УДК 621.3

И.Г. Дровникова[1], В.П. Алферов[2], С.А. Змеев[2], Е.А. Рогозин[2], А.В. Хвостов[2], А.А. Окрачков[3]

([1]Воронежский институт МВД России, [2] Воронежский Государственный Технический Университет; [3]Военно-воздушная академия им. проф. Н.Е. Жуковского и Ю.А. Гагарина; e-mail: idrovnikova@mail.ru)

ДИНАМИЧЕСКИЙ КОНТРОЛЬ БЕЗОПАСНОСТИ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ ПРОГРАММНО-АППАРАТНОГО МОДУЛЯ

Предложен метод динамического контроля безопасности информации с использованием математического аппарата марковских и скрытых марковских моделей. Реализация метода предлагается в специализированном программно-аппаратном модуле.

Ключевые слова: информационная безопасность, динамический контроль, программно-аппаратный модуль.

I.G. Drovnikova, V.P. Alferov, S.A. Zmeev, E.A. Rogozin, A.V. Khvostov, A.A. Okrachkov

DYNAMIC CONTROL OF INFORMATION SECURITY USING SOFTWARE AND HARDWARE MODULE

Method of dynamic control of information security with the use of mathematical apparatus of Markov and hidden Markov models. Implementation of the method is offered in specialized software and hardware module.

Key words: information security, dynamic control, software and hardware module.

Статья поступила в редакцию Интернет-журнала 11 марта 2014 г.

В соответствии с принятыми в документации семейства ***операционных систем (ОС)*** "Windows" понятий, программа при выполнении представлена в ОС как процесс, а основной единицей исполнения программы является поток [1]. Процесс может иметь один или несколько потоков, каждый из которых представляет собой последовательность команд процессора. Динамика выполнения прикладной программы проявляется в изменении нагрузки, создаваемой этой программой на системные ресурсы, а также в изменении состояний потока с течением времени. Под состоянием потока можно понимать следующее.

Во первых, каждый поток состоит из команд процессора, выполняющихся последовательно через определённые временные интервалы. Эти интервалы являются постоянными и соизмеримы с тактовой частотой процессора. В зависимости от типа команды на её выполнение затрачивается различное количество тактов процессора. Следовательно, команды процессора могут рассматриваться как состояния потока в конкретный момент времени. Число таких состояний определяется количеством команд того микропроцессора, на базе которого построена вычислительная система.

Во вторых, каждый поток содержит в себе подпрограммы, порядок выполнения которых (порядок появления в потоке) также имеет определённую последовательность. Переход от одной подпрограммы к другой также происхо-

дит в определенные моменты времени. Таким образом, подпрограммы, используемые потоком, тоже можно рассматривать как состояния потока. Подпрограммы, вызываемые в контексте потока, могут входить в состав как данного процесса, так и в состав другого процесса, находящегося в данный момент в системе (экспорт функций), а также могут быть элементами библиотеки с динамическим связыванием. Число таких подпрограмм в потоке является конечной величиной.

Подпрограммы отличаются друг от друга выполняемыми функциями, следовательно, набором команд процессора, их распределением и количеством. Таким образом, контролируя последовательность команд процессора, можно распознавать выполняющуюся в данный момент подпрограмму.

Следующий подход в определении состояний потока заключается в том, что в качестве состояний потока можно рассматривать линейные участки кода (последовательность команд процессора, ограниченных командами передачи управления). Число таких состояний носит ярко выраженный индивидуальный характер для каждого потока. Кроме того, почти каждый линейный участок кода индивидуален как в рамках своего потока, так и в отношении других потоков, в том числе и принадлежащих другим процессам.

Таким образом, контроль выполнения прикладных программ может осуществляться путём отслеживания состояний потоков, принадлежащих этой программе. В каждом из трёх рассмотренных выше подходов к определению состояний потока число таких состояний ограничено и поток может находиться в любой момент времени только в одном из них. Переходы из одного состояния в другое возможны только в строго определённое время, следовательно, можно утверждать, что процесс выполнения потока имеет дискретные состояния и дискретное время.

Статистический характер состояний вычислительной системы и характеристик использования её ресурсов в процессе выполнения прикладных программ, а также дискретность протекания процессов в вычислительной среде обусловливают предпочтительность использования в качестве математического аппарата формального описания – аппарата ***Марковских цепей (МЦ)*** [2].

Марковской цепью называют последовательность испытаний, в каждом из которых система принимает только одно из N состояний полной группы, причём условная вероятность $P_{ij}(s)$ того, что в $s-м$ испытании система будет находиться в состоянии j, при условии, что после $(s-1)$-го испытания она находилась в состоянии i, не зависит от результатов остальных, ранее произведённых испытаний [3].

Таким образом, МЦ в произвольный момент времени может находиться в одном из N различных состояний $S_1, S_2, \ldots, S_n$. В дискретные моменты времени система может претерпевать изменения, переходя из одного состояния в другое (возможно, в то же самое) с определённой вероятностью [4]. В таких цепях каждое состояние соответствует наблюдаемому (физическому) событию.

Если наблюдения являются вероятностной функцией данного состояния, то есть имеют место два случайных процесса, один из которых является основным и ненаблюдаемым (скрытым), а второй случайный процесс дает последовательность наблюдений, по которым можно судить о скрытом процессе, МЦ называется скрытой [5].

Соответствующая прикладной программе МЦ может быть представлена следующим образом: в качестве состояний каждого потока программы может быть или последовательность вызываемых подпрограмм, или участки линейного кода, или команды процессора. Случайный процесс, происходящий в системе, состоит в том, что в последовательные моменты времени $t_1, t_2, \ldots T$ система оказывается в тех или других состояниях. В общем случае в моменты $t_1, t_2, \ldots$ система может не только менять состояние, но и оставаться в прежнем состоянии.

Процесс, происходящий в системе, можно представить как последовательность (цепь) событий, происходящих на каждом шаге (в каждый момент времени при совершении перехода), например:

$$S_1^0, S_2^1, S_1^2, S_2^3, S_3^4, \ldots, S_i^k, \quad (1)$$

где k – номер шага.

Для МЦ вероятность перехода из любого состояния S_i в любое S_j не зависит от того, когда и как система пришла в состояние S_i.

Такая МЦ описывается с использованием вероятностей состояний $S_1, S_2, \ldots, S_n$. Обозначив через q_t состояние системы в момент времени t, найдём вероятность перехода из состояния q_{t-1} в состояние q_t по формуле:

$$a_{ij} = P\left[q_t = S_j \,|q_{t-1} = S_i\right]. \quad (2)$$

где $1 \le i,\ j \le N$.

Переходные вероятности имеют следующие свойства:

$$a_{ij} \ge 0,\ \sum_{j=1}^{N} a_{ij} = 1. \quad (3)$$

Если переходные вероятности не зависят от номера шага, то такая МЦ называется однородной [3]. Как правило, переходные вероятности записываются в виде матрицы переходных вероятностей:

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{3n} \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}. \quad (4)$$

Некоторые из переходных вероятностей могут быть равны нулю, в этом случае переход из i-го состояния в j-е за один шаг невозможен. По главной диагонали матрицы стоят вероятности того, что система не выйдет из состояния S_i, а останется в нём.

При рассмотрении МЦ часто их представляют в виде графа состояний, где стрелками указаны направления переходов между состояниями, а рядом указаны переходные вероятности.

Используя описание потока прикладной программы в виде МЦ, можно определить вероятность того, что поток совершит переход из *i*-го состояния в *j*-е за *n* шагов. Это осуществляется по формуле:

$$P_i(k) = \sum_{j=1}^{N} P_j(k-1) P_{ij} \ , \qquad (5)$$

где N – количество состояний потока,

k – число шагов, за которые поток может достигнуть состояния S_i с вероятностью P_i

Следовательно, имея описание МЦ для каждого потока всех прикладных программ, используемых в ОС, можно, наблюдая переходы из одного состояния потока в другое, находить вероятность такого перехода для всех ***Марковских моделей (ММ)***. Зная такие вероятности, можно определить по максимальному значению, какой поток в данный момент выполняется. Если он принадлежит множеству потоков того процесса, который в настоящий момент находится в состоянии выполнения, можно сделать вывод о том, что данный процесс (программа) выполняется правильно.

Принимая во внимание то, что в качестве состояний потока могут быть команды процессора, участки линейного кода программы и отдельные подпрограммы, для каждого потока независимо могут существовать три ММ.

Учитывая тот факт, что такие состояния потока, как подпрограмма и линейный участок кода программы, содержат в себе другие возможные представления состояний (в первом случае – линейные участки кода, во втором – команды процессора), то каждый поток можно представить в виде ***скрытой ММ (СММ)***.

Таким образом, выполнение потока прикладной программы в вычислительной системе может быть представлено последовательностью смены состояний этого потока. В то же время каждое такое состояние характеризуется некоторым наблюдаемым (физическим) событием, под которым понимается выполнение команды процессора или факт вызова библиотечной функции. В процессе выполнения прикладная программа меняет свое состояние, переходя от вызова одной системной функции к вызову другой, от выполнения одной команды процессора к другой. В каждом очередном состоянии наблюдаемые процессы складываются в некоторую последовательность наблюдений

$$O = [O_1, O_2, \ldots, O_t]. \qquad (6)$$

В прикладной программе можно выделить несколько уровней идентификации: уровень идентификации функций; уровень идентификации потоков; уровень идентификации процессов (прикладных программ); самый низкий уровень прикладной программы представлен командами процессора.

Сделать вывод о том, что выполнение прикладной программы протекает без отклонений, можно только тогда, когда прикладная программа будет подвержена контролю на всех уровнях, иначе контроль будет неполным.

Например, если контролировать потоки процесса, используя только последовательность вызовов в контексте данного потока подпрограмм (функций), то остается уязвимым содержимое функций. То есть, если в рамках какой либо функции будет осуществлена подмена исполнимого кода или замена какой либо динамически присоединяемой библиотеки, то для такого способа контроля обнаружить этот факт будет невозможно.

СММ позволяют повысить степень контроля в таких случаях. Рассматривая ММ, для которых состояниями являются подпрограммы (функции), можно сказать следующее.

Для СММ потока в качестве состояния этой модели могут быть использованы подпрограммы (функции). Число этих подпрограмм (N) является конечным, оно определяет число состояний в модели.

Исходными данными для такой модели являются следующие.

Множество состояний в модели. Каждому состоянию соответствует библиотечная функция, принадлежащая данной программе и входящая в множество всех функций СВТ.

Множество наблюдаемых символов в этой модели – команды процессора.

Матрица переходных вероятностей показывает вероятность перехода от одной функции к другой.

Распределение вероятностей появления символов наблюдений (команд процессора) в каждом состоянии (функции).

Начальное распределение вероятностей состояний (функций).

Такое представление потока позволяет контролировать подпрограммы (функции) на уровне команд процессора. Наблюдая последовательность появления различных команд процессора и вычисляя вероятность их появления в данной последовательности для СММ, можно определить, какая подпрограмма выполняется в данный момент с наибольшей вероятностью. Если данная подпрограмма принадлежит множеству подпрограмм потока, то значит (с определенной долей вероятности), что в данный момент времени выполняется именно этот поток. В свою очередь, если поток принадлежит множеству потоков того процесса, который выполняется в данное время, делается вывод о легальной работе процесса.

Недостаток такого подхода заключается в том, что подпрограммы не следуют в потоке одна за другой. Как правило, между ними существует некоторый промежуток, заполненный кодом программы, из которой производится вызов подпрограмм. Если контролю подвергаются только подпрограммы, то участки кода между ними не контролируются.

Для решения этой проблемы можно использовать СММ, в которой в качестве состояний используются линейные участки кода программы.

Исходными данными для такой модели являются следующие.

Множество состояний в модели. Каждому состоянию соответствует линейный участок, принадлежащий данной программе и входящий в множество всех линейных участков.

Множество наблюдаемых символов в этой модели – команды процессора.

Матрица переходных вероятностей представляет вероятность перехода от одного линейного участка кода к другому.

Распределение вероятностей появления символов наблюдений (команд процессора) в каждом состоянии (линейном участке).

Начальное распределение вероятностей состояний (линейных участков).

Такой подход позволяет контролировать весь код программы, так как любая подпрограмма также представляет собой набор линейных участков кода.

Недостатком такого подхода является слишком большое число линейных участков и переменная длина участков (от одной команды до десятков), что затрудняет процесс контроля.

Следовательно, для обеспечения надежного контроля следует использовать комбинированные методы. Например, один уровень контроля должен быть представлен в виде СММ, где состояния – это функции, а наблюдения – команды. Другой уровень контроля – ММ, описывающая потоки переходными матрицами состояний, где в качестве состояний – команды.

Дополнительным уровнем контроля может быть рассмотренный выше уровень контроля подпрограмм. Так как для каждого потока существует индивидуальный набор подпрограмм с переходными вероятностями между ними, присущими только данному потоку, то такой контроль позволит увеличить вероятность правильного распознавания потока.

В связи с тем, что в составе потока выполняются вызываемые библиотечные функции, все команды процессора, принадлежащие одному потоку, можно рассматривать в разных контекстах. Команды, принадлежащие потоку и выполняемые вне контекста какой либо библиотечной функции, и команды, выполняемые в контексте библиотечных функций. Кроме того, наблюдаемым событием является сам факт вызова библиотечных функций. Таким образом, модель контроля "защищённости" прикладных программ должна быть многоуровневой. В этом случае можно достичь наибольшего эффекта в распознавании потоков.

Для полного контроля "защищённости" прикладных программ, необходимо обеспечить трёхуровневый контроль. Первый уровень контроля представлен ММ потока, в которой в качестве состояний – команды процессора. Второй уровень контроля представлен ММ потока, состояниями которой являются библиотечные функции. Наблюдаемым событием здесь является факт вызова библиотечной функции. Третий уровень – СММ потока с состояниями в виде подпрограмм (функций), в качестве наблюдений которой используются команды процессора.

Задача контроля "защищенности" прикладной программы заключается:

- в построении для каждой программы из множества Z ММ и СММ, соответствующих каждому из трех уровней контроля и задаваемых множеством возможных состояний, матрицей переходов, множеством векторов наблюдений в каждом состоянии;

- в получении очередного наблюдаемого события;

- в определении потока, для которого вероятность перехода из предыдущего состояния в текущее максимальна.

- в проверке соответствия полученного потока потоку, выполняемому в данный момент в ОС.

Таким образом, задача динамического контроля безопасности информации формально связана с проверкой соответствия предъявленной статистической выборки определенному эталону и отнесения в соответствии с некоторым решающим правилом к состоянию "обнаружен компьютерный вирус" или состоянию "компьютерный вирус не обнаружен". Исходя из этого, задача динамического контроля безопасности информации может быть отнесена к классу задач распознавания образов.

Литература

1. ***Соломон Д., Руссинович М.*** Внутреннее устройство Windows 2000. Мастер-класс / Пер. с англ. СПб.: Питер. М.: Издательско-торговый дом "Русская редакция", 2001. 752 с.

2. ***Ларионов А.М., Майоров С.А., Новиков Г.И.*** Вычислительные комплексы, системы и сети. Ленинград: Энергоатомиздат, 1987.

3. ***Гмурман В.Е.*** Теория вероятностей и математическая статистика. М.: Высшая школа, 2003.

4. ***Рабинер Л.Р.*** Скрытые Марковские модели и их применение в избранных приложениях при распознавании речи: Обзор. ТИИЭР, 1989. № 2.

5. ***Бронштейн И.Н., Семендяев К.А.*** Справочник по математике. М.: Наука, 1986.