УДК 681.3.07

В. А. Каплун

ВИКОРИСТАННЯ «ДИНАМІЧНОЇ ФУФЕЛІЗАЦІЇ» ДЛЯ ЗАХИСТУ ВІД СТАТИЧНОГО ДОСЛІДЖЕННЯ

Вінницький національний технічний університет

Анотація

Досліджено способи захисту від несанкціонованого дослідження програм. Особливий акцент зроблено на протидію злому програм за допомогою інтерактивних дизасемблерів. Запропоновано використовувати такий авангардний спосіб захисту, як вбудовування у програму засмічуючого фрагменту у вигляді так званого «динамічного фуфеля».

Ключові слова: захист програмного забезпечення, автоматичний дизасемблер, інтерактивний дизасемблер, статичне дослідження програм.

Abstract

Studied ways to protect against unauthorized research programs. Particular emphasis is placed on opposition hacking programs through interactive disassemblers. An avant-garde use this method of protection as in the embedding program clogging up the fragment in the form of so-called "dynamic Fufel."

Keywords: protection software, automatic disassemblers, interactive disassemblers, static research programs.

Вступ

Відсутність початкових текстів зовсім не є непереборною перешкодою для вивчення і модифікації коду додатка. Методики зворотного проектування дозволяють автоматично розпізнавати бібліотечні функції, локальні змінні, стекові аргументи, типи даних, розгалуження, цикли і т. д. Сьогодні трудомісткість аналізу двійкового коду не настільки велика, щоб надовго зупинити зламників. Скільки б рівнів захисту не було передбачено, один або мільйон, програма може бути зламана – це тільки питання часу і витрачених зусиль.

Внаслідок цього першочерговою задачею зламника при зламі практично будь-якого захисту є дизасемблювання коду програми, що виконується, і одержання лістингу з мнемонічним зображенням асемблерних команд, тобто можливість статичного дослідження програми.

Загальні способи захисту від статичного дослідження

Одним з розповсюджених методів захисту від статичного дослідження є обман дизасемблера. Методи цієї групи полягають у тому, щоб заплутати дизасемблер: підсунути дані замість коду, дезорієнтувавши його логіку, повести його по помилковому сліду, підсунути зайві фрагменти коду і т. д. Всі ці способи обману можна поділити на такі групи [1]:

- різноманітні методи обфускації коду (лексична обфускація коду, обфускація даних і потоку управління програмою);
- ускладнення логіки програм шляхом вбудовування надлишкового, мертвого і недосяжного коду із застосуванням непрозорих предикатів;
- різноманітні додаткові методи боротьби з інтерактивними та автоматичними дизасемблерами, що базуються на недосконалостях дизасемблерів або на недокументованих можливостях окремих компіляторів.

Це авангардні і досить перспективні методи захисту ПЗ не тільки від дизасемблювання, а й від налагоджування. Але сучасні дизасемблери здатні розпізнати виклики стандартних функцій, виділити локальні змінні в процедурах і надати інший подібний сервіс. Особливо у цьому можуть стати у нагоді інтерактивні дизасемблери, які формують вихідний текст/лістинг так, як це роблять автоматичні дизасемблери. Однак інтерактивні дизасемблери відрізняються від автоматичних наявністю потужного інтерфейсу для користувача, що сильно полегшує здатність аналізувати код програми.

Інтерактивні дизасемблери, як правило, дозволяють [1, 2]:

- змінювати імена змінних, міток, підпрограм і т. д., вводити імена для нових адрес, видаляти наявні мітки / імена;
- шукати послідовності символів в результуючому тексті і послідовності байтів в можливість втілення коді;
- повторно аналізувати код ділянки коду в послідовність асемблерних команд або директив DB;
- задавати коментарі до підпрограм, перериванням і т. д., які автоматично розставляються біля всіх відповідних викликів;
- переглядати перелік сегментів програми;
- редагувати дизасемблювати текст з автоматичною модифікацією виконуваного коду.

Суть методу «динамічної фуфелізації»

Вищеописані способи хороші проти автоматичних дизасемблерів. Однак заплутати IDA або будь-який інший інтерактивний дизасемблер тільки з їх допомогою не вдасться. Точніше, вдасться, але лише до того моменту, поки зламник не здогадається примусово позначити засмічуючі байти як Undefined, а все після них – як код. Відразу ж після цього він отримає можливість аналізу захищуваної програми в середовищі дизасемблера. Відомо, що різні інтерактивні дизасемблери дозволяють не тільки міняти вже дизасембльований код, але і втручатися в сам процес дизасемблювання.

Проти цього існує складний, але ефективний прийом, який отримав на хакерському жаргоні назву "динамічний фуфель" [2].

Суть прийому полягає в тому, що засмічуючі байти ніяк не обходяться командами передачі управління. Вони заміщуються нешкідливими командами (NOP, STI і ін.) [3]. Вже в ході виконання програми, н "свідомо до першого запуску підпрограм, що містять ці "фуфелі". Іншими словами, фрагмент програми, який захищається від дизасемблювання, дійсно не може бути запущений у тому вигляді, в якому програма знаходиться на диску. Швидше за все, це призведе до зависання комп'ютера. Однак, необхідно зробити так, щоб після запуску програма зчитувала звідкись дані, необхідні для усунення засмічуючих байтів, і замінила їх на команди, що роблять програму повноцінною і впливають на нормальний хід виконання програми.

Як результат, злом програми, яку захистили таким способом, – тривалий і трудомісткий процес, навіть якщо дані для "дефуфелізації" містяться в коді самої програми.

Можна і ще ускладнити злом, зберігаючи ці дані на нульовій доріжці або у навмисно залишеному при форматуванні "зазорі" між розділами дискової підсистеми. У такому випадку саме по собі копіювання секретної програми нічого зламнику не дасть – він втратить дані, необхідні для її перетворення до нормально працездатного стану. Окрім того, що це надзвичайно ускладнить роботу зламника, такий спосіб захищає ще й від несанкціонованого копіювання, оскільки тут є елемент прив'язки до дистрибутивного носія.

Висновки

Проведені дослідження стосуються одного з авангардних способів захисту програмного забезпечення від статичного дослідження, а саме від дослідження кодів програм за допомогою інтерактивних дизасемблерів. Запропоновано для захисту використовувати в якості засмічуючого коду так звані «фуфелі», тобто деякі довільні, але побудовані за певними правилами, фрагменти коду, з які або «завішують» програму, або призводять до зависання усієї системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Каплун, В. А. Захист програмного забезпечення. Частина 2 : навчальний посібник / Каплун В. А., Дмитришин О. В., Баришев Ю. В. – Вінниця : ВНТУ, 2015. – 106 с.
2. Абашев А. А. Ассемблер в задачах защиты информации / Абашев Алексей, Жуков Иван, Иванов Михаил – М. : Кудиц-Образ, 2004. – 544 с.
3. Касперски К. Техника и философия хакерских атак / Крис Касперски – М. : Солон-Р, 2006. – 272 с.

Каплун Валентина Аполінаріївна, старший викладач кафедри захисту інформації, Вінницький національний технічний університет, Вінниця, e-mail: valuka8379@gmail.com.

Kaplun Valentyna, senior lecturer in information security, Vinnytsia National Technical University, Vinnytsia, email : valuka8379@gmail.com.