Санкт-Петербургский государственный
университет
Математико-механический факультет
Кафедра системного программирования

Интерактивный антируткит

Магистерская диссертация студента 661 группы
Королева Дмитрия Николаевича

Научный руководитель	..... /подпись/	ст. преп. Ю.А. Губанов
Рецензент	..... /подпись/	зав. лаб. РУНЦ И. В. Зеленчук
“Допустить к защите” заведующий кафедрой	..... /подпись/	д.ф.-м.н., проф. А.Н. Терехов

Санкт-Петербург
2012

St. Petersburg State University
Faculty of Mathematics and Mechanics
Chair of Software Engineering

Interactive Anti-rootkit

Master's graduation paper by

Korolev Dmitry

Supervisor	.....	U. A. Gubanov
Reviewer	.....	I. V. Zelenchuk
"Approved by" Head of Chair	.....	Professor A. N. Terekhov

St. Petersburg
2012

Оглавление

1. Введение

Термин «руткит» (rootkit) существует уже более 15 лет [1]. Руткиты представляет собой группу небольших и полезных программ, позволяющих разработчику сохранять доступ пользователя «root» (наиболее привилегированного пользователя операционной системы). Книга «Руткиты. Внедрение в ядро Windows» [2] дает следующее определение этого термина:

«**Руткиты** – это набор программ, обеспечивающих постоянное, устойчивое и неопределяемое присутствие на компьютере».

Следует понимать, что руткиты – это всего лишь техники обхода системных механизмов защиты и техники сокрытия в системе. Как и любая мощная технология, они могут быть использованы как во вред, так и во благо. Например, корпорация Sony встраивала подобие руткита в свои лицензионные аудиодиски для защиты от копирования [3]. Большинство антируткитов также используют руткит-технологию.

Как правило, многие разработчики вредоносного ПО используют руткиты в совокупности со всевозможными вирусами, троянами и червями. В связи с этим, появилось множество средств для обнаружения руткитов. С точки зрения принципов детектирования их можно разделить на следующие категории:

- брандмауэр [4];
- системы обнаружения вторжений (Intrusion-Detection System, IDS) [5];
- системы предотвращения вторжений (Intrusion-Prevention System, IPS) [6] .

В Приложение 1. Категории антируткитов приведено детальное описание этих категорий.

Общий принцип последовательности действий большинства инструментов такого рода следующий:

1) поиск подозрительных действий программ;
2) проверка наличия сведений о данном типе активности;

3) при нахождении известного типа активности – принятие контрмер по ее устранению, иначе – либо ничего не делает, либо блокирует дальнейшее выполнение программы.

При таком подходе существует ряд недостатков.

1. Статический алгоритм поиска «подозрительных действий». Большинство же руткитов доступны в свободном доступе, и каждый желающий при должном усердии может на его основе создать новую недетектируемую модификацию.
2. В случае новых модификаций, компьютер уязвим до тех пор, пока не получит обновление алгоритмов с сервера.
3. Нет возможности детально проанализировать подозрительные действия программ.

Описанный подход хорош для конечного пользователя, которому нет необходимости вдаваться в детали происходящего, и бесполезен для специалистов, занимающихся поиском и анализом новых видов руткитов. Для такого рода деятельности им нужны инструменты, позволяющие находить необычные особенности поведения компьютерных программ и помогать их анализировать.

2. Постановка задачи

Перед автором дипломной работы были поставлен следующий список задач.

1. Провести обзор подходов к поиску руткитов на компьютерах пользователей.
2. Провести сравнительный анализ существующих инструментов для анализа руткитов.
3. Выявить набор признаков присутствия руткитов.
4. Реализовать инструмент, позволяющий детектировать эти признаки и предоставляющий возможность проводить дальнейший анализ аномалий.
5. Убедиться в его работоспособности на существующих видах руткитов.

3. Обзор

В этом разделе проводится обзор существующих видов руткитов, а также рассказывается о техниках их обнаружения.

3.1. Классификация руткитов

В мире Windows в основе работы руткитов лежит модификация данных и кода программы в памяти операционной системы [2]. В зависимости от метода воздействия на систему их можно разделить на две группы [2]:

- изменяющие алгоритмы выполнения системных функций (Modify execution path);
- изменяющие системные структуры данных (Direct kernel object manipulation).

Классификацию руткитов можно рассмотреть и с точки зрения области памяти, с которой они работают. При таком подходе их можно разделить на следующие группы.

1. Пользовательский режим

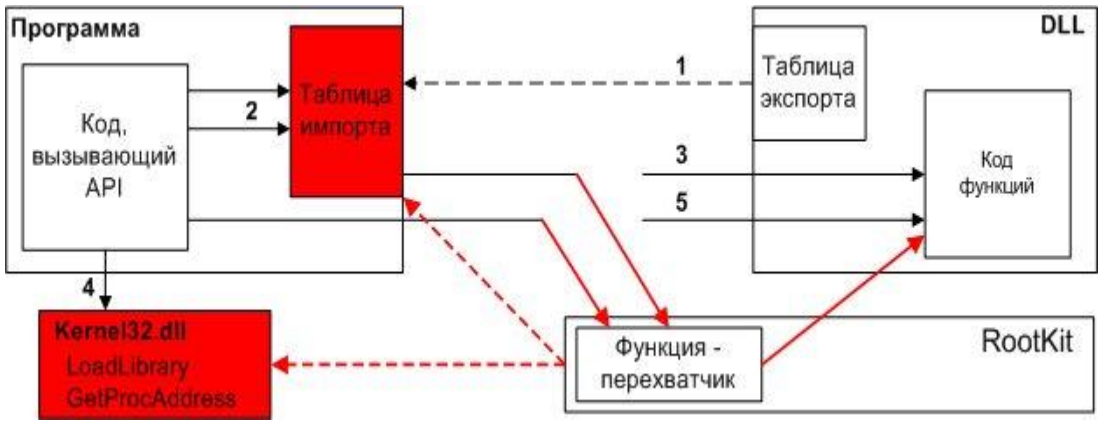

Рисунок 1. Руткит пользовательского режима

Руткиты исполняются в непривилегированном кольце 3 (с точки зрения архитектуры информационной безопасности [7]). Они используют программные расширения (например, для проводника Windows), перехват сообщений, отладчики, эксплуатируют уязвимости в безопасности, а также производят перехваты функций (function hooking) широко

используемых API (в памяти каждого отдельного процесса). На Рисунок 1 изображен пример работы руткита пользовательского режима.

2. Режим ядра

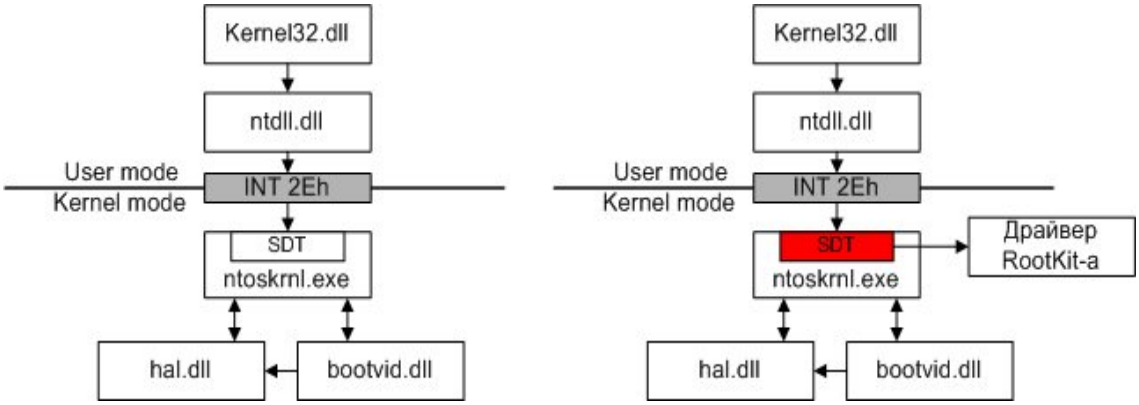

Рисунок 2. Руткит режима ядра

Руткиты исполняются в привилегированном нулевом кольце (наивысший уровень привилегий ОС [7]). Они могут встраиваться в драйверы устройств, проводить прямую модификацию объектов ядра (DKOM), а также влиять на взаимодействие между пользовательским режимом и режимом ядра. На Рисунок 2 изображен пример работы руткита режима ядра.

3. Загрузчики

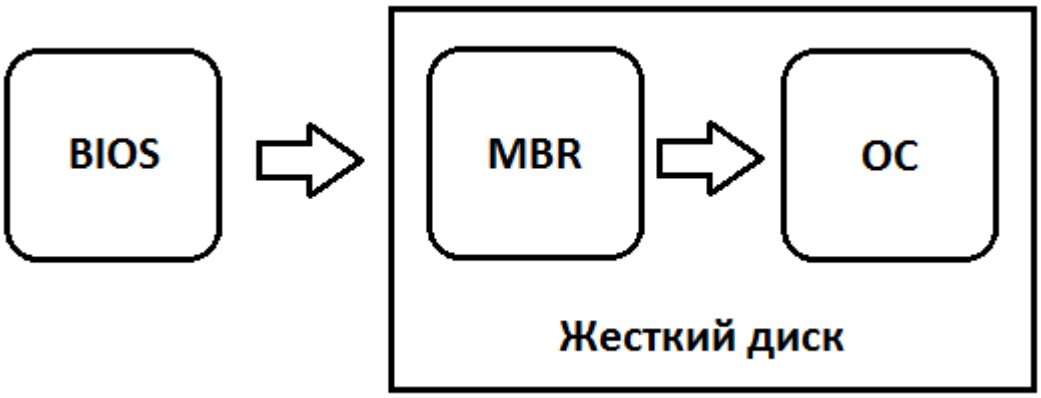

Рисунок 3. MBR

На этом уровне руткиты производят модификацию главной загрузочной записи (master boot record).

4. Режим гипервизора

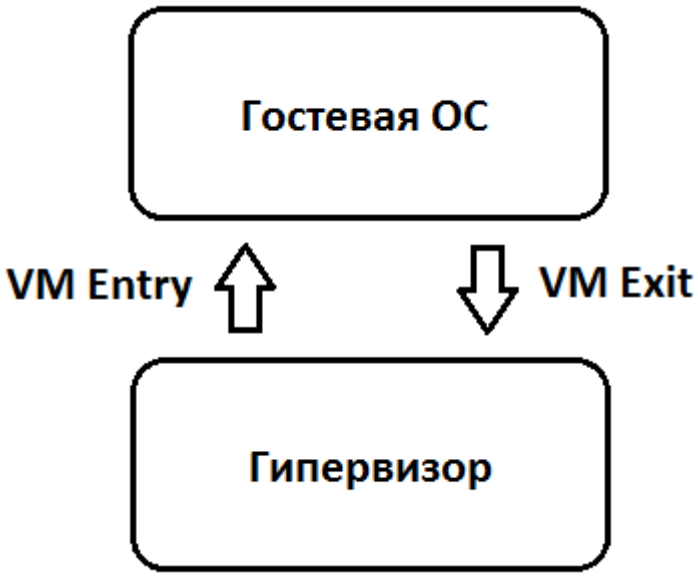

Рисунок 4. Гипервизор

Некоторые современные процессоры могут предоставлять дополнительный режим работы, известный как режим гипервизора (Hypervisor mode). По сути, гипервизор обычно является небольшим ядром, которое управляет распределением ресурсов между несколькими операционными системами и работает уровнем ниже, чем сами операционные системы (Рисунок 4). В силу этого в терминологии x86 данный режим как правило называют кольцом −1 (Ring −1). На этом уровне руткиты перехватывают аппаратные вызовы оригинальной ОС. На данный момент нет известных реализаций данного подхода в «живую» (существуют только демонстрационные версии [8]).

5. Прошивки

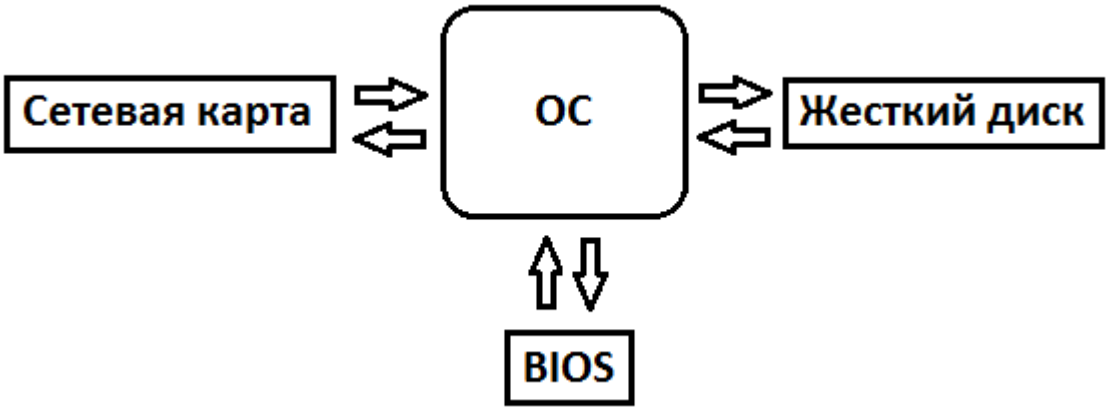

Рисунок 5. Прошивки

Руткиты данного типа создают постоянный образ в оборудовании, таком как сетевая карта, жесткий диск или BIOS (Рисунок 5).

3.2. Классификация методов обнаружения руткитов

Разработчики руткитов делают все возможное, чтобы их программы было невозможно обнаружить. В свою очередь, разработчики антируткитов находят новые техники для обнаружения вредоносного ПО. В целом, все методики обнаружения руткитов можно разделить на две большие группы:

- анализ работающей ОС;
- анализ дампов памяти.

Дамп памяти (англ. memory dump) — содержимое рабочей памяти одного процесса, ядра или всей операционной системы. В этом случае обнаружение руткитов проводится на выключенном компьютере. Техники применимые к анализу дампов памяти также применимы и к анализу работающей ОС.

Следующие техники применимы только к анализу работающей ОС.

1. Альтернативное доверенное окружение

Лучший и самый надежный способ для обнаружения руткитов уровня ОС – выключить потенциально зараженный компьютер и проверить его

память загрузкой с альтернативного доверенного источника (например, загрузочный компакт-диск или USB флэш-накопитель). Техника является эффективной, так как руткит не может скрыть свое присутствие, когда он не запущен.

2. Поведенческий метод

   Данный подход обращает внимание на поведение процессов ОС. Для выявления руткитоподобных действий предполагается наблюдение за системой и отслеживание времени и частоты вызовов API-методов или в целом за использованием процессора. Так, например, записи межсетевого экрана или IPS позволяют отслеживать поведение руткитов в сети.

3. Аппаратный метод

   Данный подход подразумевает использование стороннего аппаратного средства для анализа потенциально зараженного компьютера путем использования технологии DMA [9]. Прямой доступ к памяти (англ. Direct Memory Access, DMA) — режим обмена данными между устройствами или же между устройством и основной памятью (RAM) без участия центрального процессора. Главное преимущество данного метода в том, что с его помощью можно анализ работающей системы, не полагаясь на потенциально скомпрометированную ОС.

4. Гипервизор

   Подход заключается в проверке всего исполняемого программного кода, прежде чем выполнить его на реальном процессоре [8].

Следующие техники применимы к обоим подходам.

1. Метод сигнатур

   Используются характерные признаки руткитов. Большинство современных антивирусов, сканеров уязвимостей и систем обнаружения вторжений (СОВ) используют «синтаксические» сигнатуры, взятые непосредственно из тела атаки (файла или сетевого пакета, принадлежащего зловреду). Также существуют сигнатуры, основанные на

поведении или аномалиях — например, слишком агрессивное обращение к какому-либо сетевому порту на компьютере.

2. Детектирование перехватов

   ОС Windows использует таблицы указателей для переадресации запросов. Так, например: IDT (Interrupt Descriptor Table), SSDT (System Service Descriptor Table), IAT (Import Address Table). Для изменения поведения программ руткит может либо подменить эти таблицы или изменить некоторые указатели в них на свои функции-обработчики.

3. Сравнение данных из различных источников

   Метод заключается в получении одной и той же информации о системе из разных источников. Самый простой пример использования такого подхода это: сравнение результатов, получаемых с использованием высокоуровневого и низкоуровневого API. Другим примером может служить сравнение бинарных файлов на жестком диске с их загруженными копиями в оперативную память ОС (по этому принципу работает программа RootkitRevealer [10]).

   К сожалению, такая техника не указывает на точное место перехвата, зато помогает найти спрятанные ресурсы.

4. Проверка целостности

   Для системных библиотек вычисляется их цифровая подпись (например, с использованием криптографической хэш-функции), и затем, в ходе работы ОС, эти библиотеки проверяются на наличие изменений в их коде. По этому принципу работает программа Tripwire [11].

3.3. Сравнительный анализ существующих инструментов для анализа руткитов

В данном разделе рассматриваются программы, которые помогают в обнаружении руткитов и предоставляют полезную информацию для их дальнейшего изучения.

RootkitRevealer [6]

Программа создана Марком Руссиновичем с целью выявления скрытых ключей реестра и файлов. Принцип ее работы заключается в следующем: во временном каталоге создается и запускается исполняемый файл со случайно выбранным именем и размером. В свою очередь, этот файл при запуске извлекает и загружает из своей секции ресурсов драйвер нулевого кольца (RKREVEAL150.SYS). Благодаря такой последовательности шагов, разработчикам руткитов сложнее скрыть свое «детище» от этой прогаммы.

RootkitRevealer использует методику «Сравнения данных из различных источников». Для поиска скрытых ключей реестра RootkitRevealer осуществляет обработку файлов, полученных путем снятия образа веток реестра, используя низкоуровневую функцию Win API «ZwSaveKey». Далее эти файлы обрабатываются и сравниваются со списком, полученным через высокоуровневую функцию Win API «RegOpenKey». Поиск скрытых файлов осуществляется путем сравнения списка файлов полученного путем вызова API функций со списком файлов полученных через командный интерпретатор с использованием команды «DIR».

Однако, в случае перехвата низкоуровнего API программа ничего не обнаружит.

Process hunter [8]

Как и в случае RootkitRevealer, Process hunter использует методику «Сравнения данных из различных источников», но данная программа работает не с регистрами и файлами, а с процессами.

F-Secure BlackLight Rootkit Eliminator [12], UnHackMe [13]

Данные программы практически ничем не превосходят по своим возможностям RootkitRevealer и Process Hunter, и имеют схожее назначение – поиск скрытых файлов, ключей реестра и процессов, но существенно

проигрывают в защищенности своих процессов. Программы запускают свои процессы со статическими именами и размерами, что делает их легко детектируемыми на компьютере пользователя, что упрощает работу разработчикам руткитов по их обходу.

IceSword [14]

Отличительными особенностями данной программы являются ведение журнала создания и завершения процессов, что бывает весьма полезно, особенно в тех случаях, когда вредоносная программа работает короткое время, также полезная функция – просмотр программ, установивших перехваты IRP пакетов.

System Virginity Verifier [15]

Программа позволяет определить перехват API функций, как в пользовательском режиме, так и в режиме ядра путем побайтного сравнения системного файла-образа на диске с его копией в памяти. System Virginity Verifier способен выявлять также перехваты, установленные путем перезаписи кода системных сервисов, установку int 3 и модификацию IDT.

GMER [16]

Программа отображает скрытые процессы и сервисы, выводит список скрытых файлов, ключей реестра и драйверов. Кроме того, GMER осуществляет мониторинг создания процессов, загрузки драйверов и библиотек, использования файлов, изменений реестра, активность TCP/IP-соединений.

The Volatility Framework [17]

Программа представляет собой коллекцию инструментов для извлечения информации из статических образов оперативной памяти

компьютера. В ее арсенале имеются команды для поиска перехватов в системных таблицах (SSDT, IDT, IAT), скрытых процессов, изменений кода функций. Программа распространяется с лицензией GPL и имеет хорошую документацию.

Описанные выше программы используют различные техники обнаружения руткитов, и среди них нет универсальных инструментов для детектирования и анализа всех видов руткитов. Наиболее полезную информацию для дальнейшего анализа предоставляют IceSword (журналы создания/завершения процессов), System Virginity Verifier (указатели на перехваченные функции) и The Volatility Framework (артифакты в образе памяти). Все программы, кроме последней, предназначены для обнаружения руткитов на работающей системе.

4. Решение

В этом разделе приводится описание решения, примененного в данной работе

4.1. Выбор вспомогательного инструмента

За основу создаваемого решения был взят отладчик Windows WinDbg.

WinDbg – многоцелевой отладчик для Microsoft Windows, поставляемый компанией Microsoft. Он может быть использован для отладки пользовательских приложений, драйверов и самой ОС в режиме ядра. Приложение имеет удобный графический интерфейс.

Имеется три возможности использования WinDbg:

1) непосредственно на работающей ОС (на одном компьютере);
2) с использованием двух компьютеров (которые могут быть виртуализированы);
3) для анализа аварийных образов памяти.

Все три подхода имеют свои достоинства и недостатки.

На работающей ОС

Самый простой подход, но руткит может помешать запуску процесса отладки.

С использованием двух компьютеров

Таким образом, получается связка: сервер (машина, на которой установлен отладчик) и целевая машина (машина, подлежащая анализу). Таким образом, процессу отладки на сервере руткит помешать не может, и остается сконцентрироваться на анализе целевой машины.

Недостатком этого варианта является необходимость запуска целевой машины в «отладочном» режиме, о котором известно самой ОС, а также приложениям (на уровне ядра) в ней. Руткит может определить факт работы ОС в отладочном режиме прочитав глобальную переменную KD_DEBUGGER_ENABLED. Кроме того, руткит имеет возможность определить наличие подключенного к ОС отладчика через глобальную

переменную KD_DEBUGGER_NOT_PRESENT. Все это предоставляет руткиту возможность избежать своего обнаружение средствами отладчика.

Для анализа аварийных образов памяти

Аварийный образ оперативной памяти создается при возникновении, так называемого, «синего экрана смерти» (BSOD), который появляется при серьезных ошибках в Windows.

Таким образом, проводится анализ статического набора данных («посмертная» отладка), благодаря чему руткит может быть обнаружен по его образу в оперативной памяти компьютера.

Недостатком данного подхода является невозможность наблюдения за руткитом в действии.

4.2. Описание выбранной техники детектирования для применения в решении

Из рассмотренных выше подходов использования отладчика WinDbg, «анализ аварийных образов памяти» единственный гарантирует невозможность руткита скрыться от отладчика. В данной работе было решено использовать именно этот подход.

В качестве дампов памяти рассматривается четыре типа:

- аварийный образ памяти;
- «сырой» образ (Raw dump);
- hiberfil.sys;
- Vmem.

Аварийный образ памяти – файл, создаваемый при возникновении неустранимой ошибки, в который записывается содержимое системной памяти.

«Сырой» образ – файл, содержащий полный образ оперативной памяти операционной системы.

Hiberfil.sys – файл, который создается ОС для корректной работы режима гибернации. Когда ОС переходит в режим гибернации, все

содержимое оперативной памяти компьютера перемещается на жесткий диск именно в файл hiberfil.sys. Соответственно и объем данного файла будет равняться объему оперативной памяти. Нужно отметить, что часть оперативной памяти, находящаяся в этот момент на жестком диске не попадает в этот файл.

Vmem – файл, поддерживаемый системой виртуализации VMware [18]. В нем содержится оперативная память работающей виртуальной машины.

WinDbg работает только с образом памяти специального формата – аварийный образ памяти. Все вышеперечисленные типы образов памяти конвертируются в аварийный образ памяти, с помощью утилиты Moonsols Windows Memory Toolkit Community Edition [19].

4.3. Методы обнаружения руткитов в дампе памяти Windows при помощи WinDbg

В приведенном выше обзоре были описаны некоторые методы обнаружения руткитов в ОС. Далее рассматриваются некоторые из них применительно к образу памяти ОС.

ОС Windows содержит несколько таблиц указателей для переадресации запросов. Так, например: IDT, SSDT, IAT.

Как было сказано в обзоре, руткит может изменить значение указателей в этих таблицах или подменить сами таблицы.

4.3.1. IDT

Когда приложение пользовательского режима вызывает какую-нибудь системную функцию (выполняющуюся в режиме ядра), для ее выполнения необходимо повысить уровень привилегий. Для этого происходит программное прерывание.

Системные вызовы в Windows NT инициируются исполнением инструкции «int 2e». Команда «in»' заставляет процессор сгенерировать программное прерывание, то есть проследовать в IDT по индексу 2e и

считать находящийся там данные. После этого процессор установит значение указатель на смещение точки входа в процедуру обслуживания прерывания, указанное в таблице. Для обслуживания системных вызовов используется функция диспетчера сервисов (KiSystemService).

Команда WinDbg «!idt 2e» показывает указатель на процедуру обработки. Если указатель установлен на «KiSystemService», то все нормально, иначе – обнаружен перехват.

4.3.2. Изменение кода системных функций

Для того, чтобы изменить ход выполнения программы не обязательно подменять адреса функций, можно изменить сами функции таким образом, чтобы они выполняли необходимые действия, или переадресовывали на нужный сегмент кода. В арсенале WinDbg есть специальная команда, предназначенная для обнаружения измененных в памяти функций путем сравнения их с копиями на символьном сервере. Чтобы проверить на наличие изменений системные функции ОС достаточно воспользоваться командой «!chkimg -d nt». Результатом исполнения будет список найденных различий. Результат исполнения команды выглядит следующим образом:

```
kd> !chkimg -d nt

804d90c9-804d90cd 5 bytes - nt!KiXMMIZeroPage+30
[ fa f7 80 0c 02:e9 08 82 c7 00 ]
804d910c - nt!KiXMMIZeroPage+73 (+0x43)
[ fb:90 ]
804d9112-804d9115 4 bytes - nt!KiXMMIZeroPage+79 (+0x06)
[ 57 ff ff ff:6d c6 c1 00 ]
804d9545-804d954a 6 bytes - nt!ExAcquireResourceSharedLite+10 (+0x433)
[ fa 8b 75 08 33 db:e9 a3 c2 c1 00 cc ]
804d9564 - nt!ExAcquireResourceSharedLite+98 (+0x1f)
[ fb:90 ]
```

```
.....some entries were left out.....

804ed809-804ed80f 7 bytes - nt!CcGetActiveVacb+5 (+0x4670)
[ fa 8b 45 08 8b 48 48:e9 ee 7f c0 00 cc cc ]
804ef1dc-804ef1e3 8 bytes - nt!CcSetActiveVacb+7 (+0x19d3)
[ fa 8b 45 08 83 78 48 00:e9 70 66 c0 00 cc cc cc ]
804ef1ff-804ef20c 14 bytes - nt!CcSetActiveVacb+a3 (+0x23)
[ 8b 0a 89 48 48 89 58 50:e9 3d 66 c0 00 e9 2c 66 ]
804f0c70-804f0c73 4 bytes - nt!ExAcquireSharedStarveExclusive+7 (+0x1a71)
[ 64 a1 24 01:e9 cf 05 c6 ]
805745a3-805745a7 5 bytes - nt!NtOpenProcess+5
[ 68 60 a5 4e 80:e9 d8 b4 7b 78 ]
161 errors : nt (804d90c9-805745a7)
```

Листинг демонстрирует, что эти адреса памяти и соответственные функции были модифицированы.

4.3.3. Поиск скрытых загруженных в память модулей

Некоторые руткиты загружают образы своих программ непосредственно в память ОС.

Исполняемые модули в операционной системе Windows имеют формат MZ-PE. Его можно идентифицировать по символам «MZ» в самом начале модуля. С помощью WinDbg можно произвести поиск по образу памяти ОС на наличие этой последовательности символов. Для этого нужно воспользоваться следующей командой.

```
s -d 0x0 L?0xffffffff 0x00905a4d
```

Команда выведет все найденные строки в образе ОС, начинающиеся с MZ.

```
0:001> s -d 0x0 L?0xffffffff 0x00905a4d
01000000  00905a4d 00000003 00000004 0000ffff  MZ..............
5ad70000  00905a4d 00000003 00000004 0000ffff  MZ..............
```

```
74720000  00905a4d 00000003 00000004 0000ffff  MZ.....
……..
```

Первый столбец отображает адрес исполняемого модуля. Следующие четыре столбца – первые четыре слова в этом образе.

Если исполняемый модуль был загружен в память правильно, то команда «!lmi» выдаст известную системе информацию о модуле:

```
0:001> !lmi 01000000
Loaded Module Info: [01000000]
        Module: calc
   Base Address: 01000000
     Image Name: C:\windows\system32\calc.exe
   Machine Type: 332 (I386)
     Time Stamp: 3b7d8410 Fri Aug 17 15:52:32 2001
           Size: 1f000
       CheckSum: 1d7fc
Characteristics: 10f
Debug Data Dirs: Type  Size     VA  Pointer
            CODEVIEW    19,  160c,     a0c NB10 - Sig: 3b7d8410, Age: 1, Pdb: calc.pdb
    Symbol Type: DEFERRED - No error - symbol load deferred
    Load Report: no symbols loaded
```

В случае же, если модуль был размещен в памяти ОС путем прямого копирования байтов, системе ничего не известно о загруженном модуле и команда «!lmi» выдаст следующий результат:

```
Loaded Module Info: [0xf883ecc0]
ffffffff883ecc0 is not a valid address
```

Чтобы не вызывать команду «!lmi» в ручную для каждого найденного модуля, есть смысл автоматизировать ее вызов путем написания скрипта на встроенном языке WinDbg.

Скрипты представляют собой текстовые файлы, содержащие обычные команды отладчика и управляющие директивы: .if, .else, .elsif, .foreach, .for,

.while, .do, .break, .continue, .catch, .leave, .printf, .block. В качестве переменных могут использоваться псевдорегистры ($tn, где n - целое число), назначаемые командой r, а также алиасы. Для логических выражений есть как MASM-подобный, так и C-подобный синтаксис (в последнем случае выражение обрамляется в конструкцию @@c++(expression)).

```
.foreach ( start { s -[1]d 0x0 L?0xffffffff 0x00905a4d })
{
  .echo start;
  !lmi start;
};
```

Далее следует использовать встроенные команды WinDbg для дальнейшего анализа подозрительной области памяти. Например, можно прочитать размер исполняемого модуля (он находить по смещению 0x140), и выделить его код в отдельный файл (команда «.writemem»).

4.3.4. Перехват функций через SSDT

SSDT (System Service Descriptor Table) – таблица диспетчеризации системных сервисов, содержащая адреса точек входа сервисов ядра NT [20]. На Рисунок 6 показано нормальное выполнение системного сервиса «WriteFile», а на Рисунок 7 выполнение того же сервиса, но руткит уже заменил указатель в SSDT на свою функцию. Таким образом, руткит может контролировать выполнение системного сервиса, отфильтровывая «ненужные» данные.

.

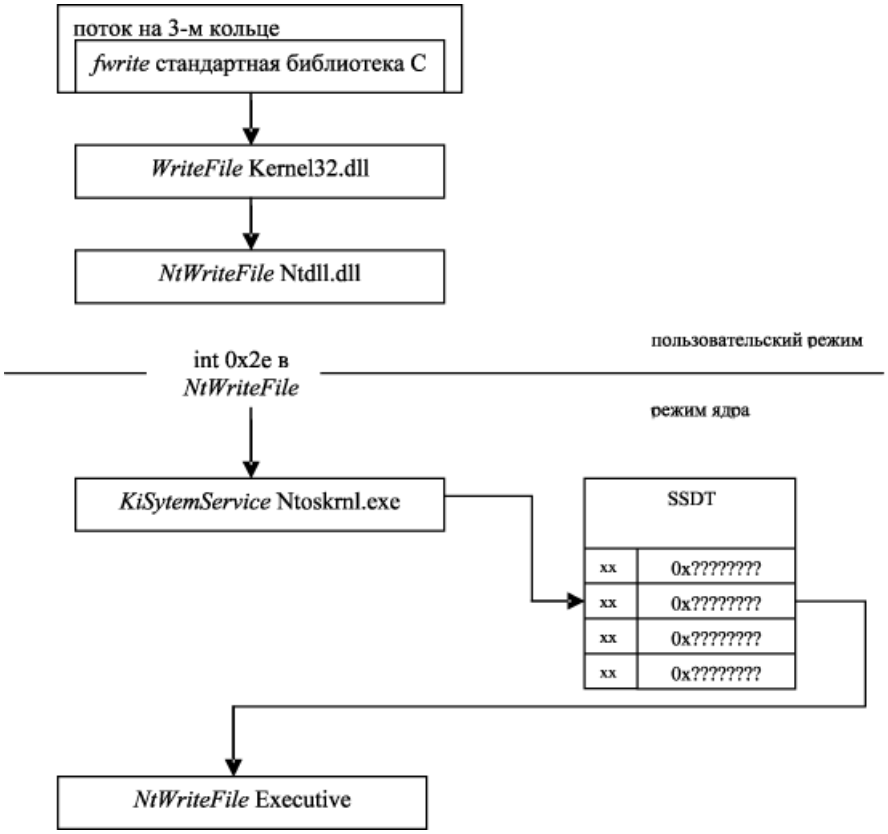

Рисунок 6. Нормальный поток выполнения кода, вызвавшего WriteFile

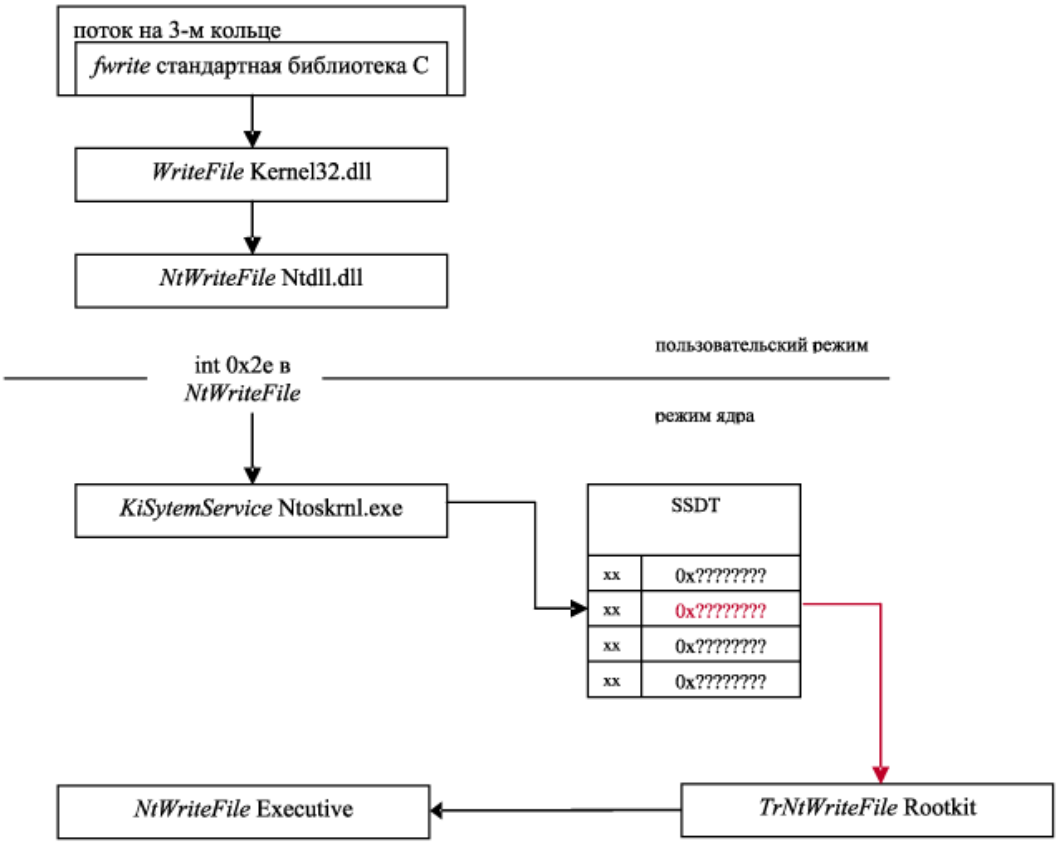

Рисунок 7. Поток выполнения кода, вызвавшего WriteFile из переписанной SSDT, что приводит к активации руткита

Все указатели, расположенные в SSDT должны вести на функции, реализованные либо в библиотеке «nt», либо в «win32k». Значит, для обнаружения перехвата такого рода достаточно проверить попадают ли указатели SSDT в соответствующие области памяти.

Разработан скрипт для автоматизации описанного процесса. Результатом его работы будет список всех указателей SSDT и модулей памяти, в которые они указывают. В случае, если обнаружен указатель, указывающий в стороннюю библиотеку, строчка будет помечена словом «HOOK».

4.3.5. Изменение указателя KTHREAD.pServiceDescriptorTable

Базовая единица, которая выполняется в системе, это поток. Поток в системе представлен структурой ядра KTHREAD (и ее расширением исполнительной системы ETHREAD). Структура (K/E)THREAD сильно варьируется от версии к версии ядра, однако существуют отдельные поля, сохраняющие свой формат и назначение. Так, например поле «ServiceTable». Оно определяет сервисную таблицу для потока. Обычно там лежит адрес KeServiceDescriptorTable или KeServiceDescriptorTableShadow, однако, никто не мешает определить потоку собственную таблицу, которой он будет пользоваться. В случае, если это значение переопределено – это сигнал перехвата.

Ниже на Рисунок 8 показаны два потока. Первый еще не вызывал USER или GDI-сервис, и его «pServiceDescriptorTable» указывает на сервисы, которые обслуживает «ntoskrnl». «PServiceDescriptorTable» второго потока указывает на таблицу дескрипторов, включая дескриптор для обслуживания USER и GDI сервисов – win32k. На Рисунок 9видно, что руткит скопировал старую SSDT, подменил указатели в ней на свои обработчики, и выставил новый указатель на подмененную таблицу дескрипторов в KTHREAD.

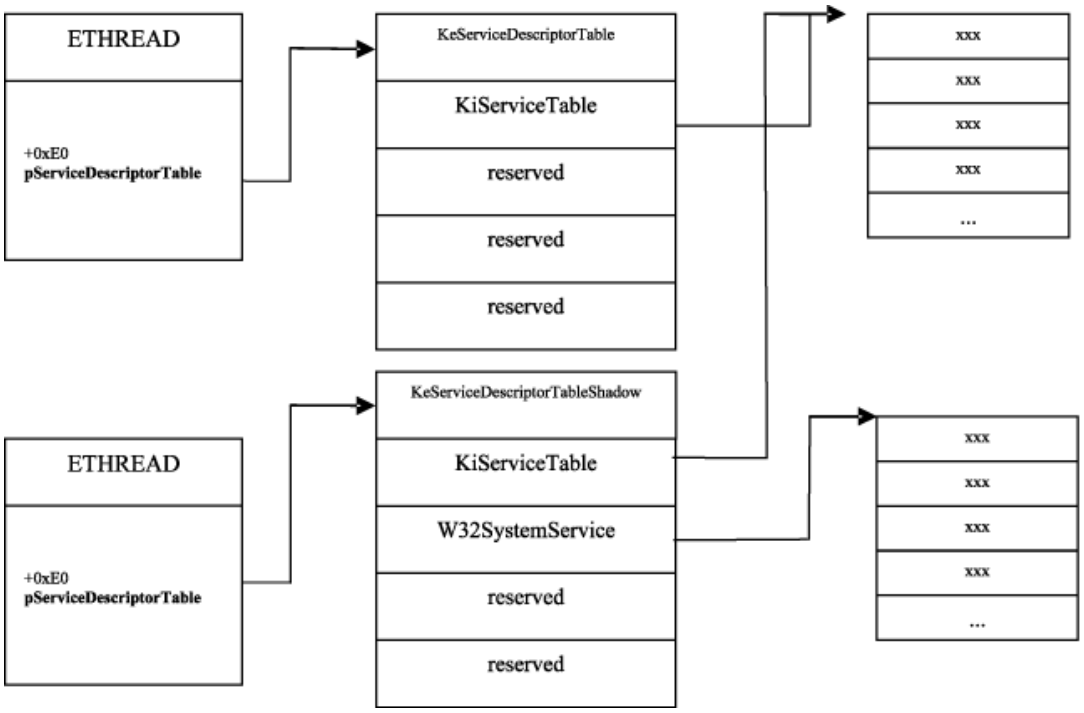

Рисунок 8. В структурах ETHREAD, указатель на таблицу SSDT, указывает на SSDT ядра

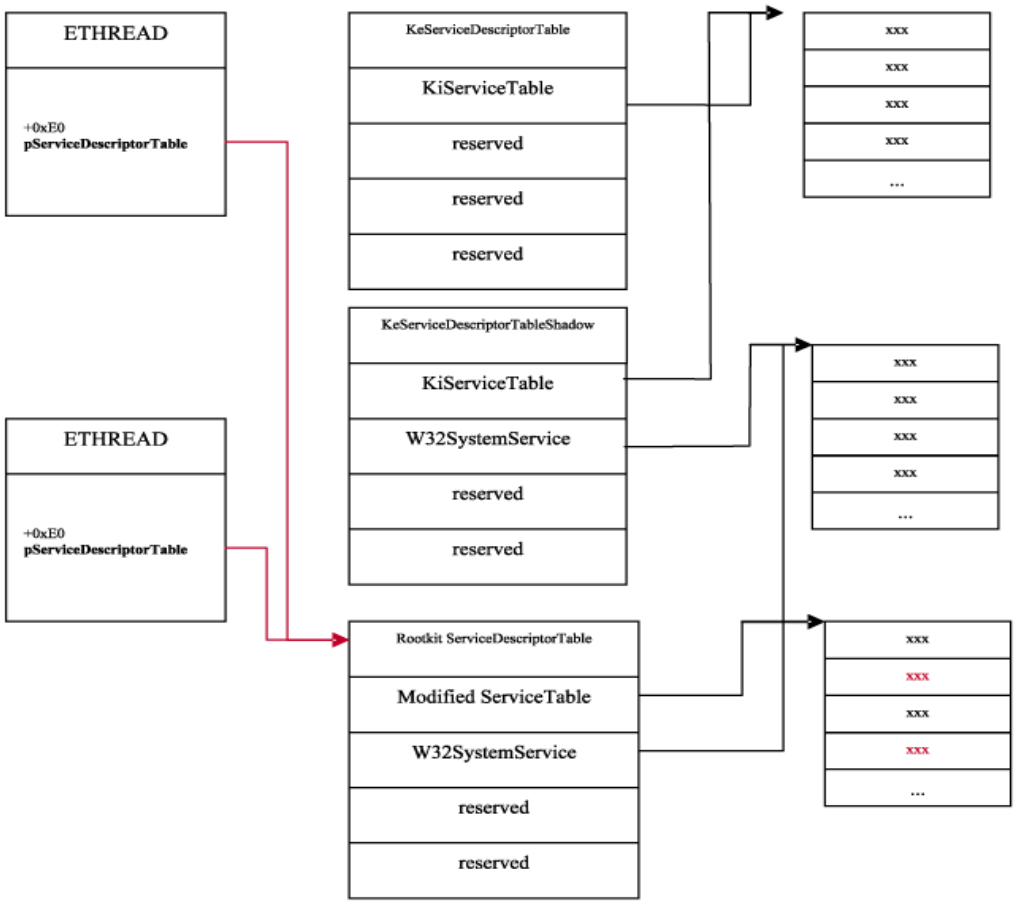

Рисунок 9. В структурах ETHREAD указатель на таблицу SSDT, указывает на SSDT руткита

Для того чтобы это проверить можно пройтись по всем процессам и его потокам и просмотреть значение этого поля. Данный алгоритм автоматизирован с помощью скриптового языка отладчика WinDbg. Результатом его работы будет список всех исполняемых потоков с их указателями на SSDT. Правильность указателей проверяется путем сравнения их значений с результатом, который возвращают системные функции «nt!KeServiceDescriptorTable» (указатель на таблицу SSDT) и «nt!KeServiceDescriptorTableShadow» (указатель на теневую таблицу SSDT) [21]. Если указатель, выставленный в потоке, имеет другое значение, то выдается предупреждение вида «Changed value of KTHREAD::ServiceTable».

4.3.6. Поиск скрытых процессов

Механизм скрытия процесса, выполняемый некоторыми руткитами на основе модификации системной очереди объектов ядра заключается в следующем. В NT 5.x процессы, которые работают на данный момент в системе, увязываются в одном двунаправленном списке. Руткит переставляет указатели соседних объектов скрываемого процесса. Таким образом, когда ядро проходится по списку, то не видит этот процесс. На Рисунок 10. Очередь процессов, выполняющихся в системе представлена системная очередь, а на Рисунок 11 видно, что руткит скрыл второй процесс.

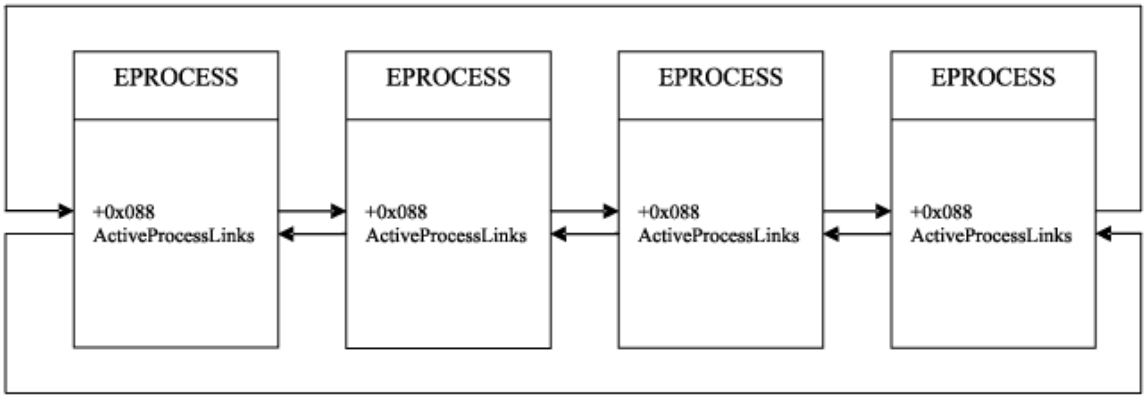

Рисунок 10. Очередь процессов, выполняющихся в системе

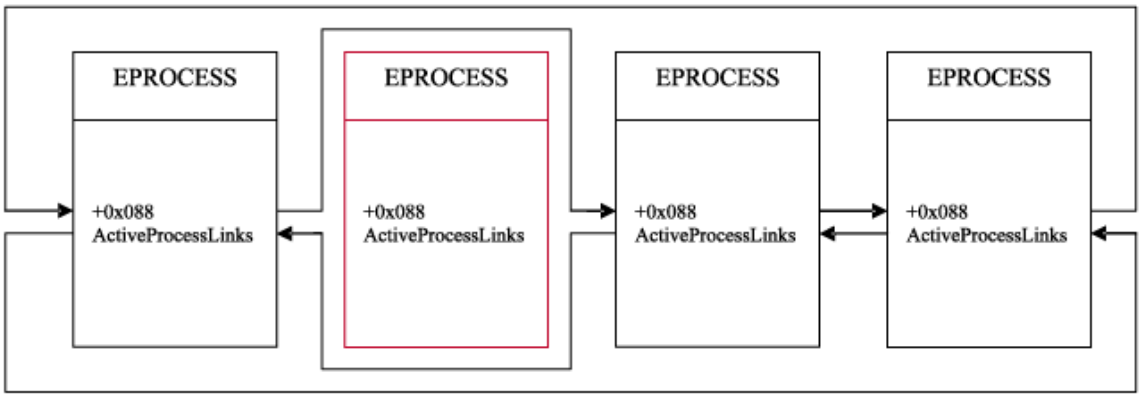

Рисунок 11. Модифицированная руткитом очередь процессов

Распределение процессорного времени в Windows NT осуществляется между исполняемыми сущностями – потоками. Процессы вообще не участвуют в перераспределении процессорного времени, и ядру без разницы – осуществлять переключение контекста на поток того же процесса или на поток другого процесса. Для этого используются очереди планировщика (диспетчера потоков). Таких очередей три: KiDispatcherReadyListHead – очередь готовых к исполнению потоков, KiWaitInListHead, KiWaitOutListHead – две очереди потоков, которые пока не планируются, то есть ждут какого-то события (например, завершение операции ввода-вывода).

Скрипт, автоматизирующий этот подход, проходит по четырем спискам потоков (KiDispatcherReadyListHead, KiWaitInListHead, KiWaitOutListHead, KiStackInSwapListHead), обращается к процессу, содержащему конкретный поток, и проверяет есть ли данный процесс в списке процессов.

5. Апробация

Для проверки работоспособности приведенных выше алгоритмов использовались следующие инструменты:

1) VMware Workstation 7.1.4 (для получения образа ОС);
2) XP Professional (ОС);
3) Moonsols Windows Memory Toolkit Community Edition (для преобразования полного образа памяти ОС в аварийный);
4) WinDbg 6.1 (отладчик).

Руткиты, подлежащие обнаружению:

1) Rootkit.Win32.Stuxnet [22];
2) Trojan-Spy.Win32.Zbot [23];
3) Rootkit.Win32.Fu [24].

В Приложение 2. Результаты апробации приведен список всех руткитов, участвовавших в тестировании.

5.1. Алгоритм тестирования

1. Создается виртуальная машина VMware с установленной ОС XP Professional с обновлением SP3.
2. Делается снимок ОС (сохраняется состояние системы).
3. Устанавливается руткит.
4. Копируется файл с расширением «.vmem».
5. Полный образ ОС, полученный на шаге 4, преобразовывается в аварийный дамп памяти.
6. Аварийный образ памяти, полученный на шаге 5, подключается к отладчику WinDbg.
7. Применяются алгоритмы анализа, описанные в решении.
8. ОС восстанавливается из снимка, полученного на шаге 2.
9. Шаги 3-8 повторяются для каждого руткита.

Прежде чем установить руткитов, алгоритмы были опробованы на чистой ОС.

Результаты получились следующими.

IDT

Перехват не обнаружен.

Изменение кода системных функций

```
!chkimg -d nt
        0 errors : nt
```

Поиск скрытых загруженных в память модулей

Все найденные модули опознаны.

Перехват функций через SSDT

Перехват не обнаружен.

Изменение указателя KTHREAD.pServiceDescriptorTable

Перехват не обнаружен.

Поиск скрытых процессов

Скрытых процессов не обнаружено.

5.2. Rootkit.Win32.Stuxnet

Руткит, который запускает вредоносный код в системе пользователя. Выполнен в виде драйвера режима ядра NT (kernel mode driver). Имеет размер 26616 байт. Руткит распространяется через сменные USB носители, используя уязвимость нулевого дня CVE-2010-2568 в LNK-файлах [25].

IDT

Перехват не обнаружен.

Изменение кода системных функций

```
!chkimg -d nt
  80501bf0-80501bf3  4 bytes - nt!KiServiceTable+64
        [ 3a 1c 5b 80:0e f8 40 b2 ]
  80501c30-80501c33  4 bytes - nt!KiServiceTable+a4 (+0x40)
        [ 86 a2 61 80:04 f6 40 b2 ]
  80501c88-80501c8b  4 bytes - nt!KiServiceTable+fc (+0x58)
        [ 16 a7 61 80:ac f4 40 b2 ]
  80501c90-80501c93  4 bytes - nt!KiServiceTable+104 (+0x08)
        [ e6 a8 61 80:f2 f4 40 b2 ]
  80501ca8-80501cab  4 bytes - nt!KiServiceTable+11c (+0x18)
        [ c6 aa 61 80:f2 f3 40 b2 ]
  80501cb0-80501cb3  4 bytes - nt!KiServiceTable+124 (+0x08)
        [ 30 ad 61 80:4e f3 40 b2 ]
  80501cc8-80501ccb  4 bytes - nt!KiServiceTable+13c (+0x18)
        [ 9a af 61 80:46 f4 40 b2 ]
  80501d14-80501d17  4 bytes - nt!KiServiceTable+188 (+0x4c)
        [ 82 c4 61 80:72 f9 40 b2 ]
  80501d68-80501d6b  4 bytes - nt!KiServiceTable+1dc (+0x54)
        [ 58 b6 61 80:d0 f7 40 b2 ]
  80501e0c-80501e0f  4 bytes - nt!KiServiceTable+280 (+0xa4)
        [ 7e b9 61 80:3e f0 40 b2 ]
  80501e50-80501e53  4 bytes - nt!KiServiceTable+2c4 (+0x44)
        [ be 84 61 80:66 f1 40 b2 ]
  80501f68-80501f6b  4 bytes - nt!KiServiceTable+3dc (+0x118)
        [ 0c 88 61 80:8a f2 40 b2 ]
  80501fa8-80501fab  4 bytes - nt!KiServiceTable+41c (+0x40)
        [ 36 8b 61 80:c2 fa 40 b2 ]
52 errors : nt (80501bf0-80501fab)
```

Поиск скрытых загруженных в память модулей

```
        Loaded Module Info: [0xf883ecc0]
ffffffff883ecc0 is not a valid address
        Loaded Module Info: [0xcb726d20]
ffffffffcb726d20 is not a valid address
```

Перехват функций через SSDT

```
        Index  Args  Check  System call
   -----  ----  -----  -----------
0019   0001  HOOK-> PROCMON20+0x480e (b240f80e)
0029   0007  HOOK-> PROCMON20+0x4604 (b240f604)
003F   0001  HOOK-> PROCMON20+0x44ac (b240f4ac)
0041   0002  HOOK-> PROCMON20+0x44f2 (b240f4f2)
0047   0006  HOOK-> PROCMON20+0x43f2 (b240f3f2)
0049   0006  HOOK-> PROCMON20+0x434e (b240f34e)
004F   0001  HOOK-> PROCMON20+0x4446 (b240f446)
0062   0002  HOOK-> PROCMON20+0x4972 (b240f972)
0077   0003  HOOK-> PROCMON20+0x47d0 (b240f7d0)
00A0   0005  HOOK-> PROCMON20+0x403e (b240f03e)
00B1   0006  HOOK-> PROCMON20+0x4166 (b240f166)
00F7   0006  HOOK-> PROCMON20+0x428a (b240f28a)
0107   0001  HOOK-> PROCMON20+0x4ac2 (b240fac2)
```

Изменение указателя KTHREAD.pServiceDescriptorTable

Перехват не обнаружен.

Поиск скрытых процессов

Скрытых процессов не обнаружено.

5.3 Trojan-Spy.Win32.Zbot

Вредоносная программа, предназначенная для ведение электронного шпионажа за пользователем (вводимая с клавиатуры информация, снимки экрана, список активных приложений и т.д.). Найденная информация передается злоумышленнику. Для передачи данных «хозяину» могут быть использованы электронная почта, FTP, HTTP (посредством указания данных в запросе) и другие способы.

IDT

Перехват не обнаружен.

Изменение кода системных функций

```
        !chkimg -d nt
0 errors : nt
```

Поиск скрытых загруженных в память модулей

```
        0x00150000
        Loaded Module Info: [0x00150000]
        150000 is not a valid address
        0x00170000
        Loaded Module Info: [0x00170000]
        170000 is not a valid address
        0x00400000
        Loaded Module Info: [0x00400000]
        400000 is not a valid address
        0x7c900000
        Loaded Module Info: [0x7c900000]
        7c900000 is not a valid address
        0x802be000
        Loaded Module Info: [0x802be000]
        ffffffff802be000 is not a valid address
```

0x802cf000
Loaded Module Info: [0x802cf000]
ffffffff802cf000 is not a valid address
0x80378000
Loaded Module Info: [0x80378000]
ffffffff80378000 is not a valid address
0x8042c000
Loaded Module Info: [0x8042c000]
ffffffff8042c000 is not a valid address
0x80473000
Loaded Module Info: [0x80473000]
ffffffff80473000 is not a valid address
0x804b6000
Loaded Module Info: [0x804b6000]
ffffffff804b6000 is not a valid address
0x806fe000
Loaded Module Info: [0x806fe000]
ffffffff806fe000 is not a valid address
0x8079a000
Loaded Module Info: [0x8079a000]
ffffffff8079a000 is not a valid address
0x807ba000
Loaded Module Info: [0x807ba000]
ffffffff807ba000 is not a valid address
0x80a24000
Loaded Module Info: [0x80a24000]
ffffffff80a24000 is not a valid address
0xf9b00cb0
Loaded Module Info: [0xf9b00cb0]
fffffffff9b00cb0 is not a valid address

Перехват функций через SSDT

Перехват не обнаружен.

Изменение указателя KTHREAD.pServiceDescriptorTable

Перехват не обнаружен.

Поиск скрытых процессов

Скрытых процессов не обнаружено.

5.4 Rootkit.Win32.Fu

Программа, предназначенная для сокрытия в системе определенных объектов, либо активности. Сокрытию, как правило, подвергаются ключи реестра (например, отвечающие за автозапуск вредоносных объектов), файлы, процессы в памяти зараженного компьютера, вредоносная сетевая активность.

IDT

Перехват не обнаружен.

Изменение кода системных функций

```
!chkimg -d nt
        0 errors : nt
```

Поиск скрытых загруженных в память модулей

```
Loaded Module Info: [0x7c900000]
7c900000 is not a valid address
Loaded Module Info: [0xd8840000]
ffffffffd8840000 is not a valid address
Loaded Module Info: [0xd8880000]
ffffffffd8880000 is not a valid address
Loaded Module Info: [0xf6d77000]
fffffffff6d77000 is not a valid address
Loaded Module Info: [0xf889ecb0]
fffffffff889ecb0 is not a valid address
```

Перехват функций через SSDT

Перехват не обнаружен.

Изменение указателя KTHREAD.pServiceDescriptorTable

Перехват не обнаружен.

Поиск скрытых процессов

```
820BD3C0  08   Waiting      | 81E0BB28  0368  *HIDDEN*   java6e.exe
```

6. Заключение

1. Проведен обзор подходов к поиску руткитов на компьютерах пользователей.
2. Проведен сравнительный анализ существующих инструментов для анализа руткитов.
3. Выделен набор признаков присутствия руткитов.
4. Реализован инструмент, позволяющий детектировать эти признаки и предоставляющий возможность проводить дальнейший анализ аномалий.
5. Работоспособность подтверждена на поиске существующих видов руткитов.

Список литературы

[1] А. Шевченко, «Эволюция руткитов» 2008.

[2] Д. Б. Г. Хоглунд, Руткиты. Внедрение в Ядро Windows, Санкт-Петербург: Питер, 2006.

[3] Wikipedia, «Sony BMG copy protection rootkit scandal» 24 апрель 2012. [В Интернете]. Доступна: http://en.wikipedia.org/wiki/Sony_BMG_copy_protection_rootkit_scandal.

[4] Wikipedia, «Брандмауэр Windows» 12 Июля 2011. [В Интернете]. Доступна: http://ru.wikipedia.org/wiki/Брандмауэр_Windows.

[5] Wikipedia, «Система обнаружения вторжений» 6 Сентябрь 2011. [В Интернете]. Доступна: http://ru.wikipedia.org/wiki/Система_обнаружения_вторжений.

[6] Wikipedia, «Система предотвращения вторжений» 3 Ноябрь 2011. [В Интернете]. Available: http://ru.wikipedia.org/wiki/Система_предотвращения_вторжений.

[7] Г. Ф. Александр Фролов, Защищенный режим процессоров Intel 80286/80386/80486, т. 4, Москва: Диалог-МИФИ, 1993, p. 234.

[8] Gangsta, «Гипервизор в роли антируткита» 10 февраль 2011. [В Интернете]. Доступна: http://habrahabr.ru/post/113519/.

[9] Wikipedia, «Прямой доступ к памяти» 27 Февраль 2012. [В Интернете]. Доступна: http://ru.wikipedia.org/wiki/Прямой_доступ_к_памяти.

[10] В. С. Mark Russinovich, «Программа RootkitRevealer» 1 ноябрь 2006. [В Интернете]. Доступна: http://technet.microsoft.com/ru-ru/sysinternals/bb897445.aspx.

[11] Wikipedia, «Open Source Tripwire» 2 апрель 2012. [В Интернете]. Доступна: http://en.wikipedia.org/wiki/Open_Source_Tripwire.

[12] F-Secure, «F-Secure» [В Интернете]. Доступна: http://www.f-secure.com.

[13] Greatis, «Unhackme» [В Интернете]. Доступна:

http://www.greatis.com/unhackme/.

[14] IceSword, «IceSword» [В Интернете]. Доступна: http://www.antirootkit.com.

[15] J. Rutkovska, «System Virginity Verifier» [В Интернете]. Доступна: http://www.softpedia.com/get/System/System-Info/System-Virginity-Verifier.shtml.

[16] «GMER» [В Интернете]. Доступна: http://www.gmer.net/.

[17] GPL, «The Volatility Framework» [В Интернете]. Доступна: https://www.volatilesystems.com/default/volatility.

[18] VMware, «VMware» [В Интернете]. Доступна: http://www.vmware.com.

[19] Moonsols, «Moonsols Windows Memory Toolkit» [В Интернете]. Доступна: http://www.moonsols.com.

[20] С. Шрайбер, Недокументированные возможности Windows 2000, Санкт-Петербург: Питер, 2002.

[21] Д. С. М. Руссинович, нутреннее устройство Microsoft Windows: Windows Server 2003, Windows XP, Windows 2000. Мастер-класс, Санкт Петербург: Питер, Русская редакция, 2008.

[22] Лаборатория Касперского, «Rootkit.Win32.Stuxnet.a» 20 Сентябрь 2010. [В Интернете]. Доступна: http://www.securelist.com/ru/descriptions/15071647/Rootkit.Win32.Stuxnet.a

[23] Лаборатория Касперского, «Trojan-Spy.Win32.Zbot.a» 24 Декабрь 2008. [В Интернете]. Доступна: http://www.securelist.com/ru/descriptions/253057/Trojan-Spy.Win32.Zbot.a.

[24] Лаборатория Касперского, «Rootkit.Win32.Fu.b» 9 Июнь 2006. [В Интернете]. Доступна: http://www.securelist.com/ru/descriptions/158955/Rootkit.Win32.Fu.b.

[25] S. Advisory, «Microsoft Security Advisory (2286198)» 2 Август 2010. [В Интернете]. Доступна: http://technet.microsoft.com/en-us/security/advisory/2286198.

[26] M. Rem, «Process Hunter» [В Интернете]. Доступна: http://ms-rem.dot-link.net/.

Приложение 1. Категории антируткитов

1. Брандмауэр – фильтрует сетевой трафик ОС.
2. Система обнаружения вторжений (Intrusion-Detection System, IDS)
    1) Сетевая (Network-based IDS, NIDS) – отслеживает вторжения, проверяя сетевой трафик и ведет наблюдение за несколькими хостами. Сетевая система обнаружения вторжений получает доступ к сетевому трафику, подключаясь к хабу или свитчу, настроенному на зеркалирование портов, либо сетевое TAP устройство. (Snort)
    2) Основанная на протоколе (Protocol-based IDS, PIDS) – представляет собой систему (либо агента), которая отслеживает и анализирует коммуникационные протоколы со связанными системами или пользователями. Для веб-сервера подобная IDS обычно ведет наблюдение за HTTP и HTTPS протоколами. При использовании HTTPS IDS должна располагаться на таком интерфейсе, чтобы просматривать HTTPS пакеты еще до их шифрования и отправки в сеть.
    3) Основанная на прикладных протоколах (Application Protocol-based IDS, APIDS) — это система (или агент), которая ведет наблюдение и анализ данных, передаваемых с использованием специфичных для определенных приложений протоколов. Например, на веб-сервере с SQL базой данных IDS будет отслеживать содержимое SQL команд, передаваемых на сервер.
    4) Локальная (Host-based IDS, HIDS) — система (или агент), расположенная на хосте, отслеживающая вторжения, используя анализ системных вызовов, логов приложений, модификаций файлов (исполняемых, файлов паролей, системных баз данных), состояния хоста и прочих источников. (OSSEC)
    5) Гибридная совмещает два и более подходов к разработке IDS. Данные от агентов на хостах комбинируются с сетевой информацией для

создания наиболее полного представления о безопасности сети. (Prelude)

3. Система предотвращения вторжений (Intrusion-Prevention System, IPS)
    1) Сетевая (Network Intrusion- Prevention System , NIPS) - отслеживают трафик в компьютерной сети и блокируют подозрительные потоки данных.
    2) Для беспроводных сетей (Wireless Intrusion Prevention Systems, WIPS) - проверяет активность в беспроводных сетях. В частности, обнаруживает неверно сконфигурированные точки беспроводного доступа к сети, атаки человек посередине, спуфинг mac-адресов.
    3) Поведенческий анализ сети (Network Behavior Analysis, NBA) - анализирует сетевой трафик, идентифицирует нетипичные потоки, например DoS и DDoS атаки.
    4) Локальная (Host-based Intrusion- Prevention System, HIPS) - резидентные программы, обнаруживающие подозрительную активность на компьютере.

Приложение 2. Результаты апробации

Вредоносное ПО	Перехват IDT 2e	Изменение кода системных функций	Скрытые загруженные в память модули	Перехват функций через SSDT	Изменение указателя KTHREAD.pServiceDescriptorTable	Скрытые процессы
Email-Flooder.Win32.Hotmail.c	-	-	+	-	+	-
Email-Flooder.Win32.SpyDerWeb	-	-	+	-	+	-
Email-Worm.BAT.BWG.f	-	-	+	-	+	-
Email-Worm.JS.Spth.Jsg.b	-	-	+	-	+	-
Rootkit.Win32.Agent.aek	-	+	+	-	-	-
Rootkit.Win32.Agent.bfh	-	+	+	-	-	-
Rootkit.Win32.HideProc.o	-	+	+	-	-	+
Rootkit.Win32.KernelBot.a	-	+	+	-	-	-
Rootkit.Win32.Pakes.a	+	-	+	-	-	-
Rootkit.Win32.Podnuha.aaf	-	+	+	-	-	-
Rootkit.Win32.Podnuha.ti	-	+	+	-	-	-
Rootkit.Win32.Qandr.an	-	+	+	-	-	-
Rootkit.Win32.Qandr.r	-	+	+	-	-	-
SpamTool.Win32.Delf.cs	-	-	+	-	-	-
Trojan.ANSI.Error32	-	-	+	-	-	-
Trojan.BAT.Adduser.e	-	-	+	+	-	-
Trojan.BAT.HaltWin.c	-	-	+	+	-	-
Worm.Win32.Radminer.c	-	-	+	-	-	-
Worm.Win32.Runfer.js	-	-	+	-	-	+
Worm.Win32.Viking.t	-	-	+	-	-	-
Rootkit.Win32.Stuxnet	-	+	+	+	-	-
Trojan-Spy.Win32.Zbot	-	-	+	-	-	-
Rootkit.Win32.Fu	-	-	+	-	-	+