ОБОЗРЕНИЕ
ПРИКЛАДНОЙ И ПРОМЫШЛЕННОЙ
Том 19 **МАТЕМАТИКИ** **Выпуск 5**
2012

В. П. Зязин, **М. В. Федюкин** (Москва, МИРЭА, ООО «Линфо»). **О защите компьютерных программ от реверс-инжениринга.**

Защита программного обеспечения от реверс-инжениринга – восстановления алгоритмов, реализуемых программой, по ее коду относится к числу актуальных задач в связи с решением проблем авторского права, предотвращения несанкционированной модификации алгоритмов и др. В докладе рассматривается метод защиты программы с помощью обфусцирующего преобразования (обфускации), т. е. преобразования программы в программу, идентичную по функционалу, но существенно затрудняющую понимание работы реализованных в программе алгоритмов. Общепринятая классификация обфусцирующих преобразований предложена в [2]. Обзор математических задач, связанных с изучением таких преобразований, достаточно полно представлен в [5].

Считается, что если злоумышленник имеет полный контроль над средой, в которой исполняется программное обеспечение, то предотвратить дизассемблирование практически невозможно. В [1] предлагается использовать разветвляющие функции (branching functions) для обфускации вызовов инструкций типа CALL, чтобы повысить трудоемкость применения статического и динамического анализа кода путем предотвращения реконструкции графа потока управления программы (control flow graph).

Предлагается следующий метод, развивающий идеи публикаций по методам обфускации, в частности, [1], для которого получена оценка сложности работы деобфускатора. Разбиваем код программы P на фрагменты, содержащие по небольшому количество инструкций кода и не позволяющие извлечь из них какую-либо полезную информацию, не зная общего контекста, в котором эти фрагменты используются в приложении. В конец каждого фрагмента добавляем безусловный переход на разветвляющую функцию, которая во время выполнения вычисляет адрес следующего фрагмента и затем осуществляет переход по этому адресу. Процесс вычисления адреса зависит от того, какие фрагменты уже были выполнены. Без знания предыдущих фрагментов нельзя однозначно вычислить адрес следующего. Чтобы выполнить данное требование, к каждому фрагменту добавляется подпись. Во время выполнения фрагмента вычисляется значение хэш-функции, зависящей от подписей всех предыдущих фрагментов, которое передается в качестве входного параметра в таблицу поиска, содержащую адрес следующего фрагмента. Без знания значения хэш-функции вычисление адреса последующего фрагмента сводится к его угадыванию.

Для противодействия динамическому анализу граф потока управления программы P модифицируется так, чтобы он содержал большее количество путей. Добавляются разветвления, зависящие от входных данных. Модифицированный граф назовем графом фрагментов. В нем каждая вершина представляет собой состояние программы, а кодами фрагментов помечены соответствующие дуги. Для каждой вершины создается несколько (не менее двух) выходящих ребер. Все выходящие ребра начинаются с одной и той же инструкции и отличаются только кодом фрагмента. В дальнейшем, эти фрагменты подвергаются диверсификации. Основная идея предлагаемого алгоритма диверсификации заключается в следующем:

1) Каждая инструкция разбивается на несколько инструкций, которые совместно семантически эквивалентны исходной инструкции.

2) Между инструкциями вставляется фиктивный код. Каждая фиктивная инструкция обращается к данным, используемым во фрагменте.

Чтобы удалить фиктивные инструкции, атакующий должен быть уверен, что впоследствии соответствующий код не повлияет на работу программы. Это сложная задача, так как атакующий видит только код данного фрагмента и не знает, какие фрагменты и в каком порядке будут выполнены впоследствии.

Алгоритм обфускации программы P строит граф фрагментов и сохраняет его структуру в памяти, позволяя осуществлять обход графа во время обфускации. У атакующего есть доступ только к обфусцированной программе P^*, которая не содержит в явном виде описание структуры графа фрагментов.

Утверждение. *В случае гарантированной защиты таблицы поиска время анализа программы P^* деобфускатором имеет экспоненциальную сложность по параметру n — количеству инструкций в исходном коде программы P.*

СПИСОК ЛИТЕРАТУРЫ

1. *Sebastian Schrittwieser*, *Stefan Katzenbeisser*. “Code Obfuscation Against Static and Dynamic Reverse Engineering”, in Information Hiding Conference 2011.
2. *Collberg C.*, *Thomborson C.*, *Low D.* A taxonomy of obfuscating transformations, 1997.
3. *Madou M.*, *Anckaert B.*, *De Sutter B.*, *De Bosschere K.* Hybrid static-dynamic attacks against software protection mechanisms. In Proceedings of the 5th ACM Workshop on Digital Rights Management, ACM, 2005, p. 75–82.
4. *Barak B.*, *Goldreich O.*, *Impagliazzo R.*, *Rudich S.*, *Sahai A.*, *Vedhan S.*, *Yang K.* On the (Im)possibility of obfuscating programs, Advances in Cryptology — Crypto01, Springer Verlag, 2001, p. 1–18.
5. *Варновский Н. П.*, *Захаров В. А.*, *Кузюрин Н. Н.* Математические проблемы обфускации. Математика и безопасность информационных технологий. Материалы конференции в МГУ, 28–29 октября 2004 г., М.: Изд-во МЦНМО, 2005.