УДК 303.7
Марков О. С.[1], Зайко Т. А.[2]
[1]студ. гр. КНТ-117 НУ «Запорізька Політехніка»
[2] канд. техн. наук, доц. НУ «Запорізька Політехніка»

РЕАЛІЗАЦІЯ УСКЛАДНЕННЯ ЛОГІКИ ЯК МЕТОД ЗАХИСТУ ВІД ДИЗАСЕМБЛЮВАННЯ

Не буде перебільшенням констатувати, що механізм дизасемблювання (статичного аналізу коду) таїть в собі велику небезпеку для переважної більшості продуктів програмного забезпечення (ПЗ). На думку Є. Касперського, фахівця кібербезпеки світового рівня, сучасні версії дизасемблерів можуть виявляти практично будь-який захист [1]. Однак це в жодному разі не є спростуванням методів та прийомів захисту від дизасемблювання кодів.

Важливим для розуміння процесу дизасемблювання, є той факт, що його виконання – процес надзвичайно кропіткий, перш за все, через складність розпізнавання коду. Так, дизасемблювання програми-хробака Морріса, що інфікував 6 200 комп'ютерів [2], спричинило застій в роботі мережі, збитки у часі від якого вимірювались мільйонами доларів США. Таким чином, умисне ускладнення логіки на декілька порядків програмістами робить зворотну інженерію і модифікацію тексту програми завданням неймовірно завищеної вартості.

Одним із найбільш ефективних засобів захисту кодів від дизасемблювання є їхнє ускладнення, або, висловлюючись неформально, маскування програми. Воно полягає в багатокроковій обфускації графа потоку керування програмою (ГПКП). Проте слід враховувати, що при застосовуванні комбінованих чи нестандартних методів маскування, об'єми програми мають вкладатися в межі обчислювальних систем [3].

Ідею ускладнення логіки можна виразити за допомогою таких

положень. По-перше, усі дуги графа потоку керування, які генеруються при маскуванні ГПКП, спричиняють стрімкий ріст часу роботи програми зловмисника, яка передусім спрямована на зняття захисту. По-друге, складність мовних конструкцій (як-от, безлічі непрозорих предикатів), що наявні у замаскованих функціях програми, нерідко зводять нанівець спроби проаналізувати та виділити «холостий» код програми.

Таким чином, реалізація вищенаведених положень при маскуванні у продуктах ПЗ суттєво підвищує рівень захищеності кодів. В процесі маскування програм виділяються декілька ідейних етапів, що можуть реалізовуватися різними програмними методами.

На першому етапі виконується збільшення розміру графа потоку управління функції. Даний крок маскування полягає у перетвореннях структури циклів функцій програми, що в найбільш примітивній реалізації виражається у клонуванні базових блоків. Більш складними засобами реалізації є переплетення окремих функцій, винесення груп операторів, вставлення функцій тощо.

На другому етапі відбувається руйнування структури ГПКП шляхом внесення в нього нових дуг та подрібнення базових блоків з додаванням нових, тимчасово не заповнених. Ціллю цього етапу є підготовка точок програми, куди в майбутньому буде внесено недосяжний, мертвий або надлишковий код.

На наступному етапі здійснюється заповнення заздалегідь підготовлених порожніх блоків інструкціями, які не впливають на результат роботи програми.

Останній етап маскування полягає у поєднанні «холостого» коду з основним. Інструментами реалізації цього етапу є, перш за все, засоби мов програмування, що використовуються, наприклад, покажчики на блоки пам’яті, логічні, відносні та умовні оператори.

Варто зазначити, що описані етапи маскування не є неухильним алгоритмом ускладнення логіки, але їх покрокова реалізація збільшить ймовірність збереження даних щонайменш від програм зловмисників-любителів.

На сьогодні, існує велика кількість ефективних (здебільшого, комерційних) дизасемблерів. Теперішніми фаворитами даної галузі можна вважати IDA Assembler та фреймворк Radare2 [4]. IDA містить вбудований відлагоджувач, декомпілятором коду застосунків на C, що дозволяє швидко аналізувати програми (на щастя, професійна версія ліцензії IDA доступна вузькому колу осіб). Radare2 є безкоштовним продуктом, що підтримується спілкою розробників; крім відлагоджувача, він також містить декомпілятор, hex-редактор, утиліти порівняння бінарних файлів тощо.

Підсумовуючи наведене, можна сказати, що боротьба проти

зловмисного дизасемблювання відбувається передусім на логічному (інформаційному), а не на фізичному рівні, що передбачає неосяжний простір для фантазії програмістів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Касперски К. Техника и философия хакерских атак / К. Касперски. – М.: Солон, 1999. – 272 с.

2. Хробак Морріса: [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Хробак_Моріса

3. Захист програмного забезпечення. Частина 2 : навчальний посібник / В. А. Каплун, О. В. Дмитришин, Ю. В. Баришев. – Вінниця : ВНТУ, 2014. – 105 с.

4. Выбираем инструменты для реверса: [Електрон. ресурс]. – Режим доступу: https://xakep.ru/2018/12/04/apps4hack/