E. P. Tumoyan, K. V. Tsyganok

Statistical Analysis for Classification of Malicious Software

Keywords: multidimensional scaling, WinAPI calls, viruses

This paper proposes a new method of the malicious code classification based on statistical analysis of traces WinAPI calls. We have developed a procedure for programs proximity measurement, taking into account the sequence of WinAPI calls, and the similarity of their arguments. Cluster analysis is used to identify groups programs and classification of the programs.

Е. П. Тумоян, К. В. Цыганок

СТАТИСТИЧЕСКИЙ АНАЛИЗ ДЛЯ КЛАССИФИКАЦИИ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Введение

Классификация вредоносного программного обеспечения (ВПО) представляет собой процесс определения того, что некоторый образец является вариантом известного ранее ВПО. Классификация позволяет антивирусному аналитику сгруппировать образцы одной вредоносной программы для проведения их совместного анализа и, в конечном счете, подготовки сигнатуры, которая обнаруживает все имеющиеся образцы. Учитывая, что большая часть вредоносных образцов регулярно переупаковывается или перешифровывается, статический анализ, в том числе сравнение контрольных сумм, позволяет классифицировать образцы только в самых простых случаях. Таким образом, актуальной задачей является разработка методов точной классификации вредоносных программ. В работе предложен новый метод классификации вредоносных программ на основе статистического анализа трасс вызовов WinAPI, выполняемых программой, и множества файлов, создаваемых программой в заражаемой системе.

1. Предыдущие исследования

Известны два направления анализа потенциально опасных образцов — статический и динамический анализ.

Для статического анализа характерно использование параметров, извлекаемых из исполняемого файла: байтовых сигнатур, дизассемблированного кода, статического графа исполнения и т.д. Работа [1] основана на статическом анализе критических вызовов API. Существует и ряд других работ, например [2], использующих статический анализ для обнаружения и классификации метаморфных вариантов вирусов. Авторы [2] разработали подход, позволяющий находить характеристические наборы значений, сохраняющиеся в каждой вариации образца.

Для методов статического анализа существует ограничение: они не могут работать с упакованными и шифрованными вирусами. Необходимо отметить, что существует класс программного обеспечения, предназначенного для снятия упаковки или шифрования с исполняемых файлов, однако они не подходят для работы с модифицированными или разработанными самостоятельно пакерами и крипторами.

Динамический анализ обходит это ограничение, поскольку учитывает характеристики поведения программы, в том числе трассы инструкций, библиотечных и системных вызовов и т.д. В работе [3] были предложены системы, выполняющие классификацию с использованием динамического анализа API. Система [4] обеспечивает защиту от вредоносных программ путем идентификации потенциально опасных объектов, мониторинга выполнения процессов и потоков объектов, составления контекстов событий исполнения и слияния контекстов связанных процессов и нитей. На основе анализа контекстов система позволяет обнаруживать вредоносные объекты простых и сложных моделей поведения.

2. Разработанный метод

Пусть имеется множество программ $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_i\}$, i=1..N_ρ, где $\boldsymbol{\lambda}_i$ — это отдельная программа. Задача классификации данного множества состоит в формировании отношения $F(\mathbf{\Lambda})$, разделяющего множество $\mathbf{\Lambda}$ на непересекающиеся подмножества ε, для которых выполняется условие:

$$\forall \lambda_i, \lambda_j \in \varepsilon_k : \lambda_i = \lambda_j, \quad (1)$$

где λ_i=λ_j— означает эквивалентность программ. Тоесть задачу классификации программ можно свести к установлению попарной эквивалентности программ.

Задачу установления эквивалентности программ для решения (1) можно свести к решению задачи установления эквивалентности алгоритмов, реализующих эти программы. Пусть есть две программы $\boldsymbol{\lambda}_1$ и $\boldsymbol{\lambda}_2$. Каждую программу можно представить в виде алгоритма A={I}, где I — это инструкции алгоритма. Два алгоритма будут эквивалентны, если для любого слова Qиз алфавита $\mathbf{\Gamma}$ оба алгоритма генерируют одно слово W, также принадлежащее $\mathbf{\Gamma}$:

$$\forall Q : W_i = W_j, \quad (2)$$

где $W_i = A_i(Q)$.

Учтем, что входными данными для программы являются не только данные, вводимые пользователем или получаемые из файлов данных, но также и любые элементы программного окружения. Расширим понятие входного слова алгоритма до понятия *контекста программы*. *Контекст* — это программное окружение, включая файлы данных, переменные в памяти, другие программы и процессы, конфигурацию аппаратного обеспечения ПЭВМ и т.д.

Выделим входной и выходной контексты программы. Входной контекст CI — это контекст, поступающий в программу, выходной контекст CO — это контекст, который генерирует программа, то есть $CO_i = \lambda_i (CI)$. Таким образом, будем считать, что при одном и том же входном контексте программы $\boldsymbol{\lambda}_i$ и $\boldsymbol{\lambda}_j$ эквивалентны, если:

$$\forall\ CI : CO_i = CO_j. \quad (3)$$

Данное утверждение легко доказать. Поскольку $Q \in CI$, а $W \in CO$, выполнение условия (3) с необходимостью приводит к выполнению условия (2).

Отметим, что полный контекст программы включает значительное количество элементов. Выполнение полного сравнения контекстов почти никогда не представляется возможным. Перехваты и внедрение кода или библиотек в другие процессы являются типовыми действиями ВПО. В данной работе эти действия частично учитываются путем контроля вызовов WinAPI.

Ниже приведен фрагмент одной из полученных трасс WinAPI. В первой позиции указываются дата и время, далее PID, имя процесса, WinAPI-вызов, аргументы текущего вызова:

“20130219110633.069”,”984”,”Zeusd93c.exe”,”CreateMutexW”,”SUCCESS”,”0x000001ac”,”lpName->(null)”

“20130219110633.069”,”984”,”Zeusd93c.exe”,”CreateFileW”,”SUCCESS”,”0x000001bc”,”lpFileName->C:\DOCUME~1\user\LOCALS~1\Temp\e883_appcompat.txt”,”dwDesiredAccess->GENERIC_ALL”

В настоящей работе предлагается следующая модель представления выходного контекста $CO \approx \{C, F\}$, где C — упорядоченное множество вызовов WinAPI, F — множество объектов файловой системы, созданных программой. Таким образом, установление эквивалентности двух программ λ_i и λ_j сводится к установлению эквивалентности множеств системных вызовов и множеств файлов, а значит, необходимо определить меру близости данных множеств.

2.1. Подобие системных вызовов

Для вычисления меры близости образцов нельзя использовать статистические параметрические методы, поскольку размер статистики — один образец. В результате проведенных исследований

и экспериментов предлагается вычисление меры близости на основе длины наибольшей общей подстроки для каждой пары трасс. Наибольшая общая подстрока — наибольшая общая непрерывная последовательность WinAPI-вызовов у сравниваемых трасс. При ее нахождении аргументы вызовов не учитывается. Однако для определения меры близости трасс был разработан алгоритм сравнения аргументов (*СтрArg*), сравнивает значения аргументов одинаковых вызовов анализируемых трасс. Вычисляется мера, равная числу совпадающих значений одинаковых аргументов, деленному на общее число аргументов в вызове.

2.2. Подобие файлов

Существует следующее ограничение: ВПО может создавать файлы различных типов, а кроме того, нетипизированные файлы. Таким образом, анализ формата файла и синтаксический разбор в общем случае представляются сложными (или вообще невозможными).Поэтому для определения подобия файлов находятся наибольшие общие подстроки файлов (*СтрFiles*), создаваемых сравниваемыми вызовами в наибольшей найденной общей подстроке вызовов. В итоге, при обнаружении WinAPI-вызова создания файла функция *СтрFiles( )* всегда принимает на вход пару имен файлов из двух сравниваемых трасс и вычисляет близость между ними.

При применении описанных выше алгоритмов будет получена матрица попарных расстояний, характеризующая разницу между образцами. Данную матрицу можно рассматривать как набор z-мерных векторов, где z — количество сравниваемых объектов, характеризующих отдельный образец. Главная проблема использования многомерных данных состоит в том,что люди могут анализировать графики только малых размерностей (двух- и трехмерные).

В настоящее время существуют различные методы снижения размерности данных [5]. В работе рассмотрены два: метод многомерного шкалирования и получение нормы вектора.

Многомерное шкалирование (МНШ) — это способ наиболее эффективного размещения объектов, приближенно сохраняющий наблюдаемые между ними расстояния. Данный алгоритм размещает объекты в пространстве заданной размерности и проверяет, насколько точно полученная конфигурация сохраняет расстояния между объектами, используя алгоритм минимизации некоторой функции (стресс), оценивающей качество получаемых вариантов отображения [6].

Альтернативным методом снижения размерности данных в представленной работе является вычисление нормы вектора в Евклидовом пространстве, который позволяет охарактеризовать каждый имеющийся образец в виде одного числа. Не исключено, что норма в другом топологическом пространстве будет также приемлема.

После проведения операции снижения размерности данных каждый из исследуемых образцов стал характеризоваться двумя величинами: по файлам и по трассе. Таким образом, можно считать, что для каждого из анализируемых файлов был получен характеристический двумерный вектор. Каждый такой вектор можно представить точкой в двумерном пространстве. Множества точек для всех анализируемых программ образуют области. Компактные области являются признаком того, что анализируемые программы подобны.

Для выделения таких компактных областей используется алгоритм нечеткой кластеризации (fuzzyclustering), который позволяет автоматически определять количество кластеров.Каждый кластер представляет собой группу во множестве исследуемых программ. Однако в случае, если размеры кластеров существенно различаются, этот алгоритм допускает ошибки. В данной работе используется модифицированный алгоритм нечеткой кластеризации на основе иерархического разбиения множества на кластеры для получения их точного количества. Полученные центры кластеров и маркеры принадлежности вектора к кластеру позволяют сделать вывод о близости программ из группы.

3. Реализация разработанного метода

Для получения данных об исполнении программы используется система CuckooSandbox [7]. CuckooSandBox— автоматическая система, предназначенная для сбора информации об исполнении вредоносных образцов, которая обеспечивает получение: трасс вызовов WinAPI, множество файлов, создаваемых данной программой, сетевые пакеты, генерируемые программой, трассы ассемблерных инструкций, выполняемых программой.Для автоматической классификации в настоящей работе используется только часть этой информации, в частности трассы вызовов WinAPI и множество файлов, создаваемых данной программой.

Проводится анализ вредоносных программ в виртуальных машинах, находящихся под управлением Cuckoo Framework. В данном исследовании использовались виртуальные операционные системы Windows XP SP2, Windows XP SP3, Windows 7. Перед обработкой нового образца виртуальная машина восстанавливается из снимка состояния. После обработки образца состояние машины (которая может быть к этому моменту заражена) сбрасывается.

Разработанная экспериментальная система основана на методе и алгоритмах, описанных в разделе 2. Процесс классификации показан на рис. 1.

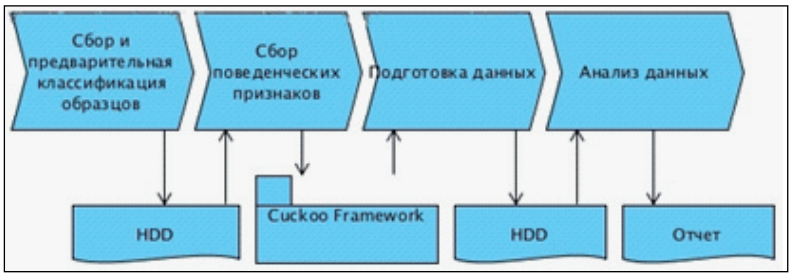

Рис. 1. Процесс классификации

Система включает интерфейс к Cuckoo Framework, подсистему форматирования данных Cuckoo на языке Python и подсистему анализа на языке Matlab. Подсистема анализа и кластеризации данных предоставляет графический интерфейс, который позволяет изменять исследуемую выборку образцов, размеры кластеров, просматривать элементы нужного кластера и представлять результаты кластеризации в виде графика.

4. Экспериментальная проверка разработанного метода

Для тестирования динамического анализатора системы использовался набор исполняемых файлов, состоящий из образцов вредоносного ПО и легальных исполняемых файлов Windows. Набор насчитывает 1080 образцов трасс программ. Вредоносные программы получены из нескольких источников, включая VTMIS и Malware.lu. При проведении эксперимента предполагается, что вредоносные программы однозначно классифицированы в данных источниках. Результаты классификациотображены на рис. 2.

В настоящее время реализованы 2 метода снижения размерности данных (МНШ и нормализации вектора). В ходе экспериментов установлено, что нормализация вектора дает более равномерное распределение образцов в полученном пространстве, в то время как целью преобразования является построение компактных групп. Поэтому для дальнейших экспериментовбыл выбран метод МНШ.

Ошибка классификации составляет около 18,5%. На графике (см. рис.2) видно, что образцы сформированы в компактные группы, принадлежащие различным вирусам. В частности, выделенная группа включает образцы вируса Zeus.

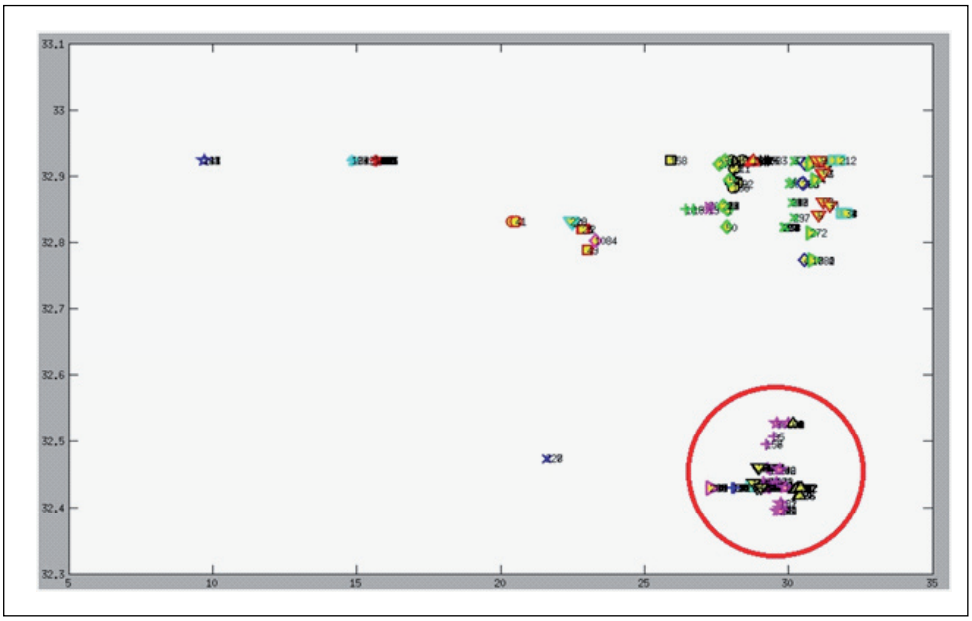

Рис. 2. Результат классификации набора ВПО

Выводы

Предложенный в работе метод обеспечивает оценку близости и кластеризацию образцов ВПО на основе поведенческих признаков. Полученные кластеры используются для классификации ВПО. Разработанный метод не только обеспечивает классификацию образцов вредоносного ПО, но и, в отличие от других методов (таких, как метод на основе алгебраических спецификаций), предоставляет информацию, которую эксперт может оценить визуально и интерпретировать в терминах предметной области.

СПИСОК ЛИТЕРАТУРЫ:

1. *Sathyanarayan V. S., Kohli P., Bruhadeshwar B.* Signature Generation and Detection of Malware Families // Proceedings of the 13th Australasian conference on Information Security and Privacy. Australia. Wollongong. 2008. P. 336–349.
2. *Leder F., Steinbock B., Martini P.* Classification and Detection of Metamorphic Malware using Value Set Analysis // Malicious and Unwanted Software (MALWARE). 4th International Conference. 2009. P. 39–46.
3. *Vinod P., Jain H., Golecha Y. K.* MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API // Proceedings of the 3rd International conference on Security of information and networks . NY: ACM New York, 2010. P. 263–269.
4. *Martynenko V.* System and method for detection of complex malware // U.S. Patent 8042186 B1. 18 Oct. 2011. –13 p.
5. *Huber P. J.* Projection Pursuit (Invited paper) // Harvard University. The Annalasof Statistics. 13. № 2. 1985. P. 435–475.
6. *Терехина А. Ю.* Анализ данных методами многомерного шкалирования. М.: Наука. Главная редакция физико-математической литературы, 1986. – 168 с.
7. Сайт проекта Cuckoo Sandbox. URL: http://cuckoobox.org (дата обращения: 15.08.2013).

REFERENCES:

1. *Sai Sathyanarayan V., Kohli P., Bruhadeshwar B.* Signature Generation and Detection of Malware Families // Proceedings of the 13th Australasian conference on Information Security and Privacy, Australia, Wollongong. 2008. P. 336–349.
2. *Leder F., Steinbock B., Martini P.* Classification and Detection of Metamorphic Malware using Value Set Analysis // Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on. 2009. P. 39–46.
3. *Vinod P., Harshit Jain, Yashwant K. Golecha* MEDUSA: MEtamorphic malware Dynamic analysis Using Signature from API // Proceedings 3rd international conference on Security of information and networks . - NY: ACM New York. 2010. P. 263–269.
4. *Martynenko V.* System and method for detection of complex malware // U.S. Patent 8042186 B1. 18 Oct 2011. –13 p.

5. *Huber P. J.* Projection Pursuit (Invited paper) // Harvard University, The Annalas of Statistics, 13, No. 2, 1985. P. 435—475.
6. *Tereshina A. Y.* analiz dannikh metodami mnogomernogo shkalirovaniay. M.: Nauka. Glavnay redaktsiay fiziko-matematicheskoiy literaturi, 1986. — 168 c.
7. Sait proekta Cuckoo Sandbox. URL: http://cuckoobox.org (15.08.2013).