Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ДОПУСТИТИ ДО ЗАХИСТУ:
В.о. завідувача кафедри
кібербезпеки та захисту інформації

_____ Іван ПАРХОМЕНКО
«_» червня 2023 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи

галузь знань 12 Інформаційні технології
(шифр і назва галузі знань)
спеціальність 125 Кібербезпека
(код і назва спеціальності)
освітній ступень бакалавр
освітня програма Кібербезпека
(назва освітньо-професійної програми)
на тему: Методи виявлення та блокування поліморфних вірусів в операційних системах

Виконавець: студент IV курсу, групи КБ-42

_____ **Еміль Ісмаілов**
(підпис) (ім'я, прізвище)

	Ім'я, прізвище	**Підпис**
Керівник	Олександр ЛАПТЄВ	

Нормоконтроль	Олександр ТОРОШАНКО	

Київ 2023

Міністерство освіти і науки України
Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій
Кафедра кібербезпеки та захисту інформації

ЗАТВЕРДЖЕНО:
В.о. завідувача кафедри кібербезпеки
та захисту інформації

_____Сергій ТОЛЮПА
«24» жовтня 2022 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності 125 Кібербезпека
(код і назва спеціальності)

освітньої програми Кібербезпека
(назва освітньої програми)

Студентові **КБ-42** **Ісмалову Емілю Ялчин огли**
(група) (прізвище ім'я по-батькові)

Тема кваліфікаційної роботи Методи виявлення та блокування поліморфних вірусів в операційних системах

1. ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Тема кваліфікаційної роботи затверджена на засіданні кафедри кібербезпеки та захисту інформації протокол №3 від 20.10.2022 р.

2. ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Методи виявлення та блокування поліморфних вірусів в операційних системах

3. ЗМІСТ РОЗРАХУНКОВО-ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Поняття поліморфного вірусу, характеристики методів виявлення, методи блокування поліморфних вірусів в операційних системах

4. ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Практична цінність	Розглянуті актуальні методи виявлення та розроблені методики детектування поліморфних вірусів в операційних системах

5. ДАТА ВИДАЧІ ЗАВДАННЯ

Дата видачі завдання: 24 жовтня 2022 року

Завдання видав _____ (підпис) Олександр ЛАПТЄВ (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) ІСМАІЛОВ Еміль Ялчин огли (ім'я, прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів робіт	Строки виконання робіт (початок-кінець)	Відмітка про виконання
1	Уточнення постановки задачі	24.10.2022 – 27.10.2022	*виконано*
2	Аналіз літератури	28.01.2023 – 11.02.2023	*виконано*
3	Обґрунтування вибору рішення	12.02.2023 – 24.02.2023	*виконано*
4	Збір даних	25.02.2023 – 24.03.2023	*виконано*
5	Написання першого розділу роботи	25.03.2023 – 07.04.2023	*виконано*
6	Написання другого розділу роботи	08.04.2023 – 20.04.2023	*виконано*
7	Написання третього розділу роботи	21.04.2023 – 05.05.2023	*виконано*
8	Підготовка ілюстративного матеріалу	06.05.2023 – 20.05.2023	*виконано*
9	Отримання рецензій	21.05.2023 – 04.06.2023	*виконано*
10	Оформлення пояснювальної записки	05.06.2023 – 08.06.2023	*виконано*
11	Підготовка до захисту	09.06.2023 – 12.06.2023	*виконано*

Завдання видав _____ (підпис) Олександр ЛАПТЄВ (ім'я, прізвище)

Завдання прийняв до виконання _____ (підпис) ІСМАІЛОВ Еміль Ялчин огли (ім'я, прізвище)

Термін подання кваліфікаційної роботи до ЕК 12 червня 2022 року

РЕФЕРАТ

Кваліфікаційної робота складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел, має 62 сторінки основного тексту, 4 сторінки додатків, 19 рисунків та 1 таблицю. Список використаних джерел містить 20 найменувань і займає 2 сторінки.

Методи дослідження кваліфікаційної роботи:

- аналіз літератури;
- аналіз документів;
- порівняння;

Метою роботи є розробка методики детектування поліморфних вірусів в інформаційних системах.

Для досягнення зазначеної мети поставлено наступні завдання:

- Огляд акутальних програмних загроз для операційної системи.
- Аналіз існуючих методів виявлення та блокування вірусів.
- На основі проведених аналізів розроблена Методика підвищення ефективності виявлення полыморфних ШПЗ.

Об’єктом дослідження є процес підвищення ефективності виявлення та блокування поліморфних вірусів в інформаційних системах.

Предметом дослідження є ефективність методів виявлення у рамках інформаційної безпеки операційної системи.

Практичною цінністю отриманих результатів є розроблена методика підвищення ефективності виявлення та блокування поліморфних вірусів у рамках інформаційної безпеки підприємства.

Ключові слова: комп’ютерні віруси, операційні системи, сигнатурний аналіз, евристичний аналіз, зловмисник, протидія.

ЗМІСТ

ВСТУП

Актуальність даної роботи визначається тією обставиною, що на даний момент поліморфні вірусі - це одна з найрозповсюдженіших та складних видів шкидливого програмного забезпечення..

У сучасному інформаційному суспільстві, де комп'ютери та мережі проникають у всі сфери життя, безпека інформації відіграє найважливішу роль. Однією з найсерйозніших загроз для інформаційної безпеки є комп'ютерні віруси. Ці шкідливі програми здатні завдати значної шкоди, порушити роботу системи та увірватися у конфіденційні дані. На особливу увагу заслуговують поліморфні віруси, які становлять особливу загрозу, оскільки вони здатні змінювати свою структуру і ефективно оминати традиційні методи виявлення.

Поліморфні віруси - це віруси, які здатні змінювати свою структуру та функції, зберігаючи свою шкідливість. Вони використовують різні методи для приховання своєї присутності та обходу традиційних антивірусних методів. Основною метою поліморфних вірусів є обхід системи захисту, щоб вони могли як інфікувати систему, а й залишатися непоміченими протягом багато часу.

Об’єктом дослідження є процес підвищення ефективності виявлення та блокування поліморфних вірусів в інформаційних системах.

Предметом дослідження є ефективність методів виявленгя у рамках інформаційної безпеки операційної системи.

Метою роботи є дослідження та аналіз методів виявлення та блокування поліморфних вірусів в операційних системах.

РОЗДІЛ 1
АНАЛІЗ КОМП'ЮТЕРНИХ ВІРУСІВ. ПОЛІМОРФНІ ВІРУСИ

1.1 Історія розвитку

У наш час комп'ютерні віруси стали невід'ємною частиною нашого цифрового життя. Вони можуть викликати серйозні проблеми від збоїв у роботі операційної системи до витоків конфіденційної інформації. Але як і коли з'явилися ці невеликі, але небезпечні програми?

Історія розвитку комп'ютерних вірусів налічує вже кілька десятиліть, починаючи з появи перших простих програм-паразитів у 1970-х роках. Тоді ці "віруси" несли скоріше експериментальний характер і були способом демонстрації можливостей нових технологій. Однак, з часом, комп'ютерні віруси набули дедалі більш витончених форм і стали засобом зловмисників для реалізації своїх зловмисних цілей.

З появою інтернету наприкінці 20-го століття комп'ютерні віруси отримали нові можливості для поширення та заподіяння шкоди. Разом з розвитком мереж та програмного забезпечення, віруси ставали все більш складними та витонченими. Їхні цілі тепер розширилися від простого пошкодження даних до крадіжки особистої інформації, шантажу і навіть кібершпигунства.

Проте прогрес у боротьбі з комп'ютерними вірусами також не стоїть на місці. Компанії з розробки антивірусного програмного забезпечення постійно вдосконалюють свої продукти, щоб забезпечити захист від нових загроз, що змінюються. Також підвищується поінформованість користувачів про заходи безпеки, що допомагає знизити ризик зараження та поширення вірусів.

Історія розвитку комп'ютерних вірусів є важливою частиною розуміння сучасної кібербезпеки. Вивчення причин та методів, якими віруси проникають у наші системи, допомагає розробникам створювати ефективні заходи захисту. Аналіз їхньої еволюції та наслідків дозволяє нам бути більш пильними та готовими до нових викликів у боротьбі з кіберзагрозами.

У цій главі буде проведено огляд історії розвитку комп'ютерних вірусів, починаючи з ранніх проявів їхнього існування та закінчуючи сучасними загрозами. Буде розглянуто основні типи вірусів, їх поширення та наслідки, а також важливі моменти у їх боротьбі та захисті від них. Зрештою, ця робота допоможе нам краще зрозуміти природу комп'ютерних вірусів та способи протидії їм, щоб забезпечити безпеку нашого цифрового середовища.

Першим комп'ютерним вірусом, відомим історія, був вірус " Creeper " , створений 1971 року програмістом Бобом Томлінсоном. У той час комп'ютери працювали в мережі ARPANET, попередниці інтернету, і Creeper був розроблений як експериментальна програма для демонстрації того, як можна пересуватися між різними вузлами мережі.

Creeper був написаний мовою програмування PDP-10 і поширювався шляхом інфікування віддалених систем. Коли вірус потрапляв на комп'ютер, він відображав повідомлення "Im the creeper, catch me if you can!" ("Я - повзунок, спіймай мене, якщо зможеш!"). Потім вірус копіював сам себе на інші комп'ютери в мережі.

Рисунок 1.1 - Реконструкція тексту, що відображається вірусом CREEPER.

Проте, замість того, щоб завдавати шкоди або завдавати пошкоджень, "Creeper" був скоріше демонстрацією можливостей програмування та переміщення по мережі.

У відповідь на Creeper була розроблена перша програма-антивірус під назвою Reaper (Жнець), яка була задіяна для видалення вірусу з заражених систем.

Важливо відзначити, що на той час термін "вірус" не використовувався у його сучасному розумінні. Комп'ютерні віруси, як явище, тільки розпочинали свій розвиток, і "Creeper" можна швидше віднести до простих програм-паразитів, які не становлять реальної загрози.

Незважаючи на свою незначність та некритичну дію, "Creeper" заклав основу для розвитку комп'ютерних вірусів і став відправною точкою для подальших досліджень та розробок у галузі кібербезпеки. Його поява показала, що комп'ютери можуть бути піддані програмним атакам і що необхідні заходи для виявлення та боротьби з такими загрозами. З часом віруси стали все складнішими і небезпечнішими, і з'явилися нові види шкідливого програмного забезпечення, такі як троянські програми, черв'яки та шпигунське ПЗ.

Зараз, більш як півстоліття після появи першого вірусу, кіберзлочинці розробляють все більш витончені і небезпечні віруси, які завдають величезної шкоди як приватним користувачам, так і організаціям. Боротьба з цими загрозами вимагає постійного вдосконалення антивірусного програмного забезпечення та дотримання заходів безпеки з боку користувачів.

Історія першого комп'ютерного вірусу "Creeper" наголошує на важливості розвитку та застосування засобів захисту від вірусів та інших кіберзагроз. Це також нагадує нам про постійну необхідність бути пильними та обізнаними про останні тенденції у кібербезпеці, щоб забезпечити безпеку наших цифрових систем та даних.

Другим значущим комп'ютерним вірусом в історії є вірус " Elk Cloner " , створений 1982 року Річардом Скрином. "Elk Cloner" став першим відомим вірусом, який розповсюджується через переносні носії інформації, такі як дискети.

Вірус "Elk Cloner" був написаний для комп'ютерів Apple II і поширювався інфікуванням програм на дискетах. Коли заражена дискета було вставлено у комп'ютер, вірус копіював себе комп'ютер і перезавантажував систему. Після перезавантаження на екрані з'являвся невеликий віршик:

Elk Cloner:

The program with a personality

It will get on all your disks

It will infiltrate your chips

Yes, it's Cloner!

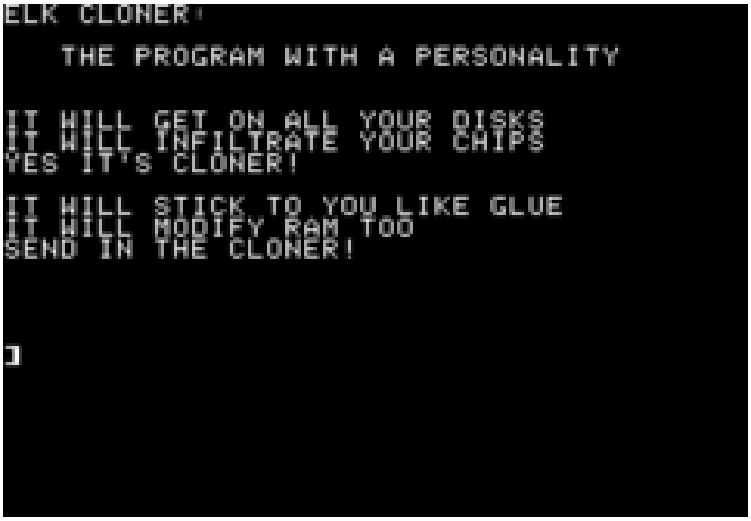


Рисунок 1.2 - Текст, який з'являвся на дисплеї після зараження

Цей вірус, хоч і не завдавав серйозної шкоди комп'ютерам або даним, отримав значну увагу свого часу. Він став першим прикладом вірусу, який поширювався через переносні носії, що відкрило нові можливості швидкого поширення вірусів між комп'ютерами.

Elk Cloner також показав, що комп'ютерні віруси можуть бути нав'язливими і залишатися в системі протягом тривалого часу, впливаючи на роботу комп'ютера і викликаючи деякі незручності для користувача.

У результаті поява вірусу "Elk Cloner" наголосила на необхідності розробки та застосування захисних заходів проти вірусів та навчання користувачів правилам безпеки при роботі з переносними носіями інформації. Цей вірус також проклав шлях для подальшого розвитку комп'ютерних вірусів та посилення уваги до кібербезпеки.

Один із знаменитих та руйнівних вірусів, який привернув увагу всього світу, - це вірус "ILOVEYOU" (Люблю тебе), який з'явився у 2000 році.

Вірус "ILOVEYOU" розповсюджувався через електронну пошту, маскувавшись під прикріплений файл під назвою "LOVE-LETTER-FOR-YOU.TXT.vbs". Коли

користувачі відкривали цей файл, вірус розпочинав свою роботу. Він копіював себе на комп'ютер і відправляв себе адресною книгою жертви, таким чином поширюючись ще ширше.

Рисунок 1.3 - Електронний лист із прикріпленим шкідливим файлом

Основна особливість вірусу полягала в тому, що він прав файли з розширеннями JPG, JPEG, VBS, JS, CSS, HTM, HTML, HTML і MP3, а також заміняв їх на копії самого себе. Внаслідок жертви втрачали свої цінні файли і система ставала нестабільною.

"ILOVEYOU" було розроблено двома молодими філіппінськими програмістами, Майклом Бациллою та Онеллі Де Гусманом. Вірус швидко поширився у всьому світі, завдаючи величезних економічних збитків. Він привернув до себе увагу міжнародних ЗМІ та громадськості, підкресливши вразливість комп'ютерних систем та необхідність більш ефективних заходів щодо захисту даних та боротьби з вірусами.

"ILOVEYOU" є прикладом того, як комп'ютерні віруси можуть завдати реальних збитків, не тільки для окремих користувачів, а й для організацій і навіть держав. Він наголосив на важливості навчання користувачам основам кібербезпеки, таких як обережність при відкритті та завантаженні файлів, а також необхідність використання надійних антивірусних програм та регулярного оновлення системного програмного забезпечення.

Давайте перейдемо до розгляду ще одного важливого комп'ютерного вірусу - Code Red (Кодовий червоний).

"Code Red" з'явився в 2001 році і став одним із найпоширеніших та руйнівних вірусів того часу. Він був націлений на операційну систему Microsoft Windows, зокрема на веб-сервери, що працюють під керуванням Microsoft Internet Information Services (IIS).

Вірус "Code Red" використовував уразливість у IIS, відому як "Buffer Overflow", щоб отримати доступ до комп'ютера та поширитися далі. Коли система була заражена, вірус перезавантажував комп'ютер і починав шукати нові вразливі сервери для подальшого поширення. Вірус також створював нову копію самого себе та залишав слід у вигляді зміни індексної сторінки на сервері.

"Code Red" отримав широку популярність через свою здатність швидко поширюватися та інфікувати тисячі серверів по всьому світу. Він викликав серйозні проблеми в роботі веб-серверів і спричинив значні фінансові втрати для багатьох організацій.

Одним із найвідоміших наслідків атаки "Code Red" був напад на веб-сайт Білого дому (whitehouse.gov). Внаслідок атаки сайт тимчасово став недоступним для відвідувачів.

Рисунок 1.4 - Скріншот веб-сайту, пошкодженого комп’ютерним хробаком Code Red.

У відповідь на спалах Code Red компанія Microsoft випустила патч, що виправляє вразливість у IIS. Проте вірус продовжував активно поширюватися, оскільки багато систем не було оновлено вчасно.

Історія "Code Red" підкреслює не лише серйозність загрози, яку становлять комп'ютерні віруси, а й важливість своєчасного оновлення програмного забезпечення та застосування патчів безпеки. Цей вірус став ще одним нагадуванням про необхідність посиленої уваги до кібербезпеки та захисту інформаційних систем від шкідливих програм та атак з боку зловмисників.

"Slammer" з'явився в січні 2003 року і став одним із найбільш руйнівних комп'ютерних вірусів в історії. Він був класифікований як "хробак" і був призначений для системи баз даних, що працює під керуванням Microsoft SQL Server.

Вірус "Slammer" використовував уразливість у SQL Server, відомому як "SQL Slammer". Ця вразливість швидко дозволяла вірусу й автоматично розповсюджувалась по мережі без необхідності взаємодії з користувачем. «Slammer» створив випадкові IP-адреси і відправив себе на цю адресу, заражаючи нові системи.

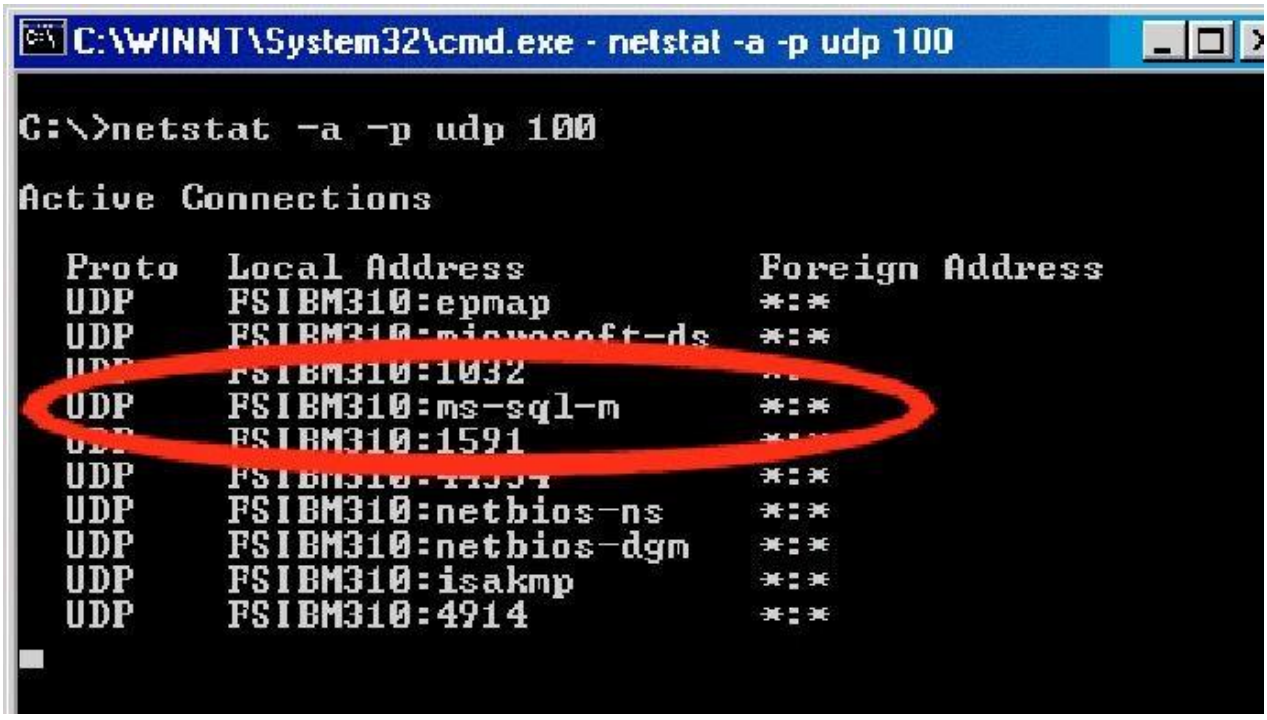


Рисунок 1.5 - SQL Slammer

"Slammer" був неймовірно швидким і агресивним вірусом. Він здатний був поширюватись настільки швидко, що за кілька хвилин заражав безліч комп'ютерів по всьому світу. За раховані хвилини він перевантажував мережі та системи, викликаючи серйозні проблеми в роботі Інтернету та багатьох компаній.

"Slammer" став причиною масових збоїв в мережах, привів до відключення інтернет-служб, затримкам в авіації та відміні операцій в медичних установах. Загальний економічний збиток, нанесений вірусом, був всерйозним.

Цей вірус підтвердив важливість оновлення та усунення вразливостей у системах, а також необхідність посилення заходів щодо захисту комп’ютерних сетей і баз даних. Він також привертає увагу до необхідності впровадження заходів щодо реагування на кібератаки та взаємодії в розвитку мережевої безпеки.

Історія "Slammer" служить згадкою про те, наскільки важливо забезпечувати безпеку інформаційних систем і активно боротися з уразливістю, щоб мінімізувати потенційні загрози та пов'язані з комп'ютерними проблемами вірусами та атаками.

"Stuxnet" був виявлений в 2010 році і відрізнявся від багатьох інших вірусів своїм унікальним характером і ціллю. Цей вірус був спеціально розроблений для атаки на промислові системи управління, в особливості на системи, які використовуються в ядерних установках Ірану.

«Stuxnet» представляв собою безліч складних і добре спланованих вірусів, які використовували уразливість, щоб проникнути в систему управління. Він поширювався через заражені USB-флешки та експлуатував уразливості в програмному забезпеченні SCADA (система збору та управління даними).

Особливістю «Stuxnet» була його здатність проникати в систему і впливати на роботу промислового обладнання, в тому числі на центрифугах, які використовуються для обігріву урана в ядерних установках. Вірус змінив параметри роботи обладнання, що приводило до його несправності та збоїв у ядерному процесі.

"Stuxnet" був революційним вірусом, який відкрив нову еру кіберпрограційних дій і надав можливість фізичного впливу на промислові системи через комп'ютерні віруси. Він викликав широкий резонанс у світовому співтоваристві та привернув увагу до проблем кібербезпеки в промислових секторах.

Цей вірус також підтвердив важливість розробки та застосування заходів щодо захисту критичних систем, таких як важливі системи управління промисловим обладнанням. Він послужив толчком для підвищення кібербезпеки в промислових і

ядерних секторах, а також для розробки більш досконалих засобів виявлення і протидії подібним атакам.

"Stuxnet" був розроблений як комплексне кіберроз'єднання, призначене для атак на системи управління промисловим обладнанням, зокрема, на системи, що використовуються в ядерних установках Ірану. Ось деякі ключові особливості та методи, які використовував "Stuxnet":

- Уразливості та експлуатація: "Stuxnet" використовував кілька вразливостей в операційних системах Microsoft Windows і програмного забезпечення SCADA (система збору й управління даними), які використовувалися для управління промисловими процесами. Ці уразливості дозволили вірусу проникнути в систему і поширитися.
- Автопоширення: "Stuxnet" використовував кілька методів для одного комп'ютера на інших поширених. Вірус може поширюватися через заражені USB-накопителі, мережеві шляхи та навіть через вразливість у мережевих протоколах. Це дозволило йому швидко поширювати і заразити безліч систем.
- Нульовий день: Важним аспектом "Stuxnet" було використано кілька "нулевих днів" - уразливостей, про які виробники ще не знали і для яких не було виправлено чи виправлено. Це давало перевагу вірусу, оскільки заражені системи не були підготовлені для захисту від таких атак.
- Передача через мережу: Вірус використовував механізми поширення через мережу, щоб передавати себе на інші комп’ютери та заражати їх. Він шукав уразливі системи SCADA, а також комп'ютери, пов'язані з програмним забезпеченням, які використовуються в ядерних установках.
- Маскування та скритність: «Stuxnet» був розроблений із застосуванням складних методів маскування та скритності. Він мав можливість скрити свою присутність в системі, змінювати файли та системні ресурси. Вірус також використовував цифрові сертифікати, що створювало враження його довжини та довіри.
- Напад на промислові системи: Однією з ключових цілей "Stuxnet" було управління промисловим обладнанням, зокрема системами, що використовуються в

ядерних установках. Вірус здатний впливати на роботу промислових процесів, змінюючи параметри роботи обладнання та викликаючи його роботу.

Історія розвитку комп'ютерних вірусів є важливим аспектом кібербезпеки і технічного розвитку. За десятиліття комп'ютерні віруси еволюціонували і стали все більш складними і зруйнованими.

Перші комп'ютерні віруси, такі як "Creeper" і "Elk Cloner", виникли на початку 1970-х років і були швидше експериментами, ніж реальними загрозами. Однак вони поклали початок тому, що почнеться розвиток і поширення вірусів в наслідку.

Со временем вирусы стали более складными и целенаправленными. "Slammer" і "Stuxnet" - це лише два приклади вірусів, які викликають серйозні наслідки і привертають увагу всього світу. "Slammer" привів до масових збоїв і проблем сетей, а "Stuxnet" продемонстрував можливість фізичного впливу на промислові системи через комп'ютерні атаки.

Історія розвитку комп'ютерних вірусів показує, що кібербезпека є неотъємною частиною нашої сучасної життя. Кожен новий вірус і атака пропонують нам покращити захист системи та розробити нові методи виявлення та запобігання загрозам.

Важливо усвідомлювати, що комп'ютерні віруси можуть бути причиною значного збитку як окремим користувачам, так і компаніям і державам в цілому. Тому впровадження заходів щодо кібербезпеки, таких як регулярні оновлення програмного забезпечення, використання сильних паролів, навчання співробітників основам кібербезпеки та створення потужної системи захисту, є необхідністю.

1.2. Види та класифікація комп’ютерних вірусів

Класифікація вірусів і шкідливих програм є важливим інструментом для розуміння різних типів загроз в кіберпросторі. У світі комп'ютерної безпеки існує широкий спектр шкідливих програм, що включають в себе не тільки віруси, але й інші види шкідливого ПО, такі як віруси, троянські програми, шпіонське ПО і рекламне ПО.

Класифікація шкідливих програм допомагає організаціям і фахівцям з кібербезпеки у визначенні характеристик, поведінки та методів поширення кожного типу загрози. Це дозволяє розробити відповідні заходи захисту та виявити стратегії, щоб більш ефективно боротися з ними.

Важливо, що класифікація вірусів і відзначених шкідливих програм не є статичною і постійною, оскільки загрози в кіберпросторі постійно еволюціонують і приєднуються до нових технологій і захисним мерам. Однак існує кілька загальноприйнятих категорій, які широко використовуються для класифікації шкідливих програм.

При класифікації вірусів і шкідливих програм зазвичай враховуються їх функціональні можливості, способи, методи зараження і впливу на заражені системи. Кожен тип загрози має свої особливості та потенційні наслідки, які можуть змінитися від простого блокування доступу до даних до кращої конфіденційної інформації або фізичного пошкодження системи.

Поняття класифікації вірусів і шкідливих програм дозволяє підвищити кібербезпеку, розробляти відповідні стратегії захисту, виявити і реагувати на різні типи загроз.

Ось короткий огляд основних видів вірусів та їх класифікації:

- Віруси: Віруси є найбільш поширеним типом шкідливої програми. Вони знаходяться у виконуваних файлах або програмах і можуть поширюватися шляхом передачі завантажених файлів або засобів мережі. Віруси можуть викликати різні негативні наслідки, від знищення даних до блокування доступу до системи.

- Хробаки: Хробаки - це саморозповсюджуються шкідливі програми, які використовують мережі для поширення та інфікування інших комп'ютерів. Вони зазвичай знаходяться в небезпечних системах і поширюються шляхом копіювання себе та відновлення копій через мережу. Хробаки можуть викликати перевантаження мережі, приводити до зниження продуктивності системи та поширювати інші шкідливі програми.

- Троянські програми: Троянські програми являють собою шкідливі програми, які маскуються під корисні або бажані додатки. Коли користувач встановлює

троянську програму, він може отримувати несанкціонований доступ до системи, створювати особисті дані, перехоплювати паролі або встановлювати додаткове шкідливе ПЗ.

- Рекламне ПЗ: Рекламне ПЗ є програмним забезпеченням, яке відображає небажану рекламу на комп’ютері користувача. Його можна встановити разом з іншими додатками або завантажити без дозволу користувача. Рекламне ПЗ може завантажувати роботу комп'ютера, відображати назойливу рекламу та збирати інформацію про користувача.

- Шпигунське ПЗ: Шпигунське ПЗ (шпигунське програмне забезпечення) слідкує за активністю користувача, збирає приватну інформацію, таку як паролі, дані банківських карт, історію відвідуваних веб-сайтів і передає їх зловмиснику. Шпигунське ПЗ часто потрапляє в систему через небажані внесення електронної пошти, підготовлені завантаження або вразливості в браузері.

Це лише деякі види вірусів і шкідливих програм. Класифікація може бути більш детальною та включати інші категорії в залежності від їх характеристик і дій. Ця класифікація допомагає фахівцям з кібербезпеки розробляти відповідні методи виявлення та захисту від різних видів загроз.

Віруси - це один із самих поширених і відомих типів шкідливих програм. Вони отримали свою назву завдяки аналогії з біологічними вірусами, які також здатні інфікувати живі організми.

Віруси характеризуються наступними особливостями:

- Впровадження: Віруси знаходяться у виконуваних файлах, програмах або документах. Вони приєднані до них і використовують їх код або структуру для свого розмноження і форми.

- Розмноження: Віруси розмножуються шляхом інфікування інших файлів або програм. Коли запускається пошкоджений файл або програма, вірус активується і поширюється на інші файли або системи.

- Поширення: Віруси можуть поширюватися різними способами, включаючи обмін файлами, електронну пошту, мережеві підключення та портативні пристрої.

Вони можуть передаватись як самостійні файли або видалятися в інші файли та програми.

- Попередження: Один із головних негативних ефектів вірусів - це їх здатність поновлювати або знищувати дані чи програми на зараженій системі. Віруси можуть викликати потерю даних, неправильне функціонування програми, блокування доступу до системи або навіть повне відключення комп'ютера.
- Скритність: Віруси часто стремляться залишатися незаметними в зараженій системі. Вони можуть змінити свою структуру або використовувати методи маскування, щоб уникнути виявлення антивірусних програм.
- Взаємодія з хостом: Віруси можуть взаємодіяти з операційною системою та іншими програмами, використовуючи різні методи. Вони можуть змінювати файли, реєстри або налаштування системи, а також копіювати себе в інші місця для забезпечення свого розмноження.
- Віруси активуються за певних умов, таких як запуск зараженого файлу або підключення до зараженої мережі. Після активації вони починають своє руйнівне або небажане дію.

Віруси можуть мати різні цілі, включаючи поширення інших шкідливих програм, кражу особистої інформації, причину фінансової втрати, порушення роботи системи або просто створення хаоса та неудобств.

Хробаки - це тип шкідливих програм, які здатні самостійно поширюватися та інфікувати інші комп'ютери або пристрої в мережі. Вони відрізняються від вірусів тим, що не вимагають підключення до виконуваних файлів або програм для свого поширення. Замість цього вони використовують мережеві підключення та вразливості в операційній системі або додатках для інфікування нових хостів.

Ось основні особливості хробаків:

- Саморозповсюдження: Одна з ключових особливостей хробаків є їх здатністю до самораспространению. Як тільки хробак потрапляє на комп'ютер або пристрій, він ищеть інші небезпечні системи в мережі і намагається передати себе на них. Це дозволяє хробакам швидко поширюватися та заражати велику кількість хостів.

- Вразливості та експлойти: Хробаки часто використовують відомі вразливості в операційній системі або додатках для свого поширення. Вони можуть використовувати відомі експлойти, щоб отримати несанкціонований доступ до системи та встановити себе на ній. Це може включати вразливості в протоколах мережі, слабких паролях або несправних програмних компонентах.

- Мережева взаємодія: Хробаки зазвичай використовують різні мережеві протоколи та служби для свого розвитку. Вони можуть використовувати загальні мережеві порти, електронну пошту, обмін файлами, віддалені підключення або інші методи зв’язку для передачі себе на інші пристрої в мережі.

- Знищення і негативний вплив: Деякі віруси можуть викликати негативний вплив на уражені системи. Це може включати видалення або зміну файлів, перезавантаження мережі, роботу програм або навіть фізичне пошкодження системи. Деякі хробаки також можуть використовуватися для встановлення додаткового шкідливого ПО або створення ботнетів.

- Приклади хробаків: Прикладами відомих хробаків є ШПЗ ILOVEYOU, який у 2000 році спричинив значні фінансові втрати та пошкодження систем у всьому світі, і хробак Conficker, який активно поширювався у 2008 році, заражаючи мільйони комп’ютерів.

Для захисту від хробаків важливо регулярно оновлювати операційну систему та програмне забезпечення, використовувати мережеві екрани та антивірусні програми, а також бути обережними при відкритті вкладених електронних листів і завантаженні файлів із ненадійних джерел.

Троянські програми, або трояни, є типом шкідливого програмного забезпечення, які маскуються під корисні або легітимні програми, щоб обманювати користувачів і отримувати несанкціонований доступ до їх систем. Трояни отримали своє ім'я в честь міфа про Троянську лошадь, яка була використана для вторгнення у місто Трою.

Ось основні особливості троянських програм:

- Маскування: Трояни обманюють користувачів, представляючи корисні або необхідні програми. Вони можуть створюватися антивірусними програмами,

програмами для оптимізації системи, відеоплеєрами та ін. д. Часто трояни поширюються через фішингові листи, неперевірені завантаження або шкідливі сайти.

- Несанкціонований доступ: ціль троянів є отриманням несанкціонованого доступу до зараженої системи. Після встановлення на комп’ютер або пристрій троян може виконувати різні дії, такі як збір особистої інформації, видалене управління системою, встановлення інших шкідливих програм або використання комп’ютера в якості частини ботнету.

- Підконтрольність: Троянські програми часто надають зловмиснику віддалений доступ до зараженої системи. Це означає, що зловмисник може керувати комп'ютером або пристроєм видачі, виконувати команди, передавати файли, переглядати екран тощо. д. У деяких випадках троянські програми можуть бути використані для шпіонажа або допомоги.

- Камуфляж: Трояни можуть використовувати різні методи, щоб відкрити свою присутність і уникнути виявлення. Це може включати шифрування свого коду, використання руткітів або зміну системних файлів. Деякі трояни також можуть використовувати антивірусні програми та захисні заходи.

- Передача даних: Троянські програми можуть перехоплювати і передавати дані із зараженої системи. Це може включати перехоплення паролів, банківських даних, особистої інформації або інших конфіденційних даних. Отримані дані можуть бути використані злочинцем для фінансових шахрайств, країв особистості або інших протиправних дій.

Рекламне програмне забезпечення (adware) - це тип шкідливого програмного забезпечення, яке відображає нав'язливу рекламу на комп'ютері або пристрої користувача. Основна мета рекламного ПЗ полягає у генерації доходу для творців шляхом показу рекламних оголошень, часто без згоди користувача.

Ось деякі особливості рекламного ПЗ:

- Нав'язлива реклама: Рекламне програмне забезпечення відображає рекламні оголошення в різних формах, включаючи спливаючі вікна, банери, вставки у веб-сторінки або відеоплеєри. Ці оголошення можуть бути дратівливими та заважати нормальному використанню комп'ютера або пристрою.

- Встановлення в пакеті з іншим програмним забезпеченням: Рекламне програмне забезпечення часто включається в пакетне програмне забезпечення, яке користувач завантажує та встановлює з Інтернету. Це може бути представлено як додаткові компоненти або параметри установки, які користувач приймає без належної уваги. Тому важливо бути уважним при встановленні нового програмного забезпечення та стежити за додатковими компонентами.

- Стеження за даними: Деякі рекламні ПЗ можуть збирати інформацію про користувача, його звички, відвідувані веб-сайти або пошукові запити. Ці дані можуть бути використані для персоналізації реклами або передані третім сторонам для маркетингових цілей. Це може викликати порушення приватності та недовіру користувачів.

- Негативний вплив на продуктивність: Рекламне програмне забезпечення може знижувати продуктивність комп'ютера або пристрою, оскільки воно займає ресурси системи для відображення рекламних оголошень. Це може призвести до уповільнення роботи, перебоїв у мережному з'єднанні або зависання програм.

- Потенційна загроза безпеці: Деякі форми рекламного програмного забезпечення можуть становити загрозу для безпеки системи. Це може бути пов'язано зі шкідливими посиланнями, підробленими веб-сторінками або рекламними оголошеннями, які можуть направляти користувача на шкідливі сайти або змушувати його завантажувати шкідливі файли.

Для захисту від рекламного програмного забезпечення рекомендується використовувати надійне антивірусне програмне забезпечення, оновлювати операційну систему та програми, уникати сумнівних веб-сайтів та неперевірених завантажень. Також корисно читати умови ліцензійних угод при встановленні нового програмного забезпечення та бути уважними до додаткових компонентів, які пропонуються під час встановлення.

Шпигунське програмне забезпечення (spyware) - це тип шкідливого програмного забезпечення, розробленого для збору інформації про користувачів без їхньої згоди. Метою шпигунського ПЗ є непомітне спостереження за активністю користувача, збирання особистих даних, паролів, історії перегляду веб-сторінок,

повідомлень, що надсилаються, та іншої конфіденційної інформації. Ось деякі особливості шпіонського ПО:

Ось деякі особливості шпигунського ПЗ:

- Прихованість: Шпигунське програмне забезпечення працює приховано на комп'ютері або пристрої користувача, зазвичай без його свідомості. Вона може бути впроваджена в програмне забезпечення, що завантажується з Інтернету, або проникнути на комп'ютер через уразливість безпеки. Важливо, що шпигунське програмне забезпечення може бути встановлено не тільки зловмисниками, але й деякими компаніями для збору даних про користувача в комерційних цілях.

- Збір даних: Основною метою шпигунського ПЗ є збір інформації про користувача. Це може включати особисті дані, такі як імена, адреси, номери телефонів, адреси електронної пошти, а також дані про фінанси, банківські реквізити, паролі, історію браузера, листування електронною поштою та багато іншого. Зібрані дані можуть бути використані для різних цілей, включаючи шахрайство, крадіжку особи або шпигунство.

- Кейлоггінг: Шпигунське програмне забезпечення може включати функцію кейлоггера, який записує всі натискання клавіш користувача на клавіатурі. Це дозволяє зловмиснику отримати доступ до паролів, логін, повідомлень та інших конфіденційних даних, які користувач вводить.

- Віддалене керування: Часто шпигунське програмне забезпечення має можливість віддаленого керування зараженим комп'ютером або пристроєм. Це дозволяє зловмиснику виконувати команди, отримувати актуальні дані, завантажувати додаткові модулі чи оновлення. Таке віддалене керування дозволяє зловмиснику залишатися непомітним та підтримувати постійний контроль над зараженою системою.

- Порушення приватності: Шпигунське ПЗ порушує приватність користувачів, захоплюючи та передаючи їх особисті дані третім сторонам. Це може спричинити загрозу безпеці, втрату конфіденційних даних, а також зловживання їхньою особистою інформацією.

Шпигунське програмне забезпечення (spyware) має ряд технічних характеристик, які дозволяють йому непомітно проникати та збирати інформацію на зараженій системі. Ось деякі з основних характеристик шпигунського ПЗ:

- Прихованість: Шпигунське ПЗ зазвичай прагне залишатися невидимим на зараженій системі, щоб користувач не знав про його наявність. Воно може використовувати різні методи приховування своєї активності, включаючи зміну імені та місцезнаходження файлів, використання прихованих процесів чи сервісів, і навіть обхід звичайних методів виявлення антивірусними програмами.
- Завантаження та встановлення: Шпигунське програмне забезпечення може бути завантажене та встановлене на комп'ютер або пристрій користувача різними способами. Це може включати шкідливі посилання, електронні листи з вкладеннями, завантаження з ненадійних джерел, використання вразливостей безпеки або навіть використання соціальної інженерії, щоб переконати користувача виконати встановлення.
- Збір інформації: Метою шпигунського ПЗ є збір різноманітної інформації про користувача та його активність. Це може включати перехоплення та записування натискань клавіш, збирання даних із файлів та папок, перегляд та запис веб-сайтів, збирання інформації про систему, таку як IP-адреса, операційна система, встановлене програмне забезпечення, а також збирання особистих даних, таких як паролі , імена, адреси електронної пошти і т.д.
- Комунікація з віддаленим сервером: Шпигунське програмне забезпечення може встановлювати зв'язок з віддаленим сервером, щоб передавати зібрані дані або отримувати команди від зловмисника. Зазвичай це відбувається через протоколи мережі, такі як HTTP, HTTPS або FTP. Шпигунське програмне забезпечення може використовувати шифрування або інші методи для забезпечення конфіденційності комунікації.
- Оновлення та віддалене керування: Шпигунське програмне забезпечення може мати механізми оновлення та віддаленого керування. Це дозволяє зловмиснику оновлювати функціональність шпигунського програмного забезпечення,

впроваджувати додаткові модулі або виконувати віддалені команди на зараженій системі.

- Перехоплення мережної активності: Деякі шпигунські програми можуть перехоплювати мережеву активність користувача, включаючи веб-серфінг, надсилання та отримання повідомлень, роботу з електронною поштою тощо. Це може бути досягнуто шляхом зміни налаштувань проксі-сервера або перехоплення мережного трафіку через шкідливі драйвери.
- Персистентність: Шпигунське програмне забезпечення може використовувати різні методи, щоб залишатися активним на системі тривалий час. Це може включати додавання записів в автозавантаження, зміну системних файлів або налаштувань, створення прихованих завдань у планувальнику завдань тощо.

Обхід захисних механізмів: Шпигунське програмне забезпечення може намагатися обійти захисні механізми, такі як антивірусні програми, брандмауери або системи виявлення вторгнень. Це може бути досягнуто за допомогою власних методів шифрування, поліморфізму або використання вразливостей у захисті системи.

Важливо, що технічні характеристики шпигунського ПЗ можуть відрізнятися в залежності від його конкретної реалізації та цілей зловмисників. Ці характеристики можуть змінюватися та розвиватися у міру розвитку технологій та появи нових методів атак.

У період з 2015 по 2021 рік було виявлено та поширено різні шпигунські програми, які викликали значний інтерес та привернули увагу в галузі кібербезпеки. Ось кілька прикладів популярних шпигунських програм за цей період:

FinFisher (також відомий як FinSpy): Це комерційне шпигунське програмне забезпечення, розроблене компанією Gamma International. Воно було використано урядовими організаціями для незаконного нагляду за цілями. FinFisher має можливість перехоплення трафіку, запису натискань клавіш, доступу до веб-камери та мікрофону, а також збирання особистих даних користувача.

Pegasus: Розроблений ізраїльською компанією NSO Group, Pegasus є шпигунською програмою, яка була використана для кібершпигунства на мобільних

пристроях. Він може отримувати повний контроль над зараженим пристроєм, перехоплювати повідомлення, записувати розмови, отримувати доступ до контактів, фотографій, електронної пошти та інших даних.

GravityRAT: Цей шпигунський Троянський кінь, виявлений у 2015 році, був поширений через електронні листи та був націлений на урядові та військові установи в Індії. Він здатний перехоплювати натискання клавіш, знімати скріншоти, збирати інформацію про систему та надсилати отримані дані на віддалений сервер.

BlackEnergy: Шпигунський ПЗ BlackEnergy отримав широку популярність у результаті кібератак на українську енергетичну інфраструктуру. Він має можливість перехоплювати дані, отримувати віддалене керування та використовуватися для кібершпигунства.

Emotet: Emotet - це поширений банківський троян, який також може використовуватися для збору конфіденційних даних. Він був виявлений у 2014 році і з того часу активно поширюється. Emotet використовується для отримання доступу до фінансової інформації, перехоплення облікових даних, а також подальшого поширення інших шкідливих програм.

Шкідливе програмне забезпечення становить серйозну загрозу для комп'ютерів, мереж та конфіденційності користувачів. Було розглянуто різні види шкідливого ПЗ, такі як віруси, черв'яки, троянські програми, рекламне ПЗ та шпигунське ПЗ, та описано їх особливості та характеристики.

Кожен тип шкідливого ПЗ має свої особливості та здібності, і їхня мета може змінюватись від знищення даних до крадіжки конфіденційної інформації. Вони можуть поширюватися через різні канали, включаючи електронні листи, заражені посилання, шкідливі веб-сайти та порти.

Для захисту від шкідливих програм, важливо вживати заходів безпеки, таких як регулярне оновлення програмного забезпечення, встановлення антивірусних програм, використання сильних паролів, обережність при відкритті вкладень і посилань з ненадійних джерел, і регулярне резервне копіювання даних.

Боротьба зі шкідливим ПЗ є безперервним процесом, оскільки зловмисники постійно створюють нові методи та варіанти атак. Тому важливо бути пильними та

поінформованими про останні загрози у сфері кібербезпеки та вживати відповідних заходів для захисту своїх систем та даних.

1.3. Поліморфні віруси. Історія та ознаки

Комп'ютерні віруси є одними з найпоширеніших та найнебезпечніших видів шкідливого програмного забезпечення. Вони являють собою програми, здатні автономно поширюватися та інфікувати комп'ютерні системи, завдаючи шкоди інформації, програмам та операційним системам. Віруси можуть бути розроблені для різних цілей, включаючи крадіжку конфіденційних даних, здирство, знищення інформації або використання комп'ютерів як ботнети для здійснення кібератак.

Одним із найбільш складних та хитрих видів комп'ютерних вірусів є поліморфні віруси. Вони мають здатність змінювати свою структуру і кодування, щоб уникнути виявлення та блокування антивірусними програмами. Поліморфні віруси використовують різні методи, такі як зміна розміру, перестановка інструкцій або додавання непотрібного коду, щоб створити нові варіанти власного коду. Такі зміни роблять їх унікальними для кожної інфікованої системи, ускладнюючи виявлення та видалення вірусу.

Поліморфні віруси мають здатність швидко поширюватися і заражати велику кількість комп'ютерів та мереж. Вони можуть використовувати різні канали розповсюдження, включаючи електронну пошту, вразливості мережі, заражені носії інформації та інші методи. У разі інфікування поліморфним вірусом комп'ютерна система стає вразливою до кібератак, включаючи крадіжку особистих даних, фінансові шахрайства або порушення нормального функціонування системи.

Виявлення та боротьба з поліморфними вірусами є складним завданням для розробників антивірусних програм та фахівців з інформаційної безпеки. Традиційні методи, засновані на сигнатурному аналізі, стають дедалі менш ефективними проти поліморфних вірусів, оскільки можуть змінювати свою сигнатуру чи кодування, щоб уникнути виявлення. У зв'язку з цим виникає необхідність у розробці нових методів та технологій, які здатні ефективно виявляти та блокувати поліморфні віруси.

Одним із методів, якими поліморфні віруси досягають мінливості своєї структури, є шифрування свого коду. При кожному новому зараженні вірус створює унікальний ключ шифрування, який використовується для зашифрування його коду. Це дозволяє вірусу оминати засоби антивірусного захисту, оскільки зашифрований код не збігається з попередньо визначеними сигнатурами або зразками вірусів, якими оперує антивірусне програмне забезпечення. Незважаючи на існуючі методи виявлення, поліморфні віруси продовжують становити серйозну загрозу операційним системам. Розвиток нових методів та технологій для ефективного виявлення та блокування поліморфних вірусів залишається актуальним завданням у галузі інформаційної безпеки.

Висновки за розділом 1

У цьому розділі ми розглянули комп'ютерні віруси та поліморфні віруси, а також їхню історію розвитку, види та класифікації, а також особливості поліморфних вірусів.

Історія розвитку шкідливого програмного забезпечення показала, що комп'ютерні віруси з'явилися задовго до появи інтернету та мереж. Спочатку вони були створені як експерименти або просто для самовідтворення. Однак, з появою глобальних мереж, віруси набули нових можливостей і поширювалися набагато швидше.

Види та класифікації комп'ютерних вірусів показали, що вони можуть відрізнятися за своєю поведінкою, методами поширення та впливу на систему. Існують віруси, які просто розмножуються та займають ресурси комп'ютера, а також складніші віруси, здатні вкрасти конфіденційні дані або пошкодити систему. Класифікація вірусів дозволяє краще зрозуміти їхню природу та визначити відповідні методи боротьби з ними.

Поліморфні віруси, які були розглянуті в останньому розділі, є особливим типом шкідливого програмного забезпечення. Вони мають здатність змінювати свою структуру і код, щоб уникнути виявлення та аналізу антивірусними програмами.

Поліморфні віруси становлять значну загрозу для інформаційної безпеки, оскільки вони можуть обходити захисні заходи та завдавати непоправної шкоди системі.

РОЗДІЛ 2
ДОСЛІДЖЕННЯ МЕТОДІВ ВИЯВЛЕННЯ ТА БЛОКУВАННЯ ПОЛІМОРФНИХ ВІРУСІВ

2.1 Технічні характеристики поліморфних вірусів

Розберемо деякі технічні властивості поліморфних вірусів. На малюнку 1 буде наведено код C++, який створює новий код і зберігає його в системі.

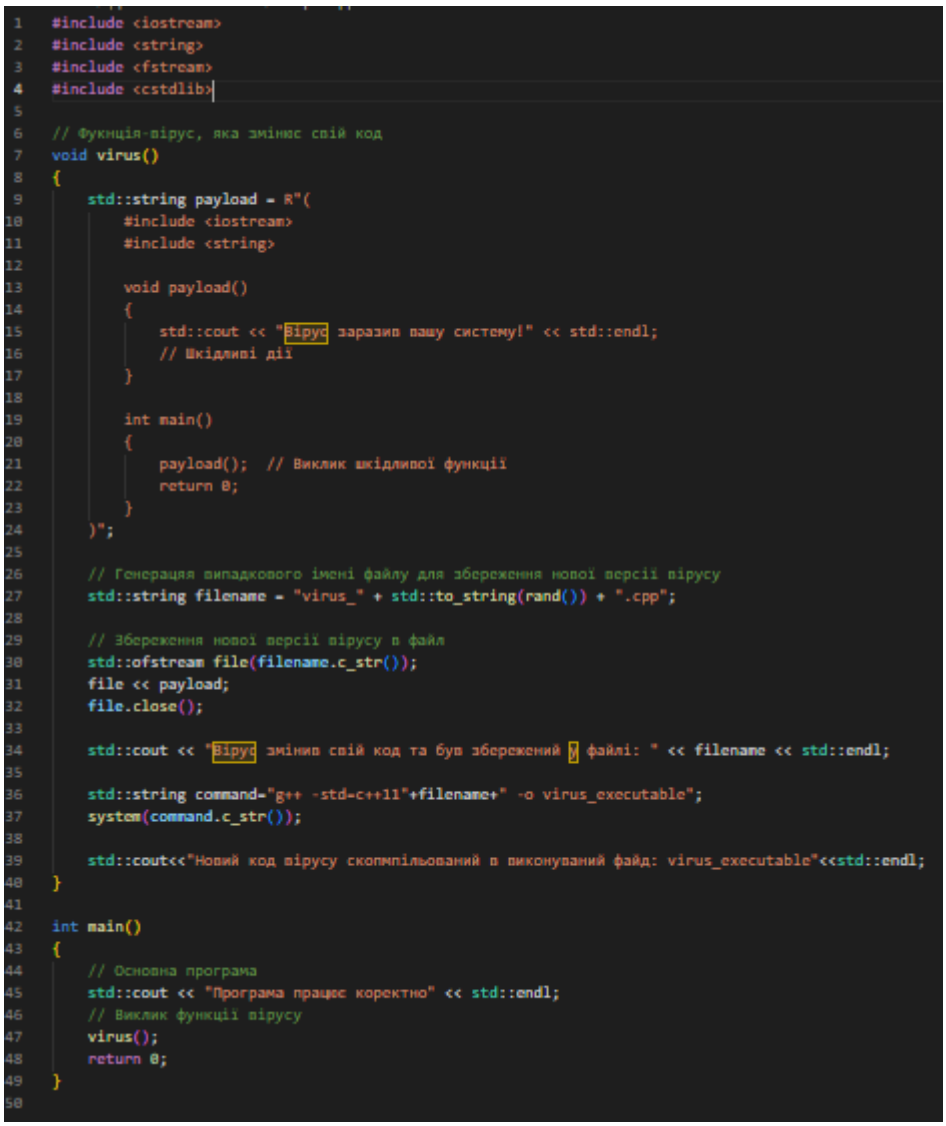
```cpp
#include <iostream>
#include <string>

// Фукнція-вірус, яка змінює свій код
void virus()
{
    std::string payload = R"(
        #include <iostream>
        #include <string>

        void payload()
        {
            std::cout << "Вірус заразив вашу систему!" << std::endl;
            // Шкідливі дії
        }

        int main()
        {
            payload();  // Виклик шкідливої функції
            return 0;
        }
    )";

    // Генерація випадкового імені файлу для збереження нової версії вірусу
    std::string filename = "virus_" + std::to_string(rand()) + ".cpp";

    // Збереження нової версії вірусу в файл
    std::ofstream file(filename);
    file << payload;
    file.close();

    std::cout << "Вірус змінив свій код та був збережений у файлі: " << filename << std::endl;
}

int main()
{
    // Основна програма
    std::cout << "Програма працює коректно" << std::endl;

    // Виклик функції вірусу
    virus();

    return 0;
}
```

Рисунок 2.1 - Код поліморфного вірусу, який змінює та копіює свій код

У цьому прикладі (рис.1) вірус представлений функції virus(). Він створює нову версію свого коду у вигляді рядка payload та зберігає її у новий файл з унікальним ім'ям. Після цього, при виклику функції virus() вірус змінює свій код і зберігає нову версію вірусу в окремий файл.

Зверніть увагу, що в реальній ситуації поліморфні віруси можуть використовувати складніші методи для зміни свого коду та приховування своєї активності. Це лише приклад, щоб проілюструвати концепцію поліморфних вірусів, які змінюють свій код мовою C++.

Цей фрагмент коду просто створює .cpp файл із зазначеним кодом. Для збірки та компіляції цієї програми нам знадобиться GCC. Передбачається, що зловмисник провів ескалацію привілеїв та отримав системний рут. На малюнку 2 ми модифікували наш вірус, щоб він зміг зібрати та скомпілювати свою нову версію.



```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>

// Фукнція-вірус, яка змінює свій код
void virus()
{
    std::string payload = R"(
        #include <iostream>
        #include <string>

        void payload()
        {
            std::cout << "Вірус заразив вашу систему!" << std::endl;
            // Шкідливі дії
        }

        int main()
        {
            payload();  // Виклик шкідливої функції
            return 0;
        }
    )";

    // Генерація випадкового імені файлу для збереження нової версії вірусу
    std::string filename = "virus_" + std::to_string(rand()) + ".cpp";

    // Збереження нової версії вірусу в файл
    std::ofstream file(filename.c_str());
    file << payload;
    file.close();

    std::cout << "Вірус змінив свій код та був збережений у файлі: " << filename << std::endl;

    std::string command="g++ -std=c++11"+filename+" -o virus_executable";
    system(command.c_str());

    std::cout<<"Новий код вірусу скомпільований в виконуваний файд: virus_executable"<<std::endl;
}

int main()
{
    // Основна програма
    std::cout << "Програма працює коректно" << std::endl;
    // Виклик функції вірусу
    virus();
    return 0;
}
```

Рисунок 2.2 - Модифікація нашого вірсу. Тепер він може компілювати створені версії.

На малюнку 3 показано результат роботи шкідливого коду. Спочатку було скомпільовано і запущено файл .poly, який створив файл .virus_executable. Під час запуску файлу виконується шкідливий код, який заражає систему.

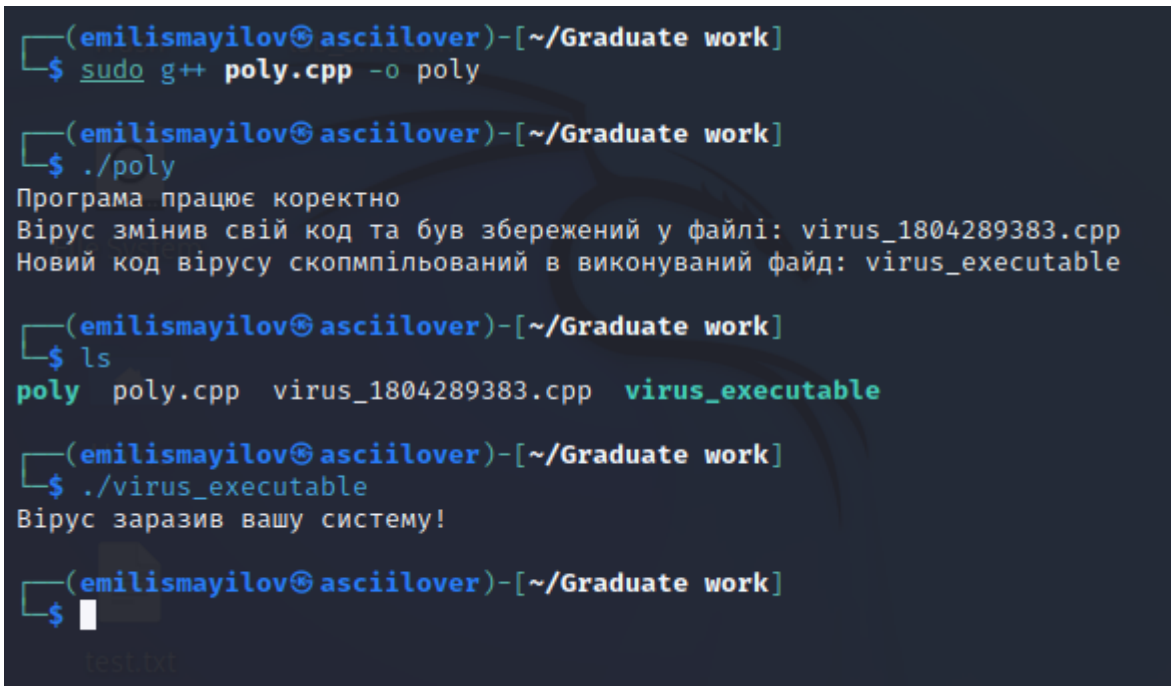


Рисунок 2.3 - Результат роботи шкідливого програмного забезпечення

При бажанні можна видалити створені .cpp файли, щоб приховати сам факт зараження системи та код ШПЗ.

2.2 Статичний аналіз поліморфного вірусу TrickBot

Давайте проведемо статичний аналіз шкідливого програмного забезпечення TrickBot.

TrickBot - ще один поліморфний троян, спочатку помічений у 2016 році, який також залишався активним у 2019-2021 роках. Він використовується для збору банківських даних, вимагання та інших кіберзлочинних дій. Вірус постійно еволюціонує, змінюючи свою структуру та методи атаки.

TrickBot є одним із найбільш поширених поліморфних троянських програм, створених для фінансових шахрайств та крадіжки конфіденційних даних. Цей шкідливий програмний інструмент був виявлений у 2016 році і продовжує активно еволюціонувати, щоб уникати виявлення та протидії.

Ось деякі ключові особливості та характеристики TrickBot:

Розповсюдження: TrickBot зазвичай розповсюджується через спамові електронні листи, що містять шкідливі вкладення або посилання на шкідливі веб-сторінки. Він також може використовувати атаки, експлойти та інші методи для зараження комп'ютерів.

Поліморфізм: TrickBot використовує техніки поліморфізму, щоб змінювати свою структуру та код, роблячи його більш складним для виявлення та аналізу. Кожна нова версія вірусу може мати різні характеристики, шифрування та механізми захисту.

Банківські функції: Однією з головних цілей TrickBot є крадіжка фінансових даних. Він здатний перехоплювати дані авторизації банківських облікових записів, включаючи логіни, паролі та іншу конфіденційну інформацію.

Мережеві атаки: TrickBot також може використовувати компрометовані комп'ютери для атак на інші системи в локальній мережі. Це дозволяє йому поширюватися та інфікувати додаткові пристрої всередині мережі.

Анти-аналіз та захист: У TrickBot застосовуються різні методи захисту, щоб ускладнити його виявлення та аналіз. Це може включати перевірку наявності антивірусних програм, антиналагоджувальні механізми, шифрування коду та інші прийоми.

Модульність: TrickBot побудований з використанням модульної архітектури, що дозволяє йому завантажувати та виконувати додаткові модулі залежно від конкретних потреб атакуючих. Це може включати модулі для перехоплення даних, розповсюдження, збирання інформації та інші функції.

Щоб отримати додаткову інформацію та індикатори компрометації, проведемо хешування (мал. 4) файлу і по отриманому хешу знайдемо вірус на VirusTotal:

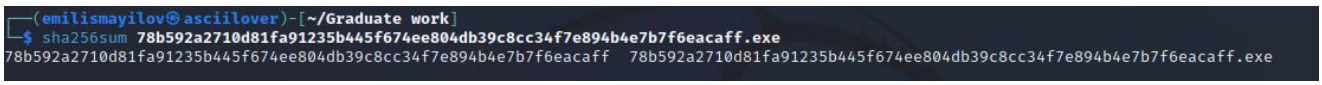


Рисунок 2.4 - Отримуємо хеш-суму шкідливого файлу

При пошуку по отриманому хешу на платформі VirusTotal знаходимо властивості даного файлу (мал. 5), які в подальшому можна використовувати як індикатори компрометації(хеші заголовків) при проведені сигнатурного аналізу.

Basic properties ⓘ

MD5	fa8117afd2dbd20513522f2f8e991262
SHA-1	f7b876edb8fc0c83fd8b665d3c5a1050d4396302
SHA-256	78b592a2710d81fa91235b445f674ee804db39c8cc34f7e894b4e7b7f6eacaff
Vhash	015046651d751bz6lz
Authentihash	6a59fe826c947d09cca87e574935d1cd2981162a3f99fa18b5a17b167508084f
Imphash	95c9dbd11f21d2c0fa6c3dccccbdebb5
Rich PE header hash	d593caf423071cf5010c4e38e2ffff78
SSDEEP	3072:KW5yc3Y4SMQwuOekD96R928AN+/uSxo+HHz/bs/k4OS:K83Y5BAxa92KrxTnz/Y/k4O
TLSH	T17CC32371F02A1FE4F694A0790842A202DD0664C723E6EE1F872F4AB85CDEAF755740F9
File type	Win32 EXE
Magic	PE32 executable (GUI) Intel 80386, for MS Windows
TrID	Win32 Dynamic Link Library (generic) (27.1%) | Win16 NE executable (generic) (20.8%) | Win32 Executable (generic) (18.6%) | Windows Icons Library (generic) (8.5%) | OS/2 Executable (generic) (8.3%)
DetectItEasy	PE32 | Linker: Microsoft Linker (14.24, Visual Studio 2019 16.4*) [GUI32]
File size	119.50 KB (122368 bytes)

Рисунок 2.5 - Результат пошуку по отриманому хешу шкідливого файлу

Також можемо бачити, що для лінкування даного шкідливого файлу використовувався компілятор Visual Studio 2019 16.4.

Виконаємо команду реадпэ для пошуку функцій імпорту в данному файлі (мал.6).

```
Imported functions
    Library
        Name:                       KERNEL32.dll
        Functions
            Function
                Hint:                   470
                Name:                   GetCommandLineA
            Function
                Hint:                   350
                Name:                   ExitProcess
            Function
                Hint:                   961
                Name:                   LoadLibraryA
            Function
                Hint:                   686
                Name:                   GetProcAddress
            Function
                Hint:                   536
                Name:                   GetCurrentProcessId
            Function
                Hint:                   629
                Name:                   GetModuleHandleA
```

Рисунок 2.6 - Функції імпорту

Як було виявлено всі функції є функціями системної бібліотеки kernel32.dll.

Цікавими для нас є такі функції:

- GetModuleHandle;
- GetCommandLine;
- LoadLibraryA;

Функція GetCommandLine є функцією операційної системи Windows, яка повертає повний командний рядок, який використовується для запуску поточного процесу. У контексті TrickBot ця функція може бути використана для отримання інформації про параметри командного рядка, передані під час запуску програми.

Ось деякі способи, як TrickBot може використовувати функцію GetCommandLine:

- Виявлення антивірусних програм та інструментів: TrickBot може проаналізувати командний рядок, отриманий за допомогою GetCommandLine, щоб перевірити, чи встановлені антивірусні програми чи інші інструменти безпеки, які можуть перешкодити його діяльності. Це може допомогти TrickBot прийняти рішення про те, які заходи безпеки слід вживати, щоб уникнути виявлення та нейтралізації.
- Персоналізація атак: TrickBot може використовувати інформацію з командного рядка для персоналізації атак. Наприклад, якщо TrickBot виявляє певний параметр командного рядка, він може змінити свою поведінку або вибрати певні модулі для виконання. Це дозволяє атакуючим налаштовувати та адаптувати TrickBot під конкретні ситуації та цілі.
- Приховування своєї діяльності: TrickBot може використовувати інформацію з командного рядка, щоб вжити заходів щодо приховання своєї активності та уникнути виявлення. Наприклад, може змінювати свою поведінку, якщо виявляє параметри, пов'язані з моніторингом чи аналізом системи. Таким чином, TrickBot прагне залишатися невидимим та уникати детектування.
- Аналіз оточення: Інформація командного рядка може допомогти TrickBot зібрати дані про оточення, в якому він виконується. Це може включати інформацію про систему, облікові записи користувача, шляхи до файлів та інші параметри. Такі

відомості можуть бути використані для вибору конкретних дій або визначення найбільш відповідних атакуючих методів.

Стратегії використання функції GetCommandLine TrickBot можуть змінюватись в залежності від його конкретних цілей та версій. Це лише деякі можливі приклади, і конкретна реалізація та використання можуть змінюватися з часом у зв'язку з еволюцією TrickBot та появою нових версій цього шкідливого програмного забезпечення.

TrickBot може використовувати різні методи та функції для забезпечення своєї модульності. Одним з таких методів може бути використання системної функції GetModuleHandleA.

GetModuleHandleA є функцією операційної системи Windows, яка дозволяє отримати дескриптор модуля (DLL) на його ім'я. Ця функція може бути використана TrickBot для завантаження та виконання додаткових модулів під час своєї роботи.

Зазвичай TrickBot завантажує свої модулі динамічно під час виконання. Він може використовувати GetModuleHandleA, щоб отримати дескриптор вже завантаженого модуля або завантажити новий модуль за допомогою таких функцій, як LoadLibrary або LoadLibraryEx.

Використання функції GetModuleHandleA дозволяє TrickBot динамічно завантажувати додатковий функціонал та модулі в міру необхідності, роблячи його більш гнучким та модульним у своїх атаках та діях.

Однак варто відзначити, що TrickBot може використовувати й інші функції та методи для забезпечення своєї модульності, і конкретні деталі його реалізації та функціоналу можуть змінюватись з часом.

При проведенні сканування програмою PEID (мал. 7) отримуємо результат *Nothing found*, що може вказувати на дві речі:

- Файл не був запакований і обфускований.

- Файл використовує невідомі для PEID методи шифрування чи обфускації, чи методи упаковки.

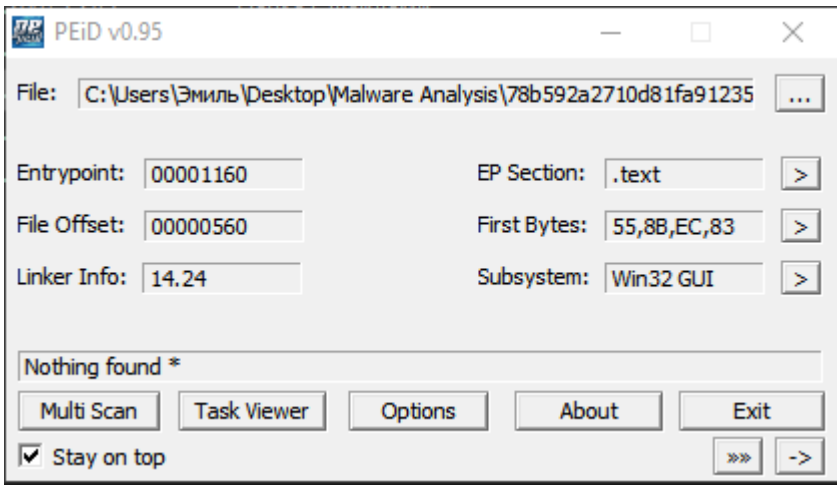


Рисунок 2.7 - Результат сканування PEID

Якщо при скануванні зразка TrickBot у PEID отримано результат "Nothing found", це може вказувати на те, що дана версія TrickBot не має точних сигнатур або ідентифікаційних маркерів, які PEID використовує для виявлення вірусів або шкідливих програм.

Можливі пояснення для відсутності виявлення у PEID можуть бути такими:

- Нова або змінена версія: TrickBot постійно розвивається та оновлюється, щоб уникнути виявлення та нейтралізації. Якщо у вас є дуже нова версія TrickBot або версія, яка була змінена, її сигнатури можуть не бути відомими або застарілими для PEID.
- Поліморфізм: TrickBot може використовувати методи поліморфізму, які дозволяють йому змінювати свою структуру, код або сигнатури з кожним новим екземпляром. Це ускладнює виявлення за допомогою статичних сигнатур, оскільки кожен зразок може мати унікальні характеристики.
- Обфускація та шифрування: TrickBot може застосовувати техніки обфускації та шифрування для утруднення виявлення статичними інструментами, включаючи PEID. Ці техніки змінюють структуру та вміст вірусу таким чином, що статичні сигнатури стають менш ефективними.
- Використання альтернативних пакерів: TrickBot може бути упакований з використанням альтернативних пакерів або протекторів, які змінюють структуру та виконуваний код, що ускладнює виявлення сигнатур.

Напишемо поки що просте yara-правило для детектування шкідливого файлу за його хеш-сумою.

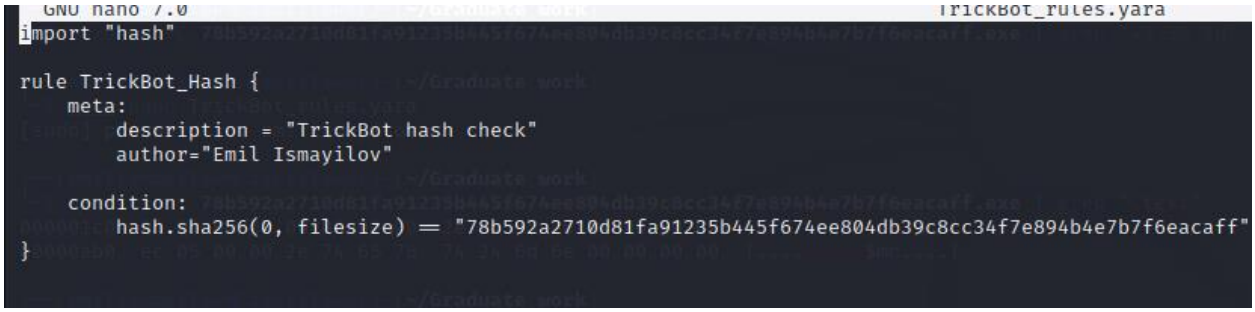


Рисунок 2.8 - Yara-правило для виявлення по хеш-сумі файлу

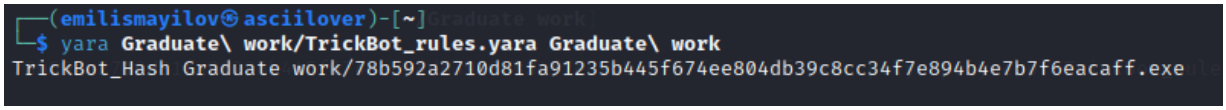


Рисунок 2.9 - Успішне детектування

Використання хеш-суми файлу для виявлення TrickBot має свої переваги та обмеження. Розглянемо їх докладніше:

1. Переваги:

- Унікальність: Хеш-сума є унікальним значенням, яке розраховується на основі вмісту файлу. Якщо хеш-сума файлу TrickBot відома і попередньо обчислена, порівняння цієї хеш-суми з хеш-сумою інших файлів може швидко визначити, чи є файл TrickBot чи ні.

- Цілісність: Хеш-сума дозволяє перевірити цілісність файлу. Якщо вміст файлу змінено, його хеш-сума також зміниться. При виявленні будь-якої зміни в хеш сумі файлу TrickBot можна припустити, що файл був скомпрометований.

2. Обмеження:

- Модифікація: Якщо зловмисник змінює вміст файлу TrickBot, додає або змінює його компоненти, то хеш сума файлу також зміниться. У цьому випадку, якщо

очікувана хеш-сума не відповідає фактичній хеш-сумі, то виявлення TrickBot з використанням тільки хеш-суми стає неможливим.

- Варіанти та модифікації: TrickBot може бути модифікований або мати різні варіанти, які можуть мати різні хеш суми. У цьому випадку, якщо хеш-сума відома тільки для однієї конкретної версії TrickBot, вона не буде ефективною для виявлення інших варіантів або модифікацій.

Використання хеш-суми файлу є одним із методів виявлення TrickBot, але не єдиним. Часто ефективність виявлення шкідливих програм підвищується при комбінуванні різних методів, як-от аналіз поведінки, евристичний аналіз, сигнатурний аналіз та інші. Комплексний підхід до виявлення та блокування поліморфних вірусів, включаючи TrickBot, може бути більш ефективним та надійним.

File name	Revil.bin 78b592a2710d81fa91235b445f674ee804db39c8cc34f7e894b4e7b7f6eacaff.exe Revil.bin.exe Revil.pe
File size	119.50 KB (122368 bytes)
Hash	fa8117afd2dbd20513522f2f8e991262 / f7b876edb8fc0c83fd8b665d3c5a1050d4396302/ 78b592a2710d81fa91235b445f674ee804db39c8cc34f7e894b4e7b7f6eacaff
Imports	ExitProcess, GetCommandLineA, GetCurrentProcessId, GetModuleHandleA, GetProcAddress
Exports	None
Time Date Stamp	2020-07-24 13:21:08 UTC
Sections	.text, .rdata, enc, .reloc
Subsystem	Win32 GUI
Resources	IMAGE

Packer	None
Compiler/Language	Visual Studio 2019 16.4
Detection	61 security vendors and 3 sandboxes flagged this file as malicious

2.3 Огляд основних методів віиявлення

Існує кілька методів виявлення поліморфних вірусів. Комбінація цих методів може забезпечити більш надійний та ефективний захист від таких шкідливих програм. Ось деякі з найпоширеніших методів виявлення поліморфних вірусів:

Сигнатурний аналіз: Цей метод заснований на пошуку унікальних сигнатур або характеристик, властивих певним поліморфним вірусам. Антивірусні програми містять базу даних відомих сигнатур вірусів, і вони порівнюють файли та систему з цією базою даних. Якщо виявляється відповідність сигнатурі, файл вважається зараженим. Однак поліморфні віруси можуть змінювати свої сигнатури, що ускладнює їхнє виявлення.

Сигнатурний аналіз є одним із класичних методів виявлення вірусів, включаючи поліморфні віруси. Він заснований на пошуку унікальних сигнатур, або характерних послідовностей байтів, які є у відомих вірусах. Однак, поліморфні віруси можуть змінювати свою структуру та код, щоб уникнути виявлення на основі сигнатур. Ось деякі приклади сигнатурного аналізу поліморфних вірусів:

1. Сигнатури зразків вірусів: Експерти з інформаційної безпеки постійно вивчають та аналізують нові вірусні загрози. Вони створюють сигнатури, що ґрунтуються на унікальних характеристиках відомих поліморфних вірусів. Наприклад, певні послідовності інструкцій у коді або наявність певних рядків або значень у виконуваному файлі можуть бути використані як сигнатурні патерни.

2. Оновлення баз сигнатур: Лабораторії боротьби з вірусами та постачальники антивірусного програмного забезпечення регулярно оновлюють свої бази сигнатур. Це дозволяє виявляти нові віруси та їх варіанти, включаючи поліморфні віруси. Коли новий вірус виявлено та проаналізовано, його сигнатура додається до бази даних, що

дозволяє антивірусному програмному забезпеченню розпізнавати його при скануванні файлів та системи.

3. Виявлення модифікацій сигнатур: Поліморфні віруси можуть змінювати свою структуру або використовувати шифрування, щоб уникнути виявлення на основі сигнатур. У разі антивірусне програмне забезпечення може використовувати евристичні методи виявлення аномальних змін у сигнатурах. Наприклад, якщо вірусна сигнатура частково змінена або замаскована шифруванням, антивірусне програмне забезпечення може застосувати додаткові аналітичні методи для виявлення шкідливої активності.

4. Порівняння сигнатур з хмарними базами даних: Деякі антивірусні програми використовують хмарні бази даних, де сигнатури оновлюються та розповсюджуються в реальному часі. Це дозволяє програмному забезпеченню швидко отримувати актуальні сигнатури та виявляти нові віруси, включаючи поліморфні віруси на основі даних, зібраних від користувачів по всьому світу.

Важливо, що сигнатурний аналіз корисний виявлення відомих вірусів та його варіантів, але може бути менш ефективним проти нових і маловідомих поліморфних вірусів. Тому евристичні методи та інші підходи до виявлення також відіграють важливу роль у боротьбі з поліморфними вірусами.

Евристичний аналіз: Цей метод ґрунтується на виявленні незвичайних або підозрілих паттернів поведінки програми, які можуть вказувати на наявність вірусу. Антивірусні програми, використовуючи евристичний аналіз, аналізують код та поведінку програми, і якщо виявляються підозрілі дії, файл вважається потенційно шкідливим. Цей метод дозволяє виявляти нові, раніше невідомі віруси, включаючи поліморфні.

Евристичний аналіз є важливим методом виявлення поліморфних вірусів, особливо тих, які можуть обходити традиційні методи виявлення з урахуванням сигнатур. Замість пошуку конкретних сигнатур вірусу, евристичний аналіз зосереджується на виявленні підозрілих чи незвичайних паттернів поведінки програми, які можуть свідчити про наявність вірусу.

Приклади евристичного аналізу поліморфних вірусів включають:

1. Аналіз поведінки коду, що виконується: Евристичний аналіз може стежити за поведінкою виконуваного коду і виявляти аномалії, такі як зміна системних файлів, модифікація прав доступу, запис важливих системних реєстрів та інші підозрілі дії. Наприклад, якщо вірус намагається змінити системні файли або додати себе в автозавантаження без згоди користувача, евристичний аналіз може спрацювати та попередити про потенційну загрозу.

2. Виявлення аномальних модифікацій файлів: Поліморфні віруси можуть змінювати свою структуру та код, щоб уникнути виявлення на основі сигнатур. Евристичний аналіз може порівнювати файли до і після виконання, щоб виявити аномальні зміни або додавання підозрілого коду. Наприклад, якщо виконуваний файл після запуску раптово змінює свою структуру або додає код шифрування, це може вказувати на наявність поліморфного вірусу.

3. Виявлення незвичайного звернення до системних викликів: Поліморфні віруси можуть використовувати системні виклики для взаємодії з операційною системою та здійснення своїх шкідливих дій. Евристичний аналіз може аналізувати послідовність та параметри системних викликів, і якщо виявляються незвичайні або неправильні виклики, це може свідчити про наявність поліморфного вірусу. Наприклад, якщо програма, яка зазвичай не вимагає прав адміністратора, намагається здійснити системний виклик, що вимагає підвищених привілеїв, це може бути ознакою шкідливої активності.

4. Аналіз змін всередині пам'яті: Поліморфні віруси можуть змінювати вміст пам'яті або використовувати техніку ін'єкцій коду для свого поширення та виконання шкідливих дій. Евристичний аналіз може контролювати зміни всередині пам'яті та виявляти підозрілі операції, такі як впровадження коду в інші процеси або модифікація важливих системних компонентів.

5. Аналіз шаблонів активності: Евристичний аналіз може шукати шаблони активності, характерні поліморфних вірусів. Наприклад, якщо вірус виявляє файли з певними розширеннями, шифрує їх і потім вимагає викуп, евристичний аналіз може виявити цей шаблон і попередити про потенційну атаку ransomware.

Аналіз поведінки: Цей метод слідкує за поведінкою програми як реального часу. Він аналізує активність програми, виявляючи аномальну чи шкідливу поведінку, таку як зміна системних файлів, створення або зміна властивостей автозавантаження та інші підозрілі дії. Антивірусні програми, що використовують аналіз поведінки, можуть виявляти поліморфні віруси, не ґрунтуючись на сигнатурах або наперед відомих характеристиках.

Методи еволюційних алгоритмів: Ці методи використовують алгоритми машинного навчання для аналізу та виявлення поліморфних вірусів. Вони досліджують структуру та код поліморфних вірусів, визначають їх характеристики та намагаються виявити подібність з іншими відомими вірусами або класифікувати їх як шкідливі.

Статичний та динамічний аналіз: Ці методи включають аналіз коду та виконання шкідливих файлів в ізольованому середовищі. Статичний аналіз досліджує код шкідливого файлу, шукає підозрілі конструкції та функції. Динамічний аналіз запускає файли в контрольованому середовищі та спостерігає їхню поведінку, щоб виявити аномальні чи шкідливі дії.

Статичний та динамічний аналіз є двома основними підходами до вивчення поліморфних вірусів. Вони надають різні методи та інструменти для аналізу шкідливого коду та виявлення його поведінки. Ось докладніше про кожен із цих підходів із конкретними прикладами:

Статичний аналіз:

Статичний аналіз включає вивчення шкідливого коду без активного виконання. Дослідники аналізують структуру, синтаксис та характеристики коду, щоб виявити потенційно шкідливу поведінку. Ось деякі методи статичного аналізу поліморфних вірусів:

- Аналіз коду та декомпіляція: Дослідники можуть аналізувати вихідний код або декомпільований код вірусу для виявлення вразливостей або характеристик, які дозволяють йому змінювати свою структуру та уникати виявлення. Наприклад,

поліморфний вірус може використовувати шифрування або код, що самодифікується, для маскування своєї справжньої функціональності.

- Аналіз сигнатур та шаблонів: Статичний аналіз може містити пошук унікальних сигнатур або шаблонів коду, які характерні для поліморфних вірусів. Наприклад, дослідники можуть шукати певні послідовності інструкцій або патерни, які вказують на використання коду, що самодифікується, або шифрування.

- Аналіз метаданих та ресурсів: Шкідливі програми, включаючи поліморфні віруси, можуть містити додаткові метадані або ресурси, які використовуються для їх функціонування або маскування. Дослідники можуть аналізувати ці дані виявлення особливих показників вірусу. Наприклад, поліморфний вірус може містити кодовані рядки або інформацію про свої варіанти, які можуть бути використані для його ідентифікації.

Динамічний аналіз:

Динамічний аналіз включає активне виконання шкідливого коду в контрольованому середовищі для спостереження за його поведінкою та виявлення його дій. Ось деякі методи динамічного аналізу поліморфних вірусів:

- Використання віртуальних машин або емуляторів: Дослідники можуть запускати вірус в ізольованому віртуальному середовищі або на емуляторі, щоб спостерігати його поведінку. Вони можуть аналізувати системні виклики, мережну активність та зміни файлової системи, щоб виявити шкідливі дії, такі як зміна або видалення файлів, зв'язок з віддаленими серверами або створення прихованих процесів.

- Моніторинг системних ресурсів: Дослідники можуть використовувати інструменти моніторингу, які відстежують зміни у системних ресурсах, таких як файли, реєстр, процеси та мережеві з'єднання. Наприклад, поліморфний вірус може намагатися змінити системні файли чи реєстр, і динамічний аналіз допоможе виявити такі спроби зміни.

- Дебаггінг та трасування коду: Використання налагоджувачів та трасувальних інструментів дозволяє дослідникам стежити за виконанням шкідливого коду та

аналізувати його роботу на рівні інструкцій. Це може допомогти виявити точки входу, алгоритми шифрування або декодування, а також інші коди маніпуляції, характерні для поліморфних вірусів.

Приклади статичного та динамічного аналізу поліморфних вірусів залежатимуть від конкретних інструментів та методів, які використовуються дослідником. Наприклад, статичний аналіз може включати використання декомпіляторів, дизассемблерів та статичних аналізаторів коду, таких як IDA Pro, Ghidra або radare2. Динамічний аналіз може включати використання віртуальних машин, середовищ розробки з відладниками, таких як OllyDbg або WinDbg, а також спеціалізованих інструментів для трасування коду та моніторингу системних ресурсів, таких як Process Monitor або Wireshark.

2.4 Огляд основних методів блокування

Методи блокування поліморфних вірусів включають різні підходи і техніки, які допомагають запобігти і обмежити їх руйнівний вплив на комп'ютерні системи. Нижче наведено методи, які можуть бути використані для ефективної боротьби з поліморфними вірусами:

Антивірусне програмне забезпечення: Встановлення та регулярне оновлення надійного антивірусного програмного забезпечення є першим та важливим кроком у боротьбі з поліморфними вірусами. Антивірусне ПЗ здійснює сканування файлів та системи на наявність вірусів, виявляє та блокує шкідливі програми, включаючи поліморфні віруси. Регулярне оновлення антивірусних баз даних забезпечує розпізнавання нових вірусів та ефективний захист від них.

Білі списки та чорні списки: Створення білих списків (list of allowed programs) та чорних списків (list of blocked programs) може бути ефективним методом блокування поліморфних вірусів. Білий список містить перелік дозволених програм, які можуть виконуватися на системі, а чорний список містить заборонені програми або типи файлів. Поліморфний вірус, що не входить до білого списку, буде автоматично блокуватися системою.

Хмарні служби антивірусного захисту: Багато антивірусних компаній надають хмарні служби, які оновлюються в режимі реального часу з інформацією про нові загрози та поліморфні віруси. Ці служби можуть блокувати відомі поліморфні віруси, перш ніж вони зможуть завдати шкоди системі.

Файрволи: Використання файрволів допомагає блокувати зовнішні з'єднання та контролювати трафік, що проходить через комп'ютерну систему. Файрволи можуть виявляти та блокувати підозрілий мережевий трафік, пов'язаний з поліморфними вірусами, запобігаючи їх поширенню та зв'язку з віддаленими серверами.

Патчі та оновлення: Регулярне оновлення операційних систем, програмного забезпечення та програм є важливим для запобігання вразливості, які можуть бути використані поліморфними вірусами для зараження системи. Встановлення останніх патчів та оновлень допомагає закривати відомі вразливості та підвищує загальний рівень безпеки системи.

Керування правами доступу: Обмеження привілеїв та прав доступу користувачів може знизити ризик зараження поліморфними вірусами. Встановлення принципу найменших привілеїв означає, що користувачі мають ті права, які необхідні виконання своїх завдань, що зменшує можливість зараження системи шкідливими програмами.

Перевірка пошти та веб-фільтрація: Використання антивірусних програм та систем фільтрації допомагає виявляти та блокувати шкідливі вкладення електронної пошти та шкідливі веб-сайти, пов'язані з поліморфними вірусами. Це допомагає запобігти їх потраплянню в систему та захищає користувачів від ненавмисного запуску шкідливих програм.

Навчання користувачів: Навчання користувачів основ безпеки інформації є важливим фактором у запобіганні атак поліморфних вірусів. Користувачі повинні бути ознайомлені з основними правилами безпеки, такими як уникнення відкриття підозрілих електронних вкладень, неперевірених посилань та завантаження файлів з ненадійних джерел.

Оновлення політик безпеки: Регулярне оновлення та вдосконалення політик безпеки в організації допомагає підвищити рівень захисту від поліморфних вірусів.

Це включає розробку суворих паролів, обмеження доступу до ресурсів і мережевих сервісів, контроль пристроїв, що підключаються до системи та інші заходи, що сприяють забезпеченню безпеки системи.

Моніторинг та виявлення інцидентів: Важливо впровадити системи моніторингу та виявлення інцидентів, які дозволяють оперативно виявляти аномальну поведінку, пов'язану з поліморфними вірусами. Це може включати моніторинг мережного трафіку, системних журналів, виявлення незвичайних активностей та підозрілих процесів.

Резервне копіювання даних: Регулярне створення резервних копій даних допомагає відновити систему у разі успішного зараження поліморфним вірусом. У разі атаки можна відновити дані за допомогою резервних копій, мінімізуючи втрати та вплив на роботу системи.

Системи запобігання вторгненням (Intrusion Prevention Systems, IPS): IPS здійснює контроль над мережним трафіком та аналізує його на наявність ознак шкідливої активності. Він може блокувати спроби зараження поліморфними вірусами, блокувати доступ до небезпечних сайтів або відстежувати незвичайну мережну активність.

Системи контролю цілісності файлів: Ці системи перевіряють цілісність файлів та ідентифікують будь-які зміни у виконуваних файлах. Якщо поліморфний вірус змінює код файлу, система контролю цілісності може виявити це та заблокувати виконання файлу.

Стратегія багаторівневого захисту: Комбінування кількох методів захисту на різних рівнях системи, таких як мережевий, серверний, клієнтський та прикладний рівні, допомагає забезпечити комплексний захист від поліморфних вірусів. Кожен рівень повинен бути зміцнений відповідними заходами безпеки та використовувати відповідні інструменти для виявлення та блокування шкідливих програм.

Успішне блокування поліморфних вірусів потребує застосування комплексного підходу та постійного оновлення заходів безпеки. Комбінація цих методів допоможе ефективно захистити операційні системи від шкідливих програм та мінімізувати ризики для інформаційної безпеки.

Висновки за розділом 2

В данном разделе был проведен анализ полиморфных вирусов, и рассмотрены важные аспекты их исследования и противодействия. Технические характеристики полиморфных вирусов были представлены, включая их способность изменять свою структуру и код для избегания обнаружения и блокирования.

Статический анализ полиморфного вируса был описан как метод исследования, позволяющий анализировать код вируса без его выполнения. Такой подход позволяет выявить характеристики и поведение вируса, а также определить его потенциальные уязвимости.

Обзор основных методов выявления полиморфных вирусов предоставил обзор различных подходов, используемых для обнаружения и идентификации этих вредоносных программ. Это включает эвристические методы, сигнатурное сканирование, виртуальные среды и многие другие.

Также был представлен обзор основных методов блокирования полиморфных вирусов. Здесь были рассмотрены различные техники и инструменты, которые позволяют предотвратить заражение и распространение полиморфных вирусов. Это включает использование антивирусных программ, обновление системного ПО, контроль доступа и прочие меры безопасности.

В итоге, изучение технических характеристик полиморфных вирусов, их статический анализ, методы выявления и блокирования позволяют сформировать комплексный подход к борьбе с этим типом вредоносных программ и обеспечить безопасность компьютерных систем.

РОЗДІЛ 3
РОЗРОБКА МЕТОДИКИ ДЛЯ ЄФЕКТИВНОГО ДЕТЕКТУВАННЯ

1.1 Комбінований метод виявлення поліморфних ШПЗ

Комплексне рішення щодо детектування поліморфного шкідливого програмного забезпечення (ВПО) включає використання різних методів та підходів для виявлення та блокування загроз. Розглянемо деякі з них:

Сигнатурний аналіз: Цей метод ґрунтується на пошуку певних сигнатур або патернів, характерних для відомих шкідливих програм. Використовується база даних сигнатур, що містить інформацію про характеристики відомих загроз. Однак поліморфні віруси можуть змінювати свою структуру або код, щоб уникнути виявлення на основі сигнатур.

Аналіз поведінки: Цей метод зосереджено на спостереженні за поведінкою програми чи процесу. Він може включати моніторинг системних дзвінків, мережеву активність, зміни файлової системи та інших дій. Поліморфні віруси можуть показувати мінливу поведінку, що ускладнює виявлення з урахуванням аналізу поведінки.

Евристичний аналіз: Цей підхід ґрунтується на визначенні потенційно шкідливої поведінки програми, ґрунтуючись на наборі правил та евристик. Може використовуватися виявлення підозрілих операцій, незвичайних шаблонів або інших аномалій. Однак, евристичний аналіз може спричинити помилкові спрацьовування або пропустити нові варіанти поліморфних вірусів.

Машинне навчання: Цей підхід використовує алгоритми машинного навчання виявлення шкідливого програмного забезпечення з урахуванням аналізу великих обсягів даних. Моделі машинного навчання можуть виявляти приховані патерни та зв'язки, що допомагає у виявленні поліморфних вірусів. Однак, потрібне широке навчання моделі та постійне оновлення для ефективного виявлення нових варіантів шкідливого ПЗ.

Приклади комплексних рішень включають комбінацію вищевказаних методів та інших технік. Наприклад, антивірусні програми зазвичай використовують сигнатурний аналіз, евристичний аналіз та машинне навчання для виявлення та блокування загроз. Додатково вони можуть використовувати хмарні служби для обміну інформацією про нові загрози та отримання актуальних сигнатур.

Комплексне рішення також може містити регулярні оновлення баз даних сигнатур, евристичних правил і моделей машинного навчання, щоб враховувати появу нових варіантів шкідливого ПЗ. Також важливо забезпечити захист на різних рівнях системи, включаючи периметральний захист, захист на рівні мережі та на рівні кінцевих точок.

Комбінування різних методів та підходів дозволяє підвищити ефективність виявлення поліморфного шкідливого програмного забезпечення та покращити загальну безпеку інформаційної системи.

Визначення процесу, що є поліморфним шкідливим програмним забезпеченням (ВПО), ґрунтується на виявленні певних ознак та аномалій у його поведінці. Ось кілька технічних ознак, які можна використовувати для визначення поліморфного ВПО:

- Зміна хеш-суми або підпису: Поліморфні ВПЗ можуть змінювати свій код або шифруватися, щоб уникнути виявлення. Тому, якщо хеш-сума або цифровий підпис процесу змінюються, це може свідчити про наявність поліморфного ВПО.
- Поліморфні ВПО можуть виявляти аномальну поведінку, таку як незвичайні мережеві активності, запис або зміна файлів, маніпуляції з процесами та системними ресурсами. Виявлення таких дій може свідчити про наявність поліморфного ВПО.
- Зміна файлової системи: Поліморфні ПО можуть змінювати або створювати файли в системі. Тому, якщо процес виявляє активність щодо зміни системних файлів або створення невідомих файлів, це може бути ознакою поліморфного ВПЗ.
- Приховування від антивірусних програм: Поліморфні ВПЗ можуть застосовувати техніки приховування, щоб уникнути виявлення антивірусних

програм. Це може включати маскування своєї поведінки чи зміну своєї структури. Виявлення таких змін може допомогти в ідентифікації поліморфного ВПЗ.

- Аналіз поведінки: При аналізі поведінки процесу можна виявити аномалії, такі як звернення до нестандартних API-функцій, виконання небажаних операцій або відхилення від типового для цього типу поведінки. Це може допомогти ідентифікувати поліморфний ВПЗ.

Ось концепт невеликого скрипту на Python, який перевіряє активність процесів та виводить повідомлення в консоль при виявленні підозрілої активності:

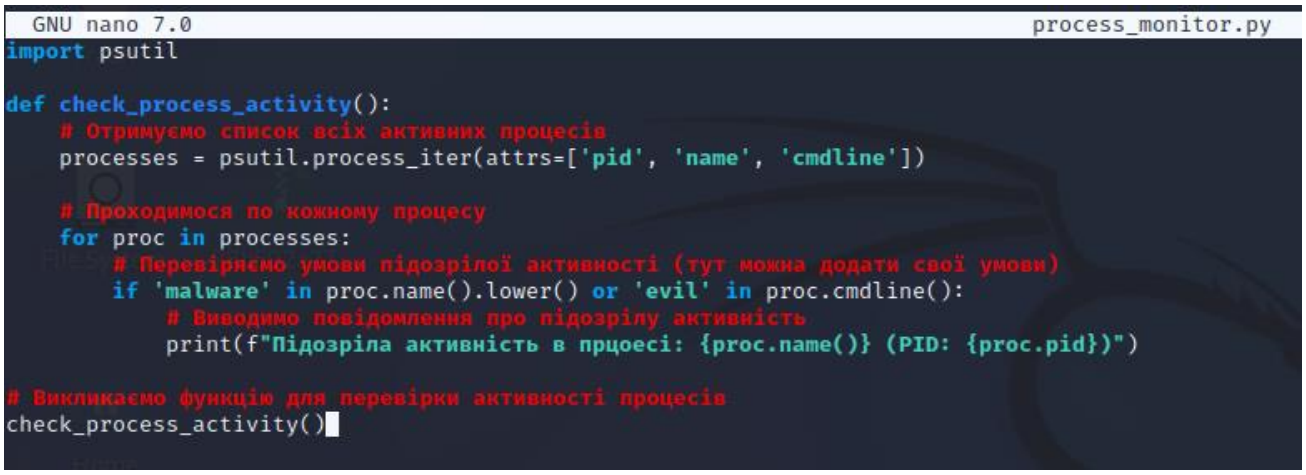


Рисунок 3.1 - Концепт ПЗ для моніторингу активності процесу

У наведеному прикладі використовується бібліотека psutil, яка надає функціональність для роботи з процесами та системними ресурсами. Переконайтеся, що ви встановили цю бібліотеку перед запуском сценарію. Ви можете встановити її, виконавши команду pip install psutil.

Зауважте, що в прикладі використовуються умови перевірки 'malware' in proc.name().lower() і 'evil' in proc.cmdline(). Ви можете адаптувати ці умови під свої вимоги, додаючи або змінюючи критерії визначення підозрілої активності процесу.

Опис методів proc.name().lower() та proc.cmdline():

- proc.name(): Цей метод повертає назву процесу. Наприклад, для процесу "notepad.exe" метод name() поверне рядок "notepad". Застосування методу name() до об'єкта процесу дозволяє отримати ім'я процесу.

- .lower(): Це метод рядка, який наводить всі символи рядка до нижнього регістру. Застосування методу lower() до рядка дозволяє порівнювати вміст без урахування регістру. У контексті скрипту це використовується для порівняння імені процесу без урахування регістру літер.
- proc.cmdline(): Цей метод повертає список аргументів командного рядка, використаних під час запуску процесу. Наприклад, якщо процес був запущений із командним рядком "python script.py --mode=debug", метод cmdline() поверне список ['python', 'script.py', '--mode=debug']. Застосування методу cmdline() до об'єкта процесу (proc) дозволяє отримати перелік аргументів командного рядка, які використовуються процесом.

У скрипті ці методи застосовуються для отримання імені процесу та списку аргументів командного рядка, які потім використовуються для перевірки наявності підозрілої активності. Метод name() використовується для перевірки імені процесу наявність ключових слів, пов'язаних з підозрілою активністю, за допомогою оператора in. Спосіб cmdline() використовується для перевірки списку аргументів командного рядка на наявність ключових слів. Ці перевірки можуть бути налаштовані під конкретні вимоги та критерії виявлення поліморфних вірусів.

Сигнатурний аналіз, також відомий як сигнатурне виявлення, є одним із методів виявлення шкідливого програмного забезпечення (ВПЗ) або конкретних типів атак. Він ґрунтується на використанні попередньо визначених сигнатур або шаблонів, які є унікальними характеристиками або відбитками ВПО.

У процесі сигнатурного аналізу система сканує файли, процеси чи мережевий трафік на наявність певних сигнатур. Сигнатури можуть бути представлені у вигляді послідовностей байт, хеш-сум, шаблонів регулярних виразів та інших форматів, які відповідають відомим характеристикам або поведінці шкідливих програм.

При порівнянні файлів або даних із сигнатурами система шукає точні збіги або часткові збіги. Якщо знайдено збіг, це може вказувати на наявність шкідливої програми або атаки.

Сигнатурний аналіз є одним із найпоширеніших методів виявлення ВПО, оскільки він відносно простий у реалізації та ефективний для виявлення відомих

шкідливих програм. Однак він має деякі обмеження, оскільки не здатний виявити нові або змінені шкідливі програми, для яких ще не створено відповідних сигнатур.

Важливо постійно оновлювати бази сигнатур, щоб вони містили інформацію про нові загрози. Крім того, сигнатурний аналіз зазвичай використовується в поєднанні з іншими методами виявлення ВПЗ, такими як евристичний аналіз, машинне навчання або поведінковий аналіз, для підвищення ефективності та точності виявлення.

Давайте додамо до нашого коду функцію для проведення сигнатурного аналізу коду процесу. Ось доповнений код, який додає функцію check_text_section для перевірки секції .text процесу та виведення повідомлення про підозрілу активність:

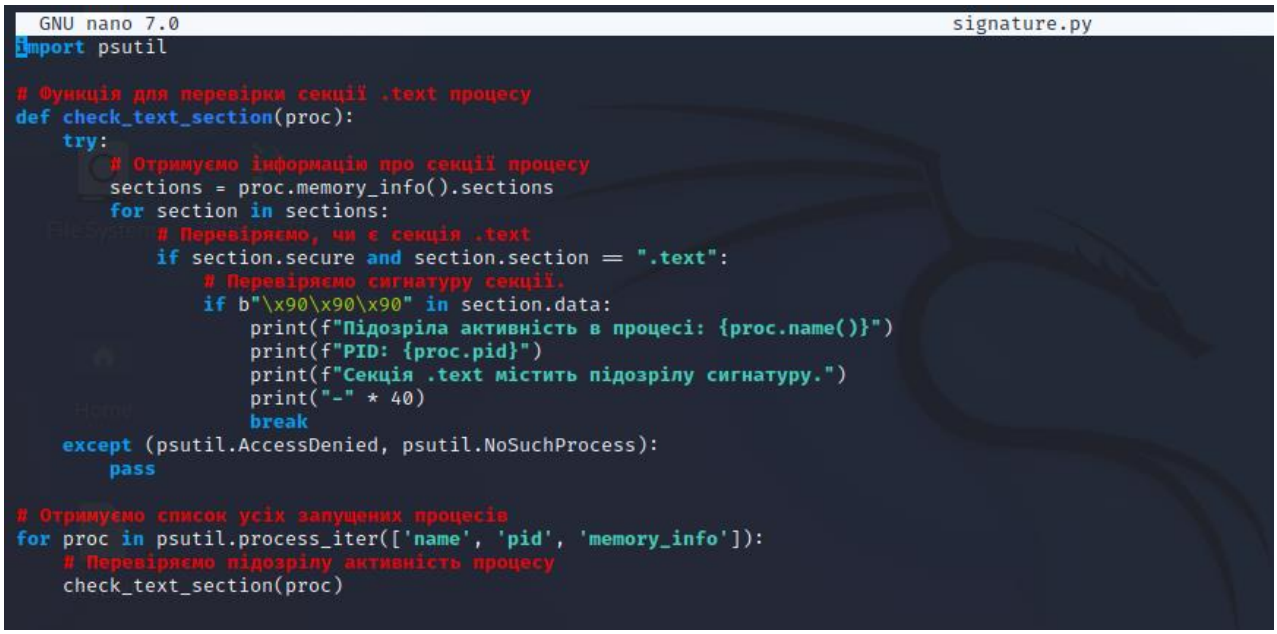


Рисунок 3.2 - Функція check_text_section

У цьому коді додано функцію check_text_section, яка приймає об'єкт процесу (proc) і виконує такі кроки:

- Отримує інформацію про секції процесу за допомогою методу memory_info().sections.
- Перевіряє кожну секцію та шукає .text.
- Перевіряє сигнатуру секції .text (у даному прикладі просто шукає послідовність байт "x90x90x90").

- Якщо виявлено підозрілу сигнатуру, виводить відповідне повідомлення в консоль, вказуючи ім'я процесу, його PID та інформацію про секцію .text.
- Потім, в основній частині коду, ми отримуємо список усіх запущених процесів за допомогою psutil.process_iter() і для кожного процесу викликаємо функцію check_text_section для перевірки підозрілої активності у секції .text.

Метод memory_info().sections використовується для отримання інформації про секції пам'яті процесу.

Коли процес виконується, його пам'ять зазвичай розділена на кілька секцій, таких як .text, .data, .bss, .rdata та інші, які містять різні типи даних та інструкції. Секція .text зазвичай містить код програми, що виконується.

Метод memory_info() повертає об'єкт pmapstruct з інформацією про пам'ять процесу, включаючи список секцій. Властивість sections цього об'єкта являє собою список секцій пам'яті, кожна з яких містить інформацію про свою адресу початку, розмір та інші атрибути.

У коді вище використовувалося властивість memory_info().sections, щоб отримати список секцій пам'яті процесу. Потім ми шукали .text в цьому списку і перевіряли її вміст на наявність заданої сигнатури.

Цей метод корисний для аналізу вмісту пам'яті процесу пошуку підозрілої активності, пов'язаної з конкретними секціями. Він може бути використаний для виявлення модифікованого або шкідливого коду, який може сховатися в .text секції або інших секціях пам'яті процесу.

Однак слід зазначити, що доступ до інформації про пам'ять процесу потребує відповідних дозволів і може бути обмежений на деяких платформах або операційних системах.

Тепер комбінуємо метод сигнатурного та евристичного аналізу. Напишемо скрипт (мал.17), який перевіряє сигнатуру секції .text і поведінку самого процесу (мал.18) (чи не намагається він модифікувати системно-важливі файли на Linux).

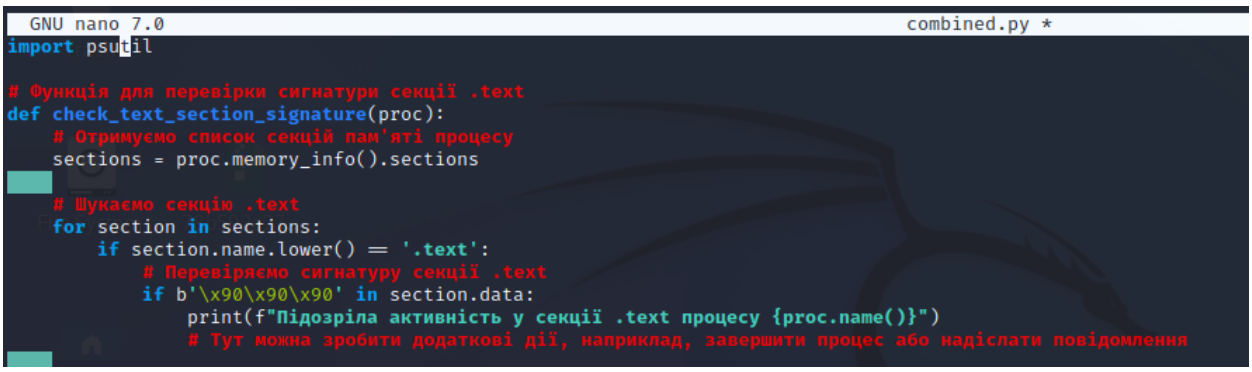


Рисунок 3.3 - Функція аналізу сигнатури

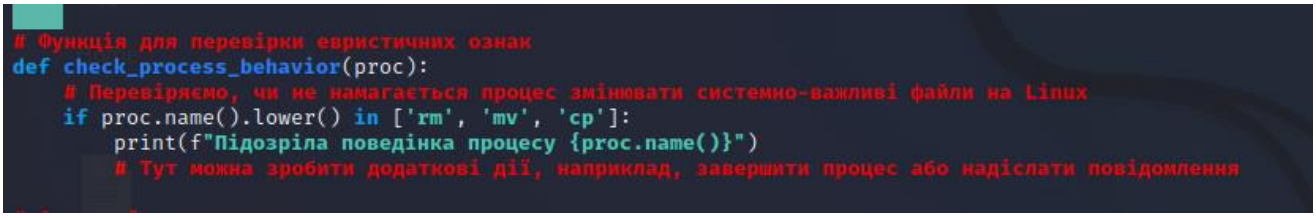


Рисунок 3.4 - Функція аналізу поведінки процеса (евристичний аналіз)

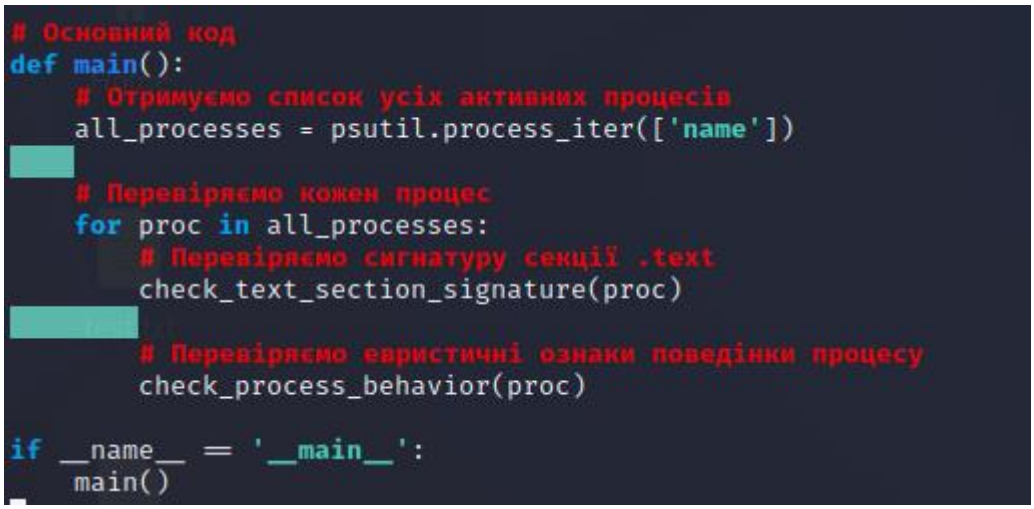


Рисунок 3.5 - Функція main()

У цьому скрипті ми також використовували модуль psutil, який надає функції для отримання інформації про процеси операційної системи. Ми визначили дві функції: check_text_section_signature() для перевірки сигнатури секції .text та check_process_behavior() для перевірки евристичних ознак поведінки процесу.

В основній частині коду ми отримуємо список усіх активних процесів за допомогою psutil.process_iter(). Потім ми застосовуємо наші функції для кожного

процесу і виводимо повідомлення про підозрілі активності, якщо така виявлена.

Цей скрипт можна доопрацювати та розширити, додавши інші перевірки та аналізи, щоб підвищити точність виявлення поліморфного шкідливого програмного забезпечення.

Висновки за розділом 3

Комбіноване рішення щодо детектування поліморфного шкідливого програмного забезпечення є поєднанням різних методів і підходів, таких як сигнатурний аналіз, евристичний аналіз та аналіз поведінки процесів. Це дозволяє виявити та аналізувати різні аспекти шкідливої активності та поведінки програм.

Сигнатурний аналіз ґрунтується на перевірці певних характеристик або сигнатур шкідливих програм. Це може бути сигнатура певних байтових послідовностей, хеш-суми або інші унікальні ознаки. Цей підхід є ефективним при виявленні відомих шкідливих програм, але може не спрацювати для поліморфних варіантів, які можуть змінювати свою сигнатуру.

Евристичний аналіз грунтується на виявленні підозрілих чи нетипових дій чи поведінки програм. Це може містити перевірку активності процесів, спроби зміни системних файлів, зв'язок з підозрілими IP-адресами та інші ознаки, які можуть вказувати на шкідливу активність. Цей підхід дозволяє виявити нові та невідомі шкідливі програми, але може також давати помилкові спрацьовування.

Аналіз поведінки процесів полягає у моніторингу та аналізі дій, які процес виконує в системі. Це може включати зміну файлової системи, мережну активність, взаємодію з системними компонентами і т.д. Шляхом аналізу такої поведінки можна виявити підозрілі чи шкідливі програми.

Комбіноване рішення поєднує ці методи та підходи для підвищення точності та надійності виявлення поліморфних шкідливих програм. При використанні комбінованого підходу можна створити систему, яка враховує різні аспекти шкідливої активності та адаптується до змінних варіантів поліморфного шкідливого програмного забезпечення.

Треба відмітити, що комбіноване рішення не є ідеальним і може мати обмеження. Шкідливі програми можуть постійно еволюціонувати та адаптуватися, щоб уникнути виявлення. Тому важливо постійно оновлювати та вдосконалювати методи виявлення та використовувати інші підходи, такі як машинне навчання та аналіз аномалій, для більш ефективного виявлення поліморфного шкідливого програмного забезпечення.

ВИСНОВКИ

Було розглянуто методи виявлення та блокування поліморфних вірусів в операційних системах. Поліморфні віруси становлять значну загрозу для інформаційної безпеки, оскільки вони здатні змінювати свою структуру та оминати традиційні методи виявлення.

У процесі вивчення даної проблематики було розглянуто різні підходи та методи, які використовуються для ефективного виявлення та блокування поліморфних вірусів. Вони включають евристичний аналіз, сигнатурний аналіз, статичний та динамічний аналіз, а також управління правами доступу.

Евристичний аналіз ґрунтується на виявленні аномальної поведінки програми, що дозволяє виявити поліморфні віруси, які не мають відомих сигнатур. Прикладом може бути аналіз поведінки програми, що змінює безліч файлів у системі або намагається отримати доступ до конфіденційних даних.

Сигнатурний аналіз, у свою чергу, заснований на зіставленні відомих сигнатур вірусів із вмістом файлів чи мережевого трафіку. Прикладом може бути виявлення поліморфного вірусу шляхом порівняння його сигнатури з базою даних антивірусного програмного забезпечення.

Статичний та динамічний аналіз дозволяють вивчити шкідливий код у статичному стані або під час його виконання. Статичний аналіз включає дослідження коду без його фактичного виконання, а динамічний аналіз передбачає запуск коду в контрольованому середовищі та аналіз його поведінки. Ці методи можуть виявити поліморфні віруси за їх характерними ознаками, такими як зміна системних файлів або створення нових процесів.

Управління правами доступу також відіграє у блокуванні поліморфних вірусів. Обмеження привілеїв та встановлення правил доступу дозволяють контролювати можливості шкідливого коду та обмежити його вплив на систему.

На закінчення, методи виявлення та блокування поліморфних вірусів в операційних системах є складним та актуальним завданням у галузі інформаційної

безпеки. Комбінація різних підходів і методів, таких як евристичний аналіз, сигнатурний аналіз, статичний та динамічний аналіз, а також управління правами доступу дозволяє ефективно боротися з цією загрозою. Однак у світлі постійного розвитку вірусних атак і появи нових варіантів поліморфних вірусів важливо постійно оновлювати та покращувати методи захисту, щоб забезпечити надійну безпеку операційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. McAfee Threat Center - https://www.mcafee.com/enterprise/en-us/threat-center.html
2. Symantec Security Response - https://www.symantec.com/security-center/threat-intelligence
3. MalwareTips Forums - https://malwaretips.com/forums/
4. BleepingComputer Forums - https://www.bleepingcomputer.com/forums/
5. National Institute of Standards and Technology (NIST) - https://www.nist.gov/
6. US-CERT (United States Computer Emergency Readiness Team) - https://www.us-cert.gov/
7. European Union Agency for Cybersecurity (ENISA) - https://www.enisa.europa.eu/
8. VirusTotal - https://www.virustotal.com/
9. Trend Micro Security Intelligence - https://www.trendmicro.com/vinfo/us/security/intelligence
10. Microsoft Security Intelligence - https://www.microsoft.com/security/blog/
11. "The Art of Computer Virus Research and Defense" by Peter Szor
12. "Malware Analyst's Cookbook: Tools and Techniques for Fighting Malicious Code" by Michael Hale Ligh, Steven Adair, Blake Hartstein, and Matthew Richard
13. "Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software" by Michael Sikorski and Andrew Honig
14. "The Shellcoder's Handbook: Discovering and Exploiting Security Holes" by Chris Anley, John Heasman, Felix Lindner, and Gerardo Richarte
15. "Reversing: Secrets of Reverse Engineering" by Eldad Eilam
16. "Gray Hat Hacking: The Ethical Hacker's Handbook" by Allen Harper, Daniel Regalado, Ryan Linn, Stephen Sims, Branko Spasojevic, and Linda Martinez
17. "The IDA Pro Book: The Unofficial Guide to the World's Most Popular Disassembler" by Chris Eagle

18. "Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation" by Bruce Dang, Alexandre Gazet, and Elias Bachaalany

19. "Rootkits: Subverting the Windows Kernel" by Greg Hoglund and James Butler

20. "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" by Dafydd Stuttard and Marcus Pinto

ДОДАТОК А

Вихідний код концепції поліморфного ШПЗ:

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>

// Фукнція-вірус, яка змінює свій код
void virus()
{
    std::string payload = R"(
        #include <iostream>
        #include <string>

        void payload()
        {
            std::cout << "Вірус заразив вашу систему!" << std::endl;
            // Шкідливі дії
        }

        int main()
        {
            payload();  // Виклик шкідливої функції
            return 0;
        }
    )";

    // Генерація випадкового імені файлу для збереження нової версії вірусу
    std::string filename = "virus_" + std::to_string(rand()) + ".cpp";

    // Збереження нової версії вірусу в файл
    std::ofstream file(filename.c_str());
    file << payload;
    file.close();

    std::cout << "Вірус змінив свій код та був збережений у файлі: " << filename <<
std::endl;

    std::string command="g++ -std=c++11"+filename+" -o virus_executable";
    system(command.c_str());

    std::cout<<"Новий код вірусу скопмпільований в виконуваний файд:
virus_executable"<<std::endl;
}

int main()
{
    // Основна програма
    std::cout << "Програма працює коректно" << std::endl;
    // Виклик функції вірусу
    virus();
    return 0;
}
```

Вихідний код Yara-правила для детектування ШПЗ TrickBot:

```
import "hash"

rule TrickBot_Hash {
    meta:
        description = "TrickBot hash check"
        author="Emil Ismayilov"

    condition:
        hash.sha256(0, filesize) ==
"78b592a2710d81fa91235b445f674ee804db39c8cc34f7e894b4e7b7f6eacaff"
}
```

Вихідний код Python-скрипту, який перевіряє заголовок процесу:

```
import psutil

def check_process_activity():
    # Отримуємо список всіх активних процесів
    processes = psutil.process_iter(attrs=['pid', 'name', 'cmdline'])

    # Проходимося по кожному процесу
    for proc in processes:
        # Перевіряємо умови підозрілої активності (тут можна додати свої умови)
        if 'malware' in proc.name().lower() or 'evil' in proc.cmdline():
            # Виводимо повідомлення про підозрілу активність
            print(f"Підозріла активність в прцоесі: {proc.name()} (PID: {proc.pid})")

# Викликаємо функцію для перевірки активності процесів
check_process_activity()
```

Вихідний код Python-скрипту, який перевіряє сигнатуру секції .text:

```
import psutil

# Функція для перевірки секції .text процесу
def check_text_section(proc):
    try:
        # Отримуємо інформацію про секції процесу
        sections = proc.memory_info().sections
        for section in sections:
            # Перевіряємо, чи є секція .text
            if section.secure and section.section == ".text":
                # Перевіряємо сигнатуру секції.
                if b"\x90\x90\x90" in section.data:
                    print(f"Підозріла активність в процесі: {proc.name()}")
                    print(f"PID: {proc.pid}")
                    print(f"Секція .text містить підозрілу сигнатуру.")
                    print("-" * 40)
                    break
    except (psutil.AccessDenied, psutil.NoSuchProcess):
        pass

# Отримуємо список усіх запущених процесів
```

```
for proc in psutil.process_iter(['name', 'pid', 'memory_info']):
    # Перевіряємо підозрілу активність процесу
    check_text_section(proc)
```

Вихідний код Python-скрипту, який перевіряє сигнатуру та поведінку процесу:

```
import psutil

# Функція для перевірки сигнатури секції .text
def check_text_section_signature(proc):
    # Отримуємо список секцій пам'яті процесу
    sections = proc.memory_info().sections

    # Шукаємо секцію .text
    for section in sections:
        if section.name.lower() == '.text':
            # Перевіряємо сигнатуру секції .text
            if b'\x90\x90\x90' in section.data:
                print(f"Підозріла активність у секції .text процесу {proc.name()}")
                # Тут можна зробити додаткові дії, наприклад, завершити процес або
надіслати повідомлення

# Функція для перевірки евристичних ознак
def check_process_behavior(proc):
    # Перевіряємо, чи не намагається процес змінювати системно-важливі файли на Linux
    if proc.name().lower() in ['rm', 'mv', 'cp']:
        print(f"Підозріла поведінка процесу {proc.name()}")
        # Тут можна зробити додаткові дії, наприклад, завершити процес або надіслати
повідомлення

# Основний код
def main():
    # Отримуємо список усіх активних процесів
    all_processes = psutil.process_iter(['name'])

    # Перевіряємо кожен процес
    for proc in all_processes:
        # Перевіряємо сигнатуру секції .text
        check_text_section_signature(proc)

        # Перевіряємо евристичні ознаки поведінки процесу
        check_process_behavior(proc)

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