Л. Л. Проценко

ПОИСК ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИСТОЧНИКОВ ЕГО РАСПРОСТРАНЕНИЯ В СЕТИ ИНТЕРНЕТ

В последнее время требования к вредоносному ПО изменились. Если раньше все вирусы были очень маленькими, то современные троянские кони могут иметь достаточно большой размер. Это обусловлено большой их функциональностью и используемыми технологиями. Между тем далеко не всегда удается незаметно загрузить на компьютер пользователя такие объемы информации. Поэтому хакеры стали использовать следующий прием. Сначала ПК заражается довольно маленькой утилитой, которая устанавливает связь с определенным сервером, скачивает оттуда другое вредоносное ПО, инсталлирует и запускает его. Особенно опасны в этом плане универсальные загрузчики. Они позволяют злоумышленнику устанавливать на ПК жертвы разные троянские кони или даже целый их «букет». Все зависит от того, что в данный момент лежит на указанном сервере.

В последнее время троянские программы все чаще стали использоваться в промышленном шпионаже. Они стали настоящей бедой для крупных мировых корпораций. Обнаружить факт промышленного шпионажа очень непросто. Компании могут долго оставаться в неведении, а обнаружив шпионские программы, скрывать факт заражения. Но иногда происшествия все же получают широкую огласку.

Директор отделения по изучению вредоносного ПО компании Computer Associates Роджер Томпсон заявил: «Я считаю преступной наивностью полагать, что атакам подверглись лишь те компании, о поражении которых стало известно общественности» [1].

Сложность обнаружения программ-шпионов заключается в том, что хакеры научились обходить методы сканирования. Это означает, что для противодействия атакам теперь приходится применять многоуровневые системы защиты.

В настоящее время антивирусные компании ведут пассивный поиск вредоносных программ. Добавление в базу происходит только после того, как программа будет прислана пользователем в лабораторию либо заразит один из ее компьютеров. Такой подход не может себя оправдать, так как это означает, что данная вредоносная программа успела поразить большое количество компьютеров (в большей своей части).

Сейчас сервер может перекомпилировать тело троянского коня каждые 10 минут. Проблема состоит в том, что необходимо как можно быстрее обнаружить источник распространения вредоносных программ, для того чтобы в дальнейшем либо взять под контроль этот сервер, либо заблокировать его.

Как уже было сказано, антивирусные компании сегодня не используют активный поиск вредоносных программ. Следовательно, необходимо вести собственный активный поиск таких программ.

Рассмотрим наиболее распространенную программу HijackThis для определения того, что установлено в системе, так как протоколы утилиты HijackThis являются стандартом для многих конференций, посвященных информационной безопасности.

Утилита анализирует системные настройки и отображает их на экране в виде списка. Важно отметить, что утилита не анализирует собранную информацию: предполагается, что пользователь самостоятельно примет решение о том, какие элементы появились в результате деятельности вредоносных программ. В частности, она может определить, какие программы были установлены через Интернет, которые маркированы в данной программе как O16 – DPF (Download Programs Files).

В сети Интернет существует большое количество форумов, в которых общаются профессионалы в защите информации. На данных форумах активно используются протоколы программы HijackThis. Посещение данных форумов является одним из самых лучших способов узнать

о новых вредоносных программах. Но тут стоит другая проблема: как охватить такое большое количество форумов и как можно быстрее?! До недавнего времени решения данной проблемы не существовало даже для малой части информации форумов. Далее будет предложено решение этой проблемы [2–4].

Для описания решения задачи анализа интернет-страниц с целью выявления источников распространения вредоносного программного обеспечения используется теория конечных автоматов.

Конечный автомат представим в виде формальной системы $M = (S, \Sigma, \delta, s_0, L)$, где S — конечное непустое множество состояний; Σ — конечный входной алфавит; δ — отображение типа $S \times \Sigma \to S$; $s_0 \in S$ — начальное состояние; $L \subseteq S$ — множество конечных состояний. Множество состояний $S = \{s_j\}$ принимает следующие условные значения:

s_0 — начальное состояние;

s_1 — анализ интернет-страницы, содержащей информацию о потенциальных источниках распространения вредоносного ПО;

s_2 — проверка, является ли данная ссылка источником распространения вредоносного ПО;

s_3 — конечное состояние.

Следовательно, множество состояний является непустым конечным множество и $S = (s_0, s_1, s_2, s_3)$.

Входной алфавит $\Sigma = \{w_i\}$ может принимать следующие условные значения:

0 — перейти к следующему элементу данного состояния;

1 — перейти к следующему состоянию, элементы для которого найдены;

2 — элементов для этого состояния нет.

Следовательно, входной алфавит является конечным множеством и $\Sigma = \{0, 1, 2\}$.

Введем следующие конечные множества:

— $A = \{a_0, ..., a_n\}$ — состоит из ссылок на все известные интернет-страницы;

— $B = \{b_0, ..., b_k\}$ — состоит из ссылок на страницы, которые содержат информацию о потенциальных источниках распространения вредоносного программного обеспечения, причем $B \subseteq A$;

— $C = \{c_0, ..., c_l\}$ — состоит из ссылок сети Интернет, которые являются потенциальными источниками распространения вредоносного программного обеспечения, причем $C \subseteq A$;

— $D = \{d_0, ..., d_t\}$ — состоит из ссылок сети Интернет, которые являются источниками распространения вредоносного программного обеспечения, причем $D \subseteq C \subseteq A$.

Введем дополнительные функции $f, f_{s_0}, f_{s_1}, f_{s_2}, f_{s_3}$, каждая из которых имеет доступ к управлению вышеописанными множествами A, B, C, D, причем:

— функция f в начале работы вызывает функцию f_{s_0}, во всех остальных случаях вызывает функции, отвечающие за обработку соответствующего состояния автомата, и возвращает значение этих функций;

— функция f_{s_0}, если есть необработанные элементы в множестве A, выбирает следующий необработанный элемент $a_i \in A$ и, если данный элемент содержит информацию о потенциальных источниках распространения вредоносного программного обеспечения, добавляет данный элемент в множество B, помечает элемент a_i как обработанный и возвращает 1, в противном случае помечает элемент a_i как обработанный и возвращает 0. Если необработанных элементов нет, то возвращается 2.

— функция f_{s_1}, если есть необработанные элементы в множестве B, выбирает следующий необработанный элемент $b_i \in B$ и анализирует страницу, расположенную по этому адресу, и если были найдены потенциальные источники распространения вредоносного программного обеспечения $c_0, ..., c_t$, то добавляет данные элементы в множество C, помечает элемент b_i как обработанный

в множестве B, помечает элементы $c_0, \ldots, c_t$ в множестве A как обработанные и возвращает 1, в противном случае помечает элемент b_i как обработанный в множестве B и возвращает 0. Если необработанных элементов нет, то возвращается 2;

— функция f_{s_2}, если есть необработанные элементы в множестве C, выбирает следующий необработанный элемент $c_i \in C$ и, если данный элемент является источником распространения вредоносного программного обеспечения, добавляет данный элемент в множество D, помечает элемент c_i как обработанный и возвращает 1, в противном случае помечает элемент c_i как обработанный и возвращает 0. Если необработанных элементов нет, то возвращается 2;

— функция f_{s_3} «печатает элементы множества D» и завершает работу.

Диаграмма состояний конечного автомата, анализирующего интернет-страницы с целью выявления источников распространения вредоносного программного обеспечения, представлена на рис. 1.

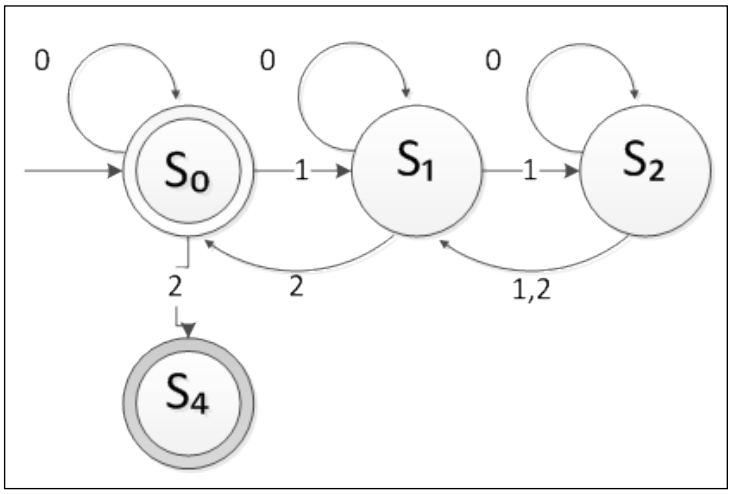

Рис. 1. Диаграмма состояний конечного автомата Мура

На каждом шаге автомат рассматривает значение функции f, а далее одну из функций $f_{s_0}, f_{s_1}, f_{s_2}, f_{s_3}$ в зависимости от состояния автомата. Следовательно, основной проблемой является выбор параметров, от которых зависят данные функции, т. е. необходимо определить параметры, анализируя которые можно найти источники распространения вредоносного программного обеспечения.

Определим для каждой функции, какие необходимы параметры:

— для функции f_{s_0} необходим параметр, с помощью которого можно определить, содержит ли анализируемая страница информацию об источниках распространения вредоносных программ или нет;

— для функции f_{s_1} необходим параметр, с помощью которого можно выделить из содержимого страницы ссылки на интернет-адреса, с которых потенциально происходит распространение вредоносных программ. Очевидно, что в общем случае для функции f_{s_1} необходим тот же параметр, что и для f_{s_0};

— для функции f_{s_2} необходим параметр, с помощью которого можно определить, является ли данная ссылка источником распространения вредоносного программного обеспечения или нет;

— для функции f_{s_3} дополнительных параметров не требуется.

В ходе анализа интернет-страниц, на которых может быть расположена информация о потенциальных источниках распространения вредоносного программного обеспечения, был сделан вывод, что наиболее приемлемая для автоматического анализа информация присутствует на интернет-страницах, содержащих протоколы работы программы HijackThis. А в

частности, нас интересуют записи вида «O16 - DPF: {33331111-1111-1111-1111-611111193423} - http://www.www2.p0rt2.com/files/777.cab». Следовательно, для того чтобы определить, содержит ли страница информацию о потенциальных источниках, достаточно найти на ней слово «O16 - DPF», что и является искомым параметром для функции f_{s_0}. На практике для облегчения поиска таких страниц используются специально подобранные запросы для поисковых машин (Google, Yahoo и др.), например «O16 - DPF».

Как уже говорилось выше, в общем случае слово «O16 - DPF» в качестве параметра подходит и для функции f_{s_1}, так как, зная алгоритм формирования протокола программы HijackThis, можно легко выделить интернет-ссылку из строки, содержащую слово «O16 - DPF».

Для функции f_{s_2} будем использовать в качестве параметра результат обработки различными антивирусными продуктами скаченного по ссылке файла.

Рассмотрим алгоритм поиска источников распространения вредоносного программного обеспечения.

Для усовершенствования метода поиска источников распространения вредоносного ПО введем дополнительные списки ссылок, которые являются доверенными или уже обрабатывались, и назовем их соответственно *список доверенных интернет-адресов* и *список уже обработанных интернет-адресов*. Для доверенных интернет-адресов (microsoft.com, google.com и т. д.) будем считать незначительной вероятность распространения вредоносного ПО с таких адресов. Список уже обработанных интернет-адресов нужен для того, чтобы не скачивать повторно один и тот же файл для проверки и не нагружать программу.

Выходными данными для предложенного алгоритма будет список интернет-адресов, участвующих в распространении вредоносного программного обеспечения, и связанные с ними вредоносные программы.

Алгоритм работы автомата представлен в виде блок-схемы (Рис. 2), которая иллюстрирует концептуальную модель автоматизированной системы детектирования вредоносного программного обеспечения и источников его распространения в соответствии с предложенным алгоритмом.

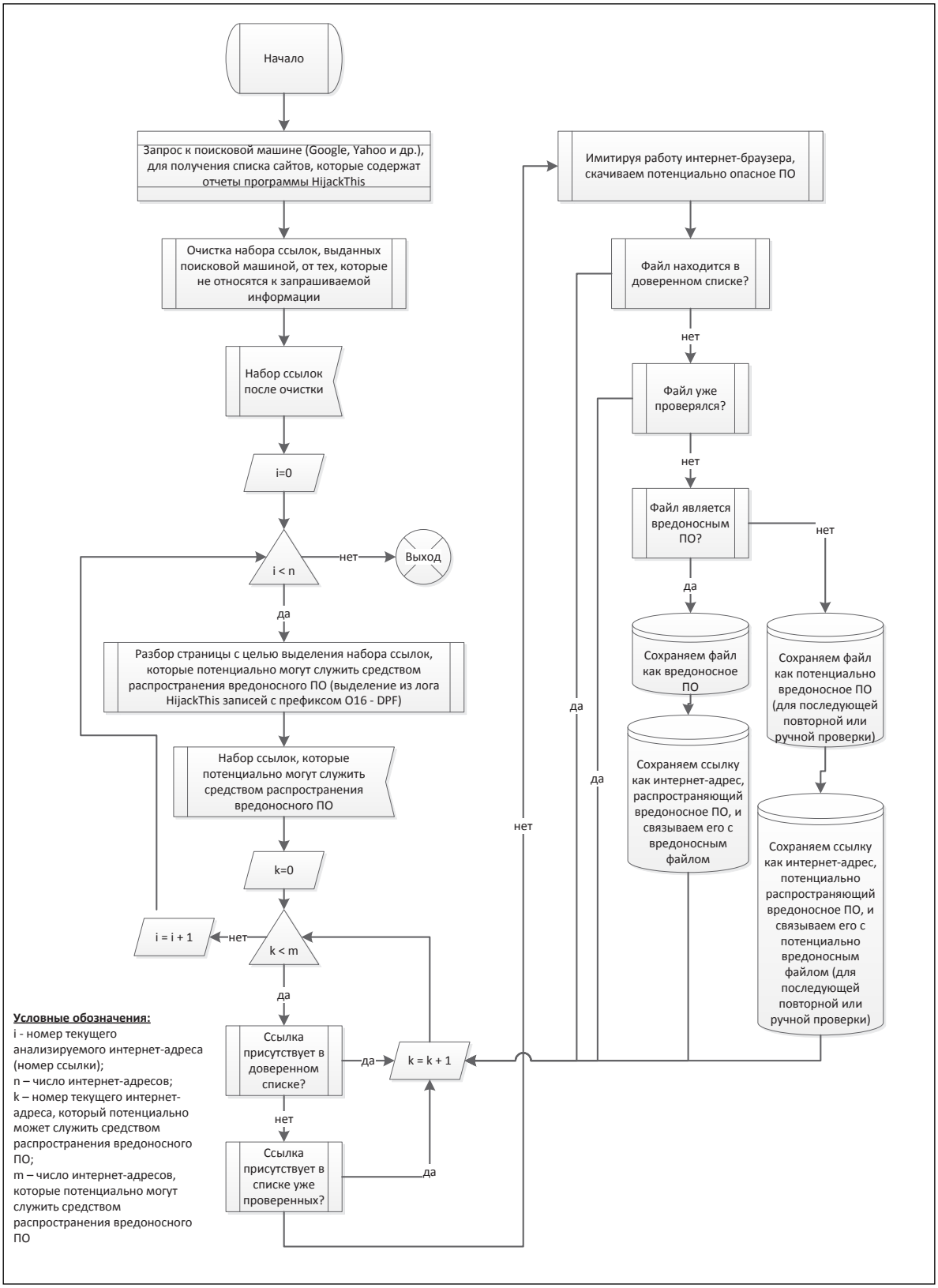

Рис. 2. Алгоритм работы автомата, описывающий концептуальную модель автоматизированной системы детектирования вредоносного программного обеспечения и источников его распространения

СПИСОК ЛИТЕРАТУРЫ:

1. Трояны крадут секреты компаний. URL: http://www.cnews.ru/news/top/index.shtml?2005/07/22/183015.
2. *Проценко Л. Л.* Автоматизированный поиск вредоносного программного обеспечения и источников его распространения в сети Интернет // Всероссийский конкурс инновационных проектов аспирантов и студентов по приоритетному направлению «Безопасность и противодействие терроризму». Каталог поданных проектов. Барнаул, 2006. С. 94–96.
3. *Проценко Л. Л.* Автоматизированный поиск вредоносного программного обеспечения и источников распространения в сети Интернет // Федеральная школа-конференция по инновационному малому предпринимательству в приоритетных направлениях науки и высоких технологий: Тезисы докладов. М.: РГУИТП, 2006. С. 128–133.
4. *Проценко Л. Л.* Автоматизированный поиск вредоносного программного обеспечения и источников его распространения // Второй Саратовский салон изобретений, инноваций и инвестиций. Саратов: Изд-во Сарат. ун-та, 2006.