НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
сентябрь–октябрь 2016 Том 16 № 5 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
September–October 2016 Vol. 16 No 5 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.056.57

ОЦЕНКА ТОЧНОСТИ АЛГОРИТМА РАСПОЗНАВАНИЯ ВРЕДОНОСНЫХ ПРОГРАММ НА ОСНОВЕ ПОИСКА АНОМАЛИЙ В РАБОТЕ ПРОЦЕССОВ

М.В. Баклановский[a,b], А.Р. Ханов[a], К.М. Комаров[b], П.А. Лозов[b]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
[b] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация
Адрес для переписки: awengar@gmail.com

Информация о статье
Поступила в редакцию 09.06.16, принята к печати 19.07.16
doi: 10.17586/2226-1494-2016-16-5-823-830
Язык статьи – русский

Ссылка для цитирования: Баклановский М.В., Ханов А.Р., Комаров К.М., Лозов П.А. Оценка точности алгоритма распознавания вредоносных программ на основе поиска аномалий в работе процессов // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 5. С. 823–830. doi: 10.17586/2226-1494-2016-16-5-823-830

Аннотация

Предмет исследования. Предложен алгоритм, позволяющий находить аномалии в поведении процессов операционной системы, вызванные исполнением ранее неизвестных участков кода программы. Алгоритм реализован в рамках системы обнаружения и подавления вредоносных программ КОДА. Предложена методика тестирования алгоритма, позволяющая уменьшить время тестирования и повысить его точность. **Метод.** Предложенный метод основан на построении модели здорового процесса по последовательностям системных вызовов, совершенных его потоками. Выбраны оценки схожести между произвольным процессом и моделью, которые позволяют свести задачу поиска аномалий к задаче классификации векторов. Для оценки точности алгоритма предложено оценивать точность классификатора методом перекрестной проверки. В качестве классификатора была выбрана нейронная сеть типа персептрон. **Основные результаты.** Разработана и реализована платформа для массового распределенного запуска экземпляров вредоносных программ в виртуальных машинах. Платформа реализована с использованием открытой библиотеки для организации распределенных вычислений BOINC. В качестве базы для тестирования использована академическая база вредоносных экземпляров, а также открытая база Malwr, содержащие 60 тысяч вредоносных программ. Из общей базы корректно отработали 33,13% программ. Составлена модель легитимных процессов, запущенных в течение получаса. Оценки поведения вредоносных программ в рамках этой модели записаны как вектора. Для классификации этих векторов произведен поиск наилучшей по точности нейронной сети. На основе перебора нейронных сетей с различными параметрами обучения и количеством нейронов на скрытом слое выбран наилучший классификатор, точность которого составила 91%. **Практическая значимость.** Результаты работы могут быть полезны при обнаружении вредоносных программ. Для работы алгоритма не требуется подключение к сети Интернет. Алгоритм позволяет находить как известные, так и новые угрозы.

Ключевые слова
поиск аномалий, обнаружение вредоносных программ, динамический анализ, анализ поведения, нейронные сети

ESTIMATION OF MALWARE DETECTION ALGORITHM ACCURACY BASED ON ANOMALY SEARCH IN PROGRAM BEHAVIOR

M.V. Baklanovsky[a,b], A.R. Khanov[a], K.M. Komarov[b], P.A. Lozov[b]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
[b] Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation
Corresponding author: awengar@gmail.com

Article info
Received 09.06.16, accepted 19.07.16
doi: 10.17586/2226-1494-2016-16-5-823-830
Article in Russian

For citation: Baklanovsky M.V., Khanov A.R., Lozov P.A., Komarov K.M. Estimation of malware detection algorithm accuracy based on anomaly search in program behavior. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 5, pp. 823–830. doi: 10.17586/2226-1494-2016-16-5-823-830

Abstract

Subject of Research. The paper deals with the algorithm of anomaly detection in the behavior of operating system processes caused by the execution of previously unknown parts of the program code. The algorithm is implemented in the novel intrusion detection system CODA. A testing algorithm allows reducing test time and increasing its accuracy. **Method.** The proposed detection method is based on creation ofbehavior model for legitimate process using sequences of system calls.

Measures of similarity between an arbitrary process and a model are proposed. They allow interpreting the problem of anomaly detection as the problem of vector classification. In order to evaluate the accuracy of the anomaly detection algorithm, the accuracy of the classifier is proposed to be evaluated by cross-validation method. Neural network of perceptron type was used as a classifier. **Main Results.**A platform for the mass distributed testing of malicious programs in virtual machines was developed. Open source library for distributed computing BOINC was used in the platformimplementation. Academic base of malware and open base Malwr was used to select 60 thousand malicious programs. From the general base33.13% of malware have workedcorrectly. A model of legitimate processes runningwithin half an hourwas created. Estimates ofmalware behavior were recorded as vectors. The most accurate neural network was searched for these vectors classification. Neural networks with different teaching parameters and different number of neurons in a hidden layer were looked over. The most precise perceptron was discovered. The accuracy of the best classifier was 91%. **Practical Relevance.** The results can be useful in malware detection. Our algorithm does not require Internet connection.It can find both old and new malware.

Keywords
anomaly detection, malware detection, dynamic analysis, behavior analysis, neural networks

Введение

Борьба с вредоносными программами – одна из самых острых проблем современной компьютерной безопасности. По данным AV-Test[1], к концу 2015 года суммарное количество компьютерных вирусов превысило 375 млн экземпляров, а ежедневно специалисты регистрируют 390 тыс. новых экземпляров. По данным Лаборатории Касперского[2], за 2015 год 34,2% пользовательских компьютеров хоть раз становились жертвами кибератак. Было зарегистрировано 1966324 атаки вредоносного программного обеспечения (ПО) для кражи денежных средств с банковских счетов.

В области технических средств противодействия вредоносным программам за последний десяток лет стал наблюдаться кризис. Соперничество, коэволюция [1, 2] между компьютерными вирусами и антивирусами привела к тому, что последние стали играть роль «догоняющих». Злоумышленники из раза в раз выпускают вредоносные программы, которые обходят все современные средства защиты.

Системы обнаружения вредоносных программ на основе поиска аномалий являются перспективным направлением, так как:

- позволяют выстроить индивидуальную защиту пользователя;
- не требуют частого обновления баз;
- способны обнаруживать как известные, так и неизвестные ранее угрозы.

В настоящей работе предложен алгоритм обнаружения аномалий в поведении процессов путем анализа последовательностей системных вызовов. Проведено тестирование алгоритма и оценена его точность. Алгоритм реализован в прототипе системы обнаружения в рамках проекта КОДА – системы обнаружения и подавления вредоносных программ.

Обзор методов поиска аномалий в поведении процессов

Алгоритмы поиска вредоносных программ по способу сбора информации делятся на два класса – динамические и статические. При статическом анализе подозрительные объекты рассматриваются без их запуска, на основе ассемблерного кода, атрибутов исполняемых файлов. Алгоритмы динамического анализа работают либо с уже запущенными программами, либо сами запускают их в изолированной среде, подвергая изучению ту информацию, которая возникла в процессе работы: анализируют поведение программы, секции кода, данных, следят за потреблением ресурсов. По типу обнаруживаемых объектов алгоритмы поиска вредоносных программ делятся на сигнатурные и аномальные. Сигнатурные стремятся выделить признаки вредоносных программ. Аномальные алгоритмы стремятся описать легитимные программы и научиться искать отклонения от нормы. В данной работе идет речь об аномальных динамических алгоритмах обнаружения.

В 1988 году в работе Д. Деннинг [3] была впервые описана система противодействия атакам, в основе которой лежало построение моделей легитимного потока событий.

В 1996 году в работе С. Форрест [4] был предложен алгоритм обнаружения атак в сетевых сервисах операционных систем (ОС) семейства Linux, основанный на поиске аномалий в последовательностях номеров системных вызовов процессов. Последовательность системных вызовов, совершенных потоком исполнения за ограниченный период времени, мы будем называть трассой. Была записана трасса из 1,5 млн вызовов сервиса sendmail. В процессе записи шло активное взаимодействие с сервисом. По полученной последовательности номеров системных вызовов строилось множество всех встретившихся непрерывных последовательностей фиксированной длины. Такие последовательности были названы stide (sequence time-delay embedding). Мы будем называть их шаблонами. Модель процесса состоит из множества шаблонов. Позднее было предложено учитывать частоту шаблонов и включать в модель процесса лишь те из них, которые встретились больше чем t раз (t-stide, частые шаблоны). Эксперименты показали,

[1] Malware Statistics & Trends Report, https://www.av-test.org/en/statistics/malware/
[2] Kaspersky Security Bulletin 2015, https://securelist.ru/analysis/ksb/27543/

что процессы различимы на основе таких моделей: если извлечь шаблоны из процесса sendmail, то модели процессов ps, ls, finger,ping, ftp, pine, httpd будут содержать от 5 до 32% шаблонов, которые имеются в модели sendmail. Был также проведен эксперимент по обнаружению атак на сервисы sendmail и lpr. Установлено, что во время атаки в трассах появляются не встречавшиеся ранее шаблоны (от 0,3 до 5,3%). В работе Форрест 1998 года [5] была предложена функция оценки близости между моделями процессов (метрика). Она равна максимальному из минимальных расстояний Хемминга между всеми парами шаблонов из двух моделей. В работе 1999 года [6] использовалась оценка локального счетчика фреймов (Locality Frame Count, LFC). Она строится не для двух моделей, а для модели процесса и трассы. LFC – это количество шаблонов на некотором участке трассы, которых нет в модели процесса. Помимо алгоритмов выделения шаблонов были испробованы алгоритмы RIPPER и HMM, адаптированные для обнаружения аномалий в процессах. Но ни один из них не дал лучший результат для всех протестированных программ.

Разработка методов аномального обнаружения атак была продолжена в работах Веспи [7, 8]. Был разработан алгоритм выделения шаблонов переменной длины. Для генерации обучающих последовательностей системных вызовов предложено составлять и использовать функциональные тесты всех наблюдаемых программ. В качестве функции оценки аномалий было предложено находить процент покрытия трасс шаблонами из модели. Также было предложено искать максимальные шаблоны с помощью алгоритма TEIRESIAS [8]. Шаблон называется максимальным, если он не содержится в каком-либо другом шаблоне, встречающимся такое же количество раз. Среди выделенных шаблонов выбиралось подмножество, которого достаточно для покрытия всей трассы, из которой она была выделена. Алгоритмы, основанные на шаблонах переменной длины, показали более точные результаты, чем алгоритмы, использующие шаблоны фиксированной длины.

Использование при анализе лишь номеров системных вызовов может оказаться недостаточным для идентификации. В работе Секар [9] показано, что точно установить контекст, в котором сделан вызов, возможно только в случае, если известен PC (program counter, адрес команды), а также стек адресов возврата. По этой информации строится детерминированный конечный автомат. При тестировании сервисов httpd, ftpd, nfc был установлен существенно меньший процент ложных срабатываний, чем в алгоритмах с поиском шаблонов. Однако запись дополнительной информации о системном вызове требует существенных накладных расходов. В работах [10, 11] в качестве информации, которая участвует в моделировании процесса, выступают аргументы системных вызовов ОС Linux. Но процесс сбора и обработки аргументов вызовов занимает существенно больше времени, чем запись лишь номеров вызовов.

Подавляющее большинство исследований на тему поиска аномалий в поведении процессов связано с анализом атак на сетевые сервисы ОС Linux. Лишь в работе, посвященной системе NORT [12], анализу подвергаются процессы ОС Windows и несколько вредоносных программ под эту платформу. Еще в 2008 году произошла революция в области атакующих программ, в результате чего в тысячи раз увеличилась интенсивность появления компьютерных вирусов и их качество. Способность аномальных систем обнаруживать подобные вредоносные программы ранее не была исследована.

В отечественных исследованиях интерес к задаче поиска аномалий в работе процессов зародился не так давно. В работе [13] предложена архитектура системы обнаружения вторжений, а в [14] описан алгоритм моделирования процессов ОС Linux. Однако точного описания алгоритма в работе не приведено, тестирование проведено на 900 примерах. В работе [15] представлена искусственная иммунная система, реализованная для ОС Windows XP, в основе которой помимо сигнатурных техник лежит поиск аномалий на основе статистики потребления ресурсов процессами, а также использования сетевого траффика. В работе [16] изучается проблема идентификации процессов по генерируемым ими событиям, и в качестве решения приводится особый вид нейронных сетей. Однако тестов на вредоносных программах проведено не было.

В существующих на рынке антивирусных продуктах также используется поиск аномалий в работе процессов ОС. В продукте Cezurity COTA[1] реализован поиск аномалий на основе «слайсов» (срезов) системы, которые включают в себя как информацию о поведении процессов, так и статическую информацию о работающих в системе программах. Составленные в разное время «слайсы» сравниваются, и на их основе строится отчет офицеру безопасности об аномалиях, возникших в системе. Подобная техника используется и в продукте Infowatch Targeted Attack Detector[2]. Однако независимый тест данного метода обнаружения[3] показал, что со своей задачей он справляется существенно хуже уже существующих проактивных технологий и HIPS-систем. Обнаружение по срезам системы учитывает лишь статистическую информацию о поведении процессов, поэтому аномалия в поведении может быть обнаружена лишь спустя несколько часов или дней, что делает невозможным автоматическое противодействие. В продукте Kaspersky Endpoint Security реализована аномальная система обнаружения на основе «белых листов»

[1] http://www.cezurity.com/sites/default/files/docs/cota/cezurity_cota_detection.pdf
[2] https://www.infowatch.ru/products/tad/operation
[3] https://habrahabr.ru/post/261923/

программ, однако атаки на уже запущенные уязвимые программы отлавливает модуль проактивной защиты, в основе которого лежит сигнатурная технология. Хотя поведенческие сигнатуры тестируются, в том числе и на легитимных программах, эта технология не имеет ничего общего с моделированием процессов, которые запущены в системе у конкретного пользователя. Более близкая к аномальному обнаружению технология реализована в продукте Symantec Sonar[1]. Было собрано около 1,2 млрд экземпляров приложений с их поведенческими профилями, методами машинного обучения они были классифицированы на вредоносные и легитимные. Целью настоящей работы является создание системы, способной строить профили процессов ОС конечного пользователя, в результате чего весь детектор вирусов может работать даже в отсутствие Интернет-подключения. Вместо 1,2 млрд легитимных приложений и 100 млн вредоносных объектов система КОДА будет контролировать поведение около сотни процессов конкретного защищаемого пользователя.

Описание системы сбора информации о работе процесса

Процесс ОС Windows состоит из набора потоков. Каждый процесс состоит хотя бы из одного потока. Программа стартует с точки входа и может исполняться сколь угодно долго. В процессе своей работы поток может работать лишь со своей оперативной памятью. Для изменения состояния ОС (чтение и запись в файл, работа с сетью, с другими процессами) он должен вызвать сервис ядра (системный вызов) ОС с помощью команды syscall, sysenter или int XXh. Все сервисы ядра пронумерованы, адреса их функции-обработчиков содержатся в системной таблице SSDT. Программы, которые вызывают системные вызовы напрямую (ассемблерными командами), практически не встречаются. Как правило, используются функции из системных библиотек ОС, которые являются обертками над этими вызовами (ntdll.dll,kernel32.dll,…).

Был написан драйвер для ОС семейства Windows, позволяющий записывать номера системных вызовов каждого потока [17]. В текущей реализации драйвер замедляет работу процессов ОС не более чем на 10% (в случае частых обращений в ядро порядка 10 000 вызовов в секунду). На рис. 1 приведена схема его работы.

Рис. 1. Схема взаимодействия между процессом, ядром операционной системы и драйвером

Было установлено, что в ОС Windows 7 системные вызовы генерируются с частотой порядка 20 тыс. в минуту в режиме простоя, а при запуске программ и интенсивном взаимодействии с ядром ОС это число может достигать сотен тысяч. В ходе экспериментов были записаны трассы всех системных процессов. Наибольший интерес представляют процессы браузеров как наиболее частые объекты атак.

Описание алгоритма обнаружения аномалий

Ранее был разработан и опубликован алгоритм, позволяющий строить модели процессов программ на основе трасс [18]. Алгоритм был протестирован на способность идентифицировать легитимные процессы ОС. Опишем кратко данный алгоритм.

Пусть даны трассы потоков известных процессов. Из этих трасс получим шаблоны переменной длины (термы) по следующему алгоритму:

1. По алгоритму Каркайнена–Сандерса [19] по трассе строится суффиксный массив и множество наибольших общих префиксов всех суффиксов.

[1] http://www.comss.ru/page.php?id=1632

2. Среди наибольших общих префиксов выбирается множество последовательностей, каждая из которых встречается не менее N раз без наложений на другие и на саму себя и имеет длину не менее L.

Модель каждого процесса – это множество термов всех его потоков. Пусть даны трассы потоков неизвестного процесса. Имея множество заранее построенных моделей процессов, нужно найти ту из них, которая в наибольшей степени соответствует данному процессу. Для самой длинной трассы неизвестного процесса вычислим взвешенную сумму двух характеристик: количество встретившихся термов и максимальная длина встретившегося терма. Модель, которая дала максимальную оценку, будем считать наиболее близкой к данному неизвестному процессу. Было установлено, что таким образом можно идентифицировать процесс практически со 100% точностью.

Теперь нужно установить, насколько аномально повел себя процесс во время записи трассы в рамках найденной для него модели. Для этого были подобраны следующие характеристики:

1. LOfMTh – длина наиболее длинного потока процесса;
2. PatCntMTh – количество термов, найденных в наиболее длинном потоке процесса;
3. LestPatMTh – длина самого длинного терма, найденного в самом длинном потоке процесса;
4. MidPatMd – средняя длина терма в модели, найденной для самого длинного потока процесса;
5. AllMdMiss – число вызовов, которые не вошли ни в один терм ни одной модели из набора;
6. RecMd – количество моделей, чьи термы были найдены в самом длинном потоке;
7. ThMaxPatL – наибольшая длина терма модели, подобранной для самого длинного потока, найденная в остальных потоках процесса;
8. ThPatCnt – количество термов модели, подобранной для самого длинного потока, найденных в остальных потоках процесса;
9. ThMiss – количество вызовов всех потоков процесса, не вошедших ни в один терм модели, подобранной для процесса по самому длинному потоку.
10. отношение AllMdMiss/LOfMT – доля всех вызовов всех потоков, не вошедших в термы ни одной модели;
11. отношение ThMiss/TotalCount, где TotalCount – суммарное количество вызовов всех потоков процесса.

Данные характеристики выбраны путем ручного сравнения моделей процессов. Ранее в работе [18] было показано, что наиболее длинные и частые термы модели встречаются в трассах потоков процесса, по которому была составлена модель. Исходя из этого, высокие значения характеристик 2, 3, 4, 7, 8 говорят о высоком соответствии трассы выбранной модели. При составлении 5, 9, 10, 11 характеристик учитывалось, что в алгоритмах поиска аномалий в более ранних работах [7, 8] уже использовался подсчет промахов – вызовов, которые не входят в шаблоны. Также при составлении характеристик учитывались следующие наблюдения:

– если в системе запущен процесс, по которому не была ранее составлена модель, то в его трассах могут быть найдены в небольшом количестве достаточно короткие термы различных моделей, чем меньше таких моделей, тем точнее был идентифицирован процесс;
– процесс, по которому составлена модель, должен иметь достаточно длинные термы из одной модели, небольшое количество ложных срабатываний;
– величину термов модели следует считать относительно средней длины по всем термам модели.

Описание процесса тестирования

Ранее тесты проводились только вручную на нескольких десятках легитимных программ и компьютерных вирусов. В этой связи оценки точности алгоритма обнаружения аномалий не были достаточно обоснованными. Для проведения тестов была разработана и реализована платформа для массового распределенного запуска экземпляров вредоносных программ в виртуальных машинах. Платформа была написана с использованием открытой библиотеки для организации распределенных вычислений BOINC. Тесты проводились на 3 компьютерах в несколько потоков и заняли меньше недели. Если бы они запускались на одной машине в один поток, то заняли бы приблизительно 41 день.

Вначале система КОДА была запущена в режиме обучения: в течение 30 мин шла активная работа с программами на компьютере, записывались трассы всех процессов чистой ОС. За это время было совершено 25233903 системных вызовов из 1805 потоков программ. По каждому процессу была построена модель. Далее все тесты проводились на той же ОС с данным набором моделей процессов.

В течение 1 мин запускался экземпляр вредоносной программы. Каждые 2 с строились трассы всех процессов, вычислялись 11 вышеописанных характеристик (вектор) для каждого процесса зараженной системы, они добавлялись в отчет о работе системы.

Заранее неизвестно, какие процессы в отчетах зараженных систем принадлежали вирусу, а какие вели себя как обычно. В связи с этим исполняемые файлы вирусов были переименованы в virus.exe. Хотя такой способ метки вредоносных процессов не учитывает, что некоторые вредоносные программы могут заражать другие процессы или устанавливают драйвера в систему. Для проведения более точного теста

следует отслеживать все активности вируса, помечая все процессы, на работу которых он мог оказать существенное влияние.

Также система была запущена на 20 мин без посторонних программ, в ходе ее работы шло активное взаимодействие с ОС: открытие окон, работа с браузером, со встроенными программами. Таким образом, был получен отчет о работе чистой системы. Каждые 2 с для каждого процесса также вычислялись и записывались 11 характеристик.

Между тестами восстанавливались снимки оригинальной системы в среде виртуализации VirtualBox. Некоторые вредоносные программы являются уже устаревшими, не могут запуститься на данной платформе. Для того чтобы отсеять вирусы, которые не могут работать и проявить свою вредоносную активность, использовались следующие приемы:

1. проверка правильности PE-заголовка;
2. проверка кода возврата запущенного процесса вредоносного экземпляра;
3. мониторинг появления диалоговых окон ошибок.

В качестве базы для тестирования была использована академическая база вредоносных экземпляров, а также открытая база malwr[1]. Около 32,85% экземпляров не прошли проверку корректности PE-заголовка и были отсеяны еще до запуска процесса тестирования. Затем произвольным образом было отобрано 60 тыс. вредоносных программ для тестов.

В результате 37,07% объектов после запуска показали стандартные окна ошибок, 21,17% вирусов приводили к аварийному завершению работы КОДА, 7,77% запущенных объектов вернули код ошибки, 0,64% были заблокированы брандмауэром Windows 7, 0,19% завершились с вердиктом «Подделка под антивирус», 0,03% завершились с сообщением «Обнаружена ВМ». В результате в тесте участвовало всего 33,13% (19875) вредоносных объектов из выборки, из которых было извлечено 269331 отчетов, в которых было 57058 векторов процессов, порожденных вредоносными файлами. В чистом отчете содержалось 2138 векторов процессов легитимных программ.

Оценка результатов тестирования

Для того чтобы оценить, насколько точно полученные вектора позволяют выделять аномальное поведение программ, нужна функция оценки, которая разделяет вредоносные вектора от легитимных. Такое разделение не является самой целью системы КОДА, так как в ее основу положен принцип нечеткой оценки и различных ответных мер на разной степени аномалии. Но это позволит оценить, насколько точно система выявляет аномалии в поведении процессов.

Пусть H – гипотеза о том, что вектор характеристик принадлежит процессу вредоносной программы, M – оценка от 0 до 1, которая посчитана для данного вектора, T – порог срабатывания. Если $M>T$, то вектор будем оценивать как вредоносный, и гипотеза H принимается, иначе оцениваем его как легитимный и H отвергается.

TruePositive – это количество векторов, полученных из процессов вредоносных программ, для которых гипотеза H принимается.

FalsePositive – это количество векторов, полученных из процессов легитимных программ, для которых гипотеза H принимается.

TrueNegative – это количество векторов, полученных из процессов легитимных программ, для которых гипотеза H отвергается.

FalseNegative – это количество векторов, полученных из процессов вредоносных программ, для которых гипотеза H отвергается.

TruePositiveRate – это доля детектируемых объектов. Если она равна 1, были обнаружены все кортежи вредоносных программ.

$$TruePositiveRate = \frac{TruePositive}{TruePositive+FalseNegative}.$$

FalsePositiveRate – это доля векторов легитимных программ, спутанных с вредоносными. Если она равна 0, то все легитимные программы были распознаны верно.

$$FalsePositiveRate = \frac{FalsePositive}{FalsePositive+TrueNegative}.$$

Обучение нейронной сети

Для проверки точности распознавания аномалий в качестве классификатора была использована нейронная сеть типа персептрон. Для обучения было использовано 1496 легитимных векторов и столько же вредоносных векторов. Для тестирования использовалось 642 легитимных вектора и 28506 вредоносных векторов. Такая разница в количестве тестовых примеров обусловлена тем, что вредоносные программы запускались автоматически, а легитимные – вручную. В дальнейшем будет проведена работа по сбору информации о работе легитимных программ.

[1] http://malwr.com/

Перебирались нейронные сети типа 3-слойный персептрон с произвольными количеством нейронов на скрытом слое, а также с различными параметрами обучения. На входном слое было 11 нейронов, по количеству компонент вектора. На выходном слое был один нейрон. В обучающей выборке он приравнивался к 1 в случае вектора вредоносной программы и 0 в случае вектора легитимной программы. Для каждой нейронной сети считалась площадь под графиком ROC-кривой (AUC), а также точность (ACC). Точность считалась по следующему принципу: выбирался порог срабатывания, при котором *FalsePositiveRate* <0,05, а *TruePositiveRate* максимален. Лучший результат по ACC и AUC дала нейронная сеть с количеством нейронов в слоях (11,99,1). На рис. 2 приведен график ее ROC-кривой. По вертикальной оси – *TruePositiveRate*, по горизонтальной – *FalsePositiveRate*. При доле ложных срабатываний, меньше 0,05, доля обнаруженных вредоносных процессов составляет 0,9074 (точность 0.91).

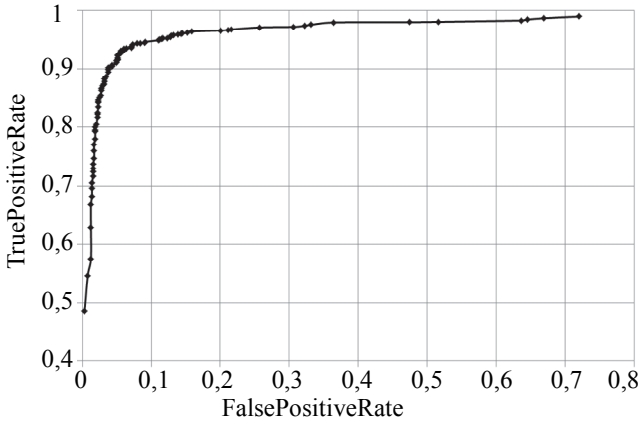

Рис. 2. ROC-кривая наиболее точной из обученных нейронных сетей

Заключение

Для оценки точности алгоритма распознавания вредоносных программ было проведено тестирование на 60 тыс. экземплярах компьютерных вирусов. Были предложены оценки, позволяющие свести задачу распознавания к задаче классификации векторов. Методом перекрестной проверки была установлена точность 91%. В дальнейшем планируется изменить алгоритм оценивания процессов и сделать его поточным. Данный тест показывает, что выделяемые признаки являются значимыми и позволяют искать вредоносные программы как аномалии в модели системы.

Также необходимо расширить набор трасс легитимных процессов. Для этого нужно установить программу мониторинга обычным пользователям и обучиться на поведении их процессов.

Алгоритм является частью системы КОДА, которая предназначена для организации системы противодействия вредоносным программам. Она будет иметь несравнимо более высокую скорость реакции на появление вредоносных программ, чем традиционные решения в этой области, так как детектор аномалий каждого пользователя будет подстраиваться под поведение каждой конкретной системы.

Литература

1. Nachenberg C. Computer virus-antivirus coevolution // Communications of the ACM. 1997. V. 40. N 1. P. 46–51. doi: 10.1145/242857.242869
2. Barat M., Prelipcean D.-B., Gavrilu D.T., A study on common malware families evolution in 2012 // Journal in Computer Virology. 2013. V. 9. N 4. P. 171–178. doi: 10.1007/s11416-013-0192-5
3. Denning D.E. An intrusion-detection model // IEEE Transactions on Software Engineering. 1987. V. SE-13, N 2. P. 222–232. doi: 10.1109/TSE.1987.232894
4. Forrest S., Hofmeyr S., Somayaji A., Longstaff T.A. Sense of self for Unix processes // Proc. IEEE Symposium on Security and Privacy. Oakland, USA, 1996. P. 120–128.
5. Hofmeyr S.A., Forrest S., Somayaji A. Intrusion detection using sequences of system calls // Journal of Computer Security. 1998. V. 6, N 3. P. 151–180.
6. Warrender C., Forrest S., Pearlmutter B. Detecting intrusions using system calls: alternative data models // Proc. IEEE Symposium on Security and Privacy. Oakland, USA, 1999. P. 133–145.
7. Wespi A., Dacier M., Debar H. Intrusion detection using variable-length audit trail patterns // Lecture Notes in Computer Science. 2000. V. 1907. P. 110–129.

References

1. Nachenberg C. Computer virus-antivirus coevolution. *Communications of the ACM*, 1997, vol. 40, no. 1, pp. 46–51. doi: 10.1145/242857.242869
2. Barat M., Prelipcean D.-B., Gavrilu D.T., A study on common malware families evolution in 2012. *Journal in Computer Virology*, 2013, vol. 9, no. 4, pp. 171–178. doi: 10.1007/s11416-013-0192-5
3. Denning D.E. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 1987, vol. SE-13, no. 2, pp. 222–232. doi: 10.1109/TSE.1987.232894
4. Forrest S., Hofmeyr S., Somayaji A., Longstaff T.A. Sense of self for Unix processes. *Proc. IEEE Symposium on Security and Privacy*. Oakland, USA, 1996, pp. 120–128.
5. Hofmeyr S.A., Forrest S., Somayaji A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 1998, vol. 6, no. 3, pp. 151–180.
6. Warrender C., Forrest S., Pearlmutter B. Detecting intrusions using system calls: alternative data models. *Proc. IEEE Symposium on Security and Privacy*. Oakland, USA, 1999, pp. 133–145.
7. Wespi A., Dacier M., Debar H. Intrusion detection using variable-length audit trail patterns. *Lecture Notes in Computer Science*, 2000, vol. 1907, pp. 110–129.

8. Wespi A., Dacier M., Debar H. An intrusion-detection system based on the Teiresias pattern discovery algorithm // Proc. EICAR '99. Aalborg, Germany, 1999. P. 1–15.
9. Sekar R., Bendre M., Dhurjati D., Bollineni P. A fast automaton-based method for detecting anomalous program behaviors // Proc. IEEE Symposium on Security and Privacy. Oakland, USA, 2001. P. 144–155.
10. Mutz D., Valeur F., Vigna G., Kruegel C. Anomalous system call detection // ACM Transactions on Information Systems Security. 2006. V. 9. N 1. P. 61–93.
11. Maggi F., Matteucci M., Zanero S. Detecting intrusions through system call sequence and argument analysis // IEEE Transactions on Dependable and Secure Computing. 2010. V. 7. N 4. P. 381–395. doi: 10.1109/TDSC.2008.69
12. Milea N.A., Khoo S.C., Lo D., Pop C. NORT: runtime anomaly-based monitoring of malicious behavior for Windows // Lecture Notes in Computer Science. 2012. V. 7186. P. 115–130. doi: 10.1007/978-3-642-29860-8_10
13. Оладько В.С., Садовник Е.А. Алгоритм выявления процессов с аномальной активностью // Вестник компьютерных и информационных технологий. 2015. № 8. С. 35–39. doi: 10.14489/vkit.2015.08.pp.035-039
14. Садовник Е.А., Оладько В.С., Ермакова А.Ю., Микова С.Ю. Функции и задачи системы обнаружения вторжений на основе анализа активности вычислительных процессов // Инновационная наука. 2016. № 1-2. С. 121–124.
15. Ваганов М.Ю. Гибридная искусственная иммунная система защиты компьютера от процессов с аномальной активностью: дис. … канд. техн. наук. СПб., 2012. 92 с.
16. Прохоров Р.С. Бихевиористическая идентификация процессов // Математические структуры и моделирование. 2013. Т. 27. № 1. С. 103–112.
17. Одеров Р.С., Тенсин Е. Способы размещения своего кода в ядре ОС Microsoft Windows Server 2008 // Сборник трудов межвузовской научно-практической конференции «Актуальные проблемы организации и технологии защиты информации». СПб.: НИУ ИТМО, 2011. С. 100–102.
18. Баклановский М.В., Ханов А.Р. Поведенческая идентификация программ // Моделирование и анализ информационных систем. 2014. Т. 21. №6. С. 120–130.
19. Karkkainen J., Sanders P., Burkhardt S. Linear work suffix array construction // Journal of the ACM. 2006. V. 53. N 3. P. 918–936. doi: 10.1145/1217856.1217858

8. Wespi A., Dacier M., Debar H. An intrusion-detection system based on the Teiresias pattern discovery algorithm. *Proc. EICAR '99*. Aalborg, Germany, 1999, pp. 1–15.
9. Sekar R., Bendre M., Dhurjati D., Bollineni P. A fast automation-based method for detecting anomalous program behaviors. *Proc. IEEE Symposium on Security and Privacy*. Oakland, USA, 2001, pp. 144–155.
10. Mutz D., Valeur F., Vigna G., Kruegel C. Anomalous system call detection. *ACM Transactions on Information Systems Security*, 2006, vol. 9, no. 1, pp. 61–93.
11. Maggi F., Matteucci M., Zanero S. Detecting intrusions through system call sequence and argument analysis. *IEEE Transactions on Dependable and Secure Computing*, 2010, vol. 7, no. 4, pp. 381–395. doi: 10.1109/TDSC.2008.69
12. Milea N.A., Khoo S.C., Lo D., Pop C. NORT: runtime anomaly-based monitoring of malicious behavior for Windows. *Lecture Notes in Computer Science*, 2012, vol. 7186, pp. 115–130. doi: 10.1007/978-3-642-29860-8_10
13. Oladko V.S., Sadovnik E.A. Algorithms for detection of abnormal activity processes. *Herald of Computer and Information Technologies*, 2015, no. 8, pp. 35–39. doi: 10.14489/vkit.2015.08.pp.035-039
14. Sadovnik E.A., Olad'ko V.S., Ermakova A.Yu., Mikova S.Yu. Functions and problems of intrusion detection system based on the analysis of the computational processes activity. *Innovatsionnaya Nauka*, 2016, no. 1-2, pp. 121–124. (In Russian)
15. Vaganov M.Yu. *Gibridnaya Iskusstvennaya Immunnaya Sistema Zashchity Komp'yutera ot Protsessov s Anomal'noi Aktivnost'yu: dis. ... kand. tekhn. nauk* [Hybrid Artificial Immune System for Computer Protection from Abnormal Activity Processes. Diss. PhD Tech. Sci.]. St. Petersburg, 2012, 92 p.
16. Prokhorov R.S. Behavioristic process identification. *Mathematical Structures and Modeling*, 2013, vol. 27, no. 1, pp. 103–112.
17. Oderov R.S., Tensin E. Methods for mounting of its code into OS Microsoft Windows Server 2008 kernel. *Sbornik Trudov Mezhvuzovskoi Nauchno-Prakticheskoi Konferentsii «Aktual'nye Problemy Organizatsii i Tekhnologii Zashchity Informatsii»* [Proc. Interuniversity Conf. on Topical Problems of Organization and Techniques in Information Security]. St. Petersburg, NRU ITMO, 2011, pp. 100–102. (In Russian)
18. Baklanovsky M.V., Khanov A.R. Identification of programs based on the behavior. *Modeling and Analysis of Information Systems*, 2014, vol. 21, no. 6, pp. 120–130.
19. Karkkainen J., Sanders P., Burkhardt S. Linear work suffix array construction. *Journal of the ACM*, 2006, vol. 53, no. 3, pp. 918–936. doi: 10.1145/1217856.1217858

Авторы

Баклановский Максим Викторович – старший преподаватель, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация; старший преподаватель, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, mb@cit.ifmo.ru

Ханов Артур Рафаэльевич – тьютор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, awengar@gmail.com

Комаров Константин Михайлович – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, erwo@mail.ru

Лозов Петр Алексеевич – студент, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, lozov.peter@gmail.com

Authors

Maxim V. Baklanovsky – senior lecturer, ITMO University, Saint Petersburg, 197101, Russian Federation; senior lecturer, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, mb@cit.ifmo.ru

Arthur R. Khanov – tutor, ITMO University, Saint Petersburg, 197101, Russian Federation, awengar@gmail.com

Konstantin M. Komarov – student, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, erwo@mail.ru

Peter A. Lozov – student, Saint Petersburg State University, Saint Petersburg, 199034, Russian Federation, lozov.peter@gmail.com