УДК 004.942-056.57

ПРИМЕНЕНИЕ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ ПРОГРАММНЫХ СРЕДСТВ ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТИ

С. Д. Жилкин, начальник отдела программного обеспечения «Праймтек», аспирант Национального исследовательского ядерного университета МИФИ; zhilkin@primetech.ru

Ключевые слова: *компьютерная безопасность, модели поведения, аномалии.*

Введение. Для решения актуальных задач в сфере компьютерной безопасности, особенно таких, например, как борьба с утечками конфиденциальных данных, хорошо зарекомендовал себя подход, связанный с построением модели поведения программного обеспечения (ПО): он позволяет проанализировать дальнейшие действия ПО относительно построенной модели.

Модель поведения ПО. Чаще всего модель строится на основании работы ПО в режиме, который считается доверенным. Такую модель называют эталонной, поскольку с ее помощью можно выявлять отклонения поведения ПО. В зависимости от характеристик, заложенных в эталонную модель, отклонения могут показать недекларированные возможности (НДВ) ПО, превышение полномочий пользователем, вредоносные действия ПО, подмену ПО другим исполняемым файлом и т. д.

В настоящее время большинство из перечисленных отклонений выявляются с помощью сигнатурного, эвристического и других методов анализа. Так обнаруживаются выполнение вредоносного кода, заложенного в ПО, несанкционированный доступ к файлам и прочие отклонения от нормальной работы ПО. Выявлению НДВ уделяется значительно меньше внимания, а между тем реализация НДВ может не быть вредоносным кодом и, следовательно, выявление ее распространенными средствами защиты [1] затруднительно. Зачастую это требует наличия исходного текста ПО [2], что также не всегда возможно. Поскольку задача выявления НДВ до сих пор решена не до конца, при рассмотрении подходов к моделированию особое внимание следует уделить их применению прежде всего для дальнейшего обнаружения НДВ.

В данной статье представлены распространенные методы моделирования ПО, указаны их достоинства и недостатки, предложен способ усовершенствования этих методов, а также приведены результаты использования каждого метода для моделирования различных типов приложений.

Подходы к моделированию. О поведении ПО можно судить по его взаимодействию с операционной системой и ее ресурсами: обращению к жесткому диску, сетевым ресурсам, вызовам функций драйверов, работе с реестром и пр. [3]. Различные события, порождаемые таким взаимодействием, являются основной информацией, на базе которой строятся модели поведения. Рассмотрим способы моделирования поведения ПО.

Наиболее простой — *общая модель*, задание модели поведения набором некоторых характеристик. Такие характеристики чаще всего носят бинарный характер, принимают значения 1 или 0 в зависимости о того, была ли обнаружена данная характеристика. Таким образом, этот подход заключается в создании набора характеристик, разрешающих или запрещающих некоторые типы действий ПО. Если возникают события, противоречащие модели поведения, считается, что найдена аномалия поведения.

Преимущества подхода: простота применения, быстродействие и высокая эффективность для простых приложений или приложений, работающих по четкому циклическому алгоритму. Недостаток: для приложений с разветвленным алгоритмом предлагаемая модель является слишком общей и, допуская выполнение конкретного действия в целом, она будет упускать случаи, когда конкретное действие разрешено только в связке с другим.

Точная модель. Этот подход к моделированию поведения ПО заключается в создании модели из событий, которые отражают поведение ПО. Точная модель представляет собой некий список действий (он может быть упорядочен по времени), которые были произведены приложением за время построения модели и считаются легитимными. События внутри такого списка чаще всего содержат информацию в текстовом виде — это, например, пути к объектам файловой системы, имена ресурсов, ключи реестра и т. д. Само множество событий старается охватить все легитимные действия ПО. После того как модель создана, происходит анализ событий, порождаемых ПО. Если находятся события, которые не входят в состав модели поведения, считается, что найдена аномалия поведения. При этом дополнительно может производиться оценка важности события, которое не входит в состав модели, а также оценка таких событий — небольшое количество может расцениваться как норма.

Преимущества подхода: относительная простота применения; высокая эффективность для простых приложений или приложений, работающих по четкому циклическому алгоритму. Более точно заданная модель по сравнению с предыдущим методом позволяет в деталях описать взаимодействие ПО с ресурсами системы и точнее выявлять факт несанкционированного доступа. Недостатки: повышенные требования к вычислительным ресурсам из-за сравнения строковых значений вследствие отсутствия математического аппарата. Кроме того, модель должна содержать все легитимные действия ПО, и ее размеры могут быть очень большими. Отсюда вытекают повышенные требования к объемам оперативной памяти, в которой должны храниться модели. При отсутствии формально заданной политики безопасности модель не сможет содержать все легитимные действия пользователя. Наконец, как и в случае общей модели, точная модель рассматривает события поодиночке, а не в связке друг с другом, что позволяет произвести вредоносное действие, каждое событие которого само по себе легитимно.

Нейронная модель представляет собой нейронную сеть, которая не несет в себе явной информации о поведении процесса, как в случае первых двух моделей. По сути, нейронная модель является набором чисел — коэффициентов и смещений нейронов, описывающих связи сети. Сеть создается и обучается таким образом, чтобы по-

данный на ее вход профиль поведения процесса давал выход, близкий к единице. Профиль поведения, которое не соответствует заложенному в модель эталону, даст результат, близкий к нулю [4].

Главным преимуществом такого подхода является способность нейронной сети обучаться распознавать почерк процесса [5]. В данную модель не закладываются жесткие правила поведения процесса, что существенно упрощает создание модели процесса, не имеющего четкого алгоритма жизненного цикла. Еще одно преимущество — быстрота получения результата соответствия поведения процесса его нейронной модели — обусловлено тем, что в таких моделях не происходит сравнения множества строковых переменных, как в случае точной модели. Основной недостаток: высокие требования к ресурсам на этапе обучения сети, так как обучение производится методом обратного распространения ошибки, что подразумевает множество итераций калибровки параметров нейронной сети [5]. Также следует отметить, что достоверность результата нейронной сети существенно зависит от разнообразия параметров, отображаемых профилем поведения.

Построение модели поведения основывается на системе мониторинга, которая может получать информацию о работе пользователей и процессов системы из ее журнальных файлов или с более низкого уровня драйверов. Чем детальнее и подробнее информация, поставляемая системой мониторинга, тем точнее модели поведения.

Построение модели в общем случае подразумевает множество итераций запуска моделируемого ПО. Во время этих итераций следует выполнить весь доступный или разрешенный функционал данного ПО. На практике чаще всего некоторое время эксплуатируют ПО в рабочем режиме, одновременно сохраняя информацию от системы мониторинга о событиях, порождаемых данным ПО. После того как ПО хотя бы раз выполнило свои основные задачи, процесс сбора данных прекращается и строится модель поведения необходимого типа, т. е. модель содержит описание не всего доступного функционала моделируемого ПО, а только того, который используется. Таким образом, любые другие действия будут расцениваться как аномалия поведения. В этом заключается основное преимущество описания поведения процесса посредством системы мониторинга — не требуется никаких экспертных данных о внутреннем устройстве ПО. В общем случае можно сделать вывод, что большее число запусков ПО обеспечит построение более качественной модели поведения.

При построении точной модели данные от системы мониторинга записываются как есть в модель ПО. Выше уже отмечалось, что в таком виде модель поведения — просто набор разрешенных действий. Усложненная точная модель может также учитывать последовательность разрешенных действий.

В случае общей и нейронной моделей требуется некоторый математический аппарат конвертации сообщений от системы мониторинга в так называемые профили поведения. Профиль поведения, по сути, есть вектор координат, описывающий характеристики поведения ПО, проявившиеся за некоторый промежуток времени. Для общей модели такой вектор представляет собой набор 0 и 1 {0, 1, 1, …, 0}, в котором каждая позиция говорит о наличии определенной характеристики. Например, первая координата может показывать, запущено ли моделируемое ПО с правами администратора, а вторая — создавало ли ПО файлы во временной папке, и т. д. В начале моделирования такой вектор полностью состоит из нулей. Затем по мере обработки сообщений, получаемых от системы мониторинга, некоторым координатам может присваиваться значение единицы.

Нейронная модель предполагает более сложный подход к построению модели. Вектор поведения, в зависимости от выбранного базиса характеристик, включает в себя числовые значения, подобно общей модели. Они несут в себе более конкретную информацию, чем просто описания общих черт поведения ПО. Так, например, модель может содержать число обращений к системным файлам, различным классам сетевых сервисов и т. д. То есть профиль поведения в нейронной модели представляет собой набор координат вида {1; 0; 43; 185; … 4}, где размерность вектора определяется числом характеристик ПО, которые можно получить с помощью системы мониторинга. Нейронная модель, и это главное ее преимущество, ориентируется не на значения вектора, а на соотношения значений координат, а также на появление новых или, наоборот, исчезновение ожидаемых координат. Кроме того, сеть можно обучить таким образом, чтобы она различала поведение моделируемого и смежного по функционалу ПО, что обычно необходимо для определения приложений некоторого пакета программ (Microsoft Office, Open Office и т. п.).

Главный недостаток всех моделей в том, что в них весь жизненный цикл ПО рассматривается как нечто целое, поэтому для него создается одна модель поведения. Это дает положительные результаты в случае, если моделируемое ПО является строго алгоритмизированным. Однако, если поведение и продолжительность работы ПО меняются от запуска к запуску, этот подход показывает низкую эффективность. Так как к такому виду ПО относится все ПО, подразумевающее взаимодействие с пользователем, можно сделать вывод, что существующие на данный момент модели не применимы к большинству распространенных приложений.

Модификация модели поведения. Для выполнения задачи построения модели, отражающей поведение ПО на не определенных заранее жизненных циклах, требуется модификация моделей поведения, позволяющая выделить фазы работы и построить модель поведения для каждой из них. Этот механизм основывается на анализе так называемых односекундных профилей поведения (профилей, описывающих работу ПО за прошедшую секунду), с помощью которых выделяются фазы активности ПО. Подробнее эти методы описаны в [4].

Предложенный механизм разделения работы на фазы не зависит от выбранного типа модели поведения, поэтому применим к каждой из них. В результате модификации модель всего жизненного цикла ПО разбивается на ряд подмоделей следующих видов: 1) модель, описывающая запуск; 2) ряд моделей, описывающих специфические действия ПО; 3) модель, описывающая завершение работы.

Результаты модификации. Рассмотрим результаты экспериментов с модифицированными моделями. Цель экспериментов — определить, в каких ситуациях и насколько эффективно предложенные методы усовершенствования демонстрируют улучшение по сравнению с базовыми методами. Для экспериментов были выбраны 83 приложения системного и прикладного характера. Главным критерием отбора приложений было наличие открытого кода, что является необходимым условием для собственноручного внедрения разного рода НДВ, проявляющихся при определенных условиях.

Для каждого ПО последовательно проводились следующие этапы экспериментов: 1) построение немодифицированных моделей поведения трех типов — общей, точной и нейронной; 2) построение модифицированных моделей трех типов; 3) выполнение условия срабатывания заложенных НДВ; 4) оценка эффективности модифицированных моделей по сравнению с обычными при выявлении НДВ. Затем выбранное ПО дополнительно работало некоторое время в штатном режиме, не вызывая заложенные НДВ, — это требовалось для того, чтобы оценить число ложных срабатываний.

Разделим заложенные НДВ на два вида: прямые и маскирующиеся под действия ПО. К прямым относится выполнение кода, реализующее необходимый злоумышленнику функционал вне зависимости от функционала самого ПО. НДВ второго типа маскируются под действия ПО и сложны для выявления. Примером таких НДВ может являться реализация сетевой утечки данных только во время штатной сетевой активности ПО. В состав набора из 83 приложений входили следующие программы с открытым исходным кодом: пакет офисных приложений OpenOffice, клиент обмена сообщениями Pidgin, графический редактор GIMP, музыкальный сервер mpd, текстовый редактор Gedit, клиент почтовых сообщений Thunderbird, сетевой torrent-клиент Azureus, сетевые клиенты точка-точка Gnecleus, видеоплееры VLC, FTP-клиент Filezilla и др.

Система мониторинга поставляет информацию о взаимодействии приложений с файловой системой, другими процессами, сетевыми ресурсами, системным реестром, модулями аутентификации, принтерами и другими устройствами. С помощью такой системы для запущенного процесса было выделено 68 характеристик поведения ПО, приведенных в таблице.

В каждое ПО внедрялся ряд НДВ — как первого, так и второго типов. Ниже представлены результаты выявления внесенных НДВ для каждого типа ПО, а также статистика по ложным срабатываниям, выявленным в ходе работы.

На рис. 1—4 приведены результаты обнаружения внедренных НДВ первого и второго типов в ПО сетевого обмена информацией (рис. 1), в мультимедийном ПО (рис. 2), в офисных пакетах (рис. 3), в системных службах и сервисах (рис. 4), а также число ложных срабатываний для каждой модели. Белым цветом обозначен результат, достигаемый при использовании стандартной модели, темным — модифицированной.

Как видно из графиков, модифицирование в первую очередь помогает улучшить характеристики нейронной модели: число ложных срабатываний сокращается, вместе с тем увеличивается процент обнаружения НДВ первого и второго типов. Также модифицирование заметно улучшает показатели общей модели, в некоторых случаях немного увеличивая число ее ложных срабатываний, что объясняется неточным разделением работы ПО на фазы при построении модели, которое не совпало с разделением в режиме реальной работы.

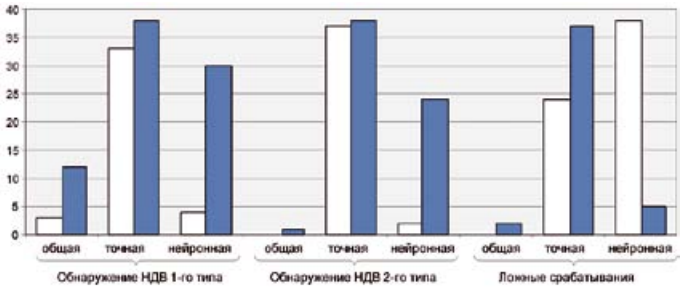

Рис. 1

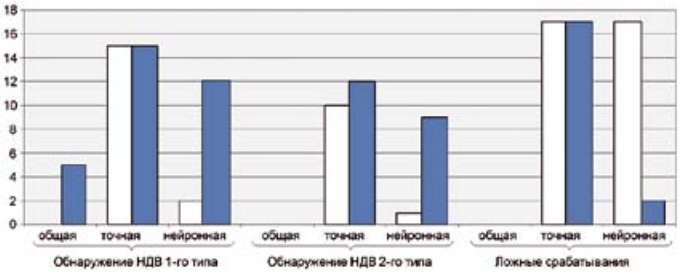

Рис. 2

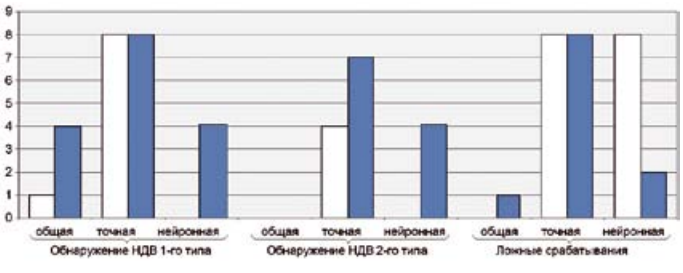

Рис. 3

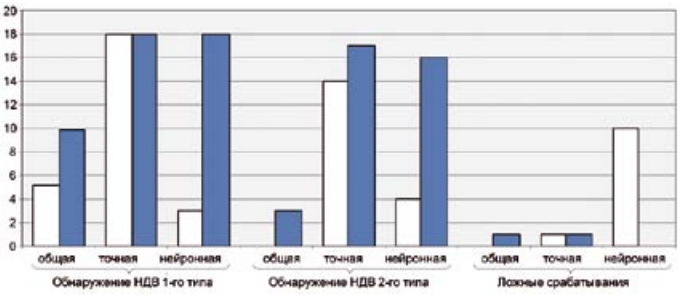

Рис. 4

Характеристики	Описание координат
1—20	Работа с файловой системой: системные файлы, пользовательские, сетевые, файлы приложений, съемных носителей
20—24	Работа с библиотеками (запись, открытие, линковка)
25—26	Работа с конфигурационными файлами
27—34	Работа с документами офисных пакетов и мультимедиа-файлами
35—41	Работа с системными ключами и значениями реестра: разделение таких веток, как hklm\software, hkcu\software и т. д.
42—51	Установление или принятие TCP- и UDP-соединений, передача данных
52—58	Запуск процессов от имени моделируемого ПО, отношения с другими процессами, характеристики процесса
59—68	Взаимодействие с системными устройствами

Сделаем выводы по каждой из рассмотренных моделей.

Общая модель. Даже будучи модифицированной, выявляет меньше половины НДВ первого типа. Крайне не эффективна для НДВ второго типа. Одинаково применима ко всем рассмотренным типам ПО. Показывает невысокий процент ложных срабатываний. Характеристики общей модели можно улучшить, если уйти от бинарных характеристик, составляющих модель, и использовать числовые переменные, чтобы точнее описать процесс.

Точная модель. Высокие показатели выявления НДВ являются отражением графика ложных срабатываний, т.е. это не обнаружения НДВ, как таковые, а ложные срабатывания, совпавшие с вызовом НДВ. Точная модель применима только к системным службам, работа которых не выходит за рамки определенных действий.

Нейронная модель. Эффективна только при модифицировании. Будучи модифицированной, показывает приемлемый уровень ложных срабатываний и достаточно высокий уровень обнаружения НДВ обоих типов. Применима ко всем рассмотренным типам ПО.

Предложенный подход к моделированию программных средств обеспечения компьютерной безопасности с разделением на фазы имеет ряд преимуществ и напрямую связанных с ними недостатков.

Преимущества модели поведения ПО:

- Предложенный подход реализует автоматизированное построение моделей, не требующее вмешательства оператора в процесс обучения и обсчета моделей.
- Моделирование охватывает весь жизненный цикл ПО, разделяя его на фазы работы. Так, обеспечивается отслеживание отклонений на протяжении всей работы ПО.
- Не требуются исходные тексты или экспертные знания о функционировании ПО. Для модуля моделирования любое ПО представляется «черным ящиком», о поведении которого можно судить только по внешним признакам.
- Значительно снижается процент ошибок первого рода у нейронной модели, а также повышаются показатели обнаружения НДВ.

Недостатки модели поведения ПО:

- Отсутствие наглядности в общей и нейронной моделях. Для оператора или администратора службы безопасности внутреннее устройство моделей не раскрывает, сколько и какие действия заложены в модель поведения. Как следствие, ручное редактирование модели невозможно.
- Хотя данный подход позволяет понять, какая последовательность действий привела к проявлению НДВ, он не локализует участок кода, содержащий реализацию НДВ.
- Выявленная аномалия не всегда указывает на НДВ. В частности, аномалия может быть признаком обнаружения вируса, подмены исполняемого файла, превышения полномочий и т.д. Точное определение характера НДВ невозможно с помощью одной конкретной модели. Чтобы отличать заражение ПО от срабатывания НДВ, необходимо совместное использование антивируса и предложенного подхода.

Заключение. Таковы общие черты подхода к разделению работы программных средств обеспечения компьютерной безопасности на фазы. Преимущества, недостатки и область применения данного подхода к существующим способам построения моделей поведения ПО зависят от характеристик, заложенных в модели поведения, а те, в свою очередь, от системы мониторинга и степени детализации информации, которую она может поставлять.

ЛИТЕРАТУРА

1. **Марков А.С., Миронов С.В., Цирлов В.Л.** Выявление уязвимостей в программном коде//Открытые системы.— 2005.— № 12.
2. Руководящий документ. Защита от несанкционированного доступа к информации. Ч. 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей.
3. **Темнов О.Д.** Анализ и исследование методов и средств обнаружения недекларированных возможностей: Науч.-техн. вестник Санкт-Петербургского госуниверситета ИТ, механики и оптики.— 2007 — № 39.
4. **Жилкин С.Д.** Методы построения моделей штатной работы ПО и алгоритмы выявления аномального поведения ПО//Безопасность информационных технологий.— 2009.— № 2.
5. **Swingler K.** Applying Neural Networks. A practical Guide.— London: Academic Press, 1996.

Получено 11.01.2010

ИНФОРМАЦИЯ

«УРАЛСВЯЗЬИНФОРМ» И HUAWEI СОЗДАЛИ ОПЫТНУЮ СЕТЬ ALL-IP

Компания Huawei и универсальный оператор связи «Уралсвязьинформ» подписали Меморандум о создании опытной сети All-IP (по технологии IMS).

IP Multimedia Subsystem (IMS, мультимедийная IP-подсистема) — комплексное решение, которое в перспективе призвано заменить все существующие сети электросвязи. Ядро сети по технологии IMS основано на коммутации пакетов и обеспечивает транзит (обмен) трафика, независимо от его происхождения (голос, мультимедийные файлы, видео). На «входе» в сеть, независимо от «последней мили» (фиксированной или беспроводной), любой трафик преобразуется в IP и затем система управляет потоками пакетов.

На опытной сети IMS планируется провести ряд испытаний, по результатам которых будет приниматься решение о строительстве коммерческой сети All-IP. В частности, будет протестировано взаимодействие ядра сети (CSCF) с оборудованием абонентского доступа мобильной сети, фиксированной сети и широкополосной сети передачи данных. Другой немаловажной задачей является проверка возможностей совместной работы элементов сети IMS Huawei и оборудования других поставщиков.

«Для нашей компании важно, что IMS умеет управлять трафиком вне зависимости от используемых абонентом устройств и «последней мили». Внедряя IMS «Уралсвязьинформ» может добиться реальной конвергенции собственных сетей фиксированной и мобильной телефонии, передачи данных. Таким образом, компания с одной стороны может существенно снизить операционные затраты за счет замены разнотипного и уже морально устаревшего оборудования на единое решение. С другой стороны, IMS позволяет предлагать абонентам конвергентные услуги и сервис нового поколения, то есть получать дополнительные доходы»,— сказал первый заместитель генерального директора ОАО «Уралсвязьинформ» **М. Крымский**.

«IMS-проект, реализуемый с «Уралсвязьинформом», является на сегодня самым масштабным в России и СНГ. Мы гордимся доверием, оказанным оператором нашей компании. Успешная реализация данного проекта ознаменует собой начало нового этапа в переходе операторов на конвергентные услуги»,— заявил **Ван Кэсян**, президент Huawei Russia.

Для реализации проекта компания Huawei планирует развернуть производство оборудования IMS на собственном предприятии в Уфе.