Математические структуры и моделирование
2013. № 1(27). С. 103–112

УДК 004.492.3

БИХЕВИОРИСТИЧЕСКАЯ ИДЕНТИФИКАЦИЯ ПРОЦЕССОВ

Р.С. Прохоров

В данной статье рассмотрен подход, позволяющий осуществить идентификацию процесса по его поведению. Подход основан на преобразовании потока событий в поток контейнеров событий и использовании такого потока для подачи на вход интеллектуального анализатора, построенного с помощью мультинейронной сети. В качестве «ядра» анализатора используется принципиально новая структура композитной нейронной сети, названная мультинейронной сетью. Подробно описаны варианты используемых преобразований, а также сами используемые преобразования.

Введение

Проблема идентификации процессов возникает в широком классе задач защиты информации, таких как борьба с вредоносным программным кодом, построение замкнутой программной среды, политика ограниченного использования программ и другие. На сегодняшний день наибольшее распространение получили подходы, основанные на анализе исполняемого файла, либо анализе непосредственно исполняемого кода [1]. Идентификация процесса может производиться по имени исполняемого файла, хэш-значении исполняемого кода или сигнатуры исполняемого кода. Все эти методы могут быть преодолены злоумышленником с помощью преобразований исполняемого кода, не влияющих на его функциональность. Несмотря на то, что разрабатываются все новые и новые алгоритмы сигнатурного анализа [2–4], всегда можно создать специальный алгоритм мутации, обходящий свежие системы защиты. В связи с этим остаётся актуальной задача разработки технологии идентификации процесса по некоторому набору признаков.

Как легко понять, проблема идентификации процессов относится к задачам распознавания образов и к ней могут быть применены соответствующие технологии искусственного интеллекта. Главная проблема, с которой приходится сталкиваться, состоит в формировании «образа» процесса, который, с одной стороны, однозначно его характеризует, а с другой стороны устойчив к воздействиям исполняемого кода, не изменяющим функционал процесса.

Омский государственный университет им. Ф.М. Достоевского
E-mail: roman.evlikat.prokhorov@gmail.com

Бихевиористический анализ является частью иммунной системы, построенной в работе [5], где он дополняется сигнатурным анализом. В работе [6] предлагается архитектура нейросетевой системы анализа поведения процессов операционной системы реального времени, используемых в специализированных встраиваемых системах. В работах [7, 8] описано формирование образа поведения процесса по особой структуре, названного свойством (feature). Принцип этого формирования заключается в повторении присущих процессу свойств.

Идея, лежащая в основе бихевиористического анализа: субъект может выполнять только такие действия, которые ему предписаны. Значит, имея набор событий, связанных с каким-либо субъектом, мы можем попытаться определить, чем является этот субъект.

Следует отметить, что задача идентификации процесса является задачей распознавания образов. Здесь можно выделить два типа образов: статический и динамический. Статический образ имеет все признаки «здесь и сейчас», ему свойственна полнота набора признаков — исследуемая сущность не может измениться, в противном случае это будет уже другая сущность. Например, задача распознавания образов на картинке. Динамический же образ растянут во времени, ему свойственна потоковая природа формирования — рождение в процессе.

Таким образом, необходимо учитывать особенности формирования образа субъекта. Такой образ также будет устойчив к нефункциональному изменению тела субъекта — изменению пассивной части, то есть части, не влияющей на поведение объекта.

Для дальнейшего изложения необходимо выделить несколько основных понятий.

Образом субъекта назовём след, оставленный его поведением в среде, для процесса — это события в компьютерной системе.

Среда — сущность, на которую воздействует субъект, которая преобразуется под его влиянием, для процесса — операционная система.

Поведение — это поток действий, для процесса — поток событий, которые произошли под влиянием данного процесса.

1. Анализ бихевиористических характеристик

Будем считать поведение — потоком событий, имеющим неопределённую длину. Пусть E (events) — множество всевозможных событий, которые могут произойти в рассматриваемой среде. Тогда для каждого субъекта, функционирующего в данной среде за время Δt, можно определить его поведение $b = (e_1, e_2, \ldots, e_n)$, где $e \in E$.

Для каждого события, произошедшего в среде, возможно определить его тип, субъект, который его породил, а также некоторые данные, присущие конкретному типу событий. Поток событий разделяется на группы так, что каждой группе соответствует субъект, породивший все события из данной группы. Необходимо присвоить каждому субъекту архетип, анализируя только поток

связанных с ним событий. В результате это сводится к задаче классификации с динамическим формированием образов субъектов.

В рассматриваемых средах функционирует некоторое конечное число субъектов. Каждый из них может породить только подмножество множества всех возможных событий, обозначим его $\overline{E}$. Тогда по мощности данного множества возможно следующее разделение сред: тривиальные, для которых $|\overline{E}| = 1$, квазидетерминированные, реальные, свободные, хаотичные, где для каждого процесса $\overline{E} = E$.

Задача идентификации субъекта в тривиальной системе имеет очевидное решение: взаимооднозначное отображение множества событий во множество архетипов субъектов. Для хаотичной системы — наоборот. Задача неразрешима, поскольку любой субъект может породить любое событие.

Квазидетерминированной системой назовём среду, в которой субъект сгенерирует хотя бы по одному событию из большинства присущих ему классов. Реальные среды — среды, в которых субъект не может раскрыть своё поведение за один интервал времени. Свободной средой назовём среду с изменяющимся субъектом. В данной статье рассматривается функционирование субъектов в *реальных* средах.

2. Формирование бихевиористического образа процесса

Рассмотрим подробнее поток событий, ассоциированный с субъектом. Пусть $a = (a_1, a_2, \ldots, a_n)$ — последовательность событий, где a_i — действие субъекта, причём для каждого a_i существует класс события $e \in E$. Назовём такую последовательность *поведением субъекта* или *поведением*. Поведение субъекта за всё время слежения назовём *глобальным*. Любую её подпоследовательность назовём *локальным* поведением.

Для формирования образа субъекта используется функция разбиения глобального поведения на набор локальных поведений. Этот набор обладает рядом следующих свойств:

1. События внутри подмножеств сохраняют порядок (поведение — последовательность событий).

2. Объединение всех локальных поведений есть глобальное поведение.

3. Одно событие может принадлежать только одному локальному поведению из всего набора.

Из данных свойств следует, что ни одно событие не будет пропущено, что позволило наиболее точно сформировать образ исследуемого субъекта у анализатора. Таким образом, с каждым новым локальным поведением, поступающим на вход анализатора, формируется образ субъекта.

3. Идентификация процесса по бихевиористическому образу

Ядром системы бихевиористического анализа является *анализатор*. Задача обработчика — определять, имеет ли субъект конкретный архетип. Такая реализация схожа с реальной иммунной системой человека, где каждое антитело реагирует только на антигены, которые попадают в радиус стимуляции. В данной статье описан детектор, который умеет отличать только антигены конкретного архетипа.

Если представить анализатор в виде «чёрного ящика», то можно определить входы и выходы.

Вход — последовательность событий за некий промежуток времени (локальное поведение).

Выход — имеет ли процесс искомый архетип или нет.

Следует отметить, что внутри такого «чёрного ящика» может быть сколь угодно сложная интеллектуальная структура.

Каждое локальное поведение преобразуется с помощью следующей функции:

$$f : B \to V. \tag{1}$$

Здесь B — множество локальных поведений, а V — множество входных векторов. Большинство интеллектуальных структур принимают на вход вектор в том или ином виде, поэтому и был выбран данный интерфейс. В качестве функции, осуществляющей преобразование (1), выбран *количественный показатель*. Для этого использован вектор длиной $m = |E|$: каждый из элементов соответствует одному классу событий, отражая количество встреченных событий такого класса в данном контейнере.

В реальных средах, особенно компьютерных системах, субъектам свойственно порождать длинные последовательности повторяющихся событий, что при использовании количественного показателя может вернуть входной вектор с большим числом нулевых значений, например $(0, 0, 0, 0, 18, 0, 0, 0, 0, 1, 0, 0)$. Как известно, эффективность любой нейронной сети стремительно падает при работе с такими входными векторами: результат просто «забивается» нулевыми значениями и получается, что выходной вектор практически не несёт полезной информации. Данная особенность сильно затрудняет традиционное использование нейронных сетей при таком подходе в данной задаче.

Для предотвращения такого эффекта необходимо сокращать число входов и обрабатывать только ненулевые части. Были рассмотрены следующие методы: группировка, трансляция и фильтрация.

Группировка — метод, при котором входной вектор разбивается на несколько векторов меньшего размера так, что сумма их длин остаётся равной длине первоначального вектора. Далее необходимо разделить множество событий на группы, соответствующие определённым классам действий. Сгруппировав их, получим объединение непересекающихся множеств.

Трансляция заключается в формировании множества надклассов событий. Каждый класс событий имеет некий образ — надкласс. Было использовано

следующее сюръективное отображение:

$$f : E \to S. \tag{2}$$

Здесь S — множество обобщённых или укрупнённых классов событий.

Фильтрация — данный метод «отсеивает» лишние классы событий, чтобы сконцентрироваться на наиболее значимых из них. Было установлено, что данный метод в решаемой задаче будет искажать результат — нельзя игнорировать какие-либо классы событий, потому что субъект, который порождает только такие события, будет иметь нулевой архетип.

Самым эффективным методом является группировка, поскольку она может сократить число входов на порядок. Очевидно, что для группировки необходимо использование нескольких дочерних обработчиков, каждый из которых работает с небольшой группой классов событий. В результате мы имеем вектор ответов «да/нет». Для получения финального вектора необходимо разработать особую интеллектуальную структуру с подуровнями, которая способна принимать финальное решение. Назовём такую структуру *мультинейронной сетью* — сетью, состоящей из множества «метанейронов» — других нейронных сетей. Для дополнительного сокращения размерности вектора была добавлена трансляция параметров внутри каждой группы. В методах сокращения числа входов была использована *частичная фильтрация*: был выделен дополнительный надкласс «другие события», куда отображались малозначимые или схожие события.

4. Мультинейронная сеть

В современном мире существует масса примеров, когда какую-то сущность можно рассмотреть как набор таких же сущностей. Она может быть представлена в виде этого набора или быть этим набором. Такой подход при рассмотрении того или иного объекта применяется обычно для того, чтобы ограничить охватываемую область. Это очень сильно способствует пониманию, как работает объект. Идея взаимодействия схожих сущностей и «советников» была заимствована у Кеннеди и Эберхарта [9].

Интеллектуальный анализатор был разбит на малые анализаторы, причём каждый из них является таким же самостоятельным анализатором, но отвечающим на вопросы более узкого профиля. Также был установлен интерфейс каждого анализатора. Далее, для каждой группы анализаторов сформирован модуль разбиения входов и модуль согласования выходов.

Модуль разбиения входа отвечает за преобразование, разбивающее исходный вектор на множество векторов, что является функцией преобразования множества входных векторов во множество наборов входных векторов меньшей размерности $f : V \to \overline{V}^n$, где n — число нейронных подсетей.

Модуль согласования выхода отвечает за преобразование $f : P \to \overline{P}^n$, где n — число компонентов, а P — множество выходных векторов. Такой модуль формирует финальный результат. Следует отметить, что каждый из компонентов может являться как простой нейронной сетью, так и мультинейронной.

Мультинейронная сеть была реализована на основе нижеописанных компонентов.

Атомарная нейронная сеть N — простая неделимая нейронная сеть, то есть не являющаяся мультинейронной, на N входов. Каждая из таких сетей имеет два выхода.

Модуль разбиения входа преобразует одиночный контейнер в набор пар «группа классов — входной вектор».

Транслятор надклассов событий преобразует конкретный класс события в укрупнённый надкласс событий. Выполняет отображение (2).

Индексное хранилище нейронных подсетей содержит набор пар «группа классов — нейронная подсеть». Для обеспечения быстродействия используется индекс, построенный на основе бинарного дерева сортировки. С помощью индекса осуществляется быстрый поиск нужной подсети. Время поиска не превышает $\log N$.

Индексное хранилище учителей для нейронных подсетей содержит набор пар «группа классов — учитель нейронной подсети». Для обеспечения быстродействия также используется индекс, построенный на основе бинарного дерева сортировки. С помощью индекса осуществляется быстрый поиск нужного учителя. Время поиска не превышает $\log N$.

Модуль согласования выхода содержит информацию об успешности каждой нейронной подсети: процент правильно определённых процессов. Принимает на вход вектор ответов «да/нет». Выносит финальный вердикт о породившем контейнер субъекте. Использует схему «уровень доверия подсети», чтобы получить финальный результат.

Функция уровня доверия подсети. Пусть мы имеем n двумерных векторов P. Представим их как два вектора R_1 и R_2. $R_i = (p_{i_1}, p_{i_2}, ..., p_{i_n})$

Рассмотрим *вектор доверия* $T = (t_1, t_2, t_n)$. Данный вектор формируется в процессе тестирования сети, после её обучения. Тогда финальный результат вычисляется по формуле

$$Result = \begin{cases} yes, & res > 0; \\ no, & res < 0; \\ not\ defined, & res = 0; \end{cases}$$

где $res = (R_1, T) - (R_2, T)$.

Вектор T формируется следующим образом. Каждый раз, когда конкретная подсеть даёт результат, и мы можем определить, правильный ли он или нет, в реестр успехов вносится соответствующая запись:

1. Для каждой подсети хранятся два значения: число успешных идентификаций s и общее число идентификаций n.

2. Если идентификация была проведена верно, то для подсети, которая провела эту идентификацию, на единицу увеличиваются значения s и n.

3. Если подсеть ошиблась, то для неё увеличивается общее число идентификаций (n).

В любое время мы можем получить уровень доверия конкретной нейронной подсети:

$$t_i = \begin{cases} \frac{s_i}{n_i}, & n_i > 0; \\ 0, & n_i = 0. \end{cases}$$

Для формирования вектора уровня доверия необходимо два этапа обучения:

1. Обучение нейронной сети. На этом этапе формируются весовые коэффициенты самих нейронных подсетей.

2. Тестирование нейронной сети. Сеть тестируется и каждому компоненту (подсети) выставляется соответствующее значение уровня доверия.

Все вышеописанные компоненты объединяются в анализатор. Обучение мультинейронной сети происходит следующим образом:

1. Входной вектор разбивается на набор n малых входных векторов с помощью модуля разбиения выхода.

2. Для каждого малого вектора события, содержащиеся в нем, транслируются в события укрупнённых классов с помощью транслятора надклассов событий.

3. Каждый из малых входных векторов подаётся на вход соответствующему учителю нейронной сети из индексного хранилища.

Вычисление результата с помощью обученной сети:

1. Входной вектор разбивается на набор n малых входных векторов с помощью модуля разбиения выхода.

2. Для каждого малого вектора события, содержащиеся в нем, транслируются в события укрупнённых классов с помощью транслятора надклассов событий.

3. Каждый из малых входных векторов подаётся на вход соответствующей нейронной подсети из индексного хранилища.

4. Каждая нейронная подсеть вычисляет результат.

5. Набор n ответов нейронных подсетей подаётся на вход модулю согласования выхода.

6. Модуль согласования выхода выдаёт финальный результат.

Таблица 1. Соотношение «сырых» и укрупнённых классов событий по группам

Название	Классы событий (C_i)	Классы событий (S_i)
Дисковые операции	11	3
Сетевая активность	26	5
Файловая активность	18	5
Операции с реестром	19	4

5. Использование мультинейронной сети

Как отмечалось, данная сеть лишена недостатка множества лишних нулевых параметров, но нуждается в более тонкой настройке, а именно, формировании групп параметров. Эффективность действия мультинейронной сети напрямую зависит от удачного распределения параметров по группам.

Было определено входное преобразование контейнера во множество малых векторов. Некоторые эффективные методы отбора данных представлены в работах [10, 11]. В данной статье, как отмечалось выше, предлагается использование группировки, трансляции и частичной фильтрации для формирования таких множеств.

Группировка. Основные рассматриваемые операции процессов поделены на четыре группы классов: дисковые операции, сетевая активность процесса, файловая активность и операции с реестром. Классы событий, составляющие каждую из этих групп, будут подробно рассмотрены далее. Для фиксирования событий в операционной системе Windows 7 была использована библиотека с открытым исходным кодом Event Tracing for Windows.

Трансляция. В каждой из групп классов событий было выделено до пяти надклассов.

Частичная фильтрация. Частичная фильтрация проявляется в добавлении «неважных» надклассов событий в каждую группу.

На основе вышеописанного построена таблица 1.

Таким образом, получено четыре трёхслойных (с одним скрытым слоем) нейронных подсети: персептроны $3-3-2$, $5-5-2$, $5-5-2$ и $4-4-2$. В качестве функции активации используется сигмовидная функция. Каждая из них имеет два выхода: уверенность в ответе «да» и уверенность в ответе «нет».

6. Результаты

В таблице 2 показана эффективность применяемого метода к некоторым процессам после тестирования и после проверки на финальном множестве. Как показывают результаты исследования, данный подход крайне эффективен для «редких» процессов со специфическим поведением и средне эффективен для «популярных» процессов с довольно широким спектром порождаемых событий.

Таблица 2. Соотношение «сырых» и укрупнённых классов событий по группам

Процесс	Тестирование			Проверка		
	Успешность		%	Успешность		%
explorer	4037	6219	64,91%	4105	6206	66,15%
System	6215	6318	98,37%	5006	6300	79,46%
chrome	4820	6348	75,93%	5068	6201	81,73%
svchost	5636	6178	91,23%	5726	6309	90,76%
ekrn	5717	6215	91,99%	5805	6355	91,35%
wmagent	5755	6274	91,73%	5676	6147	92,34%
vmware-authd	5928	6234	95,09%	6032	6309	95,61%
PDFCreator	5950	6207	95,86%	6024	6285	95,85%
Dropbox	6035	6193	97,45%	6124	6277	97,56%
Skype	6091	6233	97,72%	6198	6329	97,93%
BisonHK	6130	6231	98,38%	6145	6267	98,05%
audiodg	6229	6285	99,11%	6244	6293	99,22%
QipGuard	6197	6236	99,37%	6280	6311	99,51%
wmpnetwk	6299	6310	99,83%	6119	6128	99,85%
SynTPEnh	6264	6271	99,89%	6278	6287	99,86%
services	6208	6214	99,90%	6194	6202	99,87%
NeuroIncinerateGUI	6477	6487	99,85%	6135	6142	99,89%
PrnStatusMX	6251	6255	99,94%	6220	6227	99,89%
VCSExpress	6288	6294	99,90%	6262	6267	99,92%

Выводы

В данной статье был рассмотрен подход, позволяющий осуществить идентификацию процесса по его поведению. Эффективность колеблется в районе 0,66–0,99. Так как вредоносное программное обеспечение зачастую является сугубо специфичным, то такая система должна достаточно эффективно определять архетип данных процессов. Преодолена проблема входных векторов для нейронных сетей, почти полностью заполненных нулями, с помощью мультинейронной сети.

Литература

1. Булахов Н.Г., Калайда В.Т. Методы обнаружения и обезвреживания саморазмножающихся вирусов // Доклады ТУСУРа. Июнь 2008. № 2(18). Ч. 1. С. 78–82.
2. Christodorescu M., Jha S., Seshia S., Song D., Bryant R. Semantics-aware malware detection // The IEEE Symposiumon Security and Privacy. 2005.
3. Jha M.C. a. S. Static analysis of executables to detect malicious patterns // USENIX Security Symposium. 2003.
4. Dai R.G. a. J.L.J. Efficient Virus Detection Using Dynamic Instruction Sequences // Journal of Computers. 2009.
5. Ваганов М.Ю. Гибридная искусственная иммунная система защиты компьютера от процессов с аномальной активностью: диссертация кандидата технических наук: 05.13.19. Санкт-Петербург, 2012. 92 с.
6. Валеев С.С., Дьяконов М.Ю. Нейросетевая система анализа аномального поведения вычислительных процессов в микроядерной операционной системе // Вестник УГАТУ. 2012. Т. 14, № 5(40). С. 198–204.
7. Lo D., Cheng H., Han J., Khoo S., Sun C. Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach. // KDD'09, Июнь 28 – Июль 1, 2009. Париж, Франция.
8. Ahmadi M., Sami A., Rahimi H., Yadegari B. Iterative System Call Patterns Blow the Malware Cover // Security for The Next Generation. 2011.
9. Kennedy J., Eberhart R.C. Swarm Intelligence. Academic Press, 2001. 512 c.
10. Bourlard H., Kamp Y. Auto-association by multilayer perceptrons and singular value decomposition // Biological Cybernetics. 1988. N. 59. P. 291–294.
11. Oja E. A simplified neuron model as a principal component analyzer // Journal of Mathematical Biology. 1982. N. 15. P. 267–273.