УДК 004.056.53

МЕТОДИ ЗАХИСТУ ВІД ЗМІН ТА ДИЗАССЕМБЛЮВАННЯ ВИКОНАВЧИХ ФАЙЛІВ ОС WINDOWS

Левчук А.С.
Мелітопольський державний педагогічний університет імені Богдана Хмельницького, м. Мелітополь
0xadach@gmail.com

Постановка завдання. На сьогоднішній день Windows є найбільш поширеною операційною системою, яка працює на мільйонах комп'ютерів по всьому світу. Для цієї операційної системи розроблено багатий інструментарій програмного забезпечення, призначений для зворотної розробки, який у більшості випадків дозволяє навіть початківцям вивчати структуру та довільним чином втручатися у роботу програмних засобів (наприклад – вимикати систему захисту від копіювання). Величезна кількість постійно здійснюючих зломів – найкраще тому підтвердження [4]. У зв'язку з цим проблема захисту виконавчих файлів від впливу зловмисників є актуальною.

Метою статті є огляд підходів до аналізу бінарних виконуваних файлів та основних методів протидії цьому.

Виклад основного матеріалу. Під дизасемблюванням розуміється побудова програми на мові асемблера, еквівалентної вихідній програмі в машинному коді. Під декомпіляцією розуміється побудова програми на мові високого рівня, еквівалентної вихідній програмі на мові низького рівня (мові асемблера).

При декомпіляції програма з уявлення низького рівня транслюється в уявлення високого рівня. Подальшим етапом підвищення рівня абстракції програми може бути рефакторинг. Найчастіше дизассемблер використовують для аналізу програми (або її частини), вихідний текст якої невідомий – з метою модифікації, копіювання або злому. Рідше – для пошуку помилок в програмному забезпеченні, а також для аналізу та оптимізації створюваного компілятором машинного коду.

Зараз більшість безкоштовних декомпіляторів доступні в Інтернеті:

1. Boomerang. Програмний засіб має досить продвинутий набір алгоритмів аналізу коду, підтримувані архітектури: IA32 MIPS PPC, платформи: Windows / Linux. Якість коду, що видається, сильно варіюється: деякі функції майже ідеально відновлені, добре видно структуру коду і є вказівка типу змінних. В інших випадках функції сильно заплутані і їх майже неможливо прочитати. Дана програма ще розробляється, але її бета-версія наявна у вільному доступі.

2. RecStudio – використовує продвинутий набір алгоритмів аналізу (partial Single Static Assignment, SAP), хоча до цих пір в стадії розробки.

Програма працює цілком стабільно, є складання під Linux, також як і попередню, її легко можна знайти і скачати.

3. Hex rays є модулем розширення найбільш популярного дизасемблера Ida pro. Завдяки своїй гнучкості дозволяє при наявності достатньої кваліфікації користувача отримати найкращий результат. Його висока вартість і складність покупки для приватної особи компенсується частими витоками ліцензійних копій з великих компаній (рис. 1).

Варто відзначити, що спочатку проста архітектура x86 [2] згодом обросла сотнями нових інструкцій (підтримка чисел з плаваючою комою, SSE, MMX і інші), що значно ускладнює навіть таку теоретично просту операцію, як дизасемблювання. А з завданням декомпіляції більш-менш складних ділянок коду з публічно доступних інструментів на даний справляється тільки модуль дизасемблера Ida pro – Hex-Rays.

Маючи доступ до бінарних кодів програмного забезпечення, зловмисник може змінити виконуваний файл таким чином, щоб змінити поведінку програмного забезпечення на поведінку, відповідну зареєстрованій версії програми. Виходячи з вищевикладеного, можна сказати, що для захисту програмного забезпечення від несанкціонованого дослідження та / або модифікації, необхідно максимально ускладнити статичний та динамічний аналіз програмного засобу.

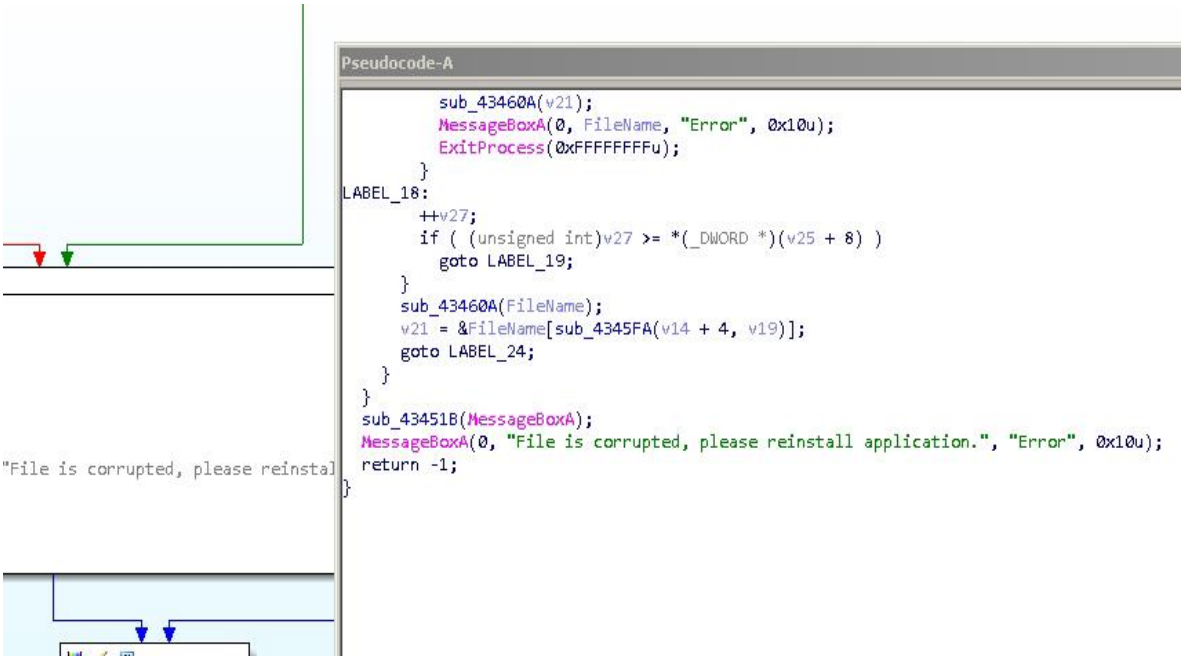

Рис. 1. Результат роботи Hex rays

Вивчення логіки роботи програми може відбуватися двома способами: статичним і динамічним. Сутність статичного методу полягає у вивченні дизасемблювання (декомпіляції) коду. Динамічний метод вивчення алгоритму програми передбачає виконання трасування програми. Під трасуванням програми розуміється виконання програми з використанням

низькорівневого відладчика, що дозволяє виконувати програму в покроковому режимі, отримувати доступ до регістрів, областей пам'яті, робити зупинку виконання програми за певними адресами [3].

Найкращим методом захисту від статичного методу аналізу додатків є упаковка/шифрування виконуваного файлу з розпакуванням під час виконання з контролем цілісності упакованих даних. Так як передача управління на оригінальний код відбувається в пам'яті, це робить неможливим проведення статичного аналізу.

Однак, в процесі динамічного аналізу такий захист може бути повністю знятий дослідником, тобто після розпакування і / або розшифровки вихідний файл знаходиться в пам'яті комп'ютера в «чистому» вигляді [6]. При наявності сучасних засобів налагодження програм, таких як Olly debugger [1] повністю виключити можливість динамічного аналізу програми неможливо, але істотно ускладнити трасування можливо.

Для протидії динамічному аналізу широко використовуються наступні механізми:

Обфускація. Ефективність даного методу досягається за рахунок використання особливостей людського фактора – чим складніше вихідний код і чим більше ресурсів використовує додаток, тим складніше досліднику зрозуміти логіку роботи програми, а, отже, і зламати застосовані засоби захисту. Обфускація (заплутування) коду може бути проведена різними способами, при цьому найбільший ефект захисту досягається при комбінуванні їх один з одним:

1. Вставка сміттєвого коду – вставка випадкового числа інструкцій, які не змінюють поведінку програми, але при цьому вносять неясність в код, збільшуючи тим самим кількість необхідного часу, необхідного досліднику на вивчення алгоритму.

2. Мутація коду – одно - або багаторазові перетворення існуючих інструкцій в інші, що не змінюють поведінку програми.

3. Віртуалізація коду – перетворення коду в код абстрактної віртуальної машини, код якої розташовується в виконуваному файлі, для виключення можливості розуміння вихідного алгоритму в ході аналізу.

Приховування викликів бібліотечних функцій - дозволяє приховати від дослідника список API-функцій, які використовує захищувана програма. Приховування може здійснюватися кількома способами:

1. Приховувані функції відсутні в таблиці імпорту, адреси цих функцій визначаються динамічно вже під час роботи захищеного файлу, а не на етапі роботи стандартного завантажувача Windows.

2. Емуляція деяких API-функцій - перенаправлення прямого виклику API-функції на виклик внутрішньої функції, яка повністю повторює поведінку прихованою функції.

Використання анти відладжувальних методів – спосіб полягає у визначенні, чи запущена програма в середовищі відладчика - якщо так, виконання програми завершується. Найбільш поширені методи:

використання API-функцій (IsDebuggerPresent, CheckRemoteDebugger Present); пошук відладчика в системі (на ім'я процесу, на ім'я або класу вікна відладчика), замір часу виконання, перевірка імені батьківського процесу.

Протидія запису дампу пам'яті дозволяє значно ускладнити створення відбитка пам'яті та його статичний аналіз. Вона полягає у зміні службових полів виконуваного файла; динамічному шифруванні невикористовуваних сторінок пам'яті процесу; переміщенні та емуляції частини коду програмного засобу.

Висновки. Отже, на сьогодні існують різні методи захисту від небажаного втручання у виконувані файли.

Література

1. Eilam E. Reversing: Secrets of Reverse Engineering / E. Eilam. – Wiley Publishing, Inc, 2005. – 118–119 с.
2. x86 : [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/X86.
3. Защита программных средств от исследования. – [Електронний ресурс]. – Режим доступу: http://sumk.ulstu.ru/docs/mszki/Zavgorodnii/8.2.3.html.
4. Касперски К. Техника хакерских атак. Фундаментальные основы хакерства / К. Касперски. – Издательство: СОЛОН-Р, 2005. – 258 с.
5. Касперски К. Искусство дизассемблирования / К. Касперски, Е. Рокко. – БХВ-Петербург, 2008. – 896 с.

Анотація. В роботі розглядається декомпіляція 32-х бітних виконуваних файлів, різні способи захисту, такі як пакування, обфускація, контроль запуску в режимі налагодження. Так само оцінюється ступінь надійності і стійкості зламуванню розглянутих засобів захисту.

Ключові слова: упаковування виконуваних файлів, засоби захисту програмного забезпечення.

Аннотация. В работе рассматривается декомпиляция 32-х битных исполняемых файлов различные способы защиты, такие как упаковка, обфускация, контроль запуска в режиме отладки. Так же оценивается степень надежности и устойчивости ко взлому рассмотренных средств защиты.

Ключевые слова: упаковка исполняемых файлов, способы защиты программного обеспечения.

Summary. The paper deals with a compilation of 32-bit executable files are different ways of protection, such as packing, obfuscation, control run in debug mode. Also assessed the reliability and stability to cracking discussed remedies.

Keywords: executable compression, methods of protecting software.