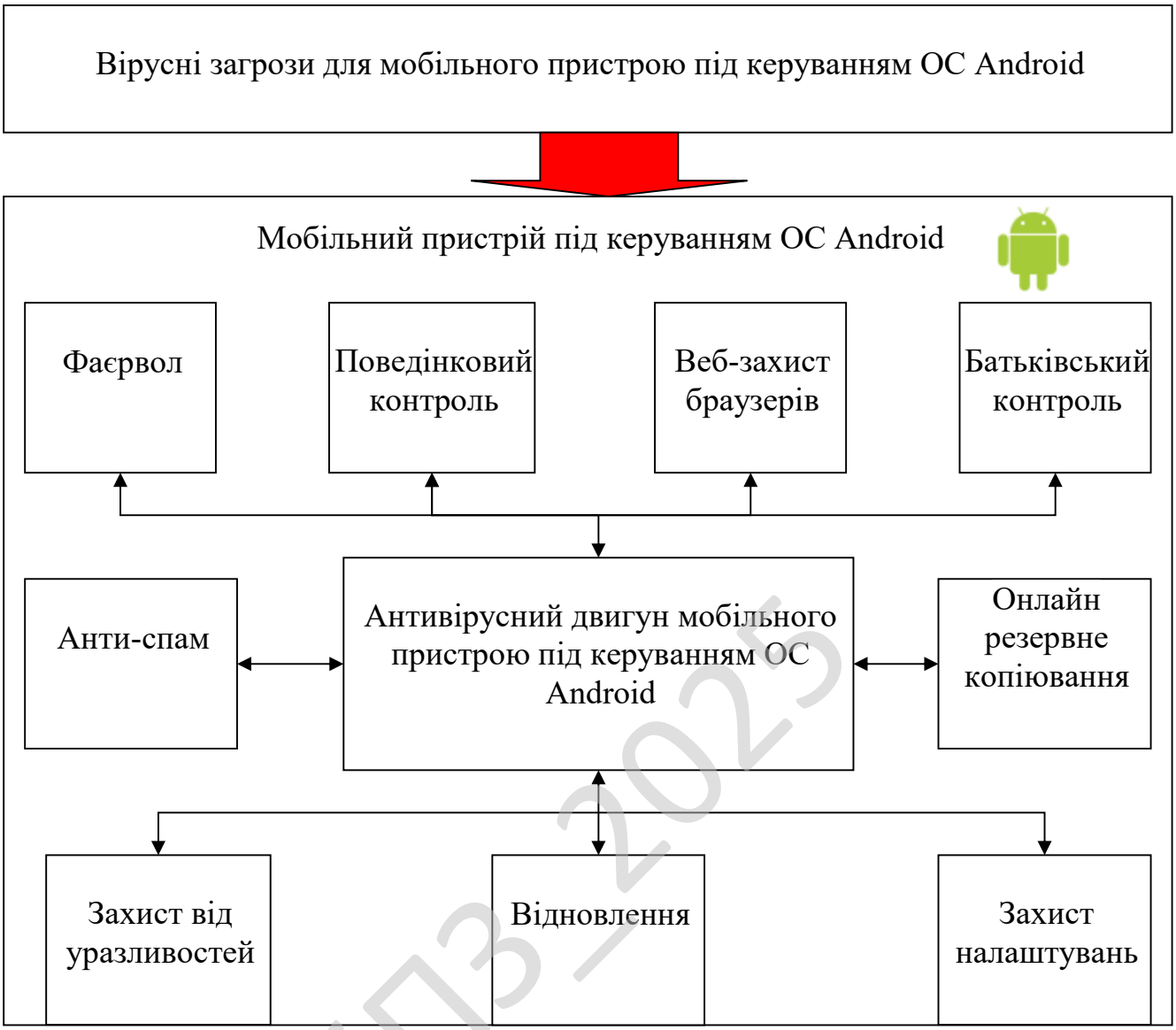

Рисунок 3.1 – Структурна схема системи

3.3 Розробка функціональної схеми

Функціональна схема системи містить у собі наступні блоки.

1. Антивірусний двигун мобільного пристрою під керуванням ОС Android:

– Методи визначення.

– Типи сканування й налаштування.

– Спеціалізовані сканери.

– Додаткові зв'язані опції.

2. Поведінковий контроль:

– Режими роботи.

– Пісочниця.

– Потенційно небезпечні дії й процедури.

– Керування компонентами.

– Системний щит.

– Захист переносних мультимедійних пристроїв.

– Самозахист.

3.Веб-захист браузерів:

– Контроль плагінів.

– Фільтри URL-адрес, блокування реклами.

– Динамічний зміст.

– Кукі.

– Віртуалізація браузера.

– Браузерний і пошуковий помічник, анти-фішинг.

– Сканування веб-змісту.

– Захист конфіденційної інформації.

4. Батьківський контроль.

5.Анти-спам.

6. Захист від уразливостей.

7. Відновлення.

8. Захист налаштувань.

9. Онлайн резервне копіювання.

10. Фаєрвол:

– Розподіл мереж на достовірні й ні.

– Виявлення й запобігання вторгнення.

– Чорний список IP.

– Блокування всього трафіку.

– Контроль програм.

Антивірусний двигун мобільного пристрою під керуванням ОС Android (Anti-virus Engine)

Також називають: антивірусний захист реального часу, захист реального часу, моніторинг файлів, захист від шкідливого ПЗ.

Методи визначення

Антивірусний двигун мобільного пристрою під керуванням ОС Android використовує велику кількість методів для виявлення шкідливого ПЗ. Антивірусні програми містять у собі велику базу зразків вірусів, які необхідно виявити під час сканування. Кожний зразок може або визначати унікальний шкідливий код або, що є більше розповсюдженим, описувати ціле сімейство вірусів. Основна особливість визначення вірусів шляхом порівняння зі зразками в тому, що антивірусна програма може визначити тільки добре відомий вірус, у той час як нові погрози можуть бути не виявлені.

Метод евристичного аналізу (heuristic-based detection) служить для виявлення навіть тих вірусів, для яких не існує зразків у базі антивірусної програми. Існує безліч різних методів евристичного аналізу. Основний принцип – ідентифікувати програмний код, що є вкрай небажаним для безпечних програмних продуктів. Як би те не було, цей метод неточний і може викликати безліч фіктивних тривог. Гарний евристичний аналіз відмінно збалансований і викликає мінімальну кількість фіктивних тривог при великій частці виявлення шкідливого ПЗ. Чутливість евристики може бути настроєна.

Віртуалізація (створення віртуального середовища, Virtualization) або пісочниця (sandboxing) є більше дієвими методами визначення погроз. Певний час зразки коду виконуються у віртуальній машині або іншому безпечному середовищі, звідки сканируємі зразки не можуть вибратися й нашкодити операційній системі. Поводження досліджуваного зразка в пісочниці перебуває під спостереженням і аналізується. Цей метод є зручним у тому випадку, коли шкідливе ПЗ запаковано невідомим алгоритмом (це звичайний спосіб виявитися невразливим для системи виявлення для вірусу), і воно не може бути розпаковано

антивірусною системою. Усередині віртуального середовища вірус розпаковує сам себе, начебто він виконується на реальній системі й Антивірусний двигун мобільного пристрою під керуванням ОС Android може просканувати розпакований код і дані.

Самозахист (Self-protection)

Поведінковий аналіз також відповідає за одну із самих критичних функцій – самозахист антивірусної програми. Будь-який антивірусний захист може виявитися марної, якщо шкідливе ПЗ може відключити її. Сучасні антивірусні програми захищають всі свої компоненти від вірусної погрози так щоб вони не могли бути відключені або ушкоджені. Самозахист припускає захист програмних процесів і потоків, файлів і директорій, записів реєстру і їхніх значень, установлених системних драйверів і служб, інтерфейсів COM і інших ресурсів, створених антивірусом і доступних для інших процесів у системі.

Запобігання інфікування найважливіших процесів є життєво необхідним для будь-якої антивірусної програми. Безліч пакетів безпеки покладаються на постійні відновлення їхньої антивірусної бази. Процес відновлення розробляється максимально невразливим для шкідливого ПЗ, щоб вірус не міг зупинити завантаження або установку відновлень або завантажити підмінні файли відновлень. Самозахист, як правило, включений в основний набір правил поведінкового аналізу, які забороняють керування ресурсами антивірусного продукту. Самозахист може йти окремо від модуля безпеки, що управляє сторонніми програмами. У другому випадку компонента антивіруси краще захищені, ніж будь-який інший додаток у системі. Обидва підходи мають місце в сучасних антивірусних рішеннях.

Веб-захист браузерів (Web and Browser protection)

Також називають: Веб-Контроль (Web Control), веб-безпека (Web Security), веб-захист (Web Protection), захист браузера (Browser Protection), захист перегляду веб-сторінок (Web Browsing Protection)

Контроль плагінів (Plugin Control)

Також називають: **захист плагінів (Plugin Prevention)**

Ланцюжки браузерних процесів, файли й інші ресурси можуть бути захищені за допомогою функціональності стандартного поведінкового контролю, але в загальному випадку ці міри недостатні. Більшість основних браузерів мають плагіни, доповнення, панелі керування, додаткові помічники, які можуть також загрожувати їхньої безпеки. Контроль компонентів у цьому випадку може прийти на допомогу. Антивірус повинен переконатися, що все компоненти браузерного додатка є достовірними й безпечними для користувача.

Фільтри URL-адрес, блокування реклами (Domain and URL Filters, Blocking Ads

Внесення в чорний список шкідливих доменів і посилань є основною функцією. Користувачам, у загальному випадку, дозволяється додавати власні адреси в список фільтрів. Коли браузер намагається з'єднатися із заблокованою адресою, фільтр виявляє цю спробу й забороняє дію до того, як почалася передача даних. Дана функціональність може вступати в дію або на мережному рівні з низькорівневим драйвером ядра, або на рівні розширення браузера.

Розширення браузера є більше розповсюдженим, тому що поводження браузера дуже легко контролюється в цьому випадку й реальне повідомлення про помилку виводиться перед мнимою помилкою з'єднання, ініціюємого шкідливим ПЗ. За замовчуванням фільтри містять шляхи призначення, відомі як злобливі або незаконні: сайти, що використовують фішинг, облудні дії, ресурси, що мають шкідливий код, погану репутацію й порнографічний зміст.

Блокування реклами може бути частиною веб-контролю. Деякі антивіруси пропонують користувачеві заблокувати рекламу, незважаючи на те, що рекламний зміст є важливим складовий бізнес моделей інтернету. Якщо антивірусне ПЗ містить фільтри ресурсів, воно може легко визначити рекламних провайдерів і з високою ефективністю заблокувати більшість рекламних оголошень.

Інші форми блокування реклами включають блокування зображень по їхньому розмірі, блокування спеціальних ключових слів і застосовуються проти настирливого й неналежного контенту. Більшість легітимних рекламних оголошень, відображаються, як правило, коректно.

Динамічний зміст (Dynamic Content)

Динамічний веб-зміст, таке як Flash ролики, Java додатки, об'єкти Active представляють нові способи обходу сучасних захистів. Антивіруси можуть контролювати й блокувати динамічні об'єкти, які завантажуються з недостовірних сайтів. Деякі продукти містять блокування схованих фреймів і спливаючих вікон, які можуть дратувати користувачів.

Найбільш твердий метод – блокування всього динамічного контенту, включаючи скрипти Java. Як би те не було, побічним ефектом цих мір може послужити некоректна робота багатьох сайтів.

Кукі (Cookies)

Файли кукі – невеликі файли, які можуть зберігатися на користувальницькому комп'ютері при відповідному запиті браузера. Веб-сайт може потім звертатися до файлів cookie щоб розширити свою функціональність і рівень взаємодії з користувачем. Як би те не було, механізм використання кукі може бути застосований для відстеження користувальницької активності в мережі.

Це може бути розцінене як неприпустиме порушення користувальницької конфіденційності. Сучасні антивірусні програми дозволяють користувачеві управляти cookie-файлами: видаляти їх або повністю забороняти їхнє створення. Незважаючи на те, що ця функціональність є невід'ємної складової сучасних веб-браузерів, користувачеві може бути більш зручно управляти безпекою комп'ютера з одного додатка – антивірусної програми.

Також як і блокування динамічного вмісту, блокування файлів cookie може викликати порушення в роботі багатьох веб-сайтів.

Віртуалізація браузера (Browser Virtualization)

Віртуалізація браузера є реакцією розроблювачів антивірусного ПЗ на граничну уразливість інтернет браузерів у настільних системах і підвищеній привабливості для кібер-атак. Браузер використовуються для серфінгу по мережі й у випадку взаємодії із сайтом зі шкідливим змістом, може з'явитися уразливість у браузері, що буде сприяти зараженню всього ПК.

Віртуалізація браузера полягає в створені віртуального середовища для роботи інтернет браузера. Всі дії браузера контролюються, якщо вони вважаються потенційно небезпечними, вони перенаправляються в пісочницю. Наприклад, браузер заражений шкідливим кодом, він намагається зберегти злобливі файли на комп'ютері й змінити ключі реєстру для наступного автозапуску вірусу при перезавантаженні системи. Якщо файлові операції й будь-які дії з реєстром будуть відбуватися у віртуальному середовищі, запуск вірусу відбудеться в пісочниці й реальному зараженні системи не відбудеться. При закритті браузера віртуальне середовище знищується, і зараження ліквідується назавжди. У той же час користувачеві дозволено завантажувати необхідні файли в реальну систему, поза пісочницею. Достовірні дії браузера не перенаправляються у віртуальне середовище, користувальницька функціональність браузера, таким чином, зберігається.

Віртуалізація браузера дуже нагадує функцію пісочниці, що ми обговорювали раніше. Відмінність полягає в тому, що цього разу в пісочниці виконуються не підозрілі додатки, а добре відомий і достовірний веб-браузер.

Рисунок 3.2 – Функціональна схема системи

Браузерний і пошуковий помічник, анти-фішинг (Browser and Search Advisor, Anti-phishing)

Також називають: **безпечний Інтернет (Safe Web)**

Призначення браузерного й пошукового помічника – надання інформації про репутацію відвідуваного веб-сайта. Якщо браузерний помічник визначає потенційно небезпечний веб-сайт, що користувач має намір відвідати, виводиться відповідне попередження. Пошуковий порадник зосереджений саме на результатах видачі пошукових машин. При запиті сторінка пошукової системи модифікуються таким чином, щоб потенційно небезпечні посилання або зовсім були невидимими, або принаймні позначені додатковою інформацією про їхню небезпеку для користувача.

Вендори антивірусного ПЗ створюють власні рейтинги інтернет сайтів, використовуючи різні критерії. Веб-сайти скануються на наявність посилань, що ведуть на шкідливі ресурси. Контент сайтів аналізується й класифікується відповідно до фільтрів ключових слів. Застосовується також додавання в білі й чорні списки. Одним з найбільш важливих параметрів ранжирування сайту антивірусом є рейтинг співтовариств. Співтовариство користувачів певного антивірусного продукту є потужною захисною одиницею, що дозволяє користувачам уникати злобливі веб-сайти. Якщо значна кількість користувачів антивірусу відвідують шкідливий сайт, існує більша ймовірність, що багато користувачів позначать цей сайт як небезпечний за допомогою антивірусного ПЗ. Ця негативна репутація згодом буде використана для попередження інших користувачів співтовариства.

Опція анти-фішингу використовуються для запобігання крадіжки платіжних даних. Існує безліч способів розпізнати сайт, що використовує фішинг. Ці методи включають аналіз змісту, виявлення недійсних або несправжніх сертифікатів, детектування підозрілих посилань і т.д. Якщо виявлено спробу фішингу з боку шкідливого сайту, він блокується антивірусом, або критичне попередження виводиться користувачеві.

Сканування веб-змісту (Web Content Scanning)

Деякі функції, які ми обговорили вище, ґрунтуються на скануванні веб-контенту. Це завдання є основною для всіх антивірусних движків і дуже схожі зі скануванням файлів. Єдина проблема полягає в методах шифруванні інформації – наприклад, протокол HTTPS або будь-який інший, працюючий за технологією SSL/TLS, що використовує передачу конфіденційних даних веб-додаткам. Цей вид шифрування розроблений з тією метою, що навіть у випадку, якщо зловмисники побажають переглянути всі дані сесії, виникнути серйозні утруднення при їхньому дешифруванні. Єдиними об'єктами, здатними дешифрувати дані є відправник і одержувач, тобто серверний додаток і користувальницький браузер, наприклад.

Це означає, що антивірусам дуже важко виявити які саме дані передаються й чи не представляють вони небезпеки для користувача. Як би те не було, сучасні антивірусні продукти можуть установлювати свої власні модулі в браузери й, таким чином, стає частиною об'єкта, що має доступу до сирої, незашифрованої інформації. Ця схема діє навіть для сканування веб-змісту, переданого в зашифрованому виді. От чому дуже важливо вести контроль за плагінами й доповненнями, що працюють усередині браузера. Якщо браузер інфікований шкідливим модулем, вірус може спокійно обійти шифрування безпечної сесії.

Захист конфіденційної інформації (Privacy Protection)

Також називають: захист персональної інформації (ID Protection), безпека особистих даних (Identity Safe), захист особистих даних (Identity Protection).

Номера кредитних карт, банківських аккаунтів, електронних платіжних систем і інших паролів, персональна інформація, адреси електронної пошти, номера документів, телефонів є строго конфіденційною інформацією, що повинна бути захищена від дій шкідливого ПЗ.

Функція захисту персональних даних дозволяє користувачеві вирішити, які дані є найбільш конфіденційними й не повинні передаватися без його згоди.

Вихідний трафік сканується на предмет даних, що захищаються, і у випадку збігу, передача даних блокується.

Менеджер паролів (Password Management) іноді входить до складу захисту персональних даних. Користувачі можуть зберігати паролі в безпеці, тому що вони перебувають у зашифрованій формі й доступні тільки власникові. Деякі антивірусні програми розширюють область шифрування, включаючи всі користувальницькі файли й, таким чином, захищають налаштування сторонніх програм і їхні журнали реєстрації подій (логи). Ця міра відноситься до різних чатів і програм обміну повідомленнями, які можуть зберігати історію повідомлень із конфіденційною інформацією на жорсткому диску. Історія браузера, файли cookie і тимчасові файли також перебувають під захистом функції забезпечення безпеки особистих даних. Користувач може настроїти періодичність очищення цих даних: щораз при закритті браузера (цей агресивний підхід може привести до довгого завантаження деяких веб-сайтів), раз у день, раз у тиждень. Використання розкладу означає, що моніторинг користувальницької активності в мережі обмежений певним періодом часу.

Одне з новітніх досягнень антивірусного інструментарію – **хмарне сканування** (scanning in the cloud). Цей метод заснований на тому, що ПК обмежені у своїх обчислювальних здатностях, у те час як антивірусні вендори мають можливість створення великих систем з величезною продуктивністю. Комп'ютерна потужність необхідна для виконання складного евристичного аналізу, а також аналізу з використанням віртуальних машин. Серверу вендорів можуть працювати з набагато більше об'ємними базами даних зразків вірусів у порівнянні із ПК у режимі реального часу. При виконанні хмарного сканування єдиною вимогою є наявність швидкого й стабільного інтернет підключення. Коли клієнтській машині необхідно відсканувати файл, цей файл відправляється на сервер вендора за допомогою мережного з'єднання й очікується відповідь. Тим часом, клієнтська машина може виконувати своє власне сканування.

Типи сканування й налаштування

З погляду користувача, існує кілька типів антивірусного сканування, Які залежать від подій, які викликали запуск процесу сканування:

– **Сканування при доступу** (On access scan) – сканування, що відбувається, коли ресурс стає доступний. Наприклад, коли файл копіюється на жорсткий диск або коли запускається виконується файл, що (запуск процесу сканування в цьому випадку іноді називається сканування при запуску). Тільки ресурс, до якого з'являється доступ, піддається скануванню в цьому випадку.

– **Сканування на вимогу** (On demand scan) провокується кінцевим користувачем – наприклад, коли користувач викликає сканування відповідною командою меню в провіднику Windows. Таке сканування також називається ручним. Тільки обрані папки й файли скануються при цьому способі.

– **Сканування за розкладом** (Scheduled scan) є звичайно повторюваною дією, що забезпечує постійну перевірку системи на предмет шкідливого ПЗ. Користувач може налаштовувати час і частоту сканування. Це сканування звичайно застосовується для повного сканування системи.

– **Сканування при завантаженні** (Startup scan) – сканування, ініціалізуєме антивірусною програмою при запуску ОС. Це сканування відбувається швидко й торкається папки автозавантаження, що запускаються процеси, системну пам'ять, системні служби й завантажувальний сектор.

Більшість продуктів дозволяє користувачам налаштовувати кожний вид сканування роздільно. Нижче зібрані одні із самих основних параметрів антивірусного сканування:

– Розширення файлів для сканування – сканувати всі файли або тільки виконувані (.exe, .dll, .vbs, .cmd і інші.);

– Максимальний розмір файлів – файли понад цей параметр не піддаються скануванню;

– Сканування файлів в архівах – чи сканувати файли в архівах, таких як .zip, .rar, .7z і інші;

– Використання евристичного аналізу – налаштування використання евристики й, опціонально, налаштування чутливості;

– Типи програм, про які повідомити тривогою – існує безліч програм які можуть бути неточно визначені як шкідливі. Звичайно вендори використовуються такі терміни, як Потенційно небажане ПЗ або програма з деяким ризиком погрози;

– Типи носіїв для сканування – чи сканувати файли на мережних сховищах або переносних пристроях зберігання даних;

– Дія, яку потрібно почати, коли погроза виявлена – спроба вилікувати зразок якщо можливо, видалення зразка, приміщення в карантин (спеціальна папка, з якої шкідливий код не може виконуватися, а може бути відправлений для подальшого дослідження безпосередньо вендору), заблокувати доступ або запитати про дію в користувача.

Безліч цих параметрів можуть впливати на швидкість сканування. Набір автоматичних правил сканування для швидкого, але в теж час ефективного сканування називається Інтелектуальне сканування (Smart Scan) або Швидке сканування (Quick Scan). У противному випадку, сканування називається повним (Full Scan) або глибоким (Deep Scan). Ми також можемо зустріти сканування переносних пристроїв, що застосовується для перевірки оптичних дисків, USB накопичувачів, флеш карт і схожих пристроїв. Користувальницьке сканування (Custom Scan) також доступно і є повністю кінцевим користувачем, що налаштовується.

Спеціалізовані сканери

Руткіт-сканування (або антируткіт-сканування) це опція, що пропонують деякі антивірусні вендори в своїх продуктах, тому що руткіти стали надзвичайно розповсюдженими за останнє десятиліття. Руткіт – особливий тип шкідливого ПЗ, що використовує хитрі прийоми для того, щоб залишатися невидимим для користувача й основних методів детектування вірусу. Він застосовує внутрішні механізми ОС для того щоб зробити себе недосяжним. Боротьба з руткітами

жадає від розроблювачів антивірусного ПЗ створення особливих способів виявлення. Руткіт-сканування намагається знайти розбіжності в роботі ОС, які можуть бути доказом наявності руткіта в системі. Деякі реалізації перевірок на наявність руткітов ґрунтуються на постійному моніторингу системі, у той час як інші реалізації антируткітного інструментарію можуть викликатися на вимогу.

Сканування файлів Microsoft Office (або сканування на предмет макровірусів) – опція, що захищає користувача від шкідливого коду усередині офісних документів. Внутрішні принципи сканування схожі із загальними методами сканування, вони просто спеціалізуються на пошуках вірусу усередині макросів. Опція сканування може бути представлена як плагін для Microsoft Office.

Додаткові зв'язані опції

Антивірусний двигун мобільного пристрою під керуванням ОС Android звичайно тісно пов'язаний з іншими компонентами пакета безпеки. Деякі продукти представляють додаткові функції, як частина антивірусного движка, інші відображають їх роздільно. Веб контроль – опція, що є типовим представником другої групи. Ми обговоримо цю опцію окремо.

Фаєрвол (Firewall)

Також називають: персональний фаєрвол, міжмережний екран, розширений брандмауер, двосторонній міжмережний екран.

Мережі

Безліч сучасних продуктів дозволяють користувачеві налаштовувати рівень довіри до всіх мереж, підключених до комп'ютера. Навіть якщо існує тільки одне фізичне підключення, ПК може бути підключений до декількох мереж – наприклад, коли ПК підключений до локальної мережі, що має шлюзи для виходу в інтернет. Антивірусний комплекс буде роздільно управляти локальним і інтернет трафіком. Кожна зі знайдених мереж може бути або довіреною, або недовіреною і різні системні сервіси, такі як загальний доступ для файлів або принтерів можуть бути дозволені або заборонені. За замовчуванням,

тільки комп'ютери з довірених мереж (trusted networks) можуть мати доступ до захищаємого комп'ютера. Підключення, реєструємі з недовірених мереж (untrusted networks) звичайно блокуються, якщо відповідна опція не дозволяє доступ. От чому інтернет з'єднання звичайно позначається як недовірені. Як би те не було, деякі продукти не розрізняють мережі в рамках одного користувальницького інтерфейсу й налаштування довірених/недовірених мереж можуть бути роздільно зазначені для кожного інтерфейсу. Термін Мережна зона (Network Zone) або просто зона (Zone) звичайно використовується замість логічної мережі.

Для недовірених мереж існує можливість настроїти стелс-режим. Цей режим дозволяє змінити поводження системи, начебто її адреса не доступна для мережі. Ця міра здатна ввести в оману хакерів, які спочатку знаходять об'єкти для атаки. Поводження системи за замовчуванням передбачає відповідь на всі повідомлення, навіть відправлені із закритих портів. Стелс-режим (також відомий як стелс-режим порту (stealth ports) запобігає виявлення ПК під час сканування портів.

Виявлення й запобігання вторгнення(Intrusion Detection/Prevention)

Також називають: Виявлення атаки, Система виявлення вторгнення, IP блокування, шкідливі порти.

Хоча всі перераховані вище терміни не є еквівалентними, вони відносяться до набору властивостей, які здатні запобігти або виявити спеціальні види атак з віддалених комп'ютерів. Вони включають такі опції, як виявлення порту сканування, виявлення підмінного IP, блокування доступу до добре відомих портів шкідливого ПЗ, використовуваних програмами вилученого адміністрування, троянським коням, клієнтами ботнета. Деякі терміни включають механізми для захисту від ARP атак (атак з підмінною адресою протоколу розширення) – ця опція може називатися захист від APR, захист від кешу ARP і т.д. Основна здатність цього виду захисту – автоматичне блокування атакуючої машини. Це опція може бути прямо пов'язана з нижченаведеною функцією.

Чорний список IP (IP Blacklist)

Використання цієї простої опції полягає в змісті в антивірусному продукті бази мережних адрес, з якими комп'ютер, що захищається, не повинен зв'язуватися. Ця база може поповнюватися як самим користувачем, при виявленні вірусів, так і автоматично обновлятися з великого списку небезпечних систем і мереж антивірусного вендора.

Блокування всього трафіку (Block All Traffic)

У випадку раптового зараження системи, деякі антивірусні рішення пропонують «нажати на кнопку екстреного гальмування», тобто заблокувати весь вхідний і вихідний трафік. Ця опція може представлятися як більша червона кнопка, або як частина налаштувань політики безпеки фаєрвола або за допомогою іконки в системному меню. Передбачається, що ця функція використовується, коли користувач довідається про те що ПК заражений й хоче запобігти небажаному використанню комп'ютера шкідливим ПЗ: крадіжку персональних даних і завантаження додаткових вірусів через інтернет. Блокування мережного трафіку може бути сполучене із завершенням всіх невідомих системних процесів. Ця опція повинна бути використана з обережністю.

Контроль програм (Program Control)

Також називають: контроль додатків, інспектор додатків.

Фільтрація мережного трафіку на програмному рівні дозволяє програмам безпеки роздільно контролювати доступ у мережу кожного додатка на ПК. Антивірусний продукт містить базу даних властивостей додатків, що чи визначає доступна мережа для чи програми ні. Ці властивості розрізняються між клієнтськими програмами, які ініціалізують підключення з локальної машини на віддалені сервери (вихідний напрямок) і серверними програмами, які сканують мережний порт і приймають з'єднання з віддалених комп'ютерів (вхідний напрямок). Сучасні антивірусні рішення дозволяють користувачеві визначати детальні правила для кожного конкретного додатка.

Загалом, поводження контролю програм залежить від політики безпеки (Firewall Policy), обраної у фаєрволі й може включати наступні режими поводження:

– **Тихий режим** (автоматичний режим) працює без втручання користувача. Всі рішення приймаються автоматично з використанням бази даних антивірусного продукту. У випадку якщо не існує явного правила для програми, що хоче одержати доступ у мережу, цей доступ може бути або завжди дозволений (режим повного дозволу – Allow All mode), або завжди заблокований (режим повного блокування – Block All mode), або спеціальний евристичний аналіз використовується для визначення подальшої дії. Алгоритм вироблення рішення може бути дуже складним і може залежати від додаткових умов, таких як рекомендації мережного співтовариства. Як би те не було, деякі продукти використовують терміни: режим повного дозволу/блокування в обхід існуючої в базі даних наборам правил і просто дозволяють, або блокують доступ будь-якому додатку в системі.

– **Користувальницький режим** (Режим. що налаштовується, – Advanced mode, Custom mode) призначається для просунутих користувачів, які хочуть контролювати кожну дію. У цьому режимі продукт автоматично справляється тільки з тими ситуаціями, для яких існують виняткові правила в базі даних. У випадку будь-яких інших дій користувачеві пропонується прийняти рішення. Деякі антивірусні рішення пропонують визначити політикові поводження, коли неможливо запитати користувача – наприклад при завантаженні комп'ютера, завершенні роботи, коли графічний інтерфейс програми недоступний або під час особливих умов – запуску гри у весь екран, коли користувач не хоче відволікатися (іноді називається Ігровий режим – Gaming mode). Звичайно всього дві опції доступні в цих випадках: режим повного дозволу й режим повного блокування.

– **Нормальний режим** (безпечний режим – Normal mode, Safe mode) дозволяє антивірусному продукту самому справлятися з більшістю ситуацій.

Навіть коли не існує явних правил у базі даних, дія програми дозволена, якщо програма вважається безпечною. Аналогічно автоматичному режиму рішення може прийматися на підставі евристичного аналізу. У випадку, коли програма безпеки не може визначити чи безпечно чи додаток ні, вона виводить оповіщення, як у користувальницькому режимі.

– **Режим навчання** (режим тренування, режим установки – Learning mode, Training mode, Installation mode) в основному використовується відразу після установки антивірусного продукту або у випадках, коли користувач установлює нове ПЗ на комп'ютер. У цьому режимі антивірусний продукт дозволяє всі дії, для яких немає записів у базі даних наборів правив і додає нові правила, які будуть дозволяти відповідні дії в майбутньому після зміна режиму безпеки. Використання режиму навчання дозволяє знизити кількість тривожних оповіщень після установки нового ПЗ.

Контроль програм звичайно містить налаштування, які можуть допомогти продукту вирішити спірні ситуації, незалежно від включеного режиму роботи. Ця особливість відома як автоматичний набір правил (Automatic rule creation). Типова опція в цьому випадку дозволяє будь-які дії додаткам із цифровим підписом від довірених вендорів, навіть якщо немає відповідного запису в базі даних. Ця опція може бути розширена іншою функцією, що дозволяє робити будь-які дії додаткам без цифрового підпису, але знайомих антивірусному продукту. Контроль програм звичайно тісно пов'язаний з іншими функціями, які ми освітимо пізніше, особливо опція поведінкового контролю.

Поведінковий контроль (Behavior Control)

Також називають: Проактивная захист (Proactive Protection, Proactive Defense), активний вірусний контроль (Active Virus Control), захисний екран від вторгнень (Intrusion Guard), основна система запобігання вторгнень (HIPS, Host-based Intrusion Prevention System), поведінковий екран (Behavioral Shield), превентивний захист.

Режими роботи (Operation modes)

Аналогічно політикам безпеки фаєрвола (Firewall Policy), які визначають прийняття рішень за допомогою компонента контролю програм (Program Control), функція поведінкового контролю може мати кілька режимів роботи. Більшість антивірусних програм не надають окремих налаштувань для поведінкового контролю, у таких рішеннях ці налаштування єдині для фаєрвола (Firewall) і модуля поведінкового контролю. Деякі антивірусні рішення дозволяють конфігурувати ці дві функції роздільно, однак доступні режими роботи будуються на тих же принципах, що й політика безпеки фаєрвола:

– **режим навчання** (Learning mode) – поведінковий контроль в автоматичному режимі створює нові набори правил для дій додатків;

– **інтерактивний режим** (Interactive mode) – при спірних ситуаціях з'являється оповіщення й користувачеві пропонується прийняти рішення;

– **тихий режим** (автоматичний режим, Silent mode) – всі дії опції відбуваються в автоматичному режимі.

Деякі антивіруси дозволяють установити ступінь захисту для поведінкового контролю. Це налаштування визначає, які дії додатків вважаються потенційно небезпечними. На нижчих рівнях захисту додаткам дозволяється вільно виконувати практично всі можливі дії, за винятком самих небезпечних. На вищих рівнях, захист є більш твердим, програми перебувають під ретельним спостереженням. Іноді, навіть абсолютно безпечні дії додатків можуть блокуватися в таких випадках.

У деяких випадках є можливість налаштування конкретних реакцій на кожне потенційно небезпечну дію. Дозволити (Allow), заборонити (Block) або запитати (Prompt) – основні доступні опції, які означають, що конкретна дія може бути дозволено, заблоковане або потрібне рішення користувача.

Налаштування поведінкового контролю за замовчуванням мають на увазі використання переважно автоматичних дій і меншого втручання користувача. У таких випадках режим безпеки часто називається «оптимальний». Це означає, що

більшість розширених функцій поведінкового контролю відключені й тільки основні способи виявлення шкідливого ПЗ активні. Якщо користувач хоче убезпечити себе від самих складних видів атак потрібно активувати відповідну опцію. Її назва може відрізнятися в різних вендорів, вона може називатися розширений моніторинг подій (Advanced events monitoring), «анти-витік» (Anti-leak). Звичайно для цієї технології використовуються унікальні назви під зареєстрованими торговельними марками.

Пісочниця, ізольована програмне середовище (Sandbox)

Під час роботи автоматичних режимів антивірусна програма може дозволити потенційно небезпечну дію (опція «Дозволити все» (Alow All/Alow Most) активована) або заблокувати абсолютно безпечні дії програми (опція «Заблокувати все» (Block All/Block Most) включена). Робота автоматичних режимів неідеальна, а використання інтерактивного режиму припускає наявності в користувача певної кваліфікації для прийняття рішень. Крім того, велика кількість питань в інтерактивному режимі може дратувати користувача. У цьому випадку альтернативою може послужити використання ізольованого програмного середовища (sandbox).

Антивірусні продукти, які містять у своєму інструментарії пісочницю, обробляють всі невідомі або підозрілі програми спеціальним методом, які гарантують, що вони не принесуть шкоди системі. Створюється спеціальне середовище, називана пісочницею, що виглядає для як справжня система, що запускаються усередині додатків. Програми можуть вільно управляти об'єктами тільки в пісочниці, їхні дії недоступні для реальної системи (наприклад, зміна записів системного реєстру). Ізольоване програмне середовище гарантує, що небезпечні дії не нашкодять ОС, що запускаємий в ній додаток не може визначити, де саме він виконується. Якщо привести приклад із внесенням змін до реєстру, то додаток намагається прочитати значення запису після зроблених змін, у цей час пісочниця повертає змінені значення, незважаючи на те, що в дійсності системний реєстр виявляється не торкнуть. Існує кілька причин, чому не можна

створити ідеальну пісочницю й чому деякі критичні дії постійно блокуються в безпечному середовищі. Ступінь ефективності ізольованого програмного середовища визначається можливістю її розпізнання з боку шкідливого ПЗ. Чим пісочниця менш помітна додатку, що запускається в ній, тим краще.

Надійні програми завжди запускаються поза ізольованим програмним середовищем, що дозволяє їм виконувати будь-які необхідні для нормальної роботи операції. Коли на комп'ютер установлюється нове невідоме ПЗ і ізольоване програмне середовище забороняє які-небудь дії додатку, користувач може додати цей додаток у список виключень. Деякі антивірусні програми мають у своєму арсеналі пісочницю як окрему функцію поведінкового аналізу. Ці продукти дозволяють, відключивши ізольоване середовище, як і раніше контролювати дія програм. Інші антивірусні рішення вбудовують пісочницю в компоненти поведінкового контролю. Також існують пакети безпеки, які дозволяють налаштовувати, у яких випадках дії додатків повинні бути автоматично заблоковані, а в яких повинне прийматися рішення на підставі поточних налаштувань політики безпеки.

Потенційно небезпечні дії й процедури (Potentially Dangerous Actions and Techniques)

Потенційно небезпечні дії, що розрізняються сучасними антивірусними рішеннями можуть бути розділені на кілька груп.

Сесія динамічного обміну даними (DDE communication) – DDE є міжпроцесорним методом зв'язку, що дозволяє одночасно запускати дві або кілька програм. Серверний додаток, що використовує DDE може одержувати дані від клієнтського додатка й відповідати йому. Деякі додатки, наприклад Internet Explorer, дозволяють іншим додаткам здійснювати контроль, використовуючи команди динамічного обміну. Ця особливість може використовуватися шкідливим ПЗ для маскування небезпечних дій під достовірні джерела.

Контроль доступу об'єктної моделі програмних компонентів (COM Access Control), контроль автоматизації протоколу OLE (OLE Automation

Control) – технологія автоматизації OLE заміняє DDE. Це більше розширений механізм міжпроцесорної взаємодії, заснована на об'єктній моделі програмних компонентів. Безліч важливих системних служб забезпечують інтерфейси для додатків за допомогою технологій COM/OLE. Коли інтерфейс використовується вірусом, складається враження, що ми маємо справу з довіреною службою, а не потенційно небезпечною.

Клієнтські служби виклику віддалених процедур і системи динамічних доменних імен, запит прикладного програмного інтерфейсу системи динамічних доменних імен (DNS/RPC Client Services, DNS API Request) – деякі системні служби, такі як клієнт DNS доступні за допомогою технологій, називаних виклик віддалених процедур, виклик локальних процедур або розширений виклик локальних процедур. Ці процедури використовуються для міжпроцесорної взаємодії. Також як і вищезгадані технології, ці служби можуть бути атаковані шкідливими ПЗ. Моніторинг зв'язаних взаємодій може запобігти зловмисному користуванню цими службами.

Контроль програмних вікон, контроль повідомлень Windows (Application Window Control, Windows Messages) – віконні повідомлення є іншим механізмом міжпроцесорної взаємодії, а також одним з найбільш використовуваних користувальницьких графічних інтерфейсів додатків. Вони можуть часто піддаватися зловмисному використанню шкідливим ПЗ. Використовуючи віконні повідомлення, можливо, імітувати основні дії користувача, наприклад клич кнопкою миші. Поки додаток має графічний інтерфейс, заснований на технології віконних повідомлень, воно може бути атаковано шкідливим ПЗ за допомогою цього методу.

Впровадження коду, впровадження процесу в системну пам'ять, міжпроцесорний доступ до пам'яті (Code Injection, Process Memory Injection, Interprocess Memory Accesses) – впровадження коду в інший процес, запущений у системі є простим методом виконання шкідливого коду під маскою довіреного процесу. Вірус може бути ознайомлений з обмеженнями поведінкового контролю

й для обходу захисту може впроваджувати код у надійний процес, щоб мати можливість зробити шкідливі дії. Захист довірених процесів від впровадження коду є самою головною в поведінковому аналізі сучасних антивірусних продуктів.

Впровадження бібліотек DLL (DLL Injection, Binary Planting) – впровадження бібліотеки DLL схоже із впровадженням шкідливого коду. Результат успішної атаки ідентичний – виконання шкідливого коду за допомогою довіреного додатка. Розходження в тому, що у випадку впровадження DLL, цілий модуль завантажується в процес, що піддається атаці,, у той час як впровадження коду має на увазі як правило, включення невеликої частини коду. Впровадження бібліотек є простим прийомом для розроблювачів вірусів, однак ця методика легко визначається антивірусними програмами.

Запуск додатків з підтримкою мережного обміну даними, запуск процесу, батьківське керування процесом (Network-enabled Application Launch, Process Launching, Parent Process Control) – в ОС Windows батьківський процес може контролювати дочірні процеси або за допомогою завдання певних команд, або використовуючи методи пов'язані із внутрішньою функціональністю процесу. Ця особливість представляє ще один метод атаки довіреного процесу шкідливим ПЗ. Антивірусні програми здійснюють моніторинг ланцюжка батьківських процесів: або всіх запущених у системі, або тільки довірених.

Завершення процесу (Process Termination) – завершення процесу й схожі види атак (завершення потоку, спроби критичного завершення процесу або потоку) припускають часткове ушкодження або повне відключення антивірусного захисту. Мети атаки в цьому випадку – процеси антивірусу. Фактичний результат успішної атаки залежить від реалізації конкретного антивірусного продукту. Атака може привести до нестабільності, зависанням, критичним помилкам або відключенню деяких функцій безпеки. Деякі антивіруси можуть розпізнавати ушкодження своїх компонентів і блокують ПК для запобігання подальших шкідливих дій.

Низькорівневий доступ до мережі, прямий доступ до мережі (Low-level Network Access, Direct Network Access) – більшість антивірусів здатні відмінно справлятися з контролем основного мережного трафіку, такого як веб-серфінг, повідомлення e-mail, але з'являються проблеми, коли справа стосується протоколів спеціального призначення. Нерідкі випадок, коли антивіруси дозволяють взаємодія з веб-сайтами (при використанні гіпертекстового протоколу передачі – HTTP) тільки довіреним джерелам, у той час як передача даних за допомогою протоколу керування мережними повідомленнями (ICMP) відбувається безконтрольно в автоматичному режимі. Таким чином, шкідливі програми, що використовують альтернативні методи передачі даних менш уразливі для сучасних антивірусних рішень.

Прямий доступ до диска (Direct Disk Access) – основний спосіб доступу до даних на жорсткому диску включає системні функції, які працюють із файлами й директоріями. Ранні версії Windows дозволяють додаткам прямо звертатися до диска й даних на ньому. Такий метод доступу до даних на диску дозволяє обходити основні способи захисту директорій. На ОС Windows Vista і більше пізніх ОС Windows, ця процедура обмежена й менш уразлива для шкідливих атак.

Доступ до оперативної пам'яті, прямий доступ до пам'яті (Physical Memory Access, Direct Memory Access) – кожний працюючий процес у системі має свою власну пам'ять, недоступну іншим додаткам за замовчуванням. У випадках, коли потрібен віддалений доступ до пам'яті, система робить це можливим за допомогою спеціальних функцій. У той же час антивірусна система здійснює контроль даного вила доступу. Ядро ОС також має власну пам'ять, недоступну іншим додаткам. Як би те не було, у старих ОС Windows була можливість доступу до об'єкта, що торкатися всієї пам'яті, включаючи область системного ядра. Це дозволяло шкідливому ПЗ обходити основні механізми доступу до пам'яті. В Windows Vista і більше пізніх системах, дана опція заборонена.

Установка драйверів пристроїв, ініціалізація драйвера (Device Driver Installation, Driver Load) – Додатка, що працюють в ОС Windows мають деякі обмеження, що особливо стосуються використання ресурсів апаратних засобів, таких як оперативна пам'ять, жорсткий диск, пристрої уведення й виводу й т.д. Коли додаток прагне використовувати апаратний засіб, воно звертається до системного ядра, що може або дозволити, або заборонити конкретну дію. Цей механізм відмінно працює із програмним кодом, що працює в так званому користувальницькому режимі. Код системного ядра у свою чергу працює в так званому режимі ядра, що дозволяє будь-який доступ до апаратних засобів без обмежень. Код системного ядра може обходити всі види захисту, включені в ОС або надавані сторонніми програмами. Додаток, що працює в користувальницькому режимі може завантажити драйвер пристрою, код якого працює в режимі системного ядра. От чому шкідливі драйвера не повинні завантажуватися, і необхідний постійний контроль за цим. На 64-бітних системах Windows цей метод практично непридатний для використання шкідливими програмами через запит цифрового підпису кожного драйвера працюючого в режимі ядра.

Установка служб (Service Installation) – Системні служби в ОС Windows – спеціальні програми, які можуть працювати, навіть коли завершений сеанс користувача. Вони є більше пріоритетними в порівнянні зі звичайними додатками, не вимагають прямої взаємодії з користувачами й можуть запускатися автоматично під час завантаження системи. Деякі служби не мають своїх власних процесів і розміщаються в інших схожих службах усередині спеціальних процесів. Служби є дуже простим способом для шкідливого ПЗ щоб закріпитися в системі. Антивірусні програми також звичайно мають одну або кілька служб. Шкідливі програми можуть відключити найважливіші компоненти антивірусу, якщо не контролювати постійно установку системних служб. Більше того, Для установки драйверів, що працюють у режимі ядра використовується той же інтерфейс, що й для установки системних служб.

Доступ до файлу HOSTS – файлу бази даних доменних імен (HOSTS File Access) – HOSTS файл – спеціальний файл, що містить відповідності мережних імен і IP адрес. Говорячи загальними словами, мережні імена це домени, а зв'язки між доменом і IP адресою визначаються за допомогою проколу системних доменних імен. Як би те не був, саме файл HOSTS використовується для перекладу мережних імен, включаючи домени, в IP адреси. Таким чином, за допомогою файлу HOSTS можливо перенаправляти домен до довільного IP адресі. Основний прийом вірусів полягає в перенапрямку серверів відновлень антивірусної програми до неіснуючих адрес, що паралізує можливість відновлення антивірусу. Інший прийом використовується для фішингу – перенапрямку домена різних електронних платіжних системам до шкідливих серверів, які виглядають ідентично оригінальному сайту й здійснення крадіжки конфіденційних платіжних даних.

Активні зміни робочого стола (Active Desktop Changes) – ранні версії ОС Windows мали можливість внесення активного вмісту користувачем на робочий стіл. Ця опція дозволяла створювати повністю робочі столи, що набудовуються. Активний робочий стіл може зловмисно використовуватися шкідливим ПЗ під маскою довіреного додатка провідника Windows. Windows Vista і більше пізні системи не мають підтримку активного робочого стола.

Папки автозавантаження й автозапуску (Autoruns, Autostart Locations) – Додаток має безліч способів для установки в ОС із наступним автозапуском при перезавантаженні системи. Деякі із цих способів дозволяють заражати різні системні процеси шкідливою бібліотекою DLL, тобто виконувати впровадження DLL. У загальному випадку, шкідливі програми використовують кілька папок автозавантаження для того, щоб улаштуватися в системі.

Реєстрація уведень із клавіатури, кейлоггінг (Keylogging, Keyboard Logging) – спостереження за діями користувача є ще однією популярною діяльністю шкідливим програм. Методи реєстрації клавішного уведення дозволяють одержати інформацію, що користувач уводив в інший додаток за

допомогою клавіатури. Використання цих методик дозволяє красти паролі, уведені в браузері, поштовому клієнті або клієнті обміну текстовими повідомленнями. Деякі прийоми кейлоггінга ґрунтуються на впровадженні DLL або захвату віконного інтерфейсу.

Захват зображень із екрана й логгінг буфера обміну (Screen and Clipboard Logging) – скринлоггінг і реєстрація буфера обміну також використовується для крадіжки точної конфіденційної інформації. Виконання знімків з екрана може бути використане для крадіжки даних кредитної картки, уведених на безпечної веб-сторінці в браузері. Логгінг буфера обміну дозволяє украсти дані, які користувач використовує для копіювання в ОС Windows. Багато користувачів переносять конфіденційну інформацію, таку як паролі через буфер обміну. Звичайно це трапляється, коли веб-додаток запитує складні паролі. З одного боку, використання складних паролів є необхідністю, тому що вони менш уразливі для злому, але з іншої сторони користувач, що використовує подібні паролі в різних додатках, фізично не в змозі їх запам'ятати й використовує програму для зберігання паролів або просто текстовий файл для копіювання й вставки пароля у відповідну форму.

Загарбник вікон, захват системних подій (Window Hooking, Windows and WinEvent Hooks) – захват віконних повідомлень Windows і так званих системних подій дозволяє ОС запропонувати ряд спеціалізованих прикладних програмних функцій для програм з метою моніторингу віконних повідомлень і сформованих повідомлень про системні події. Ці функції також можуть бути використані шкідливими ПЗ для впровадження злобливих дій, таких як впровадження DLL бібліотек або кейлоггінг.

Керування компонентами (Component control)

Також називають: **відомі компоненти (Known Components)**

Кожний додаток використовує один або кілька виконуємих модулів, які іноді називають компонентами. Основний модуль – як правило, файл із розширенням .exe, який припускає завантаження деяких динамічно зв'язаних

бібліотек (файлів з розширенням .dll), що перебувають у тій же директорії. Основні додаток використовують бібліотеки ядра системи: Kernel32.dll, KernelBase.dll, ntdll.dll, Advapi32.dll, user32.dll і інші. Безліч програм використовують сторонні бібліотеки, які встановлюються в систему разом з основним програмним модулем. Файли .dll можуть завантажуватися до пам'яті або під час ініціалізації додатка, або під час запиту певної функціональності в додатку. Таким чином, кожний додаток має певний набір файлів бібліотек, які завантажуються до пам'яті й від яких залежить його робота. Контроль компонентів (Component Control) визначає ці залежності й контролює завантаження модулів у процес додатка. Коли шкідливе ПЗ намагається впровадити свою бібліотеку DLL в інший процес, компонентний контроль розпізнає й забороняє ця небезпечна дія.

Компонентний контроль також гарантує недоторканність достовірних безпечних модулів. Будь-які спроби змінити файли надійних відомих модулів можуть бути розпізнані й заблоковані. Це відноситься як до головних файлів, що виконуються, так і до файлів динамічно зв'язаних бібліотек.

Системний щит (System Guard)

Деякі антивірусні програми мають різні рівні захисту для сторонніх програм, а також для ОС і її компонентів. Системний щит є частиною поведінкового аналізу, що відповідає за захист ОС від шкідливого ПЗ і схоронність оригінальних модулів системи. Важливі системні файли, критичні записи реєстру (у тому числі автозавантаження) і інші системні ресурси, які можуть бути інфіковані вірусом перебувають під спостереженням системного щита. Будь-яка спроба зміни ОС блокується.

Захист переносних мультимедійних пристроїв (Removable Media Protection)

Основна функціональність сучасних антивірусних програм у галузі захисту переносних мультимедійних пристроїв (USB-флешки, зовнішні HDD-диски) припускає відключення функції автозавантаження або автозапуска. Коли

переносний пристрій включається в комп'ютер, а його корінна директорія містить файл Autorun.inf, стороння програма може запуститися системою. Це може привезти до непомітного зараження комп'ютера.

Більшість антивірусних рішень також визначають спеціальний набір правил для всіх програм, розташованих на переносних пристроях. Передбачаються, що файли на переносних накопичувачах можуть з'явитися з інших ПК, інфікованих і не оснащених достатнім рівнем безпеки. От чому програми на переносних пристроях уважаються за замовчуванням потенційно небезпечними і їхньою дією строго обмежені. Деякі пакети безпеки можуть розпізнавати програми з електронним підписом від надійних джерел і не обмежувати дії таких додатків.

Інші блоки схеми були більш докладно розглянуті в структурній схемі.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування ПЗ.

– Онлайн резервне копіювання.

– Відновлення пошкоджених ресурсів.

– Моніторинг ресурсів.

– Вибір ресурсів.

– Запуск.

– Аналіз вибраних ресурсів.

– Зупинка.

– Звіт.

– Список інфікованих файлів.

– Знищення вірусів.

– Видалення файлів.

– Лікування файлів.

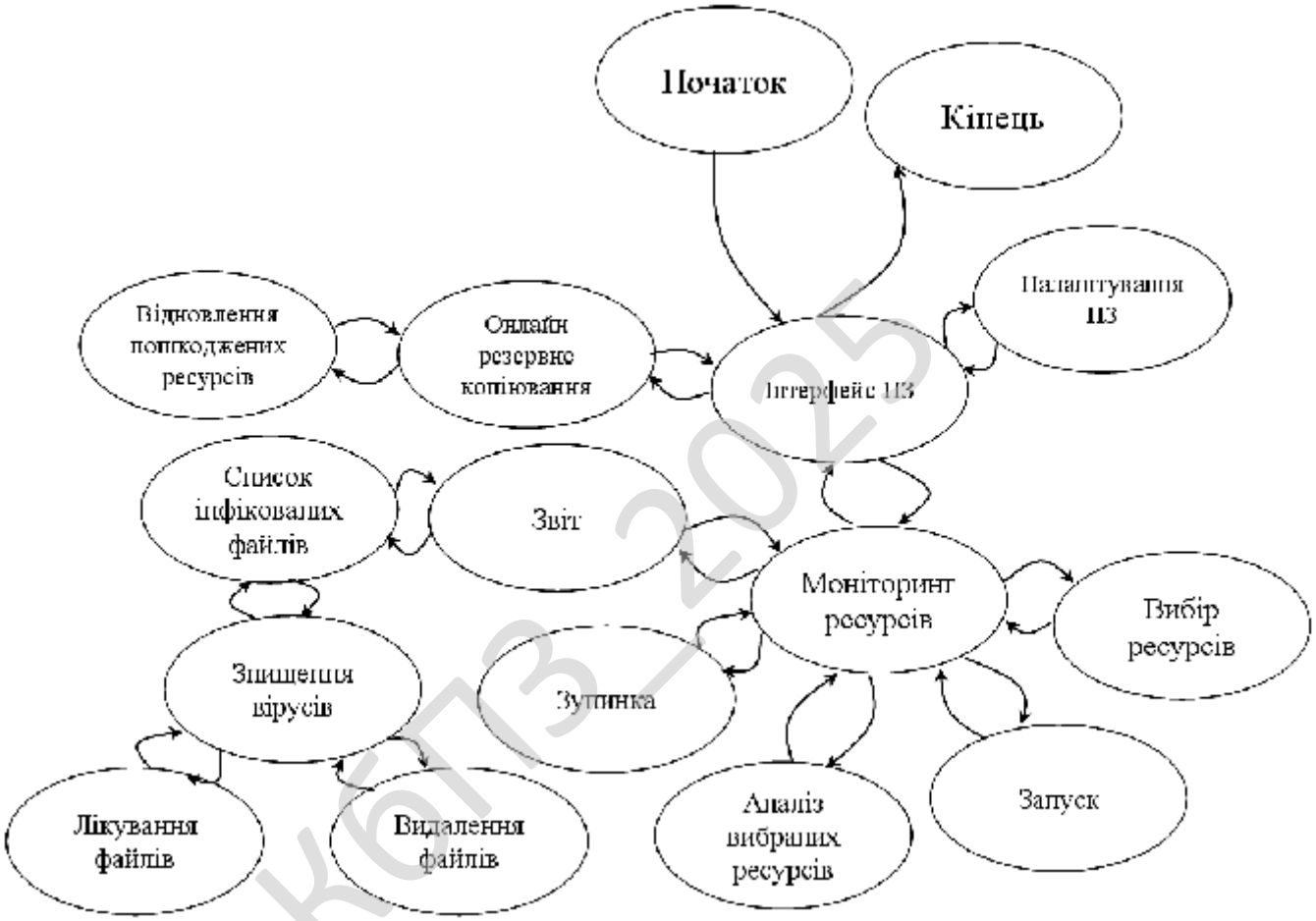

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

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

Опис алгоритмів функціонування системи

Розглянемо програмний код, в якому реалізовано програмне забезпечення системи захисту від шкідливого програмного забезпечення у ОС Android. Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю захисту від шкідливого програмного забезпечення.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

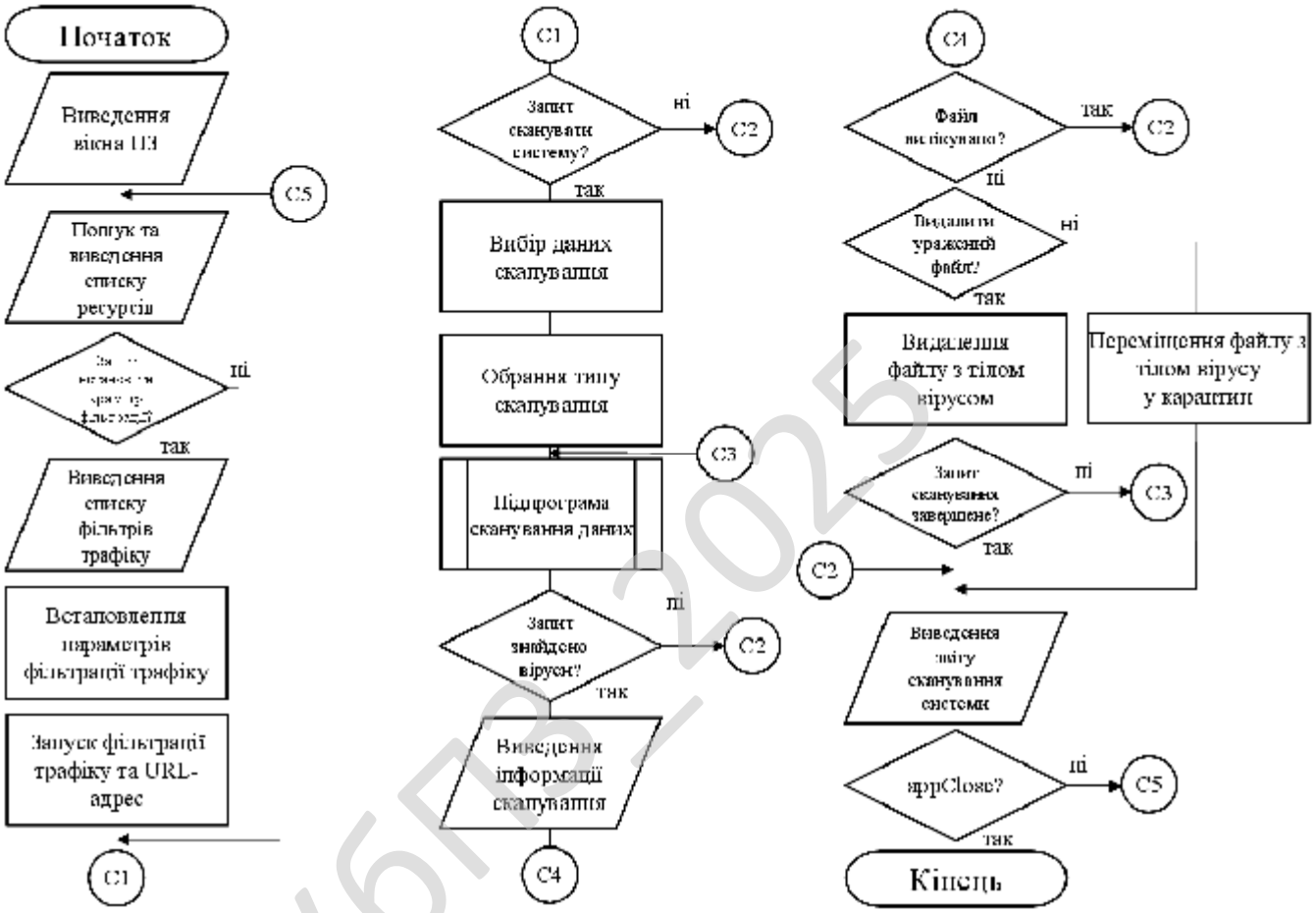

Рисунок 4.1 – Блок схема основної програми

Крім цього було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному

програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

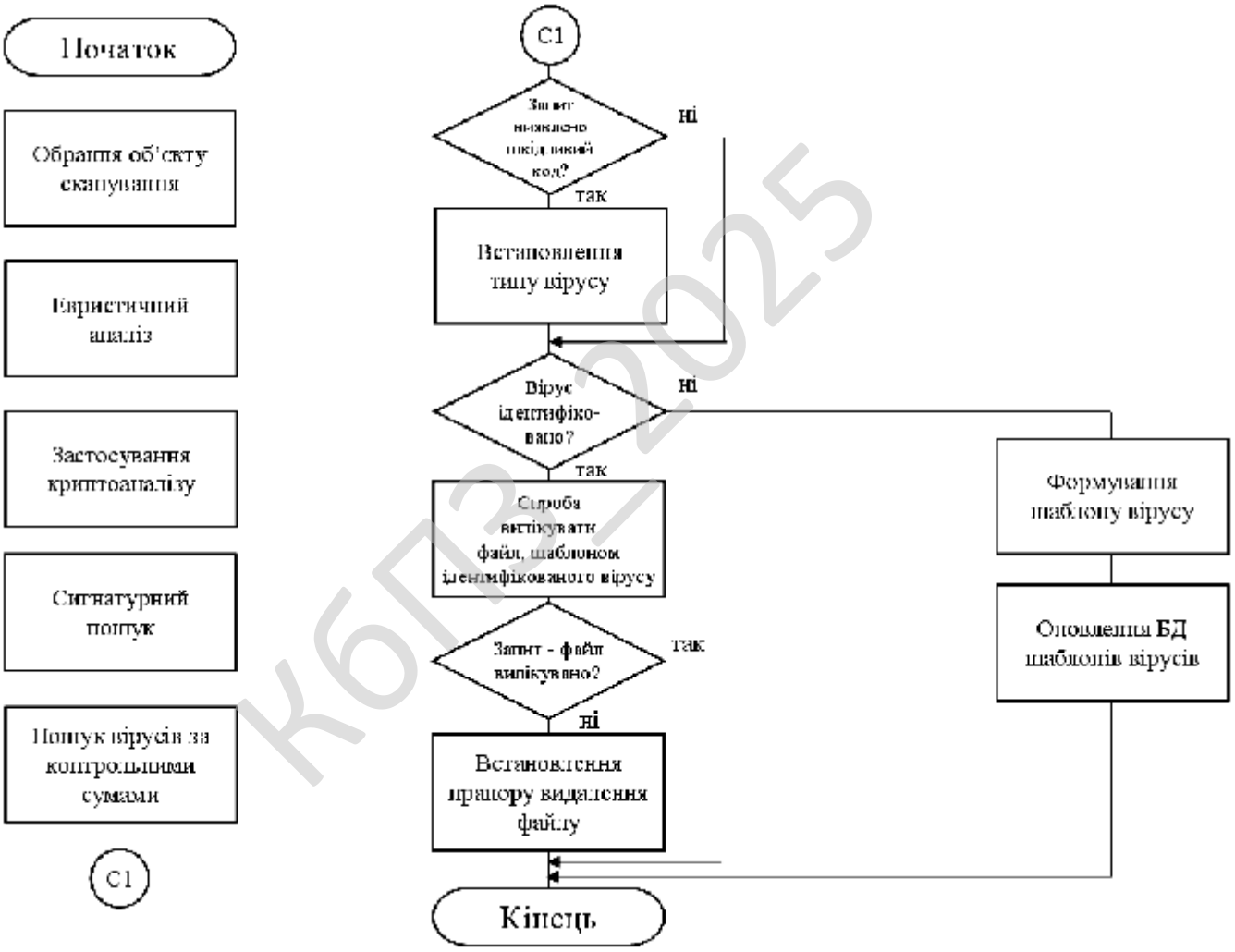

Рисунок 4.2 – Блок схема підпрограми

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний

код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Система кібербезпеки від шкідливого програмного забезпечення у операційній системі Android забезпечує виявлення, запобігання та усунення загроз, що можуть призвести до втрати конфіденційності даних або порушення нормальної роботи пристрою. Архітектура системи включає кілька основних модулів, кожен з яких виконує певні функції.

Система складається з модулів аналізу поведінки програм, сканування файлової системи, перевірки мережевої активності, журналювання подій та механізму оновлення бази даних загроз. Основний модуль відповідає за взаємодію між цими компонентами та їх координацію.

Модуль аналізу поведінки програм виконує моніторинг активності встановлених додатків. Він перевіряє запити до системних ресурсів, аналізує привілеї та поведінку в реальному часі. Вихідний код цього модуля включає механізм перевірки дозволів та виявлення нетипових дій.

Модуль сканування файлової системи здійснює перевірку файлів на предмет наявності відомих шкідливих сигнатур. Він використовує алгоритми хешування та евристичний аналіз для виявлення модифікованих або потенційно небезпечних файлів. Вихідний код цього модуля реалізує функцію перевірки контрольних сум та порівняння їх з еталонними значеннями.

Модуль перевірки мережевої активності аналізує мережеві з’єднання та передачу даних. Він відстежує комунікації додатків з віддаленими серверами та перевіряє їх на відповідність базі відомих шкідливих доменів. Вихідний код реалізує механізм перевірки мережевих запитів та їх блокування у разі виявлення загрози.

Модуль журналювання подій зберігає інформацію про виявлені загрози та підозрілу активність. Він формує звітність для користувача та адміністратора системи. Вихідний код містить функції запису логів та їх шифрування для забезпечення цілісності даних.

Механізм оновлення бази даних загроз забезпечує своєчасне поповнення списку відомих шкідливих об’єктів. Він працює у фоновому режимі та

періодично звертається до сервера оновлень. Вихідний код містить алгоритм перевірки актуальності бази та завантаження нових даних.

Архітектура системи побудована за модульним принципом, що дозволяє легко розширювати функціональність та оновлювати окремі компоненти без впливу на загальну роботу. Обмін даними між модулями відбувається через внутрішні API, що забезпечує безпеку та ізольованість процесів.

Розрахунки продуктивності показують, що система незначно впливає на ресурси пристрою. Використання багатопоточності дозволяє зменшити затримки при аналізі великих обсягів даних. Оптимізація алгоритмів сканування та кешування результатів забезпечує швидке виявлення загроз.

Система підтверджує ефективність через тестування на наборі реальних загроз. Аналіз результатів показує високий рівень виявлення шкідливого програмного забезпечення та мінімальну кількість хибних спрацьовувань.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Threefish – в криптографії симетричний блоковий криптоалгоритм, розроблений автором Blowfish та Twofish, американським криптографом Брюсом Шнайером 2008 року для використання в геш-функції Skein і як універсальну заміну наявним блоковим шифрам. Основними принципами розробки шифру були: мінімальне використання пам'яті, необхідна для використання в геш-функції стійкість до атак, простота реалізації та оптимізація під 64-розрядні процесори.

Структура алгоритму

Threefish має дуже просту структуру і може бути використаний для заміни алгоритмів блочного шифрування, бувши швидким і гнучким шифром, що працюють в довільному режимі шифрування. Threefish S-блоки не використовує, заснований на комбінації інструкцій виключаючого або, складання і циклічного

зсуву. Як і AES, шифр реалізований у вигляді підстановочно-перестановочної мережі на оборотних операціях, не будучи шифром мережі Фейстел.

Алгоритм передбачає використання tweak-значення, свого роду вектора ініціалізації, дозволяючи змінювати таким чином значення виходу, без зміни ключа, що має позитивний ефект як для реалізації нових режимів шифрування, так і на криптостойкости алгоритму.

Як результат думки авторів, що кілька складних раундів часто гірше ніж застосування великого числа простих раундів, алгоритм має нетрадиційно велику кількість раундів – 72 або 80 при ключі 1024 біт, проте, за заявою творців, його швидкісні характеристики випереджають AES приблизно вдвічі. Варто зауважити, що через 64-бітної структури шифру, дана заява має місцево лише на 64-розрядної архітектури. Тому, Threefish, як і Skein [1], заснований на ньому, на 32-розрядних процесорах показує значно гірші результати ніж на «рідному» обладнанні.

Ядром шифру є проста функція «MIX», перетворювальна два 64-бітових беззнакових числа, в процесі якої відбувається складання, циклічний зсув (ROL / ROR), і додавання по модулю 2 (XOR) .

Нижче представлений код MIX-функції для Threefish-1024 [2]:

```
<syntaxhighlight lang="C">
// Константи для циклічного зсуву
int R16 [8] [8] = {
  {55, 43, 37, 40, 16, 22, 38, 12},
  {25, 25, 46, 13, 14, 13, 52, 57},
  {33, 8, 18, 57, 21, 12, 32, 54},
  {34, 43, 25, 60, 44, 9, 59, 34},
  {28, 7, 47, 48, 51, 9, 35, 41},
  {17, 6, 18, 25, 43, 42, 40, 15},
  {58, 7, 32, 45, 19, 18, 2, 56},
  {47, 49, 27, 58, 37, 48, 53, 56},
};
// D – раунд, j – індекс в таблиці циклічного зсуву
void mix (int j, int d) {
  usigned long long rotl;
  y [0] = x [0] + x [1];
```

```
    rotl = R16 [d% 8] [j];
    y [1] = (x [1] << rotl) | (x [1] >> (64 - rotl));
    y [1] ^ = y [0];
  }
</Source>
```

Процедура розшифрування обернена процедурі зашифрування і містить зворотну функцію DEMIX.

Кожен з 72 раундів Threefish-256 і Threefish-512 має чотири MIX перетворення, Threefish-1024 – вісім звернень до MIX функції.

Безпека

За заявою авторів, алгоритм має більш високий рівень безпеки, ніж AES. Існує атака на 25 з 72 раундів Threefish, в той час як для AES – на 6 з 10. Threefish має показник фактора безпеки 2.9, в свою чергу, AES всього 1.7 [3]

Для досягнення повної дифузії, шифру Threefish-256 досить 9 раундів, Threefish-512 – 10 раундів і Threefish-1024 – 11 раундів. Виходячи з цього, 72 і 80 раундів відповідно в середньому, забезпечать кращі результати, ніж існуючі шифри. [4]

У той же час, алгоритм має набагато простішу структуру і функцію перетворення, проте виконання 72-80 раундів, на думку дослідників, забезпечує необхідну стійкість. Вживаний розмір ключа від 256 до 1024 біт зводить нанівець можливість повного перебору паролів при так званій атаці грубою силою (brute force attack) на сучасному обладнанні.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні блоки:

– Блок контексного меню.

– Блок кнопок швидкого доступу.

– Блок вибору дій.

Блок вибору дій включає в себе наступні елементи:

– Сканування ресурсів.

– Сканування трафіку.

– Веб-захист браузерів.

– Журнал подій.

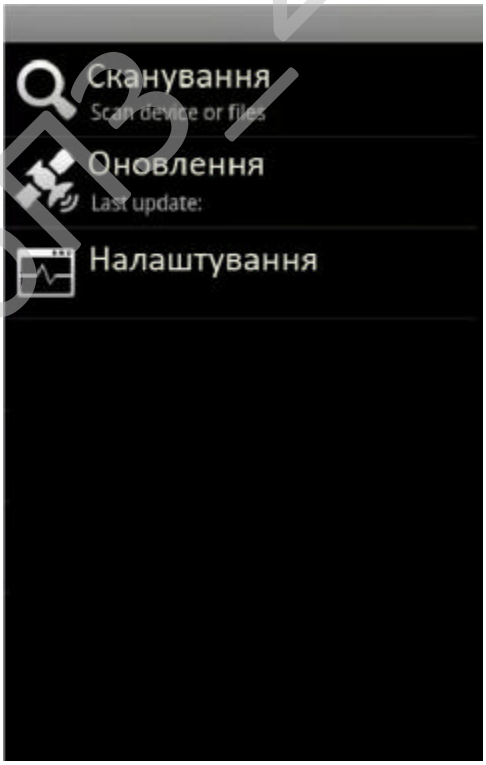

Рисунок 5.1 – Вікна програми

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

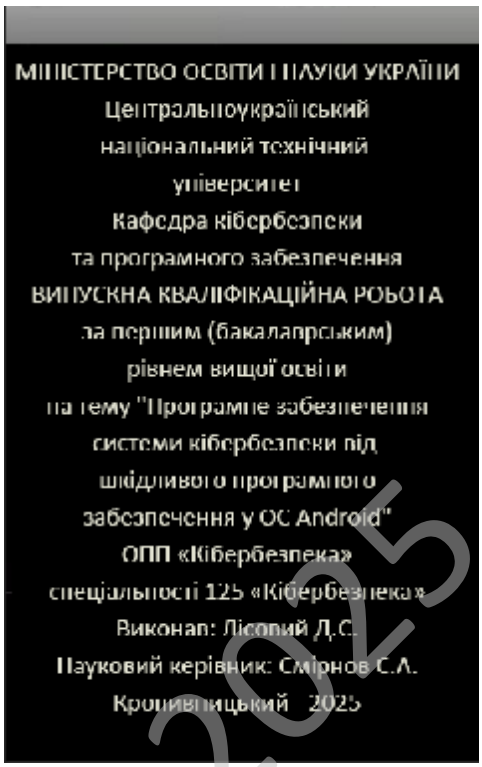

Рисунок 5.2 – Авторські дані розробленого програмного забезпечення

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

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли

програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем від шкідливого програмного забезпечення у ОС Android.

– Досліджена система від шкідливого програмного забезпечення у ОС Android.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання від шкідливого програмного забезпечення у ОС Android.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Threefish.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

25. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

26. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

27. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

28. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

29. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

30. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

31. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

32. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

33. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

34. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

35. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

36. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

37. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

38. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

39. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

40. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

41. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

42. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

44. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising

Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

45. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

46. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

47. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки від шкідливого програмного забезпечення у ОС Android;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на мобільному пристрої під керуванням ОС Android і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Android.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 75 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

Програмне забезпечення системи кібербезпеки від шкідливого програмного забезпечення у ОС Android

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 18

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
# Програма системи кібербезпеки для ОС Android проти шкідливого програмного
забезпечення

import os
import sys
import time
import datetime
import requests
import threading
import random
import logging
import re

# Налаштування логування
logging.basicConfig(filename='android_security.log', level=logging.DEBUG,
format='%(asctime)s - %(levelname)s - %(message)s')

# Клас системи кібербезпеки для ОС Android
class AndroidSecuritySystem:
    def __init__(self):
        # Ініціалізація змінних класу
        self.malware_signatures = []
        self.installed_apps = []
        self.scan_directories = []
        self.alert_recipients = []
        self.quarantine_directory = "quarantine"
        self.update_url = "http://example.com/api/update_signatures"
        self.simulation_mode = True
        # Виклик методу ініціалізації системи
        self.initialize_system()

    def initialize_system(self):
        # Ініціалізація системи кібербезпеки
        if not os.path.exists(self.quarantine_directory):
            os.makedirs(self.quarantine_directory)
        # Завантаження підписів шкідливого ПЗ
        self.load_malware_signatures()
        # Завантаження списку встановлених додатків
        self.load_installed_apps()
        # Встановлення каталогів для сканування
        self.scan_directories = ["/data/data", "/sdcard"]
        # Налаштування отримувачів оповіщень
        self.alert_recipients = ["admin@example.com"]
        logging.info("System initialized successfully.")

    def load_malware_signatures(self):
        # Завантаження базових підписів шкідливого ПЗ
        self.malware_signatures = [
            r"malicious_code",
            r"trojan_activity",
            r"virus_payload",
            r"ransomware_lock",
            r"spyware_monitor"
        ]
        logging.info("Malware signatures loaded.")

    def load_installed_apps(self):
        # Завантаження інформації про встановлені додатки
        self.installed_apps = [
            {"name": "SafeApp", "package": "com.safe.app", "version": "1.0.0"},
            {"name": "ChatSecure", "package": "com.chat.secure", "version":
"2.3.1"},
            {"name": "UnknownApp", "package": "com.unknown.app", "version":
"0.0.1"},
            {"name": "GameFun", "package": "com.game.fun", "version": "5.2.0"}
        ]
```

```
        logging.info("Installed apps loaded.")

    def update_signatures(self):
        # Оновлення підписів шкідливого ПЗ з віддаленого сервера
        try:
            response = requests.get(self.update_url, timeout=5)
            if response.status_code == 200:
                updated_signatures = response.json().get("signatures", [])
                if updated_signatures:
                    self.malware_signatures.extend(updated_signatures)
                    logging.info("Signatures updated from remote server.")
                else:
                    logging.warning("No new signatures found.")
            else:
                logging.error("Failed to update signatures. Status code: " +
str(response.status_code))
        except Exception as e:
            logging.error("Exception during signature update: " + str(e))

    def scan_directory(self, directory):
        # Сканування заданого каталогу на наявність шкідливих файлів
        detected_files = []
        for root, dirs, files in os.walk(directory):
            for file in files:
                file_path = os.path.join(root, file)
                try:
                    with open(file_path, "r", errors="ignore") as f:
                        content = f.read()
                        for signature in self.malware_signatures:
                            if re.search(signature, content):
                                detected_files.append(file_path)
                                logging.info("Malware detected in file: " +
file_path)
                                break
                except Exception as e:
                    logging.error("Error scanning file " + file_path + ": " +
str(e))
        return detected_files

    def scan_installed_apps(self):
        # Сканування встановлених додатків на підозрілу активність
        suspicious_apps = []
        for app in self.installed_apps:
            if "unknown" in app["name"].lower() or "0.0.1" in app["version"]:
                suspicious_apps.append(app)
                logging.info("Suspicious app detected: " + app["name"])
        return suspicious_apps

    def detect_suspicious_activity(self):
        # Детекція підозрілої активності в системі
        suspicious_events = []
        current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        event = {"time": current_time, "event": "Unusual network activity
detected"}
        suspicious_events.append(event)
        logging.info("Suspicious activity detected: " + str(event))
        return suspicious_events

    def quarantine_file(self, file_path):
        # Переміщення файлу до каталогу карантину
        try:
            quarantine_path = os.path.join(self.quarantine_directory,
os.path.basename(file_path))
            os.rename(file_path, quarantine_path)
            logging.info("File quarantined: " + file_path)
        except Exception as e:
            logging.error("Failed to quarantine file " + file_path + ": " +
str(e))
```

```
    def send_alert(self, message):
        # Відправка оповіщення на задані контакти
        for recipient in self.alert_recipients:
            logging.info("Alert sent to " + recipient + ": " + message)
            time.sleep(0.1)

    def perform_full_scan(self):
        # Виконання повного сканування системи за всіма параметрами
        all_detected_files = []
        for directory in self.scan_directories:
            detected = self.scan_directory(directory)
            if detected:
                all_detected_files.extend(detected)
        suspicious_apps = self.scan_installed_apps()
        suspicious_events = self.detect_suspicious_activity()
        if all_detected_files:
            for file_path in all_detected_files:
                self.quarantine_file(file_path)
                self.send_alert("Malware detected and file quarantined: " +
file_path)
        if suspicious_apps:
            for app in suspicious_apps:
                self.send_alert("Suspicious app detected: " + app["name"])
        if suspicious_events:
            for event in suspicious_events:
                self.send_alert("Suspicious event: " + event["event"])
        logging.info("Full scan completed.")

# Функція для симуляції HTTP запиту до віддаленого сервера
def simulate_http_request(url, params=None):
    try:
        response = requests.get(url, params=params, timeout=5)
        logging.info("HTTP request successful to URL: " + url)
        return response.text
    except Exception as e:
        logging.error("HTTP request failed to URL: " + url + " Error: " +
str(e))
        return None

# Функція для аналізу вмісту файлу на предмет шкідливих підписів
def analyze_file_content(file_path, signatures):
    try:
        with open(file_path, "r", errors="ignore") as f:
            content = f.read()
            for signature in signatures:
                if re.search(signature, content):
                    logging.info("Signature " + signature + " found in file: " +
file_path)
                    return True
        return False
    except Exception as e:
        logging.error("Error reading file " + file_path + ": " + str(e))
        return False

# Функція для симуляції аналізу мережевого трафіку
def scan_network_traffic():
    simulated_traffic = ["normal", "normal", "suspicious", "normal",
"suspicious"]
    detected = []
    for traffic in simulated_traffic:
        if traffic == "suspicious":
            detected.append("Suspicious network packet detected")
            logging.info("Suspicious network packet detected in simulation.")
    return detected

# Функція для аналізу системних логів на предмет аномалій
def analyze_system_logs():
    logs = [
        "System boot completed",
```

```
        "User login successful",
        "Unusual access pattern detected",
        "Malware signature match found",
        "Network anomaly detected"
    ]
    anomalies = []
    for log in logs:
        if "anomaly" in log.lower() or "malware" in log.lower():
            anomalies.append(log)
            logging.info("Anomaly found in log: " + log)
    return anomalies

# Функція для перевірки активних процесів на предмет шкідливих дій
def check_running_processes():
    processes = ["systemd", "com.android.system", "malicious_process",
"com.android.launcher"]
    detected_processes = []
    for proc in processes:
        if "malicious" in proc:
            detected_processes.append(proc)
            logging.info("Suspicious process detected: " + proc)
    return detected_processes

# Функція для ініціалізації багатопотокового сканування
def start_multithreaded_scan(security_system):
    threads = []
    thread1 = threading.Thread(target=security_system.perform_full_scan)
    threads.append(thread1)
    thread2 = threading.Thread(target=simulate_http_request,
args=("http://example.com/api/check",))
    threads.append(thread2)
    thread3 = threading.Thread(target=scan_network_traffic)
    threads.append(thread3)
    thread4 = threading.Thread(target=analyze_system_logs)
    threads.append(thread4)
    thread5 = threading.Thread(target=check_running_processes)
    threads.append(thread5)
    for t in threads:
        t.start()
        logging.info("Thread started: " + str(t.name))
        time.sleep(0.1)
    for t in threads:
        t.join()
        logging.info("Thread joined: " + str(t.name))
    logging.info("Multithreaded scan completed.")

# Функція для симуляції перевірки SMS повідомлень на підозрілі повідомлення
def check_sms_messages():
    sms_logs = [
        "Hello, how are you?",
        "Urgent: Click here for prize",
        "Meeting at 5 PM",
        "Free gift waiting for you, click now"
    ]
    suspicious_sms = []
    for sms in sms_logs:
        if "click" in sms.lower() or "prize" in sms.lower():
            suspicious_sms.append(sms)
            logging.info("Suspicious SMS detected: " + sms)
    return suspicious_sms

# Функція для аналізу журналу дзвінків на предмет підозрілих дзвінків
def check_call_logs():
    call_logs = [
        "Incoming call from +123456789",
        "Missed call from unknown",
        "Call from spam number +1987654321",
        "Call from friend"
    ]
```

```
    suspicious_calls = []
    for call in call_logs:
        if "spam" in call.lower() or "unknown" in call.lower():
            suspicious_calls.append(call)
            logging.info("Suspicious call log: " + call)
    return suspicious_calls

# Функція для аналізу використання батареї на предмет аномалій
def analyze_battery_usage():
    battery_levels = [random.randint(10, 100) for _ in range(10)]
    anomalies = []
    for level in battery_levels:
        if level < 20:
            anomalies.append("Low battery level: " + str(level) + "%")
            logging.info("Battery anomaly detected: " + str(level) + "%")
    return anomalies

# Функція для аналізу даних сенсорів пристрою
def analyze_sensor_data():
    sensor_readings = {
        "accelerometer": [random.uniform(-5, 5) for _ in range(10)],
        "gyroscope": [random.uniform(-3, 3) for _ in range(10)]
    }
    anomalies = []
    for sensor, readings in sensor_readings.items():
        for reading in readings:
            if abs(reading) > 4:
                anomalies.append("Anomaly in " + sensor + ": " + str(reading))
                logging.info("Sensor anomaly detected in " + sensor + ": " +
str(reading))
    return anomalies

# Функція для перевірки конфігурації пристрою на предмет вразливостей
def check_device_configuration():
    configuration = {
        "developer_mode": False,
        "unknown_sources": False,
        "os_version": "Android 11",
        "security_patch": "2025-02-15"
    }
    vulnerabilities = []
    if configuration["developer_mode"]:
        vulnerabilities.append("Developer mode is enabled.")
        logging.info("Device configuration issue: Developer mode enabled.")
    if configuration["unknown_sources"]:
        vulnerabilities.append("Installation from unknown sources allowed.")
        logging.info("Device configuration issue: Unknown sources allowed.")
    return vulnerabilities

# Функція для виконання періодичного сканування протягом заданого часу
def periodic_scan(security_system, interval, duration):
    start_time = time.time()
    while time.time() - start_time < duration:
        logging.info("Starting periodic scan iteration.")
        security_system.perform_full_scan()
        time.sleep(interval)

# Функція для симуляції загальної перевірки стану системи
def simulate_system_health_check(security_system):
    sms_issues = check_sms_messages()
    call_issues = check_call_logs()
    battery_anomalies = analyze_battery_usage()
    sensor_anomalies = analyze_sensor_data()
    config_vulnerabilities = check_device_configuration()
    all_issues = sms_issues + call_issues + battery_anomalies + sensor_anomalies
+ config_vulnerabilities
    if all_issues:
        for issue in all_issues:
            security_system.send_alert("Health Check Alert: " + issue)
```

```
    logging.info("System health check completed.")

# Функція для симуляції поведінки шкідливого ПЗ
def simulate_malware_behavior():
    behaviors = [
        "Attempting to access contacts",
        "Trying to send SMS without user consent",
        "Exfiltrating data to remote server",
        "Modifying system settings"
    ]
    detected_behavior = random.choice(behaviors)
    logging.info("Simulated malware behavior: " + detected_behavior)
    return detected_behavior

# Функція для запуску періодичних завдань моніторингу системи
def start_periodic_tasks(security_system):
    periodic_thread = threading.Thread(target=periodic_scan,
args=(security_system, 5, 20))
    health_check_thread = threading.Thread(target=simulate_system_health_check,
args=(security_system,))
    malware_behavior_thread = threading.Thread(target=simulate_malware_behavior)
    periodic_thread.start()
    health_check_thread.start()
    malware_behavior_thread.start()
    periodic_thread.join()
    health_check_thread.join()
    malware_behavior_thread.join()
    logging.info("Periodic tasks completed.")

# Функція для симуляції взаємодії користувача із системою
def simulate_user_interaction(security_system):
    commands = ["scan", "update", "status", "help", "exit"]
    for i in range(3):
        command = random.choice(commands)
        if command == "scan":
            security_system.perform_full_scan()
            logging.info("User command executed: scan")
        elif command == "update":
            security_system.update_signatures()
            logging.info("User command executed: update")
        elif command == "status":
            logging.info("User command executed: status")
        elif command == "help":
            logging.info("User command executed: help")
        elif command == "exit":
            logging.info("User command executed: exit")
        time.sleep(0.5)

# Функція для симуляції віддаленої перевірки файлу
def simulate_remote_verification(file_path):
    verification_url = "http://example.com/api/verify"
    params = {"file": file_path}
    result = simulate_http_request(verification_url, params)
    logging.info("Remote verification result for " + file_path + ": " +
str(result))
    return result

# Функція для перевірки виявлених файлів через віддалену перевірку
def verify_detected_files(security_system):
    for directory in security_system.scan_directories:
        detected_files = security_system.scan_directory(directory)
        for file in detected_files:
            simulate_remote_verification(file)

# Головна функція для запуску системи кібербезпеки
def main():
    security_system = AndroidSecuritySystem()
    security_system.update_signatures()
    start_multithreaded_scan(security_system)
```

```
    extra_network_issues = scan_network_traffic()
    if extra_network_issues:
        for issue in extra_network_issues:
            security_system.send_alert(issue)
    extra_log_anomalies = analyze_system_logs()
    if extra_log_anomalies:
        for anomaly in extra_log_anomalies:
            security_system.send_alert("Log anomaly: " + anomaly)
    extra_processes = check_running_processes()
    if extra_processes:
        for proc in extra_processes:
            security_system.send_alert("Suspicious process: " + proc)
    for directory in security_system.scan_directories:
        detected_files = security_system.scan_directory(directory)
        if detected_files:
            for file in detected_files:
                security_system.quarantine_file(file)
                security_system.send_alert("File quarantined after scan: " +
file)
    simulate_user_interaction(security_system)
    verify_detected_files(security_system)
    start_periodic_tasks(security_system)
    logging.info("Main security operations completed.")

# Виклик головної функції при запуску програми
if __name__ == '__main__':
    main()
# Завершення програми системи кібербезпеки для ОС Android
```

Файл ExtendedNetworkAnalyzer.py

```
import os
import sys
import time
import threading
import random
import datetime
import json
import socket
import hashlib
import requests
import numpy as np
from sklearn.ensemble import IsolationForest

class ExtendedNetworkAnalyzer:
    def __init__(self):
        self.packets = []
        self.anomalies = []
    def simulate_packet_capture(self, num_packets=100):
        for i in range(num_packets):
            packet = {"src": "192.168.1." + str(random.randint(1,254)), "dst":
"10.0.0." + str(random.randint(1,254)), "length": random.randint(20,1500),
"protocol": random.choice(["TCP", "UDP", "ICMP"]), "payload":
os.urandom(random.randint(20,100)).hex(), "timestamp": time.time()}
            self.packets.append(packet)
            time.sleep(0.01)
    def analyze_packets(self):
        threshold = 1400
        for pkt in self.packets:
            if pkt["length"] > threshold:
                self.anomalies.append(pkt)
        return self.anomalies
    def summarize_traffic(self):
        total_packets = len(self.packets)
        total_anomalies = len(self.anomalies)
        summary = {"total_packets": total_packets, "total_anomalies":
total_anomalies}
        return summary
    def run_analysis(self):
        self.simulate_packet_capture()
        self.analyze_packets()
        summary = self.summarize_traffic()
        print("Network Traffic Summary:", summary)
        if self.anomalies:
            print("Anomalies detected:")
            for anomaly in self.anomalies:
                print(anomaly)

class DeepFileScanner:
    def __init__(self, base_directory):
        self.base_directory = base_directory
        self.malicious_files = []
```

```
        self.known_patterns = ["malware", "trojan", "virus", "spyware",
"ransom"]
    def get_file_hash(self, file_path):
        hash_md5 = hashlib.md5()
        try:
            with open(file_path, "rb") as f:
                for chunk in iter(lambda: f.read(4096), b""):
                    hash_md5.update(chunk)
            return hash_md5.hexdigest()
        except:
            return None
    def check_file_for_threats(self, file_path):
        try:
            with open(file_path, "r", errors="ignore") as f:
                content = f.read().lower()
                for pattern in self.known_patterns:
                    if pattern in content:
                        return True
            return False
        except:
            return False
    def scan_directory(self, current_directory):
        try:
            for entry in os.listdir(current_directory):
                path = os.path.join(current_directory, entry)
                if os.path.isdir(path):
                    self.scan_directory(path)
                elif os.path.isfile(path):
                    threat_found = self.check_file_for_threats(path)
                    if threat_found:
                        file_hash = self.get_file_hash(path)
                        self.malicious_files.append({"path": path, "hash":
file_hash})
        except:
            pass
    def run_scan(self):
        self.scan_directory(self.base_directory)
        print("Deep File Scan Results:")
        for file_info in self.malicious_files:
            print(file_info)

class AppMonitor:
    def __init__(self):
        self.running_apps = []
        self.anomalies = []
        self.monitoring = True
    def simulate_app_list(self):
        apps = [{"name": "App" + str(i), "cpu": random.uniform(0,100), "memory":
random.uniform(50,500)} for i in range(10)]
        self.running_apps = apps
    def monitor_apps(self):
        while self.monitoring:
            self.simulate_app_list()
```

```
            for app in self.running_apps:
                if app["cpu"] > 80 or app["memory"] > 400:
                    self.anomalies.append(app)
            time.sleep(1)
    def start_monitoring(self):
        monitor_thread = threading.Thread(target=self.monitor_apps)
        monitor_thread.daemon = True
        monitor_thread.start()
    def report_anomalies(self):
        print("Real-Time Application Monitoring Report:")
        if self.anomalies:
            for anomaly in self.anomalies:
                print(anomaly)
        else:
            print("No anomalies detected.")

class RemoteManager:
    def __init__(self, host="localhost", port=9999):
        self.host = host
        self.port = port
        self.server_running = False
    def start_server(self):
        self.server_running = True
        server_thread = threading.Thread(target=self.run_server)
        server_thread.daemon = True
        server_thread.start()
    def run_server(self):
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.bind((self.host, self.port))
        server_socket.listen(5)
        while self.server_running:
            client_socket, addr = server_socket.accept()
            client_thread = threading.Thread(target=self.handle_client,
args=(client_socket,))
            client_thread.daemon = True
            client_thread.start()
    def handle_client(self, client_socket):
        data = client_socket.recv(1024).decode("utf-8")
        response = self.process_command(data)
        client_socket.send(response.encode("utf-8"))
        client_socket.close()
    def process_command(self, command):
        command = command.strip().lower()
        if command == "status":
            return "System is running"
        elif command == "scan":
            return "Scan initiated"
        elif command == "update":
            return "Update executed"
        elif command == "shutdown":
            self.server_running = False
            return "Server shutting down"
        else:
```

```
            return "Unknown command"
    def run_client(self, command, host="localhost", port=9999):
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_socket.connect((host, port))
        client_socket.send(command.encode("utf-8"))
        response = client_socket.recv(1024).decode("utf-8")
        client_socket.close()
        return response

class ThreatDetectionML:
    def __init__(self):
        self.model = None
        self.training_data = None
        self.labels = None
    def load_data(self):
        data = np.random.rand(100, 5)
        labels = np.random.randint(0, 2, 100)
        self.training_data = data
        self.labels = labels
    def train_model(self):
        self.load_data()
        self.model = IsolationForest(contamination=0.1, random_state=42)
        self.model.fit(self.training_data)
    def predict_threat(self, sample):
        if self.model is None:
            self.train_model()
        prediction = self.model.predict(np.array(sample).reshape(1, -1))
        if prediction[0] == -1:
            return True
        return False
    def update_model(self, new_data):
        if self.training_data is not None:
            self.training_data = np.concatenate((self.training_data, new_data),
axis=0)
            self.model.fit(self.training_data)
        else:
            self.training_data = new_data
            self.train_model()
    def run_detection(self):
        test_samples = [np.random.rand(5) for _ in range(10)]
        threats = []
        for sample in test_samples:
            if self.predict_threat(sample):
                threats.append(sample.tolist())
        print("Threat Detection ML Report:")
        if threats:
            for threat in threats:
                print("Threat sample:", threat)
        else:
            print("No threats detected.")

def main():
    network_analyzer = ExtendedNetworkAnalyzer()
```

```
    network_thread = threading.Thread(target=network_analyzer.run_analysis)
    network_thread.start()
    deep_scanner = DeepFileScanner(".")
    scanner_thread = threading.Thread(target=deep_scanner.run_scan)
    scanner_thread.start()
    app_monitor = AppMonitor()
    app_monitor.start_monitoring()
    remote_manager = RemoteManager()
    remote_manager.start_server()
    time.sleep(2)
    response = remote_manager.run_client("status")
    print("Remote Manager Response:", response)
    threat_ml = ThreatDetectionML()
    threat_ml.train_model()
    threat_ml.run_detection()
    for i in range(5):
        print("App Monitor Iteration", i)
        app_monitor.report_anomalies()
        time.sleep(1)
    time.sleep(2)
    print("Main execution complete.")

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

Файл FirewallProtection.py

```
import os
import sys
import time
import threading
import random
import socket
import hashlib
import requests
import json
import logging
import psutil

class FirewallProtection:
    def __init__(self):
        self.blocked_ips = set()
        self.log_entries = []
        self.monitoring = True
        self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server_socket.bind(('0.0.0.0', 8888))
        self.server_socket.listen(5)
        self.lock = threading.Lock()
    def start_firewall(self):
        self.monitoring = True
        self.firewall_thread = threading.Thread(target=self.run_firewall)
        self.firewall_thread.daemon = True
        self.firewall_thread.start()
    def stop_firewall(self):
        self.monitoring = False
        try:
            self.server_socket.close()
        except:
            pass
    def run_firewall(self):
        while self.monitoring:
            try:
                client_socket, addr = self.server_socket.accept()
                ip = addr[0]
                if self.is_ip_blocked(ip):
                    client_socket.send(b"Connection blocked")
                    client_socket.close()
                else:
                    data = client_socket.recv(1024)
                    self.process_data(ip, data)
                    client_socket.send(b"Connection allowed")
                    client_socket.close()
            except Exception as e:
                time.sleep(0.1)
    def is_ip_blocked(self, ip):
        with self.lock:
            return ip in self.blocked_ips
```

```
    def block_ip(self, ip):
        with self.lock:
            self.blocked_ips.add(ip)
            self.log_entries.append("Blocked IP: " + ip)
    def process_data(self, ip, data):
        if data and len(data) > 50:
            self.block_ip(ip)
        self.log_entries.append("Data received from " + ip + ": " + str(data))
    def get_logs(self):
        return self.log_entries

class AutomaticSignatureUpdater:
    def __init__(self, update_url):
        self.update_url = update_url
        self.signatures = []
        self.update_interval = 10
        self.updating = True
    def start_updating(self):
        self.updating = True
        self.update_thread = threading.Thread(target=self.run_update_loop)
        self.update_thread.daemon = True
        self.update_thread.start()
    def stop_updating(self):
        self.updating = False
    def run_update_loop(self):
        while self.updating:
            new_signatures = self.fetch_signatures()
            if new_signatures:
                self.signatures = list(set(self.signatures + new_signatures))
            time.sleep(self.update_interval)
    def fetch_signatures(self):
        try:
            response = requests.get(self.update_url, timeout=5)
            if response.status_code == 200:
                data = response.json()
                return data.get("signatures", [])
            return []
        except:
            return []
    def get_signatures(self):
        return self.signatures

class SystemLogAnalyzer:
    def __init__(self, log_file):
        self.log_file = log_file
        self.anomalies = []
        self.analysis_interval = 5
        self.running = True
    def start_analysis(self):
        self.running = True
        self.analysis_thread = threading.Thread(target=self.run_analysis_loop)
        self.analysis_thread.daemon = True
        self.analysis_thread.start()
```

```
    def stop_analysis(self):
        self.running = False
    def run_analysis_loop(self):
        while self.running:
            logs = self.read_logs()
            self.analyze_logs(logs)
            time.sleep(self.analysis_interval)
    def read_logs(self):
        try:
            with open(self.log_file, "r") as f:
                lines = f.readlines()
            return lines
        except:
            return []
    def analyze_logs(self, logs):
        for line in logs:
            if "error" in line.lower() or "failed" in line.lower():
                self.anomalies.append(line.strip())
    def get_anomalies(self):
        return self.anomalies
    def write_analysis_report(self, report_file):
        try:
            with open(report_file, "w") as f:
                for anomaly in self.anomalies:
                    f.write(anomaly + "\n")
        except:
            pass

class ResourceUsageController:
    def __init__(self):
        self.cpu_threshold = 80.0
        self.memory_threshold = 80.0
        self.disk_threshold = 90.0
        self.battery_threshold = 20
        self.alerts = []
        self.monitoring = True
    def start_monitoring(self):
        self.monitoring = True
        self.monitor_thread = threading.Thread(target=self.run_monitor)
        self.monitor_thread.daemon = True
        self.monitor_thread.start()
    def stop_monitoring(self):
        self.monitoring = False
    def run_monitor(self):
        while self.monitoring:
            self.check_cpu()
            self.check_memory()
            self.check_disk()
            self.check_battery()
            time.sleep(2)
    def check_cpu(self):
        cpu_usage = psutil.cpu_percent(interval=1)
        if cpu_usage > self.cpu_threshold:
```

```
            self.alerts.append("High CPU usage: " + str(cpu_usage) + "%")
    def check_memory(self):
        memory = psutil.virtual_memory()
        if memory.percent > self.memory_threshold:
            self.alerts.append("High Memory usage: " + str(memory.percent) +
"%")
    def check_disk(self):
        disk = psutil.disk_usage('/')
        if disk.percent > self.disk_threshold:
            self.alerts.append("High Disk usage: " + str(disk.percent) + "%")
    def check_battery(self):
        if hasattr(psutil, "sensors_battery"):
            battery = psutil.sensors_battery()
            if battery and battery.percent < self.battery_threshold:
                self.alerts.append("Low Battery level: " + str(battery.percent)
+ "%")
    def get_alerts(self):
        return self.alerts

class CloudSecurityIntegration:
    def __init__(self, cloud_endpoint):
        self.cloud_endpoint = cloud_endpoint
        self.upload_interval = 15
        self.logs_to_upload = []
        self.running = True
    def add_log(self, log_entry):
        self.logs_to_upload.append(log_entry)
    def start_integration(self):
        self.running = True
        self.integration_thread =
threading.Thread(target=self.run_integration_loop)
        self.integration_thread.daemon = True
        self.integration_thread.start()
    def stop_integration(self):
        self.running = False
    def run_integration_loop(self):
        while self.running:
            if self.logs_to_upload:
                self.upload_logs()
            time.sleep(self.upload_interval)
    def upload_logs(self):
        data = {"logs": self.logs_to_upload}
        try:
            response = requests.post(self.cloud_endpoint, json=data, timeout=5)
            if response.status_code == 200:
                self.logs_to_upload = []
        except:
            pass
    def fetch_commands(self):
        try:
            response = requests.get(self.cloud_endpoint + "/commands",
timeout=5)
            if response.status_code == 200:
```

```
                commands = response.json().get("commands", [])
                return commands
            return []
        except:
            return []
    def process_commands(self):
        commands = self.fetch_commands()
        for cmd in commands:
            print("Executing cloud command:", cmd)

def main():
    firewall = FirewallProtection()
    firewall.start_firewall()
    signature_updater =
AutomaticSignatureUpdater("http://example.com/api/signatures")
    signature_updater.start_updating()
    log_analyzer = SystemLogAnalyzer("system.log")
    log_analyzer.start_analysis()
    resource_controller = ResourceUsageController()
    resource_controller.start_monitoring()
    cloud_integration =
CloudSecurityIntegration("http://cloudsecurity.example.com/upload")
    cloud_integration.start_integration()
    for i in range(20):
        try:
            with open("system.log", "a") as f:
                f.write("Log entry " + str(i) + " error occurred\n")
        except:
            pass
        cloud_integration.add_log("Log entry " + str(i))
        time.sleep(1)
    anomalies = log_analyzer.get_anomalies()
    alerts = resource_controller.get_alerts()
    fw_logs = firewall.get_logs()
    print("System Log Analyzer Anomalies:")
    for a in anomalies:
        print(a)
    print("Resource Usage Alerts:")
    for a in alerts:
        print(a)
    print("Firewall Logs:")
    for log in fw_logs:
        print(log)
    cloud_integration.process_commands()
    signature_updater.stop_updating()
    log_analyzer.stop_analysis()
    resource_controller.stop_monitoring()
    cloud_integration.stop_integration()
    firewall.stop_firewall()

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