УДК 004.491.22

О ЗАДАЧЕ КЛАССИФИКАЦИИ НА ОКРЕСТНОСТИ КОРНЯ ГРАФА ПОТОКА УПРАВЛЕНИЯ ПРОГРАММЫ В КОНТЕКСТЕ ПРОЦЕССА РАЗМНОЖЕНИЯ ФАЙЛОВЫХ КОМПЬЮТЕРНЫХ ВИРУСОВ

Бородин А.В., Юдина М.А., Васильева М.А.

ФГБОУ ВО «Поволжский государственный технологический университет», Йошкар-Ола, e-mail: bor@mari-el.com

Статья посвящена исследованию методов внедрения компьютерных вирусов в исполнимый код, представленный классическими файловыми объектами. Рассматриваются особенности механизмов перехвата управления у легальной программы – контейнера размещения тела компьютерного вируса. Разработан подход к внедрению инструкций перехвата управления в код легальной программы, позволяющий снизить вероятность обнаружения факта перехвата со стороны различного рода эвристических антивирусных алгоритмов и основанный на исследовании графа потока управления программы. Предложена математическая модель описания задачи анализа графа потока управления программы-контейнера. Сформулирована задача классификации стартового кода программ, реализованных на языках высокого уровня. Формализованы стратегии внедрения инструкций перехвата управления во всех мыслимых случаях реализации легальной программы. Новизна предлагаемого подхода определяется повышением степени гарантированности перехвата управления по отношению к описанной в литературе технологии с «неизвестной» точкой входа – Entry Point Obscured (EPO) или Unknown Entry Point (UEP). Предложено решение как для случая возможной классификации окрестности корня графа потока управления, так и для случая отсутствия возможности такой классификации. Область применения предложенного подхода – разработка разрушающих программных воздействий различного назначения.

Ключевые слова: граф потока управления, классификация, компьютерный вирус, обфускация, передача управления, точка входа, язык высокого уровня

ABOUT THE PROBLEM OF CLASSIFICATION ON THE NEIGHBORHOOD OF THE ROOT OF THE CONTROL FLOW GRAPH OF THE PROGRAM IN THE CONTEXT OF PROCESS OF REPRODUCTION OF FILE COMPUTER VIRUSES

Borodin A.V., Yudina M.A., Vasileva M.A.

Federal State Budgetary Educational Institution of Higher Education «Volga State University of Technology», Yoshkar-Ola, e-mail: bor@mari-el.com

Article is devoted to a research of implementation methods of computer viruses in the executable code submitted by classical file objects. Features of mechanisms of interception of management at the licensed program – a container of placement of a body of a computer virus are considered. Approach to implementation of instructions of interception of management in the code of the licensed program allowing to reduce the probability of detection of the fact of interception from different heuristic anti-virus algorithms and based on a research of the graph of a control flow of the program is developed. The mathematical model of the description of a task of the analysis of the graph of a control flow of the program container is offered. The problem of classification of the starting code of the programs implemented in languages of the high level is formulated. The strategy of implementation of instructions of interception of management in all imaginable cases of implementation of the licensed program are formalized. The novelty of the offered approach decides by increase in degree of security of interception of management in relation to the technology described in literature on «unknown» point of entry – Entry Point Obscured (EPO) or Unknown Entry Point (UEP). The solution, as for a case of possible classification of the neighborhood of a root of the graph of a control flow, and for a case of lack of a possibility of such classification is proposed. A scope of the offered approach – development of the destroying program influences of different function.

Keywords: control flow graph, classification, computer virus, obfuskation, transfer of control, entry point, high-level programming language

Одной из важных задач при проектировании компьютерных вирусов (КВ) является разработка механизма (или нескольких механизмов) передачи управления от легальной программы (ЛП) к коду вируса [1]. Самым простым решением этой задачи является перехват управления в точке входа в ЛП [2]. Однако этот способ имеет существенный недостаток. Перехват управления в точке входа в ЛП очень легко обнаружить. На это способны практически все эвристические алгоритмы поиска КВ [3]. С другой стороны, если внедрить механизм передачи управления КВ далеко (в смысле количества инструкций/операторов) от точки входа, то вероятность получения управления КВ существенно снижается с ростом такого расстояния. Усложняется, в общем случае, и процесс внедрения, увеличивая тем самым вероятность возникновения неожиданных для вирмэйкера ошибок [4]. Последствия здесь: сокращение времени скрытого существования КВ в вычислительной системе (ВС), то есть, по сути, рост вероятности удаления КВ из ВС раньше достижения цели,

ради которой вирус разрабатывался [5, 6]. Тем не менее эта технология получила достаточно широкое распространение в КВ для 32-разрядных операционных систем семейства Microsoft Windows. Так поступают вирусы Win32.CTX, Win32.SK, Win32.Blakan, Win32.Deemo и др. Эта технология внедрения получила название технологии вирусов с «неизвестной» точкой входа – Entry Point Obscured (EPO) или Unknown Entry Point (UEP) [7].

Для преодоления перечисленных выше проблем возможно использование вирусом методов формального анализа графа потока управления (ГПУ) ЛП. При этом для сокращения требуемых для исследования ГПУ ЛП ресурсов времени и памяти ВС возможно ограничение области анализа некоторой окрестностью корня ГПУ (входного блока, на который указывает точка входа в ЛП). При этом для многих компилируемых языков программирования высокого уровня в указанной окрестности вблизи корня может встречаться много типовых блоков, чем также можно воспользоваться для упрощения механизма внедрения.

Материалы и методы исследования

В качестве фактического материала для исследования проблемы были использованы результаты дизассемблирования программ, написанных на языках программирования C++ (компиляторы Microsoft Visual Studio 2008, 2015 и 2017), Object Pascal (компилятор Free Pascal Compiler версии 3.0.4) и Visual Prolog (компилятор компании Prolog Development Center версии 7.5), реализованных для операционных систем семейства Microsoft Windows, функционирующих на процессорах семейства Intel x86.

Формальные методы анализа машинного кода базируются в работе на теоретико-множественном и теоретико-графовом подходах.

Результаты исследования и их обсуждение

Рассмотрим поставленную задачу формально. Назовем ГПУ четверку

$$G = \langle V, E, start, stop \rangle,$$

где $\langle V, E \rangle$ – ориентированный граф,

V – множество вершин или базовых блоков (последовательностей смежных инструкций, в которые поток управления входит в их начале и покидает в конце, без останова программы или возможности ветвления [8]),

E – множество дуг, дугу можно рассматривать как последовательность из двух вершин $e = (v_1,\ v_2) \in E,\ v_1, v_2 \in V$, дуги, по сути, представляют поток управления,

start – начальная вершина (корень) ГПУ, $start \in V$,

stop – конечная вершина ГПУ, $stop \in V$.

Введем понятие пути на ГПУ:

$$p = (v_0, v_1, \ldots, v_k) \underset{Def}{\Leftrightarrow}$$

$$\underset{Def}{\Leftrightarrow}\ v_k \in V, \forall i\,(i = 0, 1, \ldots, k-1)\left[v_i \in V, (v_i, v_{i+1}) \in E\right],\ k \in \mathbb{N},$$

где $\mathbb{N}$ – множество натуральных чисел. Отсюда множество всех путей на ГПУ можно определить следующим образом:

$$P \underset{Def}{=} \left\{(v_0, v_1, \ldots, v_k):\ v_k \in V,\right.$$

$$\left.\forall i\,(i = 0, 1, \ldots, k-1)\left[v_i \in V, (v_i, v_{i+1}) \in E\right], k \in \mathbb{N}\right\}.$$

Пользуясь введенными обозначениями, можно определить три полезные функции для формализации понятий, связанных с ГПУ.

1. Функция длины пути:

$$\text{len}:\ P \to \mathbb{N},$$

определенная следующим образом:

$$p = (v_0, v_1, \ldots, v_k) \in P \underset{Def}{\Rightarrow} \text{len}(p) = k.$$

2. Функция начала пути:

$$\text{beg}:\ P \to V,$$

определенная следующим образом:

$$p = (v_0, v_1, \ldots, v_k) \in P \underset{Def}{\Rightarrow} \text{beg}(p) = v_0.$$

3. Функция конца пути:

$$\mathrm{end}: P \to V,$$

определенная следующим образом:

$$p = (v_0, v_1, \ldots, v_k) \in P \underset{Def}{\Rightarrow} \mathrm{end}(p) = v_k.$$

Заметим, что $\{p \in P: \mathrm{len}(p) = 1\} = E \subset P$ и, таким образом, эти функции определены и на E.

Пользуясь введенными функциями, определим еще две множество-значные функции.

4. Функция входящих дуг в вершину:

$$\mathrm{in}: V \to 2^E,$$

определенная следующим образом:

$$\mathrm{in}(v) \underset{Def}{=} \{e \in E: \mathrm{end}(e) = v\},\ v \in V.$$

Здесь 2^E – булеан множества E (множество всех подмножеств множества E).

5. Функция исходящих дуг из вершины:

$$\mathrm{out}: V \to 2^E,$$

определенная следующим образом:

$$\mathrm{out}(v) \underset{Def}{=} \{e \in E: \mathrm{beg}(e) = v\},\ v \in V.$$

Пользуясь двумя последними функциями, отметим, что $\mathrm{in}(start) = \varnothing$ и $\mathrm{out}(stop) = \varnothing$.

Переходя непосредственно к решению поставленной задачи, заметим, что относительно просто, без привлечения методов анализа потоков данных (крайне ресурсоемкая задача), КВ может построить подграф

$$G' = \langle V', E', start \rangle,\ V' \subset V,\ E' \subset E,$$

графа G, определяемый своим множеством путей:

$$P_0'(V_g, l) = \{(v_0, v_1, \ldots, v_k): v_0 = start, v_k \in V,$$

$$\forall i\,(i = 0, 1, \ldots, k-1)\left[v_i \in V, (v_i, v_{i+1}) \in E\right], k \le l, v_k \in V_g \vee k = l\},$$

где V_g – множество вершин, заканчивающихся командой передачи управления без явного задания адреса перехода (например, ret, iret для процессоров семейства x86), множество V_g, фактически, ограничивает возможность построения ГПУ без анализа потоков данных, l – глубина анализа графа потока управления.

Продолжая формирование языка описания математической модели процесса анализа ГПУ программы-контейнера, положим, что V является порождающим множеством (доменом) для семейства мультимножеств $\mathbb{V}$. Для любого мультимножества стандартно определена функция высоты [9]:

$$\mathrm{hgt}: \mathbb{V} \to \mathbb{N}_0,$$

значением которой является максимальная кратность вхождения элементов мултимножества-аргумента:

$$\mathrm{hgt}\,\boldsymbol{A} \underset{Def}{=} \max_{x \in V} k_{\boldsymbol{A}}(x).$$

Здесь использованы следующие обозначения: $\mathbb{N}_0$ – множество неотрицательных целых чисел, $k_{\boldsymbol{A}}: V \to \mathbb{N}_0$ – функция кратности элементов мультимножества $\boldsymbol{A}$. Таким образом, равенство $\mathrm{hgt}\,\boldsymbol{A} = 1$ означает, что все элементы мультимножества $\boldsymbol{A}$ входят в него ровно один раз (все элементы $\boldsymbol{A}$ уникальны).

Определим, далее, функцию контента

$$\mathrm{cont}: P \to \mathbb{V}$$

следующим образом:

$$p = (v_0, v_1, \ldots, v_k) \in P \underset{Def}{\Rightarrow} \mathrm{cont}(p) = \{v_0, v_1, \ldots, v_k\}.$$

С использованием этой функции факт отсутствия циклов в пути p на ГПУ можно представить следующим образом:

$$\text{hgt cont } p = 1.$$

Теперь можно ввести множество путей без циклов:

$$P_0''(V_g, l) = \left\{ p \in P_0'(V_g, l) : \text{ hgt cont } p = 1 \right\}.$$

Простейшей предлагаемой стратегией перехвата управления КВ, у ЛП является внедрение инструкций передачи управления коду КВ во все базовые блоки из множества

$$\left\{ \text{end } p : \ p \in P_0''(V_g, l) \right\}$$

при условии реализации механизма однократного выполнения кода КВ, в том числе при многократных передачах управления на этот код.

Во многих случаях простейшую стратегию можно улучшить. Изучение опыта исследователей компьютерных вирусов [10], а также материалов дизассемблирования множества программ позволило сделать вывод о том, что для многих компиляторов можно выделить инвариантную окрестность ГПУ, определяемую множеством $P_0'(V_g, l)$, при $l \le l_0$, где l_0 – некоторое пороговое значение, зависящее от компилятора. (Интересно, например, что для компилятора языка логического программирования Visual Prolog компании Prolog Development Center значение l_0 оказалось существенно больше, чем у других исследованных компиляторов.) Этот факт позволяет предварительно, на этапе проектирования КВ, составить базу данных типовых окрестностей ГПУ G', изучить их и выделить только лучшие (для внедрения инструкций передачи управления КВ) базовые блоки из множества

$$V' = \bigcup_{p \in P_0'(V_g, l)} \text{cont } p.$$

Таким образом, при внедрении инструкций передачи управления компьютерному вирусу последний предварительно решает задачу классификации по базе данных, логически состоящей из записей вида $\langle G', V_b' \rangle$, где V_b' – множество базовых блоков, наиболее удобных для внедрения и обеспечивающих гарантированное получение КВ управления, $V_b' \subset V'$. Иначе говоря, происходит поиск множества V_b' по ключу G'. Далее КВ производит внедрение инструкций перехвата управления в базовые блоки из множества V_b'.

В случае, когда КВ не удается решение задачи классификации по созданной базе данных, то ему следует воспользоваться простейшей стратегией. Это менее эффективный путь, однако и у него имеются определенные преимущества перед классической технологией ЕРО.

Интересно, что условие применения простейшей стратегии перехвата управления в рамках предлагаемой технологии наталкивает на идею противодействия нашему подходу: компиляторам с языков высокого уровня следует обфусцировать стартовые коды (типовые блоки), активно используя инструкции передачи управления без явного задания адреса перехода. Это приведет к росту мощности множества V_g и укорочению длин путей из этого множества. Данная ситуация может свести на нет эффективность предлагаемого подхода. Критерием качества противодействия в этом случае может стать величина

$$\frac{\left|V_g\right| \times \left|P_0''(V_g, l)\right|}{\sum\limits_{p \in P_0''(V_g, l)} \text{len } p}.$$

Чем больше эта величина, тем труднее для КВ решить задачу эффективного перехвата управления. В то же время, учитывая консервативность разработчиков компиляторов и требования удобства низкоуровневой отладки кода [11], можно предположить, что разработанная в рамках данной статьи методика внедрения разрушающих программных воздействий в исполнимые файлы операционных систем еще долгое время сможет быть эффективно использованной.

Как и технология ЕРО, предложенный подход к внедрению КВ в ЛП провоцирует антивирусные сканеры при поиске заражения по сигнатурам просматривать весь код программы [7], что существенно снижает быстродействие антивирусного программного обеспечения. Увеличение времени на сканирование программного обеспечения означает определенное снижение вероятности обнаружения на фиксированных промежутках времени. Это первый шаг к достижению цели исследования. С другой стороны, обеспечение изменчивости кода КВ за счет обфускации на основе, например, функционально инвариантного псевдослучайного изменения ГПУ КВ [12] позволяет свести на нет возможности сигнатурного поиска, тем самым оставляя возможность обнаружения лишь со стороны эвристических алгоритмов. Именно в этом случае предложенная технология может в максимальной степени продемонстрировать свои сильные стороны.

Заключение

Подводя итог, отметим, что в рамках представленной работы был предложен механизм передачи управления от ЛП к КВ, отличающийся нетривиальным подходом к размещению инструкций/операторов перехвата. Данный подход усложняет обнаружение КВ эвристическими алгоритмами, с одной стороны, увеличивает вероятность передачи управления вирусу, с другой стороны, и, наконец, снижает вероятность фатальных ошибок в системе ЛП + КВ. Данный подход, особенно в сочетании с перспективными методами обфускации [12, 13] внедренного в ЛП кода, а также обфускации карты памяти [14], способен, по мнению авторов, существенно повысить живучесть КВ.

Также следует отметить, что предложенный подход позволяет говорить о формировании новых путей расширенного воспроизводства технологий компьютерных вирусов [15], что, хотя и не является целью настоящей статьи, однако демонстрирует значительный общий потенциал подхода.

Список литературы

1. Бородин А.В. Феномен компьютерных вирусов: элементы теории и экономика существования. Йошкар-Ола: Марийский государственный технический университет, 2004. 144 с.

2. Гульев И.А. Компьютерные вирусы, взгляд изнутри, М.: ДМК. 1998. 304 с.

3. Вавренюк А.Б., Зайчик А.Ю., Иванов М.А., Кутепов С.В., Прилуцкий С.О., Смирнов А.А., Тараканов О.В., Шустова Л.И. База эвристических признаков комплекса программных средств антивирусной защиты компьютерных систем, функционирующих под управлением ОС Linux // REDS: Телекоммуникационные устройства и системы. 2013. Т. 3. № 2. С. 144–147.

4. Бородин А.В. Оптимизация стоимости владения объектно-ориентированной метасистемой в условиях заданной модели угроз // Обозрение прикладной и промышленной математики. 2006. Т. 13. В. 5. С. 843–844.

5. Бойко А.А. Способ аналитического моделирования процесса распространения вирусов в компьютерных сетях различной структуры // Труды СПИИРАН. 2015. № 5 (42). С. 196–211.

6. Далингер Я.М., Бабанин Д.В., Бурков С.М. Математические модели распространения вирусов в компьютерных сетях различной структуры // Информатика и системы управления. 2011. № 4 (30). С. 3–11.

7. Климентьев К.Е. Компьютерные вирусы и антивирусы: взгляд программиста. М.: ДМК Пресс, 2013. 656 с.

8. Ахо А.В., Лам М.С., Сети Р., Ульман Дж.Д. Компиляторы: принципы, технологии и инструментарий. М.: Издательский дом «Вильямс», 2018. 1184 с.

9. Петровский А.Б. Пространства множеств и мультимножеств. М.: Едиториал УРСС, 2003. 248 с.

10. Касперски К. Записки исследователя компьютерных вирусов. СПб.: Питер, 2005. 316 с.

11. Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих. М.: БИНОМ, Лаборатория знаний, 2015. 170 с.

12. Бородин А.В. Линейные конгруэнтные последовательности максимального периода в задачах обфускации программ // Кибернетика и программирование. 2016. № 6. С. 1–19. DOI: 10.7256/2306-4196.2016.6.18499.

13. Вахрамеева Т.Е., Романова А.А., Сенькова А.А., Бородин А.В. Рандомизация потока управления как дополнительный метод обфускации программ // Россия в многовекторном мире: национальная безопасность, вызовы и ответы. Двадцатые Вавиловские чтения: материалы международной междисциплинарной научной конференции. Ч. 2. Йошкар-Ола: Поволжский государственный технологический университет, 2017. С. 203–205.

14. Козлова К.А. Пул переменных программы как объект-хранилище с рандомизацией карты памяти // Инженерные кадры – будущее инновационной экономики России: материалы II Всероссийской студенческой конференции (г. Йошкар-Ола, 21–25 ноября 2016 г.). Ч. 4. Информационные технологии – основа стратегического прорыва в современной промышленности. Йошкар-Ола: Поволжский государственный технологический университет, 2016. С. 48–50.

15. Петрова Д.И. Гипотеза о возможности расширенного воспроизводства технологий компьютерных вирусов // Инженерные кадры – будущее инновационной экономики России: материалы III Всероссийской студенческой конференции (г. Йошкар-Ола, 21–24 ноября 2017 г.). Ч. 4. Информационные технологии – основа стратегического прорыва в современной промышленности. Йошкар-Ола: Поволжский государственный технологический университет, 2017. С. 82–85.