УДК 004.63

Христич А.Ю., студентка гр. СП -10-1д, Пожуєв А.В., доцент, к.ф.-м.н., науковий керівник

АНАЛІЗ РЕ-ФАЙЛІВ ТА ДОСЛІДЖЕННЯ МЕТОДІВ ЇХ ЗАХИСТУ

Запорізька державна інженерна академія, кафедра ПЗАС

Сьогодні питання безпеки програмного забезпечення стоїть дуже гостро у всьому світі. Вже багато років аспект захисту програмних продуктів та комп'ютерів закладений до політики різних провідних корпорацій та компаній. Чому ж так важко й важливо виявити загрозу? Найбільш небезпечна ситуація, коли шкідливий код виконується не сам по собі, а разом із застосунком, який користувач сам встановив. Цього можна добитися впровадженням коду до PE- файлів. Але такий метод можна використовувати не тільки для нанесення ушкоджень,а й для захисту програмних продуктів. Portable Executable (PE) - формат виконуваних файлів, об'єктного коду і динамічних бібліотек, використовуваний в операційній системи Microsoft Windows [1]. PE- файл представляє собою заголовок (header), сторінковий імідж (image page) і необов'язковий оверлей (overlay). Представлення PE-файлу в пам'яті називається його віртуальним образом (virtual image) або просто образом.

Є три класичних підходи впровадження коду:

1. **Статичне впровадження коду** - зміна коду програми безпосередньо на диску [2]і. До реалізації цього підходу відносять такі методи :

1.1. Впровадження в заголовок - мінімальна кратність фізичного вирівнювання становить 0x200 байт, а заголовок зазвичай займає близько 0x300 байт. Виходячи з цього, в заголовку, як правило, знайдеться деякий вільний простір для впровадження коду.

1.2. Впровадження в секцію без зміни її віртуального положення - для вибору придатної до впровадження секції та виявлення вільного місця треба проаналізувати всі секції і вибрати таку, віртуальний розмір якої не потрібно буде змінювати після впровадження.

1.3. Впровадження в секцію зі зміною її віртуального положення - впровадження в початок або в кінець кодової секції. Найзручніший метод - впровадження коду у початок кодової секції. Треба відключити вирівнювання у файлі коду, а потім “підігнати” розміри та атрибути до потрібних. Тобто, виконується розширення простору для запису коду.

1.4. Створення нової секції - при його використанні розмір впроваджуваного коду необмежений. У таблиці секцій треба створити новий елемент та зробити вирівнювання розмірів та атрибутів інших секцій відповідно до величини нової секції.

1.5. Створення NTFS потоку - створити новій потік можна за допомогою API функцій а потім зчитувати дані за допомогою утиліти та коду – завантажника.

2. **Динамічне впровадження коду** - зміна коду завантаженої програми в пам'яті. Фізичний образ програми при цьому не змінюється. Серед найпоширеніших методів є такі:

2.1. IAT Hooking- здійснюється шляхом модифікації Import Address Table безпосередньо в пам'яті. Після завантаження файлу в пам'ять системний розвантажник змінює IAT, прописуючи адреси імпортованих функцій.

2.2. Inline Function Hooking- модифікується безпосередньо код цільової функції там, де це необхідно, і згодом управління передається сторонньому коду.

3. **Впровадження dll у процес** - обидва розглянутих методи вимагають наявності впроваджуваного коду в пам'яті процесу [3]. Існує ряд методів для досягнення цієї мети шляхом впровадження dll в цільовий процес:

3.1. Впровадження dll за допомогою реєстру – за допомогою ключа у реєстрі Windows всі додатки, які імпортують функції з User32.dll будуть завантажувати в свій адресний простір зазначені dll.

3.2. Впровадження dll з використанням Windows Hooks- для установки Windows Hook потрібно викликати API метод SetWindowsHookEx(), параметри якої вказують тип хука і функцію dll, яка буде обробляти перехоплені події (filter function). Після установки хука,

система поміщає dll в пам'ять цільового процесу.

3.3. Впровадження dll з використанням віддаленого потоку- створення у чужому процесі потік, який завантажить у даних процес потрібну нам dll.

Після аналізу усіх вищезазначених методів, було створено програму, яка дозволяє захищати Win32 файли. Цей проект був створений для того щоб попередити несанкціоноване використання лабораторних робіт. Захист був реалізований методом впровадження dll з використанням віддаленого потоку. Після запуску програми мається можливість вибрати необхідний файл та ввести пароль. Після підтвердження починається процес впровадження. Більшість API функцій Windows дозволяють процесу управляти тільки самим собою, але є й такі, які дозволяють виконувати управління над чужими, як CreateRemoteThread. Треба створити у необхідному процесі свій потік за допомогою функції CreateRemoteThread, а потім створеному віддаленому потоку можливість завантажувати певну dll за допомогою функції LoadLibrary. Існують дві реалізації цієї API: LoadLibraryA , LoadLibraryW . Перша призначена для роботи з рядком в ANSI- кодуванні. Варто виділити блок пам'яті в адресному просторі віддаленого процесу за допомогою VirtualAllocEx. Потім скопіювати в отриманий блок рядок з повним ім'ям бібліотеки функцією WriteProcessMemory, після чого отримати за допомогою GetProcAddress адресу LoadLibraryA всередині Kernel32.dll , так як вона містить функцію завантаження DLL і завжди проектується на один і той же діапазон адрес, тому в різних процесах точки входу в API- функцію будуть збігатися. І, зрештою, викликати CreateRemoteThread, створюючи віддалений потік в необхідному процесі, який викличе відповідну LoadLibrary і передасть їй адресу виділеної ділянки пам'яті. На цьому етапі DLL впроваджена в віддалений процес, а її функція DllMain отримала повідомлення DLL_PROCESS_ATTACH і може приступити до виконання потрібного коду. Коли DllMain поверне управління, віддалений потік вийде з LoadLibrary і повернеться у функцію BaseThreadStart, яка в свою чергу викличе ExitThread і завершить цей потік. При наступному запуску змінюваного файлу з’явиться запит про пароль.

Розглянувши всі методи впровадження коду до виконуваного файлу можна надати такі рекомендації:

1. Якщо потрібно щоб впроваджуваний код виконувався при кожному завантаженні програми, то варто використовувати методи статичного впровадження.
2. Треба пам'ятати, зміна файлу методом статичного впровадження дуже помітна та більшість антивірусів і брандмауерів зараз містять функцію контролю цілісності.
3. Якщо потрібно виконання коду лише раз після завантаження програми, а потім необхідно щоб він зник, варто використати динамічне впровадження, ця властивість забезпечує коду високу скритність.
4. При наявності невеликого досвіду у роботі з захистом виконуваних файлів варто скористатися методами впровадження коду до заголовку або впровадженням dll за допомогою реєстру.
5. Якщо впроваджуваний код має великий розмір краще звернутися до методів правки таблиці імпорту або створення нової секції.
6. Для найбільшої скритності коду треба використовувати методи створення NTFS потоку або впровадження в секцію зі зміною її віртуального положення.
7. Якщо треба захистити консольні застосунки не варто використовувати впровадження dll за допомогою реєстру, бо вони не імпортують функції з User32.dll.

Література

1. Вікіпедія [Електронний ресурс]:вільна бібліотека. / режим доступу: https://uk.wikipedia.org/, вільний.
2. Касперски К. «Техника внедрения и удаления кода из PE- файла» [Електронний ресурс]/ режим доступу: www.insidepro.com/kk/018/018r.shtml, вільний.
3. Jeffrey Richter «Load Your 32-bit DLL into Another Process’s Address Spase Using INJLIB» / J. Richter. - Microsoft System Journal/9 №5, May 1994.