НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
В.о. завідувача кафедри

_____**Микола ГРАЙВОРОНСЬКИЙ**
(підпис)

«_____»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною программою «Системи, технології та математичні методи кібербезпеки»
спеціальності: 125 «Кібербезпека»

на тему: «Методика сигнатурного антивірусного захисту з застосуванням сканерів.

Виконав: здобувач вищої освіти **IV** курсу, групи ФБ- зп81
(шифр групи)

Бойко Яков Анатолійович
прізвище, ім'я, по батькові) (підпис)

Керівник Гальчинський Л.Ю. доцент НТУУ «КПІ ім. Ігора Сікорського», к.т.н. , доцент,

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

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

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ - 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО"

Дипломна робота на тему: «Методика сигнатурного антивірусного захисту з застосуванням сканерів»

Виконав: студент гр. ЗІБ-зп81 Бойко Я.А.

Перевірив: Гальчинський Л.Ю., к.т.н, доцент

Київ 2020

ЗАВДАННЯ

Завдання проекту полягає у наступному:

- аналіз літературних джерел стосовно протидії вірусним загрозам;
- дослідження методики сигнатурного антивірусного захисту з застосуванням сканерів;
- побудова прототипу антивірусної програми за допомогою підключення власних та запозичених сигнатур (github) для виявлення шкідливих програмних засобів.
- використання шістнадцяткового редактора та інструменту «YARA» для сканування файлів та програм.
- провести дослідження основних антивірусних програм;
- тестування прототипу антивірусної програми.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
	Визначення теми та мети дослідження	15.04.20 – 20.04.20	виконано
	Опрацювання літературних джерел	20.04.20 – 30.04.20	виконано
	Визначення інструменту створення прототипу антивірусної програми	30.04.20 – 5.05.20	виконано
	Створення сигнатурного аналізатора	5.05.20 – 10.05.20	виконано
	Оформлення дипломної роботи	10.05.20 – 29.05.20	виконано

Реферат

Робота обсягом 89 сторінок містить 5 ілюстрацій, 12 таблиць та 4 додатки.

Об’єктом дослідження є методика сигнатурного антивірусного захисту з застосуванням сканерів.

Предметом дослідження виступає процес дослідження методики сигнатурного антивірусного захисту з застосуванням сканерів.

Результати роботи представлені у вигляді рисунків, що характеризують кроки сканування сигнатури, використання текстових шаблонів для перевірки файлу, часовий проміжок між створенням вірусу та скануванням його підпису тощо.

Отримані результати мають практичне застосування при виборі різних типів антивірусного захисту, а також розуміння їх переваг та недоліків, використанні сигнатурного підходу до виявлення вірусів, створенні прототипів антивірусних програм та підключенні різних шаблонів для знаходження шкідливого програмного забезпечення.

Антивірусний захист, інформаційна безпека, сигнатура, антивірусна програма, сигнатурний аналізатор, вірусний сканер, шкідливе програмне забезпечення.

ABSTRACT

The work includes 64 pages, 5 figures, 1 table 36 literary references and 2 appendices.

The object of research is the method of signature anti-virus protection using scanners.

The subject of the study is the process of studying the methodology of signature anti-virus protection using scanners.

The results are presented in the form of figures that characterize the steps of scanning the signature, the use of text templates to scan the file, the time gap between the creation of the virus and scanning its signature, and so on.

The obtained results have practical application in choosing different types of antivirus protection, as well as understanding their advantages and disadvantages, using a signature approach to virus detection, prototyping antivirus programs and connecting various templates to find malware.

Antivirus protection, information security, signature, antivirus program, signature analyzer, virus scanner, malware.

.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – прикладни́й програмний інтерфейс (інтерфейс програмування застосунків, інтерфейс прикладного програмування), набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

BIOS – напівпостійна пам'ять, також CMOS RAM — пам'ять на материнській платі комп'ютера, призначена для зберігання налаштувань і деяких відомостей про систему.

CD (compact disc) — переносний оптичний диск для збереження інформації (даних) у цифровому вигляді, тобто формату зберігання даних.

CRC (Cyclic redundancy check) — алгоритм обчислення контрольної суми, призначений для перевірки цілісності даних.

DVD (Digital Versatile Disc) — цифровий багатоцільовий диск, носій інформації у вигляді диска, зовні схожий з компакт-диском, однак має можливість зберігати більше інформації за рахунок використання лазера з меншою довжиною хвилі, ніж для звичайних компакт-дисків.

GNU GPL (General Public License)(Загальна публічна ліцензія GNU або Загальна громадська ліцензія GNU) — одна з найпопулярніших ліцензій на вільне програмне забезпечення, створена Річардом Столменом для проекту GNU.

GUI (Graphical user interface) – графічний інтерфейс користувача, тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

LAN (Local Area Network) — локальна мережа, яка являє собою об'єднання певного числа комп'ютерів на відносно невеликій території.

SFC (System File Checker) — утиліта Microsoft Windows, що дозволяє користувачеві знаходити і відновлювати пошкодження системних файлів Windows. Компонент доступний в Windows 98, Windows 2000 і всіх наступних версіях операційних систем сімейства Windows NT.

TCP / IP (TCP – Transmission Control Protocol «протокол керування передаванням» і IP – Internet Protocol — «міжмережевий протокол») — систематизований стек протоколів, що поділяється на чотири рівні, які корелюються з еталонною моделлю OSI: прикладний (application), транспортний (transport), міжмережевий (internet) та рівень доступу до середовища передачі.

VB – засіб розробки програмного забезпечення, створений і підтримуваний корпорацією Microsoft, який складається з мови програмування і середовища розроблення.

VBA (Visual Basic for Applications) – Visual Basic для Додатків, дещо спрощена реалізація мови програмування Visual Basic, вбудована в лінійку продуктів Microsoft Office (включаючи версії для Mac OS), а також в багато інших програмних пакетів, такі як AutoCAD, WordPerfect і ESRI ArcGIS.

WAN (Wide Area Network) — комп'ютерна мережа, що охоплює величезні території (тобто будь-яка мережа, чиї комунікації поєднують цілі мегаполіси, області або навіть держави і містять у собі десятки, сотні а то і мільйони комп'ютерів).

ВСТУП

З появою веб-технологій комп'ютер починають використовувати абсолютно нові верстви населення Землі. В даний час комп'ютер міцно увійшов у повсякденне життя. Його можливості використовуються на роботі, при проведенні дозвілля, в побуті та інших сферах життя людини.

Сьогодні неможливо зустріти користувача персонального комп'ютера, який не чув би про комп'ютерні віруси. В Інтернеті такі шкідливі програми існують у величезній кількості. Багато розповсюджувачів зловмисного програмного забезпечення успішно застосовують у своїй практиці передові досягнення ІТ-індустрії. У результаті те, що має служити на благо користувачів, в кінцевому підсумку може обернутися для них великими проблемами.

Незважаючи на великі зусилля як теоретичного, так і практичного характеру щодо вирішення проблеми захисту інформації, його сучасний стан дуже далекий від будь-якого надійного розв'язання. Більше того, виникла ціла низка нових проблем, які пов'язані з захистом від комп'ютерних вірусів. Фахівці навіть кардинально змінюють свої погляду на проблему організації антивірусного захисту з урахуванням загроз від проникнення шкідливих програм.

Віруси роблять неможливим нормальне функціонування комп'ютера, тому, безперечно, для лікування програм, знищення вірусів і профілактики постійно потрібні нові антивірусні заходи та системи виявлення вторгнень.

Після цієї вірусної пандемії використовується велика кількість різних методів вирішення даної проблеми. Такі підходи містять як сильні, так і слабкі сторони. За оцінками фахівців в даній галузі великої ефективності набули методи, які використовують сигнатурний підхід. Для забезпечення безперебійної роботи сигнатурних сканерів в антивірусних програмах та систем виявлення вторгнень необхідна постійна підтримка баз даних, які відповідають за зразки шкідливого програмного забезпечення.

При розробці антивірусної стратегії необхідно пам'ятати, що основними шляхами поширення комп'ютерних вірусів є шкідливе програмне забезпечення, системи електронної пошти в локальних мережах, а також флешки, які використовуються службовцями в їхніх домашніх комп'ютерах. Тому використання сигнатурного антивірусного захисту відіграє ключову роль у забезпеченні надійності та цілісності передачі та збереженні інформації.

На сьогоднішній день більшість комп’ютерів та мереж захищенні антивірусним програмним забезпеченням. В умовах використання сучасних інформаційних технологій – це необхідний чинник існування, що дозволяє захистити інформаційні ресурси та апаратне обладнання.

Метою нашого дослідження є розробка методики сигнатурного антивірусного захисту з застосуванням сканерів.

Об’єктом роботи є методики сигнатурного антивірусного захисту з застосуванням сканерів.

Предметом роботи виступає процес дослідження методики сигнатурного антивірусного захисту з застосуванням сканерів.

Структура роботи. Курсова робота складається зі вступу, трьох розділів основної частини, висновків, списку використаної літератури та додатків.

1 ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ВІРУСНОГО СКАНУВАННЯ

1.1 Поняття шкідливого програмного забезпечення

На сьогоднішній день більшість комп’ютерів та мереж захищенні антивірусним програмним забезпеченням. В умовах використання сучасних інформаційних технологій – це необхідний чинник існування, що дозволяє захистити інформаційні ресурси та апаратне обладнання.

Зловмисне програмне забезпечення здійснює величезний вплив на оточуючий світ. Зростаюча кількість інцидентів в сфері комп'ютерної безпеки з 1988 року [7, 8] свідчить про те, що зловмисне програмне забезпечення є епідемією. Занурення у всесвітню павутину з відключеним антивірусом та брандмауером на один день переконає будь-якого з користувачів комп’ютера в широкому використанні та шкідливості таких програм.

У грудні 1999 року один із експериментів був проведений Центром суперкомп'ютерів Сан-Дієго (SDSC) (San Diego Supercomputer Center) [39]. Під час дослідження [39] було встановлено на комп’ютер Red Hat Linux 5.2 без патчів безпеки та під’єднано до Інтернету. Протягом восьми годин після запуску на комп'ютер була здійснена атака. Протягом 21 дня після встановлення комп'ютер зазнав 20 атак. Приблизно через 40 днів після його використання комп'ютер вважався "скомпрометованим". Зловмисне програмне забезпечення може спричинити наслідки, які починаються від спотворення веб-сайтів [39] та закінчуватись втратою людського життя [6].

Комп'ютерні віруси – це створені людиною деструктивні програми, які призначенні для зараження комп'ютерних систем та створюють неприємності їх користувачам. Комп'ютерний вірус, як правило, завантажується на комп'ютерну систему без відома самого користувача. Це викликає несанкціоновані та небажані зміни компонентів комп'ютера або інформації.

Створення вірусів розпочалося в середині 1980-х років, коли використання ПК почало зростати в сфері бізнесу та на індивідуальному рівні. Оскільки комп'ютерні ігри були дуже популярні в той час, а такі програми, як текстові процесори та електронні таблиці, користувалися великим попитом, віруси поширювались за допомогою приєднання до цих ігор та програм. Оскільки дискети переважно використовувались для передачі даних, поширювачі шкідливого програмного забезпечення створювали віруси, які пов’язані з використанням цих носіїв інформації. Пізніше в 1995 році з'явився новий тип вірусу, який називався макровірус. Ці віруси називаються макро-вірусами, оскільки вони написані макро-мовами. Макроси – це невеликі виконувані програми, які входять до складу файлів документів, як Microsoft Word. Пізніші такі віруси поширювалися в основному через електронні листи та Інтернет.

Антивірусні програми націлені на запобігання поширенню та виконанню деструктивних операцій на комп’ютерах. На даний час антивірусні програми стали важливим компонентом для кожного комп'ютера. Антивірусне програмне забезпечення виконує ряд таких важливих функцій, як запобігання дії вірусів та захисту файлів, сканування та виявлення вірусів, видалення вірусів із заражених файлів та відновлення пошкоджених файлів та об’єктів.

Найважливіша функція антивірусної програми полягає в захисті комп'ютера від будь-якого типу атак за допомогою різних типів вірусів: хробаків, троянських коней, шпигунських програм, рекламного та іншого зловмисного програмного забезпечення. При підключенні до комп'ютера зовнішнього запам'ятовуючого пристрою, як CD, DVD, флешка, встановлений антивірус відразу сканує носій інформації та забезпечує блокування передачі вірусу. Антивірусне програмне забезпечення також захищає комп'ютер від вірусних загроз, що надходять з локальної мережі (LAN), WAN або Інтернету.

При встановленні антивірусу спочатку сканується основна пам'ять комп'ютера, а потім усі зовнішні пристрої зберігання інформації для виявлення

вірусів. При знаходженні вірусу антивірусна програма намагається його видалити та відновити заражений файл до початкової стадії. Якщо файл занадто сильно пошкоджений вірусом, то антивірус може видалити його.

1.2 Класифікація методів вірусного сканування

Однією з найважливіших функцій будь-якої антивірусної програми є виявлення наявності вірусу в комп'ютерній системі. Після виявлення вірусу антивірусна програма, як правило, інформує користувача про це. Оскільки вірус часто записує свій код у програму в декількох різних місцях, антивірус намагається видалити його та відновити початкову програму.

Антивірусна програма використовує різні методи виявлення вірусів. Оскільки характеристики вірусів відрізняються, то способи їх виявлення також різні. Усі типи вірусів неможливо виявити жодним єдиним методом. До популярних методів, що застосовуються антивірусними програмами для виявлення вірусів, відносимо наступні: сканування сигнатур, перевірка цілісності, евристичне сканування, емуляція та моніторинг діяльності.

До першого типу виявлення вірусів відносимо просте сигнатурне сканування (Simple signature scanning). Прості комп'ютерні віруси копіюють себе у кожен виконуваний файл, який вони заражають. Дані типи вірусів реплікують свою ідентичну копію по байтах кожен раз, коли вони заражають новий файл. Їх можна легко виявити за допомогою пошуку конкретного рядка байтів, який називається "сигнатурою вірусу", який отримується з тіла вірусу.

Сигнатура вірусу – це послідовність байтів, яка можна знайти лише в коді програми вірусу. Для вилучення сигнатури вірусу дослідникові необхідно ретельно проаналізувати структуру вірусу. Коли з'являється новий вірус, експерт повинен дослідити інфікований файл і знайти шаблон або "сигнатуру" даного вірусу. Після знаходження сигнатури вірусів їх зберігають в спеціальній базі даних антивірусної

програми. Така база даних сигнатур розповсюджується серед клієнтів разом із антивірусною програмою. Антивірусна програма використовує сигнатуру для сканування кожної вразливої області комп'ютера, включаючи виконувані файли, завантажувальні записи, файли документів з макросами тощо для виявлення наявності будь-якої з цих послідовностей. Якщо знайдено будь-який з цих сигнатур, то цільова програма вважається зараженою.

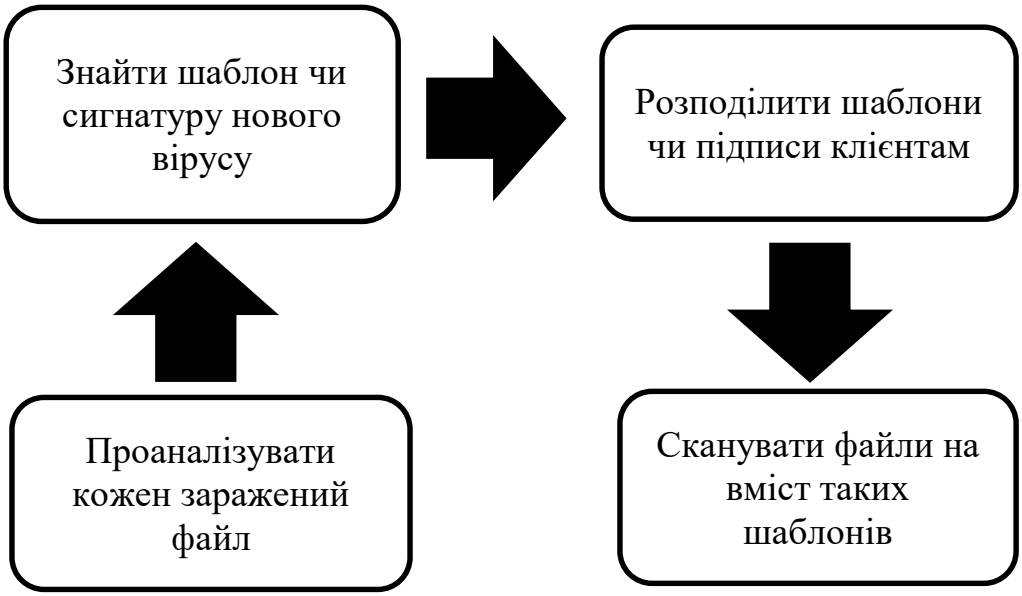


Рис. 1.1 Кроки сканування сигнатури.

Сканування сигнатур становить собою простий і надійний спосіб сортування, але він містить ряд обмежень. До таких обмежень відносимо спроможність виявляти лише відомі віруси, сигнатури яких вже встановлені та включені до бази даних. Сканування не здатне виявити інші варіанти вже відомого вірусу, хоча відмінності між їх сигнатурами незначні.

Сканування загальної сигнатури (Generic signature scanning) вирізняється тим, що використовує шаблон, який знайдений у родині вірусів. Це більш швидкий метод виявлення всіх вірусів, що належать до однієї родини. Такий метод ефективний тому, що більшість вірусів не створюються з самого початку, а формуються зміною коду раніше наявних вірусів. У таких випадках виявляється

багато подібних характеристик між основним вірусом та його варіантами. Сканування загальної сигнатури використовує різні способи для виявлення всіх варіантів родини вірусів. Цей метод також здатний знаходити нові та майбутні віруси тієї ж родини. Сканування загальної сигнатури також називають евристичним скануванням сигнатур.

Перевірка цілісності (Integrity checking) – це ще один метод виявлення вірусів. Даний метод виявляє наявність вірусів за допомогою порівняння хеш-значення файлу з хеш-значенням його незараженої версії. Якщо різниці між двома значеннями хешу не виявлено, файл вважається незараженим. Перевірка цілісності здійснюється за допомогою збереження невеликої «контрольної суми» або «хеш-значення» або «знімку» або «відбитку» незаражених програм (наприклад, виконуваних файлів, завантажувальних записів тощо) у захищеному місці на початку жорсткого диску. Під час перевірки цілісності відбувається зіставлення нової контрольної суми програм з початковими або оригінальними їх варіантами. Якщо обидва хеш-значення збігаються, то файли вважаються немодифікованими та незараженими. Якщо ж вони не співпадають, антивірусна програма повинна використовувати додаткові функції для встановлення того, чи модифікація є "вірусоподібною" або чи вона спричинена діями користувача.

Рис. 1.2 Кроки перевірки цілісності.

Перевірка цілісності ефективна у боротьбі з шкідливим програмним забезпеченням тому, що віруси вносять зміни до своїх хост-програм. Іншими словами, жоден вірус не може заразити хост-програму, не змінивши її код. Дана ситуація створює різницю між хешем незараженого та зараженого файлу. Однак перевірка цілісності не спрацює, якщо комп'ютер буде містити віруси при першому скануванні та збереженні контрольної суми.

Оскільки перевірка цілісності не використовує сигнатуру або підписи вірусів, її можна використовувати для виявлення будь-якого типу вірусу, включаючи нові та невідомі віруси. Крім того, проходження перевірки цілісності набагато швидше, ніж проходження сканування сигнатури.

Як і у скануванні сигнатури, перевірки цілісності використовується також для всього диска або програми перед її виконанням. Метод перевірки цілісності застосовується не тільки сканерами вірусів, але й іншими системами безпеки. Деякі системи CMOS BIOS мають налаштування для контролю змін у завантажувальних записах. Системи Windows 2000 та XP сканують свої системні файли під час завантаження як частина механізму їх перевірки (SFC). Деякі операційні системи зберігають резервну копію своїх основних системних файлів у спеціальних місцях та відновлюють їх у випадку пошкодження або зараження.

Евристичне сканування – це ще один метод виявлення вірусів, який не пов’язаний із сигнатурою та цілісністю. За допомогою евристичного антивірусного сканування вивчається цільова програма (виконуваний файл, завантажувальний запис або файли-документи з макросом) та аналізується її програмний код для визначення подібності його до вірусу. Іншими словами, евристичний сканер виявляє команди в межах програми, які не зустрічаються в типових прикладних програмах, таких як механізм реплікації вірусу, розповсюдження хробака чи корисне навантаження троянця. Якщо код цільової програми видається вірусоподібним, то сканер повідомляє про можливе зараження.

Оскільки евристичний метод не використовує сигнатуру вірусів, він здатен виявити нові та невідомі віруси, які ще не були проаналізовані дослідниками та розробниками антивірусного програмного забезпечення.

В евристичній методиці не використовується інформація про цілісність та не вимагається збереження «хеш-значення» програм, коли комп'ютер ще знаходиться у незараженому стані.

Види евристичного сканування:

- За допомогою кодової аномалії евристичне сканування шукає певні коди чи команди в межах програми, які не зустрічаються в типових прикладних програмах, таких як механізм реплікації вірусу, розповсюдження хробака або корисне навантаження троянця.
- В аномалії протоколу, наприклад, додатки, які використовують протоколи TCP / IP, наслідують певні моделі. Якщо сканер виявить будь-яке відхилення від стандартів, то він запідозрить вторгнення через мережу.
- На даний час використовується змішаний евристичний метод, при якому антивіруси дотримуються комбінації різних евристичних підходів разом із скануванням загальної сигнатури для автентифікації виявлення та уникнення помилкових позитивних результатів.

Евристичний метод поділяється на статичне та динамічне сканування. Основна відмінність цих двох підходів полягає в тому, що динамічний метод використовує емуляцію процесора, а при статичному методі такого не відбувається. При статичному евристичному методі антивірус досліджує послідовності команд цільової програми на операції, які зазвичай використовуються вірусами. На відміну від сигнатурного підходу дані послідовності не є специфічними для одного вірусу. Натомість вони мають бути максимально загальними для виявлення активності багатьох різних вірусів. Наприклад, якщо статична евристична антивірусна програма знайде операцію відкриття файлу, після якої проводяться операції читання та запису файлу, а також

знайде рядок "VIRUS" у програмі, він може повідомити, що файл заражений невідомим вірусом. Статичний евристичний метод має високу швидкість і часто виконується перед емуляцією процесора.

З іншого боку, динамічне евристичний метод використовує технологію на основі емуляції, яка завантажує цільову програму в програмний емулятор процесора. Емулятор процесора виступає як модельований віртуальний комп'ютер. Програмі дозволяється вільно виконуватись на цьому віртуальному комп'ютері. Коли імітується цільова програма, її вірусоподібні операції визначаються та каталогізуються. З цього каталогу вірусних операцій антивірусна програма визначає, чи цільова програма схожа на вірус.

Оскільки звичайне сигнатурне сканування не є ефективним проти поліморфних та метаморфних вірусів, багато антивірусних програм використовують емуляційну технологію для виявлення вірусів. Цей метод також є ефективним проти зашифрованих вірусів.

Метод моніторингу поведінки намагається виявити певну активність вірусу, наприклад, спроби переформатувати диск, що, як правило, не входить до діяльності загальних програми. В іншому випадку програма може спробувати перемістити файл в одну з папок операційної системи. Такі дії негайно досліджуються методом моніторингу поведінки.

Однак моніторинг поведінки не відрізняється від вищезгаданих методів. Моніторинг поведінки – це широке поняття, яке включає такі методи, як евристичне сканування, емуляція процесора та інші загальні методи сканування.

Методи сканування вірусів можуть бути класифіковані як специфічні методи (для виявлення конкретних вірусів або конкретного типу вірусів) та загальні методи (для виявлення вірусу будь-якого типу). Сканування сигнатури є специфічним методом, тоді як перевірка цілісності, евристичне сканування, моніторинг поведінки відносимо до загальних методів.

До загальних методів відносимо верифікацію:

- розміру програми
- контрольної суми
- зміни інформації
- адреси диска
- об'єктної програми
- об'єктної програми за допомогою стиснення та декомпресії

Сканування може бути за запитом або під час доступу. Сканування за запитом проводиться в режимі офлайн. Користувач повинен натиснути на кнопку, щоб розпочати операцію сканування або запланувати дану операцію пізніше. Хоча сканування за запитом або за вимогою здатне виявляти віруси, воно не заважає вірусам заражати інші файли.

З іншого боку, другий тип сканування запускається в момент доступу до файла або виконання програми. Сканування під час доступу здійснюється сканером-резидентом автоматично, коли отримується доступ до файла для копіювання, редагування чи інших подібних цілей. Сканер-резидент працює як модуль-резидент пам'яті і запускає перевірку для сканування файлу на льоту перед тим, як доступ до нього отриманий. Цей метод забезпечує цінний захист, оскільки він вловлює вірусні інфекції в режимі реального часу і не дозволяє вірусу заражати інші файли всередині системи.

1.3 Обмеження методів вірусного сканування

Кожен із зазначених методів має свої сильні та слабкі сторони. Сканування за допомогою сигнатури – це простий та надійний метод виявлення відомих вірусів, але його не використовують для новостворених та невідомих вірусів. Загальні методи можуть виявляти нові віруси, але не можуть вилікувати заражені файли. Нижче наведено деякі важливі недоліки, з якими стикаються традиційні методи сканування.

Один із основних недоліків сканування сигнатури полягає в часовому проміжку між створенням вірусу та його виявленням. Оскільки метод вимагає деякого часу для визначення сигнатури, розповсюдженню сигнатур серед клієнтів у всьому світі та використання нової бази для сканування, то новий вірус може легко поширюватися, заподіяти шкоду та не бути виявленим протягом визначеного періоду.

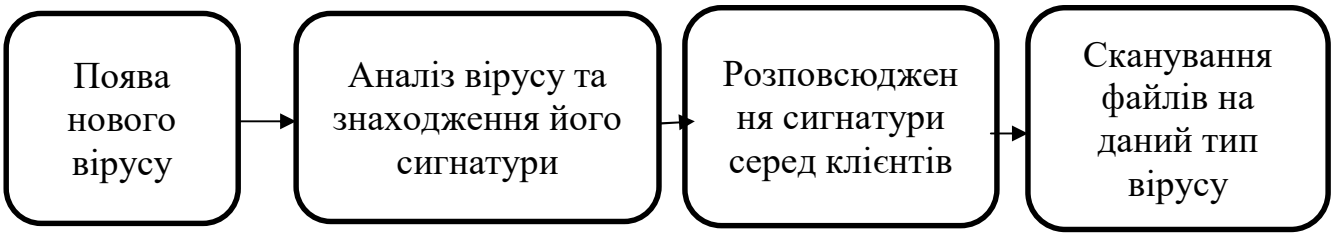


Рис. 1.3 Часовий проміжок між створенням вірусу та його сигнатурним скануванням.

Протягом проміжку часу, що охоплює створення вірусу та його виявленням, нове шкідливе програмне забезпечення, яке не включене до бази даних вірусів, не виявляється зазначеним методом. Значний часовий розрив створює ще більше можливостей для створення та поширення нових вірусів.

Подолання обмежень:

- Антивірусні компанії можуть застосовувати різні методи (наприклад, використання автоматизованих методів вилучення сигнатур, подання клієнтами вірусних зразків тощо) для того, щоб зменшити цей часовий проміжок між створенням вірусу і розповсюдженням його сигнатури.
- Вищезгаданий недолік сканування сигнатури може бути подоланий за допомогою загальних методів виявлення вірусів, таких як евристичне сканування, емуляція процесора та перевірка цілісності.

Сканування сигнатури не має ефективності без бази даних вірусів. Двигун сканування повинен посилатися на базу даних сигнатур для перевірки файлів. На жаль, оновлення бази даних сигнатур не здійснюється за допомогою разового

заходу, її потрібно постійно оновлювати. Оновлення бази даних сигнатур відіграє важливу функцію, оскільки під час сканування можна виявити лише ті віруси, сигнатури яких уже визначені та збережені в базі даних.

База сигнатур повинна бути оновлена як антивірусною компанією, так і її користувачем. Антивірусна компанія виконує дану функцію для комерційних цілей, але кінцевий користувач здійснює дану процедуру лише для того, щоб бути в безпеці від вірусів. Якщо база даних сигнатур не оновлюється, то повне сканування навіть не гарантує, що на комп'ютері немає вірусів.

Оскільки нові віруси створюються майже щодня, антивірусні компанії постійно відкривають нові віруси та регулярно оновлюють свої дані. З часом розмір цих файлових даних стає дуже великим і вимагає зайвої кількості часу для завантаження. Ця ситуація особливо погіршується при повільних та ненадійних Інтернет-з'єднаннях.

Подолання обмежень:

- Автоматичне оновлення – це звичайна методика, яка взята за правило на машині клієнта для завантаження нових оновлень із веб-сайту розробників антивірусу.

До обмежень сканування сигнатур відносимо:

- Метод сканування сигнатур дозволяє виявити лише відомі віруси, сигнатури яких були вже включені до бази даних щодо вірусів. Даний підхід не спроможний виявити жоден невідомий вірус.
- Крім того, існують певні типи вірусів, наприклад, поліморфні та метаморфні віруси, які змінюють свої команди у вірусному організмі при кожній інфекції. Оскільки вони не мають статичних рядків, їх часто неможливо виявити за допомогою сканування сигнатур.
- При загальному скануванні використовуються wild cards в сигнатурі для сканування декількох вірусів з однієї родини. Але це може не завжди спрацювати,

оскільки нові віруси тієї ж родини можуть не потрапляти в межі wildcards. У такому випадку новий вірус може бути не виявленим.

- Виокремлення сигнатури вірусів – це дуже кваліфікована робота.

Дослідники в сфері створення антивірусних програм повинні мати знання з різних мов та середовищ програмування. Віруси завантажувального сектора створені на мові асемблера, а віруси Windows написані на c / c ++ або Delphi, VB або VBA. Деякі також написані на Java та JavaScript. Дослідники повинні розуміти мови низького та високого рівнів програмування, включаючи внутрішні функції операційних систем.

Подолання обмежень:

- Завдання аналізу символів вірусу та вилучення його сигнатури можна прискорити за допомогою автоматизованих методів (Принцип TRIZ-25: Самообслуговування). Дослідникам слід аналізувати код лише в тому випадку, якщо автоматизовані системи не в змозі впоратися з потенційним вірусом.
- Сканування загальних сигнатур дозволяє виявити невідомі віруси. Не всі віруси створені спочатку або з нуля. Багато авторів змінюють коди вірусів і створюють десятки їм подібних. У таких випадках антивірусні дослідники намагаються знайти загальні сигнатури, які можуть бути присутніми у вірусній родині, замість того, щоб знайти індивідуальні сигнатури для кожного специфічного виду.
- Поліморфні, метаморфні та інші подібні віруси, які неможливо виявити за допомогою сканування сигнатур, можна виявити за допомогою інших методів, таких як моніторинг поведінки та перевірка цілісності (Принцип TRIZ-6: Універсальність).

Наступні обмеження пов’язані з нестачею часу сканування та ресурсів. Збільшення кількості вірусів призводить і до збільшення кількості їх сигнатур. Така ситуація збільшує розмір бази даних сигнатур і збільшує час сканування. З тисячами можливих типів вірусів і сотнями гігабайт файлосховищ операція

сканування часто перевищує прийнятні межі. Якщо сканер містить меншу кількість алгоритмів, то існує можливість пропущення деяких вірусів. З іншої сторони, якщо сканер включає всі можливі методи, то процес сканування стане надзвичайно повільним. Основне наше завдання полягає в тому, щоб включити всі можливі методи виявлення вірусів та щоб час сканування не тривав занадто довго.

Способи вирішення ситуації:

- Одним із методів підвищення швидкості сканування є перевірка лише тих вірусів відповідно до обраного типу файлів. Наприклад, сигнатура завантажувального сектору не буде використовуватися для сканування макрофайлу. Цей метод зменшує загальну кількість пошукових операцій та скорочує час сканування.
- Ще одна методика полягає в тому, щоб сканувати лише певну область файлу, а не сканувати весь файл. Залежно від типу файлу, наприклад, .com чи .doc, сканери можуть вирішити, які області файлу більше схожі на вміст вірусу. Цей метод дозволяє уникнути сканування всього файлу.
- Патент 5684875 (винайдений дослідником Гансом Елленбергером, листопад 1997 р.) Пропонує не запускати всі алгоритми виявлення, які можуть зайняти тривалий час для сканування. Він пропонує вибрати лише один або кілька алгоритмів, які потрібно запускати кожен раз при завантаженні сканера, які слід обирати випадковим чином або відповідно інших критеріїв відбору. Навіть якщо вірус не буде виявлений кількома такими алгоритмами пошуку, шанс на його виживання дуже низький, оскільки його можна знайти іншим алгоритмом.
- Патент 7036147 (винайдений дослідником Герсі, правонаступником McAfee, квітень 2006 р.) розкриває спосіб прискореного сканування шляхом призначення завдання сканування двом різним потокам. Винахід виконує два потоки паралельно, перший потік для зчитування даних, а другий потік для сканування даних. Перший потік операції виконується паралельно другому потоку таким чином, що в той час, як перша частина даних сканується, то друга частина даних,

що підлягають скануванню, зчитується і кешується. Цей метод дозволяє уникнути затримки, яка викликана операціями зчитування диска.

Наступні обмеження пов’язані з перевіркою цілісності:

- Порівняння контрольної суми або CRC має ефективність, якщо програма зберігається на жорсткому диску. Але якщо програма зчитується або виконується з дискети чи компакт-диска, даний метод втрачає свою ефективність. Оскільки незаражена контрольна сума не існує для файлів на дискетах, компакт-дисках або флешках, метод контрольної суми не може виявити заражені файли, що знаходяться на цих носіях.
- Перевірка цілісності не приносить користі, якщо комп'ютер уже заражений в той момент, коли вперше програма знімає контрольну суму. У таких ситуаціях комп'ютер буде заражений та створить безпечний притулок для вірусів для інфікування інших програм.
- Коли зміна програми виявляється за допомогою перевірки цілісності, антивірусу часто важко визначити, чи зміна була викликана вірусом чи викликана користувачем. Наприклад, користувач може оновити програму, встановивши нову версію або скопіювати оновлений файл – що може призвести до помилкового позитиву. Навіть якщо антивірусна програма може попросити користувача прийняти рішення з цього приводу, багато користувачів можуть не бути достатньо обізнаними для визначення, чи дані зміни є справжніми або чи вони спричинені вірусами.
- Перевірка цілісності підходить лише для файлів, які ніколи не змінюються протягом певного часу. Даний метод не використовується для файлів документів або даних, які користувач часто модифікує. Крім того, метод підходить лише для обмеженої кількості файлів (наприклад, системних і критичних файлів), оскільки для цього потрібно багато місця для резервного зберігання.

Наступні обмеження пов’язані із статичним евристичним скануванням.

Хоча програми статичного евристичного виявлення можуть бути досить швидкими, вони здатні розпізнати лише деякі з численних різних способів виконання операцій вірусами. Наприклад, віруси можуть відкривати файли, використовуючи різні коди та методи. У таких випадках статичний евристичний механізм виявлення повинен перевіряти велику кількість різних способів реалізації вірусу для того, щоб впевнено виявити його.

Оскільки для статичного евристичного методу потрібна база даних, що охоплює велику кількість можливих перестановок таких операцій, ситуація може з часом стати некерованою. Ця проблема буде особливо гострою, якщо автор шкідливого програмного забезпечення пише програму "генератор вірусів", яка створює тисячі вірусів одночасно, змінюючи порядок своїх розділів коду, але не зменшуючи його ефективної поведінки. З такою великою кількістю вірусів стане надзвичайно важко боротися з допомогою статичного евристичного сканування.

Також існують певні обмеження техніки емуляції. Хоча динамічна евристична методика є найбільш надійною у виявленні вірусоподібних операцій, існує багато проблем у її реалізації.

- Деякі випадки можуть вимагати великої емуляції перед здійсненням вірусоподібної операції. Наприклад, вірус може працювати в режимі холостого циклу або простою в 50 000 разів перед виконанням операції "відкриття файлу". У такому випадку перед досягненням операції з відкриття файлу потрібно буде наслідувати дуже велику кількість команд. Це значно уповільнить роботу антивірусної програми.
- По-друге, деякі віруси активуються лише за умови дотримання певних умов. Наприклад, вірус може здійснювати свою інфекційну активність в певну дату (наприклад, 13 числа місяця). Якщо ця умова не буде дотримана, динамічна евристична антивірусна програма не буде спостерігати інфекційну поведінку та не виявить вірус.

- По-третє, метод емуляції може не спрацювати добре для резидентних програм пам'яті. Динамічна евристична антивірусна програма починає емуляцію у головній точці введення цільової програми. Однак інфекційний вірусний код, що мешкає в пам'яті, не доступний через основну точку входу хост-програми. Натомість інфекційний вірусний код виконується лише тоді, коли відбувається переривання, до якого під’єднаний вірус, викликаний іншою програмою, відмінною від зараженої хост-програми.

Таким чином, навіть якщо динамічна евристична антивірусна програма дуже довго емулює заражену програму, то інфекційний вірусний код може бути не досягнутий, і, таким чином, підозрілі вірусні операції можуть не виявитись. Деякі проблеми спричиняють уразливість антивірусних програм. Як виявлення шкідливого програмного забезпечення стають складнішим, так і методи створення вірусів. Багато поліморфних та метаморфних вірусів використовують антивірусні методи і їх дуже важко виявити. Крім того, виробники вірусів можуть вразити антивірусну систему різними способами. Наприклад, вірус може залишатися в пам’яті та обманювати антивірусну програму, паралізувати її основні функції або використовувати лазівки антивірусної програми, щоб завдати шкоди комп’ютеру.

Деякі віруси створюють сильно стиснуті версії значно більших файлів, знаючи, що антивірусна система потребує величезного часу та ресурсів для декомпресії даного файлу для його перевірки на наявність вірусів у ньому. Ця ситуація може використовувати надмірну фізичну / віртуальну пам'ять і уповільнити роботу системи, якщо розмір декомпресованого файлу надзвичайно великий.

Для подолання даного обмеження слід вдатися до наступних кроків. Для уникнення вищезазначеної проблеми деякі антивірусні системи дотримуються механізму тайм-ауту для того, щоб вийти або припинити тривалу операцію сканування вірусів.

До інших проблеми сканування відносимо:

- Проблема в скануванні мережевих вірусів – звичайні сканери призначені для сканування вірусів на локальній машині. Вони не підходять для мережевих вірусів, які застосовують різні спеціальні методи для подорожі з однієї машини на іншу.
- Проблема сканування віддалених файлів – типовий сканер вірусів знаходиться на локальній машині та сканує локальні файли. Але вони не можуть сканувати віруси, що знаходяться на іншій машині. Наприклад, коли ви отримуєте доступ до файлу на віддаленому сервері, сканер не може дослідити файл, що знаходиться на віддаленому сервері.
- Сканування комп'ютера неможливе, якщо комп’ютерна машина надзвичайно сильно інфікована і не може завантажуватися з жорсткого диска.
- Проблема помилкових позитивів. Жоден метод сканування не є повністю надійним та гарантованим доказом виявлення вірусу. Кожен метод сканування може призвести до помилкових позитивних результатів, а це означає, що неінфікований файл може бути помилково ідентифікований як такий, що містить вірус. Помилкові позитиви настільки ж руйнівні, як і хибні негативи.
- Проблема несумісності з прикладним програмним забезпеченням. Хоча розробники антивірусів ретельно перевіряють свою продукцію на різних версіях операційних систем та іншого популярного прикладного програмного забезпечення, є численні інші програми та драйвери пристроїв, які не перевіряються в їх лабораторії. Отже, існує безліч випадків різних антивірусних програм, які впроваджують карантин або видаляють драйвера пристроїв і тим самим роблять пристрій дисфункціональним.

Отже, основними функціями антивірусної програми є запобігання поширенню вірусів та захист файлів, сканування та виявлення вірусів, видалення вірусу із заражених файлів та відновлення пошкоджених файлів та об’єктів.

Антивірусна програма зазвичай використовує різні стратегії виявлення та видалення вірусів. Популярні методи виявлення вірусу – це сканування сигнатур,

евристичне дослідження та перевірка цілісності. Однак кожен із цих методів містить свої сильні та слабкі сторони.

Сканування сигнатур є найпоширенішим методом виявлення вірусів. Але він не може виявити віруси, сигнатури яких відсутні у базі даних. Іншим популярним методом є використання евристичного алгоритму для пошуку вірусів на основі загальної поведінки. Даний метод може виявитись складним, але він має здатність знаходити невідомі або нові віруси. Перевірка цілісності – це ще один загальний метод, але він застосовується лише для певних типів файлів, які, як правило, не змінюються операціями користувача.

2 СИГНАТУРНИЙ АНТИВІРУСНИЙ ЗАХИСТ

2.1 Основні підходи до виявлення вірусів

Лише сама кількість та різноманітність шкідливих програм виступає однією з причин, що ускладнює проблему їх виявлення. Науковці Крістодореску (Christodorescu) та Джа (Jha) [10], МакГрау (McGraw) та Моррісет (Morisett) [35] дають детальний опис різних типів шкідливих програм.

Під терміном шкідливе програмне забезпечення розуміють програмні засоби, що несанкціоновано впроваджуються у комп'ютерну систему, здатні викликати порушення політики безпеки та завдавати шкоди інформаційним ресурсам, а в окремих випадках – і апаратним ресурсам комп'ютерної системи [7].

Для опису шкідливого програмного забезпечення запропоновано численні визначення. Наприклад, дослідники Крістодореску та Джа [11] описують шкідливе програмне забезпечення як програму, мета якої є зловмисною. МакГрау та Моррісет [35] визначають шкідливий код як "будь-який код доданий, змінений, або вилучений з програмної системи з метою навмисного заподіяння шкоди або підриву визначеної функції системи». Дослідники Васудеван (Vasudevan) та Єррабаллі (Yerraballi) [49] дали визначення шкідливому програмному забезпеченню як "загальне поняття, яке включає в себе віруси, трояни, шпигунські програми та інший інтрузивний код».

Дослідники Макграу та Моррісет зазначають, що категоризація шкідливого коду стає все складнішою проблемою, оскільки нові версії є комбінаціями тих, що належать до існуючих категорій.

Канонічні приклади шкідливих програм включають в себе хробаки, троянські коні та віруси. Таке визначення передбачає дещо детальнішу дискусію стосовно шкідливого програмного забезпечення.

Комп'ютерні хробаки реплікують себе, виконуючи власний код, який незалежний від інших програм. Основна відмінність вірусу від хробака полягає в

тому, що останній не потребує хазяїна для заподіяння шкоди. Ще одна відмінність між вірусами та хробаками полягає в моделі їх поширення. Взагалі віруси намагаються поширюватися через програми / файли на єдиній комп'ютерній системі. Однак хробаки поширюються через мережеві зв’язки із метою зараження якомога більше комп'ютерних систем, підключених до неї.

Троянський кінь – це зловмисне програмне забезпечення, яке вбудоване зловмисником у додаток або систему. На перший погляд видається, що програма чи система виконує якусь корисну функцію (наприклад, поширює інформацію стосовно місцевої погоди), але здійснює деякі несанкціоновані дії (наприклад, захоплення інформації про натисненні клавіші та надсилання її до шкідливого хосту). Троянські коні, як правило, пов'язані з доступом та надсиланням несанкціонованої інформації від свого хоста. Такі троянські коні також можна віднести до категорії шпигунських програм. Шкідливе програмне забезпечення не обмежене такими видами шкідливої діяльності. Вбудована зловмисна програма може також виконувати функцію сповільненої часової бомби [28]. Наприклад, зловмисне програмне забезпечення може створити такий сценарій; «4 травня 2010 року скомпроментована система буде запрограмована на відхилення всіх запитів для послуг».

Комп'ютерний вірус – це код, який реплікується, вставляючи себе в інші програми. Програма, в яку вставився вірус, вважається інфікованою та є його місцем поширення. Віруси для свого функціонування потребують своїх господарів чи місця поширення, тобто йому потрібна наявна хост-програма, щоб завдати шкоди. Наприклад, для того, щоб потрапити в комп'ютерну систему, вірус може приєднатися до якоїсь програмної утиліти (наприклад, додатку для обробки тексту). Запуск програми для обробки тексту може активувати вірус, який почне дублювати себе та виводити з ладу детектори виявлення шкідливих програм в комп'ютерній системі.

По суті, у життєвому циклі програмного забезпечення є дві фази, під час яких вставляється шкідливе програмне забезпечення. Ці фази називаються передрелізною та післярелізною. Внутрішня загроза або інсайдер, як правило, єдиний тип хакера, який спроможний вставити зловмисний код в програмне забезпечення перед його випуском, коли він досягне кінцевого користувача. Інсайдер – це надійний розробник програмного забезпечення, який виконує певні функції в організації, перед тим, як продукт потрапить до своїх кінцевих споживачів. Усі інші спеціалісти чи організації, які беруть на себе роль хакера, вставляють зловмисний код лише після випуску продукту, коли програмне забезпечення доступне для своєї цільової аудиторії.

Створюючи нову шкідливу програму, автори використовують одну або обидві з наведених нижче методик: обфускація (заплутування коду) та модифікацію поведінки [11] для того, щоб обійти детектори з виявлення шкідливих програм. Обфускація намагається приховати справжні наміри шкідливого коду, не розширюючи поведінку, яку проявляє шкідливе програмне забезпечення. Модифікація поведінки ефективно створює нову шкідливу програму, хоча суть зловмисного програмного забезпечення може не змінитися. Широке використання вищезазначених методів розробниками зловмисних програм поряд із тими, про які згадували дослідники [12, 19], говорить про те, що повторно використаний код є головним компонентом у розробці нових шкідливих програм.

2.2 Сканування на основі сигнатури та аномалії

Детектор зловмисного програмного забезпечення – це реалізація певної методики виявлення шкідливих програм для захисту системи на основі визначення зловмисної поведінки та слугує емпіричним засобом оцінки можливостей знаходження небезпечних програм.

Детектори шкідливих програм містять два приймаючі входи. Один із входів використовується для отримання відомостей про шкідливу поведінку, що є оберненими знаннями, які отримані під час фази навчання. Таким чином, під час виявлення на основі аномалії стає відомим, що таке аномальна поведінка, виходячи з її знання про те, що є звична або нормальна. Оскільки аномальна поведінка містить зловмисну поведінку, деякі прояви зловмисної поведінки перехоплюються програмним забезпеченням на основі аномалії. При використанні методу на основі сигнатури знання про те, що є шкідливим, походить із сховища, яке зазвичай оновлюється у ручному режимі спеціалістами, які змогли виявити небезпечну поведінку та висловити це у формі, придатній для сховища сигнатур, і в кінцевому рахунку для прочитання його комп'ютерною машиною.

Інший вхід використовується для програми, яка перевіряється. Як тільки детектор з виявлення зловмисного програмного забезпечення отримує повну інформацію про те, що вважається шкідливою поведінкою (нормальною поведінкою), і програму, яка перевіряється, тоді він може використовувати свою технологію виявлення для визначення стосовно шкідливості чи доброякісності програми. Часто системи виявлення вторгнень (Intrusion Detection Systems)(IDS) та детектори з виявлення зловмисних програм (malware detector) використовуються синонімічно, при цьому детектор з виявлення зловмисного програмного забезпечення є лише компонентом повної IDS.

Методи, які використовуються для виявлення зловмисного програмного забезпечення, можна класифікувати в дві категорії: виявлення на основі аномалії та виявлення на основі сигнатури. Метод виявлення на основі аномалії використовує свої знання про те, що є нормальною поведінкою, для того, щоб вирішити шкідливість програми, яка перевіряється. Детектори на основі сигнатури використовує характеристики того, що вже є відомим як шкідливе, у вирішенні небезпечності програми, яка перевіряється. Отже, така характеристика зловмисної поведінки є запорукою ефективності методу виявлення на основі сигнатури.

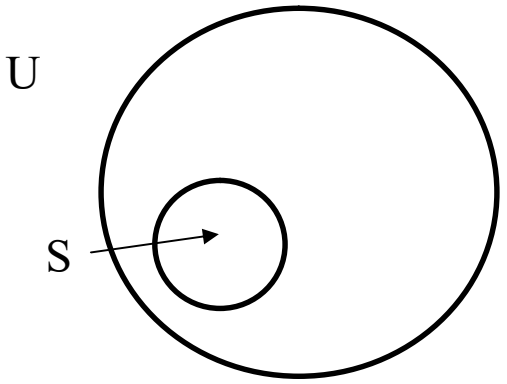
На малюнку 1 зображено взаємозв'язок між різними видами методів виявлення зловмисних програм. Кожна з методів виявлення використовує один з трьох різних підходів: статичний, динамічний або гібридний (див. Малюнок 1). Конкретний підхід або аналіз методики на основі аномалії чи сигнатури визначається тим, як методика збирає інформацію для виявлення шкідливих програм.

Рис. 2.1. Виявлення шкідливого програмного забезпечення на основі сигнатурного підходу.

Статичний аналіз використовує синтаксичні або структурні властивості програми (статичний) / процес (динамічний) для визначення її шкідливості. Наприклад, статичний підхід до виявлення на основі сигнатури використовує лише структурну інформацію (наприклад, послідовність байтів) для визначення зловмисності, тоді як динамічний підхід використовує інформацію під час виконання програми (наприклад, системи, що бачать на стеку виконання). Взагалі статичний підхід намагається виявити зловмисне програмне забезпечення перед

виконанням програми, яка перевіряється. І навпаки, динамічний підхід намагається виявити шкідливу поведінку під час виконання програми або після її виконання. Існують гібридні методи, що поєднують два підходи [38]. У цьому випадку для виявлення шкідливих програм використовується статична та динамічна інформація.

У сигнатурному захисті для виявлення шкідливих програм використовується модель зловмисної поведінки, яку часто називають підписом. В ідеалі за допомогою сигнатури можна ідентифікувати будь-яке шкідливе програмне забезпечення, яке виявляє зловмисну поведінку. Як і будь-які дані, які існують у великій кількості, що вимагає зберігання, для сигнатури необхідно мати сховище. До репозитарія або сховища звертаються для уточнення інформації стосовно шкідливості програми на основі сигнатури. В даний час досвід експертів відіграє ключову роль у створенні моделей або сигнатур. Одним з головних недоліків сигнатурного методу полягає в тому, що він не здатен виявити атаку нульовою дня, тобто стосовно неї не міститься відомостей у репозитарію.



U = Всі відомі вияви шкідливої поведінки

S = Всі відомі зразки сигнатур

Рис.2.2: Обмеження сигнатурного методу.

Рис.2.2 ілюструє основний недолік методів, які засновані на сигнатурі. Оскільки сукупність можливих шкідливих поведінок, U, нескінченно велика, то невідомо як презентувати U за допомогою сигнатури. Крім того, сховище

сигнатур є слабким наближенням до U. Ще одним недоліком методу полягає в тому, що для розробки сигнатур зазвичай потрібна участь експертів. Це не тільки може призвести до людської помилки, але й забирає значно більше часу, ніж якби розробка сигнатур проводилась автоматизовано. Зважаючи на те, що деякі зловмисні програми поширюються надзвичайно швидко, можливість швидко розробити сигнатуру набуває першорядного значення. Автоматизовані конструктори сигнатур існують [26], але вони ще недосконалі.

Деякі сигнатурні методи використовують той факт, що велика кількість шкідливих програм є похідними від попередніх версій. Таким чином, для того, щоб скористатися даним спостереженням, необхідно створити сигнатуру зловмисних програм таким чином, щоб фіксувати шкідливу сутність або інваріантну частину. Головна мета побудови зазначеної парадигми сигнатур полягає в тому, щоб зробити детектори зловмисного програмного забезпечення менш сприйнятливими до обфузації. Ще однією важливою причиною побудови сигнатур в даний спосіб є мінімізація кількості сигнатур шкідливих програм, які зберігаються у сховищі або репозитарії. Наразі зберігання сигнатур не є проблемою, але в майбутньому може викликати труднощі, які пов’язані з часовою складовою програмного забезпечення для виявлення шкідливого вмісту.

2.3 Сигнатурне виявлення шкідливого програмного забезпечення на основі динамічного підходу (Dynamic Signature-based Detection)

Виявлення шкідливих програм становить галузь, яка найбільше хвилює не лише дослідницьку спільноту, але й ширшу громадськість. Методи, які розробляються дослідниками та науковцями для виявлення зловмисних програм, реалізуються за допомогою впровадження детекторів виявлення шкідливих програм. Динамічне сигнатурне виявлення характеризується використанням виключно інформації, зібраної під час виконання програми, для вирішення

стосовно її зловмисності та пошуками моделі поведінки, які виявляють шкідливі її наміри.

До першого динамічного сигнатурного методу відносимо систему виявлення вторгнень на основі нормування (Rule based IDS Approach). У роботі Ільгуна (Ilgun) та ін. [22] атаки моделюються як діаграми переходу стану. Припускається, що існує деякий механізм аудиту даних. Цей механізм передає дані препроцесору, який змінює їх таким чином, щоб їх можна було проаналізувати за допомогою діаграми переходу стану. Потім дані порівнюються з відомими проникненнями, які існують у вигляді діаграм переходу стану.

Дослідник Елліс та ін. [15] пропонують поведінковий підхід до виявлення хробаків (Behavioral Approach to Worm Detection), заснований на відомій шкідливій поведінці. Автори представляють чотири різних поведінкових сигнатури. До базових сигнатур відносимо ті, які можна ідентифікувати за допомогою моніторингу потоків даних, що надходять та виходять з одного вузла. Базова сигнатура виявляється тоді, коли сервер змінюється на клієнта. Для свого поширення хробак після компрометування сервера повинен діяти як клієнт для наступного хосту, що призводить до зараження більшої кількості машин. Такий підхід втрачає свою ефективність, якщо застосований в одноранговому середовищі.

Відповідно до наступної базової сигнатури альфа на вхід та альфа на вихід хробаки зазвичай надсилають подібні дані по вузлах, і тому часто мають схожі, якщо не однакові посилання на вхід та вихід даних. Даний підхід має певні обмеження в деяких сервісах, оскільки немає нічого надзвичайного в пересилці подібних даних. Наприклад, немає нічого незвичайного у файлових серверах, які отримують та надсилають подібні дані.

Інша форма поведінкової сигнатури називається розгалуження. Даний підхід просто встановлює поріг кількості нащадків, яку може мати хост у будь-який момент часу. Відношення нащадків – приклад спонукальної сигнатури, який

передбачає, що існує деяка низка заражених хостів, які відповідають за інфікування інших хостів, викликаючи експоненціальний ріст заражень та є причиною швидкого поширення хробака. Для створення сигнатури відповідно до даного типу поведінки необхідно врахувати наступне: (1) глибина дерева, (2) кількість нащадків у дереві, (3) середній коефіцієнт розгалуження та (4) час, який потрібен для досягнення певної глибини дерева.

Автори проаналізували сигнатуру сервер-клієнт та альфу на вхід / альфа на вихід. Сигнатура, яка визначає перехід від сервера до клієнта, успішно використовувалася до активних хробаків. Підпис альфа на вхід та на вихід залежить від порогового значення, вибраного для альфи. Наприклад, значення альфи 1 містить великий показник помилкової тривоги.

2.4 Сигнатурне виявлення шкідливого програмного забезпечення на основі статичного підходу (Static Signature-based Detection)

Статичне сигнатурне виявлення на основі сигнатури характеризується вивченням та перевіркою програми на предмет послідовностей коду, які виявляють зловмисні наміри програми.

Мета полягає в тому, щоб отримати доступ до коду, що є відображенням поведінки програми. Сигнатура зазвичай представлена послідовностями коду. Сигнатурний метод використовує свої відомості (наприклад, послідовності команд, які вважаються відповідальними за появу зловмисної поведінки) і порівнює програму з відомими сигнатурами. Основна перевага методу, заснованого на статичному підписі, полягає в тому, що програму можна проаналізувати і однозначно встановити її шкідливість без запускання.

Експерти в галузі інформаційних технологій Сун (Sung) та ін. [47] запропонували метод, який називається Статичний аналіз для шкідливих виконуваних файлів (Static Analysis for Vicious Executables)(SAVE). Форма

підпису для даного вірусу задається послідовністю викликів API Windows. Кожен виклик API представлений 32-бітним номером. Найбільш значущі 16 біт відповідають модулю, до якого належить виклик API, тоді як найменш значущі 16 біт відповідають позиції функцій API у векторі функцій API. Евклідова відстань обчислюється між відомими підписами та послідовністю викликів API, знайдених у програмі. Середнє значення трьох функцій подібності дає подібність послідовності API програми з послідовністю сигнатур у репозитарію. Якщо різниця становить 10 відсотків або менше, то програма позначається як шкідлива. Дослідник Сун та ін. порівняли SAVE з 8 детекторами шкідливих програм. Порівнювали детектор шкідливих програм SAVE: Norton, McAfee Unix Scanner, McAfee, Dr. Web, Panda, Kasperksy, F-Secure та Anti Ghostbusters. Сун та ін. протестували всі ці сканери з W32.Mydoom, W32.Bika, W32.Beagle та W32.Blaster.Worm. SAVE був єдиним детектором, який зміг виявити всі варіанти згаданого шкідливого програмного забезпечення в процесі дослідження.

Наступний сигнатурний метод виявлення шкідливого програмного забезпечення на основі статичного підходу називається семантика (Semantics-aware). У праці дослідника Christodorescu та ін. [11] підписи зловмисних програм представлені шаблонами. Кожен шаблон – це набір трьох взаємозв'язаних команд, змінних та символічних констант. Шаблони намагаються узагальнити сигнатуру екземпляра зловмисного програмного забезпечення, але при цьому зберігають суть поведінки зловмисного коду.

Три кроки необхідні для визначення шкідливості програми. По-перше, програма перетворюється на незалежне проміжне представлення платформи (IR), яке є варіантом мови x86. Далі, графік потоку управління обчислюється для проміжного подання програми і порівнюється з графіком потоку управління шаблону. Нарешті, порівняння проводиться за допомогою пар "опис-використання". Якщо для кожної пари "опис-використання", яка знайдена в шаблоні, є відповідна пара в IP-коді програми, тоді вона є шкідливою.

Результати науковців Крістодореску (Christodorescu) та ін. вказують на той факт, що підхід на основі шаблонів має можливість виявляти варіанти зловмисного програмного забезпечення з нульовими помилковими позитивами. Під час експерименту отримали 21 екземпляр хробаків електронної пошти з таких родин шкідливих програм: Netsky, B [e] agle та Sober. Вражаюча здатність виявлення була досягнута лише за допомогою двох шаблонів, а саме петля дешифрування та шаблону масової розсилки. Хоча не було виявлено масової розсилки хробака Sober, але реалізація проводилась в той час, коли не здійснювалася підтримка бібліотеки часу виконання Microsoft.

Дослідник Крістодореску та ін. також встановлено, що в 2000 доброякісних програмах Windows їх алгоритм не видав помилкових позитивних результатів. Автори також оцінили стійкість їх методів до обфускації.

Дослідники Кумар (Kumar) і Спаффорд (Spafford) [27] запропонували загальний сканер (Generic Virus Scanner), який виявляв віруси на основі регулярної відповідності експресії. Оскільки запропонований сканер був розроблений виключно для SunOS, реалізація авторів важко було порівняти з вірусними сканерами інших операційних та файлових систем.

Фахівці в сфері комп’ютерних технологій Крейбіч (Kreibich) та Кроукрофт (Crowcroft) [26] запропонували «Медові стільники» («Honeycomb»), що є системою, яка використовує принаду (honeypot) для генерування сигнатур та виявлення шкідливих програм, що виникають із мережевого трафіку. Методика авторів працює за припущенням, що трафік, який спрямовується на принаду, є підозрілим. Сигнатура, яка мало використовуються, видаляється з черги підписів. Крім того, якщо знаходиться така сама сигнатура, то вона не додається до пулу підписів. Це допомагає зберегти пул підписів в найменшій кількості.

2.5 Гібридний сигнатурний підхід до знаходження вірусів (Hybrid Signature-based Detection)

Гібридне сигнатурне виявлення використовує статичні та динамічні властивості для визначення шкідливості програми. Дослідник Морі та ін. [37] запропонували інструмент аналіз та виявлення шкідливого мобільного коду (Analyzing and Detecting Malicious Mobile Code), який виявляє самошифрувальні та поліморфні віруси. Традиційні методи не можуть виявити такі типи вірусів, оскільки вони обходять їх методики. Іншими словами, шифруючи себе, віруси не демонструють шаблонів від яких залежать відповідні методики їх виявлення. Тому автори створили інструмент для вирішення цього питання.

Використовуючи підхід науковців Морі та ін., код розшифровується в емуляторі операційної системи. Після того, як вірус розшифрував свою корисне навантаження, на ньому проводиться статичний аналіз для виявлення системних викликів, здійснених з корисного навантаження. В ході експерименту їх інструментом вдалося виявити 600 зразків вірусів / хробаків.

Дослідники Ло (Lo) та ін. [33] запропонували інструмент фільтр шкідливих кодів (MCF: Malicious Code Filter). Даний метод використовується для аналізу виконуваних файлів, що не вимагає від програміста надання офіційних специфікацій. Відповідно до методу використовується виконуваний файл. Потім створюється проміжний образ, який є CFG виконуваного файлу. За допомогою аналізу різних зловмисних програм автори розробили ознаки «контрольного сигналу» для шкідливих програм. Під час використання даної методики аналітик в кінцевому підсумку приймає рішення стосовно шкідливості програми. Ознаки контрольного сигналу слугують керівництвом для людини-аналітика. Дані ознаки контрольного сигналу описують різні класи шкідливих програм. Таким чином, експерт створює фільтр, який базується на ознаках контрольного сигналу.

Отже, незважаючи на велику кількість сигнатурних методів виявлення зловмисних програм, вони містять як слабкі, так і сильні сторони. Дати однозначну відповідь на питання, який підхід краще використовувати, надзвичайно важко, навіть якщо розробити універсальну метрику, оскільки деякі операційні та файлові системи більш вразливі від шкідливого програмного забезпечення. В даний час оцінка ефективності методів виявлення зловмисних програм виглядає досить умовно.

3 ЗАБЕЗПЕЧЕННЯ СИГНАТУРНОГО ЗАХИСТУ ЗА ДОПОМОГОЮ СКАНЕРА «CLAMAV»

3.1 Порівняння основних видів антивірусних програм зі сканером ClamAV

Антивірусних програма зі сканером ClamAV є вільним продуктом та розповсюджується за ліцензією GPL і доступний для всіх основних операційних систем.

Основне призначення ClamAV не полягає в захисті робочих станцій, а зосереджується на поштових шлюзах, що накладає певний відбиток на характеристики продукту. У той же час існують і десктопні версії антивіруса і навіть комерційні продукти на його основі.

Для визначення ефективності сканера ClamAV в середовищі Linux було протестовано можливості даного антивірусного продукту. Для тестування використовувалися той же набір шкідливого ПО, який використовували для перевірки MSE, що дає можливість додатково порівняти результати. В обох випадках використовувалася остання на сьогодні версія ClamAV - 0.102.2.

До основних переваг сканера відноситься висока швидкість сканування, яка є однією з найвищих серед відомих антивірусів. Враховуючи те, що основне призначення ClamAV полягає в тому, щоб сканувати поштовий трафік на шлюзах, стає зрозумілим, чому швидкість покладена в основу антивірусу, навіть за рахунок втрати ефективності, що підтверджується результатами.

Таблиця 3.2 – Порівняння Linux версії ClamAV з основними антивірусними програмами.

	Ефективність антивірусних програм під час виявлення та знешкодження шкідливого програмного забезпечення

Основні антивірусні програми	ClamAV 0.102.2	AVAST Free 20.3.2405	AVG 20.3.3120	Bitfender Antivirus Free Edition	Kaspersky Total Security 2020
Швидкість сканування (5 балів максимум)	5	4	3	3	4
Ефективність у знаходженні шкідливих програм та їх видалення	64%	69%	71%	82%	93%
Безкоштовність	+	+	+	–	–

Незважаючи на те, що безкоштовний і відкритий ClamAV показав себе досить ефективним, але його надійності виявилось недостатньо для захисту робочих станцій.

Враховуючи той факт, що основне використання антивірусу ClamAV – це насамперед шлюзи і це перший рубіж антивірусного захисту інфраструктури, то результат досить задовільний, з урахуванням можливості сканування трафіку (і не тільки поштового) на льоту. Дійсно, 64% вірусів відсіяні на першому етапі, що в рази зменшує кількість вірусних інцидентів у мережі. В даному вимірі антивірусна програма ClamAV є досить конкурентною, як рішення для шлюзів, у порівнянні її з комерційними альтернативами. Висока швидкість і безкоштовність – це основні сильні сторони даного сканера для невеликих і середніх мереж.

На офіційному сайті ClamAV присутнє посилання на Immunet 3.0 - комерційний продукт на основі ClamAV і призначений для захисту робочих станцій під управлінням Windows. Даний продукт пропонує до використання три антивірусних сканера: ClamAV, TETRA і хмарний інструмент на базі ClamAV, що вимагає наявності інтернет з'єднання. Існує також безкоштовна версія, в якій доступні тільки ClamAV і хмарний сканер.

Хмарний сканер можна поєднувати з двома іншими, але спільне використання ClamAV і TETRA не рекомендується. Антивірусний механізм в хмарі досить незвичайний, але цілком прийнятний, оскільки все більша кількість вендорів пропонують дані послуги. Незважаючи на обов’язкове з’єднання з Інтернетом, такий підхід містить велику кількість позитивних моментів: завжди актуальна база даних, висока швидкість реагування на нові загрози тощо.

Отже, ClamAV в ході тестування показав досить неоднозначні результати. З одного боку висока швидкість при середньому рівні сканування робить його непоганим, з урахуванням безкоштовності, рішенням для шлюзів, з іншого – результат явно недостатній для захисту робочих станцій. Тому найкраще використаання ClamAV не для захисту ПК користувачів, а на шлюзах і Linux-серверах в якості першого антивірусного захисту.

3.2 Визначення функціональних можливостей антивіруса ClamAV

Clam AntiVirus — пакунок антивірусного ПЗ, що працює у багатьох операційних системах, включаючи Unix-подібні ОС, Microsoft Windows та Apple Mac OS X. Випускається під GNU General Public License і є вільним програмним забезпеченням. Головна мета Clam AntiVirus — інтеграція з серверами електронної пошти для перевірки файлів, прикріплених до повідомлень. У пакунок входить масштабований багатопотоковий демон clamd, керований з

командного рядка сканер clamscan, а також модуль оновлення сигнатур з Інтернету freshclam.

У жовтні 2013 компанія Cisco поглинула фірму Sourcefire [5][6], що займалася розробкою таких вільних проектів, як система виявлення атак Snort, антивірусний пакет ClamAV і платформа для виявлення загроз Razorback. Розробники ClamAV запевнили спільноту, що проекти як і раніше будуть розвиватися як відкриті, і всі налагоджені канали взаємодії з спільнотою будуть збережені. Більш того, розробка Snort, ClamAV та інших відкритих проектів буде продовжена об'єднаною командою інженерів з Sourcefire і Cisco. До основних функціональних можливостей антивіруса ClamAV відносимо наступні:

- Підтримка командного рядка
- Відповідність популярним поштових форматам (використовується на поштових шлюзах)
- можливість використання з більшістю поштових серверів, включаючи реалізацію milter-інтерфейсу для Sendmail
- Перевірка різних архівних файлів: Zip, RAR, Dmg, Tar, Gzip, Bzip2, OLE2, Cabinet, CHM, BinHex, SIS and others
- Перевірка виконуваних файлів ELF executables and Portable Executable files packed with UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack and obfuscated with SUE, Y0da Cryptor and others
- Перевірка популярних офісних форматів: MS Office and MacOffice files, HTML, Flash, RTF and PDF
- сканер у виді бібліотеки Ci;
- сканування файлів і пошти «на льоту»;
- визначення понад 546 000 вірусів, хропаків, троянів, повідомлень фішингу;
- підтримка сканування mbox, Maildir і «сирих» поштових файлів;

- аналіз файлів формату Portable Executable, запакованих UPX, FSG чи Petite.
- у рамках тестування додана можливість аналізу загроз з використанням відеокарт

Отже, ClamAV – це антивірусна програма, яка містить в собі інтерфейсу командного рядка. Даний сигнатурний аналізатор використовується для сканування файлів та папок на вміст вірусів, зловмисних програм чи інших кібернетичних загроз. Антивірусна програма ClamAV включає в себе як сканер, так і базу даних вірусів (CVD). Даний список потрібно періодично оновлювати для забезпечення захищеності від відомих загроз. Даний сигнатурний сканер не вміщує в себе поведінкові та евристичні алгоритми для пошуку загроз із вразливістю нульового дня. Незважаючи на це, незалежні випробувальні лабораторії досить сприятливо оцінили його можливості. Насправді, ClamAV часто випереджає комерційних виробників у лабораторних тестах. Базою даних керує Cisco Talos, який придбав інструмент початкового виробника Sourcefire. Незважаючи на те, що ClamAV використовується з відкритим кодом, значні технічні ресурси направлені на виявлення найвідоміших вірусів та загроз.

Оскільки ClamAV використовує інтерфейс командного рядка, тому він розроблений для запуска в середовищі терміналів, таких як Linux або командний рядок Windows (DOS), а тому не містить типового візуального інтерфейсу користувача, якого очікують більшість користувачів.

Цілком можна використовувати ClamAV без графічного інтерфейсу користувача (GUI), але це передбачає запам’ятовування списку необхідних команд для виконання сканувань, оновлення бази даних та безпечного видалення файлів із карантину.

Інструмент ClamAV – це антивірусна програма з відкритим кодом та базою даних, яка спочатку розроблена для операційних систем на базі Linux та підтримується Cisco Talo. Його можна використовувати як інструмент командного

рядка (CLI), який ідеально підходить для захисту серверів, але містить ряд незручностей для користувачів настільних ПК, які пов’язані із запам’ятовуванням довгих списків команд.

3.3 Створення прототипу антивірусної програми

Створення прототипу антивірусної програми необхідний для аналізу файлів та програм на вміст шкідливого програмного забезпечення. Вимогою до такого сигнатурного сканера слугує постійне та повне доповнення новими сигнатурними даними про шкідливі програмні засоби. До основних вимог також відноситься вартість програмного продукту, тобто він має бути безкоштовним. Сигнатурний аналіз використовується для пошуку шкідливих програм відповідно до їх сигнатури, тобто набору певних ознак, за якими можна з впевненістю визначити, що даний файл відноситься до вірусів.

Для вирішення даної проблеми існують різні методики. Один із підходів – це використання сигнатури, яка складається з N байт шкідливого об'єкта. Порівняння можна також проводити відповідно до маски для підвищення ефективності чи впровадження додаткових умов при пошуку. Проект «YARA» (code.google.com/p/yara-project) використовується для визначення відповідності між певною сигнатурою та файлом, який досліджується.

Самі розробники позиціонують його як інструмент для допомоги в дослідженні шкідливого програмного засобу та його класифікації. Для знаходження шкідливих програм достатньо використати текстові, бінарні та шістнадцяткові шаблони, в яких описуються їх формалізовані ознаки.

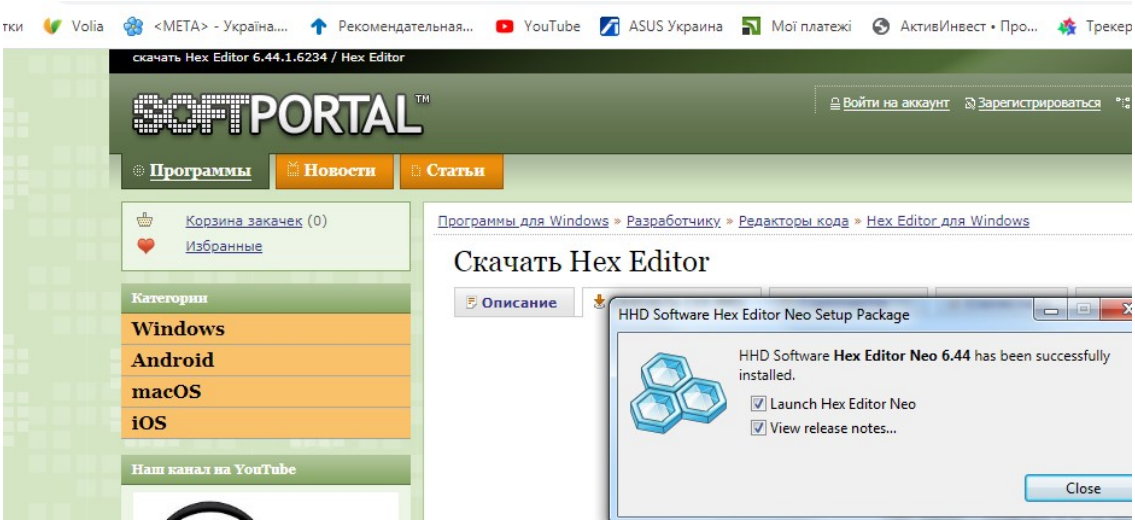


Рис.3.1 Використання шістнадцяткового редактора в проекті «YARA».

Кожна сигнатура складається з набору рядків і деякого логічного виразу, на основі якого визначається логіка сигнатурного аналізатора. Якщо для досліджуваного файлу виконуються умови одного з правил, він визначається відповідним чином (наприклад, певний тип хробака).

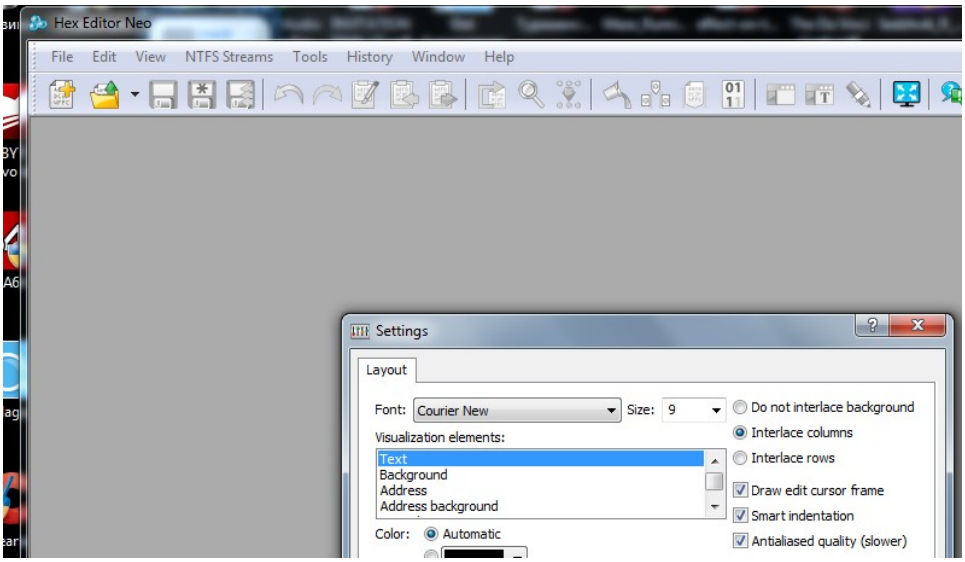


Рис.3.2 Вибір опцій в шістнадцятковому редакторі.

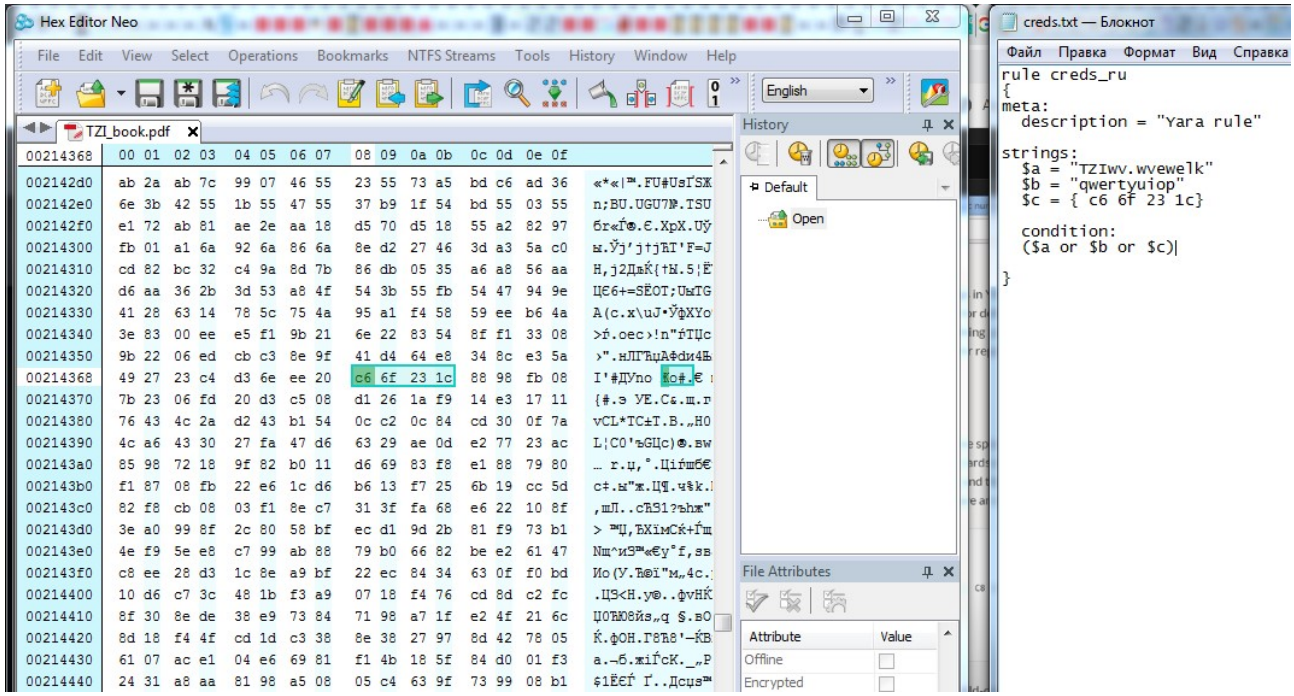


Рис.3.3 Вибір необхідної сигнатури в шістнадцятковому редакторі.

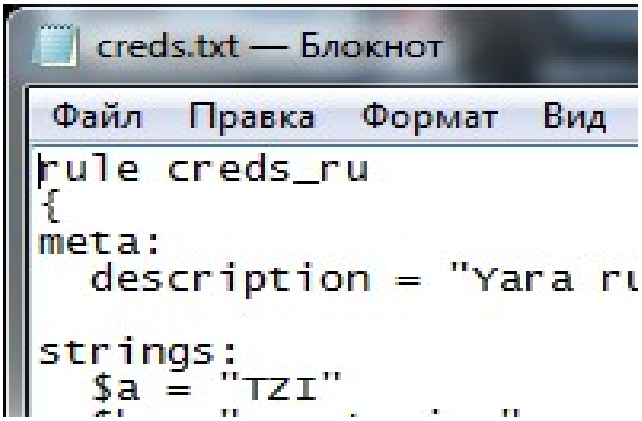


Рис.3.4 Використання сигнатури в проекті «YARA».

В даному правилі стверджується, що будь-який файл, який містить хоча б один рядок, який описаний в змінних $ a, $ b, $ c, повинен класифікуватися як шкідливе програмне забезпечення. Інструмент «YARA» вже використовує велику кількість проектів, серед яких хотілося б виділити «VirusTotal Malware Intelligence Services (vt-mis.com)», «jsunpack-n (jsunpack.jeek.org)» та «We Watch Your Website (wewatchyourwebsite.com)».

Оскільки проект «YARA» написаний на Python, тому його легко можна встановити на Linux, Windows та Mac.

В командному рядку синтаксис програми надзвичайно простий: Yara32 [OPTION] ... [RULEFILE] ... FILE | PID

Тобто формат виклику програми наступний: спочатку йде ім'я програми, потім список опцій, після чого вказується файл з правилами, а в самому кінці – ім'я досліджуваного файлу (або каталогу, що містить файли) чи ідентифікатор процесу. Для розробки безкоштовного антивірусу можна використати сигнатурну базу даних з популярного антивіруса ClamAV, яка знаходиться у вільному доступі (clamav.net/lang/en). У розділі «Latest Stable Release» розташоване посилання на останню версію антивірусного продукту, а також посилання для скачування вірусних баз ClamAV. Найбільш ефективні – це файли main.cvd (db.local.clamav.net/main.cvd) та daily.cvd (db.local.clamav.net/daily.cvd). Перший з файлів містить основну базу сигнатур, а другий – найповнішу на даний момент базу з різними доповненнями. Для поставленої мети цілком вистачить daily.cvd, в якому зібрано понад 100 000 зразків шкідливого програмного забезпечення. Для отримання бази ClamAV – її необхідно перетворити в потрібний формат за допомогою скрипту, який написаний Метью Річардом clamav_to_ yara.py (bit.ly/ij5HVs).

Опція -r вказує, що сканування необхідно проводити рекурсивно по всім підпапкам поточного каталога.

Отже, правило – це основний механізм програми, що дозволяє віднести заданий файл до будь-якої категорії. Правила описуються в окремому файлі (або файлах) і за своїм виглядом дуже нагадують конструкцію struct {} з мови C / C ++.

```
rule_Example
{
strings:
$ A = "star.exe"
```

$ B = " https://ru.wikipedia.org/wiki/Portable_Executable"

$ C = " https://github.com/VirusTotal/yara/releases/tag/v4.0.1"

condition:

($ A or $ b or $ c)

}

Отже, при створенні сигнатурного аналізатора було вироблено наступні основні принципи:

1. Кожна правило починається з ключового слова rule, після якого йде ідентифікатор правила. Ідентифікатори можуть мати такі ж імена, як і змінні в C / C ++, тобто складатися з букв і цифр, причому перший символ не може бути цифрою. Максимальна довжина імені ідентифікатора - 128 символів.

2. Зазвичай правила складаються з двох секцій: секція визначень (strings) і секція умови (condition). У секції strings задаються дані, на основі яких в секції condition буде прийматися рішення, чи задовольняє заданий файл певним умовам.

3.Кожний рядок в розділі strings має свій ідентифікатор, який починається зі знака $, як оголошення змінної в php. В інструменті «YARA» підтримуються звичайні рядки, укладені в подвійні лапки ("") та шістнадцяткові рядки, укладені у фігурні дужки {}, а також регулярні вирази:

$a = "text here"

$b = {E2 34 A1 C8 23 FB}

4.У секції condition міститься вся логіка правила. Ця секція повинна містити логічне вираз, що визначає, в якому випадку файл або процес задовольняє умові або правилу. Зазвичай в цій секції йде звернення до раніше оголошених рядків. А ідентифікатор рядка розглядається в якості логічної змінної, яка повертає true, якщо рядок був знайдений в файлі або пам'яті процесу, і false – в іншому випадку. Відповідно до вищезгаданого правила файли і процеси, що містять рядок star.exe та один з двох URL, повинні бути віднесені до категорії rule_Example (відповідно до назви правила).

5. Шістнадцяткові рядки дозволяють використовувати три конструкції, які роблять їх більш гнучкими: підстановки (wildcards), діапазонів (jumps) та альтернативного вибору (alternatives). Підстановки – це місця в рядку, які невідомі та можуть приймати будь-яке значення. Позначаються вони символом «?»:

$a = {A3 56 ?? B1 ?? CD}

Даний підхід дуже зручний при завданні рядків, довжина яких відома, а вміст може змінюватися. Якщо ж частина рядка може бути різної довжини, зручно використовувати діапазони:

$b = {A1 14 [3-7] 51 C5}

Даний запис означає, що в середині рядка може бути від 3 до 7 різних байт. Можна реалізувати також і альтернативний вибір:

$ Hex_string = {B3 32 (14 A3 | 41) 36}

Це означає, що на місці третього байта може бути 14 A3 або 41, такого запису відповідають рядки B3 32 14 A3 36 або B3 32 41 36.

6. Для перевірки того, чи заданий рядок знаходиться на певному місці в файлі або адресному просторі процесу, використовується оператор at:

$a at 20 and $b at 30

Якщо рядок може знаходитися всередині певного діапазону адрес, використовується оператор in:

$a in (0..20) and $b in (0..30)

Іноді виникають ситуації, коли необхідно вказати, що файл повинен містити певну кількість із заданого набору. Виконується це за допомогою оператора of:

```
rule_example2
{ strings:
  $a = "one"
  $b = "two"
  $c = "three"
  condition:
```

2 of ($a, $b, $c) }

Наведене правило вимагає, щоб файл містив будь-які два рядки з усіх ($a, $b, $c). Замість вказівки конкретного числа рядків у файлі можна використовувати змінні any (хоча б один рядок із заданої множини) і all (всі рядки із заданої множини).

7. Застосування однієї умови до багатьох рядків дуже схожа на оператор of, тільки більш потужна – for..of:

for expression of string_set: (boolean_expression)

Цей запис треба читати так: з рядків, заданих в string_ set, принаймні expression штук повинно задовольняти умові boolean_expression. Або, іншими словами: вираз boolean_expression обчислюється для кожного рядка з string_set, і expression з них повинні повернути значення True.

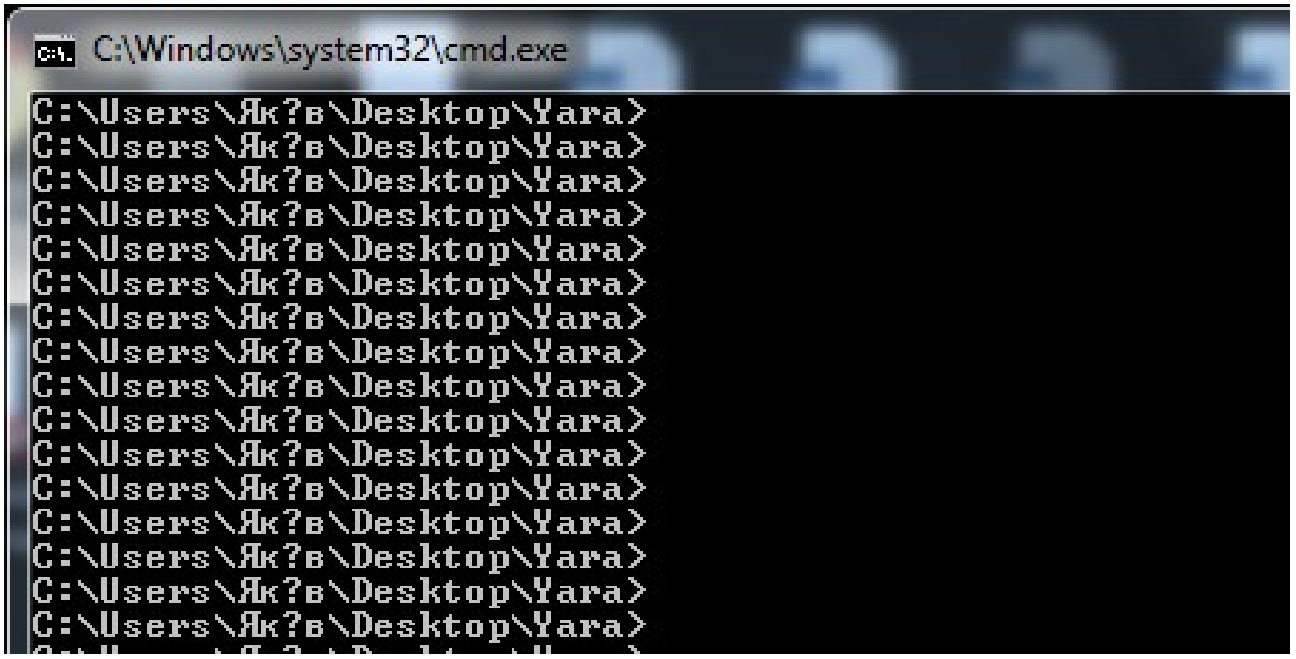


Рис.3.5 Використання текстової сигнатури для перевірки файлу.

Отже, використання прототипу антивірусної програми для аналізу файлів та програм на вміст шкідливого програмного забезпечення є надійним безкоштовним засобом. Сигнатурні зразки можна завантажувати з відповідних репозитаріїв, як

github, використовувати безкоштовні бази даних, як ClamAV чи створювати їх самому.

Для знаходження шкідливих програм використовуються текстові, бінарні та шістнадцяткові шаблони, в яких описуються їх формалізовані ознаки. Для опрацювання інформації, яка пов’язана із шістнадцятковими даними, можна використовувати спеціальні редактори. Синтаксис такого сканера простий, але він вимагає використання базових навиків роботи в командному рядку та позбавлений графічного інтерфейсу.

ВИСНОВКИ

В дипломній роботі розроблена методика сигнатурного антивірусного захисту з застосуванням сканерів. Реалізація проекту дає можливість порівняти основні антивірусні програми, визначити їх слабкі та сильні сторони, а також створити сканер за допомогою підключення власних та запозичених сигнатур (github) для виявлення шкідливих програмних засобів. Під час проведення дослідження використовувався шістнадцятковий редактор та інструмент «YARA» для перевірки файлів та програм. Застосування даного програмного засобу містить ряд переваг. Однією з них є безкоштовність та наявність великої кількості сигнатурних зразків.

В першому розділі було дане визначення ряду ключових понять, проведено аналіз методів вірусного сканування та їх обмеження.

Другий розділ був присвячений аналізу сигнатурного методу до виявлення шкідливого програмного засобу в системах виявлення вторгнень.

В третьому розділі було проведене дослідження ефективності основних антивірусних програм та розроблено прототип антивірусного сканера для виявлення шкідливих програм.

Джерельну базу дослідження становлять в основному зарубіжні публікації.

.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Adelstein F., Stillerman M., and Kozen D. Malicious code detection for open firmware. In Proceedings of the 18th Annual Computer Security Applications Conference, 2002.

2. Bergeron J., Debbabi M., Desharnais, Erhioui M.M., and Tawbi N. Static detection of malicious code in executable programs. Int. J. of Req. Eng., 2001.

3. Bergeron J., Debbabi M., Erhioui M.M., and Ktari B. Static analysis of binary code to isolate malicious behavior. In 8th Workshop on Enabling Technologies: Infrastructure for Collaborative Entrerprises, 1999.

4. Boldt M. and Carlsson B. Analysing privacy-invasive software using computer forensic methods. http: // www. e-evidence. info/ b. html , January 2006.

5. Castaneda F., Sezer E. C., and Xu J. Worm vs. worm: preliminary study of an active counter-attack mechanism. Proceedings of the 2004 ACM Workshop on Rapid Malcode, 2004.

6. CERT/CC, Carnegie Mellon University. http: // www. cert. org/ present/ cert-overview-trends/ module-2. pdf , May 2003.

7. CERT/CC, Carnegie Mellon University. http: // www. cert. org/ present/ cert-overview-trends/ module-4. pdf , May 2003.

8. CERT/CC, Carnegie Mellon University. http: // www. cert. org/ stats/ cert \ stats. html \#incidents , last updated: April 2006.

9. Christodorescu M. and Jha S. Static analysis of executables to detect malicious patterns. Usenix Security Symposium, 2003.

10. Christodorescu M. and Jha S. Testing malware detectors. In Proceedings of the International Symposium on Software Testing and Analysis, July 2004.

11. Christodorescu M., Jha S., Seshia S., Song D., and Bryant R. Semantics-aware malware detection. In Proceedings of the 2005 IEEE Symposium on Security and Privacy, pages 32–46, 2005.

12. Ciubotariu M. Netsky: a conflict starter? Virus Bulletin, May 2004.

13. Cowan C., Pu C., Maier D., Walpole J., Bakke P., Beattie S., Grier A., Wagle P., Zhang Q., and Hinton H. Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. In Proceedings of the 7th USENIX Security Conference, Jan. 1998.

14. Debbabi M., Giasson E., Ktari B., Michaud F., and Tawbi N. Secure self-certified cots. In Proceedings of the 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, pages 183–188, 2000.

15. Ellis D., Aiken J., Attwood K., and Tenaglia S. A behavioral approach to worm detection. In Proceedings of the 2004 ACM Workshop on Rapid Malcode, pages 43–53, 2004.

16. Filiol E. Malware pattern scanning schemes secure against black-box analysis. Journal of Computer Virol., 2006.

17. Forrest S., Perelson A. S., Allen L., and Cherukuri R. Self-nonself discrimination. In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy, May 1994.

18. Giffin J. T., Jha S., and Miller B. Detecting manipulated remote call streams. 11th USENIX Security Symposium, 2002.

19. Gordon J. Lessons from virus developers: The beagle worm history through april 24, 2004. SecurityFocus, May 2004.

20. Halfond W. and Orso A. Amnesia: Analysis and monitoring for neutralizing sqlinjection attacks. In Proceedings of the 20th IEEE/ACM Internation Conference on Automated Software Engineering, pages 174 – 183, 2005.

21. Hofmeyr S., Forrest S., and Somayaji A. Intrusion detection using sequences of system calls. Journal of Computer Security, pages 151 – 180, 1998.

22. Ilgun K., Kemmerer R. A., and Porras P. A. State transition analysis: A rule-based intrusion detection approach. IEEE Transactions on Software Engineering, 1995.

23. Kirda E., Kruegel C., Vigna G., and Jovanovic N. Noxes: A client-side solution for mitigating cross-site scripting attacks. In the 21st ACM Symposium on Applied Computing (SAC), 2006.

24. Ko C., Fink G., and Levitt K. Automated detection of vulnerabilities in privileged programs by execution monitoring. In Proceedings of the 10th Annual Computer Security Applications Conference, pages 134–144, December 1994.

25. Ko C., Ruschitzka M., and Levitt K. Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In Proceedings of the 1997 IEEE Symposium on Security and Privacy, 1997.

26. Kreibich C. and Crowcroft J. Honeycomb – creating intrustion detection signatures using honeypots. In 2nd Workshop on Hot Topics in Network, 2003.

27. Kumar S. and Spafford E. H. A generic virus scanner in c++. In Proceedings of the 8th Computer Security Applications Conference, pages 210 – 219, 1992.

28. Landwehr C., Bull A., McDermott J., and Choi W. A taxonomy of computer program security flaws. ACM Computing Surveys (CSUR), 26(3):211–254, 1994.

29. Lee R. B., Karig D. K., McGregor P., and Shi Z. Enlisting hardware architecture to thwart malicious code injection. International Conference on Security in Pervasive Computing (SPC), 2003.

30. Lee W. and Stolfo S. Data mining approaches for intrusion detection. In Proceedings of the 7th USENIX Security Symposium, 1998.

31. Li W., Wang K., Stolfo S., and Herzog B. Fileprints: Identifying file types by n-gram analysis. 6th IEEE Information Assurance Workshop, June 2005.

32. Linn C. M., Rajagopalan M., Baker S., Collberg C., Debray S. K., and Hartman J. H. Protecting against unexpected system calls. Usenix Security Symposium, 2005.

33. Lo R.W., Levitt K.N., and Olsson R.A. Mcf: Malicious code filter.

Computers and Society, pages 541–566, 1995.

34. Masri W. and Podgurski A. Using dynamic information flow analysis to detect attacks against applications. In Proceedings of the 2005 Workshop on Software Engineering for secure sytems –Building Trustworthy Applications, 30, May 2005.

35. McGraw G. and Morrisett G. Attacking malicious code: A report to the infosec research council. IEEE Software, 17(5):33–44, 2000.

36. Milenkovic M., Milenkovic A., and Jovanov E. Using instruction block signatures to counter code injection attacks. ACM SIGARCH Computer Architecture News, 33:108–117, March 2005.

Додаток А

Рис. 1 Використання сигнатур на виявлення шкідливого програмного забезпечення з github.

Додаток Б

Рис. 2 Використання шістнадцяткового коду для виявлення вмісту шкідливого програмного засобу в файлі.