ПОИСК ПОХОЖИХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ КОДА ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ БЕСПРИЗНАКОВОГО РАСПОЗНАВАНИЯ

А.С. Юмаганов, В.В. Мясников

Самарский государственный аэрокосмический университет имени академика С.П. Королёва (национальный исследовательский университет) (СГАУ), Самара, Россия

В статье рассматривается задача поиска похожих последовательностей кода в исполняемых бинарных файлах программ. Предлагается метод, основанный на разбиении опкодов функций на функциональные группы, формировании описания функций через сопоставление кода исполнения с кодом функций некоторой "стандартной" библиотеки, понижении размерности получаемого косвенного описания, которое на завершающем шаге используется для осуществления поиска. Проводятся экспериментальные исследования применимости представленного метода для решения поставленной задачи, имеющие целью продемонстрировать его работоспособность и эффективность.

Ключевые слова: поиск, последовательность кода, беспризнаковое распознавание.

Введение

С каждым днем количество новых вредоносных программ неуклонно растет [1]. При этом способ детектирования вредоносного программного обеспечения (ПО), основанный исключительно на сравнении сигнатур проверяемого исполняемого модуля, оказывается недостаточным. В силу того, что большая часть новых вирусов является модификацией старых, эффективным методом анализа представляется сравнение исполняемого кода с ранее встречавшимися. К сожалению, прямое сравнение последовательностей кода в силу различных причин (различные опции компиляции/оптимизации кода, вносимые изменения и улучшения в код вирусов и т.п.) оказывается малоэффективным. Это делает задачу построения методов высокоуровнего анализа кода все более актуальной, а применяемые средства обнаружения, основанные на методах распознавания и анализа данных, все более востребованными.

В данной статье представлен метод поиска функций исполняемого бинарного файла, схожих с известными функциями из некоторого "архива" программ. Данный метод можно напрямую использовать и для детектирования вредоносного ПО, если в качестве "архива" программ использовать набор существующих. Учитывая чрезвычайную сложность получения первичного всеобъемлющего описания кода (графа) исполнения, предлагаемый метод поиска основан на принципах беспризнакового распознавания - интересующая нас функция представляется через ее отношения (схожести/близости) с функциями из некоторой - стандартной - библиотеки. Получаемое таким образом избыточное описание для реализации последующего поиска сокращается с использованием метода главных компонент (англ.: Principle Component Analysis - PCA).

Работа построена следующим образом. В первом разделе дано краткое описание предложенного метода. Второй раздел посвящен способу построения описания функции через набор вспомогательных функций (стандартную библиотеку). В третьем разделе кратко описан метод снижения размерности, применяемый для получения окончательного пред-

ставления функции. В четвертом разделе представлен алгоритм поиска похожих функций на основе полученного описания. В пятом разделе приводится способ оценки эффективности метода и результаты проведенных экспериментов. В заключении приводятся выводы, представлен список использованной литературы.

1. Основные понятия и предлагаемый метод решения

В работе далее используются следующие понятия:

- *текущая библиотека* - набор функций исследуемого исполняемого файла;
- *архивные данные* – набор известных функций и их описания через стандартную библиотеку;
- *стандартная библиотека* – вспомогательный набор функций, применяемый для сравнения функций архивных данных и текущей библиотеки.

Решаемая задача неформально формулируется следующим образом: для заданной (или каждой) функции текущей библиотеки найти *наиболее похожую* функцию из архивных данных. Конкретизация понятия (*меры*) *сходства* в разных задачах и разных вариантах решения может быть, вообще говоря, различным. В рамках данной работы предлагаемая мера сходства конкретизируется во втором разделе.

Решение представленной задачи предлагается осуществлять в несколько этапов.

На первом этапе производится представление функции(ий) текущей библиотеки через отношения (похожести) с функциями некоторой наперед выбранной - стандартной - библиотеки. Получаемое описание на втором этапе сокращается с использованием метода PCA. Сокращенное описание заносится в БД описаний, где также хранятся описания функций из архивных данных. Собственно поиск на третьем этапе реализуется средствами БД, выполняя упорядочивание описаний функций архивных данных по критерию евклидовой близости.

Представленный метод, состоящий из указанных трех этапов, имеет ряд настроечных параметров, которые можно использовать для повышения эффективности предлагаемого решения.

2. Представление функций через стандартную библиотеку

Проведя анализ кода исследуемого исполняемого файла с помощью дизассемблера, можно получить разбиение кода ассемблера на функции. Каждая функция состоит из набора команд процессора.

Для удобства, все команды процессора разбиваются на K=45 функциональных групп [2]. Каждая группа содержит в себе набор команд, предназначенных для выполнения действий одного вида. Примерами таких групп являются, например, набор команд пересылки данных, группа команд передачи управления. Такое разбиение предъявляет требования к используемой стандартной библиотеки: каждая функциональная группа команд

должна иметь своего представителя как минимум в одной из функций стандартной библиотеки.

Рассмотрим команды процессора внутри некоторой функции. Для каждой группы команд этой функции строится распределение (пространственное) команд, входящих в эту группу, по длине функции следующим образом.

Пусть $n_0^k,...,n_{N_k-1}^k$ – абсолютные смещения (позиции) относительно начала функции команд группы k, N_k - общее количество команд этой группы в данной функции, $N=\sum_{k=0}^{K-1}N_k$ - общее количество команд в функции - длина функции. Определим пространственное распределение k-го типа команд как абсолютную частоту попадания команд этого типа в некоторый (относительный) приведенный i-ый интервал $(I=100)$:

$$\tilde{f}_i^k=\sum_{j=0}^{N_k-1}I\left(\frac{n_j^k}{N}\cdot 100\in(i-1,i]\right),\quad i=\overline{1,I} \qquad (1)$$

здесь $I(.)$ - индикатор события, принимающий значения "0" или "1" в зависимости от истинности соответствующего аргумента.

Для устранения влияния на результат поиска малых смещений команд в коде используем дополнительно ядро. Тогда оценка распределения команд в коде имеет вид:

$$f_i^k=\sum_{j=1}^{N}f_j^k\frac{1}{h}K(\frac{i-j}{h}),\quad i=\overline{0,I},$$

где $K(r)$ – функция ядра, h – ширина окна. В качестве функции ядра может выступать, например, ядро Гаусса:

$$K(r)=\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}r^2).$$

В результате получаем K векторов приведенного ниже вида, совместно характеризующих конкретную функцию:

$$a_k=\left(f_1^k,f_2^k,\ldots,f_I^k\right)^T,k=\overline{0,K-1}.$$

Данный набор векторов совместно формирует матрицу описания:

$$A=(a_0,a_1,\ldots,a_{K-1}).$$

Аналогичную матрицу, именуемую далее C, можно построить для каждой функции, в том числе - и для функций стандартной библиотеки.

Мера схожести двух функций, задаваемых матрицами *A* и *C*, назовем следующую величину:

$$\mu(A,C)=\sum_{k=0}^{K-1}\alpha_k\mu_{\cos}(\bar{a}_k,\bar{c}_k),$$

где

$$\mu_{\cos}(\bar{a},\bar{c})=\frac{\sum_{i=1}^{I}a_i c_i}{\sqrt{\sum_{i=1}^{I}a_i{}^2}\sqrt{\sum_{i=1}^{I}c_i{}^2}},$$

а величины $\alpha_k \geq 0$, удовлетворяющие условию

$$\sum_{k=0}^{K-1}\alpha_k=1,$$

- суть параметры метода.

При полном совпадении функций мера схожести принимает значение "1", при полном несоответствии – "0".

Пусть далее J – число функций в стандартной библиотеке, каждая из которых имеет описание в виде матрицы B_j. Тогда, сравнивая исследуемую функцию текущей библиотеки с каждой из функций стандартной библиотеки, получим следующий вектор описания этой функции:

$$\bar{x}_A=(\mu(A,C_0),\mu(A,C_1),\ldots,\mu(A,C_{J-1}))^T.$$

3. Метод снижения размерности описания

Так как $J>>1$, воспользуемся *методом снижения размерности* – методом главных компонент. Этот метод сводится к вычислению собственных векторов и собственных значений ковариационной матрицы соответствующих векторов:

$$B=\mathrm{E}\{(x-\mathrm{E}(x))(x-\mathrm{E}(x))^T\}.$$

Отсортировав собственные значения ковариационной матрицы по убыванию, расположим соответствующие собственные векторы в аналогичной последовательности. Если ограничиться $I<J$ максимальными собственными значениями, получим матрицу Y перехода к новой размерности, составленную из I собственных векторов. Тогда для получения вектора признаков размерности $I<J$ воспользуемся следующим выражением (индекс, содержащий ссылку на первоначальное описание далее опустим):

$$z=Yx. \qquad (1)$$

Вектор z выступает в качестве *окончательного представления исследуемой функции через набор функций стандартной библиотеки* и не содержит элементов первичного описания, что позволяет предлагаемый подход ассоциировать с беспризнаковыми методами распознавания [3].

Описанный выше способ описания исследуемых данных через функции стандартной библиотеки применяется как для описания архивных функций, так и для описания функций текущей библиотеки. Матрица перехода к новой размерности *Y* вычисляется предварительно и полагается далее неизменной.

4. Поиск похожих функций

Заключительным этапом предлагаемого метода является поиск похожих функций по полученным описания в виде векторов (1).

Пусть z – вектор, описывающий представление функции текущей библиотеки через стандартную библиотеку, z^* – вектор, описывающий представление архивной функции через стандартную библиотеку. Для сравнения векторов признаков воспользуемся евклидовой метрикой (расстоянием):

$$d\left(z, z^*\right) = \sqrt{\sum_{i=0}^{I-1} (z_i - z_i^*)^2}\ , \qquad (2)$$

где z_i - значение i-ой компоненты вектора признаков первой функции, z_i^* - значение i-ой компоненты вектора признаков второй функции. При $d = 0$ функции считаются одинаковыми.

Предположим, что при модификации функции ее размер изменяется не более чем на 25%. Тогда воспользуемся следующим алгоритмом поиска функций похожих на рассматриваемую:

1. Определяем длину функции N.
2. Получаем список функций архивных данных, размер которых отличается от исследуемой функции не более чем на 25%.
3. Для каждой функции из списка, полученного на втором шаге, найдем евклидово расстояние до исследуемой функции.
4. Сортируем полученный на предыдущем шаге результат по возрастанию величины d.

В результате для рассматриваемой функции получаем упорядоченный по уменьшению схожести (увеличению расстояния (2)) список функций архивных данных.

5. Результаты экспериментов

Для проверки эффективности предложенного метода поиска схожих функций возьмем в качестве архивных данных функции одной библиотеки, а в качестве текущей – функции такой же библиотеки, только другой версии. Для определения априори похожих (идентичных) функций считалось, что при модификации кода библиотеки имена функций не менялись, и среди функций архивных данных нет функций с одинаковым именем.

При указанном допущении оценить качество распознавания можно, получив список похожих функций $\{F_l\}_{l=1,...,L}$, отсортированный по уменьшению схожести. Для этого сопоставим этому списку функций бинарную последовательность $\beta = (\beta_1, \beta_2, ..., \beta_L)$, такую, что при наличии на l-ой позиции функции с тем же именем $\beta_l = 1$, иначе $\beta_l = 0$. Введем следующие меры, используемые обычно как критерии качества информационного поиска [4,5].

Точность для k-ой позиции списка P_k:

$$P_k = \frac{\sum_{l=1}^{k} \beta_l}{k}$$

характеризует отношение количества функций имеющих тоже имя, что и исследуемая функция на первых k позициях упорядоченного списка к общему количеству функций в списке.

2) Полнота для k-ой позиции списка R_k:

$$R_k = \frac{\sum_{l=1}^{k} \beta_l}{K}$$

характеризует отношение количества функций имеющих тоже имя, что и исследуемая функция на первых k позициях упорядоченного списка к общему количеству функций с тем же именем среди архивных функций.

3) Средняя точность для списка:

$$AveP = \sum_{k=1}^{L} P_k (R_k - R_{k-1}), R_0 = 0 .$$

Тогда средняя точность для всех функций текущей библиотеки вычисляется по формуле:

$$P = \frac{1}{S} \sum_{s=0}^{S-1} AveP_s ,$$

где S– количество функций в текущей библиотеке.

Пусть архивные данные представлены библиотекой libtiff 4.0.3 [6], а текущая библиотека представлена одной из следующих версий библиотеки libtiff: libtiff 4.0.4, libtiff 4.0.5, libtiff 4.0.6. В каждой из данных библиотеках содержится в общей сложности около 1200 функций, объем кода - 498 кб.

Результаты экспериментальных исследований по оценки качества работы предложенного метода представлены в табл. 1.

Табл.1 Средняя точность распознавания для разных версий библиотеки libtiff

	libtiff 4.0.4	**libtiff 4.0.5**	libtiff 4.0.6
P	0.8496	0.8479	0.8391

На основании полученных результатов можно сделать вывод как о работоспособности предложенного метода поиска схожих последовательностей кода в исполняемых файлах, так и о достаточно высокой его эффективности.

Заключение

В работе предложен метод поиска похожих последовательностей кода в исполняемых бинарных файлах программ. Метод использует формирование описания функций через сопоставление кода исполнения с кодом функций некоторой "стандартной" библиотеки, понижении размерности получаемого косвенного описания, которое на завершающем шаге используется для осуществления поиска. Приводятся результаты экспериментальных исследований предложенного метода, демонстрирующие его работоспособность и эффективность.

Дальнейшими направлениями работы являются:

- параметрическая оптимизация предложенного метода,
- разработка и использование мер близости, основанных на структурном совместном анализе графов исполнения функций.

Литература

1. 2015 Internet Security Threat Report: Attackers are bigger, bolder, and faster [Электронный ресурс]. – 2015. – URL: http://www.symantec.com/connect/blogs/2015-internet-security-threat-report-attackers-are-bigger-bolder-and-faster (дата обращения 27.03.2016).
2. x86 Assembly Language Reference Manual [Электронный ресурс]. – 2010. – URL: https://docs.oracle.com/cd/E19253-01/817-5477/817-5477.pdf (дата обращения 27.03.2016).
3. Vapnik, V. An Overview of Statistical Learning Theory // Neural Networks, IEEE Transactionson, 1999. Vol. 10, № 5. P.988-999.
4. Buckland, M. K. The relationship between recall and precision/ M.K. Buckland, F.C. Gey // JASIS. – 1994. – Vol. 45. No. 1. – pp. 12-19.
5. Powers, D. M. W. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. // Journal of Machine Learning Technologies. – 2011. – Vol. 2. No. 1. – pp. 37–63.
6. LibTIFF - TIFF Library and Utilities [Электронный ресурс]. – 2015. – URL: http://www.libtiff.org/ (дата обращения 27.03.2016).