УДК 004.415.53 DOI: 10.26565/2519-2310-2020-2-02

ОГЛЯД СТАТИЧНИХ МЕТОДІВ АНАЛІЗУ ЗЛОВМИСНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Олексій Прищепа, Олександра Доценко

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
olexiiprish@gmail.com, aleksandra.docenko99@gmail.com

Рецензент: Володимир Хома, д.т.н., проф., Опольський політехнічний Університет, Ополе, Польща
xoma@wp.pl

Поступила: Травень 2020.

Анотація: *У сучасному світі проблема втрат від дій зловмисного програмного забезпечення (в т.ч. звичайного, яке несе в собі ознаки не декларованих функцій) продовжує оставатися вкрай актуальною. Тому створення та модифікація антивірусних рішень захисту і аналізу зловмисного програмного забезпечення (ПЗ) є актуальним, та перспективним напрямком досліджень. Це обумовлено відсутністю існування єдиного, універсального способу який забезпечує стовідсоткове знаходження шкідливого коду. У роботі розглядаються склад та основні компоненти статичного аналізу. Визначено основні способи статичного аналізу, та наведені відповідні приклади майже всіх з них. Зроблено висновок, що основними перевагами статичного аналізу є те, що за допомогою використання відносно простого набору команд та інструментів, можливо виконати аналіз зловмисного програмного забезпечення, та частково зрозуміти, як він працює. Звернено увагу на той факт, що статичний аналіз не дає стовідсоткову впевненість в тому, що досліджене ПЗ є зловмисним. Враховуючи це, для забезпечення більш змістовного аналізу потрібно збирати якомога більше даних про структуру файлу, його можливі функції та ін. Аналіз файлів на можливу присутність зловмисного програмного коду, забезпечується за рахунок використання відповідних програм перегляду їх структури та складу. Більш інформативним способом є аналіз Portable Executable – формату. Він складається з аналізу різних секцій коду полів та ресурсів. Оскільки статичний аналіз не завжди надає потрібний рівень гарантій, то на етапі прийняття остаточного класифікаційного рішення (ПЗ зловмисне або ні) краще використовувати алгоритми машинного навчання. Такий підхід надає змогу обробляти великі масиви даних з більшою точністю, стосовно визначення характеру ПЗ, яке аналізується. Головною метою роботи є аналізі існуючих способів статичного аналізу зловмисного ПЗ, та огляд особливостей їх подальшого розвитку.*

Ключові слова: *аналіз; програмне забезпечення; зловмисне програмне забезпечення; статичний аналіз.*

1 Вступ

Практично кожного дня, у світі хтось створює новий вірус, «троянського коня», або експлойт [1] для будь-якої звичайної програми. Кожного дня мільйони людей та десятки компаній втрачають гроші, бази персональних даних, та іншу важливу інформацію. І при цьому кожного дня спеціалісти з інформаційної безпеки (ІБ) з різних країн, модифікують і створюють відповідні, засоби та системи захисту інформації, складають різні захисні модулі на програмні продукти, доповнюють антивірусні бази новими сигнатурами новоз'явлених зловмисних програм, блокують ресурси, що розповсюджують вірусні модулі, рекламу тощо.

В міру того, як мережеві зловмисники, стають все більш витонченими і впроваджують атаки з використанням нового, складного зловмисного ПЗ, його виявлення у системі та відповідне оперативне реагування на них, має вирішальне значення. Аналіз даних на предмет виявлення зловмисного ПЗ, став обов'язковим навиком для відповідної категорії фахівців з ІБ, в межах їх боротьби зі складним зловмисним ПЗ та цільовими атаками.

Мета даної роботи полягає в аналізі існуючих методів статичного аналізу зловмисного ПЗ, та розгляді особливостей їх подальшого розвитку. В межах роботи розглядається тільки ПЗ, що адаптовано для операційних систем (ОС) сімейства Windows.

2 Основна частина

Зловмисне програмне забезпечення (*або шкідливі програми*) - це збірний термін, що позначає шкідливу програму або код, який може завдати шкоди комп'ютерній системі [1].

В цілому, шкідливе ПЗ створюється для реалізації двох основних цілей. Перша з них зводиться до отримання вигоди від вторгнення до комп'ютера (або мережі) жертви. Напри-

клад, зловмисник пробує отримати доступ до управління комп'ютером, мережею-жертви, або викрадає чутливу (*конфіденційну*) інформацію, здійснюючи, згодом, спроби вимагання грошей. Друга група цілей зводиться до матеріальних благ [2].

Є два основних напрямку аналізу зловмисного ПЗ, це статичний та динамічний.

Статичний аналіз базується на аналізі зловмисної програми без її «запуску», наприклад, аналізу її коду за допомогою евристичних методів [3].

Динамічний аналіз, навпаки, для аналізу поведінки підозрілого ПЗ передбачає запуск та ретельне спостереження за всіма його діями [3].

Кожен з цих шляхів ділиться навпіл. Так наприклад, *статистичний аналіз* поділяється на: - *базовий* та *просунутий* статистичний аналіз. Динамічний аналіз поділяється на : - *базовий* та *просунутий динамічний аналіз*.

Статичний аналіз – це аналіз, програмного коду, структури та його функцій для визначення призначення цієї програми, без необхідності його запуску/активації [3]. Статистичний аналіз складається з наступних способів виявлення зловмисного ПЗ: 1 - багаторазове сканування антивірусом; 2 - визначення типу файлу та звірення знайденого гешу програми; 3 - пошук по строкам; 4 - перевірка інформації у *PE*-заголовку (*PE - Portable Executable*); 5 - аналіз дизасемблерного коду.

2.1 Багаторазове сканування антивірусом

Під скануванням антивірусом полягає процес послідовного аналізу потенційно зловмисного ПЗ декількома антивірусами. Сканування антивірусом це перше, що треба зробити коли з'явилася проблема. Цей спосіб не є ідеальний, бо він, в цілому, покладається на відповідні бази даних з відомими частинами коду, а також виконують зіставлення по образу (*сигнатурі*), щоб ідентифікувати підозрілий код. Проблемою є те, що автори шкідливого ПЗ можуть постійно модифікувати та змінювати код, роблячи їх цим невідомими (*т.ч. безпечними*) для антивірусних сканерів. Крім того практично всі новостворені віруси, чи віруси, які з часу свого створення поки ще не набули широкого розповсюдження, теж залишаються в певному сенсі «невидимками». Причина проста - вони ще не внесені до відповідних баз даних (*сигнатур шкідливого коду*) сканерів (*т.з. реактивність оновлень*) [4].

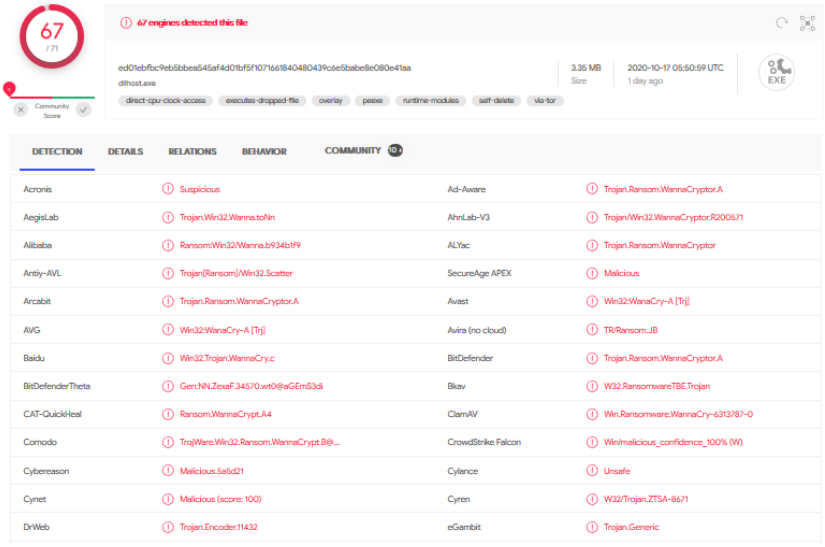

Рис. 1 – Сканування трояну *WannaCryptor*

Для сканування підозрілих файлів можна скористатися інтерактивною Інтернет службою *VirusTotal* [5]. Вона дозволяє завантажити файл, який потім сканується різними антивірусами, з видачею результатів у реальному часі. В якості прикладу на рис.1 наведено результат сканування вірусу. Бачимо, що він був перевірений 71 антивірусом, але тільки 67 змогли його ідентифікувати, як шкідливий.

На рис. 2 показаний результат сканування вірусу *AntiKiez.exe*, якій, на момент його тестування, не був ідентифікований жодним з наявних антивірусних рішень. Тобто, якщо цій вірус потрапить у комп'ютер, то не один з присутніх на ньому антивірусів не зможе зупинити його дію у системі.

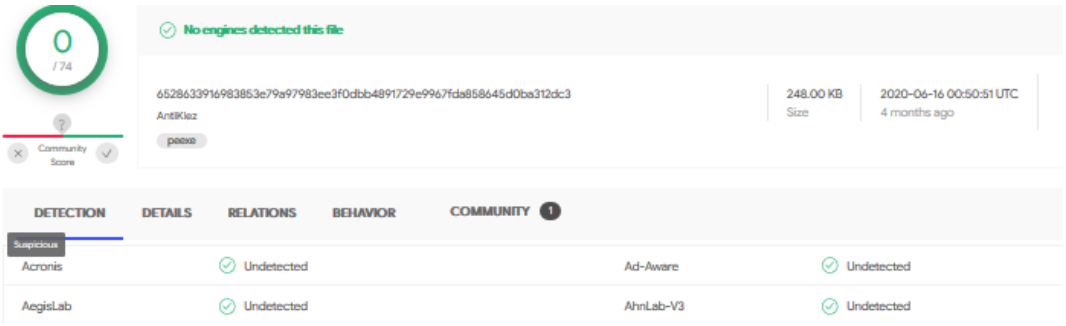

Рис. 2 – Результат сканування вірусу AntiKiez

На рис. 3 наведено результат аналізу файлу getcolor.exe. Ця програма є інструментом для знаходження потрібного кольору, та не несе в собі ніяких шкідливих дій. На логічне питання: - "Чому 3 антивіруси детектували його, як вірус?", можна припустити, що причина полягає в особливостях виконання коду програми.

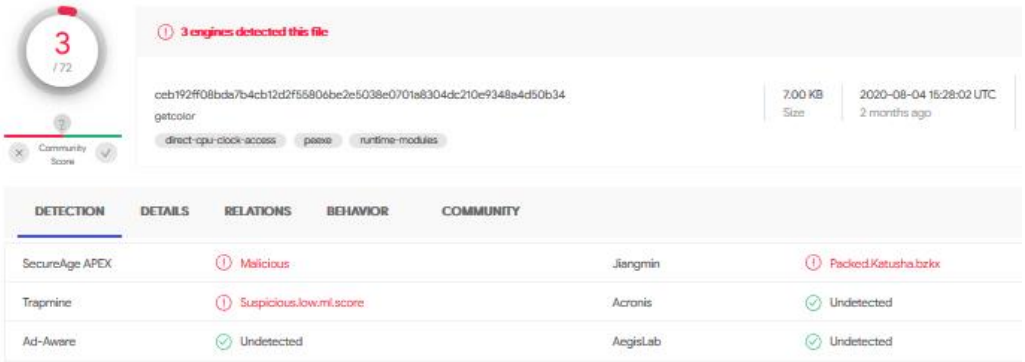

Рис. 3 – Результат сканування програми «getcolor»

2.2 Визначення типу файлу та звірення знайденого гешу програми

При аналізі типу досліджуваного файлу треба ідентифікувати на яку саме операційну систему (ОС), та яку архітектуру процесорів, перш за все націлено дане вірусне ПЗ.

Наприклад, у ОС Windows бінарні файли мають формат PE (*Portable Executable*), де це файли типу **.exe, *.dll, *.sys,* .com та ін. Але тип файлу не є гарантовано вірною інформацією, а для впевненості треба додатково перевірити цифровий підпис. Різні файли мають різні підписи, які використовуються для визначення їх типу. Перевірити цифровий підпис можливо у в hex-редакторі (рис. 4), першими двома байтами є 0x4D, 0x5A, що записуються у ASCII вигляді, як MZ [4]. Другими є сигнатура з двох PE байтів та двох нульових (0x50, 0x45, 0x00, 0x00). Третім значенням є поле Magic, воно вказує на розрядність програми, тобто, чи є файл 32-розрядним (0x010B), чи 64-розрядним (0x020B).

В цьому сенсі, в ОС Linux все декілька складніше, так як кожен файл, у якого встановлений прапор *executable,* є виконуваним.

Гешування - це поширений метод однозначної ідентифікації шкідливого ПЗ. Заражений файл «проходить» через програму хешування, в результаті чого отримується унікальний рядок (хеш), який служить ідентифікатором шкідливого ПЗ.

За цією ознакою в мережі Інтернет можна знайти цей вірус та метод боротьби з ним.

pFile	Raw Data															Value
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	80	00	00	00	
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	i s p
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode
00000080	50	45	00	00	4C	01	02	00	BB	37	56	43	00	00	00	00	PE
00000090	00	00	00	00	E0	00	0F	01	0B	01	02	32	00	0E	00	00	
000000A0	00	0A	00	00	00	00	00	00	00	10	00	00	00	10	00	00	

Рис. 4 – Вивід початку PE файлу

Однією з найбільш поширених функцій хешування, застосовуваних аналітиками з питань інформаційної безпеки (ІБ), є *MD5* (*Message Digest 5*), хоча серія *SHA* (*Secure Hash Algorithm*) теж користується популярністю [3].

В цілому, знайдений хеш можливо використовувати в двох якостях: 1 - як ідентифікатор шкідливого ПЗ; 2 - як позначку, для пошуку у вірусних БД.

Знайти геш програми можливо з використанням відповідних рішень, т.з. геш-шифраторів. Так наприклад, на рис. 5, наведено приклад підрахунку значення гешу за алгоритмами *MD5*, *CRC32* та серії *SHA*. Отримавши відповідні значення можливо перевірити цілісність програми, або перевірити наявність його у вірусних БД.

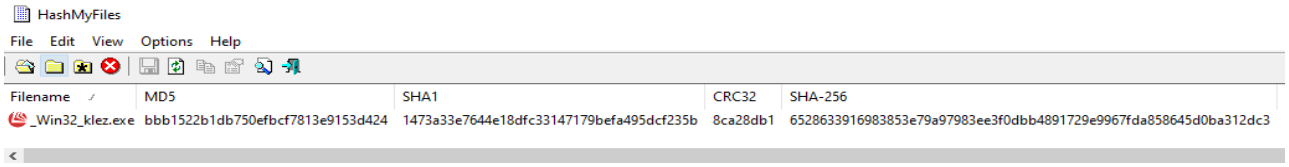

Рис. 5 – Приклад відображення гешів файлу

2.3 Пошук по строкам

Пошук по строкам може надати велику кількість корисної інформації. Наприклад, якщо програма звертається до якогось *URL*, то можливо знайти відповідну адресу, яка зберігається у вигляді строки. Строки у виконавчому файлі зберігаються у форматах *ASCII* або *Unicode*. Для здійснення пошуку, зазвичай, використовують відповідні пошукові програми. Коли така програма шукає строки в форматі *ASCII* або *Unicode*, то вона ігнорує контекст та форматування [3]. Це дозволяє аналізувати файли різних типів та знаходити потрібні строки в любому місці. Недоліком цього способу є те, що іноді знайдена послідовність може бути інтерпретована, як *строка*, хоча це можуть бути і адреса у пам'яті. Тому, в таких випадках, остаточне визначення вердикту лягає на розгляд користувача. З іншого боку, якщо користувачі отримали строки, які схожі з назвами Windows API функцій, то це вже дає додаткову інформацію про здатності програми. Як вже було зазначено вище, строки можуть містити в собі посилання на імена файлів, IP та URL-адреси, домені імена, команди для ініціювання атаки та ключі реєстру. Також, великі строки, котрі складаються з декількох і більше слів, та мають логічне формулювання речення, можуть бути повідомленням [4]. Характерним представником відповідного способу детектування зловмисного ПЗ, є програма *pestudio (*рис. 6*)*.

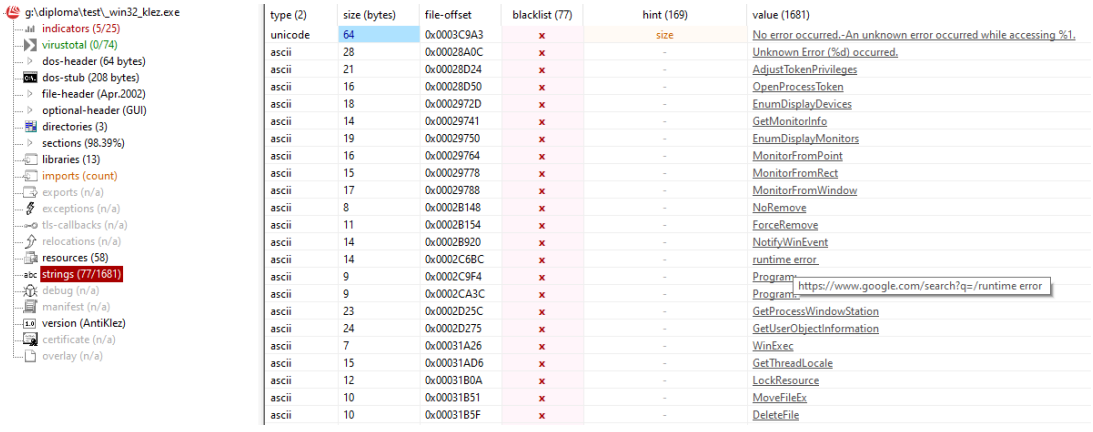

Рис. 6 – Список знайдених строк у файлі

Програма *pestudio* це багатофункціональний інструмент, який може відобразити ASCII та Unicode -строки. В наведеному на рис. 6 прикладі, можна побачити, таблицю з різними строками, де найбільший «інтерес» представляють саме рядки з поміткою «*blacklisted*».

2.4 Перевірка інформації у PE-заголовку

Переносний виконуючий формат (PE, *Portable Executable*) використовується в ОС *Windows* для виконуючих файлів, об'єктного коду та бібліотек динамічної компонування [6]. Формат PE - це структура даних, яка містить інформацію, необхідну системному завантажувачу даної ОС для управління виконавчим кодом. Практично будь-який файл для ОС Windows, в якому є виконавчий код, має формат *PE*, але іноді застарілі формати файлів все ж трапляються у шкідливих програмах.

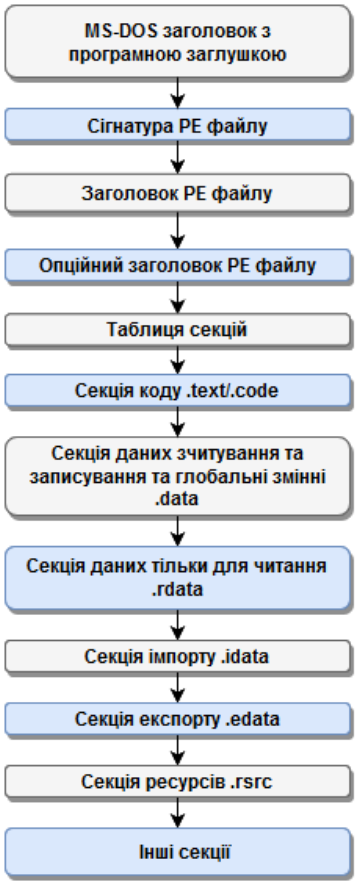

Рис. 7 – Структура образу *PE* файлу

PE-файли починаються з заголовка, який містить інформацію про код, тип додатку, необхідних бібліотечних функціях і вимогах до дисковому простору. Далі починаються секції. Секція, у вигляді коду, містить команди, які будуть виконуватися процесором, а секція, що містить дані, може являти собою різні типи даних (*зчитування та записування даних програми, таблиці імпорту та експорту, різні ресурси*).

Формат *PE*-файлу організований як лінійний потік даних. Він починається з заголовка MS-DOS, програми-заглушки реального режиму і підпису *PE*-файлу. Відразу за ним слідкує *PE* заголовок і необов'язковий заголовок. Крім того, з'являються усі заголовки розділів, за якими слідують всі тіла розділів. Деякі з цих розділів не з’являються у програмі, бо не використовуються. Кінець файлу - це кілька інших областей різної інформації, включаючи: - інформацію про переміщення; - інформацію про таблиці символів; - інформацію про номер рядка і дані таблиці рядків [4]. На рис.7 наведено графічне представлення еталонної моделі *PE*-файлу.

Заголовки зберігають в собі необхідну інформацію для завантаження *PE*-файлу і першим йде DOS-заголовок.

Починати перевірку краще з кінця, бо там більше корисної інформації. Тому почнемо аналіз з файлових залежностей та імпорту. Кожна програма, яка взаємодіє з файлами, реєстром, мережею або якимось пристроєм (*гаджетом*), залежить від функцій, які експортує сама ОС. Ці функції називаються програмним інтерфейсом додатку (API), що необхідні для забезпечення взаємодії у файлах динамічно підключених бібліотек (DLL). Перевірка DLL та API-функції, які використовує шкідлива програма, може надати певні уявлення о її можливостях. Таку перевірку можна здійснити, наприклад, у програмі *pestudio*, вибравши вкладку ***libraries*** (див. Рис.8), де бачимо перелік динамічних бібліотек, які використовують у програмі. Крім того, здійснивши перехід до вкладки ***imports*** можна побачити API-функції, які імпортовані з цих бібліотек.

Слід підкреслити, що у програмі *pestudio* є функція виділення тих функцій, які більш ймовірно вказують на шкідливість програми, тобто на те, що програма є вірусом (див. рис. 9). Цій спосіб є доцільним в тому випадку, коли файл, що аналізується не є запакованим.

Також, на рис. 10 приведено приклад порожньої таблиці імпорту, що є підозрілим, та на це треба звертати увагу.

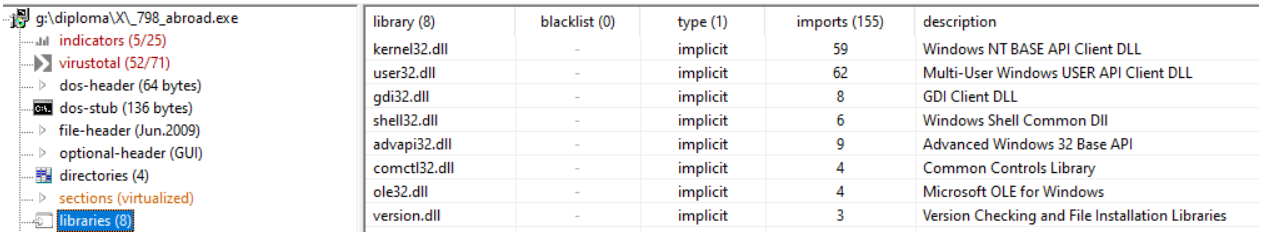

Рис. 8 – Вивід таблиці імпортованих функцій одного вірусу

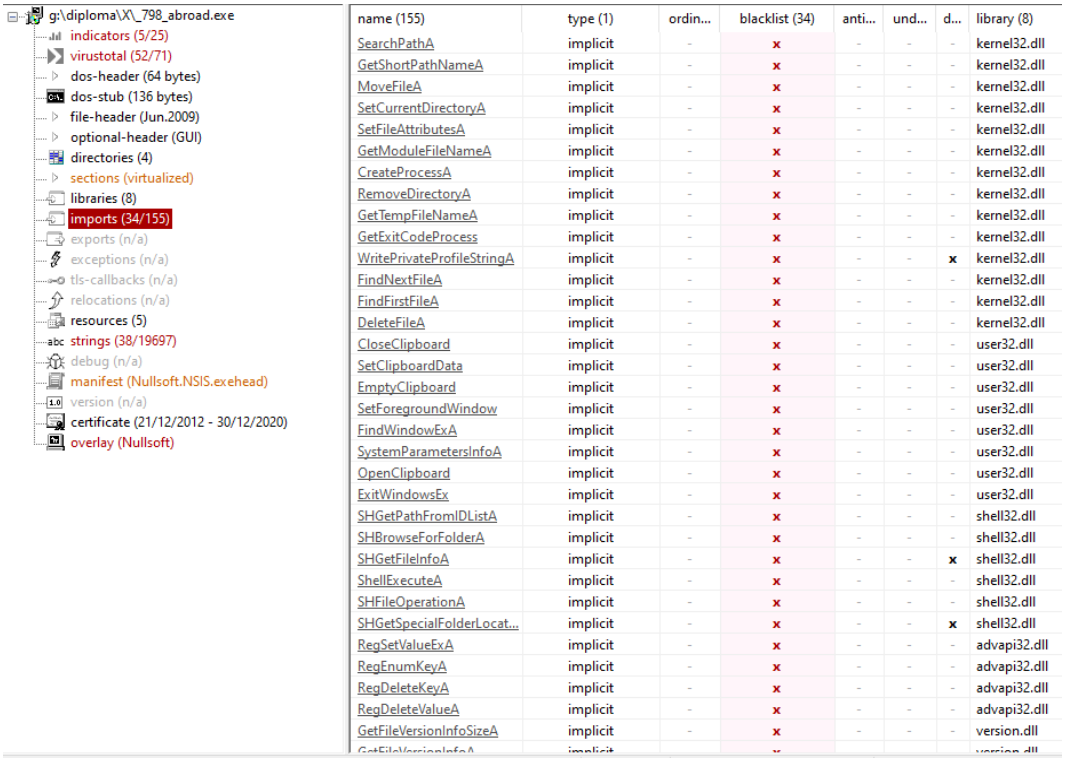

Рис. 9 – Приклад списку функцій, які вважаються підозрілими

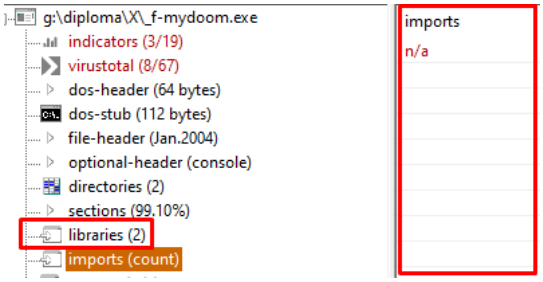

Рис. 10 – Вивід імпортованих функцій з вірусу

Наступним кроком є перевірка експорту. Майже всі випадки з експортованими функціями зустрічаються у *DLL* файлах ніж у файлах типу *EXE*. *DLL* файли здатні експортувати функції для інших програм. Перевірка експортованих функцій може надати уявлення о можливостях *DLL* (*див. список функцій на Рис. 11*). При цьому, варто підкреслити, що імена функцій експорту не завжди дають представлення о можливостях зловмисної програми, бо зловмисник може навмисно використовувати випадкові чи підставні імена, щоб більш ускладнити процес аналізу файлу.

Наступним йде аналіз таблиці секцій *PE*-файлу.

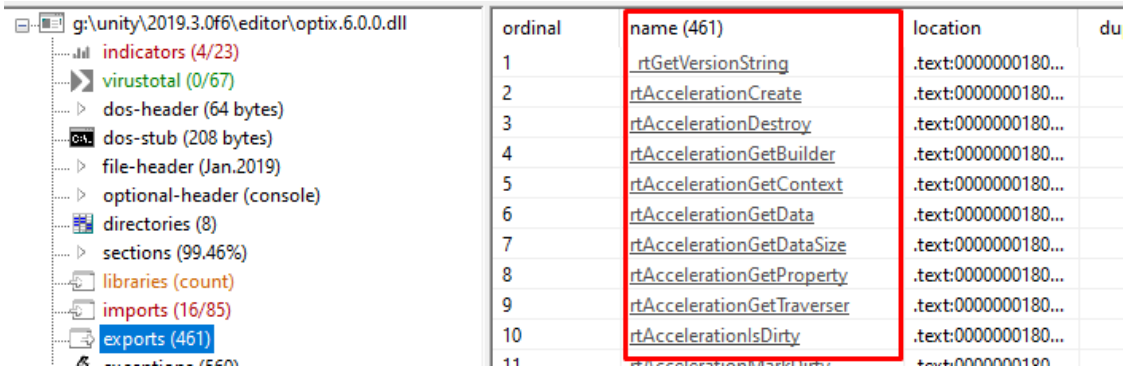

Рис. 11 – Експортовані функції

Зміст *PE*-файлу поділений на секції. Так, у прикладі, що приведено на рис. 12, є дві секції – *UPX0* та *UPX1*, що є назвою пакувальника файлів (*це все є підозрілим, тому це слід мати на увазі*). Далі можна побачити, що перша секція (*перша секція частіше це секція коду*) має характеристику ***writeble*** *(позначено, як **)*, а це значить, що вона може змінюватися. А як відомо, змінювання коду при роботі програми, є однією з характерних можливостей вірусних програм. Також за модифікування файлу відповідає мітка ***self-modifying*** *(позначено, як ***)*.

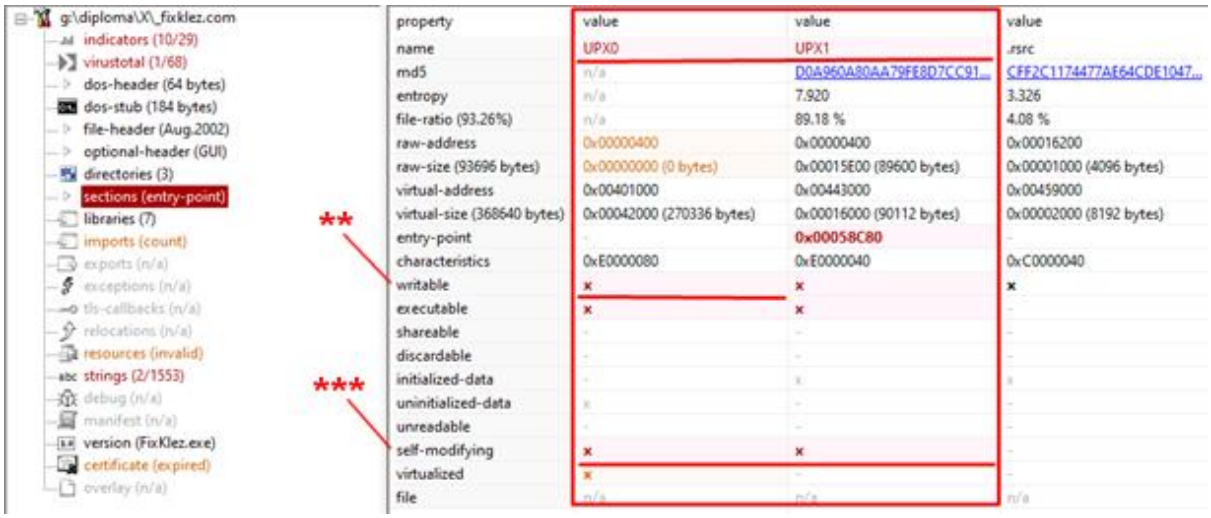

Рис. 12 – Вивід таблиці секцій

Варто зазначити, що при аналізі, також, може допомогти інформація про час компілювання програми. Вона може бути корисною, наприклад, при будуванні графіку атаки, але частіше зловмисники її змінюють. На рис. 13 показана мітка часу з датою компіляції на 1996 рік, яка, в певної мірі, може збити з пантелику, у разі якщо атака пройшла нещодавно.

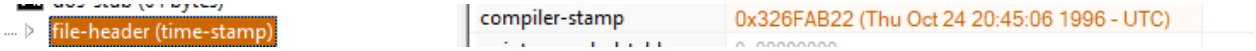

Рис. 13 – Фрагмент інтерфейсу з міткою часу

2.5 Аналіз дизасемблерного коду

Найбільш точним способом зрозуміти чи є програма шкідливою, це зробити її дизасемблювання, але це і найбільш складніший спосіб.

Дизасемблювання - вкрай копітка робота і на глибоку реконструкцію програми розміром в п'ять-десять мегабайт можуть піти багато годин. Значні трудовитрати роблять такий

спосіб аналізу надзвичайно непривабливим і, в якійсь мірі, безперспективним [3, 7]. Так, після проведення дизасемблювання підозрілого ПЗ отримаємо тисячі, чи навіть десятки тисяч строчок коду, що помітно ускладнює прямий аналіз. - Але у цьому випадку треба відштовхуватися від того, що переважна більшість вірусів, «троянів» та інших зловмисних програмних продуктів мають ряд характерних рис – своєрідних *індикаторів*, що відрізняють їх від пересічної, легітимної програми. Однак, надійність таких *індикаторів* істотно нижче і певний відсоток шкідливих програм при цьому залишиться непоміченим [7].

В цілому, кількість всіляких *індикаторів*, які прямо або побічно вказують на зараженість файлу, досить велика, але навіть вони дозволяють виявити до чотирьох з п’яти всіх існуючих вірусів [7].

Також при проведенні дизасемблювання, окрім тривалого оброблення файлу та великої кількості складно аналізованій інформації є ще одна суттєва обставина. А саме: - брати програму та відразу її дизасемблювати може не вийти, бо багато програм є «упакованими». В цьому разі можна отримати на виході безглуздий набір команд, адресів, даних та ін. Тому, спочатку треба «розібратися» з пакувальником, який може бути побудований на десятці команд, або мати і більш складний варіант, наприклад, з використанням поліморфного пакувальника, який генерує зашифровані екземпляри [3, 7].

2.6 *Рекомендації, що до покращення аналізу*

Таким чином, з урахування всього вище зазначеного, можна стверджувати, що для покращення здатності детектувати зловмисне ПЗ, треба максимально широко використовувати всі можливі шляхи аналізу.

Серед інших статичний аналіз слід використовувати в тому випадку, коли немає часу та ресурсів, чи тоді коли в наявності є тільки фрагмент файлу, коду або повідомлення.

Для більш ретельного аналізу сумнівних програм їх треба спочатку повністю розпакувати та розшифрувати. В тому випадку коли це виконавчій файл, необхідно провести його дизасемблювання. Отримавши повністю всю потрібну інформацію можна починати сегментний аналіз та інші методи знаходження шкідливих або не декларованих фрагментів коду.

Для покращення здатності прийняття рішень щодо класифікації програм, треба всіляко впроваджувати алгоритми машинного навчання. По закінченню певного етапу навчання, системи машинного інтелекту здатні автоматично аналізувати великі об’єми даних набагато швидше і точніше ніж це виконує людина.

3 Висновки

1. В роботі проведено стислий огляд відомих способів аналізу зловмисного ПЗ, та наведені основні мітки класифікації ознак.

2. Визначено, що аналіз зловмисного ПЗ ділиться на два види: статичний та динамічний. Статичний аналіз це аналіз зловмисного ПЗ без його завантаження в віртуальну пам’ять. А динамічний аналіз навпаки, реалізує аналіз поведінки вірусу у системі, та його взаємодію з мережею та системою.

3. Звернено увагу на то, що статичний аналіз є першим кроком у аналізі зловмисного ПЗ. Він дозволяє вилучити корисну інформацію, котра отримана з виконуваного файлу, та допомагає надалі у порівнянні і класифікації видів зловмисного ПЗ.

4. Підкреслено, що станом на теперішній час, базовий статичний аналіз зловмисного ПЗ складається з 5-х компонентів: - багаторазового сканування антивірусом; - визначення типу файлу та наступної звірки гешу; - пошуку по строкам; - перевірки відомостей у PE-заголовку; - аналіз дизасемблерного коду.

5. Помітною перевагою використання статичного аналізу для вирішення завдання виявлення зловмисного ПЗ є те, що за допомогою впровадження відносно простого набору команд і інструментів можливо забезпечити аналіз зловмисного ПЗ, та частково усвідомити, як він саме працює. При цьому треба пам’ятати, що статичний аналіз все-таки не дає стовідсот-

кову впевненість в тому, що проаналізована програма є зловмисною. Враховуючи це, для отримання більш об'єктивного результату аналізу, треба збирати якомога більше даних про досліджуваний файл, його структуру і можливі функції, а для прийняття рішення щодо остаточної класифікації програми (*зловмисна або ні*) потрібно використовувати алгоритми машинного навчання. Саме такий підхід надає змогу обробляти великі масиви даних з більш великою точністю, щодо результатів класифікації досліджуваних програм.

Посилання

[1] Вредоносное ПО [Електронний ресурс] / Malwarebytes Режим доступу до ресурсу: https://ru.malwarebytes.com/malware/ (дата звернення 19.05.2020) - Загл. с экрана

[2] Вредоносные программы (malvare) [Електронний ресурс] / Anti-Malware. Режим доступу до ресурсу: https://www.anti-malware.ru/threats/malware (дата звернення 19.05.2020) - Загл. с экрана

[3] Michael S. Practical Malware Analysis: The Hands – On Guide to Dissecting Malicious Software / Michael S., Andrew H.; пер. с англ. Черников С. – Санкт-Питербург : Питер, 2018. - 786 с. (Серия «Для профессионалов»).

[4] Монаппа К. А. Анализ вредоносных программ / пер. с анг. Д. А. Беликова. – М.: ДМК Пресс, 2019. – 452 с.: ил.

[5] VirusTotal Доступ до електронного ресурсу: https://www.virustotal.com/gui/

[6] Peering Inside the PE: A Tour of the Win32 Portable Executable File Format [Електронний ресурс] / Pietrek M. // Microsoft Docs. - 2010 Доступ до електронного ресурсу: https://docs.microsoft.com/en-us/previous-versions/ms809762(v=msdn.10)-?redirectedfrom=MSDN (дата звернення 11.05.2020) - Загл. с экрана

[7] Борьба с вирусами: опыт контртеррористических операций / Касперски К. // Системный администратор. – 2004. – Режим доступу до журн.: http://citforum.ru/security/virus/virii_dis/ (дата звернення 18.05.2020) - Загл. с экрана

Рецензент: Владимир Хома, д.т.н., проф., Опольский Политехнический Университет, Ополе, Польша.
E-mail: xoma@wp.pl

Поступила: Май 2020.

Авторы:

Прищепа Алексей, студент факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: olexiiprish@gmail.com

Доценко Александра, студентка факультета компьютерных наук, ХНУ имени В.Н. Каразина, Харьков, Украина.
E-mail: aleksandra.docenko99@gmail.com

Обзор статических методов анализа зловредного программного обеспечения.

Аннотация: В современном мире проблема потерь от действий вредоносных программ (или обычных, которые несут в себе признаки не декларируемых функций) продолжает быть крайне актуальной. Поэтому создание и модификация антивирусных решений защиты и анализа вредоносных программ (ПО) является актуальным, и перспективным направлением исследований. Это обусловлено отсутствием существования единого, универсального способа который обеспечивает стопроцентное нахождения вредоносного кода. В работе рассматриваются состав и основные компоненты статического анализа. Определены основные способы статического анализа, и приведены соответствующие примеры многих из них. Сделан вывод, что основными преимуществами статического анализа является то, что с помощью использования относительно простого набора команд и инструментов, можно выполнить анализ вредоносных программ, и частично понять, как он работает. Обращено внимание на тот факт, что статический анализ не дает стопроцентную уверенность в том, что исследовано ПО является злонамеренным. Учитывая это, для обеспечения более содержательного анализа нужно собирать как можно больше данных о структуре файла, его возможные функции и др. Анализ файлов на возможное присутствие вредоносных кода, обеспечивается за счет использования соответствующих программ для просмотра их структуры и состава. Более информативным способом является анализ *Portable Executable* - формата. Он состоит из анализа различных секций кода полей и ресурсов. Поскольку статический анализ не всегда предоставляет необходимый уровень гарантий, то на этапе принятия окончательного классификационного решения (злонамеренное ПО или нет) лучше использовать алгоритмы машинного обучения. Такой подход позволит обрабатывать большие массивы данных с большей точностью, по определению характера ПО, анализируется. Главной целью работы является разбор существующих способов статического анализа злонамеренного ПО, и обзор особенностей их дальнейшего развития.

Ключевые слова: Анализ; программное обеспечение; вредоносное программное обеспечение; статические методы анализа.

Reviewer: Volodymyr Khoma, Dr. of Sciences (Eng.), Full Prof., The Opole University of Technology, Opole, Poland.
E-mail: xoma@wp.pl

Received: May 2020.

Authors:

Alex Pryshchepa, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: olexiiprish@gmail.com

Aleksandra Docenko, student of CSD, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: aleksandra.docenko99@gmail.com

Overview of static methods of analysis malicious software.

Annotation: In today's world, the problem of losses from the actions of malicious software (or ordinary software, which has the characteristics of undeclared functions) continues to be extremely relevant. Therefore, the creation and modification of anti-virus solutions for protection and analysis of malware (software) is a relevant and promising area of research. This is due to the lack of a single, universal method that provides 100% finding malicious code. The paper considers the composition and main components of static analysis. The main methods of static analysis is identified, and relevant examples of almost all of them are given. Got concluded that the main advantages of static analysis are that by using a relatively simple set of commands and tools, it is possible to perform malware analysis and partially understand how it works. Attention is drawn to the fact that static analysis does not give 100% certainty that the investigated software is malicious. With this in mind, to provide a more meaningful analysis, you need to collect as much data as possible about the structure of the file, its possible functions, etc. Analysis of files for the possible presence of malicious code is provided through the use of appropriate programs to view their structure and composition. A more informative way is to analyze the Portable Executable format. It consists of the analysis of various sections of the code of fields and resources. Since static analysis does not always provide the required level of guarantees, it is better to use machine learning algorithms at the stage of making the final classification decision (malicious or not). This approach will make it possible to process large data sets with greater accuracy in determining the nature of the software is analyzed. The main purpose of this work is to analyze the existing methods of static malware analysis, and review the features of their further development.

Keywords: Analysis; Software, Malware; Static analysis of malware.