Т. В. Петрова, А. А. Краснопевцев

МЕТОД ОЦЕНКИ СЛОЖНОСТИ КОМПРОМЕТАЦИИ СИСТЕМЫ ЗАЩИТЫ ПРИЛОЖЕНИЙ, ИСПОЛЬЗУЮЩЕЙ ПРОГРАММНО-АППАРАТНЫЕ МОДУЛИ

В рамках сформулированного метода оценки сложности компрометации предлагается проводить оценку посредством теории конечных автоматов [1, 2]. Защищаемое приложение интерпретируется в виде конечного автомата, в соответствии со следующим набором правил:

— множество состояний — виртуальные метки, обозначающие начало отдельного базового блока в приложении;

— множество входов составляют переменные, в соответствии со значениями которых осуществляется передача управления в приложении;

— множество выходов составляют базовые блоки приложения;

— начальными состояниями автомата помечаются такие состояния, которые напрямую достижимы из точки входа защищаемого приложения;

— конечными состояниями автомата помечаются состояния, которые на графе потока управления приложения представлены висячими вершинами;

— функции переходов и выходов конечного автомата представляют собой таблицу переходов в соответствии с графом потока управления.

Множество состояний S — множество виртуальных меток в приложении. Каждая виртуальная метка соответствует началу базового блока приложения. Таким образом, множество состояний будет строиться в соответствии с ветвлениями графа потока управления приложения ρ.

Множество входов X — набор переменных приложения ρ, в соответствии со значениями которых происходит тот или иной переход в графе потока управления защищаемого приложения. Иными словами, если в зависимости от значения некоторой переменной в теле программы происходит переход от одной вершины графа потока управления к другой, то данная переменная является элементом множества X.

Множество выходов Y — набор инструкций, которые приложение будет выполнять при переходе из одного состояния в другое, — базовые блоки приложения.

Отображения, интерпретирующие функцию переходов h и функцию выходов f, задаются таблицей связей между состояниями автомата и связей между состояниями автомата и элементами множества выходов.

В результате такой интерпретации система защиты приложения представляется в виде универсальной алгебры с носителем алгебры, составленным из конечных автоматов, описывающих приложения. Множеством преобразований, задаваемых данной алгеброй, будет являться набор преобразований, которые осуществляются над защищаемым приложением с целью приведения его к виду, трудному для анализа. Основное преимущество предложенной модели заключается в том, что данная модель применима для описания программно-аппаратных методов защиты приложений. В связи с тем, что класс программно-аппаратных методов зачастую воздействует на переходы защищенного приложения или же на базовые блоки приложения, модель, представленная в виде конечного автомата, интерпретирует само приложение с позиции, наиболее интересной для программно-аппаратных методов защиты приложений.

Применение предложенного метода моделирования поведения приложения дает возможность оценки сложности компрометации системы защиты приложений, использующих внешний аппаратный модуль. Предлагаемый метод оценки может применяться как для приложений, компилируемых

в промежуточное представление, для которого он и разрабатывался, так и для приложений, компилируемых непосредственно в машинный код. Более того, в некоторых случаях предлагаемый метод оценки сложности компрометации систем защиты приложений может быть использован и для оценки сложности компрометации программных систем защиты приложений.

Сам метод оценки сложности компрометации исследуемой системы защиты приложений от несанкционированного копирования реализуется посредством совместного анализа возможных атак на систему защиты и анализа исследований, которые должен осуществлять аналитик при попытке ее компрометации.

Для предложенного метода защиты приложений, посредством разделения графа потока управления приложения [3], моделирование сложности компрометации будет происходить по следующим правилам.

На первом шаге анализа необходимо построить автомат, описывающий защищаемое приложение. Построение автомата будет осуществляться в соответствии с правилами, представленными ранее. В результате после построения автомата из графа потока управления приложения будет построена математическая абстракция, интерпретирующая защищаемое приложение с позиции процесса передачи управления.

Далее, необходимо проанализировать, какой вклад вносят те или иные преобразования, осуществляемые над защищаемым приложением. В ходе ранее предложенного метода защиты приложений предлагается сначала использовать «классические» запутывающие преобразования. В результате применения выбранного набора преобразований в автомате, описывающем защищаемое приложение, будет происходить модификация различных множеств, составляющих автомат.

Рассмотрим воздействие каждого преобразования на автомат, описывающий приложение более детально:

– $r_u\{(A,k)\} \rightarrow \{A'\}$, $A' = ru\ (A,k) = \{X', Y', S', Ss', Se', f', h'\}$ – преобразование, аргументами которого являются автомат A, интерпретирующий защищаемое приложение P, и элемент ключа $k \in K$. В зависимости от элемента ключа преобразование осуществляет добавление вершины к графу потока управления приложения или, с точки зрения конечного автомата, осуществляется добавление элемента к множеству выходов Y, множеству входов X и множеству состояний S. Множества начальных и конечных состояний (Sb,Se) автомата A остаются неизменными, чтобы не нарушить логику работы приложения P.

– $r_d\{(A,k)\} \rightarrow \{A'\}$, $A' = r_d\ (A,k) = \{X', Y', S', S_s', S_e', f', h'\}$ – добавление «мертвого» кода к графу потока управления приложения. Рассматриваемое преобразование осуществляет добавление вершины к оригинальному графу потока управления приложения. В результате код, который был добавлен, не несет никакой функциональной нагрузки. Выполнение внедренного кода не вызывает никакой модификации переменных. Однако в связи с тем, что сложность восстановления оригинального графа потока управления и, как следствие, оригинального автомата A основывается на количестве базовых блоков приложения, применение описываемой операции позволит увеличить сложность анализа защищенного приложения.

– $r_c\ \{(A,k)\} \rightarrow \{A'\}$, $A' = r_c\ (A,k) = \{X', Y', S', S_s', S_e', f', h'\}$ – усложнение пути в графе потока управления защищаемого приложения. Суть преобразования заключается в том, что существующий элемент множества выходов $y \in Y$ конечного автомата A приложения P отображается в несколько элементов. В результате применения данного преобразования к графу потока управления защищаемого приложения происходит замена одной вершины графа на несколько. Причем логика работы защищенного приложения не изменяется. Добавленные вершины в совокупности интерпретируют тот же участок кода приложения, что и оригинальная вершина.

Последнее преобразование – преобразование затемнения, которое реализуется посредством разделения конечного автомата, описывающего защищаемое приложение. В результате

преобразования происходит «выгрузка» функций f и h автомата, описывающего защищаемое приложение. Сложность анализа приложения в худшем случае будет сводиться к восстановлению оригинального автомата, который описывает защищенное приложение. В связи с тем, что автомат, описывающий приложение, является автоматом Мура, так как выход зависит только от нового состояния, в которое автомат перешел в зависимости от входного сигнала, сложность восстановления автомата по известному набору множеств X, Y, S будет зависеть от количества элементов в множестве состояний. В результате построения системы защиты в соответствии с приведенным алгоритмом сложность восстановления оригинального конечного автомата будет эквивалентна полному перебору всех возможных автоматов.

Величина, характеризующая число возможных графов, построенных на имеющемся наборе вершин, получается за счет того, что при представлении графа G в виде матрицы смежности E размер матрицы будет n Ч n, т. е. в матрице будет n^2 элементов. Каждый элемент, в соответствии с определением матрицы смежности, может принимать значение $e_{i,j} \in \overline{0,1}$, в результате полное количество простых ориентированных графов будет составлять n^2. Необходимо учесть, что полностью несвязный граф, хотя и может существовать, однако осуществление анализа такого графа не имеет смысла. Бессмысленность анализа несвязного графа в том, что таким графом интерпретируется запутанное линейное приложение либо линейное приложение с большим количеством недостижимого кода. Значение полученной величины числа графов, построенных на имеющемся множестве вершин, может быть уменьшено за счет введения свойства слабой связности графа, однако, с точки зрения злоумышленника, данное свойство рассматриваемого графа потока управления не является обязательным. В связи с тем, что перед моментом разделения объектов автомата A может быть осуществлен процесс запутывания приложения ρ, граф G, интерпретирующий защищенное приложение, может не иметь свойства слабой связности. Иными словами, сложность компрометации системы защиты приложений посредством разделения конечного автомата, описывающего приложение, будет иметь сложность компрометации методом перебора, ограниченную функцией $0(2^{n2})$. Оценка снизу для предложенного метода защиты приложений осуществляется посредством оценки числа бинарных деревьев, которые можно построить на вершинах множества. Количество таких деревьев, в соответствии с теоремой Кэли для деревьев [4, 5], определяется величиной $\Omega(n^{n-2})$.

СПИСОК ЛИТЕРАТУРЫ:

1. *Фомичев В. М.* Дискретная математика и криптология. Курс лекций. М.: ДИАЛОГ-МИФИ, 2003. – 400 с.
2. *Князьков В. С., Волченская Т. В.* Введение в теорию автоматов. URL: http://www.intuit.ru/department/algorithms/intavth/.
3. *Краснопевцев А. А.* О защите приложений с использованием внешнего аппаратного модуля // Безопасность информационных технологий. 2011. № 1. С. 102-104
4. *Харари Ф., Палмер Э.* Перечисление графов / Пер. с англ. М.: Мир, 1977. – 326 с.
5. *Харари Ф.* Теория графов / Пер. с англ. и предисл. В. П. Козырева; под ред. Г. П. Гаврилова. Изд. 2-е. М.: Едиториал УРСС, 2003. – 296 с.