УДК 681.324

МАТЕМАТИЧЕСКИЕ И ПРОГРАММНЫЕ СРЕДСТВА ОБФУСКАЦИИ ПРОГРАММ

А. Г. Поздеев, В. Н. Кривопалов, Е. В. Ромашкин, Е. Д. Радченко

Одной из наиболее важных проблем в сфере защиты ПО от нелегального использования является защита от обратной инженерии и несанкционированной модификации. Целью данной работы является анализ подходов к защите программ от исследования методами запутывания (обфускации) кода, создание представления компьютерных программ, содержащего всю необходимую для производимых преобразований информацию, и программная реализация данного представления. Такое представление применимо для программ, исполняемых в различных средах, и позволяет решать широкий спектр задач анализа программ, подлежащих защите.

Традиционно запутывающим преобразованием (обфусцирующим преобразованием, или обфускацией) программы неформально называют такое преобразование кода программы, которое, полностью сохраняя функциональность программы, затрудняет её анализ и возможность модификации [1, 2]. Основная сложность формализации понятия запутывающего преобразования связана с описанием критериев «затруднения» анализа программы [3]. Задача обфускации — придумать алгоритм, который, получая на вход некоторую программу P, преобразует её в программу $O(P)$ таким образом, что код программы $O(P)$ не должен раскрывать информацию об алгоритмах, параметрах и внутренней структуре программы P.

Выделяют три ключевых подхода к применению запутывающих преобразований [4]. В первом случае программа запутывается на её языке программирования так, что P и $O(P)$ — это программы, записанные на том же языке. Второй подход предусматривает разработку специального языка программирования, не допускающего возможности анализа программ, написанных на нём. В третьем подходе предполагается использование промежуточного языка, называемого языком обфускации и предназначенного исключительно для внесения путаницы в тексты на нём. В данной работе представлены исследования и разработки, ориентированные на третий подход.

Сформулированы следующие требования к языку обфускации: он должен соответствовать парадигме объектно-ориентированного программирования, допускать трансляцию из программ в различных исполняемых кодах и обратную трансляцию в исходное представление и позволять в процессе трансляции программы в него производить сбор и анализ всей необходимой для проведения запутывающих преобразований информации. Язык обфускации, удовлетворяющий этим требованиям, создан и используется в практической разработке автоматизированной системы обфускации программ.

В рамках практической части работы реализована библиотека программ для запутывания и анализа программного кода, состоящая из трех компонентов. Задача первого компонента — трансляция входной программы из языка исполняемого кода в язык обфускации. Второй компонент — непосредственно обфускатор, реализующий запутывающие преобразования для программы на языке обфускации. Кроме того, обфускатор в наглядном виде представляет полученную аналитическую информацию, содержащуюся в описании программы. Третий компонент выполняет обратную трансляцию программы в язык исполняемого кода.

Внутреннее представление программы на языке обфускации содержит большое количество аналитической информации. Для представления этой информации в удобном для исследователя виде в системе используется визуализация графов потоков управле-

ния и потоков данных. Реализована возможность построения подграфов по заданным критериям. Таким образом, система может использоваться и как деобфускатор [5, 6].

Разработан и реализован метод запутывания программ — метод обфускации по данным. Он является комбинацией и модификацией существующих методов запутывания и обладает уникальными свойствами. Суть метода следующая.

В графе вызовов функций программы по некоторому заданному критерию выбирается поток управления, соединяющий две смежные вершины графа. Соответствующий поток данных из вызывающей функции (функция A) в вызываемую (функция B) разбивается на N потоков таким образом, что через разные образованные потоки проходят данные, принимающие значения из непересекающихся диапазонов. Далее создаются N копий функции B (B_i, $i = 0, 1, \ldots, N-1$), и вызывающая функция модифицируется таким образом, чтобы каждый из образованных потоков данных направлялся в отдельную копию функции B. На этом этапе происходит и образование новых потоков управления: управление из функции A может быть передано на любую из функций B_i.

Поскольку диапазоны допустимых значений разных потоков данных не пересекаются, то формируется однозначное отображение из множества диапазонов $\{I_j : j = 0, 1, \ldots, N-1\}$ значений данных в множество функций $\{B_j\}$. В каждую функцию B_j внедряется проверка на принадлежность входного аргумента интервалу I_j. Если результат проверки — истина, то далее работа программы продолжается без изменений. При отсутствии внешних воздействий (со стороны исследователя программы) результат проверки всегда будет равен истине. Однако, если проверка вернет значение ложь (воздействие со стороны исследователя), то для противодействия исследователю нужно каким-либо образом нарушить правильное функционирование программы, например изменить значения в произвольных потоках данных или передать управление на некоторую функцию, которая в нормальных условиях не должна получить управление. Таким образом, исследователь не может произвольно изменить значение данных в рассматриваемом потоке и, чтобы пройти по всем функциям B_j, ему нужно перебрать все диапазоны значений. Следовательно, программа должна быть запущена как минимум N раз, и время исследования возрастает в N раз.

Данный метод предназначен для применения в комбинации с другими методами обфускации, затрудняющими обнаружение схожей функциональности функций B_j.

ЛИТЕРАТУРА

1. *Collberg C., Thomborson C., Low D.* Manufacturing cheap, resilient, and stealthy opaque constructs // Proc. Symp. Principles of Programming Languages (POPL'98), Jan. 1998.
2. *Collberg C., Thomborson C., Low D.* Taxonomy of Obfuscating Transformations. Режим доступа: `http://www.cs.arizona.edu/collberg/Research/Publications/CollbergThomborsonLow97a/index.html`, свободный.
3. *Чернов А. В.* Исследование и разработка методологии маскировки программ: Дис. канд. физ.-мат. наук. Моск. гос. ун-т им. М. В. Ломоносова. М., 2003. 133 с.
4. *Лифшиц Ю. М.* Запутывание (обфускация) программ. Обзор. СПб.: СПб. отд. Мат. инст. им. В. А. Стеклова РАН, 2004. Режим доступа: `http://logic.pdmi.ras.ru/~yura/of/survey1.pdf`, свободный.
5. *Чернов А. В.* Анализ запутывающих преобразований программ / Библиотека аналитической информации [Электронный ресурс]. Режим доступа: `http://www.citforum.ru/security/articles/analysis/`, свободный.
6. *Barak B., Goldreich O., Impagliazzo R. et al.* On the (Im)possibility of Obfuscation Programs // LNCS. 2001. V. 2139. P. 1–18.