Математические структуры и моделирование
2010, вып. 21, с. 104–112

УДК 004.056.0

РЕАЛИЗАЦИЯ СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ КАК ЧАСТИ ИСКУССТВЕННОЙ ИММУННОЙ СИСТЕМЫ

М.Ю. Ваганов

В статье рассматривается реализация системы обнаружения вредоносных программ на основе анализа активности рабочей станции.

Введение

Современный мир невозможно представить без средств коммуникаций и вычислительной техники, в которых главенствующую роль играет программное обеспечение. Информационные технологии прогрессируют очень быстро, охватывая все более широкие области человеческой деятельности: информатизация в отраслях производства, бизнеса, образования, государственных учреждениях и т.д., интеграция автоматизированных систем в локальные и глобальные вычислительные сети. Поэтому безопасность информационных технологий является одним из важнейших аспектов обеспечения их функционирования.

Постоянно растущие масштабы распространения различных компьютерных угроз, основной целью которых, часто является хищение конфиденциальных данных, заставляют разработчиков антивирусного программного обеспечения искать новые, проактивные методы в борьбе с неизвестными угрозами. Одним из таких направлений является использование моделей и решений, уже достаточно хорошо проработанных в эпидемиологии и иммунологии [1].

Целью данной работы является реализации системы обнаружения вредоносных программ на основе анализа активности рабочей станции, являющейся первым шагом в создании модели искусственной иммунной системы, способной реагировать как на известные, так и на новые, не встречавшиеся ранее разновидности вредоносного программного обеспечения.

1. Иммунные системы

Иммунная система живого организма – сложный комплекс структур, способный различать нормальную и аномальную жизнедеятельность клеток, которая может быть представлена как результат действия различных возбудителей, в

Омский государственный университет им. Ф.М. Достоевского.
E-mail: asterxpst@gmail.com

том числе бактерий и вирусов. Конечной целью иммунной системы является формирование иммунного ответа, ответственного за уничтожение чужеродного агента, которым может оказаться болезнетворный микроорганизм, инородное тело, ядовитое вещество или переродившаяся клетка самого организма.

Все формы иммунного ответа можно разделить на приобретенные и врожденные. Основное различие между ними состоит в том, что приобретенный иммунитет высокоспецифичен по отношению к конкретному типу антигенов и позволяет быстрее и эффективнее уничтожать их при повторном столкновении.

Ввиду непрекращающейся адаптации возбудителей, изменения методов распространения и инфицирования также не останавливается эволюционное развитие иммунных систем, в результате которого последние приобрели такие важные особенности, как адаптируемость, наличие памяти, толерантность (терпимость, по отношению к аномалиям, произошедшим на ранних стадиях функционирования системы) и избирательность.

2. Искусственная иммунная система

Под искусственной иммунной системой принято понимать программные комплексы, принципы функционирования которых аналогичны иммунным системам живых организмов [2]. В данной работе предпринимается попытка создания программного комплекса, использующего принципы иммунной системы [3] для детектирования вредоносного кода. В качестве целевой платформы выбраны операционные системы семейства Windows, а именно: XP, 2000, Server 2003, Vista, Server 2008 и 7. В качестве критериев состояния системы выбраны статистические параметры, на которые в большинстве случаев оказывают влияние вредоносные программы. Все параметры разбиты на четыре группы:

- Связанные с работой файловой системы (создание/модификация исполняемых файлов, создание файлов с нестандартными именами, создание файлов с расширениями, не соответствующими заголовку и т.п.).
- Связанные с реестром операционной системы (модификация и удаление ключей, в том числе ключей, гарантированно не связанных с наблюдаемым процессом).
- Связанные с сетью (обращение к устройству \\DEVICE\\TCP и т.п.)
- Связанные с запуском и работой других процессов (запуск процессов с уровнем доступа, допускающим его остановку или запись в его адресное пространство, доступ к \\DEVICE\\PHYSICALMEMORY, открытие потоков других процессов, использование хуков, выгрузка системных процессов).

Каждому параметру приписывается вес, характеризующий степень его опасности. Значения весовых коэффициентов определялись в процессе компьютерного эксперимента. Для инициализации системы выбирается гарантированно не зараженная система в виде вектора параметров. В процессе работы системы

подсистема защиты набирает статистику параметров и периодически вычисляет отклонения состояния системы от «здорового». При обнаружении значительных отклонений идентифицируется подозрительное приложение и переводится в карантин. В дальнейшем приложения, попавшие в карантин, проверяются антивирусными средствами.

2.1. Структура системы

Рассмотрим общую структуру системы, алгоритмы работы отдельных модулей, а так же характеристики приложений, использующиеся в процессе выявления подозрительной или вредоносной активности.

Программный комплекс состоит из ядра и нескольких групп модулей, работающих в пользовательском режиме и в режиме ядра с различным уровнем привилегий (рис. 1).

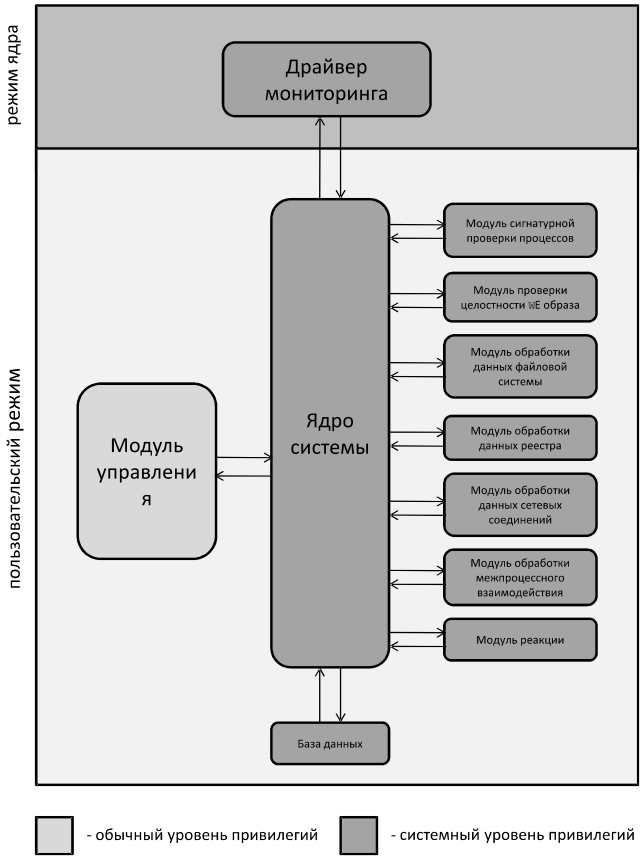

Рис. 1. Структура системы

Рассмотрим более подробно каждый из модулей.

- Ядро. Представляет собой Windows Service, запускаемый с привилегиями SYSTEM в процессе загрузки операционной системы. Основной задачей,

возложенной на ядро, является обеспечение коммуникации между драйвером мониторинга и модулями обработки информации. Так же осуществляется взаимодействие с GUI-модулем управления, позволяющее изменять настройки системы в процессе работы.

- Драйвер мониторинга. Представляет собой WDM Kernel mode driver, включающий функции сбора различных проявлений активности процессов, а именно: работу с файловой системой и устройствами, сетевую и межпроцессную активность. Также отвечает за создание и обновление таблицы процессов, которая требуется для выявления скрытых и замаскированных процессов. В целях повышения производительности и стабильности работы системы, а так же упрощения отладки на стадии разработки обработка собранной информации осуществляется отдельными модулями, функционирующими в пользовательском режиме.

- Модули обработки данных. Занимаются сортировкой информации, получаемой от драйвера мониторинга. Именно здесь принимается решение о том, является ли активность приложения или процесса подозрительной.

- GUI-модуль управления. Является Windows Forms приложением, позволяющим пользователю или администратору производить тонкую настройку системы. К ключевым возможностям относятся:

  – загрузка и выгрузка модулей;

  – изменение чувствительности системы;

  – возможность создавать правила и исключения обработки активности для отдельных процессов и групп процессов.

  Связь модуля управления и ядра системы осуществляется посредством зашифрованного TCP/IP соединения, поэтому настройка может производиться как локально, так и удаленно.

- Модуль реакции. Отвечает за остановку работы процесса в случае признания деятельности последнего вредоносной.

- База данных. Хранит статистическую информацию о деятельности всех процессов системы.

Все компоненты, за исключением драйвера мониторинга, реализованы на языке C# с использованием .NET Framework версии 4.0. Для написания драйвера использовалась комбинация языков C++ и Assembler.

2.2. Процесс загрузки системы

Рассмотрим пошаговую загрузку системы:

1. Windows загружает драйвер мониторинга.

2. Драйвер мониторинга загружает ядро системы и приступает к процедуре инициализации, включающей в себя следующие шаги:

   - составление таблицы процессов с использованием разных методик;
   - установка обработчиков на всевозможные системные события (I/O события, создание сокет-соединений и т.п.).

3. Ядро системы приступает к процедуре инициализации, включающей в себя:

   - проверка целостности модулей путем сверки контрольных сумм;
   - чтение настроек из конфигурационных файлов;
   - загрузку модулей.

2.3. Построение таблицы процессов

Одной из важных характеристик процесса является его видимость для пользователя в диспетчере задач. Практически для всех процессов, за исключением нескольких системных, скрытость воспринимается как негативная характеристика. Для эффективного выявления скрытых процессов нам требуется составление собственной версии таблицы процессов. Ввиду достаточно большого количества вариантов сокрытия процесса в данной работе одновременно используется несколько различных способов получения списка процессов, реализованных как в пользовательском режиме, так и в режиме ядра.

В пользовательском режиме использовались следующие способы получения списка процессов:

- ToolHelp API (вызов функции CreateToolhelp32Snapshot);
- Native API (вызов функции ZwQuerySystemInformation);
- использование списка открытых хендлов;
- для процессов, имеющих окна, можно использовать GetWindowThreadPro-cessId;
- использование прямого системного вызова.

В режиме ядра использовались следующие способы получения списка процессов:

- использование ZwQuerySystemInformation;
- анализ данных структуры EPROCESS;
- анализ списков потоков планировщика;
- перехват системных вызовов;

- анализ списка таблиц хэндлов;
- сканирование PspCidTable;
- перехват SwapContext.

Скрытые процессы выявляются путем сравнения данных, полученных при помощи вышеописанных вариантов, с возвращаемыми функцией CreateToolhelp32Snapshot, выбранными в качестве эталонных.

2.4. Сигнатурное сканирование

Перед запуском процесса система производит сканирование исполняемого файла на предмет наличия известных вирусных сигнатур. В случае обнаружения совпадений запуск процесса блокируется. Стоит отметить, что разработка сигнатур до сих пор является процессом, тяжело поддающимся автоматизации [4]. Виной тому нарастающий полиморфизм и метаморфизм вирусов и червей, что делает синтаксические сигнатуры бессмысленными. Таким образом, для поддержания эффективности сигнатурных сканеров разработчики вынуждены помещать в базу десятки сигнатурных кодов, ассоциированных с одним и тем же вирусом. Очевидно, что проблема автоматизации выделения конкурентоспособных сигнатур достойна отдельного исследования и выходит за рамки поставленных в данной работе задач. Однако полностью отказаться от использования сигнатурного метода обнаружения вредоносного кода невозможно, ввиду его высокой эффективности, применительно к давно известным угрозам. Для повышения общей защищенности системы были использованы сигнатурные базы открытого антивируса CalmAV.

2.5. Целостность PE образа

Для выявления замаскированного вредоносного кода, перед запуском приложения, система производит проверку структуры исполняемого файла. Проверка состоит из следующих шагов:

- анализ таблицы импорта;
- анализ секции данных (проверка на наличие кода);
- проверка избыточности циклической суммы (CRC32);
- эмуляция загрузки файла с последующей проверкой соответствия структуры образа в памяти.

2.6. Работа с файловой системой и сетью

Для регистрации всех обращений к файловой системе в драйвер мониторинга был включен функционал драйвера фильтра файловой системы, основной задачей которого является перехват IRP-пакетов с командами IRP_MJ_CREATE.

Тот факт, что драйвер фильтра занимает в иерархии более высокий уровень, нежели драйвер файловой системы, позволяет ему модифицировать поток между приложениями и драйвером файловой системы.

Для регистрации сетевых соединений используется перехват события IRP _MJ _DEVICE_CONTROL для системных устройств \\DEVICE\\RAWIP, \\DEVICE \\UDP, \\DEVICE \\TCP, \\DEVICE\\IP.

Как уже было сказано выше, обработка данных, получаемых драйвером, осуществляется в отдельных модулях, работающих в пользовательском режиме.

3. Полученные результаты

Тестирование работоспособности системы проведено на серии известных вредоносных программ. Для сравнения использовались две идентично сконфигурированные виртуальные машины: VirtualBox, Windows XP SP2 с включенным стандартным сетевым экраном. Обеим машинам был обеспечен доступ в сеть Internet.

В процессе тестирования одна из машин все время оставалась «здоровой», другая «заражалась» через некоторое время после начала эксперимента. Ниже приведены сравнительные графики сетевой активности, а также интенсивности обращения к жесткому диску, для «здоровой» и «зараженной» машин.

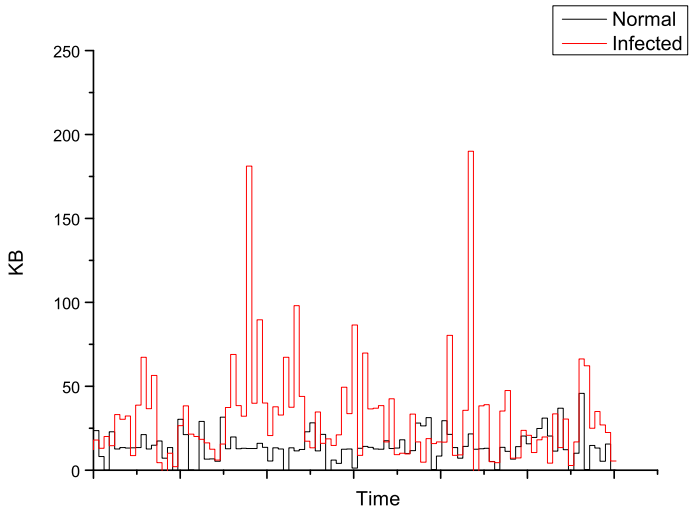

Рис. 2. Сетевая активность

В качестве известной вредоносной программы был использован троянский конь Trojan-Downloader.Win32.Agent.bet (здесь и далее названия приведены в соответствии с терминологией лаборатории Касперского). Как видно из графиков (рис. 2 и рис. 3), «зараженная» машина отличается повышенной сетевой активностью, а также увеличением загрузки центрального процессора. Кроме того, были зафиксированы попытки сокрытия процесса, попытки записи в си-

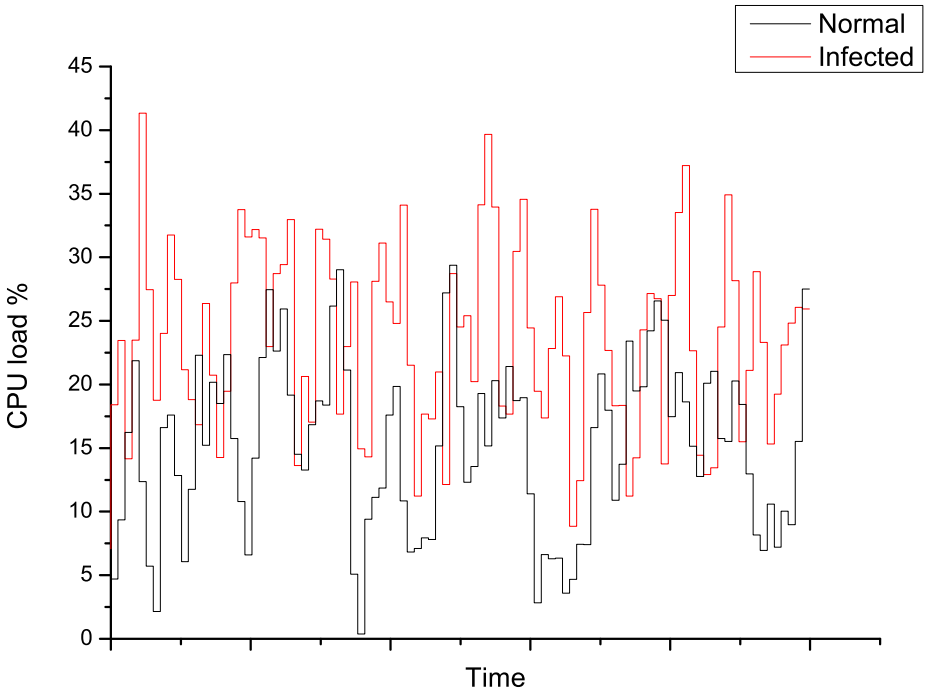

Рис. 3. Загрузка центрального процессора

стемные каталоги и реестр процессом, не производившим подобные действия во время обучения.

3.1. Ошибки второго рода

Большинство случаев несрабатывания системы относится к ситуациям, когда заражение приложения происходит в результате переполнения буфера (речь идет о приложениях и процессах, имеющих доступ к сети). В качестве примера можно привести поведение Trojan-Ransom.Win32.Hexzone.aer, прдставляющего собой Internet Explorer Browser Helper Object (BHO). Ситуация усугубляется наличием у некоторых процессов привилегий, необходимых для работы с реестром, что позволяет вредоносному коду закрепиться в системе.

К трудно обнаружимым вторжениями также относятся те, в которых проявления вредоносной программы минимальны (например бэкдоры, способные сохранять минимальную сетевую активность до получения команд извне). Ниже приводится сравнительный график сетевой активности для «здоровой» машины, а также машины, зараженной троянским конем Backdoor.Win32.VB.bdg (рис.4).

Однако серия успешно проведенных экспериментов позволяет говорить о высокой эффективности используемой модели. На данный момент число успешных индикаций зараженного состояния составляет не менее 73%.

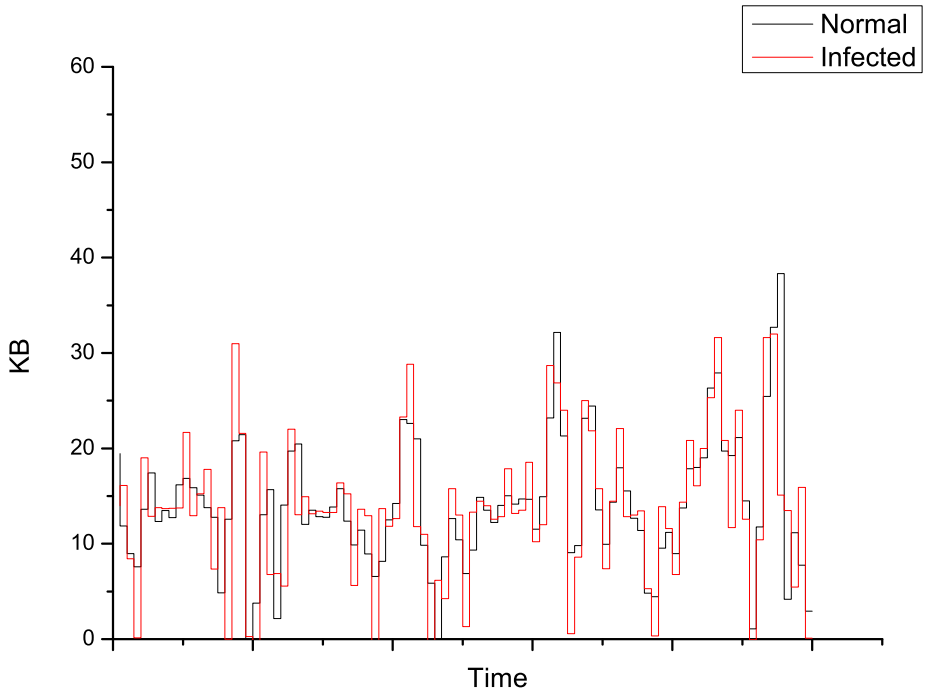

Рис. 4. Сетевая активность

Литература

1. Tarakanov A. O., Skormin V. A., Sokolova S. P. Immunocomputing: Principles and Applications. NY: Springer, 2003.
2. Иммунология / Хаитов Р.М. [и др.]. М.: Медицина, 2009. 320 с.
3. Дасгупта В. Искусственные иммунные системы. М.: Физматлит, 2006. 344 с.
4. Newsome J., Karp B., Song D. Polygraph: Automatically Generating Signatures for Polymorphic Worms. URL: http://www.ece.cmu.edu/ dawnsong/papers/polygraph.pdf (дата обращения: 01.02.2010).
5. Платонов В.В. Программно-аппаратные средства обеспечения информационной безопасности вычислительных сетей. М.: Академия, 2007. 240 с.
6. Руссинович М., Соломон Д. Внутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP, Windows 2000. Санкт-петербург: Питер, Русская редакция, 2008. 992 с.
7. Peakman M., Vergani D. Basic and clinical immunology. NY: Churchill Livingstone, 1997.
8. Shibli A., Mwakalinga J., Muftic S. MagicNET: The Human Immune System and Network Security System // IJCSNS. 2009. V. 9, N. 1. P. 87–94.
9. Automatic Virus Analysis in the Digital Immune System. IBM T. J. Watson Research Center, 2000. URL: http://www.scribd.com/doc/24058575/Automatic-Virus-Analysis-in-the-Digital-Immune-System (дата обращения: 10.08.2009).