Отличие заключается в модуле оценки и в модуле работы с базами данных. У агентов оценки доверия и анализа защищенности модуль оценки содержит только блок агрегирования, который ведет перечень объектов, анализирует полученные от различных агентов мониторинга оценки и сохраняет их в базу, вычисляет обобщенную оценку объектов для АИС и распространяет эти оценки среди агентов мониторинга. Модуль хранения информации содержит дополнительно блок хранения оценок. У агента адаптации (подробнее в [5]) модуль оценки содержит только блок адаптации, а вместо блока оценок к модуле хранения информации находится блок параметров.

Таким образом, архитектура программного комплекса состоит из совокупности программных агентов. Предложенная модель оценки доверия к субъектам разделена на подзадачи, которые выполняются различными типами агентов. Предложенные архитектуры программных агентов позволяют учесть указанные возможности для реализации алгоритма контроля над внутренним злоумышленником и использовать в качестве критерия обнаружения предложенную модель оценки доверия к субъектам информационной системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бешта А.А., Новикова Ю.В.* Способ численной оценки состояния автоматизированной информационной системы // Научно- технический вестник Поволжья. – 2013. – № 2. – С. 89-92.
2. *Бешта А.А.* Архитектура агента контроля над внутренним злоумышленником на основе механизма оценки доверия // Известия ЮФУ. Технические науки. – 2012. – № 12 (137). – С. 104-110.
3. *Бешта А.А., Кирпо М.А.* Построение модели доверия к объектам автоматизированной информационной системы для предотвращения деструктивных воздействий на систему // Известия Томского политехнического университета. Управление, вычислительная техника и информатика. – 2013. – Т. 322, № 5. – С. 104-108.
4. *Губанов Д.А.* Обзор онлайновых систем репутации / доверия. – М.: ИПУ РАН, 2009. Интернет-конференция по проблемам управления. – 25 с. – Режим доступа: http://ubs.mtas.ru/bitrix/components/ bitrix/forum.interface/ show_file.php?fid=1671.
5. *Бешта А.А.* Многоагентная эволюционирующая система как средство контроля над внутренним злоумышленником // Вестник волгоградского государственного университета. Серия 10. Инновационная деятельность. – 2012. – Вып. 6. – С. 93-97.

Статью рекомендовал к опубликованию д.т.н., профессор Л.К. Бабенко.

Бешта Александр Александрович – Волгоградский государственный университет; e-mail: abewta@rambler.ru; 400062, г. Волгоград, пр-т Университетский, 100; тел.: 88442460368; кафедра информационной безопасности; ассистент.

Beshta Alexander Alexandrovich – Volgograd State University; e-mail: abewta@rambler.ru; 100, Ave University, Volgograd, 400062, Russia; phone: +78442460368; the department of information security; assistant.

УДК 004.492

Л.К. Бабенко, А.С. Кириллов

МОДЕЛИ ОБРАЗЦОВ ВПО НА ОСНОВЕ ИСПОЛЬЗУЕМЫХ СИСТЕМНЫХ ФУНКЦИЙ И СПОСОБОВ ПОЛУЧЕНИЯ ИХ АДРЕСОВ

Рассматриваются вопросы построения модели вредоносного программного обеспечения (ВПО), ориентированной на включение в структуру образцов не только информации о конкретных системных функциях, но и способах получения их адресов. Такая модель может быть использована для эффективного обнаружения и классификации неизвестных ра-

нее экземпляров ВПО и в отличие от более традиционных будет более гибкой, устойчивой к изменениям, и не зависимой от упакованности и зашифрованности образца ВПО. В работе приведено описание современных методов динамического анализа ВПО и выделены недостатки и ограничения существующих методов, по отношению к предлагаемой модели. Также было приведено описание разработанной системы сбора информации об образцах ВПО, описаны критерии для построения подобной системы и технологии используемые при ее реализации и в частности реализации системы перехвата выбранных функций.

Вредоносное программное обеспечение; модель; динамический анализ; системные функции.

L.K. Babenko, A.S. Kirillov

MODEL OF MALWARE BASED ON SYSTEM FUNCTION AND METHOD OF IT IMPORTING

This article describes features of building malware model, which includes information of calling function and method of it importing. This model can be used to effectively detect and classify unknown malware, unlike more traditional models, this model will be more flexible, resistant to change, and not dependent on the packed and protected malware samples.This paper describes modern methods of dynamic malware analysis and highlights flaws and limitations of existing methods in relation to proposed model. Also, described developed system that collecting information about malware samples, described the criteria for constructing such systems and technologies used for implementation, in particular of the function interception system.

Malware; model; dynamic analysis; system functions.

Обнаружение неизвестного ранее вредоносного программного обеспечения является важнейшей задачей, по причине уверенного роста рынка киберпреступлений, согласно отчету [1] количество экземпляров вредоносного программного обеспечения (ВПО), обнаруженного в течение месяца в 2013 г., выросло в 3 раза по отношению к цифре 2008 г. Киберпреступность выходит на все более новый уровень, на сегодняшний день, пожалуй, наиболее часто появляются в новостях заголовки о распространении очередного банковского трояна. При этом, на ряду с киберпреступностью, вредоносное программе обеспечение стали использовать для саботажа объектов энергетической инфраструктуры некоторых стран [2]. И, несомненно, исследователи и исследовательские группы многих стран работают над разработкой инновационных методов обнаружения ранее не известных образцов ВПО. И каждый автор предлагает совершенно различные техники позволяющие решить эту задачу, их описание а также достоинства и недостатки приведены в [3].

В данной работе рассматриваются вопросы построения модели ВПО, ориентированной на включение в структуру образцов не только информацию о конкретных системных функциях, но и способах получения их адресов. Такая модель может быть использована для эффективного обнаружения и классификации неизвестных ранее экземпляров ВПО.

Исходя из поставленной цели, можно выделить следующие задачи:

1. Выделить системные функции анализ вызовов которых будет производится.
2. Разработать правила ранжирования для функций и способов получения их адресов согласно приведенному ниже методу.
3. Разработать и реализовать систему сбора информации об исследуемом образце.
4. Разработать и реализовать аппарат для выполнения сбора необходимой информации в процессе выполнения исследуемого образца.
5. Осуществить сбор статистических данных на тестовой выборке ВПО и легитимного ПО.

6. Разработать математическую модель на основе полученных статистических данных.
7. Проверить работоспособность модели путем проведения экспериментальных исследований на случайной выборке.

В качестве системных в работе рассматриваются функции Windows API и Windows Native API. Данные функции позволяют программе работать с различными ресурсами, предоставляемыми ОС.

Для ОС Windows можно выделить несколько способов получения адресов функций API:

1. Получение адресов на этапе сборки программы, то есть компоновщик самостоятельно определяет какие сигнатуры в коде программы соответствуют системным функциям и присваивает им соответствующие адреса, при этом данные о таких функциях всегда заносятся в таблицу импорта исполнимого файла.
2. Получение адресов 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. Получение адресов системных вызовов путем ручного разбора памяти процесса с поиском подходящей таблицы экспорта, с последующим получением из нее адреса нужной функции.

При этом необходимо учитывать, что авторы вредоносного программного обеспечения, пытаются скрыть истинный список используемых ими системных функций, затрудняя анализ [4], потому как в противном случае, техники аналогичные [5] способны обнаружить вредоносное программное обеспечение в процессе статического анализа. Для 2 пункта при текстовом представлении имени функции такой способ импорта является вполне стандартным, однако, как было сказано выше, вирусописатели пытаются скрыть факт импорта той или иной функции. Для того, чтобы достичь этой цели, часто прибегают к шифрованию имен функций, расшифровывая их только по мере надобности (перед использованием функции GetProcAddress), то есть в коде программы данные строки всегда находятся в зашифрованном виде. Тогда как для легитимных программ такие техники просто не нужны, и имена всех импортируемых функций присутствуют в теле программы в открытом виде. Что касается импорта по ординалу, это также достаточно редкое явление, основным недостатком которого является возможность изменения порядка следования функции в таблице экспорта PE файла от версии к версии, тем самым лишая ПО переносимости на различные редакции и версии ОС Windows. 3 пункт приведенного списка стоит упомянуть отдельно, потому как для любого легитимного ПО использование подобной техники только затрудняет работу и повышает сложность разработки, кроме того согласно [5] данный способ позволяет существенно затруднить анализ образца ВПО, причем как статический, так и динамический.

Перейдем к рассмотрению указанных в данной работе задач.

Выбор системных функций анализ вызовов которых будет производится, разработать правила ранжирования для функций и способов получения их адресов.

Согласно списку приведенному компанией Microsoft [6] количество доступных функций Windows API (без учета функций Native API) превышает цифру в несколько тысяч. Обработка такого набора функций является весьма не тривиальной задачей и рассмотрение ее актуально.

Совокупность потенциально опасных функций, выбирается, основываясь на предшествующих исследованиях в данной области, а также на анализе исходных текстов публично доступных вирусов. Эта совокупность следующая.

- фиксация в среде;
- запуск полезной нагрузки:
- взаимодействие с командным центром.

Под фиксацией в среде запуска подразумевается тот способ, которым осуществляется повторный запуск образца ВПО в системе, например, после перезагрузки компьютера. Данный тип функциональных возможностей включает в себя все функции которые могут обеспечить или формировать инфраструктуру для повторного запуска экземпляра ВПО. Что касается запуска полезной нагрузки, то в соответствующую группу войдут функции, обеспечивающие запуск тела экземпляра ВПО, например это функции создания удаленных потоков, модификации памяти процесса, функции запуска исполнимых файлов и пр. Функции, вошедшие в группу обеспечивающих взаимодействие с командным центром, в большинстве своем являются функциями работы с сетью.

В процессе анализа как статического (извлечения данных об используемых функциях), так и динамического (получения списка функций используемых фактически) формируется список функций, характеризующий исполнимый файл. Полученный список функций ранжируется согласно степени потенциальной вредоносности и частоты использования. Как правило значения выбираются в диапазоне от 1 до 9, где 1 – соответствует минимальной степени опасности, 9 – максимальной степени опасности.

В соответствии с данными о способе получения адреса системного вызова для каждой функции выбирается коэффициент, значение которого отражает степень потенциальной опасности в связи с используемым способом получения адреса на функцию, данный коэффициент может быть выбран в диапазоне от 1 до 4, в соответствии с описанными выше способами импорта.

Выполнив соответствующие подсчеты для всех функций, используемых исследуемым образцом и всех полученных данных, определяется факт вредоносности образца, тип и семейство угроз.

Разработка и реализация системы сбора информации об исследуемом образце. При разработки архитектуры системы сбора информации об исследуемом образце учитывались такие моменты как:

1. Минимизация влияния на процесс в который выполнено внедрение компонента сбора требуемой информации.
2. Требование высокой производительности системы для максимально быстрой передачи информации о вызванной функции.
3. Мультисессионность работы системы.

Важность данных критериев обусловлена теми фактами, что со стороны вредоносного ПО применяются техники определения того факта что процесс исследуемого исполнимого файла находится во враждебной для него среде, т.е. исследовании, в том числе одной из такой техник является замер времени выполнения той или иной функции и сверка времени ее выполнения с контрольным значением. Важность второго критерия обусловлена тем фактом что при низкой производительность системы вывода информации может быть оказано влияние на исследуемый процесс, также процесс может экстренно завершится, что приведет к потери части данных. Важность мультессионности обусловлена возможностью распределения анализа экземпляров ВПО, тк анализ одного экземпляра занимает достаточно продолжительное время. При реализации мультисессионности очень важно наладить работу с множеством сред для анализа, на начальном этапе, множество

сред для анализа будет обеспечиваться за счет виртуальных машин на базе XEN и за счет предоставляемого им программного интерфейса будет реализовано управление системой снимков ОС, для быстрого возврата ОС в первоначальное состояние, тк исследуемый образец может произвести существенные изменения в среде, которые могут нарушить ее работу или повлиять на ход анализа следующего образца.

Ключевой частью системы является перехват выбранных функций, который, может быть осуществлен с использованием 2-х различных типов перехвата [7]:

1. Перехват путем физической модификации исполнимого файла.
2. Перехват путем модификации памяти загруженного образа.

Перехват путем физической модификации исполнимого файла. Данный метод заключается в модификации бинарного кода исполнимого файла и замены нужных байт на команду перехода, вставке дополнительных инструкций, замене тела функции, также данный тип включает модификацию таблицы импорта и использование библиотек оберток, которые имеют тот же набор функций что и оригинальная библиотека, но перед выполнением функций оригинальной библиотеки осуществляют требуемые действия.

Перехват путем модификации памяти загруженного образа. Данный метод предполагает замену первых байт целевой функции на команду перехода, которая направляет поток управления на функцию-перехватчик, которая, по завершении работы возвращает управление перехватываемой функции. В ОС Windows реализован механизм общего пользования библиотеками, который заключается в том, что если библиотека единожды загружена в память, то при следующей попытке ее загрузить она будет спроецирована в адресное пространство процесса, такой механизм позволяет экономить оперативную память. Особенность заключается в том что модификации такого образа действительна только в рамках текущего процесса. Таким образом, данный метод может быть реализован локально в пределах нужного процесса, либо путем модификации первоначального образа библиотеки, перехват функций которой требуется выполнить.

Из всех вышеописанных методов, наиболее подходящим методом для осуществления перехвата функций является метод модификации образа памяти требуемой библиотеки в пределах нужного процесса, тк данный метод позволяет осуществлять получение данных только от нужного процесса, не требует достаточно высоких прав в системе для его реализации, не нарушает целостность исследуемого исполнимого файла.

Рассмотрим данный метод применительно к ОС Windows подробнее.

Описание реализации данного метода применительно к ОС Windows, а также описание библиотеки для его реализации даны в [8]. На рис. 1 схематично показана разница между потоком выполнения без перехвата и с перехватом.

Ядро метода заключается в замене 5 первых байт так называемого пролога функции на команду перехода, переводящую указатель текущей команды на требуемый адрес, тем самым заставляя выполнить заданный код, после выполнения заданного кода, выполняются 5 байт кода пролога и происходит переход на команду, следующую за первой командой перехода.

В соответствии с приведенными критериями была разработана и реализована система сбора информации о вызовах системных функций на основе клиент-серверной архитектуры. Схематическое изображение всех компонент системы показано на рис. 2.

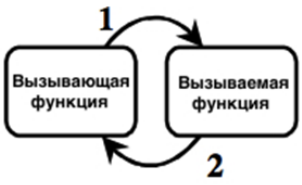

Схема вызова функции до перехвата

Рис. 1. Схематическое изображения потока управления при вызове функции без перехвата и с перехватом

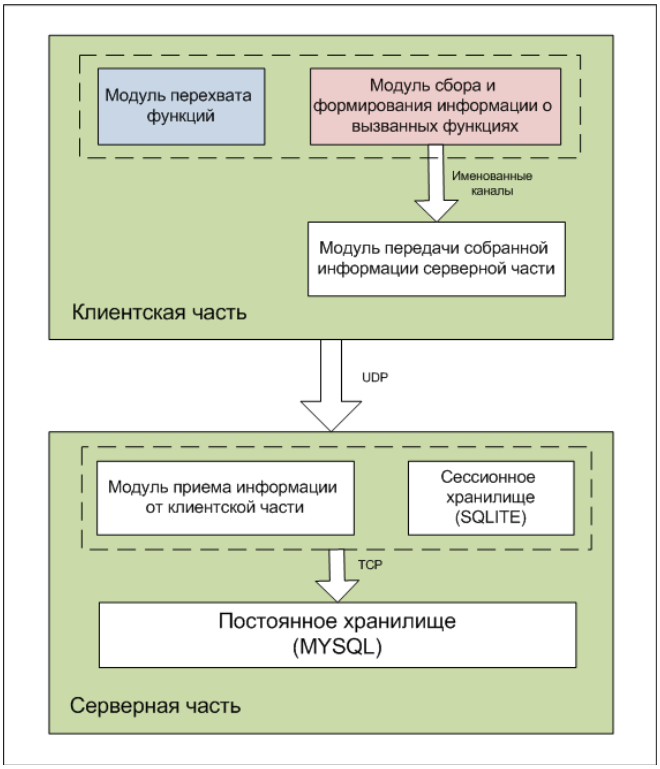

Рис. 2. Общий вид системы сбора информации о вызовах системных функций

Клиентская часть, реализована во внедряемой в исследуемый процесс исполнимого файла библиотеке. Данная библиотека состоит из следующих модулей:

1. Модуль перехвата требуемых функций.
2. Модуль сбора и формирования информации о вызванных функциях.
3. Модуль передачи собранной информации серверной части.

Кратко рассмотрим каждый модуль клиентской части.

Модуль перехвата системных функций. Данный модуль реализован на базе библиотеки Microsoft Detours, предоставляющий удобный интерфейс для перехвата нужной функции, причем как функции Windows API, так и других методом локального сплайсинга, который описан выше. Так же в библиотеке Microsoft Detours реализованы функции запуска указанного процесса с внедрением в него указанной библиотеки. Задача данного модуля состоит в осуществлении перехвата указанных функций в указанном процессе и его потомках. Важным моментом является перехват функций в процессах-потомках и процессах в которые было произведено внедрение кода: требуется перехватывать не все процессы-потомки, а только те в которых может быть произведено внедрение кода анализируемого исполняемого файла, в противном случае, будет получено много паразитных данных, не относящихся к анализируемому исполняемому файлу. Для решения задачи выбора процессов-потомков для перехвата следует рассмотреть современные и популярные техники внедрения кода в запускаемый или уже запущенный процесс. Согласно [9] современные техники внедрения кода можно разделить на техники внедрения кода на следующие типы:

Техники внедрения кода в уже запущенный процесс. Для данных методов характерен поиск процесса для внедрения средствами ToolHelp API, PSAPI или функции QuerySystemInformation. После выбора процесса для внедрения требуется получить его описатель путем вызова функции OpenProcess с указанием при вызове соответствующих прав, далее тем или иным способм выполняется модификация процесса и запуск дополнительного потока в его адресном пространстве. Таким образом, в данном методе ключевым является вызов OpenProcess.

Техники внедрения кода в запускаемый процесс. Для данных техник характерен запуск процесса для внедрения путем вызова функции CreateProcess с флагом CREATE_SUSPENDED и дальнейшие манипуляции с адресным пространством процесса. Ключевым является вызов функции CreateProcess.

Универсальные техники внедрения кода. Данные техника представлена техникой постановки оконных хуков путем вызова функции SetWindowsHookEx, данная функция выполняет регистрацию оконного хука, который позволяет к любому приложению подгружать указанную библиотеку-обработчик установленного хука. Ключевым является вызов функции SetWindowsHookEx.

В соответствии с приведенными техниками, тактика перехвата в процессах-потомках и процессах в который был внедрен код будет строится на основе разбора параметров указанных функций и предварительной подготовке целевых процессов.

Модуль сбора и формирования информации о вызванных функциях. Данный модуль предоставляет интерфейс для задания информации о вызванной функции и формирует структуру сообщения передаваемого на клиентскую часть. Передаваемое сообщение состоит из следующих полей:

- имя вызванной функции;
- код операции;
- идентификатор процесса в котором произошел вызов;
- идентификатор потока в котором произошел вызов;
- время вызова.

Данная информация упаковывается в С – структуру и передается посредством именованного канала в следующий модуль – модуль передачи собранной информации серверной части. Такая архитектура обусловлена тем, что требуется максимально быстро передать всю информацию о вызове и при этом не нарушить работу процесса в котором находится внедряемая библиотека.

Модуль передачи собранной информации серверной части. Данный модуль представляет из себя отдельный поток ОС выполняющийся в контексте процесса в который произошло внедрение, данный поток выполняет сбор данный пришедших через именованный канал, постановку их в очередь для отправки и отправку на серверную часть посредством UDP сокета. Протокол UDP так же выбран из соображений производительности и минимизации задержек на передачу информации.

Серверная часть реализована в виде нескольких python скриптов. Первый из них выполняет запуск 2-х процессов, данные процессы выполняют следующие функции:

1. Прием информации от клиентской части и передачу в меж процессную очередь для обработки другим процессом.
2. Извлечение данных из межпроцессной очереди, разбор и запись в БД SQLITE с целью дальнейшего анализа.

Таким образом, данный скрипт обеспечивает агрегацию информации о запуске одного исполняемого файла.

Серверная часть представляет собой набор скриптов, один из которых выполняет прием данных о сеансе запуска исполняемого файла и помещает их в SQLITE базу, при этом, данный процесс выполняется в приделах одного процесса ОС, для минимизации влияния на другие сессии анализа образца. По завершении сессии каждый обработчик передается оперативные данных из SQLITE таблицы в БД на базе MYSQL которая хранит информацию о всех исполнимых файлах чей анализ был произведен.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Fortinet 2013 Cybercrime Report [Электронный ресурс] // URL: http://www.fortinet.com/sites/default/files/whitepapers/Cybercrime_Report.pdf (дата обращения: 23.05.2013).
2. Stuxnet [Электронный ресурс] // Википедия, свободная энциклопедия. – 2013. URL: http://en.wikipedia.org/wiki/Stuxnet (дата обращения: 23.05.2013).
3. *Бабенко Л.К., Кириллов А.С.* Модели образцов вредоносного программного обеспечения на основе используемых системных функций и способов получения их адресов // Материалы XIII Международной начно-практической конференции «ИБ-2013» Ч. 1. – Таганрог: Изд-во ЮФУ, 2013. – С. 181-186.
4. Static Analysis vs Dynamic Imports – Part 1/3 (Technical Article) [Электронный ресурс] // Portcullis Computer Security. – 2013. URL: http://www.portcullis-security.com/static-analysis-vs-dynamic-imports-part-1-technical-article/ (дата обращения: 23.05.2013).
5. Patent US8161548 - Malware detection using pattern classification [Электронный ресурс] // Google Patents. – 2012. URL: http://www.google.com/patents/US8161548 (дата обращения: 23.05.2013).
6. Windows API Sets (Windows) [Электронный ресурс] // Microsoft Windows Dev Center - Desktop. – 2012. URL: http://msdn.microsoft.com/en-us/library/windows/desktop/hh802935(v=vs.85).aspx (дата обращения: 23.05.2013).
7. Hooking [Электронный ресурс] // Википедия, свободная энциклопедия. – 2013. URL: http://en.wikipedia.org/wiki/Hooking (дата обращения: 25.10.2013).
8. *Hunt G., Brubacher D.* Detours: Binary Interception of Win32 Functions // WINSYM'99 Proceedings of the 3rd conference on USENIX Windows NT Symposium. – 1999. – Vol. 3. – С. 14-14.
9. Code Injection Techniques - InfoSec Institute [Электронный ресурс] // InfoSec Institute. – 2013. URL: http://resources.infosecinstitute.com/code-injection-techniques/ (дата обращения: 25.10.2013).

Статью рекомендовал к опубликованию д.т.н., профессор Я.Е. Ромм.

Бабенко Людмила Климентьевна – Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Южный федеральный университет»; e-mail: blk@fib.tsure.ru; 347928, г. Таганрог, ул. Чехова, 2, корпус "И"; тел.: 88634312018; кафедра безопасности информационных технологий; профессор.

Кириллов Алексей Сергеевич – e-mail: kirillovalexeys@gmail.com; кафедра безопасности информационных технологий.

Babenko Lyudmila Klimentevna – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education “Southern Federal University”; e-mail: blk@fib.tsure.ru; Block “I”, 2, Chehov street, Taganrog, 347928, Russia; phone: +78634312018; the department of security of information technologies; professor.

Kirillov Alexey Sergeevich – e-mail: kirillovalexeys@gmail.com; the department of security of information technologies.