УДК 336

ИСПОЛЬЗОВАНИЕ СЕТЕЙ ПЕТРИ ДЛЯ ОБФУСКАЦИИ ДВОИЧНОГО КОДА АЛГОРИТМА

Ошуркевич Максим Валерьевич
Магистрант кафедры ПИКС
Белорусский государственный университет
информатики и радиоэлектроники (Беларусь, г. Минск)

Цель статьи рассмотреть определение сети Петри и использование сети Петри для защиты программного кода

Ключевые слова: программный код, сети Петри, двоичный код, система, алгоритм, обфускация

USE OF PETRI NETS FOR OBFUSCATION OF BINARY ALGORITHM CODE

Ashurkevich Maksim
Belarussian State University
of Informatics and Radioelectronics (Belarus, Minsk)

The purpose of the article is to consider the definition of Petri nets and the use of Petri nets to protect programcode

Keywords: program code, Petri nets, binary code, system, algorithm, obfuscation

Маркировка сетей Петри

Фишка (точка, маркер) - это примитивное понятие сетей Петри. Фишки присваиваются и можно считать, что они принадлежат позициям. Количество и положение фишек при выполнении сети Петри могут меняться. Фишки используются для определения выполнения сети Петри [1].

Определение 2: Маркировка p сети Петри C = (P, T, I, O) есть функция, отображающая множество позиций P в множество неотрицательных целых чисел N: μ: P → N

Маркировка μ есть присвоение фишек позициям сети Петри.

Маркировка μ может быть также определена как n-вектор μ = (μ1, μ2, ... , μn), где n = |P| и каждое μi ϵ N, I = 1,2, ..., n. Вектор μ определяет количество фишек для каждой позиции pi сети Петри. Количество фишек в позиции pi есть μi. Связь между определениями маркировки как функции и как вектора очевидным образом устанавливается соотношением μ (p_i)=μi.

Определение 3: Маркированная сеть Петри M=(C, μ) есть совокупность структуры сети Петри C = (P, T, I, O) и маркировки μ и может быть записана в виде M = (P, T, I, O, μ).

Задача достижимости сети Петри

Задача достижимости - одна из самых важных задач анализа сетей Петри. Были поставлены следующие четыре задачи достижимости для сети Петри C = (P, T, I. O) с начальной, маркировкой μ.

Определение 7: Задача достижимости. Выполняется ли для данной μ': μ'ϵ R(C, μ)?

Определение 8: Задача достижимости подмаркировки. Для подмножества P' ϵ P и маркировки μ' существует ли μ" ϵ R(C, μ). такая, что μ"(p_i) = μ'(pi) для всех p_iϵ P"?

Задача достижимости подмаркировки ограничивает задачу достижения до рассмотрения только подмножества позиций, не принимая во внимания маркировки других позиций. Задача достижимости нуля выясняет, является ли достижимой частная маркировка с нулем фишек во всех позициях. Задача достижимости нуля в одной позиции выясняет, возможно ли удалить все фишки из одной позиции.

Все эти задачи являются эквивалентными. Эти теоремы и доказательства описаны в работе Хэку[2].

«Поскольку задачи подмножества и равенства для множеств достижимости сетей Петри неразрешимы, то возможно, что неразрешима также и сама задача достижимости. Однако в настоящее время вопрос, разрешима ли (или неразрешима) задача достижимости, открыт. На сегодняшний

день не существует ни алгоритма, решающего задачу достижимости, ни доказательства того, что такого алгоритма не может быть[3].»

Для анализа свойств сетей Петри наиболее удобно использовать граф представления множества достижимости сетей Петри. В этом дереве представлены все достижимые состояния сетей Петри. Общий путь построения дерева достижимости сети Петри заключается в определении всех разрешенных переходов в соответствующей маркировке с последующим анализом соответствующего очередного состояния (маркировки), достигающихся независимых автоматических последовательностях запусков всех разрешенных переходов предыдущей маркировки.

Решение задачи достижимости с использованием дерева достижимости фактически сводится к методу полного перебора. Поэтому защищенность здесь определяется только временем и ресурсами, затраченными па полный перебор. А это два наиболее важных критерия, определяющие степь защищенности.

Таким образом, сети Петри являются прекрасным инструментарием для защиты исходного кода.

Построение сети Петри

Сеть Петри - абстрактное понятие. Для решения поставленной задачи будем использовать модель, схожую по поведению с сетью Петри. Представим её в терминах, описанных выше.

В качестве меток сети выступают ячейки памяти. Разобьем метки на три типа: фиксированные метки, метки с вычисляемыми значениями, метки с автоматически сгенерированными значениями.

Первая группа - это те метки, в которых мы хотим получать искомые константы (целевые значения). Эти константы берутся из исходного кода, и именно вместо них подставляются инструкции, которые вычисляют сеть. Такие метки строго фиксированы, потому, что необходимо уметь

однозначно определять, в какой ячейке находится константное значение для той или иной операции исходного кода. Каждый раунд изменяет сеть: все метки могут быть пересчитаны.

Поэтому важно соблюдение порядка выполнения сети. Целевые значения в сети идут в том же порядке, что и в коде (т.е. чем позже команда в коде, тем больше номер раунда)

Ко второй группе относятся те метки, значение которых неизвестно в момент создания сети. Эти значения должны быть подобраны таким образом, чтобы после очередного раунда в заданной ячейке из первой группы получилось искомое значение.

Но если все метки сети считать неизвестными, то сложность построения такой сети возрастает. Поэтому часть неизвестный заполняется случайным образом сгенерированными значениями (третья группа).

В ходе исследования было получено, что при заполнении половины меток сети случайными значениями, всегда существует решение для данной сети, позволяющее построить её по указанным выше правилам.

Каждый узел (метка) имеет двух родителей, соответственно значение каждого узла вычисляется как сумма значений родителей. Исходя из этого, можно построить систему линейных диофантовых уравнений для расчета неизвестных узлов.

Определение 8: Диофантово уравнение — это уравнение вида

$P(x_1, .. ,x_m) = 0$

где P — целочисленная функция (например, полином с целыми коэффициентами), а переменные принимают целые значения.

Определение 2.12: Линейное диофантовое уравнение имеет вид:

$a_1x_1 + a_2x_2 + ... + a_kx_k = d$

При разработке инструментального средства для защиты использовался метод решения системы линейных диофантовых

уравнений в кольце вычетов через нормальную форму Эрмита - Смита в кольце вычетов согласно методу, предложенному Авдошиным, который также имеет вычислительную сложность $O(n^3)$.

Сложность решения линейного уравнения в целых числах в кольце вычетов полиномиальна и равняется $O(n^3)$, значит и сложность построения такой сети Петри тоже равняется $O(n^3)$.

Получаем сеть, которую можно построить за полиномиальное время, а задача достижимости для неё является NP - полной. Такая сеть является хорошим инструментом для защиты исходного кода.

Оценка сложности взлома сети Петри

В том случае, если часть данных о процессе защиты известна, то задача восстановления кода становится полиномиальной. Например, если известен алгоритм защиты и структура, то при

анализе возможно построение сети Петри, а значит и удаление всех инструкций сети за полиномиальное время.

В тоже время если известен генератор псевдослучайных чисел(ГПСЧ) и его начальное значение, то можно выделить последовательность раундов, что тоже приводит к полиномиальному решению задачи.

Таким образом для эффективности защиты при помощи сети Петри, необходимо обеспечить секретность:

- Используемой сети Петри
- Применяемого при защите ГПСЧ
- Принципа масштабирования или генерации, в том случае, если сеть Петри получается при помощи самоподобия или каким-либо иным алгоритмом.

Защита исходного кода с помощью сети Петри

Для защиты программного продукта с помощью реализуемой системы защиты необходимы:

- исполняемый файл защищаемого продукта;
- виртуальный адрес защищаемой функции.

Обычно не весь исходный код несет в себе ценную информацию, а лишь какая-то его часть, например функция, реализующая алгоритм, являющийся интеллектуальной собственностью. Именно такую функцию необходимо защищать.

Имея такие начальные данные, защита строится по следующему алгоритму:

1. Из исходного исполняемого файла получается набор ассемблерных инструкций;
2. Ассемблерные инструкции преобразуются в набор инструкций LLVMIRBuilder;
3. В полученный код встраивается сеть Петри;
4. Обфусцированный код выполняется на виртуальной машине LLVM;
5. По данному виртуальному адресу в исполняемый файл встраивается новый исполняемый файл, сгенерированный виртуальной машиной LLVM, который и выполняет основную логику исходной программы.

Литература

1. Котов В. Е. Сети Петри/ В. Е. Котов//М.: Наука. Главная редакция физико-математической литературы — 1984 — 160 с.

2. Four F., Защита программ с помощью сетей Петри./ F. Four //Портал для программистов «BACM» — 2011 — Режим доступа: http://www.wasm.ru/article.php?article=petri (дата обращения 22.10.2011).

3. Peterson, J. L., Petri net theory and the modeling of systems./ J. L. Peterson //Prentice-Hall, Englewood cliffs — 1981 — 290 стр