УДК 004.492

ОБНАРУЖЕНИЕ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ РАЗЛИЧНЫХ СПОСОБОВ КОМПОНОВКИ ИСПОЛНИМЫХ ФАЙЛОВ

Бабенко Л.К., Кириллов А.С.

Южный федеральный университет, Таганрог, e-mail: blk@tsure.ru, kirillovalexeys@gmail.com

Предложенный в данной работе метод обнаружения вредоносного программного обеспечения (ВПО) позволяет обнаруживать вредоносные образцы на основе интегрированных данных статического и поведенческого анализа. Всеобщая информатизация выводит на новый уровень актуальности различные угрозы информационной безопасности, в том числе вирусные угрозы. В качестве первостепенной меры защиты от вирусных угроз можно назвать ее своевременное обнаружение. Для решения поставленной задачи обнаружения ВПО используется выработанный набор количественных признаков, отражающих качественные данные об исследуемом образце, в частности информацию о способе линковки образца, информацию о вызванных системных функциях и их контексте. При этом сформированные признаки полностью инвариантны к структуре образца ВПО, таким образом ВПО выделяется как монолитный класс, а не набор семейств. По результатам экспериментальных исследований выявлено что ошибка классификации образцов ВПО более чем на 42 % меньше, нежели у конкурирующих методов.

Ключевые слова: статический анализ, поведенческий анализ, аномалии, WINAPI, шелл-код, упаковщик, K-средних

MALWARE DETECTION ON THE BASIS OF THE METHODS OF EXECUTABLE FILES LINKING

Babenko L.K., Kirillov A.S.

Southern Federal University, Taganrog, e-mail: blk@tsure.ru, kirillovalexeys@gmail.com

The method of detection of malicious software, proposed in this paper, allows detecting malicious samples based on integrated static and behavioral analysis data. Informatization brings to a new level of relevance various threats to information security, including virus threats. As a primary measure of protection against virus threats, one can name its timely detection. To solve the problem of malware detection, developed a set of quantitative characteristics reflecting qualitative data of analyzed sample, in particular information of linking method, information of called system functions and their context. At the same time, developed characteristics are completely invariant to the structure of the malware sample, thus malware is considered as a monolithic class, and not a set of families. According to the results of experimental studies, it was revealed that the error of classification of malware samples is more than 42 % less than competing methods.

Keywords: static analysis, behavioral analysis, anomalies, WINAPI, shellcode, packer, K-means

На данный момент существует достаточно много антивирусных продуктов, однако вирусных угроз отнюдь не стало меньше [1]. Среди них есть как массовые проявления вредоносного ПО, например совсем недавно у именитого производителя антивирусных средств Avast было компрометировано ПО CCliner, в результате чего заражению подверглись 2,27 млн пользователей [2]; так и образцы ВПО, предназначенные для целевых атак [3] на конкретные объекты, например набор инструментов WhiteBear группировки Turla [4], который предназначен для атак на дипломатические цели в Европе, Азии и Южной Африке. Если обратиться к более раннему времени, то в 2010 г. было выявлено вредоносное ПО Stuxnet [5], целью работы которого была компрометация SCADA [6] систем крупных промышленных предприятий. Что касается массовых угроз, то в 2011 г. был обнаружен впечатляющих размеров ботнет ZeroAccess [7], общее количество зараженных компьютеров составило 9 млн.

Представленные примеры говорят о том, что, несмотря на достаточно большой промежуток времени, который существует между этими громкими происшествиями, принципиально ничего не изменилось, периодическое выявление таких достаточно разных по своей сути новых угроз говорит о принципиальных недостатках существующих методов обнаружения ВПО.

Описание предлагаемого метода

Предлагаемый в данной работе метод обнаружения ВПО предполагает использование данных статического и поведенческого анализа за счет не просто компоновки результатов данных видов анализа, а их интеграции для выработки набора признаков, на основе которых будет выполняться классификация, удается зарегистрировать разницу в декларируемом функционале и фактическом функционале образца, что позволило

достичь уровня ошибок первого рода – 1,5 %, уровня ошибок второго рода – 4,5 %.

Новизна предлагаемого метода заключается в совместной интеграции результатов статического и поведенческого анализов, для проведения классификации используются количественные признаки, выработанные на основе качественных данных, в частности данных о способе линковки, вызываемых системных функциях и их контексте. Используемый набор признаков является инвариантным к структуре исследуемого образца, тогда как другие методы, опираясь, в частности, на последовательности вызовов системных функций, отражают его структуру. Такой подход позволяет рассмотреть ВПО как один монолитный класс ПО, а не множество небольших, фактически семейств ВПО. Соответственно, основное достоинство – это стойкость к различным видам атак, которые в большинстве своем направлены на скрытие функционала, то есть структуры образца ВПО.

Под системными функциями подразумеваются функции WINAPI. Под контекстом вызова системной функции понимается идентификатор процесса в котором выполнен вызов, и исполнимый модуль, в котором выполнен вызов.

Хочется подчеркнуть, что, несмотря на дальнейшую фокусировку на поведенческом анализе, важнейшим этапом является предварительный статический анализ. Цель статического анализа – извлечь из файла исследуемого образца таблицу импорта, сформированную в процессе линковки, которая содержит в себе набор используемых образцом функций, этот набор может достаточно точно характеризовать функционал, чем и пользуются антивирусы, для определения вредоносности.

В предлагаемом методе используется такая информация, как идентификатор процесса, он носит вспомогательный характер, в частности учитываются только вызовы, выполненные в первом процессе из цепочки запущенного образца, это связано с тем, что порождённые образцом процессы могут принадлежать другим исполнимым файлам, либо их память может быть модифицирована таким образом, что код, исполняемый в этих процессах, не будет совпадать с первоначальным кодом исследуемого образца.

Данные о модуле, в котором выполнен вызов, используются для того, чтобы однозначно определить, что вызванная функция принадлежит именно адресному пространству анализируемой программы. Данный момент является одним из важнейших при проведении анализа полученных данных, так как позволяет выделить именно тот функционал, который задействуется непосредственно анализируемым образцом, таким образом, удаляя «зашумляющие» вызовы, произведенные из подключаемых библиотек. Кроме того, без учета данного момента невозможно корректно определить способ линковки образца.

Одно из главных мест в предложенном методе занимает способ линковки исполнимого файла образца. Для ОС Windows можно выделить несколько способов линковки:

1. Статическая линковка (стандартный способ линковки),

2. Динамическая линковка путем вызова системной функции GetProcAddress, которая позволяет получить адрес указанной функции, основываясь:

- на текстовом представлении имени функции;
- порядковом номере функции в таблице экспорта загруженной библиотеки.

3. Ручная динамическая линковка путем ручного разбора памяти процесса.

Разработчику ВПО приходится прибегать к способам 2 и 3 для скрытия [8] списка используемых функций, так как антивирусы уже давно способны определять вредоносное ПО в случае если разработчик линкует программу с системными библиотеками общепринятым и наиболее комфортным способом.

Использование данных поведенческого анализа, а также данных о способе и данных линковки исполнимого файла позволяет зарегистрировать аномалии, выражающие разницу в декларируемом и реальном функционале исследуемого образца.

Формирование признакового пространства

Задачу обнаружения ВПО можно представить как задачу классификации легитимного и вредоносного программного обеспечения. В качестве алгоритма для проведения классификации предлагается использовать алгоритм К-средних, данный алгоритм был выбран по причине своей простоты и вместе с тем достаточной эффективности для решения поставленной задачи, что подтверждено результатами экспериментов. Ниже представлен набор признаков, на основе которых будет формироваться вектор, представляющий каждый анализируемый образец:

$$P1 = \{0,1\},\ u > 0 \Rightarrow 1,$$

$$P2 = \{0,1\},\ m > 0 \Rightarrow 1,$$

$$P3 = \frac{r}{r_i},\ r_i = t - r,$$

$$P4 = \frac{g}{g_i},\ g_i = t - g,$$

u – кол-во функций, вызванных из неизвестного модуля,
m – кол-во вызванных функций, адрес на которые получен способом 3,
t – общее кол-во вызванных функций,
r – кол-во вызванных функций, адрес на которые получен способом 1,
g – кол-во вызванных функций, адрес на которые получен способом 2.

Приведенные признаки выработаны на основе проведенных теоретических и экспериментальных исследований. Рассмотрим их подробнее.

P1 – бинарный признак, обозначающий наличие функций, вызываемый модуль которых не может быть установлен. Наличие таких функций является аномальной ситуацией и позволяет выделить образцы ВПО. Любой загружаемый стандартными средствами системы модуль в процесс регистрируется в PEB (Process Environment Block) и существует однозначная ассоциация адресного пространства загруженного модуля и информации о нем в структуре PEB. Если вызов какой-либо функции выполнен из адресного пространства, данные о котором не зарегистрированы в PEB, это свидетельство того, что данное адресное пространство было выделено средствами прямой манипуляции с памятью процесса, что, как правило, свойственно достаточно специализированному ПО. Причем для образцов ВПО данная техника является одной из базовых, что подтверждается наличием понятия shellcode [9], который широко освещен в тематической литературе.

P2 – бинарный признак, обозначающий наличие вызванных функций, адрес на которые получен способом номер 3. Как уже было сказано выше, данный способ может быть реализован целенаправленно, разработчиком ВПО, для того, чтобы обойти средства статического и поведенческого анализа.

P3 – вещественное число, обозначающее соотношение количества функций, адрес на которые получен привычным способом номер 1, то есть на этапе загрузки программы к остальным вызванным функциям. Данный признак наилучшим образом позволяет выделить образцы легитимного ПО. На рис. 1 приведено распределение легитимных (Benign Software) и вредоносных (Malware) образцов ПО по данному признаку, подтверждая действенность признака P3. Для удобства восприятия образцы легитимного ПО и ВПО разведены по отдельным областям с одинаковой разметкой оси абсцисс. Каждая точка на оси абсцисс соответствует значению признака P3 для соответствующего образца. Вертикальная линия показывает значение признака равное 5,5. Данное значение было получено на основе экспериментальных данных, то есть значения признака P3 для образцов легитимного ПО, как правило, больше 5,5. В случае если значение переменной *r* равно 0, значение признака P3 стремится к бесконечности, для проведения расчетов, в этом случае значение признака P3 принимается как 0. В случае если значение переменной *r* соответствует значению переменной *t*, переменной r_i присваивается значение 1.

P4 – вещественное число, обозначающее соотношение количества функций, адрес на которые получен способом номер 2, то есть путем вызова функции GetProcAddress к остальным вызванным функциям. Данный признак позволяет выделить образцы «подозрительного» ПО, как правило он усиливает эффект признаков P1, P2 и P3, характеризующих образцы соответственно вредоносного и легитимного ПО. На рис. 2 приведено распределение легитимных (Benign Software) и вредоносных (Malware) образцов ПО по данному признаку, подтверждая действенность признака P4. Для удобства восприятия образцы легитимного ПО и ВПО разведены по отдельным областям с одинаковой разметкой оси абсцисс. Каждая точка на оси абсцисс соответствует значению признака P4 для соответствующего образца. Вертикальная линия показывает значение признака равное 0,5. Данное значение было получено на основе экспериментальных данных, то есть значения признака P4 для образцов легитимного ПО, как правило, не превышало 0,5. В случае если значение переменной *g* равно 0, значение признака P4 стремится к бесконечности, для проведения расчетов, в этом случае значение признака P3 принимается как 0. В случае если значение переменной *g* соответствует значению переменной *t*, переменной g_i присваивается значение 1.

Тестовые выборки

Важнейшим этапом, предшествующим проведению всей дальнейшей работы, является формирование тестовых выборок, именно корректная реализация данного этапа позволяет говорить о справедливости полученных выводов. Для определения эффективности предложенного метода будет производиться сравнение с другими разработками, предварительно рассмотрим, как их авторы формировали выборки для проведения экспериментальных исследований. В работе [10] авторы использовали 81 образец ВПО и 69 образцов легитимного ПО, 26 из 81 образца были собраны лабораторией NTT Secure Platform Laboratory[11]. Относи-

тельно происхождения набора легитимного ПО информации не предоставлено. В работе [12] авторы использовали 615 образцов легитимного ПО и 1615 образцов ВПО, данных о происхождении не представлено. В работе [13] авторы используют выборку из 2199 образцов легитимного ПО, из них 20 были получены с сайта http://download.com как 20 самых популярных программ. В общей выборке был 261 инсталляционный пакет, 1194 программы стандартных для ОС Windows и 744 сторонние программы. Выборка образцов ВПО насчитывает 44 экземпляра из 23 семейств, выборка была получена с сайта http://openmalware.org/.

В случае образцов легитимного ПО были взяты популярные программы, такие как архиваторы, менеджеры загрузок torrent и т.д. Также в выборке образцов легитимного ПО присутствуют образцы, упакованные распространенными упаковщиками типа UPX, ASPACK, образцы системных утилит для администрирования системы, самописные программы для работы с файлами и др. ресурсами ОС, также в выборке присутствуют как графические, так и консольные программы. Таким образом, в выборке легитимного ПО представлен достаточно полный набор образцов, покрывающий широкий спектр функционала. Для формирования выборки образцов ВПО, использовались материалы с сайта virusshare.com, где можно получить архивы ВПО обнаруженного за последнее время в различных источниках. Месячный архив насчитывает порядка 120000 экземпляров. Из данного архива были случайно выбраны 150 экземпляров ВПО. Выборка легитимного ПО составила 71 экземпляр.

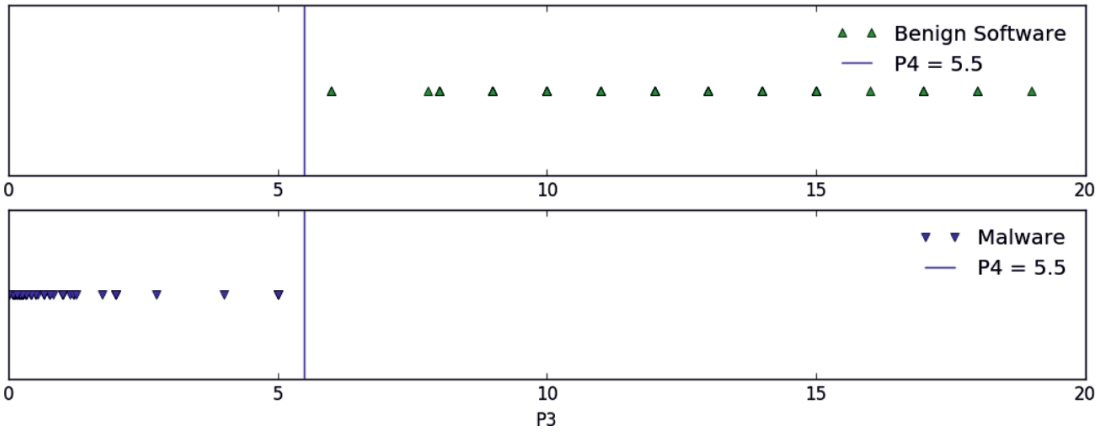

Рис. 1. Распределение легитимных и вредоносных образцов ПО по признаку P3

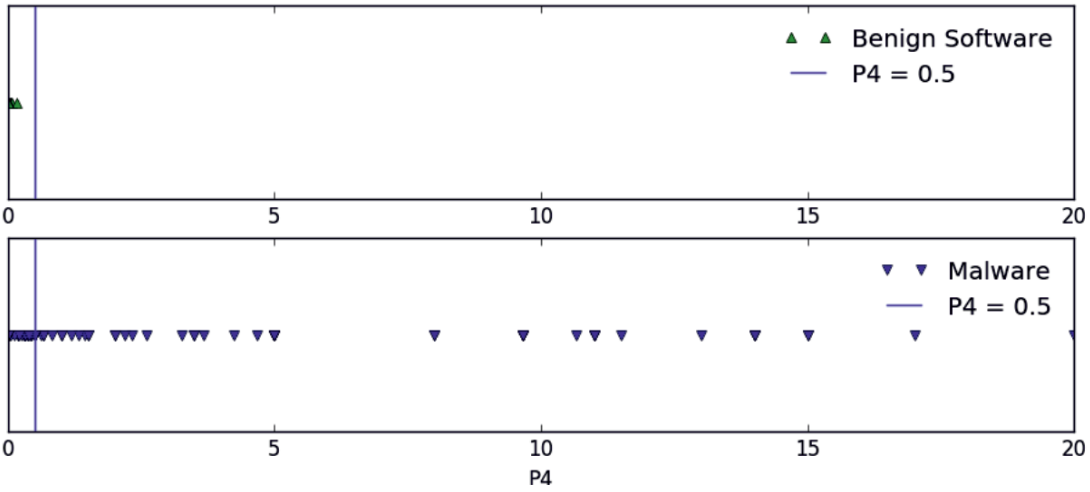

Рис. 2. Распределение легитимных и вредоносных образцов ПО по признаку P4

Сравнение результатов разработанного метода с другими методами

№ п/п		Ошибка определения ВПО	Ошибка определения легитимного ПО
1	Антивирусы, согласно данным [12]	26,68%	
2	Метод, представленный в работе [11]	4%	–%
3	Метод, представленный в работе [12]	3,59%	–
4	Метод, представленный в работе [13]	4,35%	1,6%
5	Предложенный метод для кол-ва перехватываемых функций 336	8,3%	22%
6	**Предложенный метод для кол-ва перехватываемых функций 438**	**1,5%**	**4,5%**

Результаты исследования и их обсуждение

Результаты проведения экспериментальных исследований приведены в таблице. Также в данной таблице приведены результаты работ [11–13].

Для реализации поведенческого анализа использовалась свободно распространяемая платформа Cuckoo Sandbox, которая была модифицирована в соответствии с требованиями метода. В частности, в ходе проведения экспериментальных исследований возникли сложности при работе с образцами, упакованными стандартными упаковщиками, такими как UPX, ASPACK, проблема заключалась в том, что упаковщик по своей сути близок к вредоносному ПО, и все подобные образцы обозначались именно как вредоносные. Для решения данной проблемы в процедуру поведенческого анализа была введена процедура предварительной распаковки образцов и анализ уже распакованных. В случае если распаковать образец корректно не удавалось, он отправлялся на анализ в неизмененном виде, так как авторы ВПО часто подделывают сигнатуры распространенных упаковщиков. В процессе анализа получившихся результатов стало очевидно, что имеет место большое количество ошибок определения легитимного ПО (строка 5 таблицы). Для решения данной проблемы в платформу Cuckoo Sandbox были добавлены дополнительные сигнатуры системных функций, которые свойственны легитимному ПО, в результате чего количество ошибок работы упало в несколько раз до уровня, представленного на строке 6 таблицы.

Заключение

Исходя из полученных данных, ошибка классификации образцов ВПО у предложенного в данной работе метода до 42% меньше, чем у конкурирующих методов, в том числе для метода, который предполагал классификацию образцов ВПО и легитимного ПО на заранее известные классы. В качестве дальнейшего развития предложенного метода планируется провести подробный анализ неверно классифицированных образцов с целью уменьшения ошибки классификации образцов легитимного ПО, а также расширить выборки для получения более значимых результатов.

Работа выполнена при поддержке гранта РФФИ № 15-07-00597.

Список литературы

1. Malware Statistics & Trends Report [Электронный ресурс]. – Режим доступа: http://www.av-test.org/en/statistics/malware/ (дата обращения: 05.10.17).

2. Avast reckons CCleaner malware infected 2.27M users [Электронный ресурс]. – Режим доступа: https://techcrunch.com/2017/09/18/avast-reckons-ccleaner-malware-infected-2-27m-users/ (дата обращения: 05.10.17).

3. Explained: Advanced Persistent Threat (APT) – Malwarebytes Labs [Электронный ресурс]. – Режим доступа: https://blog.malwarebytes.com/cybercrime/malware/2016/07/explained-advanced-persistent-threat-apt/ (дата обращения: 05.10.17).

4. Turla использует для шпионажа инструменты WhiteBear [Электронный ресурс]. – Режим доступа: https://threatpost.ru/turla-apt-used-whitebear-espionage-tools-against-defense-industry-embassies/22164/ (дата обращения: 05.10.17).

5. Stuxnet: первые жертвы – Securelist – Всё об интернет-безопасности [Электронный ресурс]. – Режим доступа: https://securelist.ru/stuxnet-pervye-zhertvy/24277/ (дата обращения: 05.10.17).

6. SCADA [Электронный ресурс]. – Режим доступа: http://www.securitylab.ru/news/tags/SCADA/ (дата обращения: 05.10.17).

7. Trojan.Zeroaccess [Электронный ресурс]. – Режим доступа: https://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99 (дата обращения: 05.10.17).

8. Static Analysis vs Dynamic Imports – Part 1/3 (Technical Article) [Электронный ресурс]. – Режим доступа: http://www.portcullis-security.com/static-analysis-vs-dynamic-imports-part-1-technical-article/ (дата обращения: 05.10.17).

9. Koziol J. et al. The Shellcoder's Handbook // Edycja polska. Helion, Gliwice. – 2004.

10. Tobiyama S., Yamaguchi Y., Shimada H., Ikuse T., Yagi T., Malware Detection with Deep Neural Network Using Process Behavior // Computer Software and Applications Conference (COMPSAC). – 2016. – vol. 2. – P. 577–582.

11. OVERVIEW of NTT Secure Platform Laboratories [Электронный ресурс]. – Режим доступа: http://www.seclab.ecl.ntt.co.jp/e/profile/ (дата обращения: 05.10.17).

12. Anderson B., Quist D., Neil J., Storlie C., Lane T., Graph-based malware detection using dynamic analysis // J. Comput. Virol. – 2011. – № 7(4). – P. 247–258.

13. Wüchner T., Ochoa M., Pretschner A., Malware detection with quantitative data flow graphs // In Proceedings of the 9th ACM symposium on Information, computer and communications security. – 2014. – P. 271–282.