С. Д. Жилкин

ПРИМЕНЕНИЕ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Введение

В число актуальных задач сферы компьютерной безопасности уже продолжительное время входит задача построения модели поведения программного обеспечения (ПО) с целью анализа дальнейших действий ПО относительно построенной модели. Чаще всего модель строится на основании работы ПО в режиме, который считается доверенным. Такую модель называют эталонной моделью поведения. Относительно эталонной модели затем можно выявлять отклонения в поведении 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.

Следует отметить, что на данный момент большинство из перечисленных отклонений обнаруживаются с помощью сигнатурного, эвристического и других методов анализа. Так обнаруживается выполнение вредоносного кода, заложенного в ПО, несанкционированный доступ к файлам и прочие отклонения от нормальной работы ПО. Однако куда меньшее внимание уделено выявлению недекларированных возможностей. Следует иметь в виду, что реализация НДВ может не являться вредоносным кодом и, значит, не будет выявлена распространенными средствами защиты [1]. Также зачастую выявление НДВ требует наличия исходного текста ПО [2], что далеко не всегда возможно обеспечить. Поэтому задача выявления НДВ до сих пор является не до конца решенной, и при рассмотрении подходов к моделированию в данной статье особое внимание уделяется возможности применения рассматриваемых методов прежде всего для дальнейшего обнаружения НДВ.

Ниже представлены распространенные на сегодняшний день методы моделирования ПО, указаны их достоинства и недостатки. Предложен способ усовершенствования методов, а также приведены результаты использования каждого метода для моделирования различных типов приложений.

1. Подходы к моделированию

О поведении ПО можно судить по его взаимодействию с операционной системой и ее ресурсами: обращение к жесткому диску, сетевым ресурсам, вызовы функций драйверов, работа с реестром и прочее [3]. Различные события, порождаемые таким взаимодействием, являются основной информацией, на базе которой строятся модели поведения. На данный момент предложены и используются различные способы моделирования поведения ПО [4]. Рассмотрим первый и наиболее простой из них — *общая модель*, задание модели поведения набором некоторых характеристик. Такие характеристики носят чаще всего бинарный характер — принимают значения 1 или 0 в зависимости о того, наблюдалась ли данная характеристика. Таким образом, этот подход заключается в создании набора характеристик, разрешающих или запрещающих некоторые типы действий ПО. Если находятся события, которые противоречат модели поведения, считается, что найдена аномалия поведения. Преимущества такого подхода: простота применения, быстродействие и высокая эффективность для простых приложений или приложений, которые работают по четкому циклическому алгоритму. Недостаток: для приложений с разветвленным алгоритмом предлагаемая модель слишком общая и, допуская выполнение конкретного действия в целом, будет упускать случаи, когда конкретное действие разрешено только в связке с другим.

Второй подход к моделированию поведения ПО заключается в создании модели из событий, которые отражают поведение ПО. Такая модель называется *точной*. Точная модель представляет собой некоторый список (возможно, хронологически упорядоченный) действий, которые были произведены приложением за время построения модели и считаются легитимными. События внутри такого списка чаще всего содержат информацию в текстовом виде, например пути к объектам файловой системы, имена ресурсов, ключи реестра и т. д. Само множество событий старается покрыть все легитимные действия ПО. После того как модель создана, производится 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 количества таких событий — небольшое количество таких событий может расцениваться как норма. Преимущества такого подхода: относительная простота применения; высокая эффективность для простых приложений или приложений, которые работают 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.

Недостатками являются повышенные требования к вычислительным ресурсам из-за сравнения строковых значений вследствие отсутствия математического аппарата. Кроме того, так как модель должна содержать все легитимные действия ПО, ее размеры могут быть очень большими. Отсюда вытекают повышенные требования к объемам оперативной памяти, в которой должны храниться модели. В случае отсутствия 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.

Рассмотрим третий вид — *нейронную модель*. Данная модель является нейронной сетью и не несет в себе явной информации о поведении процесса, как это было в случае первых двух моделей. По сути, модель является набором чисел — коэффициентов и смещений нейронов, описывающих связи сети. Сеть создается и обучается таким образом, чтобы поданный на ее вход профиль поведения процесса давал выход, близкий к 1. Профиль поведения, которое не соответствует заложенному в модель эталону, даст результат, близкий к 0 [5].

Главным преимуществом такого подхода является особенность нейронной сети, позволяющая ей обучаться распознавать почерк процесса [6]. Таким образом, в модель не закладываются жесткие правила поведения процесса, что существенно упрощает задачу создания модели процесса, не имеющего четкого алгоритма жизненного цикла. Еще одним преимуществом является быстрота получения результата соответствия поведения процесса его нейронной модели в силу того, что в таких моделях не происходит сравнения множества строковых переменных, как в случае точной модели.

Основным недостатком являются высокие требования к ресурсам на этапе обучения сети, так как обучение производится методом обратного распространения ошибки, что подразумевает множество итераций калибровки параметров нейронной сети [6]. Также следует отметить, что достоверность результата нейронной сети очень сильно зависит от разнообразия параметров, отображаемых профилем поведения.

2. Построение модели поведения

Построение модели поведения основывается прежде всего на некоторой системе мониторинга. Система мониторинга может получать информацию о работе пользователей и процессов системы на основании журнальных файлов этой системы или с уровня ниже — уровня драйверов. Чем детальнее и подробнее информация, поставляемая системой мониторинга, тем точнее получаются 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 хоть раз выполнило свои основные задачи, процесс сбора данных прекращается и строится модель поведения необходимого типа. Тем самым модель содержит описание не всего доступного функционала моделируемого ПО, а только того, который используется. Таким образом, любые другие действия будут расценены 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 говорить, что большее число запусков ПО обеспечит построение более качественной модели поведения.

В случае построения точной модели данные от системы мониторинга записываются как есть в модель ПО. Как отмечалось выше, 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. По сути, профиль поведения есть некоторый 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 показывает наличие 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 1.

Нейронная модель подразумевает 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 включать в себя число обращений к системным файлам, различным классам сетевых сервисов и т. д. Т. е. профиль поведения в нейронной модели представляет собой набор координат вида {1; 0; 43; 185; … 4}, где размерность вектора определяется числом характеристик ПО, которое можно получить с помощью системы мониторинга. Главным преимуществом нейронной модели является то, что она ориентируется не на значения вектора, а на соотношения значений координат, а также на появление новых или, наоборот, исчезновение ожидаемых координат. Еще одним преимуществом является то, что сеть также можно обучить таким образом, чтобы она различала поведение моделируемого и смежного по функционалу ПО. Это часто необходимо для определения приложений некоторого пакета программ (Microsoft Office, Open Office и т. п.).

Главный недостаток всех моделей заключается в том, что они рассматривают жизненный цикл ПО как нечто целое и на весь жизненный цикл создается одна модель поведения. Это дает положительные результаты в случае, если моделируемое ПО является строго алгоритмизированным. Однако в случае, когда поведение и продолжительность работы ПО меняются от запуска к запуску, такой подход показывает низкую эффективность. Так как к такому виду ПО относится всё ПО, подразумевающее взаимодействие с пользователем, становится очевидно, что существующие на данный момент модели неприменимы к большинству распространенных приложений.

3. 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 выделенной фазы. 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 другой статье на данную тему [6]. Предложенный механизм разделения работы на фазы не зависит от выбранного типа модели поведения и 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, и 3) модель, описывающая завершение работы.

4. Результаты модификации

В данном разделе представлены результаты серии экспериментов с модифицированными моделями рассмотренных типов. Эксперименты проводились с целью определить, в каких ситуациях и насколько эффективно предложенные методы усовершенствования показывают улучшения по сравнению с базовыми методами. Для серии экспериментов был выбран ряд ПО, как системного, так и прикладного — всего 83 приложения. Главным критерием выбора приложений являлось наличие открытого кода. Это было необходимо для собственноручного внедрения разного рода НДВ, проявляющихся при определенных условиях. Для каждого ПО последовательно проводились следующие этапы: 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, для того чтобы оценить число ложных срабатываний. Заложенные НДВ разделим на два вида: прямые и маскирующиеся под действия ПО. К прямым относится выполнение кода, реализующее некоторый необходимый злоумышленнику функционал вне зависимости от функционала самого ПО. НДВ второго типа маскируются под действия ПО и сложны для выявления. Примером такого 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 Open Office, 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. Итого с помощью такой системы для запущенного процесса было выделено 68 характеристик (таблица 1).

Таблица 1. Характеристики поведения ПО

№ хар-ки	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

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.

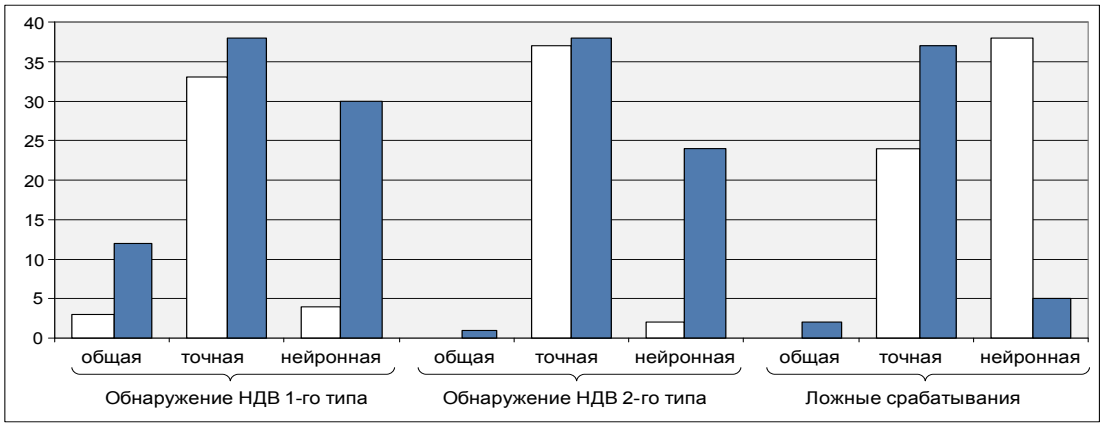

Рис. 1. Обнаружение внедренных НДВ в ПО сетевого обмена информацией

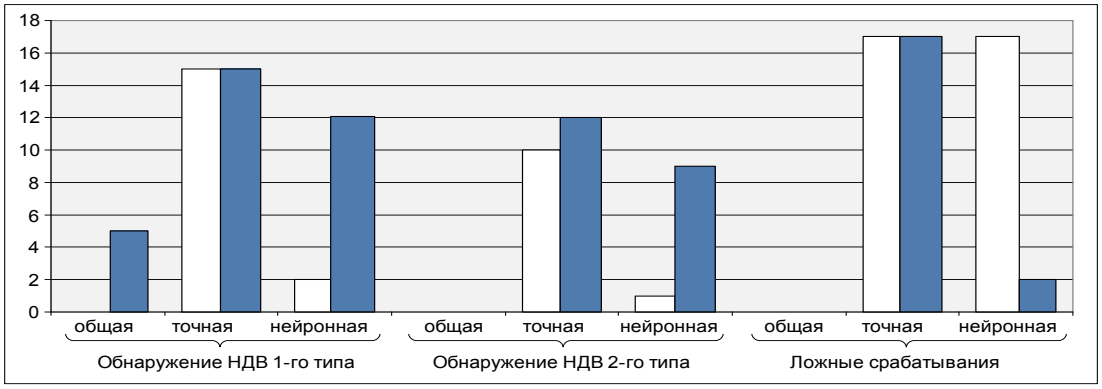

Рис. 2. Обнаружение внедренных НДВ в мультимедийном ПО

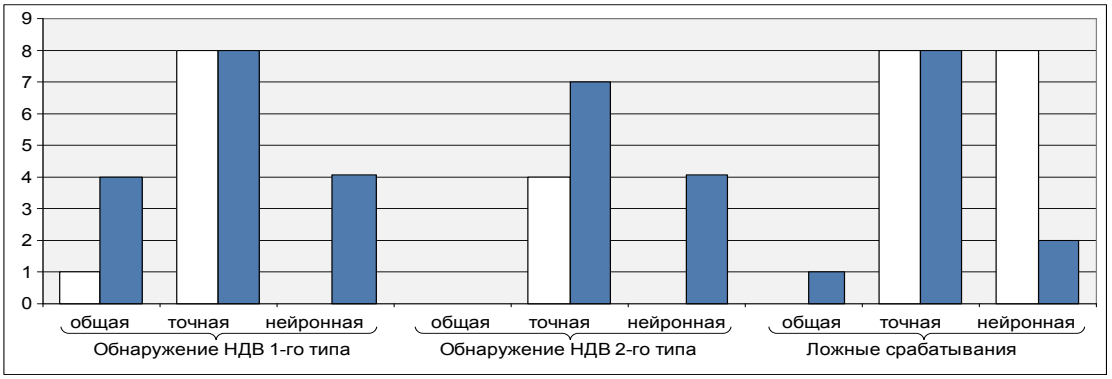

Рис. 3. Обнаружения внедренных НДВ в офисных пакетах

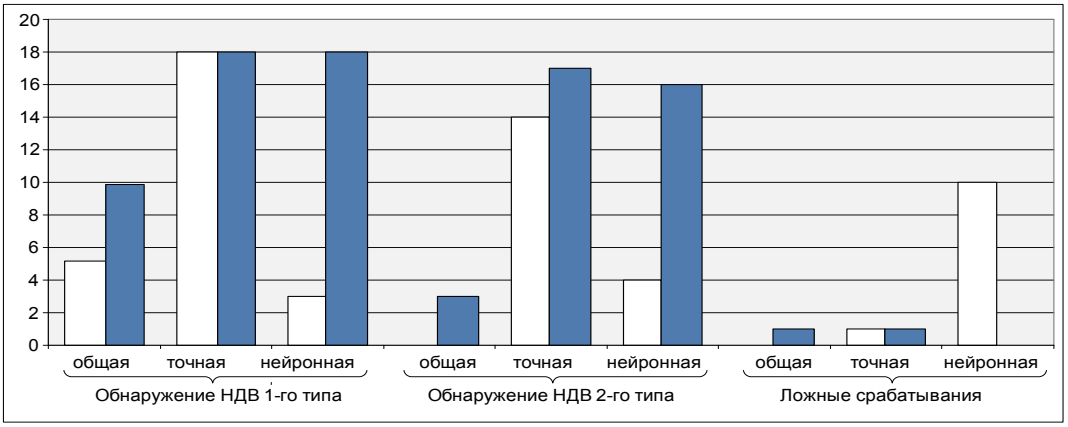

Рис. 4. Обнаружение внедренных НДВ в системных службах и сервисах

На рис. 1–4 представлены результаты обнаружения внедренных НДВ первого и второго типов с помощью различных моделей, а также число ложных срабатываний для каждой модели. Белым обозначен результат, достигаемый при использовании стандартной модели, темным — модифицированной.

Как видно из графиков, модифицирование помогает, прежде всего, улучшить характеристики нейронной модели: число ложных срабатываний сокращается, вместе с тем увеличивается процент обнаружения НДВ как первого, так и второго типов. Также модифицирование заметно улучшает показатели общей модели, в некоторых случаях немного увеличивая число ее ложных срабатываний. Увеличение ложных срабатываний при модифицировании объясняется неточным разделением работы ПО на фазы при построении модели, которое не совпало с разделением в режиме реальной работы. В целом, выводы, которые можно сделать о каждой из рассмотренных моделей, представлены ниже.

Общая модель. Даже будучи модифицированной, эта модель выявляет меньше половины НДВ первого типа. Крайне неэффективна 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. Эта 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 с разделением на фазы имеет ряд преимуществ и напрямую связанных с ними недостатков. Преимущества:

· Предложенный подход реализует автоматизированное построение моделей, не требующее вмешательства оператора в процесс обучения и обсчета моделей.

· Моделирование покрывает 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 снижает процент ошибок первого рода у нейронной модели, а также повышает показатели обнаружения НДВ.

Недостатки:

· Отсутствие наглядности в общей и нейронной моделях. Для оператора или администратора безопасности внутреннее устройство моделей не дает понять, сколько и какие действия заложены в модель поведения. Как следствие, невозможно ручное редактирование модели.

· Несмотря на то что 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. Необходимо совместное использование антивируса и предложенного подхода, для того чтобы отличать заражение ПО от срабатывания НДВ.

Заключение

В данной статье представлены общие черты подхода к разделению работы ПО на фазы, а также показаны результаты применения этого подхода к существующим способам построения моделей поведения ПО. Показаны преимущества, недостатки и область применения такого метода. Следует отметить, что результаты, представленные в статье, прежде всего зависят от характеристик, заложенных в модели поведения, которые, в свою очередь, зависят от используемой системы мониторинга и подробности той информации, которую она может поставлять.

СПИСОК ЛИТЕРАТУРЫ:

1. *Марков А. С., Миронов С. В., Цирлов В. Л.* Выявление уязвимостей в программном коде // Открытые системы. 2005. № 12. URL: http://www.osp.ru/os/2005/12/380655.

2. Руководящий документ «Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей» от 4 июня 1999 г. № 114.

3. *Темнов О. Д.* Анализ и исследование методов и средств обнаружения недекларированных возможностей // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2007. № 39. С. 45–40.

4. Круглый стол «Вестника Связи» // Вестник Связи. 2006. № 12. С. 4–18.

5. *Жилкин С. Д.* Методы построения моделей штатной работы ПО и алгоритмы выявления аномального поведения ПО // Безопасность информационных технологий. 2009. № 2. С. 59–66.

6. *Swingler K.* Applying Neural Networks. A Practical Guide. Morgan Kaufmann; Pap/Dsk edition. 1996.