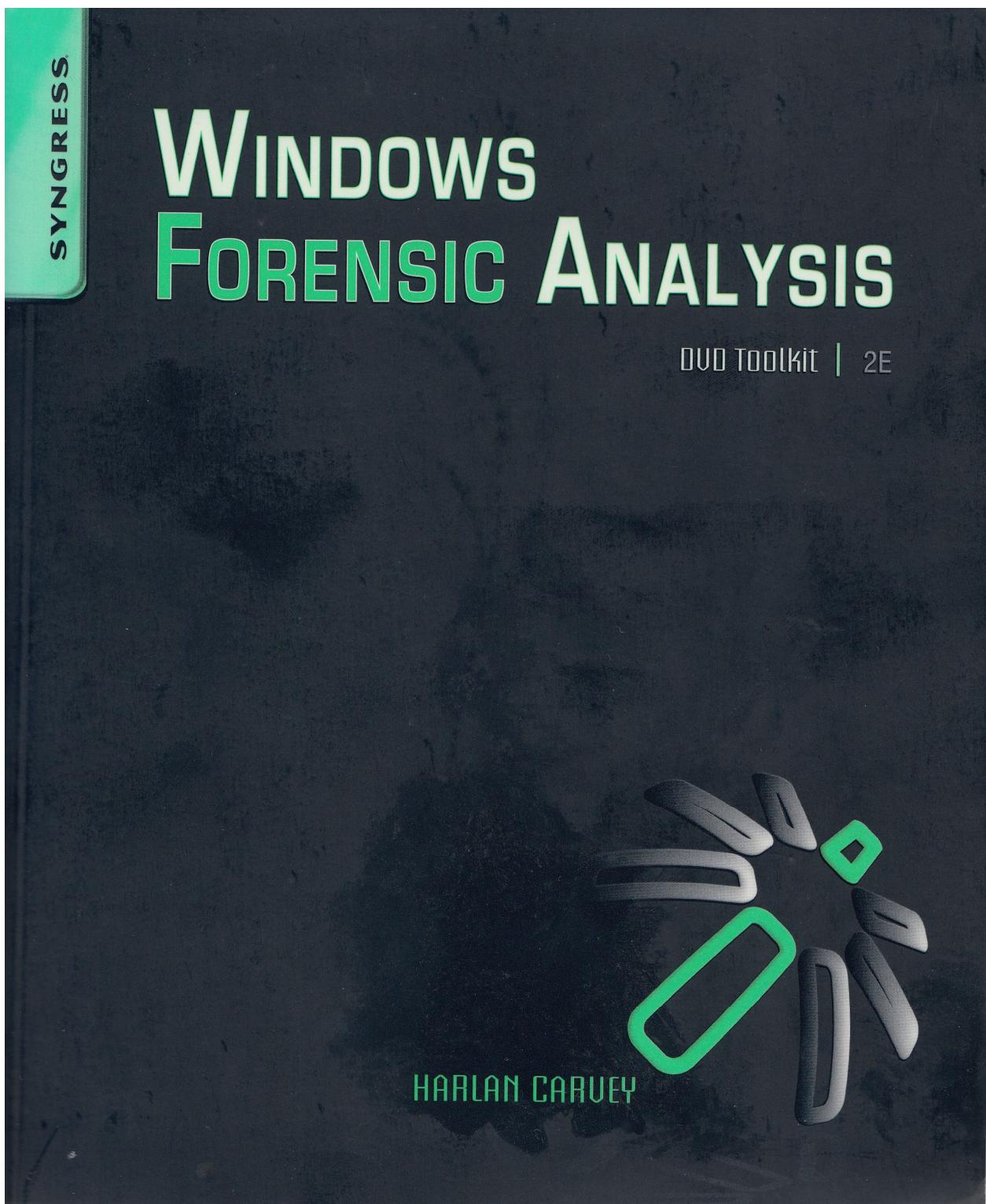


Харлэн Карви

**Криминалистическое исследование
Windows**





Глава 1

Исследование работающей системы: Сбор энергозависимых данных

Содержание этой главы:

- Исследование работающей системы
 - Какие данные нужно собрать
 - Энергонезависимые данные
 - Методики исследования работающей системы
-
- ✓ Краткое изложение
 - ✓ Быстрое повторение
 - ✓ Часто задаваемые вопросы

Введение

Сегодня специалисты все чаще сталкиваются с ситуациями, когда традиционная, широко распространенная методика проведения судебного компьютерно-технического исследования – отключение компьютера от питания и создание побитового образа системного НЖМД, через устройство блокирования записи – просто не является приемлемой. Например, специалисты нередко имеют дело с серверами, которые нельзя выключать, так как они чрезвычайно важны для деловых операций организации, где проводится осмотр (вымка). В некоторых случаях эксперты и специалисты по расследованию инцидентов не могут ответить на имеющиеся вопросы, используя только содержимое образа НЖМД. Например, я спрашивал у сотрудников правоохранительных органов о том, какой подход лучше всего использовать в ситуациях, связанных с исчезновением детей, которых выманили из дома или школы с помощью мгновенных сообщений, особенно когда в настройках приложений для обмена мгновенными сообщениями по умолчанию указано не сохранять журналы разговоров.

Такие вопросы волнуют не только сотрудников правоохранительных органов. Во многих случаях лучшие источники информации или данных для исследования доступны только в памяти компьютера (сетевые соединения, содержимое окна приложения для обмена мгновенными сообщениями, память, используемая процессом такого приложения, ключи шифрования, пароли и т. д.). В других случаях, экспертов просят выяснить: работает ли в системе троянская или другая вредоносная программа, и была ли скопирована конфиденциальная информация из системы. Специалистам по расследованию инцидентов, главным образом, задают вопросы о том: какие действия происходили в системе, пока она работала, и на эти вопросы нельзя ответить, используя традиционный, «пуристский» поход к компьютерно-технической экспертизе. Сотрудники ИТ-отделов находят в журналах брандмауэров и систем обнаружения вторжений записи о необычном или подозрительном трафике и завершают работу систем, которые являются источником трафика, прежде чем определить, какой процесс отвечает за передачу этой информации. В таких ситуациях требуется, чтобы эксперт провел исследование работающего компьютера – сбор данных из системы, работа которой еще не завершена. Такое исследование само по себе поднимает несколько вопросов, которые мы рассмотрим в этой главе.

Возможно, более важно то, что требование провести исследование работающей системы больше не зависит от решений организаций. Вместо этого исследование работающей системы является обязательным согласно требованиям законодательства, а также регулятивных органов (например, согласно стандарту защиты информации в индустрии платежных карт Visa). В случае нарушения безопасности системы эти регулятивные органы задают три основных вопроса:

- Была ли система взломана?
- Содержала ли взломанная система конфиденциальные данные? (Определение конфиденциальных данных см. в соответствующих законодательных и нормативных документах.)
- Если ответ на оба предыдущих вопроса – «да», привел ли взлом системы к раскрытию этих конфиденциальных данных?

Однако, многие организации просто не готовы к инцидентам, и поэтому действия их сотрудников, отвечающих за устранение проблем, могут подвергнуть эти организации большему риску, чем сами инциденты. В основном это связано с тем, что подход многих ИТ-отделов, заключающийся в завершении работы систем и стирании всех их данных, не предусматривает сбор необходимой информации, чтобы ответить на неизбежные вопросы. Эти вопросы всегда возникают, когда юридический отдел или отдел контроля за соблюдением нормативных требований узнает об инциденте, а затем выясняется, что на эти вопросы нельзя ответить.

Исследование работающей системы

Сегодня эксперты сталкиваются с ситуациями, когда выключить компьютер (или несколько компьютеров) и создать образы накопителей не представляется возможным. По мере того, как использование электронной коммерции продолжает расти, простой системы измеряется в сотнях или тысячах долларов в минуту в зависимости от количества невыполненных транзакций. Следовательно, выключение системы для того, чтобы создать образ НЖМД, может серьезно повлиять на чистую прибыль. Кроме того, некоторые компании гарантируют своим клиентам (и указывают это в соглашениях об уровне обслуживания), что системы будут находиться в работоспособном состоянии 99,999 процентов времени (за исключением перерывов на техническое обслуживание, конечно). Для того чтобы создать образ одного накопителя, систему, возможно, потребуется вывести из эксплуатации на несколько часов (в зависимости от конфигурации системы).

Информационная магистраль – это больше не место только для хакеров-любителей и шутников. В киберпространстве совершаются множество серьезных преступлений, а действия злоумышленников становятся все более изощренными. Существуют программы, которые могут попасть в систему и украсть личную информацию (пароли, личные файлы, налоговые декларации и т. д.), но код некоторых из этих программ не записывается на НЖМД, эти программы существуют только в памяти компьютера. После завершения работы системы все признаки присутствия таких программ исчезают.

В апреле 2006 года компания Seagate выпустила первые накопители емкостью 750 Гб. Сегодня мне регулярно встречаются внешние накопители, емкость которых превышает 1,5 Тб, а также многотерабайтные системы хранения в сетях клиентов. Представьте себе систему с массивом RAID 5 из восьми накопителей, общая емкость которых достигает 8 Тб. Сколько времени потребовалось бы вам, чтобы клонировать данные этих накопителей? При определенных конфигурациях специалисту может потребоваться более четырех часов, чтобы создать и проверить образ одного 80-гигабайтного НЖМД. И нужно ли клонировать данные всей системы, если вас интересуют только действия, выполненные отдельным процессом, а не тысячи файлов, находящиеся в системе?

В отдельных случаях, нам нужно собрать некоторую информацию о работающей системе перед тем, как завершить ее работу, создать побитовый образ НЖМД и провести более традиционный судебный компьютерный анализ. Информация, которая представляла бы для нас интерес, энергозависима по своей природе, то есть она перестает существовать после отключения питания системы. Энергозависимая информация обычно находится в физической памяти, или оперативном запоминающем устройстве (ОЗУ), и включает в себя данные о процессах, сетевых соединениях, содержимом буфера обмена и тому подобные сведения. Эта информация описывает состояние системы в тот момент, когда вы находитесь перед ней, сидите за клавиатурой или удаленно получаете доступ к этой системе. Эксперт может столкнуться с ситуацией, когда необходимо быстро собрать и проанализировать (см. следующую главу) данные, чтобы определить характер и масштаб инцидента. Когда, при подготовке к созданию образа НЖМД традиционным способом, питание отключается от системы, эта информация просто исчезает. Однако также нужно иметь в виду, что любые действия, которые вы совершаете в работающей системе (запуск антивирусных приложений, поиск файлов или данных кредитных карт, изменение конфигурации системы и т. д.), оставят свои собственные артефакты и, возможно, перезапишут важные или относящиеся к делу данные. Следовательно, необходимо уделить главное внимание сбору и сохранению этих энергозависимых данных.

У нас действительно есть для этого возможности – инструменты и приемы, предназначенные для сбора энергозависимых данных из работающей системы, которые позволяют нам получить более полное представление о состоянии системы, а также предоставляют нам дополнительную информацию. Именно это представляет собой исследование работающей системы (*live response*): доступ к работающей системе и сбор энергозависимых (и в некоторых случаях энергонезависимых) данных.

Существует еще один термин: клонирование данных работающей системы (*live acquisition*), который часто путают с исследованием работающей системы. Исследование работающей системы связано со сбором энергозависимых данных, а клонирование данных работающей системы представляет собой создание образа накопителя такой системы. В этой главе мы начнем знакомиться с инструментами, приемами и методиками для проведения исследования работающей системы. Когда мы говорим об исследовании работающей системы, мы должны понимать, *какую* информацию нам нужно собрать из системы и *как* это сделать. В данной главе мы рассмотрим эти вопросы сбора энергозависимых данных; в следующей главе мы познакомимся со способами анализа этих данных. Затем мы поговорим о нескольких решениях, существующих для клонирования данных работающей системы. Анализ образа, созданного во время клонирования данных работающей системы, будет рассматриваться в оставшихся главах этой книги.

Прежде чем мы начнем обсуждать инструменты и способы для исследования работающей системы, нужно обратить внимание на две важные темы: принцип обмена Локара и порядок изменяемости энергозависимых данных. Эти понятия лежат в основе этой главы и исследования работающей системы в общем, и мы рассмотрим их подробно.

Принцип обмена Локара

При проведении исследования работающей системы эксперты и специалисты по расследованию инцидентов должны не забывать об одном очень важном принципе. Когда мы взаимодействуем с работающей системой (как пользователи или специалисты (эксперты)), в этой системе происходят изменения. В работающей системе изменения происходят просто с течением времени: процессы выполняются, данные сохраняются и удаляются, сетевые соединения завершаются и создаются и т. д. Некоторые изменения происходят просто во время работы системы. Кроме того, изменения происходят, когда эксперт выполняет в системе программу для сбора энергозависимых или других данных.

Запуск программы приводит к тому, что информация загружается в физическую память, и при этом информация в физической памяти, используемая другими, уже запущенными процессами может быть записана в файл подкачки. Когда эксперт собирает информацию и отправляет ее из системы, создается новое сетевое соединение. Все эти изменения можно вместе объяснить принципом обмена Локара. Изменения, которые происходят в системе, когда сама система находится в режиме ожидания, называются «динамикой улик» (evidence dynamics) и похожи на дождь, смывающий возможные улики на месте преступления.

В начале XX века работа доктора Эдмона Локара (Edmond Locard) в области криминалистики и реконструкции места преступления стала известна как принцип обмена Локара. Согласно этому принципу, когда два объекта соприкасаются, между ними происходит обмен веществами или вещества одного объекта переносятся на другой. Если вы посмотрите популярный телевизионный сериал «Место преступления» (CSI), то неизменно услышите, как один из криминалистов говорит о возможном переносе веществ. Обычно это происходит после сцены, когда автомобиль врезается в какой-нибудь объект или когда эксперт исследует тело и находит вещество, не соответствующее обстановке.

Тот же принцип относится к области цифровых данных. Например, когда два компьютера связываются по сети, между ними происходит обмен информацией. Информация об одном компьютере появится в памяти процесса и/или файлах журналов другого компьютера (наглядную демонстрацию этого принципа см. во вставке «Локар и Netcat»). Когда периферийное устройство, например, съемный накопитель (флеш накопитель, iPod и т. д.), присоединяется к компьютеру с операционной системой (ОС) Windows, информация об этом устройстве остается на компьютере. Если эксперт взаимодействует с работающей системой, в этой системе происходят изменения по мере того, как выполняются программы, а данные копируются из системы. Эти изменения могут оставлять временные (память процесса, сетевые соединения) или постоянные (файлы журналов, записи реестра) следы.

Инструменты и ловушки...

Локар и Netcat

Можно использовать простые инструменты, например, Netcat (<http://en.wikipedia.org/wiki/Netcat>), чтобы продемонстрировать принцип обмена Локара. Если вы не знакомы с netcat («nc.exe» в ОС Windows), достаточно сказать, что netcat – универсальный инструмент, позволяющий принимать и передавать информацию через сетевые соединения.

Для этого примера вам потребуется три инструмента: netcat («nc.exe»), «pmdump.exe» (www.ntsecurity.nu/toolbox/pmdump/) и «strings.exe» (<http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx>) или BinText (доступный по адресу www.foundstone.com/us/resources/proddesc/bintext.htm). Можно выполнить этот пример, используя одну или две системы, но пример будет нагляднее при использовании двух систем. Если вы используете одну систему, создайте два каталога и поместите в каждый из них копию инструмента netcat.

Сначала запустите netcat в режиме ожидания с помощью следующей команды:

```
C:\test>nc -L -d -p 8080 -e cmd.exe
```

Эта команда указывает netcat прослушивать порт 8080 в ожидании соединения (в скрытом режиме) и, когда соединение будет установлено, запустить командную строку. После того как вы введете эту команду и нажмете Enter, откройте диспетчер задач и обратите внимание на идентификатор процесса (PID), который вы только что создали. (В этом примере я использую инструмент netcat версии 1.11 NT, который я нашел на сайте www.vulnwatch.org/netcat. На момент написания данной книги этот веб-сайт был

(недоступен.)

Теперь откройте другую командную строку в той же системе или перейдите к другой системе и откройте командную строку там. Введите следующую команду, чтобы подключиться к только что созданному прослушивающему процессу netcat:

```
C:\test2>nc <IP-адрес> 8080
```

Эта команда указывает netcat запуститься в режиме клиента и подключиться к IP-адресу, используя порт 8080, где ожидает соединения прослушивающий процесс. Если вы выполняете пример в одной системе, укажите для IP-адреса значение 127.0.0.1.

После установления соединения вы увидите обычный заголовок командной строки, в котором указывается версия операционной системы и сведения об авторских правах. Введите в командной строке несколько команд, например, **dir** или любую другую, просто чтобы отправить информацию через это соединение.

В системе, где выполняется прослушивающий процесс netcat, откройте другую командную строку и используйте программу «**pmdump.exe**» (которая будет рассмотрена позднее в этой главе), чтобы получить содержимое памяти прослушивающего процесса:

```
C:\test>pmdump <PID> netcat1.log
```

Эта команда получит содержимое памяти, используемой процессом, и сохранит его в файле «**netcat1.log**». Если хотите, можно также создать дамп памяти процесса клиентской стороны соединения. Теперь, когда память процесса сохранена в файле, можно завершить работу обоих процессов. Примените программу «**strings.exe**» к файлу памяти прослушивающего процесса или откройте этот файл в программе BinText, чтобы увидеть IP-адрес клиента. Выполнив то же самое с файлом памяти клиента, вы увидите информацию о системе, где выполнялся прослушивающий процесс, что доказывает принцип обмена Локара.

Программы, которые мы используем для сбора информации, могут оказывать другое воздействие на работающую систему. Например, программе, возможно, потребуется прочитать несколько разделов реестра, и пути к этим разделам будут переданы в память. Системы Windows XP выполняют упреждающую выборку для приложений, поэтому, если эксперт запускает программу, которую пользователь уже запускал в системе, отметки времени последнего доступа и последнего изменения файла упреждающей выборки (а также содержимое самого файла) для этого приложения будут изменены. Если программа, которую запускает эксперт, не использовалась раньше, будет создан новый файл упреждающей выборки в каталоге «Prefetch» (при условии, что количество pf-файлов в каталоге «Prefetch» не превышает 128 ... , но подробнее об этом вы узнаете позднее в данной книге).

Эксперты должны не только понимать, что эти изменения будут происходить, но и должны документировать эти изменения и уметь объяснить, какое влияние оказали их действия на работающую систему, в разумных пределах. Например, эксперт должен уметь определять, какие pf-файлы в каталоге «Prefetch» в ОС Windows XP появились в результате его действий, а какие – в результате действий пользователя. То же справедливо в отношении параметров реестра. Как и в случае с возможностями упреждающей выборки в Windows XP, действия эксперта будут оказывать влияние на системный реестр. Например, в реестре могут появиться записи, и в связи с этим отметки времени LastWrite разделов реестра будут обновлены. Некоторые изменения могут производиться оболочкой (т. е. Проводником Windows) в связи с тем, что система функционирует, а не быть непосредственным результатом ваших действий или выполнения ваших инструментов.

Необходимо тестиировать инструменты, которые вы используете, и понимать принципы их работы, чтобы уметь задокументировать и объяснить, какие артефакты в системе появились в результате ваших действий, а какие – в результате действий пользователя или злоумышленника.

Совет

При принятии решения, проводить исследование работающей системы или нет, важно не забывать, что, хотя ваши действия оказывают влияние на систему (процессы загружаются в память, файлы создаются на накопителе и т. д.), такое же влияние оказывает ваше бездействие. Подумайте об этом. Пока система функционирует, в ней постоянно происходят какие-то события. Даже когда система находится в состоянии простоя, в ней выполняются процессы и совершаются действия. В случае с Windows XP просто подождите 24 часа и будет создана точка восстановления системы (по умолчанию). Подождите три дня и система выполнит неполную дефрагментацию. Также подумайте о том, что если кто-то несанкционированно копирует данные из системы, а вы ждете и ничего не делаете, этот человек будет продолжать похищать данные. Поэтому, перед тем как решить, проводить исследование работающей системы или нет, вы должны ответить на вопрос: «Должен ли я (а) ничего не делать или (б) предпринять правильные меры, чтобы как можно лучше защитить организацию при данных обстоятельствах?»

Порядок изменяемости энергозависимых данных

Мы знаем, что энергозависимая информация существует в памяти работающей системы и что определенные виды энергозависимой информации могут быть более изменчивы, чем другие. То есть некоторая информация в работающей системе может иметь значительно более короткий срок хранения, чем другие данные. Например, времена соединений, если они не используются, истекают иногда в течение нескольких минут. Вы можете убедиться в этом, перейдя на отдельный сайт или установив какое-нибудь другое соединение и просмотрев сведения об этом соединении при помощи программы «netstat.exe». Затем закройте используемое клиентское приложение, и состояние этого сетевого соединения изменится со временем, пока сведения о соединении, в конце концов, не исчезнут из выходных данных программы «netstat.exe». Системное время, однако, меняется значительно быстрее, а содержимое буфера обмена остается постоянным, пока оно либо не будет изменено, либо система не будет отключена от питания. Кроме того, одни процессы, такие как службы (которые называются «демоны» (*daemons*) в мире UNIX), выполняются долго, а другие процессы могут быть очень недолговечными: они быстро выполняют свои задачи, прежде чем исчезнуть из памяти. Это означает, что сначала нужно собрать определенную информацию, чтобы она не была изменена позднее, а другие энергозависимые данные, которые, похоже, более долговечны, можно собрать позднее.

Отличный источник информации по этому вопросу – документ № 3227 из серии RFC, «Guidelines for Evidence Collection and Archiving» (www.faqs.org/rfcs/rfc3227.html). Этот документ, опубликованный в феврале 2002 года, остается актуальным сегодня, так как основные руководящие принципы не меняются, несмотря на изменение технологий. В нем указаны такие принципы для сбора данных, как сбор наиболее полной информации о системе; ведение подробных записей; указание разницы между всемирным координированным временем (UTC), местным временем и системным временем; и сведение изменений в данных к минимуму. Мы будем учитывать эти принципы при рассмотрении темы исследования работающей системы.

Совет

В RFC-документе № 3227 указано, что нужно обратить внимание на разницу

между системным временем и UTC-временем, а также вести подробные записи на тот случай, если вам нужно будет объяснить или обосновать свои действия (в документе сказано «давать показания») даже несколько лет спустя.

Особый интерес в этом RFC-документе представляет раздел 2.1 «Order of Volatility», в котором перечислены определенные типы энергозависимых данных в порядке: от наиболее до наименее изменчивых. Элементы, которые со временем изменяются или исчезают быстрее других (например, процессы, сетевые соединения и т. д.), должны быть собраны в первую очередь. И наоборот, информацию, которая меньше подвержена изменениям, например, физическая конфигурация системы, можно собрать позже. Используя эти руководящие принципы, мы можем понять, какие типы данных нужно собрать из системы, где искать эти данные, какие инструменты применять, чтобы найти их, и даже как получить эти данные из системы, тем самым сводя к минимуму воздействие на целевую систему и в то же время собирая данные, необходимые для проведения анализа.

Когда проводить исследование работающей системы

Возможно, самый главный вопрос, который беспокоит экспертов и специалистов по расследованию инцидентов, – «Когда нужно проводить исследование работающей системы?». Сегодня, во многих случаях (например, уголовные или гражданские дела, внутренние корпоративные расследования), отсутствует предопределенный набор условий для проведения исследования работающей системы. На самом деле, во многих ситуациях даже не рассматривается возможность исследования работающей системы и последующего сбора энергозависимых данных. Решение проводить исследование работающей системы зависит от ситуации, среды (учитывая цели эксперта, политику компании или применяемые законодательные акты) и характера существующей проблемы.

Давайте рассмотрим несколько примеров. Предположим, с вами связался системный администратор и сообщил об обнаружении необычного сетевого трафика. Он получил оповещение системы обнаружения вторжений (СОВ) и, проверив журналы брандмауэра, обнаружил в них подозрительные записи, которые, похоже, соотносятся с оповещением СОВ. Администратор предполагает, что необычный трафик исходит из отдельной системы, которая находится во внутренней сети. У администратора уже есть данные об оповещениях СОВ и сетевые журналы, но вы решаете выполнить перехват сетевого трафика, чтобы получить о нем более полное представление. Вы понимаете, что теперь у вас есть информация о сетевом трафике, но как связать ее с отдельной системой? Это довольно легко, не так ли? Ведь у вас есть IP-адрес системы (исходный или целевой IP-адрес в перехваченном сетевом трафике), а если вы также перехватили Ethernet-кадры, то у вас будет и MAC-адрес. Но как потом связать трафик, который вы видите в сети, с отдельным пользователем или процессом, выполняющимся в системе?

Чтобы точно определить источник подозрительного трафика (процесс, который его создает), нужно будет собрать информацию о выполняющихся процессах и сетевых соединениях из системы, прежде чем завершать ее работу. Другая информация, собранная во время исследования работающей системы может показать, что кто-то вошел в систему удаленно, через сеть или программу скрытого удаленного администрирования или, что процесс был запущен как назначенное задание.

Какие другие ситуации могут предполагать или даже требовать проведение исследования работающей системы? Например, случаи «троянской защиты», когда вредоносные действия приписываются троянской программе или программе скрытого удаленного администрирования. В октябре 2002 года на компьютере Джюлиана Грина (Julian Green) было обнаружено несколько (в некоторых отчетах говорится о более 170) незаконных изображений. В результате судебной экспертизы было обнаружено, что в его

системе присутствует несколько троянских программ, которые получали доступ к незаконным сайтам всякий раз, когда пользователь запускал веб-браузер. С Джулиана Грина были сняты все обвинения.

На следующий год Аарон Кэффри (Aaron Caffrey) заявил, что троянские программы позволяли другим лицам управлять его компьютером и атаковать другие системы, в чем он и обвинялся. Защита Кэффри строилась на том, что, хотя в системе не было обнаружено троянских программ во время судебной экспертизы, троянская программа все-таки *могла* быть ответственной за эти действия. Этого аргумента было достаточно, чтобы Аарона признали невиновным.

Задним числом мы понимаем, что в подобных случаях было бы лучше, чтобы информация о сетевых соединениях и выполняющихся процессах была собрана при изъятии компьютеров, особенно если работа систем еще не была завершена, когда эксперт прибыл на место инцидента. Та информация могла бы подсказать, выполнялись ли в то время подозрительные процессы и установил ли кто-то другой соединение с системой, чтобы управлять ею и передавать в нее файлы, проводить атаки на другие системы и т. д.

Проведение исследования работающей системы означает, что вы будете собирать информацию о состоянии системы во время ее работы, в том числе информацию о процессах и файлах, к которым получают доступ эти процессы, а также информацию об исходящих и о входящих сетевых соединениях системы и о процессах, использующих эти соединения. Фактически исследование работающей системы – единственный способ получить эту информацию, так как она полностью исчезнет после выключения системы.

Как обсуждалось ранее, еще одним доводом в пользу проведения исследования работающей системы является то, что саму систему нельзя выключать без веских (я имею в виду *действительно веских*) причин. Простой крупных критических систем, например, тех, что используются в электронной коммерции, измеряется в невыполненных транзакциях или сотнях (даже тысячах) долларов в минуту. Так как процесс создания образов накопителей может занять изрядное количество времени (большинство систем такого типа использует несколько накопителей в конфигурации RAID), предпочтительно иметь серьезные факты, чтобы обосновать необходимость выключения системы и вывода ее из эксплуатации. Такие факты могут потребоваться не только системному администратору, чтобы обосновать эти действия руководителю ИТ-отдела, но и главному финансовому директору, чтобы обосновать эти действия совету директоров.

Еще один фактор, который следует принять во внимание, – это законы, требующие уведомлять пользователей о раскрытии их конфиденциальных данных. Начиная с появления закона SB 1386 штата Калифорния, компании, если их система безопасности была нарушена и при этом были раскрыты конфиденциальные данные, должны уведомлять об этом своих клиентов, постоянно проживающих в Калифорнии, чтобы эти клиенты могли защититься от кражи персональных данных. На момент написания этой книги другие штаты тоже начали следовать примеру Калифорнии, и даже ходили слухи о соответствующем федеральном законе. Это означает, что компании, хранящие и обрабатывающие конфиденциальную информацию, не могут просто молчать об определенных типах нарушений системы безопасности.

Термин *конфиденциальные данные* на самом деле намного шире определения, содержащегося в законе SB 1386; учитывайте также закон Калифорнии CA 1298, в котором дается определение закрытой медицинской информации (protected health information, PHI). А регулятивные органы, такие как совет по стандартам безопасности индустрии платежных карт (PCI Council), также предоставляют определение «конфиденциальных данных» (совет PCI занимается данными кредитных карт). Кроме того, эти регулятивные органы устанавливают требование защищать данные, о которых идет речь, и даже определяют некоторые меры для этого; согласно стандарту защиты информации в индустрии платежных карт версии 1.1, требуется иметь план расследования

инцидентов компьютерной безопасности (п. 12.9), а также ежегодно проверять этот план (п. 12.9.2).

Помимо всего прочего, компании, хранящие и обрабатывающие конфиденциальные данные (независимо от их определения), захотят точно знать, произошла ли утечка конфиденциальных данных во время нарушения системы безопасности; это связано с тем, что законодательные и регулятивные документы требуют, чтобы организации сообщали пользователям о том, что их данные были раскрыты. В случае если система безопасности компании была нарушена, но нельзя точно определить, какие данные были похищены, от такой компании могут потребовать, чтобы она сообщила своим клиентам обо всех имеющихся данных, которые, возможно, были раскрыты. И в большинстве случаев факт предупреждения клиентов о том, что их личные данные сейчас находятся в руках неизвестного лица (или, нередко, нескольких неизвестных лиц), может иметь значительное, пагубное влияние на компанию. Клиенты могут перестать пользоваться услугами компании и рассказать о происшедшем своим друзьям и родственникам в других штатах. Новые клиенты, возможно, решат обратиться к конкуренту. Потеря текущего и будущего дохода изменит лицо компании и может привести к банкротству. Поэтому, почему компания должна просто подозревать, что ее система безопасности была нарушена, а конфиденциальные данные похищены, и покорно уведомлять своих клиентов? Разве компания сначала не захочет наверняка узнать, что конфиденциальная информация о ее клиентах была похищена? Разве вы бы не захотели узнать об этом?

Инструменты и ловушки...

Исследование работающей системы и конфиденциальные данные

Это ловушка, в которую попадают многие организации, но они не знают об этом, пока в нее не окажутся. Часто организации просто не готовы к инцидентам, а в нескольких случаях из моей практики организации были подготовлены, но планы расследования составлялись ИТ-отделом без какого-либо согласования с другими отделами. В результате, когда вредоносная программа обнаруживается в организации с помощью определенных средств, сотрудника ИТ-отдела мгновенно приступают к поиску и очищению зараженных систем. На совещании кто-то упоминает о работе, выполненной сотрудниками ИТ-отдела, а кто-то из юридического отдела или отдела контроля за соблюдением нормативных требований слышит об этом и говорит: «Эти шесть зараженных систем были расположены в том отделении компании, которое обрабатывает данные кредитных карт... Были какие-нибудь из этих данных похищены или раскрыты?».

Давайте посмотрим: сотрудники ИТ-отдела определили каждую зараженную систему, отключили ее от сети, выполнили проверки на наличие вирусов, возможно, подключили ее к изолированному сегменту сети и запустили полное обновление системы (Windows Update) или просто очистили накопитель, переустановили операционную систему и восстановили как можно больше пользовательских данных. Находились ли в системе конфиденциальные данные? Во многих случаях мы можем сказать «да». Были ли эти данные похищены? На данном этапе мы этого не знаем и не можем дать ответ на этот вопрос просто потому, что у нас нет информации, которую можно проанализировать. Вся информация, которая могла бы у нас быть, исчезла, когда работа системы была завершена. В таких случаях действия ИТ-отдела подвергают организацию большему риску, чем сам инцидент, так как некоторые регулятивные органы утверждают, что если нельзя точно определить, что конфиденциальные данные не были раскрыты, нужно предположить, что они были раскрыты, и, следовательно, организация будет обязана сообщить, что все конфиденциальные данные были раскрыты.

Помимо приблизительных расходов на уведомления клиентов из-за нарушения системы безопасности, таких как убытки в результате падения доверия клиентов, следует

учитывать также прогнозируемые, более поддающиеся определению, расходы, такие как фактическая стоимость уведомления клиентов о раскрытии их конфиденциальной информации, штрафы, наложенные регулятивными органами, иски предъявленные в результате раскрытия конфиденциальных данных, и т. д. Теперь сравните эти расходы с расходами, связанными с фактическим принятием мер по защите конфиденциальных данных, хранящихся и обрабатываемых в вашей организации; к таким мерам относятся многие действия, требуемые регулятивными органами, например, составление и ежегодная проверка плана расследования инцидентов компьютерной безопасности, а также способность определять инциденты и проводить их расследование. Некоторые из этих мер включают в себя способность собирать необходимую информацию, посредством исследования работающей системы, чтобы определить, какие конфиденциальные данные, если таковые имелись, возможно, были раскрыты.

Например, рассмотрим случай, когда «анонимное» лицо в Интернете заявляет, что похитило у организации конфиденциальную информацию. Это лицо утверждает, что взломало систему безопасности организации через Интернет и смогло собрать имена клиентов, номера социального страхования, адреса, данные кредитных карт и другую информацию. Руководство организации захочет узнать, произошло ли это на самом деле, и, если да, как это лицо смогло сделать то, что оно, по его словам, совершило. Эксперту нужно будет выполнить исследование работающей системы и проанализировать энергозависимые данные, например, запущенные процессы и сетевые соединения. Эксперты, возможно, также будут заинтересованы найти вредоносную программу, которая присутствует в памяти, но не записывает какую-либо информацию (например, в файлах журналов) или даже исполняемый файл на накопитель.

Еще одна причина для проведения исследования работающей системы – использование этого метода для начальной оценки инцидента. Специалисты по расследованию инцидентов имеют дело не только с большими емкостями запоминающих устройств, но и с более крупными и рассредоточенными инфраструктурами. Теперь системы приложений электронной коммерции могут работать не на серверах, расположенных на двух-трех полках в отдельном центре обработки данных, а состоять из кластеров, когда кластер приложений может находиться в одном здании, а кластер базы данных – в другом. Сетевые подключения в организации больше не реализуются посредством отдельного сегмента сети в здании; вместо этого несколько простых сетей охватывают несколько кварталов в городе или даже нескольких городах. По существу, специалистам по расследованию инцидентов нужно средство, используя которое они могли бы проанализировать эти системы и сократить объем исследуемых данных, определяя зараженные системы и ранжируя их по степени их приоритетности. Один из способов сделать это – использовать метод исследования работающей системы, чтобы определить местонахождение артефактов, относящихся к инциденту, таких как файл (или несколько файлов в системе), запущенная служба или выполняющийся процесс, раздел реестра и т. д. Например, в главе 4 рассматривается отдельная тема артефактов реестра, которые могут относиться к инциденту, связанному с кражей данных в организации. Если бы обнаружилось, что вор использовал съемное запоминающее устройство, например, iPod, можно было бы «прочесать» всю инфраструктуру, чтобы определить каждую систему, к которой подключалось это отдельное устройство, и время последнего отключения устройства от любой системы. Эта единственная проверка сократила бы количество систем, которые, возможно, заражены или вовлечены в инцидент, с нескольких тысяч (или, в некоторых случаях, нескольких сотен тысяч) до только тех, к которым вор на самом деле подключал устройство.

Какие данные нужно собрать

На данном этапе мы уже готовы рассмотреть типы энергозависимых данных, которые можно встретить в работающей системе, и узнать об инструментах, которые можно использовать для сбора этих данных во время исследования работающей системы.

При проведении исследования работающей системы вы, вероятнее всего, сначала захотите собрать содержимое физической памяти, или ОЗУ. Принимая во внимание принцип обмена Локара, достаточно ясно, что, собирая сначала содержимое ОЗУ, вы сводите к минимуму оказываемое на него воздействие. Вы уже знаете, что другие инструменты, которые вы запускаете для сбора другой энергозависимой информации, будут загружаться в память (как и инструмент, используемый для сбора содержимого ОЗУ), изменяя ее содержимое.

Мы обсудим тему сбора и анализа содержимого ОЗУ в главе 3. Вот перечень отдельных типов энергозависимых данных, которые мы рассмотрим в этой главе:

- системное время;
- пользователи, вошедшие в систему;
- открытые файлы;
- сетевая информация;
- сетевые соединения;
- информация о процессах;
- сопоставление процесса с портом;
- память процесса;
- состояние сети;
- содержимое буфера обмена;
- информация о службах и драйверах;
- история командной строки;
- подключенные сетевые накопители;
- общие ресурсы.

Мы познакомимся с несколькими инструментами, которые можно использовать для поиска этой информации в ОС Windows. В этой главе вы, по всей вероятности, заметите, что существует тенденция использовать инструменты с интерфейсом командной строки, а не с графическим интерфейсом пользователя. Возможно, вы думаете, что это связано с тем, что инструменты командной строки оставляют меньший «след в памяти», т. е. они потребляют меньший объем памяти, используют меньше библиотек динамической компоновки (DLL-файлы) и в целом оказывают меньшее воздействие на систему. Отчасти это так, но не забывайте, что фактический объем памяти, занимаемый отдельным инструментом, можно определить только посредством тестирования этого инструмента. На сегодняшний день у меня нет информации о том, что подобные тестирования проводились, а их результаты были опубликованы.

Предупреждение

Никогда не следует делать предположений об инструменте и о том, какой «след» оставляет в памяти этот инструмент, запущенный в системе. Без тщательного анализа и тестирования (информацию об анализе исполняемых файлов см. в главе 6) вы никогда не узнаете, какое воздействие оказывает исполняемый файл на систему или какие артефакты остаются после его использования.

Главная причина, по которой мы обращаем внимание на использование инструментов с интерфейсом командной строки, состоит в том, что эти инструменты обычно очень просты, реализовывают одну основную, конкретную функцию, и их выполнение намного проще автоматизировать посредством пакетных файлов или скриптов. Инструменты командной строки можно связать вместе с помощью пакетных файлов или скриптовых языков, а их выходные данные обычно отправляются на консоль

(т.е. стандартное устройство вывода), а затем могут быть перенаправлены в файл или сокет. Инструменты с графическим интерфейсом пользователя, с другой стороны, требуют, как правило, чтобы выходные данные охранялись в файле, так как в большинстве из них есть пункт «Файл» (“File”) с командами «Сохранить» (“Save”) и «Сохранить как» (“Save As”) в раскрывающемся меню. Большинство создателей инструментов с графическим интерфейсом пользователя не всегда разрабатывают свои продукты с учетом проблем расследования инцидентов или проведения судебной экспертизы. Одна из наших целей – свести к минимуму воздействие принимаемых нами мер на систему (особенно с учетом последующего клонирования и судебного анализа), поэтому нам нужно получать необходимые данные из системы как можно быстрее и эффективнее, а также избегать сохранять файлы в системе.

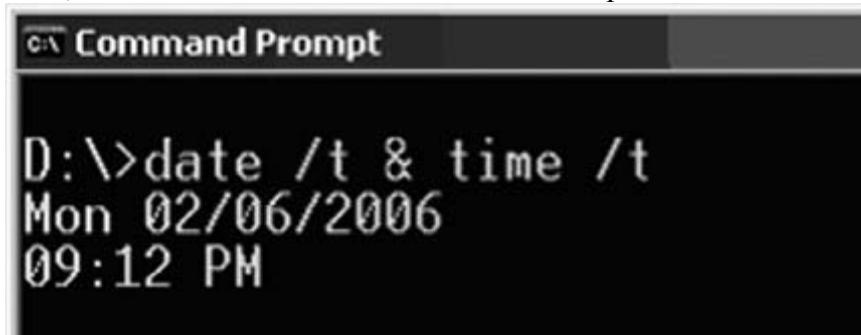
Это не значит, что инструменты с графическим интерфейсом пользователя ни в коем случае нельзя применять для исследования работающей системы. Если существует инструмент с графическим интерфейсом, который идеально отвечает вашим требованиям, то, конечно, вы можете его использовать. Но заранее подумайте о том, как вы будете получать данные из системы.

Независимо от того, какие инструменты вы решите использовать, никогда не забывайте ознакомиться с лицензионным соглашением перед началом работы с ними. Некоторые инструменты можно использовать без всяких ограничений, тогда как другие инструменты необходимо заплатить, если вы собираетесь работать с ними в корпоративной среде. Внимательно прочитав эти лицензионные соглашения заранее, вы сможете избежать серьезных проблем в будущем.

Системное время

Один из первых элементов информации, который нужно собрать во время расследования инцидента, – это системное время. Это предоставит вам много контекста к данным, которые будут собраны позднее во время расследования, и поможет вам разработать точную времененную шкалу событий, произошедших в системе.

На илл. 1.1., показан самый известный способ отображения системного времени.



Илл. 1.1. Отображение системной даты и системного времени в Windows XP.

Записки из подполья...

Получение системного времени

Системное время можно получить с помощью простого Perl-скрипта, например:

```
print localtime(time)."\n";
```

Этот скрипт показывает системное время в формате местного времени на основе заданных в системе параметров часового пояса и летнего времени, но время можно также отобразить в формате среднего времени по Гринвичу (Greenwich Mean Time, GMT), используя, например, следующий скрипт:

```
print gmtime(time)."\\n";
```

Perl-скрипт «*systime.pl*», расположенный на носителе, который идет в комплекте с этой книгой, показывает, как можно получить системное время, используя интерфейс прикладного программирования (API) Windows. «*systime.exe*» – отдельный исполняемый файл, скомпилированный из этого Perl-скрипта с помощью программы Perl2Exe.

Еще один способ получить эту информацию – использовать инструментарий управления Windows (WMI), чтобы получить доступ к классу *Win32_OperatingSystem* и отобразить параметр *LocalDateTime*.

Эксперту важно знать не только текущее системное время; количество времени, в течение которого работает система, может также предоставить большое количество контекста для расследования. Например, сравнив количество времени, в течение которого работает система, с количеством времени, в течение которого выполняется процесс, можно получить представление о том, когда попытка применения эксплойта или попытка взлома, возможно, завершилась успешно (подробнее о получении данных о процессах вы узнаете позднее в этой главе).

Кроме того, при записи системного времени эксперт должен указать реальное время. Оба эти значения позволяют эксперту позднее определить, было ли системное время правильным. Информация о так называемой разнице в показаниях часов (*clock skew*) позволяет получить более точное представление о фактическом времени, когда совершались события, зарегистрированные в файлах журналов. Эта информация может быть бесценна, когда вы пытаетесь объединить отметки времени из нескольких источников.

Еще один важный элемент связанной со временем информации – это параметры часового пояса, установленные на компьютере. Операционные системы Windows, использующие файловую систему NTFS, сохраняют отметки времени файлов в формате всемирного координированного времени (Universal Coordinated Time, UTC), который аналогичен формату GMT. Операционные системы, использующие файловую систему FAT, сохраняют отметки времени файлов на основе локального системного времени. Не менее важно учитывать эту информацию во время анализа образа данных (эта тема будет рассмотрена позднее в этой книге), но она также может иметь существенное значение при исследовании работающей системы удаленно, особенно если вы находитесь на расстоянии нескольких часовых поясов от исследуемой системы.

Инструменты и ловушки...

Инструменты и лицензирование

В этой главе мы будем обсуждать различные инструменты, которые можно использовать для сбора данных из работающей системы. Некоторые из этих инструментов встроены в систему, а другие, сторонние инструменты, доступны в Интернете. В большинстве случаев я пытаюсь предоставлять ссылки, по которым можно найти эти инструменты, но ссылки в Интернете могут изменяться или исчезать. При использовании сторонних инструментов вы должны быть осведомлены о лицензионных соглашениях, которые принимаете, загружая эти инструменты. В некоторых лицензионных соглашениях указано, что инструмент можно использовать как угодно, если только вы не являетесь консультантом и не используете этот инструмент в качестве консультанта. Инструменты, доступные на веб-сайтах Microsoft и Sysinternals, были недавно обновлены, и при первом запуске этих инструментов появляется диалоговое окно лицензионного соглашения. Это означает, что вы не только должны принять это лицензионное соглашение, но и, если запускаете инструмент из пакетного файла в системе, в которой вы раньше никогда не запускали этот инструмент, не забыть указать переключатель */accepteula* в командной строке. Если вы этого не сделаете, выполнение пакетного файла

зависнет. Кроме того, после того как вы примите лицензионное соглашение, используя диалоговое окно или переключатель командной строки, этот инструмент создаст в реестре раздел, чтобы зафиксировать тот факт, что вы приняли лицензионное соглашение.

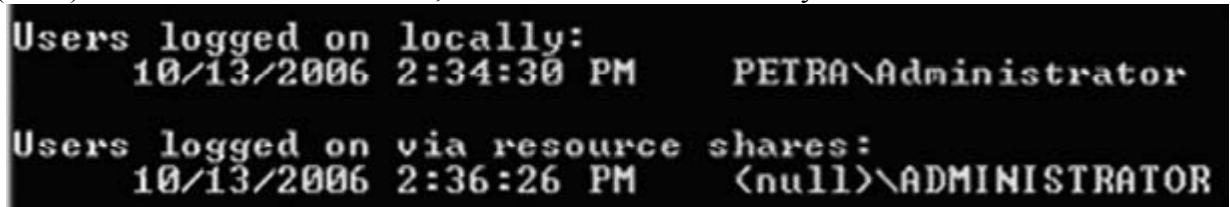
Пользователи, вошедшие в систему

Во время расследования вы, возможно, захотите узнать, какие пользователи вошли в систему и каким образом: локально (через консоль или клавиатуру) или удаленно (например, используя команду *net use* или подключенный сетевой ресурс). Эта информация позволит вам добавить контекст к другим данным, собранным из системы, например, контекст пользователя выполняющегося процесса, владелец файла или время последнего доступа к файлу. Эту информацию также полезно сопоставить с журналом событий безопасности, особенно если включен соответствующий аудит (аудит событий входа и выхода и т. д.).

PsLoggedOn

Возможно, самый известный инструмент для определения пользователей, вошедших в систему, – это «*psloggedon.exe*» (<http://technet.Microsoft.com/en-us/sysinternals/bb897545.aspx>). Этот инструмент показывает эксперту имя пользователя, вошедшего в систему локально (через клавиатуру), а также пользователей, выполнивших вход в систему удаленно (например, через подключенный сетевой ресурс).

Как показано на илл. 1.2., «*psloggedon.exe*» показывает пользователей, вошедших в систему удаленно. Чтобы продемонстрировать этот пример, я вошел в ОС Windows 2000 (Petra) из своей ОС Windows XP, а затем выполнил команду в целевой системе.



```

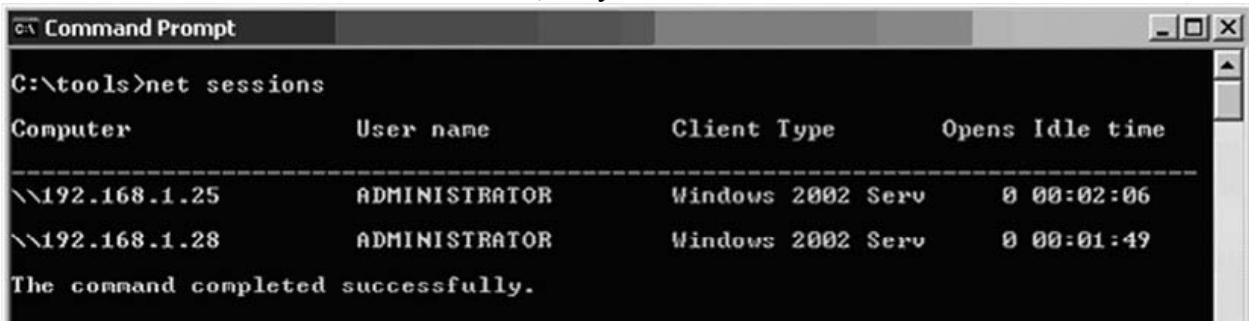
Users logged on locally:
10/13/2006 2:34:30 PM      PETRA\Administrator

Users logged on via resource shares:
10/13/2006 2:36:26 PM      <null>\ADMINISTRATOR
  
```

Илл. 1.2. Выходные данные инструмента «*psloggedon.exe*» в ОС Windows 2000.

Net Sessions

Net sessions – это собственная команда ОС Windows (посредством исполняемого файла «*net.exe*»), которую можно применять, чтобы увидеть не только имя пользователя, используемое для получения доступа к системе через удаленный сеанс входа, но и IP-адрес и тип клиента, из которого получен доступ к системе. На илл. 1.3., показаны выходные данные команды *net sessions*, запущенной в ОС Windows 2003.



Computer	User name	Client Type	Opens	Idle time
\\"192.168.1.25	ADMINISTRATOR	Windows 2002 Serv	0	00:02:06
\\"192.168.1.28	ADMINISTRATOR	Windows 2002 Serv	0	00:01:49

The command completed successfully.

Илл. 1.3. Выходные данные команды *net sessions* в ОС Windows 2003.

В выходных данных команды *net sessions* на илл. 1.3., показано, что вход в систему Windows 2003 выполнен из двух ОС Windows XP с использованием учетной записи администратора. Ни в одном из сеансов не были открыты файлы, но ни один из этих сеансов не продолжался слишком долго (о чем свидетельствуют значения времени, указанные в столбце «Время простоя» («Idle time») выходных данных).

LogonSessions

«logonsessions.exe» (доступный на веб-сайте Microsoft, <http://technet.microsoft.com/en-us/sysinternals/bb896769.aspx>) – это инструмент командной строки, перечисляющий все активные сеансы входа в системе. На илл. 1.4., показана часть выходных данных инструмента «logonsessions.exe» в ОС Windows XP (имя системы – Ender).

```
[6] Logon session 00000000:000478c7:
User name: ENDER\Harlan
Auth package: NTLM
Logon type: Interactive
Session: 0
Sid: S-1-5-21-1606980848-308236825-682003330-1004
Logon time: 2/6/2006 4:19:42 PM
Logon server: ENDER
DNS Domain:
UPN:
 304: C:\WINDOWS\system32\wscntfy.exe
1372: C:\WINDOWS\Explorer.EXE
1828: C:\Program Files\CyberLink\PowerDVD\DVDLauncher.exe
1252: C:\WINDOWS\BCMSMMSG.exe
 892: C:\Program Files\Synaptics\SynTP\SynTPLpr.exe
 788: C:\Program Files\Synaptics\SynTP\SynTPEnh.exe
1000: C:\WINDOWS\System32\spool\drivers\w32x86\3\hpztsb07.exe
1016: C:\Program Files\Viewpoint\Viewpoint Manager\ViewMgr.exe
1932: C:\Program Files\QuickTime\qttask.exe
 108: C:\Program Files\Real\RealPlayer\RealPlay.exe
 552: C:\Program Files\Mozilla Firefox\firefox.exe
```

Илл. 1.4. Выходные данные инструмента «logonsessions.exe» в ОС Windows XP.

«Logonsessions.exe» предоставляет намного больше информации, чем другие инструменты, как показано во фрагменте выходных данных на илл. 1.4.. Например, этот инструмент показывает используемый пакет проверки подлинности (информация о том, что используется пакет проверки подлинности Kerberos, а не LAN Manager может быть полезна для вашего расследования), тип входа в систему, активные процессы и т. д.

Совет

Файл «logonsess.txt», расположенный в каталоге \ch1\dat на носителе, который идет в комплекте с этой книгой, содержит выходные данные инструмента «logonsessions.exe» из системы Windows 2003, использовавшейся для примера на илл. 1.3.

Еще один полезный и удобный инструмент – это «netusers.exe», бесплатная утилита с сайта Somarsoft.com. Используя в «netusers.exe» переключатели *-local* и *-history*, можно получить краткий отчет о том, когда все локальные пользователи последний раз входили в систему. Время последнего входа в систему хранится в реестре. Мы подробно рассмотрим эту тему в главе 4. «Netusers.exe» позволяет найти эту информацию в работающей системе.

Однако имейте в виду, что эти инструменты не покажут вам пользователей, вошедших в систему через программу скрытого удаленного администрирования. Программы скрытого удаленного администрирования и трояны, например, печально известная программа SubSeven, позволяют пользователям «входить» в троянскую программу через прямое TCP-соединение в обход механизмов проверки подлинности Windows. Поэтому эти соединения не будут отображаться в таких инструментах, как «psloggedon.exe». Но выходные данные этих инструментов могут быть полезны, чтобы показать, что пользователь, которого вы ранее обнаружили в системе, не отображается в

этом списке. Эти выходные данные можно также использовать, чтобы показать скрытые функциональные возможности, даже если механизм для этих функциональных возможностей не был обнаружен.

Открытые файлы

Если выходные данные инструмента «psloggedon.exe» показывают, что пользователи вошли в систему удаленно, вы также захотите узнать, какие файлы они открыли, если это имело место. Часто, когда пользователь получает доступ к системе удаленно, он может выполнять поиск какой-либо конкретной информации или открывать файлы. Пользователь в корпоративной среде может иметь общий ресурс и позволять другим пользователям просматривать изображения, загружать музыкальные файлы и т. д. Злоумышленники могут «посещать» плохо защищенные системы Windows, такие как Windows 2000, подключенные к Интернету и не имеющие пароля администратора (а также брандмауэра), чтобы выполнить поиск файлов, получить к ним доступ и скопировать их. Как команда *net file*, так и инструмент «psfile.exe» (<http://technet.microsoft.com/en-us/sysinternals/bb896649.aspx>) или «openfiles.exe» (встроенный в Windows XP Pro и Windows 2003) покажут вам файлы, открытые в системе через удаленное соединение.

Сетевая информация (таблица кэшированных имен NetBIOS)

Иногда, когда злоумышленник получает удаленный доступ к системе, он хочет узнать, какие другие системы доступны в сети и какие системы можно «увидеть» (в сетевом смысле) посредством взломанной системы. В своей практике я часто сталкивался с разнообразными случаями: иногда в системе создавались и исполнялись пакетные файлы, в других случаях злоумышленник запускал команды *net view* с помощью внедрения SQL-кода (используя браузер, чтобы отправлять команды в систему через веб-сервер и сервер базы данных). Когда соединения с другими системами устанавливаются с использованием NetBIOS (так же, как при входах в систему, подключении к общим ресурсам и т. п.), системы сохраняют список других систем, которые они «видят». Просматривая содержимое таблицы кэшированных имен, можно определить другие системы, вовлеченные в инцидент.

Давайте рассмотрим один пример. Моя домашняя «сеть» состоит из одного ноутбука и нескольких сеансов VMware, которые отображаются как автономные системы в виртуальной сети. Чтобы продемонстрировать кэширование имен NetBIOS, я запустил сеанс VMware с ОС Windows 2000 и вошел в систему, чтобы просмотреть IP-адрес, назначенный через протокол DHCP. Затем я вернулся в хостовую операционную систему (Windows XP Pro SP2) и в командной строке ввел **nbtstat -A 192.168.1.22**, чтобы просмотреть таблицу имен в «удаленной» системе. Потом я ввел команду **nbtstat -c**, чтобы увидеть кэшированные имена NetBIOS в хостовой операционной системе. На илл. 1.5., показаны выходные данные этой команды.

NetBIOS Remote Cache Name Table			
Name	Type	Host Address	Life [sec]
PETRA	<20> UNIQUE	192.168.1.22	440

Илл. 1.5. Кэш таблицы имен NetBIOS.

Сейчас вы, возможно, думаете: «Ну и что? Почему это так важно?». Дело в том, что если бы я был злоумышленником и получил бы доступ к одной системе, я, возможно, был бы заинтересован в том, чтобы также получить доступ к другим системам. Для этого мне было бы необходимо узнать, какие системы находятся в сети и какие у них есть

уязвимости. По существу, я бы начал искать легкие цели. Однако, если бы я начал сканирование систем на наличие уязвимостей, я мог бы предупредить кого-нибудь о том, что происходит. Кроме того, для того чтобы выполнить сканирование на наличие уязвимостей, мне бы нужно было скопировать свои инструменты в уже взломанную систему, а это могло бы предупредить кого-нибудь о моих действиях. Но я могу использовать инструмент «nbtstat.exe», чтобы обнаружить потенциально уязвимые системы. Например, на илл. 1.6., показаны выходные данные команды, которую я запустил, чтобы получить кэшированные имена NetBIOS.

NetBIOS Remote Machine Name Table			
Name	Type	Status	
PETRA	<00>	UNIQUE	Registered
PETRA	<20>	UNIQUE	Registered
WORKGROUP	<00>	GROUP	Registered
PETRA	<03>	UNIQUE	Registered
WORKGROUP	<1E>	GROUP	Registered
I Net~Services	<1C>	GROUP	Registered
IS~PETRA.....	<00>	UNIQUE	Registered
WORKGROUP	<1D>	UNIQUE	Registered
...MSBROWSE...	<01>	GROUP	Registered
ADMINISTRATOR	<03>	UNIQUE	Registered

MAC Address = 00-0C-29-EC-6B-96

Илл. 1.6. Выходные данные команды *nbtstat -A 192.168.1.22*.

В выходных данных команды *nbtstat*, показанных на илл. 1.6., мы видим, что в систему вошел администратор и что в системе запущен веб-сервер IIS. Как специалисты по тестированию систем на проникновение, так и злоумышленники будут использовать информацию в таблице имен NetBIOS (<http://en.wikipedia.org/wiki/NetBIOS>) в любой системе, которую они смогут взломать, чтобы обнаружить другие уязвимые системы. Статьи № 163409 (<http://support.microsoft.com/kb/q163409>) и № 119495 (<http://support.microsoft.com/kb/119495/EN-US>) из базы знаний Microsoft предоставляют много информации о данных, имеющихся в таблице имен.

Сетевые соединения

Получив сообщение об инциденте, эксперт должен как можно быстрее собрать информацию об исходящих и о входящих сетевых соединениях системы, безопасность которой была нарушена. Эта информация может исчезать со временем, и, если пройдет слишком много времени, она будет потеряна. Эксперт может приступить к работе и, после первоначального осмотра, определить, что злоумышленник все еще находится в системе и получает к ней доступ. Или он может обнаружить, что червь или IRC-бот (вредоносная программа, которая, после установки в систему, создает исходящее соединение с IRC-сервером, ожидая команд) передает информацию из системы, выполняет поиск других систем, которые можно заразить, обновляется или входит в сервер контроля и управления. Эта информация может предоставить сведения или добавить контекст для других данных, уже собранных экспертом. Не в каждой системе будет установлен брандмауэр, а брандмауэр, настроенный так, чтобы регистрировать успешные входящие и исходящие соединения системы, встречается еще реже. Так же редко в системах можно увидеть такое

приложение, как Port Reporter (<http://support.microsoft.com/kb/837243>), предназначенное для записи и регистрации информации о сетевых соединениях. Эксперт должен быть готов действовать быстро и собирать нужную информацию эффективно и своевременно.

В моей практике было несколько случаев, когда клиент предоставлял мне файлы-образы, созданные из системы, и спрашивал: «Были ли скопированы конфиденциальные данные из системы?». Не имея хотя бы какой-нибудь сетевой информации, я всегда давал один и тот же ответ: «Это невозможно узнать». Я также был в нескольких ситуациях, когда наличие информации о сетевых соединениях значительно уменьшило бы область поиска, особенно если данные, которые сначала насторожили клиента и заставили его сообщить об инциденте, не имели ничего общего со взломом, обнаруженным впоследствии. В одной ситуации анализ временной шкалы образа системы показал, что злоумышленник получал доступ к системе через программу удаленного скрытого администрирования в то время, когда два разных администратора получали доступ к системе, чтобы исправить две отдельные проблемы. Наличие информации о входящих и об исходящих сетевых соединениях системы было бы чрезвычайно полезно для обнаружения основного вторжения.

Netstat

Netstat – это, возможно, самый известный инструмент для сбора информации о сетевых соединениях в системе Windows. Этот инструмент командной строки прост и удобен в использовании и предоставляет простое представление TCP- и UDP-соединений и их состояний, статистики сетевого трафика и другой подобной информации. «Netstat.exe» – это встроенный инструмент, то есть он входит в состав дистрибутива операционной системы.

Самый распространенный способ запуска netstat – использовать переключатели *-ano*, которые указывают программе отображать сетевые TCP- и UDP-соединения, прослушивающие порты и идентификаторы процессов (PID), использующие эти сетевые соединения. На илл. 1.7., показаны выходные данные команды *netstat -ano*.

C:\>netstat -ano

Active Connections

Proto	Local Address	Foreign Address	State	PID
TCP	0.0.0.0:135	0.0.0.0:0	LISTENING	1344
TCP	0.0.0.0:445	0.0.0.0:0	LISTENING	4
TCP	127.0.0.1:1026	0.0.0.0:0	LISTENING	1992
TCP	127.0.0.1:1031	127.0.0.1:1032	ESTABLISHED	2536
TCP	127.0.0.1:1032	127.0.0.1:1031	ESTABLISHED	2536
TCP	192.168.1.25:139	0.0.0.0:0	LISTENING	4
TCP	192.168.1.25:1310	216.239.51.104:80	ESTABLISHED	2536
TCP	192.168.1.25:1323	209.18.34.78:80	ESTABLISHED	2536
TCP	192.168.1.25:1326	209.18.34.41:80	ESTABLISHED	2536
TCP	192.168.1.25:1327	209.18.34.41:80	ESTABLISHED	2536
TCP	192.168.37.1:139	0.0.0.0:0	LISTENING	4
TCP	192.168.206.1:139	0.0.0.0:0	LISTENING	4

Илл. 1.7. Фрагмент выходных данных команды *netstat -ano* в ОС Windows XP.

Совет

При обычных обстоятельствах, ОС Windows 2000 не отвечает на переключатель *-o* во время выполнения инструмента «netstat.exe». Однако статья № 907980 из базы знаний Microsoft содержит описание исправления, которое позволяет версии «netstat.exe» в ОС Windows 2000 указывать идентификатор процесса, использующего сетевое соединение, перечисленное в выходных данных.

В выходных данных команды `netstat -ano` на илл. 1.7., показаны сетевые соединения, состояние каждого соединения и, в крайнем правом столбце, идентификаторы процессов, использующих порты. В выходных данных `netstat` эксперту нужно искать любые необычные соединения. Например, во многих пользовательских системах можно довольно часто увидеть соединения (исходящие из клиентского порта с большим номером) с удаленной системой, которые устанавливаются через порт 80. Идентификатор процесса, использующего это соединение, обычно соответствует веб-браузеру. Но эксперта можно легко ввести в заблуждение. Я расследовал случаи, когда инструмент «`wget.exe`» использовался для того, чтобы соединиться с удаленной системой через порт 80 и загрузить вредоносные и хакерские программы. Сами по себе и без дальнейшего изучения эти соединения выглядели бы для эксперта (и для системы обнаружения вторжений) как допустимый трафик браузера.

Совет

Статья № 137984 из базы знаний Microsoft (<http://support.microsoft.com/kb/137984>) содержит описание состояний, перечисленных в выходных данных инструмента «`netstat.exe`».

Команда `netstat`, с переключателем `-r`, покажет таблицу маршрутизации и все постоянные маршруты, разрешенные в системе, если таковые имеются. Такая команда может предоставить очень полезную информацию для эксперта или даже просто для администратора, выполняющего поиск и устранение неисправностей в системе. Мне встречались системы, которые были настроены так, чтобы передавать файлы в другое место как часть бизнес-процесса, и этот процесс работал только в том случае, если в системе был разрешен постоянный маршрут. Это было обусловлено тем, что этот постоянный маршрут перенаправлял определенный трафик через соединение виртуальной частной сети, а не через обычные маршруты из инфраструктуры. Определяя причину неисправности, которая была мне непонятна, я встретил постоянный маршрут и сказал об этом одному из специалистов по системам. Эта информация всколыхнула его память, и мы смогли найти и устранить неисправность.

Информация о процессах

Эксперт всегда захочет узнать, какие процессы выполняются в системе, безопасность которой, возможно, нарушена. Обратите внимание на слово *всегда*. При просмотре выполняющихся процессов в окне диспетчера задач можно увидеть некоторые сведения о каждом процессе. Однако во время экспертизы вы захотите собрать намного больше информации, которая не отображается в диспетчере задач, в том числе:

- полный путь к исполняемому образу (exe-файл);
- командную строку, используемую для запуска процесса, если это имело место;
- количество времени, в течение которого выполняется процесс;
- контекст безопасности/пользователя, в котором выполняется процесс;
- сведения о модулях, загруженных процессом;
- содержимое памяти процесса.

Диспетчер задач показывает только часть этой информации, но он не предоставляет все данные. Например, некоторые вредоносные программы устанавливаются под именем «`svchost.exe`», то есть, используя имя допустимого процесса в системах Windows (см. вставку «`Svchost`»). Исполняемый образ для этого процесса находится в каталоге «`system32`» и защищен функцией «Задача файлов Windows» (дополнительную информацию см. во вставке «Задача файлов Windows»). Это означает, что, пока функция «Задача файлов Windows» выполняется и в ее работу не вносится несанкционированных изменений, попытки заменить или модифицировать защищенный файл приведут к тому, что новый файл будет автоматически заменен «заведомо

исправной» копией из кэша, а в журнале регистрации событий будет создана запись об этом.

Почему это так важно? Если вы просматриваете список процессов в диспетчере задач, как вы определите, какой из этих процессов «подозрительный»? Простой способ найти подозрительные процессы – просмотреть полный путь к исполняемому файлу («svchost.exe» выполняющийся не из каталога C:\Windows\system32, будет подозрительным) или командную строку, которая использовалась для запуска процесса (например, команда «inetinfo.exe», запущенная с аргументами `-L -d -p 80 -e cmd.exe`, должна быть подозрительна для многих администраторов и экспертов; эта командная строка свидетельствует о том, что netcat используется в качестве программы скрытого удаленного аргументирования). Многие вредоносные программы маскируются, используя имена допустимых файлов. Например, червь W32/Nachi помещает копию утилиты Trivial File Transfer Protocol (TFTP) в каталог C:\Windows\system32\Wins и называет ее «svchost.exe». Когда эта программа выполняется, в диспетчере задач невозможно на самом деле отличить ее от допустимой версии «svchost.exe».

Ваш компьютер защищен?

Защита файлов Windows

Функция «Защиты файлов Windows» была добавлена в ОС Windows 2000 и также присутствует в ОС Windows XP и 2003. Вкратце, эта функция защищает критические системные файлы от случайного изменения или удаления. Если система не была взломана так, что работа функции «Защиты файлов Windows» может быть нарушена, то при попытке изменить или удалить защищенный файл функция активируется и автоматически заменяет этот файл заведомо исправной копией из кэша. Затем в журнале событий создается запись о событии с идентификатором 64001 (<http://support.microsoft.com/default.aspx?scid=kb;en-us;236995>).

Статья № 222193, озаглавленная «Description of the Windows File Protection feature» (<http://support.microsoft.com/kb/222193/EN-US>), из базы знаний Microsoft содержит исчерпывающее описание этой функции, а также различных разделов реестра, связанных с ней.

В главе 5 более подробно рассматривается функция защиты файлов Windows, распространенный способ, применяемый для нарушения ее работы, и средства для обнаружения использования этого способа.

Теперь давайте рассмотрим несколько инструментов, которые можно использовать для просмотра более детальной информации о процессах.

Tlist

Инструмент «tlist.exe», включенный в состав средств отладки от Microsoft (www.microsoft.com/whdc/devtools/debugging/default.mspx), показывает большое количество информации о выполняющихся процессах. Например, переключатель `-v` позволяет эксперту узнать идентификатор сеанса, идентификатор процесса, имя процесса, связанные с процессом службы, а также командную строку, использовавшуюся для запуска процесса, как показано ниже:

```
0      344  svchost.exe Svcs: LmHosts,SSDPNS,WebClient
Command Line: C:\WINDOWS\System32\svchost.exe -k LocalService
```

Другие переключатели покажут эту информацию по отдельности. Переключатель `-c` покажет только командную строку, использовавшуюся для запуска каждого процесса, а переключатель `-s` покажет связанные службы (или заголовок окна, если с процессом не

связано никаких служб). Переключатель *-t* покажет дерево задач, перечисляя каждый процесс под его родительским процессом, следующим образом:

```
System (4)
  smss.exe (628)
    csrss.exe (772)
    winlogon.exe (1056)
    services.exe (1100)
      svchost.exe (1296)
      svchost.exe (1344)
      svchost.exe (1688)
    wsctfy.exe (1184)
```

Инструмент «tlist.exe» также позволяет выполнить поиск всех процессов, имеющих отдельный загруженный модуль, используя переключатель *-m*. Например, файл «wsock32.dll» предоставляет функциональные возможности для работы в сети и описан как 32-разрядная библиотека Windows Socket. Чтобы перечислить все процессы с этим загруженным модулем, введите следующую команду:

```
D:\tools>tlist -m wsock32.dll
```

Эта команда возвращает идентификатор и имя для каждого процесса, например:

WSOCK32.dll - 1688 svchost.exe	
wsock32.dll - 344 svchost.exe	
WSOCK32.dll - 1992 alg.exe	
WSOCK32.dll - 1956 explorer.exe	Program Manager
wsock32.dll - 452 ViewMgr.exe	AXTimer
WSOCK32.dll - 480 realplay.exe	

Tasklist

Утилита «tasklist.exe», включенная в состав Windows XP Pro и Windows 2003 (как видите, она отсутствует в Windows XP Home), является заменой для инструмента «tlist.exe». Различия между этими двумя инструментами едва различимы и в основном касаются имен и способов реализации переключателей. Однако «tasklist.exe» предоставляет опции для форматирования выходных данных с возможностью выбора таких форматов, как таблица, CSV (значения, разделенные запятыми) и список. Переключатель */v* (от англ. *verbose* – подробный) предоставляет большую часть информации о перечисленных процессах, в том числе имя образа (но не полный путь к нему), идентификатор, имя и номер сеанса для процесса, статус процесса, имя пользователя контекста, в котором выполняется процесс, и заголовок окна, если процесс имеет графический интерфейс. Эксперт может также использовать переключатель */svc*, чтобы получить информацию о службах для каждого процесса.

PsList

Инструмент «pslist.exe» (<http://technet.microsoft.com/en-us/sysinternals/bb896682.aspx>) показывает основную информацию о выполняющихся процессах в системе, в том числе количество времени, в течение которого выполняется каждый процесс (как в режиме пользователя, так и в режиме ядра). Переключатель *-x* отображает подробности о потоках и памяти, используемой каждым процессом. «Pslist.exe», запущенный с переключателем *-t*, покажет дерево задач почти в таком же виде, как и инструмент «tlist.exe». Однако этот инструмент не предоставляет такой информации о процессе, как путь к исполняемому образу, командная строка, использовавшаяся для запуска процесса, или пользовательский контекст, в котором выполняется процесс.

ListDLLs

Инструмент «listdlls.exe» (<http://technet.microsoft.com/en-us/sysinternals/bb896656.aspx>) перечисляет модули или DLL-файлы, используемые процессом. Он покажет полный путь к образу загруженного модуля, а также то, отличается ли версия DLL-файла, загруженного в память, от образа файла, присутствующего на диске. Эта информация может быть чрезвычайно важна для эксперта, так как каждая программа загружает или «импортирует» определенные DLL-файлы. Эти DLL-файлы предоставляют фактический код для использования, поэтому разработчикам приложений не нужно заново писать общие функции каждый раз, когда они создают новое приложение. Каждый DLL-файл предоставляет определенные функции, перечисляя их в своей таблице экспорта, а программы получают доступ к этим функциям, указывая DLL-файл и эти функции в своих таблицах импорта. Это позволяет вам увидеть (при помощи соответствующего инструмента), какие DLL-файлы загружает программа. Однако некоторые программы могут загружать дополнительные DLL-файлы, которые не включены в таблицу импорта; например, браузер Internet Explorer может загружать панели инструментов и объекты модуля поддержки браузера, код для которых указан в DLL-файлах. Шпионские программы, трояны и даже руткиты используют способ, который называется *внедрение DLL*, чтобы загружаться в область памяти запущенных процессов таким образом, что они будут работать и выполняться, но не будут отображаться в списке процессов, потому что фактически являются частью другого процесса. Это не то же самое, что дочерний процесс (как показано в выходных данных инструмента «tlist.exe», запущенного с переключателем *-t*), так как выполняющаяся вредоносная программа не имеет своего собственного идентификатора процесса.

Часть выходных данных, отображаемых инструментом «listdlls.exe», включает в себя командную строку, использовавшуюся для запуска каждого процесса, как показано ниже:

```
svchost.exe pid: 1292
Command line: C:\WINDOWS\system32\svchost -k DcomLaunch
```

Используя «listdlls.exe» (с переключателем *-d dllname*), можно перечислить процессы, загрузившие отдельный DLL-файл, примерно таким же образом, как в инструменте «tlist.exe». Эта функция может быть чрезвычайно полезна в ситуациях, когда эксперт обнаруживает отдельный DLL-файл и хочет узнать, загружали ли его какие-нибудь другие процессы.

Handle

Инструмент «handle.exe» (<http://technet.microsoft.com/en-us/sysinternals/bb896655.aspx>) показывает различные дескрипторы, открытые процессом в системе. Сюда входят не только открытые дескрипторы файлов (для файлов и каталогов), но и порты, разделы реестра и потоки. Эта информация может быть полезна для того, чтобы определить, к каким ресурсам получает доступ процесс во время своей работы. На илл. 1.8., показан фрагмент выходных данных инструмента «handle.exe», запущенного без переключателей в ОС Windows XP SP2.

Илл. 1.8. Фрагмент выходных данных инструмента «handle.exe».

На илл. 1.8., показано несколько дескрипторов, открытых процессом «svchost.exe», – в данном случае это несколько файлов журналов в каталоге «Windows». Когда я писал эту книгу, например, один из дескрипторов, открытых процессом «winword.exe», содержал полный путь к документу Microsoft Word.

Для «handle.exe» есть несколько полезных переключателей, таких как *-a*, чтобы показать все дескрипторы, и *-u*, чтобы показать имя пользователя-владельца для каждого дескриптора.

Инструменты и ловушки...

Процессы и инструментарий управления Windows (WMI)

Perl-скрипт «proc.pl», расположенный в каталоге \ch2\code на носителе, который идет в комплекте с этой книгой, показывает, как можно использовать язык Perl для реализации WMI и поиска информации о процессах через класс *Win32_Process*. Как скрипт, так и отдельный исполняемый файл с именем «proc.exe» (скомпилированный из Perl-скрипта при помощи программы Perl2Exe и также доступный на сопроводительном носителе) показывают следующую информацию о процессе: идентификатор, имя, пользовательский контекст, идентификатор родительского процесса, командная строка (при наличии таковой), путь к исполняемому образу (при наличии такового) и сведения о связанных службах.

Можно запустить как скрипт, так и исполняемый файл локально или в удаленной системе. Просто введите имя исполняемого файла, чтобы запустить его в локальной системе. Синтаксис для запуска исполняемого файла в удаленной системе следующий:

```
C:\tools>proc <система> <пользователь> <пароль>
```

Пример для запуска исполняемого файла показан ниже:

```
C:\tools>proc WebSvr Administrator пароль
```

Фрагмент выходных данных инструмента «proc.exe»:

PID	:	668
Name	:	spoolsv.exe
User	:	NT AUTHORITY\SYSTEM
Parent PID	:	1100 [services.exe]
CmdLine	:	C:\WINDOWS\system32\spoolsv.exe
Exe	:	C:\WINDOWS\system32\spoolsv.exe
Services	:	Spooler

Можно легко изменить скрипт, чтобы его выходные данные отображались в другом формате, например, в CSV, который подходит для открытия и анализа данных в электронной таблице.

Perl-скрипт «procmon.pl» (и соответствующий исполняемый файл, «procmon.exe»), расположенный в том же каталоге, представляет собой интересный пример использования инструментария WMI, чтобы отслеживать создание процессов в локальной системе. Просто запустите инструмент «procmon.exe» из командной строки, и во время выполнения он создаст отчет, содержащий идентификатор, пользовательский контекст и путь к исполняемому файлу (и командную строку) нового процесса, как показано ниже:

PID	USER	PROCESS
---	-----	-----
3208	Harlan	C:\WINDOWS\system32\cmd.exe ("C:\WINDOWS\system32\cmd.exe")
1768	Harlan	C:\WINDOWS\system32\ping.exe (ping 192.168.1.1)
3100	Harlan	C:\WINDOWS\system32\sol.exe (sol)

Такие инструменты, как «procmon.exe», чрезвычайно полезны в том отношении,

что их можно использовать, чтобы расширить аудит создания процессов и получить представление о процессах, создаваемых во время установки приложений и вредоносных программ.

Теперь должно быть ясно, что не существует единого инструмента или средства, отображающего всю информацию, которую вы, возможно, захотите узнать о процессах, найденных во время исследования системы. Возможно, вы захотите запустить только один инструмент для быстрого обзора («tlist.exe» или «tasklist.exe» будет подходящим вариантом) или несколько инструментов (например, «pslist.exe» с переключателем *-x* и утилиту «listdlls.exe»). В зависимости от степени детализации, которая необходима для расследования, вы, возможно, также захотите запустить инструмент «handle.exe». Степень детализации информации, которую вы хотите получить, будет зависеть от вашего расследования. Мы подробнее обсудим эту тему позднее в этой главе, а также в главе 3, где рассмотрим вопросы сопоставления и анализа данных.

Инструменты и ловушки...

Svchost

Svchost – это процесс, который часто выполняется в системах Windows 2000, XP и 2003. Он несколько раз отображается в диспетчере задач: два или более раза в ОС Windows 2000 в стандартной конфигурации (без других установленных приложений), пять раз в ОС Windows XP и семь раз в ОС Windows 2003. Каждый экземпляр «svchost.exe» выполняет одну или несколько служб, что можно увидеть, запустив команду *tasklist /svc* в ОС Windows XP Pro и 2003 или *tlist -s* в ОС Windows 2000.

Статья № Q314056 (<http://support.microsoft.com/default.aspx?scid=kb;en-us;314056>) из базы знаний Microsoft предоставляет дополнительную информацию о процессе «svchost.exe» в ОС Windows XP, а статья № Q250320 (<http://support.microsoft.com/default.aspx?scid=kb;en-us;250320>) содержит подобную информацию относительно ОС Windows 2000.

Вкратце, «svchost.exe» предоставляет общий процесс для выполнения служб из DLL-файлов. Каждый экземпляр svchost может выполнять одну или несколько служб. При запуске svchost считывает следующий раздел реестра, чтобы получить группы служб, которые он должен выполнить:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Svchost
```

Обратите внимание на пробел в имени «Windows NT». Это очень важно, так как по умолчанию не существует раздела Software\Microsoft\WindowsNT.

Некоторые троянские программы и программы скрытого удаленного администрирования пытаются копировать себя в целевую систему, используя имя файла «svchost.exe». Backdoor.XTS и Backdoor.Litmus – примеры вредоносных программ, которые пытаются спрятаться за именем «svchost.exe», вероятнее всего, из-за того, что администраторы и эксперты не будут удивлены увидеть несколько экземпляров процесса svchost, перечисленных в диспетчере задач. Как показывает практика, копирование ложного файла «svchost.exe» в каталог «system32» в системах Windows является просто неудачным решением, так как этот файл защищен функцией «Защита файлов Windows» в ОС Windows 2000, XP и 2003.

Совет

«Pulist.exe» – это инструмент из пакета Resource Kit, указывающий процессы, выполняющиеся в системе, а также контекст пользователя (учетная запись пользователя, от имени которой выполняется процесс) для каждого процесса.

Сопоставление процесса с портом

Когда сетевое соединение открыто в системе, некий процесс должен быть ответственным за это соединение и использовать его. То есть каждое сетевое соединение и открытый порт связаны с процессом. Эксперту доступно несколько инструментов, чтобы выполнить это сопоставление процесса с портом.

Netstat

В ОС Windows XP и Windows 2003 программа «netstat.exe» предлагает переключатель *-o*, чтобы показать идентификатор процесса, ответственного за сетевое соединение. После того как вы соберете эту информацию (см. команду *netstat -ano* выше), вам нужно будет сопоставить ее с выходными данными такого инструмента, как «*tlist.exe*» или «*tasklist.exe*», чтобы определить имя процесса, использующего это соединение, и дополнительную информацию об этом процессе.

В ОС Windows XP с пакетом обновления версии 2 (SP2) и выше есть дополнительный переключатель *-b*, который «покажет исполняемый файл, участвующий в создании каждого соединения или прослушивающего порта». Этот переключатель также включен в состав «*netstat.exe*» в ОС Windows 2003 SP1 и может предоставлять дополнительную информацию о процессе, использующем отдельный порт. В некоторых случаях в выходных данных будут также показаны модули (DLL-файлы), используемые процессом. На илл. 1.9., показан фрагмент выходных данных команды, запущенной в ОС Windows XP SP2.

TCP	192.168.1.8:1036	205.188.69.61:5190	ESTABLISHED	1976
	[AOLSoftware.exe]			
TCP	192.168.1.8:1038	64.12.189.249:443	ESTABLISHED	1976
	[AOLSoftware.exe]			
TCP	192.168.1.8:1039	64.12.25.220:5190	ESTABLISHED	3624
	[aim6.exe]			
TCP	192.168.1.8:2702	199.45.62.18:80	TIME_WAIT	0
TCP	192.168.1.8:2703	199.45.62.18:80	TIME_WAIT	0
TCP	192.168.1.8:2706	213.200.97.206:80	TIME_WAIT	0
TCP	192.168.1.8:2707	213.200.109.28:80	TIME_WAIT	0
TCP	192.168.1.8:2709	213.200.109.28:80	TIME_WAIT	0
UDP	0.0.0.0:1182	*:*		1752
C:\WINDOWS\system32\mswsock.dll				
c:\windows\system32\WS2_32.dll				
c:\windows\system32\DNSAPI.dll				
c:\windows\system32\dnssrserv.dll				
C:\WINDOWS\system32\RPCRT4.dll				
[svchost.exe]				

Илл. 1.9. Фрагмент выходных данных команды *netstat -anob*, запущенной в ОС Windows XP SP2.

Fport

«Fport.exe» давно является одним из самых популярных инструментов для сопоставления процесса с портом в системе Windows. Выходные данные этого инструмента легко понять; однако чтобы получить их, инструмент нужно запускать, используя учетную запись администратора. Это может быть проблемой, если вы проводите расследование инцидента в ситуации, когда учетная запись, использовавшаяся для входа в систему, является учетной записью пользователя и не имеет прав администратора.

Tcpvcon

Инструмент «tcpvcon.exe» доступен на сайте Microsoft (первоначально он был одним из инструментов, предоставляемых на сайте Sysinternals.com) и является одним из лучших средств для получения из ОС Windows информации о сопоставлении процесса с портом. По умолчанию «tcpvcon.exe» покажет только информацию о TCP-соединениях и выведет на консоль информацию в формате табличного списка, как показано ниже:

```
[TCP] C:\Program Files\Mozilla Firefox\firefox.exe
PID: 3476
State: ESTABLISHED
Local: wintermute.adelphia.net:5918
Remote: yahoo.com: http
[TCP] C:\Program Files\Mozilla Firefox\firefox.exe
PID: 3476
State: ESTABLISHED
Local: wintermute.adelphia.net:5919
Remote: yahoo.com: http
```

Используя переключатели *-a* и *-c*, можно указать инструменту «tcpvcon.exe» отображать сведения обо всех соединениях (как TCP, так и UDP) в CSV-формате, в котором данные легко анализировать (конечно же, с помощью языка Perl) или просматривать в программе Excel. Используя переключатель *-n*, можно указать инструменту «tcpvcon.exe» не преобразовывать IP-адреса в имена, поэтому выходные данные появятся немного быстрее. Эти выходные данные легко проанализировать, используя несколько инструментов.

Совет

В большинстве случаев вам нужно будет получить IP-адрес удаленной системы, чтобы однозначно определить эту систему. Однако иногда вы, возможно, также захотите задокументировать доменное имя удаленной системы, так как некоторые злоумышленники или авторы вредоносных программ используют DNS-серверы и это имя может быть со временем более полезным, чем IP-адрес.

Совет

Если вы проводите исследование ОС Windows 2000, в которой не установлено исправление, упомянутое мной ранее в этой главе, предназначенное для того, чтобы инструмент «netstat.exe» мог указать идентификатор процесса для каждого сетевого соединения, отличным вариантом будет использовать «tcpvcon.exe».

Так как каждая запись о сетевом соединении размещается в отдельной строке, выходные данные легко анализировать, используя инструменты автоматизации (см. главу 2).

Имейте в виду, что такие инструменты, как «tcpvcon.exe», используют API в DLL-файлах, встроенных в систему, чтобы извлечь их информацию. Исходя из этого, вам, возможно, также потребуется использовать инструмент сканирования портов, такой как Nmap (www.nmap.org), чтобы удаленно собрать информацию об открытых портах из потенциально взломанной системы. При этом можно обнаружить ряд портов, открытых в режиме прослушивания и ожидающих соединения. Такое поведение характерно не только для веб-серверов и FTP-серверов, но и для программ скрытого удаленного администрирования. Если вы сканируете систему и обнаруживаете, что определенные порты открыты, но ни netstat, ни любой другой инструмент, показывающий сетевые соединения или информацию о сопоставлении процесса с портом, не отображает тот же самый открытый порт, то вы определенно столкнулись с головоломкой. В данном случае

следует перепроверить результаты сканирования и убедиться, что вы сканировали ту систему, которую нужно (да-да, бывает и такое). Если проблема не исчезает, то, возможно, вы имеете дело с руткитом (подробную информацию о руткитах см. в главе 7).

Совет

Сравнение перехваченных сетевых пакетов или результатов сканирования портов с выходными данными инструмента «netstat.exe» или «fcsrvcon.exe» – отличный способ подтвердить полученные данные. Во время одного из моих последних дел, сотрудники клиента собрали информацию из перехваченных сетевых пакетов и журналов сетевых устройств, а затем сопоставили эту информацию с отдельными системами. Используя выходные данные инструмента «netstat.exe», они смогли подтвердить, что имеют дело с верными системами, так как они могли ясно увидеть обозначения IP-адресов и портов источника и назначения. Они также подтвердили тот факт, что системы не заражены руткитами, которые обычно пытаются скрыть процессы, файлы, сетевые соединения и разделы реестра.

Память процесса

Работающая система может иметь любое количество выполняющихся процессов и любой из этих процессов может быть подозрительным или вредоносным по своей природе. Когда процесс выполняется в системе, ему чаще всего присваивается то же имя, что и файлу, в котором расположен исполняемый образ, а в системах Windows в особенности файл может быть назван как угодно. Злоумышленники не настолько любезны, чтобы давать своему вредоносному коду имя, которое было бы легкоизвестным, например, «badstuff.exe» (от англ. *bad stuff* – вред). В большинстве случаев они присваивают своему файлу менее броское имя или пытаются скрыть намерение программы, используя имя приложения, обычно встречающегося в системах Windows (смотри вставку «Svchost»).

После того как вы используете инструменты, рассмотренные выше, и обнаружите процесс, который по вашему мнению является подозрительным, вы, возможно, решите, что вам нужна дополнительная информация о том, чем занимается этот процесс. Эту информацию можно получить, создав дамп памяти, используемой процессом. Для выполнения этой задачи существует несколько инструментов. Как отмечалось выше, тема сбора содержимого ОЗУ (а также памяти, используемой отдельными процессами) будет подробно рассмотрена в главе 3.

Состояние сети

Эксперту чрезвычайно важно получить информацию о состоянии сетевых интерфейсных плат, подключенных к системе. Например, сегодня многие ноутбуки выпускаются со встроенными беспроводными сетевыми платами, поэтому невозможно, просто взглянув на компьютер, определить, подключен ли компьютер к точке доступа беспроводной сети, и, если да, то какой IP-адрес он использует. Зная состояние сетевой платы до того, как данные системы будут клонированы, вы сможете получить представление о последующих этапах экспертизы.

Ipconfig

Инструмент «ipconfig.exe», встроенный в ОС Windows, можно использовать для того, чтобы получить информацию о сетевых интерфейсных платах и их состоянии. Самый полезный переключатель для эксперта – это */all*, который применяется, чтобы показать конфигурацию сетевых плат в системе. Таким образом, можно узнать, используется ли DHCP, состояние сетевой платы, IP-адрес сетевой платы и многое другое.

Эта информация может оказаться полезной во время расследования, так как, возможно, вам придется анализировать журналы сетевого трафика, а IP-адрес системы в определенный момент времени мог быть изменен. Кроме того, многие веб-службы электронной почты (такие как Yahoo! Mail) записывают IP-адрес системы, из которой пользователь создал сообщение электронной почты, в заголовке этого сообщения. Однажды я принимал участие в расследовании случая, когда бывший сотрудник отправлял раздражающие (но не оскорбительные) сообщения электронной почты в нашу компанию. Просмотрев заголовки электронной почты, мы смогли определить, откуда он отправлял эти сообщения. Некоторые из них были отправлены из местного копировального центра, а другие – из местной публичной библиотеки. Благодаря помощи администраторов копировального центра и библиотеки, мы смогли еще больше сузить область поиска. В случае с публичной библиотекой мы смогли определить отдел библиотеки и то, что компьютер, который использовался, находился на 2-м этаже (один из администраторов попросил сотрудника библиотеки вводить команду `ipconfig /all` на нескольких системах, пока он не нашел IP-адрес, о котором идет речь). Излишне говорить, что бывший сотрудник был потрясен, когда ему предъявили эту информацию, и перестал присыпать эти сообщения электронной почты. Если бы он не был уволен и отправлял бы эти сообщения со своего рабочего компьютера через службу Yahoo! Mail, мы бы также смогли определить его местонахождение.

PromiscDetect and Promqry

Иногда во взломанных системах может быть установлен анализатор сетевых пакетов, предназначенный для того, чтобы перехватывать сетевой трафик (например, учетные данные для входа в другие системы), или узнать, что представляют собой другие системы в сети и какие службы в них выполняются. Возможность перехватывать и анализировать пакеты включена в код некоторых вредоносных программ, или злоумышленники самостоятельно загружают и устанавливают анализаторы. Для того чтобы сетевая интерфейсная плата перехватывала трафик таким образом, она должна работать в неизбирательном режиме. Администратор или эксперт может не увидеть этого, так как ничего не указывает на то, что сетевая плата работает в таком режиме. Отсутствует значок в области уведомлений или элемент панели управления, который бы явно указывал эксперту на то, что система используется для перехвата и анализа трафика.

Существуют инструменты, способные определить, работает ли сетевая плата в неизбирательном режиме. Один из таких инструментов – это «`kpromiscdetect.exe`» (www.ntsecurity.nu/toolbox/promiscdetect/). Другой инструмент – это «`gromqry.exe`» (доступный на веб-сайте Microsoft), написанный Тимом Рэйнсом (Tim Rains). Главное различие между двумя этими инструментами состоит в том, что «`gromqry.exe`» можно запускать для анализа удаленных систем, позволяя администратору проверить домен на наличие систем, которые, возможно, перехватывают и анализируют сетевой трафик.

Инструменты и ловушки...

Неизбирательный режим

Perl-скрипт «`ndis.pl`» (расположенный в каталоге `\ch2\code` на носителе, который идет в комплекте с этой книгой) реализовывает код инструментария WMI, чтобы определить параметры сетевой интерфейсной платы. В частности, он предназначен для того, чтобы определить, работает ли сетевая плата в неизбирательном режиме и способна ли она перехватывать и анализировать сетевые пакеты.

Файл «`ndis.exe`» в том же каталоге – это отдельная исполняемая версия этого скрипта, предоставленная для тех, у кого не установлен интерпретатор языка Perl в системе Windows.

На илл. 1.10., показан фрагмент выходных данных, возвращенных из инструмента

«ndis.exe».

Выходные данные, показанные на илл. 1.10., были созданы посредством запуска анализатора пакетов Wireshark (ранее известном как Ethereal; www.wireshark.org) на компьютере с беспроводной сетевой платой и последующего выполнения инструмента «ndis.exe». Выделенная часть выходных данных наглядно показывает, что беспроводная сетевая плата работает в неизбирательном режиме.

Как Perl-скрипт, так и соответствующий исполняемый файл предназначены только для запуска в локальной системе. Однако, незначительные изменения в коде позволяют запустить скрипт (или исполняемый файл после изменения и повторной компиляции скрипта) для анализа удаленных систем таким же образом, как и инструмент «promqry.exe».

```
Dell Wireless WLAN 1350 WLAN Mini
    NDIS_PACKET_TYPE_MULTICAST
    NDIS_PACKET_TYPE_DIRECTED
--> NDIS_PACKET_TYPE_PROMISCUOUS <--
    NDIS_PACKET_TYPE_BROADCAST

VMware Virtual Ethernet Adapter for VMnet8
    NDIS_PACKET_TYPE_MULTICAST
    NDIS_PACKET_TYPE_DIRECTED
    NDIS_PACKET_TYPE_BROADCAST

Broadcom 440x 10/100 Integrated Controller
    NDIS_PACKET_TYPE_MULTICAST
    NDIS_PACKET_TYPE_DIRECTED
    NDIS_PACKET_TYPE_BROADCAST
```

Илл. 1.10. Фрагмент выходных данных инструмента «ndis.exe» в ОС Windows XP.

Подобные инструменты можно также использовать для того, чтобы определить, какие активные сетевые интерфейсы, возможно, имеются в работающей системе. Для того чтобы иметь возможность подключиться к беспроводной сети, к моему старому компьютеру Toshiba Tecra 8100 нужно было подключить плату PCIMCIA, тогда как многие современные системы, с которыми я имел дело, выпускаются со встроенной поддержкой беспроводной сети. Вы не увидите устройств, подключенных к самому корпусу ноутбука, или мерцающих индикаторов, как в разъеме RJ-45 при Ethernet-подключении. Такая возможность просто встроена в компьютер. Поэтому когда какой-нибудь сотрудник на совещании сидит за своим ноутбуком, он может, как делать заметки, так и просматривать веб-страницы в Интернете или отправлять электронную почту. Доступ к беспроводной сети становится все более распространенным не только потому, что такая возможность предоставляется во многих местах, но и потому, что поддержка такой возможности встраивается прямо в ноутбуки.

Доступ к беспроводной сети может быть входом в вашу организацию или даже маршрутом, которым кто-нибудь пользуется, чтобы получать информацию из вашей инфраструктуры. В моей практике был случай, когда сотрудница по связям с общественностью в нашей компании решила, что ей нужно брать свой личный ноутбук на

совещания, чтобы иметь возможность получать доступ в Интернет. Однако она приняла такое решение, не связавшись ни с сотрудниками ИТ-отдела, ни даже со мной (я был администратором системы безопасности). Когда она включила свой ноутбук, то обнаружила точки доступа беспроводной сети нашей компании, защищенные WEP-ключами и фильтрацией MAC-адресов. Во время совещания этой сотруднице срочно понадобился доступ в Интернет, и, так как она не связалась с нами заранее, она решила подключиться к открытой точке доступа беспроводной сети, которую обнаружила ее система; эта точка доступа принадлежала компании, расположенной по соседству и была абсолютно не защищена. Установив это соединение, она создала точку входа в нашу инфраструктуру, которая обходила все применяемые нами механизмы защиты, в том числе брандмауэры и антивирусные приложения. В данном случае было трудно сказать, какая ситуация наносит больший ущерб – то, что ее подключение могло использоваться как канал для заражения нашей инфраструктуры, или правовые последствия в том случае, если безопасность инфраструктуры другой компании была бы нарушена, а анализ механизмов журналирования показал бы, что наша сотрудница была в то время подключена к их сети.

Во время расследования, как правило, рекомендуется собрать информацию об активных сетевых интерфейсах в исследуемой системе. Это добавит контекст не только к собираемым энергозависимым данным, но и к анализу образа НЖМД (этот тема будет рассмотрена позднее в этой книге).

Содержимое буфера обмена

Буфер обмена – это просто область памяти, где могут храниться данные для последующего использования. Большинство приложений ОС Windows предоставляют эту функциональную возможность посредством опции «Правка» (“Edit”) в строке меню. После нажатия пункта меню «Правка» (“Edit”) появляется раскрывающее меню с такими командами, как «Вырезать» (“Cut”), «Копировать» (“Copy”) и «Вставить» (“Paste”). В состав приложения Microsoft Word 2003 включена функция «Буфер обмена Office» (“Office Clipboard”).

Буфер обмена чаще всего используется для того, чтобы упростить перемещение данных определенным образом – между документами или между окнами приложений на рабочем столе. Пользователь выделяет текст или другие данные, выбирает команду «Копировать» (“Copy”), а затем – «Вставить» (“Paste”), чтобы вставить эти данные в какое-либо другое место. Команда «Вырезать» (“Cut”) убирает данные из документа, в котором работает пользователь, и перемещает их в буфер обмена.

Многие пользователи не понимают одного: однажды утром они могут включить свой компьютер, поработать с файлом, а затем копировать какую-нибудь информацию в буфер обмена. Предположим, пользователь редактирует документ, содержащий конфиденциальные данные, и личные сведения о клиенте нужно добавить в этот документ. Пользователь находит и выделяет эти сведения, затем копирует их в буфер обмена и вставляет в документ. Пока компьютер продолжает работать, пользователь не выходит из системы и ничего не добавляется в буфер обмена, чтобы заменить имеющиеся там данные, эти данные остаются в буфере обмена.

Попробуйте когда-нибудь подойти к своему компьютеру, откройте блокнот или документ Word и используйте сочетание клавиш **Control + V**, чтобы вставить в документ данные, которые в данный момент находится в буфере обмена. Попробуйте сделать это на других компьютерах. Возможно, вы будете удивлены тому, что увидите. Как часто вы обнаружите URL-адреса, фрагменты переписки из приложений для обмена мгновенными сообщениями или целые части текста, все еще доступные в буфере обмена? Буфер обмена не виден в системе, но он там есть, и в этом состоит проблема – настолько серьезная, что существует статья из базы знаний Microsoft, которая называется «How to Prevent Web Sites from Obtaining Access to the Contents of Your Windows Clipboard»

(<http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q224993&>), касающаяся браузера Internet Explorer версий 4–6.

Данные, находящиеся в буфере обмена, могут быть полезны в различных случаях, например, при расследовании кражи информации или интеллектуальной собственности, мошенничества или домогательства. Иногда такая информация может предоставить вам зацепки по делу, а иногда в буфере обмена можно найти изображения или целые части документов.

Файл «rclip.exe» (доступный на сайте <http://unxutils.sourceforge.net>) – это инструмент командной строки, который можно использовать для получения содержимого буфера обмена. Такие инструменты командной строки, как «rclip.exe», позволяют легко автоматизировать сбор информации посредством пакетных файлов или скриптов.

Инструменты и ловушки...

Содержимое буфера обмена

Язык Perl предоставляет простой интерфейс к API для получения доступа к содержимому буфера обмена. Следующий скрипт выводит содержимое буфера обмена в виде строки:

```
use strict;
use Win32::Clipboard;
print "Clipboard contents = ".Win32::Clipboard()->Get()."\\n";
```

Чтобы воспользоваться расширенными возможностями модуля Win32::Clipboard, ознакомьтесь с документацией для этого модуля.

Информация о службах/драйверах

Службы и драйверы запускаются автоматически при запуске системы в соответствии с записями в реестре. Большинство пользователей даже не видят, что эти службы выполняются как процессы в системе, потому что на самом деле не существует явных признаков этого, как в случае с процессами (например, можно увидеть выполняющие процессы в окне диспетчера задач). Тем не менее, эти службы выполняются. Вовсе не все службы устанавливаются пользователем или системным администратором. Некоторые вредоносные программы устанавливают себя в качестве службы или даже в качестве системного драйвера.

Инструменты и ловушки...

Информация о службах

Perl-скрипт «svc.pl», расположенный в каталоге \ch2\code на носителе, который идет в комплекте с этой книгой, использует инструментарий WMI (получая доступ к классу Win32_Service), чтобы найти информацию о службах в локальной или удаленной системе. Файл «svc.exe» – отдельный исполняемый файл Windows, скомпилированный из этого Perl-скрипта с помощью программы Perl2Exe.

Как Perl-скрипт, так и исполняемый файл, показывают следующую информацию о службах:

- имя службы;
- DisplayName для службы;
- StartName (контекст, используемый для запуска службы);
- строка описания для службы;

- идентификатор процесса для службы (эту информацию можно использовать, чтобы сопоставить службу с информацией о процессе);
- путь к исполняемому образу для службы;
- режим запуска для службы;
- текущее состояние службы;
- состояние службы;
- тип службы (драйвер ядра, разделяемый процесс и т. д.);
- идентификатор тега – уникальное значение, используемое, чтобы упорядочить запуск служб в группе служб, назначенных для загрузки.

На илл. 1.11., показан пример информации, отображаемой этим инструментом.

Как Perl-скрипт, так и исполняемый файл можно изменить, чтобы выводить эту информацию в разных форматах, в том числе в формате CSV, чтобы упростить анализ этой информации путем преобразования выходных данных в формат, удобный для открытия в электронной таблице.

```
Name      : UMWdf
Display   : Windows User Mode Driver Framework
Start     : NT AUTHORITY\LocalService
Desc      : Enables Windows user mode drivers.
PID       : 1660
Path      : C:\WINDOWS\system32\wdfmgr.exe
Mode      : Auto
State     : Running
Status    : OK
Type      : Own Process
TagID     : 0
```

Илл. 1.11. Фрагмент выходных данных инструмента «svc.exe» в ОС Windows XP.

История командной строки

Предположим, вы начали исследования системы и видите, что на экране есть одна или несколько командных строк. В зависимости от ситуации, во введенных пользователем командах, например, *ftp* или *ping*, могут быть скрыты ценные данные. Чтобы увидеть эти раннее введенные команды, нужно переместить полосу прокрутки командной строки вверх (если в командную строку было введено несколько команд, область отображения увеличится над видимой частью окна командной строки), но такой возможности может и не быть. Если пользователь ввел команду *cls*, чтобы очистить экран, вы не сможете использовать полосу прокрутки, чтобы увидеть команды, введенные ранее. Вместо этого нужно использовать команды *doskey /history*, которая выведет историю команд, введенных в эту командную строку, как показано ниже:

```
D:\tools>doskey /history
move proc.exe d:\awl2\ch2\code
perl2exe -small d:\awl2\ch2\code\proc.pl
move proc.exe d:\awl2\ch2\code
y
cd \awl2\ch2\code
proc
cd \perl2exe
perl2exe -small d:\awl2\ch2\code\procmon.pl
```

```

procmon
move procmon.exe d:\awl2\ch2\code
cd d:\awl2\ch2\code
procmon
cd \tools
openports -fport
openports -netstat
cls
doskey /history
cd \tools
dir prom*
promqry
dir prom*
promqry

```

Я приведу вам пример того, когда я использовал эту команду. Я читал курс по расследованию инцидентов на Западном побережье, и во время перерыва на обед я «взломал» системы студентов. На некоторых компьютерах я специально открыл командную строку и ввел несколько команд, а затем ввел команду *cls*, чтобы очистить экран. Когда студенты вернулись, я заметил что один из них, в задней части аудитории, немедленно закрыл (не свернул, а закрыл) командную строку, которая была у него на экране. Как было задумано, «зацепки», которые я оставил в командной строке, предоставляли контекст к остальной части «взлома», что смогли обнаружить студенты, которые не закрыли свои командные строки. Однако я признаю, что у меня никогда не было возможности использовать эту команду вне учебной среды. Во всех случаях, когда я имел дело с работающей системой, пользователи не использовали командную строку. Однако это не означает, что такого не случится с вами.

Подключенные сетевые накопители

Во время экспертизы вы, возможно, захотите узнать, какие накопители или общие ресурсы сопоставлены с исследуемой системой. Такие сопоставления (подключения) могли быть созданы пользователем и могут быть признаком злого умысла (в том случае, если пользователь узнал пароль администратора и получает доступ к системам на предприятии). Кроме того, в файловой системе или реестре могут отсутствовать долговременные данные об этих подключениях к общим ресурсам в других системах, хотя энергозависимую информацию о подключении сетевых накопителей можно сопоставить с информацией о сетевых соединениях, которые вы уже получили.

На илл. 1.12., показаны выходные данные программы «di.exe» (*di* означает *drive info* – информация о накопителе), которую можно найти на сопроводительном носителе.

Drive	Type	File System	Path	Free Space
C:\	Fixed	NTFS		1.15 GB
D:\	Fixed	NTFS		8.18 GB
E:\	Fixed	NTFS		5.19 GB
F:\	CD-ROM			0.00
G:\	Removable	FAT32		974.45 MB
Z:\	Network	NTFS	\\\192.168.1.71\c\$	2.96 GB

Илл. 1.12. Выходные данные программы «di.exe».

Выходные данные программы «di.exe», показанные на илл. 1.12., являются результатом запуска этой программы в ОС Windows XP Home с одним накопителем, подключенным к небольшому серверу Windows 2003, а именно к общему ресурсу *C\$* на этом сервере.

Обратите внимание, что в выходных данных «di.exe» также показан съемный накопитель, которому назначен символ G:\. Это USB флеш накопитель, артефакты которого мы будем рассматривать в главе 4.

Общие ресурсы

Помимо ресурсов, используемых исследуемой системой, вы также захотите получить информацию о ресурсах, которые предоставляет эта система. Информация об общих ресурсах, доступных в системе, хранится в разделе реестра HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\lanmanserver\Shares, но ее также можно получить из работающей системы, используя такой инструмент командной строки, как «share.exe», который имеется на сопроводительном носителе. (Исходный код на языке Perl также доступен.)

Фрагмент выходных данных инструмента «share.exe»:

Name	-> SharedDocs
Type	-> Disk Drive
Path	-> C:\DOCUMENTS AND SETTINGS\ALL USERS\DOCUMENTS
Status	-> OK

Совет

В этой главе мы обсуждали использование инструментария WMI посредством языка Perl. Я показал вам несколько примеров и предоставил код на накопителе, который идет в комплекте с этой книгой. Microsoft также предоставляет доступ к инструментарию WMI посредством встроенного инструмента командной строки «wmic.exe» (<http://support.microsoft.com/kb/290216/en-us>). Инструмент «wmic.exe» может быть чрезвычайно полезен для сбора разнообразной информации из удаленных систем на предприятии. В марте 2006 года Эд Скудис (Ed Skoudis) опубликовал статью в блоге SANS Handler's Diary (<http://isc.sans.org/diary.html?storyid=1229>), в которой предоставлен ряд очень эффективных и полезных опций командной строки для использования в «wmic.exe».

Энергонезависимые данные

Во время исследования работающей системы вы, возможно, не захотите ограничиваться сбором только энергозависимой информации. В зависимости от ситуации, эксперту может потребоваться собрать информацию, которая обычно не исчезает даже при перезагрузке системы, например, содержимое разделов реестра или файлов. Эксперт может решить, что нужно извлечь информацию из реестра или, что нужно собрать информацию о файлах или из файлов либо для дополнительного анализа, либо потому, что злоумышленник все еще может находиться в системе и совершать в ней определенные действия. В таких случаях эксперт может решить, что для того, чтобы отследить злоумышленника (или ботнет), ему нужно оставить систему работающей и подключенной к сети, а также сохранить определенную информацию до того, как она будет изменена или удалена.

После того как система будет запущена, в ней могут произойти изменения, например, ее накопители могут быть подключены к другим системам сети и наоборот, могут быть запущены службы или установлены приложения. Эти изменения могут не сохраниться после перезагрузки, и поэтому эксперту, возможно, нужно будет задокументировать информацию о них.

Параметры реестра

Некоторые параметры реестра могут повлиять на последующие этапы судебного анализа и экспертизы. Хотя эти параметры сами по себе являются энергонезависимыми,

они могут повлиять на то, как вы будете продолжать проводить экспертизу или на то, будете ли вы продолжать экспертизу вообще.

Существует несколько способов для сбора информации из реестра. Мой любимый (если вы еще не догадались) – написать Perl-скрипт, который предоставляет различные функциональные возможности для получения отдельных параметров или всех параметров и подразделов отдельного раздела. Инструмент командной строки «reg.exe», входящий в состав средств поддержки Windows 2000 и встроенный в ОС Windows XP и 2003, предназначен для получения доступа к реестру и управления реестром.

ClearPageFileAtShutdown

Этот параметр реестра указывает операционной системе очистить файл подкачки после завершения работы. Так как ОС Windows использует архитектуру виртуальной памяти, некоторая память, используемая процессами, выгружается в файл подкачки. Когда система выключается, информация в файле подкачки остается на накопителе и может содержать такую информацию, как зашифрованные пароли, фрагменты переписки из приложений для обмена мгновенными сообщениями и другие строки и биты информации, которые могут предоставить вам важные данные для расследования. Хотя большинство экспертов понимают, что файл подкачки – это неструктурный блок данных, в основном без контекста (то есть обнаружение интересующей вас строки в файле подкачки не предоставит вам важной информации, например, о процессе, с которым связана эта строка), благодаря достижениям в области анализа памяти Windows (подробную информацию по этой теме см. в главе 3) существуют способы расширить информацию, доступную в дампе физической памяти Windows, путем анализа файла подкачки. Если файл подкачки будет очищен во время завершения работы, эту потенциально ценную информацию будет намного труднее, или вовсе невозможно, получить.

Microsoft предоставляет статьи из базы знаний об этом параметре реестра, которые применимы как к ОС Windows 2000 (<http://support.microsoft.com/kb/182086/EN-US>), так и к ОС Windows XP (<http://support.Microsoft.com/kb/314834/EN-US>).

DisableLastAccess

В файловых системах, используемых с ОС Windows, есть возможность отключать обновление времени последнего доступа к файлам. Согласно документации Microsoft, эта возможность предназначалась для увеличения быстродействия, особенно в высокопроизводительных файловых серверах. В обычных рабочих станциях, а также в персональных компьютерах и ноутбуках, которые используют большинство пользователей (домашние компьютеры, ПК служащих и т. д.) этот параметр не обеспечивает какого-то заметного увеличения быстродействия. Чтобы активировать эту возможность в ОС Windows 2003, следующий параметр следует установить в значении 1:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate
```

Согласно руководству по настройке производительности для ОС Windows 2003, этот параметр не существует по умолчанию и его нужно создать.

Совет

В статье № 555041 из базы знаний Microsoft (<http://support.microsoft.com/kb/555041>) этот параметр называется *DisableLastAccess*, но в статье № 849372 (<http://support.microsoft.com/kb/894372>) этот параметр называется *NtfsDisableLastAccessUpdate*. В статье № 150355 из базы знаний Microsoft этот параметр называется *NtfsDisableLastAccess* в ОС Windows NT версий 3.51 и 4.0.

В ОС Windows XP и 2003 можно выполнить запрос об этом параметре или включить этот параметр используя команду *fsutil*. Например, чтобы выполнить запрос об этом параметре, используйте такую команду:

```
C:\>fsutil behavior query disablelastaccess
```

Если этот параметр реестра был установлен, особенно до того, как вы начали исследование системы, вероятно вы не найдете полезной информации, касающейся времени последнего доступа к файлам. Это значит, что вам нужно будет использовать другие способы анализа, например те, что описаны в главе 4.

Предупреждение
Функциональная возможность <i>NtfsDisableLastAccessUpdate</i> по умолчанию <i>включена</i> в ОС Vista. Не забывайте об этом при проведении расследования инцидентов или компьютерно-технических экспертиз. На момент написания этой книги документация для судебных экспертов по этой проблеме все еще находилась в процессе составления.

Autoruns

Несколько областей в реестре (и в файловой системе) называются *местами автозапуска*, потому что они предоставляют возможность автоматически запускать приложения, обычно без прямого взаимодействия с пользователем. Некоторые из этих мест автоматически запускают приложения во время загрузки системы, другие – во время входа пользователя в систему, а третьи – в то время, когда пользователь выполняет отдельное действие. В случаях, когда приложение запускается при выполнении пользователем определенного действия, пользователь не будет знать, что он запускает другое приложение.

Поиск таких мест в реестре может показаться чрезвычайно трудной или даже невыполнимой задачей, но не спешите огорчаться – для автоматического запуска может использоваться ограниченное количество мест. Это количество может быть большим, но оно ограничено. Вместо того чтобы перечислять эти места здесь, я собираюсь более подробно рассмотреть тему анализа реестра позднее в этой книге. Однако если вы решили, что вам нужно собрать эту информацию в ходе расследования инцидента, для этого существует два способа. Первый способ – использовать такой инструмент как «reg.exe» (упоминавшийся ранее) для сбора данных из отдельных разделов и параметров. Второй способ – использовать инструмент Autoruns (<http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx>), который сделает все вместо вас. Создатели программы Autoruns (Марк Руссинович (Mark Russinovich) и Брайс Когсвелл (Bryce Cogswell), теперь сотрудники Microsoft) постоянно обновляют список областей, которые проверяет этот инструмент. В некоторых случаях я замечал, что новые области добавлялись в инструмент до того, как они начинали широко использоваться вредоносными программами. Autoruns выпускается как с графическим интерфейсом, так и с интерфейсом командной строки, и обе версии имеют одинаковые функциональные возможности. Например, можно использовать переключатель *-t* в версии с интерфейсом командной строки, чтобы не показывать записи об элементах с цифровой подписью Microsoft (записи об исполняемых файлах, которые были подписаны поставщиком) или переключатель *-v*, чтобы проверить цифровые подписи.

Autoruns также отлично проверяет области автозапуска в файловой системе, например каталог «Назначенные задания» (“Scheduled Tasks”). Иногда администраторы используют назначенные задания, чтобы предоставить себе более высокие привилегии (то есть системные привилегии) для выполнения таких задач, как просмотр областей реестра,

доступ к которым обычно запрещен даже администраторам. Злоумышленник, который получает доступ к системе на уровне администратора, может сделать что-нибудь похожее, чтобы еще больше расширить свое присутствие в системе.

Еще одна область в реестре, которая может предоставить ценную информацию во время расследования, – это защищенное хранилище (см. вставку «Задокументированное хранилище»). Информация, находящаяся в защищенном хранилище, хранится в зашифрованном формате в реестре. Если вы создадите образ системы, то такие инструменты, как Forensic ToolKit от компании AccessData, расшифруют и восстановят эту информацию. Однако иногда проще собрать эту информацию во время исследования работающей системы, особенно если время играет существенную роль и эта информация имеет отношение к делу.

Записки из подполья...

Задокументированное хранилище

Задокументированное хранилище – область памяти, где хранится конфиденциальная информация пользователя. Когда система выключается, эта информация хранится в зашифрованном формате в реестре, а когда пользователь входит в систему, эта информация помещается в память. ОС Windows помещает в хранилище такую информацию для последующего использования, как пароли и данные автоматического заполнения для веб-форм.

Содержимое задокументированного хранилища можно просмотреть, используя такие инструменты, как «pstorerewiew.exe» (www.ntsecurity.nu/toolbox/pstorerewiew) или Protected Storage Explorer (www.codeproject.com/KB/cpp/psexplorer.aspx).

Информация в задокументированном хранилище может быть полезна в случаях, связанных с доступом к веб-сайтам и использованием паролей для таких сервисов, как Hotmail и MSN.

Информация в задокументированном хранилище может быть также полезна для злоумышленников. Я видел несколько систем, зараженных IRC-ботами, которые по команде отправляли злоумышленнику информацию из задокументированного хранилища. (IRC-бот – вредоносная программа, которая после установки подключается к IRC-каналу, ожидая команд; оператор канала может подать одну команду, которая затем может быть выполнена тысячами ботов.) Девятнадцатого февраля 2006 года Брайан Кребс (Brian Krebs) опубликовал статью (www.washingtonpost.com/wp-dyn/content/article/2006/02/14/AR2006021401342.html) в журнале *Washington Post Magazine* о хакере, написавшем программное обеспечение для ботов и контролировавшем тысячи систем. В этой статье говорилось, что хакер мог ввести одну команду (*pstore*) и получить из всех зараженных систем информацию, хранящуюся в задокументированном хранилище, в частности имена пользователей и пароли для учетных записей PayPal, eBay, Bank of America и Citibank, а также для учетных записей электронной почты военных и правительственные организаций.

Информация, сохраняемая службой задокументированного хранилища, доступна посредством функции «Автозаполнение» (“AutoComplete”), встроенной в браузер Internet Explorer. Чтобы получить доступ к окну «Настройка автозаполнения» (“AutoComplete Settings”), показанному на илл. 1.13., щелкните по пункту «Сервис» (“Tools”) в строке меню Internet Explorer, выберите пункт «Свойства обозревателя» (“Internet Options”), затем вкладку «Содержание» (“Content”) и нажмите кнопку «Параметры» (“AutoComplete”).

Пользователи этих зараженных систем использовали браузер Internet Explorer с включенной функцией автозаполнения, чтобы получить доступ к своим учетным записям онлайн-магазинов и банков, делая их доступными для таких злоумышленников, как хакер из статьи Брайана.



Илл. 1.13. Диалоговое окно «Настройка автозаполнения» в Internet Explorer версии 6.0.

Такие инструменты, как PassView (www.nirsoft.net/utils/pspv.html) и Protected Storage Explorer (www.forensicideas.com/tools.html) позволяют просмотреть информацию из защищенного хранилища через удобный графический интерфейс, а «pstreview.exe» (www.ntsecurity.nu/toolbox/pstreview) – это инструмент командной строки, отправляющий ту же информацию на стандартное устройство вывода. Возможно, вам нужно будет собрать эту информацию во время исследования системы, особенно если расследуемый вами инцидент связан с пользователями, которые получали доступ к веб-сайтам, требующим ввода паролей. Можно извлечь эту информацию из образа данных, но для этого потребуются специальные дорогостоящие инструменты, предназначенные для дешифрования данных. Возможно, вы столкнетесь с ситуациями (например, связанными с кражей данных, лицами, пропавшими без вести и т. д.), когда вам нужно будет быстро собрать информацию из системы, а не ждать пока образ системы будет создан и отправлен в лабораторию для извлечения и анализа данных.

Журналы регистрации событий

Журналы регистрации событий – это, по существу, файлы в файловой системе, но они могут изменяться. На самом деле, в зависимости от того, как они сконфигурированы, и аудит каких событий выполняется, они могут изменяться довольно быстро.

В зависимости от того, как настроены политика аудита в целевой системе и как вы получаете доступ к этой системе во время расследования инцидента, в журнале регистрации событий могут создаваться записи. Например, если вы решили выполнить команды в системе из удаленного места (то есть система находится в другом здании или другом городе, и вы не можете добраться к ней быстро, но хотите сохранить некоторые ее данные), а политика аудита настроена надлежащим образом, то журнал событий безопасности будет содержать записи о каждом вашем входе в систему. Если будет создано достаточное количество таких записей, можно, в конце концов, потерять ценную информацию, которая относится к расследованию. Чтобы получить эти записи о событиях, можно использовать такие инструменты, как «psloglist.exe» и «dumpEvt.exe», или скопировать из системы сами файлы .evt (это зависит от уровня доступа и прав используемой учетной записи). Подробная информация об анализе журналов событий ОС Windows будет предоставлена в главе 5.

Сейчас вы, возможно, думаете: «Ладно, допустим у меня есть все эти инструменты и программы, и мне нужно расследовать инцидент. Какие данные мне нужно собрать?». Дежурный ответ: «Все зависит от обстоятельств». Я знаю, это, вероятно, не тот ответ, который вы хотели услышать, но я попробую объяснить вам, почему этот ответ единственno *верный*.

Энергозависимые данные, которые наиболее полезны для экспертизы, зависят от типа расследуемого инцидента. Например, если инцидент связан с удаленным вторжением или троянской программой скрытого удаленного администрирования, то информация о процессах, сетевых соединениях и сопоставлении процесса с портом (возможно, даже содержимое определенных разделов реестра) будет наиболее ценная для эксперта. Однако, если сотрудник компании подозревается в краже данных, являющихся собственностью компании, или нарушении корпоративной политики допустимого использования, такая информация, как сведения о накопителях, подключенных к системе, история просмотра веб-страниц, содержимое буфера обмена, будет более полезной для экспертизы.

Самое главное – знать, какая информация доступна для вашей экспертизы, как можно ее получить и использовать. Когда вы начнете рассматривать различные типы инцидентов и информацию, необходимую для их расследования, вы увидите частичное совпадение между различными используемыми инструментами и данными, представляющими интерес для расследования. Хотя, возможно, вы не разработаете универсальный файл, запускающий все команды, которые вы хотите использовать для каждого расследования, вы поймете, что наличие нескольких меньших пакетных файлов (или файлов настроек для программы Forensic Server Project, которая описана позднее в этой главе) является лучшим подходом. Таким образом, вы можете собрать только ту информацию, которая нужна для каждой ситуации.

Устройства и другая информация

Возможно, вы решите собрать из системы другие типы данных, которые не являются энергозависимыми по своей природе, но вы хотите зарегистрировать их для документации. Например, возможно, вы захотите узнать какие-нибудь сведения о НЖМД, установленном в компьютере. Perl-скрипт «di.pl» реализовывает WMI, чтобы перечислить различные накопители, подключенные к системе, а также показать сведения о разделах. Скрипт «ldi.pl» реализовывает WMI, чтобы собрать информацию о логических накопителях (C:\, D:\ и т. д.), в том числе о локальных встроенных накопителях, съемных накопителях и удаленных общих ресурсах. Скрипт «sr.pl» перечисляет информацию о точках восстановления системы в ОС Windows XP (дополнительную информацию о точках восстановления системы см. в главах 4 и 5).

Инструмент DevCon, доступный на сайте Microsoft, можно использовать для документирования информации об устройствах, подключенных к системе Windows. Являясь аналогом диспетчера устройств, но с интерфейсом командной строки, DevCon может показывать имеющиеся классы устройств, а также состояние подключенных устройств.

Несколько слов о выборе инструментов

В этой главе и в других главах этой книги упоминается несколько инструментов, которые можно использовать для выполнения определенных задач. Эта книга не предназначена для того, чтобы служить исчерпывающим перечнем инструментов; это просто невозможно. Вместо этого я попытаюсь объяснить вам, где нужно искать данные, и показать вам несколько способов, используя которые можно собрать данные, необходимые для экспертизы. Иногда самое главное – просто знать, что эти данные есть в системе.

При сборе информации из работающих систем нам чаще всего придется взаимодействовать с самой операционной системой, используя имеющийся интерфейс API. Разные инструменты могут использовать разные вызовы API для сбора одинаковой информации.

Всегда рекомендуется знать, как ваши инструменты собирают информацию. Какие вызовы API использует исполняемый файл? К каким DLL-файлам он обращается? Как инструмент отображает выходные данные, и как эти данные отличаются от данных других подобных инструментов?

Тестируйте свои инструменты, чтобы определить, какое воздействие они оказывают на работающую систему. Оставляют ли они какие-нибудь артефакты в системе? Если да, то какие? Не забывайте документировать информацию об этих артефактах, так как это позволит вам определить (и задокументировать) меры, которые вы предпринимаете, чтобы уменьшить воздействие от использования, и обосновать применение, этих инструментов. Например, в ОС Windows XP осуществляется упреждающая выборка для приложений, то есть при запуске приложения некоторая информация о нем (например, кодовые страницы) сохраняется в файле с расширением .pf, который находится в каталоге %WINDIR%\Prefetch. В этом каталоге может храниться не больше 128 pf-файлов. Если вы проводите расследование инцидента, и в этом каталоге находится меньше 128 pf-файлов, одно из действий, которое запускаемые инструменты окажут на систему, будет заключаться в том, что pf-файлы для этих инструментов будут добавлены в каталог «Prefetch». В большинстве случаев это, возможно, не будет проблемой. Однако предположим, что ваша методика исследования включает в себя использование инструмента «nc.exe» (netcat). Если кто-то уже использовал «nc.exe» в системе, то в результате запуска любого файла с таким именем будет перезаписан существующий pf-файл для «nc.exe», что в свою очередь, возможно, приведет к уничтожению улик (например, изменение отметок времени MAC файла или таких данных в файле, как путь к исполняемому образу).

Проведение тестирования и проверки собственных инструментов может показаться тяжелой задачей. Ведь кто захочет выполнять процесс тестирования для каждого инструмента? Что ж, возможно, это придется делать вам, так как мало сайтов разработчиков предоставляют такую информацию о своих инструментах; большинство инструментов изначально не были предназначены для использования при проведении расследования инцидентов или судебных компьютерных экспертиз. Но после того как вы настроите свою платформу (инструменты, процесс и т. д.) для тестирования, вы сможете без труда документировать сведения об используемых инструментах и тестировать эти инструменты. Документирование сведений об инструментах и тестирование инструментов напоминает тестирование или анализ потенциально вредоносных программ (эта тема будет подробно рассмотрена в главе 6).

К основным этапам документирования относится статическое и динамическое тестирование. Статическое тестирование подразумевает документирование уникальной идентифицирующей информации об инструменте, например:

- место, откуда был получен инструмент (URL-адрес);
- размер файла;
- криптографические хэш-значения для файла, вычисленные с использованием известных алгоритмов;
- такая информация из файла, как заголовки PE-файла, сведения о версии файла, таблицы импорта/экспорта и т. д.

Эту информацию легко получить, используя инструменты командной строки и скриптовые языки, например, Perl, а весь процесс сбора (а также архивирования информации в базе данных, электронной таблице или плоском файле) несложно автоматизировать.

Инструменты и ловушки...

Встроенные инструменты

Большинство экспертов, с которыми я общаюсь, неохотно используют собственные инструменты систем Windows, особенно те инструменты, которые постоянно находятся в самих системах, аргументируя это тем, что, если безопасность системы нарушена достаточно серьезно, то выходным данным этих инструментов доверять нельзя. Это отличная точка зрения, но вы можете использовать ее с выгодой для себя в своем анализе.

Специалисты по расследованию инцидентов, как правило, предпочитают запускать свои инструменты с CD- или DVD-дисков, которые являются неизменными носителями, то есть если в исследуемой системе есть вирус, он не сможет заразить ваши инструменты. Другой подход, заимствованный из мира Linux, заключается в использовании статических исполняемых файлов, то есть исполняемых файлов, не зависящих от каких-либо библиотек (в мире Windows – это DLL-файлы) зараженной системы. Это довольно сложно осуществить в отношении PE-файлов Windows, однако существуют способы, используя которые можно имитировать этот эффект. Один из этих способов включает в себя получение доступа к таблице импорта PE-файла и изменение имени DLL-файлов, к которым выполняется обращение, изменение имени этих DLL-файлов в файловой системе, а затем копирование этих файлов на компакт-диск. Проблема такого подхода состоит в том, что неизвестно, насколько далеко нужно углубляться в рекурсивную структуру DLL-файлов. Если вы посмотрите на заголовок таблицы импорта исполняемого файла (в главе 6 структура PE-файлов рассматривается подробно), вы, вероятно, увидите несколько ссылок на DLL-файлы и их функции. Если вы перейдете к этому DLL-файлу, то увидите экспортные функции в таблице экспорта, но вы, вероятно, также увидите, что этот DLL-файл импортирует функции из другого DLL-файла. Как видите, отслеживание всех этих функций только для того, чтобы инструмент, который вы хотите использовать, не получал доступа к библиотекам в зараженной системе, возможно, быстро станет трудно контролировать, и вы рискуете на самом деле нанести вред системе, которую хотите спасти.

Еще один вариант – использовать выбранные вами инструменты, которые будут обращаться к встроенным DLL-файлам, чтобы предоставить необходимые функциональные возможности посредством их экспортных API. Таким образом, можно выполнить часть дифференциального анализа, то есть использовать два различных способа, чтобы просмотреть одну и ту же информацию. Например, один из способов определить открытые порты в системе – выполнить сканирование портов этой системы. Один из способов узнать, какие порты используются и какой сетевой трафик исходит из системы, – сабрать сетевые пакеты. Ни один из этих способов не зависит от операционной системы или кода самой системы. Чтобы выполнить дифференциальный анализ, нужно сравнить выходные данные команды `netstat -ano` либо с результатами сканирования портов, либо с информацией из перехваченных сетевых пакетов. Если в результате применения любого из этих способов появляется информация, которая отсутствует или не показана в выходных данных инструмента «`netstat.exe`», то, возможно, вы имеете дело со случаем, когда нарушена работа исполняемых файлов или самой операционной системы. В главе 7 предоставляется еще один пример того, как использовать дифференциальный анализ, чтобы определить, работает ли в системе руткит режима пользователя.

Динамическое тестирование представляет собой запуск инструментов во время использования программ наблюдения для документирования изменений, происходящих в системе. Инструменты создания снимков для сравнения, такие как InControl5, чрезвычайно полезны для этой задачи, как и инструменты наблюдения, такие как RegMon и FileMon, оба из которых были первоначально доступны на сайте Sysinternals.com, но сегодня предоставляются на веб-сайте Microsoft. RegMon и FileMon позволяют вам увидеть не только разделы реестра и файлы, к которым обращается процесс, но и разделы

реестра и файлы, которые создаются или изменяются. Возможно, вы также решите использовать такие инструменты, как Wireshark, чтобы наблюдать за входящим и исходящим трафиком системы во время тестирования инструментов, особенно если статический анализ инструмента показывает, что тот импортирует функции, предоставляющие возможность работы в сети, из других DLL-файлов.

Методики исследования работающей системы

При проведении исследования работающей системы, фактическая методика или процедура, используемая для получения данных из системы, может изменяться в зависимости от ряда факторов. Работая консультантом и специалистом по расследованию инцидентов, я обнаружил, что, прежде всего, нужно полностью понимать, какие инструменты доступны и какие из них можно включить в свой набор инструментов (учитывая проблемы, связанные с покупкой и лицензированием программного обеспечения, а также другие платы и ограничения), а затем решить, какие из них работают, в зависимости от ситуации.

Существует два основных способа для проведения исследования работающей системы Windows: локально и удаленно.

Методика локального исследования

Проведение исследования работающей системы локально подразумевает, что вы сидите за консолью системы, вводите команды на клавиатуре и сохраняете информацию локально: либо непосредственно на НЖМД, либо на съемный (флеш накопитель, внешний USB-накопитель) или сетевой (общий) ресурс, который отображается как локальный ресурс. Эта методика очень часто применяется в ситуациях, когда специалист имеет непосредственный физический доступ к системе и своим инструментам на компакт-диске или флеш накопителе. Сбор информации локально из нескольких систем можно часто провести намного быстрее, чем обнаружить сетевое соединение или получить доступ к беспроводной сети. Имея достаточное количество места на внешних накопителях и достаточный уровень доступа, специалист по расследованию инцидентов может быстро и эффективно собрать нужную информацию. Чтобы еще больше оптимизировать свои действия, специалист может записать все свои инструменты на компакт-диск и управлять ими через пакетный файл или скрипт, позволяющий ограниченную гибкость (например, накопитель, подключенный по USB, сопоставлен с другим символом накопителя, ОС Windows установлена на диске D:\ и т. д.).

Самый простой способ реализовать методику локального исследования – использовать пакетный файл. Я предпочитаю применять пакетные файлы и Perl-скрипты, потому что, вместо того чтобы без конца вводить команды (и без конца делать ошибки), я могу записать команды один раз и затем выполнять их автоматически. Пример простого пакетного файла, который можно использовать во время исследования работающей системы, выглядит следующим образом:

```
tlist.exe -c > %1\tlist-c.log
tlist.exe -t > %1\tlist-t.log
tlist.exe -s > %1\tlist-s.log
tcpvcon.exe -can > %1\tpcvcon-can.log
netstat.exe -ano > %1\netstat-an0.log
```

Вот так, три инструмента и пять простых команд. Сохраните этот файл как «local.bat» и запишите его на компакт-диск вместе с копиями соответствующих инструментов. Возможно, вы также захотите добавить на компакт-диск доверенные копии интерпретатора командной строки (cmd.exe) для каждой операционной системы. Прежде чем запустить пакетный файл, посмотрите, какие сетевые накопители доступны в системе,

или подключите USB флеш-накопитель к компьютеру и посмотрите, какой символ накопителя будет ему назначен (например, F:\), затем запустите пакетный файл следующим образом (D:\ – это дисковод CD-ROM):

```
D:\>local.bat F:
```

После того как выполнение пакетного файла завершится, у вас будет пять файлов на флеш-накопителе. Конечно, вы можете добавлять в пакетный файл разнообразные команды в зависимости от того, какой объем данных вы хотите получить из системы.

Разработано несколько бесплатных наборов инструментов для исследования систем локально; среди них – Incident Response Collection Report (IRCR; версия 2.3 на момент написания этой книги была доступна по адресу <http://tools.phantombyte.com/>) и Windows Forensic Toolchest (WFT, доступный по адресу www.foolmoon.net/security/wft/index.html и созданный Монти Макдугалом (Monty McDougal)). Хотя они имеют разные способы реализации и форматы выходных данных, основная функциональность обоих наборов инструментов в основном одинаковая: запустите внешние исполняемые файлы, управляемые пакетным файлом Windows, и сохраните выходные данные локально. WFT хорошо сохраняет необработанные данные и позволяет специалисту сохранять выходные данные команд в виде HTML-отчетов.

Еще один подход к разработке методики локального исследования состоит в объединении как можно большего количества возможностей в одном приложении, используя интерфейс Windows API. Попытка реализовать такой подход была выполнена в инструменте Nigilant32 от компании Agile Risk Management LLC (www.agilerm.net; возможности этого инструмента были включены в состав программы F-Response, доступной на сайте www.f-response.com). Nigilant32 использует те же вызовы Windows API, что и внешние утилиты, для сбора информации из системы (см. илл. 1.14.) и имеет дополнительные возможности выполнения проверок файловой системы и создания дампа содержимого физической памяти (ОЗУ).



Илл. 1.14. Графический интерфейс программы Nigilant32.

Любопытный момент, касающийся наборов инструментов, запускаемых при помощи пакетных файлов, состоит в том, что их используют многие специалисты. Работая на объекте клиента или участвуя в конференции, я часто общаюсь со специалистами, которым интересно сравнивать свои подходы с подходами других экспертов. В свои наборы некоторые включают инструменты, перечисленные в главе 5 моей первой книги, *Windows Forensics and Incident Recovery*, или другие инструменты, о которых они узнали из других источников. Если разобраться, то эти наборы инструментов, как ни странно, во многом совпадают. Наборы инструментов, запускаемые при помощи пакетных файлов, применяют исполняемые файлы, которые используют те же (или похожие) вызовы Windows API, что и инструменты типа Nigilant32.

Многие рассмотренные здесь инструменты (WFT, Nigilant32 и даже многие инструменты с интерфейсом командной строки) также предоставляются в составе дистрибутива Helix, собранного Дрю Фэйхи (Drew Fahey) и доступного на веб-сайте e-Fense (www.e-fense.com/helix/). В состав Helix входят программы загрузочной версии Linux, а также программы для исследования работающей ОС Windows, и многие считают этот дистрибутив чрезвычайно полезным.

Методика удаленного исследования

Методика удаленного исследования, как правило, представляет собой выполнение ряда команд по отношению к какой-нибудь системе по сети. Эта методика очень полезна в ситуации с несколькими системами, так как процесс входа в систему и запуска команд легко автоматизировать. Среди специалистов такая возможность называется *расширяемость*. Некоторые инструменты отлично работают в сочетании с утилитой «psexec.exe» с сайта Sysinternals.com, а дополнительную информацию можно легко собрать посредством использования инструментария WMI. Независимо от применяемого подхода имейте в виду, что вам потребуются учетные данные для входа в каждую систему и что каждый раз при входе в систему для того, чтобы выполнить команду и собрать выходные данные, будет добавлена запись в журнал событий безопасности (при условии, что включен соответствующий уровень аудита). Учитывая эти факторы, мы видим, что порядок изменяемости энергозависимых данных немного сместится, поэтому я рекомендую в первую очередь использовать команду для сбора содержимого журнала событий безопасности.

В качестве основы для реализации этой методики можно использовать пакетный файл Windows. Используя три аргумента в командной строке (имя или IP-адрес системы и учетные данные для входа: имя пользователя и пароль), можно легко включить в скрипт ряд команд для сбора необходимой информации. Вам потребуется выполнить некоторые команды, используя инструмент «psexec.exe», который копирует исполняемый файл в удаленную систему, выполняет его и позволяет вам собрать выходные данные из стандартного устройства вывода или перенаправить выходные данные в файл так же, как если бы вы выполняли ту же команду локально. Для других команд в качестве аргументов используются UNC-путь (имя системы, перед которым добавляется две обратные косые черты (\\\)) и учетные данные для входа, поэтому вам не нужно будет применять «psexec.exe». Наконец, для сбора данных можно реализовать инструментарий WMI посредством языка VBScript или Perl. Microsoft предоставляет доступ к центру скриптов (www.microsoft.com/technet/scriptcenter/default.mspx) с различными примерами кода WMI, реализованного на различных языках, в том числе Perl, что делает создание пользовательского набора инструментов таким же простым, как операция копирования.

Пакетный файл, который мы использовали при локальном исследовании, довольно просто реализовывать для методики удаленного исследования:

```
psexec.exe \\%1 -u %2 -p %3 -c tlist.exe -c > tlist-c.log
psexec.exe \\%1 -u %2 -p %3 -c tlist.exe -t > tlist-t.log
psexec.exe \\%1 -u %2 -p %3 -c tlist.exe -s > tlist-s.log
psexec.exe \\%1 -u %2 -p %3 -c tcpvcon -can > tcpvcon-can.log
psexec.exe \\%1 -u %2 -p %3 c:\windows\system32\netstat.exe -ano >
%1\netstat-ano.log
```

Этот пакетный файл («remote.bat») располагается на компьютере эксперта и запускается следующим образом:

```
C:\forensics\case007>remote.bat 192.168.0.7 Administrator пароль
```

После того как выполнения пакетного файла завершится, эксперт получит на своем компьютере выходные данные команд в пяти файлах, готовых для исследования.

Если вы заинтересованы в использовании инструментария WMI для сбора информации удаленно, но плохо разбираетесь в языке программирования VBScript, возможно, вы захотите познакомиться с инструментом командной строки «wmic.exe», встроенным средством реализации для WMI. Эд Скудис (Ed Skoudis) написал для центра SANS Internet Storm Center отличное руководство для начинающих (<http://isc.sans.org/diary.php?storyid=1622>) по использованию «wmic.exe», которое включает в себя различные примеры использования этого инструмента, например, сбор списка установленных исправлений из удаленных систем. Используя «wmic.exe» можно посредством WMI сделать запрос относительно множества информации, доступной как класс Win32. Например, чтобы отобразить процессы, выполняющиеся локально в системе, можно использовать следующую команду:

```
C:\>wmic PROCESS GET ProcessId,Name,ExecutablePath
```

Это довольно простая команда, после выполнения которой можно увидеть выходные данные прямо в консоли. Можно также перенаправить выходные данные в файл и даже выбрать различные форматы, например, CSV (для открытия данных в программе Excel или их анализа при помощи Perl) или HTML-таблица. Используя такие переключатели, как */Node:*, */User:* и */Password:*, можно включить несколько команд инструмента «wmic.exe» в пакетный файл и собрать еще большее количество данных из удаленных систем. Кроме того, администраторы могут использовать эти команды, чтобы составить инвентарный список аппаратных средств и программного обеспечения, определить системы, в которых нужно установить исправления, и т. д. WMI – сам по себе мощный интерфейс к управляемым системам Windows, а инструмент «wmic.exe» предоставляет легкий доступ к командам автоматизации.

При использовании в коде правильной обработки ошибок и восстановления после ошибок, а также регистрации действий, это может стать эффективным и расширяемым способом для быстрого сбора информации из нескольких систем, которая управляется из центрального сервера и хранится там. ProDiscover Incident Response от компании Technology Pathways (www.techpathways.com) – это коммерческий инструмент, реализовывающий эту методику. Специалист по расследованию инцидентов может установить агент из центрального сервера, опросить агент относительно имеющейся информации, а затем удалить его. Благодаря интерфейсу ProScript API (программы ProDiscover) на основе языка Perl специалист может автоматизировать весь процесс. Этот подход сводит к минимуму количество записей о входе в систему, которые появятся в журнале событий безопасности, а также количество программ, которые нужно будет установить в удаленной системе. Программа ProDiscover Incident Response имеет дополнительные возможности для получения содержимого физической памяти (на момент написания этой книги – из ОС Windows 2000, XP и 2003, но не из систем Windows 2003 с пакетом обновления SP1 или более поздних версий), а также для клонирования данных НЖМД работающей системы по сети.

Также следует упомянуть инструмент F-Response (www.f-response.com), созданный и разработанный Мэттом Шенном (Matt Shannon). (Мэтт начал с разработки программы Nigilant32, доступной на сайте www.agilerm.net.) Хотя F-Response не является инструментом такого же типа, как ProDiscover Incident Response или любой другой инструмент, выполняющий исследование работающей системы посредством агента или получения удаленного доступа, он, несомненно, изменил привычное положение дел в области расследования инцидентов. Если коротко, F-Response предоставляет удаленный доступ, в режиме *только для чтения*, к НЖМД удаленной системы. Существует три

версии программы F-Response (Field Kit, Consultant и Enterprise), и все они работают в основном одинаково: после установки и настройки агента можно получить доступ к удаленному накопителю так же, как к локальному накопителю в режиме только для чтения на своем компьютере. Этот способ работает в локальной сети, между зданиями (которые находятся по разные стороны улицы или в разных частях города), при подключении к удаленному центру обработки данных – везде, где существует возможность подключения через TCP/IP. Агент F-Response можно запускать в ОС Linux, Mac OS X и Windows, а (начиная с версии программы 2.03) в ОС Windows физическая память системы будет отображаться в вашей локальной системе как виртуальный накопитель. По существу, F-Response предоставляет независимый от инструментов (можно использовать любой инструмент по своему желанию для создания образа удаленного накопителя) механизм для выполнения действий по расследованию инцидента. Хотя F-Response *не* позволяет, например, запустить «tlist.exe» и получить список процессов из удаленной системы, он предлагает средство для легкого клонирования содержимого ОЗУ (эта тема будет подробно рассмотрена в главе 3). На носителе, который идет в комплекте с этой книгой, содержится PDF-документ, подробно описывающий процесс удаленной установки F-Response версии Enterprise Edition (EE); оказывается, это скрытый способ развертывания F-Response EE. Данный документ также доступен на веб-сайте F-Response для зарегистрированных пользователей этой программы.

Недостаток методики удаленного исследования состоит в том, что специалист должен уметь получить доступ к системам и в некоторых случаях войти в эти системы через сеть. Если вход в систему, принятый в Windows (через NetBIOS), ограничен каким-либо образом (NetBIOS не установлен, брандмауэры/маршрутизаторы блокируют эти протоколы и т. д.), эта методика не будет работать.

Комбинированный подход (Использование FSP)

Я знаю, что говорил о том, что существует два подхода к методикам исследования системы, и это правда. Однако существует и третий подход, который на самом деле представляет собой просто сочетание методик локального и удаленного исследования, поэтому условно мы будем называть его *комбинированной методикой* (по правде говоря, я просто не смог придумать какое-нибудь экстравагантное имя). Эта методика чаще всего используется в ситуациях, когда специалист по расследованию инцидентов не может войти в эти системы удаленно, но хочет собрать всю информацию из нескольких систем и сохранить ее в центральном сервере. Специалист (или его помощник) отправляется к системе с компакт-диском или флеш-накопителем (желательно с активированным переключателем защиты от записи), получает доступ к системе и запускает инструменты, чтобы собрать информацию. В то время как инструменты будут выполняться, каждый из них будет отправлять свои выходные данные по сети в центральный «судебный сервер». Таким образом, удаленный вход в системы не выполняется, доверенные инструменты запускаются из неподдающегося изменению источника, и очень мало информации записывается на НЖМД целевой системы. При правильном подходе и планировании, специалист может свести к минимуму свое взаимодействие с системой, уменьшая количество решений, которые ему надо принять относительно входных команд и аргументов, а также уменьшая вероятность ошибки.

Предупреждение

Как вы знаете, согласно принципу обмена Локара, при взаимодействии объектов между ними будет происходить обмен веществами. Упоминания о запущенных командах появятся в реестре, а в ОС Windows XP системные файлы будут добавлены в каталог «Prefetch». Невозможно провести исследование работающей системы, не оставляя артефактов. Самое главное: понимать, как свести к минимуму количество этих

артефактов, и тщательно документировать свои действия во время исследования.

Возможно, самый простой способ реализовать комбинированную методику – использовать пакетный файл. Вы уже познакомились с различными инструментами и программами для сбора различной информации, имеющимися в вашем распоряжении. В большинстве рассмотренных примеров, как и в случае с локальной методикой, мы применяли инструменты с интерфейсом командной строки и перенаправляли их выходные данные в файл. Итак, как получить информацию, которую вы собрали, из системы? Один из способов – использовать инструмент netcat, часто описываемый как «швейцарский армейский нож для протокола TCP/IP» из-за большого количества операций, которые можно выполнить с его помощью. Сейчас мы не будем подробно изучать все возможности netcat, мы будем использовать этот инструмент для передачи информации из одной системы в другую. Сначала нам нужно настроить «прослушиватель» на нашем судебном сервере, и мы сделаем это, используя следующую команду:

```
D:\forensics>nc -L -p 80 > case007.txt
```

Эта командная строка указывает программе netcat («nc.exe») прослушивать порт 80, и любая информация, входящая в этот порт, отправляется (на самом деле перенаправляется) в файл с именем «case007.txt». Используя эту настройку, мы можем легко изменить наш пакетный файл, чтобы, вместо того чтобы записывать выходные данные команд в файл, отправлять их через netcat в «прослушиватель» на судебном сервере:

```
tlist.exe -c | nc %1 %2 -w 5
tlist.exe -t | nc %1 %2 -w 5
tlist.exe -s | nc %1 %2 -w 5
tcpvcon -can | nc %1 %2 -w 5
netstat.exe -ano | nc %1 %2 -w 5
```

Сохраните этот файл с именем «hybrid.bat», а затем запустите его из командной строки (D:\ – это дисковод CD-ROM):

```
D:\>remote.bat 192.168.1.10 80
```

Запустив этот пакетный файл, мы в целости и сохранности получим на судебном сервере данные из целевой системы для хранения и анализа.

Совет

Если вы предпочитаете не передавать такую информацию по сети «открытым текстом», на сайте SourceForge доступна версия netcat с шифрованием, которая называется cryptcat (<http://sourceforge.net/projects/cryptcat/>).

Несколько бесплатных инструментов реализовывают эту комбинированную методику. Один из них – это Forensic Server Project (FSP), выпущенный вместе с моей первой книгой (*Windows Forensics and Incident Recovery*, опубликованной издательством Addison-Wesley) в июле 2004 года и немного улучшенный к моменту выпуска первого издания моей второй книги, которая называется *Windows Forensic Analysis*. FSP – это бесплатный инструмент с открытым исходным кодом, написанный на языке Perl. Идея для FSP появилась в результате использования инструмента netcat, посредством которого эксперт запускал бы инструмент с компакт-диска, вставленного в дисковод целевой системы, а затем перенаправлял бы выходные данные команды через netcat. Вместо того

чтобы показывать выходные данные команды на экране (стандартное устройство вывода), netcat отвечал бы за отправку этой информации на работающий прослушиватель на сервере, где выходные данные команды сохранялись бы (а не записывались бы на накопителе целевой системы). Такой способ хорошо работал в некоторых ситуациях, но, по мере того как количество команд возрастало, а сами команды начали иметь разнообразные варианты аргументов, эта методика стала несколько громоздкой. Так как приходилось вводить большее количество команд, существовала большая вероятность возникновения ошибок, и иногда даже пакетный файл, предназначенный для автоматизации всех операций, не был решением проблемы. Поэтому я решил создать Forensic Server Project, платформу для (максимально возможной) автоматизации сбора и хранения данных, а также управления этими данными, во время исследования работающей системы.

FSP состоит из серверного и клиентского компонентов. Серверный компонент известен как FSP (на самом деле, мне в голову не пришло какое-нибудь другое остроумное название, а имя «Back Orifice» было уже занято). Вы копируете файлы для FSP на свою рабочую станцию (я использую ноутбук, работая на месте инцидента), и когда они будут выполняться, FSP будет работать, ожидая соединений. FSP выполняет простые задачи по управлению делами, регистрируя и сохраняя информацию для дел, над которыми вы работаете. Когда устанавливается соединение с клиентским компонентом, FSP будет реагировать соответствующим образом на команды, которые ему передаются. Клиентский компонент называется First Responder Utility (FRU). Компонент FRU запускается в целевой системе с компакт-диска или USB флеш накопителя. FRU – на самом деле очень простая платформа сама по себе в том, что для сбора информации с целевой системы она использует сторонние инструменты, например, те, что были рассмотрены в этой главе. Когда одна из команд выполняется, FRU перехватывает выходные данные команды (которые вы обычно видите на консоли, в окне командной строки) и отправляет их компоненту FSP, который сохраняет эту информацию и регистрирует это действие. FRU также может выполнять сбор отдельных параметров реестра или всех параметров в отдельном разделе реестра. После того как все команды будут выполнены, а все данные собраны, FRU «укажет» FSP, что файл журнала можно закрыть, что и делает FSP.

FRU управляется файлом настроек (т. е. файлом с расширением .ini), который имеет формат, похожий на INI-файлы в ОС Windows 3.1, и состоит из четырех разделов. Первый раздел, [Configuration], содержит параметры по умолчанию, с помощью которых FRU подключается к FSP, а именно сервер и порт, к которым нужно подключиться. Этот способ полезно использовать в небольших предприятиях или больших предприятиях, где сбор данных во время расследования инцидентов поручается региональным офисам. Однако эти параметры можно изменить в командной строке.

Далее идет раздел [Commands], где перечислены внешние сторонние инструменты, которые будут выполняться для сбора данных. Фактически это может быть любой PE-файл Windows, который отправляет свои выходные данные на стандартное устройство вывода (т. е. консоль). Я написал несколько небольших инструментов на языке Perl, а затем скомпилировал их в отдельные исполняемые файлы, чтобы их можно было запускать в системах, где не установлен интерпретатор языка Perl. Многие из них полезны при сборе ценной информации из систем, и их можно запустить посредством INI-файлов компонента FRU. Формат этого раздела отличается от других разделов и очень важен. Формат каждой строки выглядит следующим образом:

```
<индекс>=<командная строка>:<имя файла>
```

Индекс – это порядковый номер выполняемой команды; например, возможно, вы захотите запустить одну команду перед любыми другими, поэтому индекс позволяет

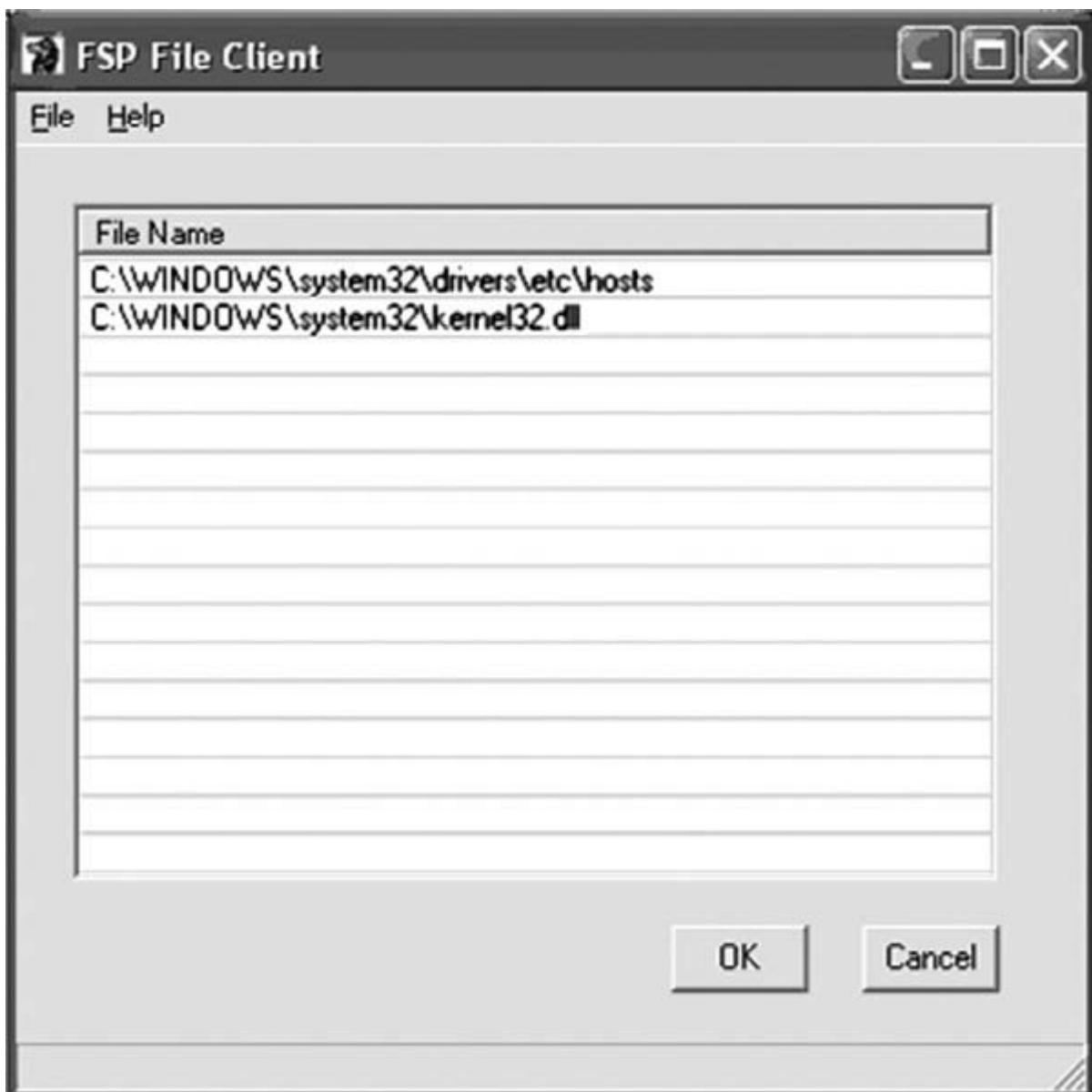
расположить команды в определенном порядке. Командная строка – это имя инструмента, который вы собираетесь запустить, а также все параметры команды, которые вы хотели бы включить, как если бы вы выполняли команду из командной строки самой системы. Первые две части разделены знаком равенства (=), а за ними следует двойное двоеточие (::). В большинстве случаев последние части одной из этих строк были бы разделены точками с запятыми, но некоторые инструменты (например, «psloglist.exe» с сайта Sysinternals.com) имеют параметры, включающие в себя возможность использования точки с запятой, поэтому мне пришлось выбрать символы, которые вряд ли будут использоваться в командной строке как разделитель. Наконец, последний элемент – это имя файла, который будет создан, чаще всего это имя инструмента с расширением .dat. Когда выходные данные отправляются в серверный компонент FSP, они будут записаны в файл (в указанном каталоге) с именем файла, к которому добавляется имя исследуемой системы. Таким образом данные можно собрать из нескольких систем, используя один и тот же выполняющийся экземпляр FSP.

Важный комментарий, касающийся инструментов, используемых с FRU: так как система, с которой взаимодействует эксперт, подключена к сети и работает, следует изменить имя применяемых сторонних инструментов. Рекомендуется добавлять к началу имен файлов что-то уникальное, например, *f_* или *fru*. Это нужно делать частично из-за того, что взаимодействие эксперта с системой будет регистрироваться каким-либо образом (подробнее об этом см. в главе 4), и из-за возможности упреждающей выборки в ОС Windows XP (подробнее об этом см. в главе 5). Помните о принципе обмена Локара? В связи с этим следует принять меры для того, чтобы артефакты, которые вы оставляете после себя в системе, отличались от всех других артефактов.

Пример из INI-файла компонента FRU выглядит следующим образом:

```
6=tcpvcon.exe -can:::tcpvcon-can.dat
```

Существует еще один клиент, предназначенный для копирования файлов из целевой системы, если эксперт захочет выполнить эту операцию. На илл. 1.15., показан графический интерфейс клиента копирования файлов (FCLI).



Илл. 1.15. Графический интерфейс клиента копирования файлов.

Чтобы использовать клиент FCLI, эксперт просто запускает его и выбирает меню «Файл» (“File”), а затем команду «Config», чтобы ввести IP-адрес и порт, используемый компонентом FSP. Затем эксперт выбирает меню «Файл» (“File”) и команду «Открыть» (“Open”) и указывает файлы, которые нужно скопировать. Указав все необходимые файлы, эксперт просто нажимает кнопку «OK». Клиент FCLI сначала соберет отметки времени MAC (изменения, доступа и создания) файла и другие метаданные, а затем вычислит хэш-значения MD5 и SHA-1 для файла. Эта информация будет отправлена в компонент FSP. Затем FCLI копирует двоичное содержимое файла на сервер. После того как операции копирования будет завершена, FSP вычислит хэш-значения для копированного файла и сверит их с хэш-значениями, полученными из клиента FCLI до начала операции копирования. Все действия происходят автоматически, без взаимодействия с экспертом, и регистрируются серверным компонентом FSP.

На носителе, который идет в комплекте с этой книгой, содержится несколько видеороликов, демонстрирующих, как настроить и использовать FSP, а также инструкции о том, где загрузить нужный плеер.

Краткое изложение

В этой главе мы познакомились с исследованием работающей системы, а именно со сбором энергозависимых (и некоторых энергонезависимых) данных из работающих систем Windows. Как отмечалось выше, работающая система содержит много данных, которые мы можем использовать, чтобы расширить наше представление об инциденте; нужно только собрать эти данные перед тем, как отключать питание системы и создавать образ НЖМД. Мы также говорили о том, что из-за изменений в области информационных технологий мы все чаще сталкиваемся с ситуациями, в которых сбор энергозависимых данных является единственным приемлемым вариантом.

В этой главе я не ставил себе целью показать вам *самый* эффективный способ сбора энергозависимых данных просто потому, что такого способа не существует. Моей целью было предоставить вам достаточно информации, чтобы, исходя из собственных потребностей и обстоятельств конкретной ситуации, вы могли не только применять свои собственные эффективные способы, но и, при изменении ситуации, применять более эффективные способы и обосновывать их применение.

Все Perl-скрипты, упомянутые и описанные в этой главе, доступны на сопроводительном носителе вместе с отдельными исполняемыми файлами, скомпилированными с помощью программы Perl2Exe. Скрипты ProScript для программы ProDiscover (от компании Technology Pathways) также доступны на сопроводительном носителе, но предоставляются только как Perl-скрипты.

Быстрое повторение

Исследование работающей системы

- Согласно принципу обмена Локара, когда два объекта соприкасаются, между ними происходит обмен веществами. Это правило также относится к области цифровых данных.
- Любое действие, даже просто присутствие, эксперта в работающей системе оказывает воздействие на эту систему и оставляет после себя артефакт. Артефакты создаются в системе во время ее работы без какого-либо взаимодействия с пользователем.
- Если артефакт отсутствует в том месте, где он должен находиться, это само по себе является артефактом.
- Порядок изменяемости энергозависимых данных показывает нам, что некоторые данные имеют более короткий «срок жизни» или «срок хранения», чем другие данные.
- При расследовании инцидента в первую очередь следует собирать наименее долговечные данные.
- Следует ясно понимать, когда необходимо проводить исследование работающей системы, и документировать причины такого решения.
- Если исследованию работающей системы не будет уделено должное внимание, действия, связанные с расследованием инцидента, могут подвергнуть организацию большему риску, чем сам инцидент.
- В соответствии с политикой безопасности некоторых организаций исследование работающей системы может быть первым этапом расследования инцидента. Специалисты по расследованию инцидентов должны внимательно следовать утвержденным процедурам и документировать свои действия.

Какие данные нужно собрать

- Большое количество данных, которые могут дать эксперту представление о его деле, доступно в системе, пока та подключена к питанию и работает, и некоторые из этих данных доступны только ограниченное время.
- Энергозависимые данные, которые вы собираете из системы, часто будут зависеть от того, с каким типом расследования или инцидента вы имеете дело.
- При сборе энергозависимых данных нужно помнить как о порядке изменяемости энергозависимых данных, описанном в документе № 3227 из серии RFC, так и о принципе обмена Локара.
- Сбор энергозависимых данных и использование этих данных для подкрепления расследования основывается на ведении подробной документации.

Энергонезависимые данные

- Энергонезависимые данные (например, параметры системы) могут оказать влияние на ваше расследование, поэтому вам, возможно, нужно будет собрать эти данные в ходе исследования работающей системы.
- Некоторые энергонезависимые данные могут повлиять как на ваше решение продолжать исследование работающей системы, так и на решение провести последующий анализ образа НЖМД.
- Решение о том, какую энергонезависимую информацию нужно собрать во время исследования работающей системы, зависит от таких факторов, как сетевая инфраструктура, политики безопасности и расследования инцидентов или конфигурация систем.

Методики исследования работающей системы

- Существует три основных типа исследования работающей системы: локальное, удаленное и комбинированное. Знание вариантов, которые вам доступны, и наличие возможностей для реализации этих вариантов обеспечат большую гибкость при сборе информации.
- Выбор методики будет зависеть от таких факторов, как сетевая инфраструктура, варианты развертывания и, возможно, даже структура политик вашей организации. Тем не менее, у вас есть несколько доступных вариантов.
- Выбирая методику исследования, не забывайте о том, что в результате ваших действий в системе будут оставаться артефакты. Ваши действия будут являться прямым воздействием на систему, что приведет к тому, что в состоянии системе будут происходить изменения: добавление разделов реестра (см. главу 4 о съемных USB-накопителях), добавление или изменение файлов и загрузка исполняемых образов в память. Однако эти изменения, до некоторой степени, поддаются количественному определению, и вы должны тщательно документировать информацию о выбранной методике и своих действиях.

Часто задаваемые вопросы

Вопрос: Когда следует проводить исследование работающей системы?

Ответ: Не существует строго определенных правил относительно того, когда следует проводить исследование работающей системы. Однако, так как все большее число регулятивных органов (например, SEC, НИРАА, FISMA, Visa PCI и другие) указывает необходимые к применению меры и механизмы безопасности, а также вопросы, которые следует проанализировать и на которые следует дать ответ (был ли получен доступ к конфиденциальной информации?), исследование работающей системы становится еще более важным.

Вопрос: Я принимал участие в расследовании дела, в котором обвиняемый, в конце концов, применил «тロянскую защиту». Как исследование работающей системы помогло бы решить эту проблему или подготовится к ней?

Ответ: Выполнив сбор информации о процессах, запущенных в системе, сетевых соединениях и других областях, где обычно находятся артефакты троянских программ или программ скрытого удаленного администрирования, вы могли бы исключить возможность того, что такие программы выполнялись в работающей системе. Анализ образа данных, который включает в себя исследование файловой системы, в том числе назначенных заданий и т. д., позволяет определить вероятность того, что троянская программа была установлена. Но сбор энергозависимых данных из работающей системы предоставляет все необходимые сведения, чтобы выяснить, выполнялась ли троянская программа в тот момент, когда работа системы еще не была завершена.

Вопрос: Сейчас я не провожу исследование работающей системы. Почему я должен начать это делать?

Ответ: Часто организация выбирает подход «стереть и переустановить», который заключается в том, что администраторы стирают данные с накопителя системы, безопасность которой предположительно была нарушена, затем переустанавливают операционную систему с незараженного носителя, снова устанавливают все приложения и восстанавливают данные из резервных копий. Такой подход считается наименее дорогостоящим. Однако такой подход никак не помогает определить *причину* инцидента. Некоторые, возможно, скажут: «Я переустановил систему и загрузил все исправления», и это здорово, но не все инциденты происходят из-за отсутствия обновлений или исправлений. Иногда причиной инцидента становится ненадежный пароль или отсутствие пароля учетной записи или приложения (например, пароль *sa* (системного администратора) SQL Server) или неправильно сконфигурированная служба. Никакие исправления не устроят такие проблемы. Если вы не определите и не устраниете причину, инцидент, вероятно, случится снова, причем незамедлительно после того, как обновленная, «с иголочки», система подключится к сети. Кроме того, как я показал в этой главе, пока система работает, в ней доступно множество ценных данных (например, физическая память, выполняющиеся процессы, сетевые соединения и содержимое буфера обмена), которые могут существенно повлиять на расследование.

Глава 2

Исследование работающей системы: Анализ данных

Содержание этой главы:

- Анализ данных

- ✓ Краткое изложение
- ✓ Быстрое повторение
- ✓ Часто задаваемые вопросы

Введение

Теперь, когда вы собрали энергозависимые данные из системы, возникает вопрос: «Как «слушать» то, что должны сказать эти данные?» или «Как понять то, что они рассказывают?» Как, после сбора списка процессов, определить, какой процесс (при наличии такого) является вредоносным? Как узнать о том, что кто-то взломал систему и в данный момент получает к ней доступ? Наконец, как использовать собранные энергозависимые данные, чтобы составить более полную картину активности в системе, особенно когда вы создали образ накопителя и выполняете его анализ?

Цель этой главы – ответить на такие типы вопросов. Задача, которую вам нужно решить во время экспертизы, во многом определяет, какие сведения или артефакты вам нужно будет найти в собранных энергозависимых данных. Как в большом количестве данных отыскать те, которые вам нужны? Я ни в коем случае не думаю, что в этой главе я смогу ответить на все ваши вопросы; вернее, я надеюсь предоставить достаточно информации и примеров, чтобы в случае возникновения ситуации, которую мы не рассмотрели, у вас был метод, посредством которого вы могли найти ответ самостоятельно. Возможно, к тому времени, когда вы дочитаете эту главу, вы сможете лучше понять, для чего были собраны энергозависимые данные и о чем они могут вам рассказать.

Анализ данных

Существует несколько источников информации, которые подскажут вам, какие данные следует собрать из работающей системы, чтобы найти причину неправильной работы приложения или определить характер инцидента. Ознакомьтесь в Интернете с такими сайтами, как e-Evidence Info (www.e-evidence.info), где ежемесячно добавляются новые ссылки на презентации и доклады с конференций, а также статьи, в которых обсуждаются различные темы, в том числе сбор энергозависимых данных. Несмотря на то, что многие из этих ресурсов имеют отношение к теме *сбора данных*, лишь в некоторых на самом деле затрагивается вопрос сопоставления и *анализа данных*. Мы рассмотрим эти вопросы в данной главе.

Вначале нужно посмотреть на выходные данные инструментов, то есть на собранную информацию, чтобы понять, какие зафиксированные данные имеются в наличии. При использовании таких инструментов, как те, что рассматривались в первой главе, вы получаете снимок состояния системы на определенный момент времени. Очень часто признак проблемы можно быстро обнаружить в выходных данных одного

единственного инструмента. Например, можно увидеть что-то необычное в окне диспетчера задач или в выходных данных инструмента «tlist.exe» (например, нетипичный путь к исполняемому образу или странную командную строку). Для эксперта, который знаком с системами Windows и знает, как выглядят стандартные или обычные процессы, такие признаки могут быть отчетливо видны и бросаться в глаза.

Совет

В статье № Q263201 (<http://support.microsoft.com/kb/263201/en-us>) из базы знаний Microsoft предоставляется информация о стандартных процессах в системах Windows 2000.

Однако многие эксперты и даже администраторы недостаточно хорошо знакомы с системами Windows, чтобы распознать стандартные или обычные процессы с первого взгляда. Это особенно верно, если учитывать тот факт, что в версиях Windows (например, Windows 2000, XP, 2003 или Vista) выполняется очень много процессов, которые можно считать обычными. Например, стандартные процессы в ОС Windows 2000 отличаются от процессов в ОС Windows XP, и это только если речь идет об операционной системе, установленной с нуля, с настройками по умолчанию и без дополнительного программного обеспечения. Также имейте в виду, что для различных конфигураций аппаратных средств часто требуются драйверы и приложения. Этот список вариаций можно продолжать, но нельзя забывать о том, что то, что считается обычным или допустимым процессом может зависеть от множества разных факторов, поэтому вам нужен метод для исследования имеющихся данных и определения источника исследуемой проблемы. Это важно, так как наличие метода означает, что у вас есть этапы, которым нужно следовать, а если что-то нужно изменить или добавить, вы можете легко это сделать. Разве, не имея метода, вы сможете определить, что пошло не так и что можно сделать, чтобы это улучшить? А если вы не знаете, что сделали, как вы это исправите?

Возможно, самый лучший способ начать – сразу погрузиться в работу. Выполнять сопоставление и анализ энергозависимых данных проще, если у вас есть представление о том, что вы ищите. Одна из самых больших трудностей, с которыми при возникновении инцидента сталкиваются специалисты и ИТ-администраторы, – это отслеживание источника инцидента, опираясь на имеющуюся информацию. Рассмотрим, например, случай, когда сетевая система обнаружения вторжений выдает предупреждение или когда в журнале брандмауэра появляется странная запись. Очень часто это может быть результатом заражения вредоносной программой (например, червем). Предупреждение или запись журнала обычно содержит такую информацию, как IP-адрес и порт источника, а также IP-адрес и порт назначения. IP-адрес источника идентифицирует систему, из которой исходит трафик, и, как вы видели в главе 1, если у вас есть порт источника сетевого трафика, можно использовать эту информацию, чтобы определить приложение, которое отправляет трафик, и выявить вредоносную программу.

Предупреждение

Имейте в виду, что, для того чтобы трафик появился в сети, какой-то процесс где-то должен был его создать. Однако некоторые процессы недолговечны (например, загрузчик, который загружает и устанавливает другой файл, такой как троян, и завершает свою работу), и попытка обнаружить процесс, основываясь на трафике, зарегистрированном только один раз в журналах брандмауэра четыре часа назад, может быть тщетной. Если трафик отображается в журналах регулярно, не забудьте проверить все возможные варианты, в том числе исследуйте исходный IP-адрес трафика, чтобы найти систему, передающую его, или даже установите программу для перехвата и анализа сетевых пакетов, чтобы определить, были ли пакеты подделаны.

Также следует помнить о том, что создатели вредоносных программ часто пытаются скрыть присутствие своих приложений в системе, используя обычное имя или имя, похожее на имя допустимого файла, которое может распознать администратор. Если эксперт выполнит поиск этого имени в Интернете, он получит информацию, свидетельствующую о том, что это безопасный файл или допустимый файл, используемый операционной системой.

Предупреждение

Расследуя инцидент, связанный с атакой червя в корпоративной среде, я определил, что компонент червя был установлен в системе как служба Windows, которая запускалась из исполняемого образа с именем «alg.exe». Выполнив поиск информации об этом имени файла, администратор узнал, что это допустимая служба, которое называется «Служба шлюза уровня приложения» (“Application Layer Gateway Service”). Настройки этой службы хранятся в реестре в разделе CurrentControlSet\Services, в подразделе ALG, и путь к ее исполняемому образу указан как %SystemRoot%\system32\alg.exe. Однако обнаруженная мной служба была расположена в подразделе Application Layer Gateway Service (первая подсказка: неправильное имя подраздела), а путь к ее исполняемому файлу был указан как %SystemRoot%\alg.exe. Нужно быть очень внимательным при выполнении поиска сведений об именах файлов, так как даже самого лучшего специалиста может сбить с толку информация, возвращаемая в результате такого поиска. Я видел, как бывалые специалисты по анализу вредоносных программ ошибались при определении типа файла, используя для этого только его имя.

Чтобы лучше прояснить все эти моменты, давайте рассмотрим несколько примеров.

Пример 1

Один из самых распространенных сценариев – когда администратор или служба поддержки получает сообщение о необычных или подозрительных действиях в системе. Это может быть случай, когда пользователь сообщает о необычном поведении системы или когда администратор сервера обнаруживает подозрительные файлы на веб-сервере, но при попытке удалить их получает сообщение, что эти файлы удалить нельзя, так как они используются другим процессом.

В таких случаях специалист по расследованию инцидентов будет иметь дело с системой, которую нельзя выключить (из-за ограничений временного или коммерческого характера) для создания образа данных и проведения его анализа, и ему нужно будет быстро (но исчерпывающее) ответить на все вопросы. Очень часто это можно выполнить посредством исследования работающей системы, во время которого можно быстро собрать и проанализировать достаточное количество информации, чтобы получить наиболее полное представление о текущем состоянии системы. Несмотря на то, что при сборе информации из работающей системы процесс сбора можно повторить, саму информацию обычно нельзя копировать повторно, так как работающая система постоянно находится в состоянии изменения.

Всякий раз, когда что-то происходит в системе, это является результатом какого-нибудь процесса, выполняющегося в этой системе. Хотя это утверждение может показаться «объективно очевидным даже случайному наблюдателю» (фраза, которую один из моих преподавателей в аспирантуре повторял несколько раз за урок, обычно при наличии дифференциального уравнения шестого порядка), этот факт часто упускают из-за стресса или напряжения, связанного с расследованием инцидента. Но на самом деле для того чтобы что-то произошло в системе, необходимо какое-нибудь участие процесса или потока выполнения.

Совет

В своей статье «Exploiting the Rootkit Paradox with Windows Memory Analysis» Джесси Корнблюм (Jesse Kornblum) отмечает, что руткиты, как и большинство вредоносных программ, нужно запустить или выполнить. Понимание этого факта – ключ к исследованию работающей системы.

Итак, как специалисту по расследованию инцидентов обнаружить подозрительный процесс в системе? Ответ – при помощи сбора и анализа данных работающей системы. И поверьте, в моей практике были случаи, когда клиент предоставлял мне НЖМД системы (или образ НЖМД) и просил рассказать ему, какие процессы выполнялись в системе. Дело в том, что *необходимо* иметь информацию, собранную из работающей системы, чтобы показать, что происходило во время работы этой системы. Используя инструменты, рассмотренные в главе 1, можно собрать информацию о состоянии системы на определенный момент времени, создав снимок этого состояния. Так как информация, которую вы собираете, существует в энергозависимой памяти, после выключения системы эта информация исчезает.

В данном сценарии у меня есть система Windows 2000, поведение которой кажется странным. Система содержит веб-сервер интрасети, работающий под управлением платформы IIS (Internet Information Server) версии 5.0, а пользователи, пытавшиеся получить доступ к страницам на сервере, сообщили, что не могут найти никакой информации и видят только пустые страницы в своих браузерах. Это странно, так как можно было бы ожидать увидеть, например, сообщение об ошибке. Поэтому я запускаю платформу Forensic Server (т. е. Forensic Server Project [FSP], см. главу 1) на своем судебном компьютере (IP-адрес – 192.168.1.6), используя следующую команду:

```
C:\fsp>fspc -c cases -n testcase1 -i "H. Carvey" -v
```

Затем я беру компакт-диск, содержащий мои инструменты и компонент First Responder Utility (т. е. «fruc.exe», также см. главу 1), и нахожу проблемную систему. В таких случаях я самога начала придерживаться минималистского подхода; я предпочитаю сводить к минимуму воздействие на систему (не забывайте о принципе обмена Локара) и оптимизировать свои усилия и время на исследование. Для этого я постепенно составляю минимальный набор сведений о состоянии, которые мне нужно извлечь из работающей системы, чтобы получить о ней достаточно полное представление, которое поможет мне обнаружить потенциально подозрительные действия.

Совет

На носителе, который идет в комплекте с этой книгой, содержатся видеоролики, демонстрирующие использование компонентов платформы FSP.

Я также определяю набор инструментов, которые можно применить для извлечения этой информации (файл «fruc.ini», используемый с компонентом First Responder Utility [FRU] в этом сценарии, содержится в каталоге «ch2\samples» на сопроводительном носителе). Раздел [Commands] файла «fruc.ini» содержит следующие записи:

```
1=psloggedon.exe::psloggedon.dat
2=netusers.exe -l -h::netusers-lh.dat
3=tlist.exe -c::tlist-c.dat
4=tlist.exe -s::tlist-s.dat
5=tlist.exe -t::tlist-t.dat
```

```

6=handle.exe -a -u::handle-au.dat
7=listdlls.exe::listdlls.dat
8=tcpvcon.exe -can::tcpvcon-can.dat
9=autorunsc.exe -l -d -s -t -w::autorunsc-ldstw.dat
10=svc.exe::svc.dat 11=auditpol.exe::auditpol.dat

```

Каждая команда выполняется по порядку, и в данном списке можно увидеть команды для сбора информации о пользователях, вошедших в систему (локально и удаленно), а также о предыдущих входах в систему, местах автозапуска, процессах, сетевых соединениях и открытых портах, службах и политике аудита в системе. Этот набор команд позволяет не только получить исчерпывающее представление о состоянии системы на момент времени, но и собрать информацию, которая может помочь как при непосредственном анализе данных, так и на последующих этапах исследования.

Совет

Как отмечалось в главе 1, компонент FRU можно запускать с использованием разных файлов конфигурации (т. е. INI-файлов). В случаях, связанных с возможным нарушением корпоративной политики допустимого использования (когда пользователи используют информационные системы не по назначению), вам могут потребоваться дополнительные INI-файлы, чтобы собрать содержимое буфера обмена и, возможно, информацию из защищенного хранилища.

Найдя компьютер с проблемной системой Windows 2000, я вставляю компакт-диск с утилитой FRU в его дисковод CD-ROM, запускаю командную строку и ввожу следующую команду:

```
E:\>fruc -s 192.168.1.6 -p 7070 -f fruc.ini -v
```

Примечание

При каждом исследовании работающей системы я настоятельно рекомендую собирать полное содержимое физической памяти (ОЗУ), прежде чем выполнять любые другие действия. Это позволит вам получить содержимое ОЗУ в максимально возможном нетронутом состоянии перед внесением дополнительных изменений в состояние системы. Хотя эта тема выходит за рамки данной главы, таким образом обеспечивается плавный переход к главе 3.

Через несколько секунд все нужные мне данные извлекаются из системы и надежно сохраняются на моем судебном компьютере для анализа.

Совет

Данные, которые я собрал во время этого сценария, находятся в каталоге «ch2\samples» на сопроводительном носителе, в архиве с именем « testcase1.zip ».

Вернувшись на судебный компьютер, я вижу, что, как и ожидалось, каталог « testcase1 » содержит 16 файлов. Одно из преимуществ платформы FSP состоит в том, что она самодокументирующаяся; файл « fruc.ini » содержит список инструментов и команд, используемых для запуска этих инструментов при сборе данных. Так как этот файл и сами инструменты находятся на компакт-диске, их нельзя изменить; поэтому, пока у меня хранится этот компакт-диск, у меня будет неизменная информация о том, какие инструменты (версия каждого из них и т. д.) запускаются в системе и какие параметры используются для запуска этих инструментов. Один из файлов в каталоге « testcase1 » – это « case.log », содержащий перечень данных, отправленных на сервер компонентом FRU, и хэш-значения MD5 и SHA-1 для файлов, в которых были сохранены эти данные. Кроме

того, я вижу файл «case.hash», в котором сохраняются хэш-значения MD5 и SHA-1 для файла «case.log» после его закрытия.

Интересующая меня информация содержится в других 14 файлах в каталоге для этого дела. Как правило, на начальном этапе анализа я в первую очередь просматриваю данные на наличие каких-нибудь необычных процессов. Чаще всего я начинаю с выходных данных команды *tlist -c*, так как в них показана команда, использовавшаяся для запуска каждого активного (и видимого) процесса в системе. Например, один из процессов, который сразу бросается в глаза, – это сам процесс FRUC (что наглядно демонстрирует важность создания в первую очередь дампа физической памяти):

```
1000 FRUC.EXE
Command Line: fruc -s 192.168.1.6 -p 7070 -f fruc.ini -v
```

Просматривая остальную часть данных в файле, я вижу много обычных процессов, то есть процессов, выполнение которых я привык видеть в ОС Windows. Затем я встречаю процесс для веб-сервера IIS, который, как мне известно, выполняется в системе:

```
736 inetinfo.exe
Command Line: C:\WINNT\system32\inetsrv\inetinfo.exe
```

Просматривая данные далее, я вижу процесс, который сразу же кажется мне необычным и подозрительным:

```
816 inetinfo.exe
Command Line: inetinfo.exe -L -d -p 80 -e c:\winnt\system32\cmd.exe
```

Большинство веб-серверов IIS имеет только один выполняющийся экземпляр процесса «inetinfo.exe», а в этой системе их два. Кроме того, обычная версия процесса «inetinfo.exe» выполняется по умолчанию из каталога «system32\inetsrv», как мы видели в случае с экземпляром «inetinfo.exe» с идентификатором процесса 736. Однако экземпляр процесса «inetinfo.exe» с идентификатором 816 выполняется из каталога «system32»; кроме того, команда, использовавшаяся для запуска этого процесса, подозрительно похожа на команду, которая используется для запуска инструмента netcat!

Желая получить больше информации об этом процессе и отмечая, что команда для процесса с идентификатором 816, похоже, связывает этот процесс с портом 80 (что объяснило бы необычное поведение, о котором сообщили пользователи), я открываю для просмотра файл с выходными данными восьмой команды, запущенной из файла «fruc.ini», то есть *tcpvcon.exe -can*:

```
TCP, C:\WINNT\system32\inetsrv\inetinfo.exe,736,,127.0.0.1:443,.*.*
TCP, C:\WINNT\system32\inetsrv\inetinfo.exe,736,,127.0.0.1:21,.*.*
TCP, C:\WINNT\system32\inetsrv\inetinfo.exe,736,,127.0.0.1:25,.*.*
TCP, C:\WINNT\system32\inetsrv\inetinfo.exe,736,,127.0.0.1:1026,.*.*
TCP, C:\WINNT\system32\inetsrv\inetinfo.exe,816,,127.0.0.1:80,.*.*
```

Обычно я ожидал бы увидеть, что процесс с идентификатором 736 привязан к порту 80, но в данном случае к этому порту привязан процесс с идентификатором 816.

Как видите, я установил, что процесс с идентификатором 816 является подозрительным, и, похоже, что этот процесс обуславливает необычное поведение, о котором сообщили пользователи. Проверяя выходные данные других команд, я не вижу каких-нибудь необычных выполняющихся служб или упоминаний об этом процессе в местах автозапуска. Выходные данные инструмента «handle.exe» показывают, что этот процесс выполняется от имени учетной записи администратора, но, похоже, что с ним не связано никаких открытых файлов. Кроме того, в выходных данных команды *tcpvcon.exe -*

can показано, что в данный момент в этой системе отсутствуют соединения с портом 80. На данном этапе я установил проблему, и теперь мне нужно определить, как эта программа попала в систему и впоследствии начала выполняться как процесс.

Пример 2

Еще один популярный сценарий, наблюдаемый в сетевой среде, – когда записи о необычном трафике, исходящем из системы, появляются в журналах брандмауэра или системы обнаружения вторжений. Чаще всего администраторы замечают что-то подозрительное, например, необычный трафик, выходящий из сети. К таким примерам часто относятся заражения IRC-ботом или червем. Обычно IRC-бот заражает системы в результате просмотра пользователем веб-страницы, содержащей какой-нибудь код, использующий уязвимость в веб-браузере. Сначала, как правило, в систему копируется исходная программа-загрузчик, которая затем связывается с другим веб-сайтом, чтобы загрузить и выполнить фактический код IRC-бота. Затем IRC-бот получает доступ к каналу IRC-сервера и ожидает команд от владельца ботнета.

Предупреждение

IRC-боты довольно долгое время были огромной проблемой, так как целые армии ботов, или ботнеты, оказались вовлечеными в несколько киберпреступлений. В номере журнала *Washington Post Magazine* от 19 февраля 2006 года Брайан Кребс (Brian Krebs) представил миру статью о владельце ботнета под псевдонимом 0x80. В его рассказе ясно показывалось, с какой легкостью развивались ботнеты, и каким образом их можно использовать. (Статья доступна по адресу www.washingtonpost.com/wp-dyn/content/article/2006/02/14/AR2006021401342.html.) Всего через несколько месяцев Роберт Лемос (Robert Lemos) в своей статье на сайте SecurityFocus (www.securityfocus.com/news/11390) предупреждал, что владельцы IRC-ботов, похоже, отказываются от клиент-серверной платформы в пользу одноранговой архитектуры, что значительно затрудняет борьбу с ботнетами.

Сразу после заражения системы червь попытается получить доступ к другим системам и заразить их. Черви обычно делают это посредством сканирования IP-адресов и поиска такой же уязвимости (многие современные черви пытаются использовать несколько уязвимостей), которую они использовали для заражения данного компьютера. Некоторые черви довольно опасны при сканировании; червь SQL Slammer (www.cert.org/advisories/CA-2003-04.html) свирепствовал в Интернете в январе 2003 года, создавая столько трафика, что сервера и даже банкоматы, соединенные через Интернет, подверглись массированным DoS-атакам (атаки типа «отказ в обслуживании»).

Упоминание о DoS-атацах заставляет вспомнить о еще одном важном аспекте этого сценария. Иногда стороннее лицо сообщает ИТ-администраторам, что их системы, возможно, заражены. В таких случаях владелец системы, которая сканируется червем или подвергается DoS-атаке, как правило, увидит исходный IP-адрес трафика в перехваченных сетевых пакетах, выполнит поиск сведений о владельце этого IP-адреса (обычно это диапазон адресов, а не отдельный IP-адрес, назначенный какому-нибудь пользователю) и попытается связаться с ним. Совершенно верно, даже в 2009 году не так уж редки случаи, когда кто-то постучится к вам в дверь, чтобы сообщить, что ваши системы заражены.

Независимо от того, как администраторов извещают о заражении, вопрос расследования остается прежним. Одна из трудностей, связанных с такими вопросами, состоит в том, что, имея в своем распоряжении IP-адрес и номер порта (которые были получены из заголовков перехваченных сетевых пакетов), администратор должен затем определить характер инцидента. Как правило, для этого необходимо установить физическое местонахождение системы, а затем собрать и проанализировать информацию из этой системы.

Данный сценарий начинается и развивается почти таким же образом, как и предыдущий, так как я запускаю FSP на судебном компьютере, отправляясь к целевой системе с компакт-диском с компонентом FRUC и собираю энергозависимые данные из этой системы.

Совет

Данные, которые я собрал во время этого сценария, находятся в каталоге «ch2\samples» на сопроводительном носителе, в архиве с именем « testcase2.zip ».

Вернувшись к судебному компьютеру, я открываю выходные данные команды *tlist.exe -t* (которая показывает дерево задач с перечислением каждого процесса, с отступом под родительским процессом), и процесс с идентификатором 980 кажется мне странным:

```
System Process (0)
System (8)
    SMSS.EXE (140)
    CSRSS.EXE (164)
    WINLOGON.EXE (160) NetDDE Agent
        SERVICES.EXE (212)
            svchost.exe (404)
            spoolsv.exe (428)
            svchost.exe (480)
            regsvc.exe (532)
            mstask.exe (556) SYSTEM AGENT COM WINDOW
            snmp.exe (628)
            VMwareService.e (684)
            WinMgmt.exe (600)
            svchost.exe (720)
                wuauctl.exe (1080)
            inetinfo.exe (736)
            svchost.exe (1192)
    LSASS.EXE (224)
explorer.exe (520) Program Manager
    VMwareTray.exe (1232)
    VMwareUser.exe (1256)
    WZQKPICT.EXE (1268) About WinZip Quick Pick
    CMD.EXE (812) Command Prompt - svchost 192.168.1.28 80
        svchost.exe (980)
```

Чтобы понять, почему этот процесс кажется мне странным, важно знать, что в ОС Windows 2000, настроенной по умолчанию, обычно выполняется только два экземпляра «svchost.exe».

Совет

Статья № Q250320 (<http://support.microsoft.com/?kbid=250320>) из базы знаний Microsoft предоставляет описание процесса «svchost.exe» в ОС Windows 2000, а в статье № Q314056 (<http://support.microsoft.com/kb/314056/EN-US/>) описывается то же процесс в ОС Windows XP. В примере выходных данных команды *tlist -s* не только показано два выполняющихся экземпляра «svchost.exe», но и указывается раздел реестра, в котором перечислены группы, продемонстрированные в статье. Также ознакомьтесь со статьей № Q263201 из базы знаний Microsoft (<http://support.microsoft.com/?kbid=263201>), где перечислены стандартные процессы, встречающиеся в ОС Windows 2000.

В выходных данных команды *tlist -t* показан дополнительный экземпляр «svchost.exe», который, похоже, запущен из окна командной строки, а не из процесса «services.exe», как в случае с другими экземплярами «svchost.exe».

Проверяя выходные данные команды *tlist -c*, чтобы просмотреть параметры команды, использовавшиеся для запуска процесса с идентификатором 980, я вижу:

```
980 svchost.exe
    Command Line: svchost 192.168.1.28 80
```

В выходных данных команды *tcpvcon.exe -can* показано, что процесс с идентификатором 980 использует порт клиента:

```
TCP,C:\WINNT\system\svchost.exe,980,ESTABLISHED, 192.168.1.22:1103,
192.168.1.28:80
```

Если бы клиент выполнялся, а в ОС Windows 2000 можно было бы использовать переключатель *-o*, то выходные данные команды *netstat.exe -ano* также показали бы мне, что процесс с идентификатором 980 имеет активное соединение с портом 80 удаленной системы.

```
TCP      192.168.1.22:1103      192.168.1.28:80      ESTABLISHED      980
```

На данном этапе, исходя из имеющейся у меня информации, мне, возможно, нужно будет понаблюдать за сетевым трафиком. Установив анализатор пакетов или используя систему с установленным анализатором пакетов (таким как Wireshark с сайта www.wireshark.org), чтобы начать перехватывать трафик, я увижу, какие данные передаются из одной системы в другую. Судя по другим собранным энергозависимым данным, похоже, что процесс с идентификатором 980 не используется для открытия файлов (согласно данным инструмента «handle.exe») и что в системе нет других необычных процессов.

При просмотре процессов в системе полезно немного разбираться в том, как создаются процессы по отношению друг к другу. Например, как было показано в выходных данных команды *tlist -t* (полученных из системы Windows 2000), большинство системных процессов порождается процессом с именем «System» (процесс с идентификатором 8 в ОС Windows 2000 и идентификатором 4 в ОС Windows XP), а большинство пользовательских процессов порождается процессом «explorer.exe», то есть оболочкой, которая в инструменте «tlist.exe» называется «Program Manager». Обычно (я очень аккуратно использую это слово, так как возможны исключения) мы видим, что процесс System является родительским процессом для процесса «services.exe», который в свою очередь является родительским процессом для, скажем так, многих служб. Процесс «services.exe» является родительским для процессов «svchost.exe», например. С другой стороны, командная строка («cmd.exe») будет отображаться как дочерний процесс для процесса «explorer.exe», а любая команда, запущенная из командной строки, например, *tlist -t*, будет отображаться как дочерний процесс для «cmd.exe».

Итак, какое отношение это имеет к исследованию работающей системы? Посмотрите еще раз на выходные данные команды *tlist -t*. Вы увидите экземпляр процесса «svchost.exe» (с идентификатором 980), выполняющийся как дочерний процесс для «cmd.exe», который сам является дочерним процессом для «explorer.exe» – совсем не то, что мы ожидали увидеть для «svchost.exe»!

Давайте рассуждать дальше. Что если выполняющийся «svchost.exe» (с идентификатором 980) был установлен как служба? Хотя мы бы не заметили этого в выходных данных команды *tlist -t*, мы бы увидели что-то необычное в выходных данных команды *tlist -c*, в которых показаны команды, использовавшиеся для запуска каждого

процесса. Благодаря функции «Защита файлов Windows» («Windows File Protection, WFP) вредоносный процесс «svchost.exe» мог бы быть создан в любом каталоге, кроме «system32». Функция WFP – это механизм (впервые появившийся в ОС Windows 2000), защищающий определенные системные (и другие очень важные) файлы. Если выполняется попытка изменить эти файлы, функция WFP активируется и автоматически заменяет измененный файл новой копией файла из своего кэша (создавая запись об этом действии в журнале регистрации событий). В ОС Windows 2000 были уязвимости, используя которые можно было легко нарушить работу функции WFP, но они были исправлены. Поэтому, при условии, что работа функции WFP не была никоим образом нарушена, мы бы ожидали, что вредоносный процесс «svchost.exe» будет выполняться из другого каталога, например, из «Windows\System» или «Temp», предупреждая нас о злом намерении.

Предупреждение

Работу функции WFP можно нарушить во всех системах Windows, в некоторых случаях довольно легко. Недокументированная функция API, доступная посредством файла «sfc_os.dll» и экспортруемая по порядковому номеру 5, была названа *SfcFileException*; эта тема обсуждается на сайте Bitsum Technologies (www.bitsum.com/aboutwfp.asp), а также на других сайтах в Интернете (конечно, за исключением microsoft.com). Согласно описанию этой API-функции, при ее правильном вызове функция WFP будет отключена на одну минуту – этого достаточно, чтобы изменить или заменить защищенный файл. Функция WFP не опрашивает файлы, которые она защищает, поэтому при возобновлении работы WFP ничего не указывает ей о том, что файл был изменен. При обычных обстоятельствах, когда в операционной системе создается событие изменение файла, WFP активируется и проверяет, произошло ли это событие для защищенного файла, и если да, заменяет этот файл «заведомо исправной» копией файла из кэша и создает запись об этом в журнале регистрации событий. Когда WFP отключается на одну минуту посредством API-функции *SfcFileException*, нет ничего, что могло бы обнаружить факт изменения файла или предупредить об этом факте. Мы подробнее рассмотрим эту тему в главе 5, где я также продемонстрирую инструмент анализа, обнаруживающий признаки таких действий во время анализа образа данных.

Пример 3

Инструмент «psexec.exe» от Microsoft (<http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>) позволяет наглядно продемонстрировать, как можно найти необычный или подозрительный процесс в системе. Очень часто злоумышленник получает доступ к системе каким-нибудь способом, пользуясь привилегиями уровня администратора, и повышает эти привилегии до уровня системы. Это делается частично для того, чтобы (а) не дать администратору возможности заметить, что злоумышленник находится в системе или выполняет какой-нибудь процесс, и (б) не дать администратору возможности просто остановить выполнение вредоносного процесса.

Итак, сначала мы загрузим экземпляр инструмента «psexec.exe» с сайта Microsoft (Sysinternals), а затем запустим его, используя следующую команду:

```
C:\tools>psexec -s cmd
```

На данном этапе у нас все еще открыта командная строка, но она выполняется с привилегиями уровня системы. Теперь, чтобы добавить данные для наблюдения, запустите игру «Косынка», введя **sol** в командной строке. Вы заметите, что игра не открылась на рабочем столе, но приглашение на ввод команды снова появилось без сообщения об ошибке.

Теперь откройте другую командную строку и введите команду **tlist.exe -t**. Выходные данные будут выглядеть примерно так, как показано ниже (для краткости показан сокращенный фрагмент):

```
D:\tools>tlist -t
System Process (0)
System (4)
    smss.exe (968)
    csrss.exe (1032)
    winlogon.exe (1060)
        services.exe (1104)
            svchost.exe (1360)
            svchost.exe (1704)
                wscntfy.exe (316)
                svchost.exe (1968)
                svchost.exe (352)
                spoolsv.exe (872)
                scardsvr.exe (932)
                alg.exe (1768)
                PSEXESVC.EXE (1560)
                    cmd.exe (3664)
                    sol.exe (3832)
    lsass.exe (1116)
explorer.exe (372) Program Manager
    DLACTRLW.EXE (780)
    cmd.exe (2748) Command Prompt - tlist -t
        tlist.exe (2196)
    cmd.exe (2684) \\WINTERMUTE: cmd
        psexec.exe (3448)
```

Обратите внимание, что под процессом «explorer.exe» перечислены командные строки, выполняющиеся как для «tlist.exe», так и для «psexec.exe» (процессы с идентификаторами 2748 и 2684 соответственно). Однако вы также увидите запись о выполняющемся процессе «PSEXESVC.EXE» над «explorer.exe» в древовидном представлении процессов. Это связано с тем, что этот процесс выполняется с привилегиями уровня системы. Под процессом «PSEXESVC.EXE», с отступом, который обозначает дочерний процесс, находится еще одна командная строка (процесс с идентификатором 3664), а ниже этого процесса находится дочерний процесс «Косынка». Процессы, выполняющиеся как службы (например, «sol.exe»), не запускаются в интерактивном режиме, в отличие от тех случаев, когда они обычно запускаются пользователем.

Я использовал этот пример, чтобы показать, что может произойти, когда при создании учетных записей пользователей не соблюдается принцип наименьших привилегий. Часто какой-нибудь загрузчик попадает в систему после посещения пользователем вредоносного веб-сайта или другим способом, например, через вложение электронной почты. Затем загрузчик загружает вредоносную программу, и, так как учетная запись пользователя выполняется с привилегиями администратора, эта вредоносная программа имеет возможность делать все, что может пользователь, например, создавать назначенные задания или устанавливать службы Windows. Если вредоносная программа получит привилегии выше учетной записи администратора, он будет иметь беспрепятственный доступ ко всем ресурсам системы. Кроме того, эта вредоносная программа больше не будет взаимодействовать с рабочим столом, а это означает, что некоторые артефакты не будут доступны.

Гибкий анализ

Возможно, одна из самых распространенных причин совсем не проводить исследование работающей системы заключается в неумении найти источник проблемы в большом количестве собранных данных. Многие инструменты для сбора энергозависимой (и энергонезависимой) информации во время исследования работающей системы собирают такое количество данных, что их обработка может показаться эксперту непосильной задачей. В примерах дел, показанных в этой главе, мне не нужно было собирать большое количество данных, чтобы точно определить источник проблемы. При использовании инструментов для сбора данных в этих примерах я учитывал два простых факта: вредоносная программа должна выполняться, чтобы оказывать воздействие на систему, и должна работать стабильно, чтобы оказывать постоянное воздействие на систему (в идеале, создатели вредоносных программ хотят, чтобы те сохраняли работоспособность после перезагрузок и входов пользователей в систему). Мы также используем эти два простых правила в своем анализе, чтобы найти источник проблемы в имеющихся данных. Чтобы выполнить быстрый и гибкий анализ, необходимо рассмотреть способы автоматизации и сокращения объема данных.

Хотя примеры дел довольно просты, они, тем не менее, показывают, что я имею в виду. Методика, используемая для поиска подозрительного процесса в каждом деле, не сильно отличается от методики, применявшейся для расследования инцидента, связанного с ботом russiantopz (www.securityfocus.com/infocus/1618) в 2002 году. На самом деле она похожа на дифференциальный анализ (т. е. поиск различий между двумя состояниями). Однако важно иметь в виду, особенно при проведении исследования работающей системы в качестве сотрудника правоохранительных органов или консультанта, что в большинстве случаев исходные данные о состоянии системы до момента инцидента будут недоступны, и вам нужно будет полагаться на понимание внутренних механизмов базовой операционной системы и приложений, чтобы получить представление об этих исходных данных. Например, если бы в первом примере сведения о процессах содержали запись только об одном экземпляре «inetinfo.exe» и если бы я не знал, работает ли веб-сервер в зараженной системе, я мог бы сопоставить то, что мне известно (т. е. что процесс «inetinfo.exe» выполняется), с выходными данными инструмента «svc.exe», которые в данном случае выглядят так:

```
736,W3SVC,World Wide Web Publishing Service
,C:\WINNT\system32\inetsrv\inetinfo.exe,Running,Auto,Share Process,#
```

Это сопоставление можно автоматизировать посредством использования скриптовых инструментов, и, если окажется, что допустимая служба (например, та, что показана выше) сопоставляется с допустимым процессом («inetinfo.exe» с идентификатором 736), мы таким образом выполним сокращение объема данных.

Примечание

Инструмент «svc.exe», используемый в этих примерах, собирает информацию о службах в системе и показывает результаты в формате значений, разделенных запятыми (.csv), чтобы эти результаты можно было легко проанализировать или открыть для анализа в программе Excel. Заголовки столбцов в выходных данных – идентификатор процесса, имя службы, отображаемое имя службы, путь к исполняемому образу, состояние службы, режим запуска службы, тип службы, а также наличие (#) или отсутствие (*) строки описания для службы. Создатели вредоносных программ часто не удосуживаются добавить строку описания для своей службы; отсутствие такой строки будет поводом тщательнее исследовать эту службу.

Практическое правило, о котором должен помнить компетентный эксперт во время анализа энергозависимых данных, заключается в том, что существование процесса «inetinfo.exe» без наличия выполняющейся службы W3SVC (служба веб-публикаций) может свидетельствовать о присутствии вредоносной программы или, по крайней мере, процесса, который заслуживает пристального внимания.

Однако эксперт в то же время не должен забывать, что процесс «inetinfo.exe» также поддерживает службу FTP Microsoft и службу SMTP, как показано в выходных данных команды *tlist -s*:

```
736 inetinfo.exe Svcs: IISADMIN,MSFTPSVC,SMTPSVC,W3SVC
```

Проще говоря, выполняющийся процесс «inetinfo.exe» без соответствующих выполняющихся служб может свидетельствовать о проблеме. Опять же эту проверку также можно автоматизировать. Например, если бы выходные данные инструментов FRUC были проанализированы и введены в базу данных, можно было бы использовать инструкции SQL, чтобы извлечь и сопоставить информацию.

Совет

Во время представления своей презентации на конференции BlackHat DC 2007 Кевин Мандиа (Kevin Mandia) заявил, что большое количество инцидентов, расследованных его компанией за предыдущий год, свидетельствует о том, что для поддержания сохранности своих продуктов создатели вредоносных программ все чаще используют способ установки под видом службы Windows. Как показывает мой опыт, это правда. На самом деле в нескольких случаях я видел, как вредоносная программа попадала в систему и создавала службу, которая затем предоставляла доступ к командной строке и перенаправляет ее данные в удаленную систему. Когда злоумышленник подключается на другом конце, он получает, хоть и через командную строку, доступ системного уровня к взломанной системе.

Это показывает, что при определенных знаниях и усилиях можно быстро и эффективно решить проблемы посредством использования средств автоматизации и сокращения объема данных. Автоматизация необходима, так как расследование инцидентов обычно характеризуется стрессом и напряжением – состояния, при которых существует большая вероятность совершить ошибку. Автоматизация позволяет нам систематизировать метод работы и иметь возможность снова и снова применять тот же метод. Если мы понимаем, какие артефакты и энергозависимые данные предоставят нам достаточно полную картину о состоянии системы, мы можем быстро собрать и сопоставить информацию, а также определить характер и масштаб инцидента. Это позволяет провести более гибкое, быстрое и при этом тщательное исследование, используя документированный метод работы. Затем можно при необходимости собрать дополнительные энергозависимые данные. Изначально используя такой минималистский подход, мы можем уменьшить количество данных, которые нужно проанализировать и сопоставить, и в целом провести лучшее исследование.

Что касается анализа и автоматизации, практические правила, используемые экспертами для поиска подозрительных процессов в собранных энергозависимых данных, главным образом основаны на опыте и понимании внутренних механизмов базовой операционной системы и приложений.

Совет

Несколько лет тому назад мой друг часто присыпал мне энергозависимые данные, которые собирал во время расследования различных инцидентов. Он использовал ряд инструментов и пакетный файл для сбора этих данных, а спустя много времени после

завершения дела присыпал мне файлы необработанных данных и просил меня выяснить причину инцидента. Не зная состояние и тип исходной системы, мне приходилось искать ключ к решению этой задачи в присланных данных. Это отличный способ для совершенствования навыков и даже для разработки некоторых инструментов сопоставления данных.

Некоторые из этих правил можно систематизировать в виде процедур или даже скриптов, чтобы сделать процесс анализа и сокращения объема данных более эффективным. В качестве примера рассмотрим процесс «svchost.exe». Некоторые создатели вредоносных программ пользуются тем, что обычно в системах Windows выполняется несколько экземпляров «svchost.exe» (как показывает мой опыт, два экземпляра этого процесса выполняются в ОС Windows 2000, пять – в ОС Windows XP SP2 и семь – в ОС Windows 2003), и присваивают это имя своим программам. Нам известно, что допустимый процесс «svchost.exe» следует нескольким простым правилам, одно из которых заключается в том, что процесс возникает из исполняемого образа, расположенного в каталоге «system32». Следовательно, можно написать Perl-скрипт, который обработает выходные данные команды `tlist -c` и сразу отметит экземпляры процесса «svchost.exe», выполняющегося *не* из каталога «system32».

Такой подход отличается от метода «искусственного неведения», при котором сокращение объема обрабатываемых данных выполняется посредством отбрасывания всех заведомо допустимых данных, а все, что останется после этого, скорее всего, подлежит тщательному изучению. (Термин «искусственное неведение» (англ. *artificial ignorance*) был создан Маркусом Ранумом (Marcus Ranum), о котором можно узнать на сайте www.ranum.com). Я довольно эффективно использовал этот метод в корпоративной среде, не только во время расследования инцидентов, но и при проведении проверок сети на наличие шпионских программ и других проблем. Я написал Perl-скрипт, который обращался к основному контроллеру домена и получал список всех рабочих станций, которые тот «видит» в сети. Затем я подключился к каждой рабочей станции, используя учетные данные администратора домена, извлек содержимое раздела Run (дополнительную информацию об этом разделе реестра см. в главе 4) из каждой системы и сохранил эту информацию в файле на своем компьютере. После запуска скрипта первый раз я получил несколько страниц данных, которые нужно было обработать. Поэтому я начал изучать некоторые найденные элементы и обнаружил, что многие из них связаны с допустимыми приложениями и драйверами. В связи с этим я создал список заведомо допустимых элементов, а затем, при сканировании систем, я сверял полученную информацию с данным списком и записывал в свой файл журнала только те данные, которых *не было* в списке. Довольно быстро я сократил свой файл журнала примерно до половины страницы.

Это один из подходов, который можно использовать, чтобы быстро проанализировать собранные энергозависимые данные. Однако самое главное для гибкого анализа и быстрого расследования – сократить объем данных, которые фактически нужно исследовать. Это можно реализовать, например, путем преобразования данных в более удобный формат или же отсеивания заведомо допустимых артефактов.

Повышение компетентности

Что будет, если вы столкнетесь с более сложной ситуацией, чем уже рассмотренные сценарии? Эксперты по вопросам информационной безопасности постоянно подчеркивают в СМИ, что киберпреступления становятся все более изощренными (что соответствует действительности). Итак, как справляться с более сложными инцидентами? Ведь не все процессы, связанные с инцидентом, могут быть такими долговечными, как те, что были показаны в примерных сценариях. Например, загрузчик может попасть в систему через уязвимость в веб-браузере, но, загрузив целевую

программу на компьютер, он завершит свою задачу и больше не будет активен. Следовательно, информация об этом процессе, в том числе о сетевых соединениях, используемых этим процессом, больше не будет доступна.

Недавно я расследовал инцидент, связанный с зашифрованным исполняемым файлом, который не могли идентифицировать более двадцати антивирусных приложений. Кроме того, значительную трудность при расследовании представлял тот факт, что ни в одной из зараженных систем не было ни одного выполняющегося процесса с таким же именем, что у загадочного файла. Динамический анализ (см. главу 6) вредоносной программы показал, что эта программа внедряется в пространство процесса браузера Internet Explorer и завершает свою работу. Эта информация объясняла тот факт, что мы не могли найти выполняющийся процесс, используя то же имя, что у загадочного файла, и показывала, что проблема связана с браузером Internet Explorer («iexplore.exe»). Эти выводы подтверждались в том числе тем, что ни в одной из систем, из которых мы собирали энергозависимые данные, браузер Internet Explorer не выполнялся на рабочем столе. Итак, мы имели дело с активным и выполняющимся процессом «iexplore.exe», передававшим огромное количество трафика в сеть и Интернет, но при этом на рабочем столе не было ни одного открытого окна браузера.

В данном конкретном деле интересен был не столько метод, используемый для внедрения кода, а тот факт, что обнаруженный загадочный файл не удалось идентифицировать многочисленными антивирусными приложениями. Вернее для меня самым интересным было то, что признаки этой проблемы были удивительно похожи на работу червя Setiri, который был представлен несколькими исследователями из компании SensePost (www.sensepost.com/research_conferences.html) на конференции BlackHat в Лас-Вегасе в 2002 году. Setiri работал посредством получения доступа к браузеру Internet Explorer как COM-сервер и создания трафика через этот браузер. Интересно отметить, что Дейв Рот (Dave Roth) написал Perl-скрипт «IEEvents.pl» (www.roth.net/perl/scripts), который, при внесении незначительных изменений, запускает браузер Internet Explorer в невидимом режиме (т. е. окно браузера не отображается на рабочем столе) и находит веб-страницы и т. п.

В чем смысл всего этого? Дело в том, что я просто хотел показать, насколько сложными бывают некоторые инциденты. Получение скрытого доступа к системе посредством загрузчика, который сам сначала попадает в систему через уязвимость веб-браузера, является хоть и эффективным, но не очень сложным способом по сравнению с внедрением кода в пространство памяти процесса.

Еще один способ, применяемый создателями шпионских и вредоносных программ, чтобы начать (и поддерживать) выполнение своего кода, – использование объектов модуля поддержки браузера (*browser helper object*, ВНО; дополнительные сведения об объектах ВНО см. в главе 4). Например, два объекта ВНО, встречающихся в системе, – это Adobe PDF Reader Link Helper и DriveLetterAccess. Их можно найти в пространстве процесса Internet Explorer, используя инструмент «listdlls.exe»:

```
C:\Program Files\Adobe\Acrobat 7.0\ActiveX\AcroIEHelper.dll
C:\WINDOWS\System32\DLA\DLASHX_W.DLL
```

Имейте в виду, что, по мере того как меняются версии программного обеспечения, так же могут меняться пути к их файлам в файловой системе. Например, если установлена программа Adobe Reader версии 8, путем к объекту Adobe PDF Reader Link Helper будет C:\Program Files\Common Files\Adobe\ActiveX\AcroIEHelper.dll.

Если кто-то нарушает безопасность системы Windows из сети, можно ожидать увидеть артефакты входа в систему (в виде записей в журнале событий безопасности или обновления последнего времени входа в систему для данного пользователя), открытые файлы в системе или даже процессы, запущенные данным пользователем. Если

злоумышленник не использует механизмы входа в систему от Microsoft (удаленный рабочий стол, команда *net use* и т. д.), а получает доступ к системе через программу скрытого удаленного администрирования, можно ожидать увидеть выполняяющиеся процессы, сетевые соединения и тому подобное.

Имея представление о характере инцидента, можно эффективно направить действия по расследованию инцидента на решение проблемы, не только с точки зрения сбора данных, но и с точки зрения сопоставления и анализа данных.

Реагирование

Вопрос, возникающий сразу после подтверждения инцидента, очень часто звучит так: «И что делать теперь?» Я не люблю этого говорить, но ваши действия действительно зависят от инфраструктуры. Например, в примерах дел, в этой главе, вы видели «инциденты», в которых проблемный процесс выполнялся от имени учетной записи администратора. Здесь это необходимая часть сценария дела, но при расследовании инцидента можно довольно часто найти процесс, выполняющийся в контексте администратора или даже системы. В таких случаях большинство специалистов считает, что больше нельзя доверять информации, поступающей из системы (т. е. нельзя доверять тому, что инструменты для сбора данных предоставляют точное представление о системе), и что единственным приемлемым решением будет начать все заново: переустановить операционную систему с незараженного (т. е. с исходного установочного) носителя и восстановить все данные из резервных копий.

Мне кажется, что такое решение сопряжено с огромным количеством проблем, особенно из-за того, что, вероятно, вам довольно скоро придется делать то же самое снова. Вероятно, вы подумали про себя: «Что?!» Я объясню. Предположим, вы находите подозрительный процесс и, используя такой инструмент, как «*pslist.exe*», видите, что этот процесс выполняется не очень долго относительно общего времени работы самой системы. Это свидетельствует о том, что процесс был запущен через некоторое время после начальной загрузки системы. Например, в то время как я сижу здесь и пишу эту главу, моя система работает уже более восьми часов. Я вижу это в крайнем правом столбце «Время работы» («Elapsed Time») в выходных данных инструмента «*pslist.exe*», как показано ниже:

smss	1024	11	3	21	168	0:00:00.062	8:28:38.109
csrss	1072	13	13	555	1776	0:00:26.203	8:28:36.546

Однако я вижу и другие процессы, запущенные значительно позже загрузки системы:

uedit32	940	8	1	88	4888	0:00:03.703	4:07:25.296
cmd	3232	8	1	32	2008	0:00:00.046	3:26:46.640

Хотя отметки времени MAC (время изменения, обращения и создания) в файлах могут быть изменены, чтобы ввести эксперта в заблуждение, данные о количестве времени, в течение которого выполняется процесс, фальсифицировать намного сложнее. Имея эту информацию, эксперт может разработать времененную шкалу для того периода, в течение которого, возможно, произошел инцидент, и определить общий масштаб инцидента (похоже на подход, использовавшийся в примерах дел ранее). Цель – определить основную причину инцидента, чтобы проблему, приведшую к нарушению безопасности, можно было исправить в этой системе, а, впоследствии, и в других системах. Если этого не сделать, возвращение компьютера с переустановленной системой в сеть, вероятно, приведет к тому, что безопасность этой системы снова будет нарушена. Если инцидент произошел из-за уязвимостей в системе, их можно исправить путем

установки исправлений. Однако если основной причиной инцидента является ненадежный пароль администратора, никакое количество исправлений не исправит эту проблему. То же можно сказать об уязвимостях в конфигурации приложения, используемых, например, сетевыми червями.

Теперь давайте рассмотрим другой случай, в котором подозрительный процесс запущен как служба, а выходные данные инструмента «pslist.exe» показывают, что этот процесс выполняется примерно столько же времени, сколько и сама система. Так как, похоже, не существует функций Windows API, позволяющих злоумышленнику изменять отметки времени последней записи (LastWrite) в разделах реестра, эксперт может извлечь эту информацию из работающей системы и определить, когда служба была установлена в систему. (Отметки времени MAC в файлах можно легко изменить, используя общедоступные API). Компетентный эксперт знает, что для того, чтобы установить службу Windows, должна использоваться учетная запись с правами администратора, поэтому проверка сведений о входе пользователей в систему и об их действиях в системе может привести к обнаружению основной причины инцидента.

Повторюсь, важно определить основную причину инцидента, чтобы можно было исправить ситуацию не только во взломанной системе, но и в других системах.

Предупреждение
В статье № Q328691 из базы знаний Microsoft (http://support.microsoft.com/default.aspx?scid=kb;en-us;328691), «MIRC Trojan-related attack detection and repair», содержится такое утверждение в разделе «Направления атаки»: «Анализ сложившейся ситуации свидетельствует о том, что злоумышленники, вероятнее всего, проникли в системы, воспользовавшись ненадежными или пустыми паролями администратора. Корпорация Майкрософт не располагает данными о том, что в ходе атак использовались ранее неизвестные уязвимости системы безопасности». Простая переустановка операционной системы и приложений, а также восстановление данных в зараженных системах приведет к тому, что безопасность систем будет нарушена снова, пока будут использоваться те же параметры конфигурации. В корпоративной среде используются совместные учетные записи администратора с легко запоминаемыми (т. е. ненадежными) паролями, и в переустановленной системе, вероятнее всего, будут использоваться те же имена и пароли учетных записей, что и до инцидента.

Определение этой основной причины иногда кажется невыполнимой задачей, однако наличие точных знаний, наборов соответствующих навыков, а также данной книги может значительно ее упростить.

Предотвращение

Для того чтобы облегчить задачу расследования инцидентов (учитывая тот факт, что специалисты по расследованию инцидентов обычно являются сотрудниками ИТ-отделов), ИТ-отделы могут, помимо переустановки операционных систем и приложений, воспользоваться руководствами по усилению защиты систем и процедурами управления конфигурацией. Например, разрешая на сервере выполнение только тех служб и процессов, которые необходимы для работы самой системы, вы уменьшаете поверхность атаки этой системы. Затем, необходимо максимально безопасно сконфигурировать те службы, выполнение которых вы разрешаете. Если у вас выполняется веб-сервер IIS, система может работать как веб-сервер, а также как FTP-сервер. Если вам не нужен выполняющийся FTP-сервер, отключите, удалите или даже не устанавливайте его вообще. Затем сконфигурируйте веб-сервер так, чтобы использовались только необходимые сопоставления сценариев (веб-серверы IIS с удаленным сопоставлением сценария ida-файла не были подвержены заражению червем Code Red в 2001 году); кроме того, можно

даже установить инструмент UrlScan (<http://technet.microsoft.com/en-us/security/cc242650.aspx>).

Совет

Инструмент UrlScan от Microsoft поддерживает IIS версии 6.0, а в блоге Nazim's IIS Security Blog (<http://blogs.iis.net/nazim/default.aspx>) говорится, что UrlScan версии 3.0 позволяет защитить веб-сервер IIS от некоторых последних атак с внедрением SQL-кода.

Такой же минималистский подход применяется к настройке учетных записей пользователей в системе. Пользователи должны иметь такой уровень доступа к системе, который им действительно необходим. Если пользователю не нужен доступ к системе, чтобы входить с консоли или удаленно из сети, он не должен иметь учетную запись в этой системе. Я расследовал несколько случаев, когда в системах оставались старые учетные записи пользователей с ненадежными паролями, а злоумышленники смогли получить доступ к данным системам посредством этих учетных записей. В другом случае данные, полученные из взломанной системы, показывали, что вход в нее был выполнен с использованием учетной записи, назначенной пользователю, который в то время летел в самолете на высоте 10 000 метров над Средним Западом. Однако пользователь редко использовал свою учетную запись для доступа к системе, и эта учетная запись оставалась без присмотра.

Если поверхность атаки системы уменьшается, злоумышленнику будет значительно труднее получить доступ к системе, чтобы нарушить безопасность данных или использовать ее как средство для проведения последующих атак. В таком случае злоумышленник может оставить в системе больше следов в виде записей журналов и сообщений об ошибках, что сделает его попытки атак более очевидными для администраторов, или просто отказаться от взлома системы, так как та уже не является легкой мишенью. В любом случае я лучше буду иметь дело с несколькими мегабайтами файлов журналов, в которых сообщается о неудачных попытках нарушения безопасности (как в случае с распространением червя Nimda; см. www.cert.org/advisories/CA-2001-26.html), чем с системой, которую постоянно взламывают из-за отсутствия надлежащих средств защиты или контроля. Если будут предприняты хотя бы эти меры по уменьшению поверхности атаки и возможностей нарушения безопасности системы, у эксперта будет больше информации, которую можно анализировать, в виде файлов журналов или других типов данных.

Краткое изложение

Поле того как вы соберете энергозависимые данные из работающей системы, необходимо выполнить анализ этих данных и провести эффективное и оперативное расследование. Эксперты могут быть часто ошеломлены одним только объемом энергозависимых данных, который им нужно обработать, и эта задача кажется им еще более сложной, если они не знают, что искать. Получив представление о характере инцидента, эксперт может начать сокращать объем данных, отыскивая и анализируя информацию о заведомо допустимых процессах, сетевых соединениях, пользователях в системе и т. д. Он может также автоматизировать некоторые процессы сопоставления данных, чем еще больше сократит объем данных и уменьшит количество возможных ошибок.

Все эти меры позволяют провести оперативное, тщательное и эффективное расследование инцидентов.

Быстрое повторение

Анализ данных

- Во время исследования работающей системы эксперты часто испытывают стресс, напряжение и замешательство. Эксперты могут использовать способы сокращения объема данных и автоматизации, чтобы провести эффективное расследование.
- После сбора и анализа данных, заключительные этапы расследования могут основываться на нетехнических факторах, таких как деловая или политическая инфраструктура организации.
- Проведение анализа основной причины при расследовании инцидента может значительно способствовать экономии времени и денег в будущем.
- Применение минималистского подхода к конфигурированию системы часто задерживает или вовсе предотвращает возникновение инцидента. При попытках взлома системы с усиленной защитой останется больше следов, а, возможно, даже будут поданы предупреждения об этих действиях.

Часто задаваемые вопросы

Вопрос: Какая разница между процессом и службой?

Ответ: С точки зрения исследования работающей системы между ними нет почти никакой разницы за исключением того, как запускается и в каком контексте пользователя выполняется каждый из них. Службы Windows, которые фактически являются процессами, могут запускаться автоматически при начальной загрузке системы. Когда процесс запускается как служба, он чаще всего выполняется с привилегиями уровня системы, тогда как процессы, запускаемые автоматически посредством пользовательского куста реестра, будут выполнять в контексте этого пользователя.

Вопрос: Время от времени я вижу записи о необычном трафике в журналах брандмауэра. Похоже, что трафик возникает в одной из систем в моей сети и передается в подозрительную систему. Когда я вижу записи о таком трафике, я перехожу к данной системе и собираю энергозависимые данные. Однако я не нахожу активных сетевых соединений или активных процессов, использующих исходный порт, указанный в трафике. Через шесть часов я снова вижу записи об этом трафике. Что мне делать?

Ответ: В файле «fruc.ini», использовавшемся в примерах дел, я применял инструмент «autorunsc.exe» с сайта Microsoft (Sysinternals), чтобы собрать информацию о местах автозапуска. Не забудьте проверить назначенные задания, а также любые необычные процессы, которые могут запускать дочерние процессы для создания этого трафика.

Вопрос: Я пытаюсь расследовать инцидент, но не нахожу никаких признаков этого инцидента в системе.

Ответ: То, что кажется «необычным» или «подозрительным» поведением в системе Windows, часто объясняется недостатком знаний о системе и не связано с фактическим инцидентом. Я встречал экспертов, которые сомневались в существовании определенных файлов и каталогов («Prefetch» и т. д.) по той простой причине, что они были плохо знакомы с системой. Припоминаю случай, когда администратор удалил все файлы с расширением .pf, которые он обнаружил в каталоге C:\Windows\Prefetch (см. главу 5). Несколько дней спустя многие из тех файлов загадочным образом вернулись, и он подумал, что безопасность системы была нарушена трояном или программой скрытого удаленного администрирования.

Глава 3

Анализ памяти Windows

Содержание этой главы:

- Сбор памяти процесса
- Создание дампа физической памяти
- Анализ дампа физической памяти

- ✓ Краткое изложение
- ✓ Быстрое повторение
- ✓ Часто задаваемые вопросы

Введение

В главе 1 мы обсуждали сбор энергозависимых данных из работающей операционной системы Windows. Согласно документу № 3227 из серии RFC (где описывается порядок изменяемости энергозависимых данных), одним из первых элементов данных, подлежащих сбору во время исследования работающей системы, является содержимое физической памяти, которую обычно называют ОЗУ. Несмотря на то, что особенности сбора отдельных частей энергозависимой памяти, таких как сетевые соединения или выполняющиеся процессы, давно известны и широко обсуждаются, тема сбора, синтаксического разбора и анализа полного содержимого физической памяти – относительно новая область исследования даже сегодня. Эта область исследования стала развиваться в последние несколько лет, начиная с лета 2005 года, по крайней мере, с общественной точки зрения.

Самый важный вопрос, на который нужно ответить на данном этапе, – «Зачем?». Зачем нужно обирать содержимое ОЗУ? Как это поможет, насколько это важно, и что можно пропустить, если не выполнить сбор и анализ содержимого физической памяти? До сих пор некоторые эксперты собирали содержимое ОЗУ в надежде найти то, что они не нашли бы на накопителе во время исследования его образа, – например, пароли. Некоторые программы приглашают пользователя ввести пароль, и, если диалоговое окно не отображается, то пароль, вероятнее всего, можно найти в памяти. Специалисты по анализу вредоносных программ просматривают память при исследовании вредоносных программам с зашифрованным или запутанным кодом, так как, когда такие программы запускаются, они расшифровываются в памяти. Вредоносные программы все чаще создаются с кодом, запутанным таким образом, что провести статический, автономный анализ будет в лучшем случае чрезвычайно трудной задачей. Однако если вредоносной программе разрешить выполнение, она будет существовать в памяти в расшифрованном состоянии, что облегчит исследование ее функций. Наконец, руткиты скрывают процессы, файлы, разделы реестра и даже сетевые соединения от инструментов, которые обычно используются для перечисления этих элементов, но посредством анализа содержимого ОЗУ мы можем узнать, какие элементы были скрыты. Мы также можем найти информацию о процессах, которые со временем завершили свою работу.

В 2008 году Грег Хоглунд (Greg Hoglund), более известный как создатель первого жизнеспособного руткита для систем Windows и веб-сайта rootkit.com, а также как исполнительный директор компании HBGary, написал небольшую статью, которая

называется «The Value of Physical Memory Analysis for Incident Response» (www.hbgary.com/resources.html). В этой статье Грэг описывает, какие данные можно извлечь из содержимого физической памяти, и, что, возможно, более важно, значение этих данных относительно решения вопросов расследования инцидентов и судебного компьютерного анализа.

Краткая история

В прошлом «анализ» дампов физической памяти заключался в поиске строк (при помощи утилиты *grep*) в «образе» файла с целью найти пароли, IP-адреса, адреса электронной почты или другие строки, которые могли дать эксперту ключ к решению стоящей перед ним задачи. Недостаток этого метода «анализа» заключается в том, что найденную информацию невозможно связать с отдельным процессом. Имеет ли найденный IP-адрес отношение к делу, или он использовался каким-нибудь другим процессом? Как быть со словом, похожим на пароль? Это пароль, который злоумышленник использовал для доступа к троянской программе в системе, или это часть переписки из программы обмена мгновенными сообщениями?

Возможность выполнить некоторый анализ дампа физической памяти довольно долгое время была одним из самых заветных желаний судебных экспертов. Другие специалисты (такие как я) признавали потребность в легкодоступных инструментах и платформах для получения дампов физической памяти и анализа их содержимого.

Летом 2005 года на конференции Digital Forensic Research Workshop (DFRWS; www.dfrws.org) была предложена задача на анализ памяти, чтобы стимулировать обсуждения, исследования и разработку инструментов в этой области. Всех желающих приглашали загрузить два файла, которые содержат дампы физической памяти (полученные при помощи модифицированного инструмента «dd.exe», доступного в дистрибутиве Helix 1.6), и ответить на вопросы, опираясь на сценарий, предоставленный на веб-сайте. Крис Бетц (Chris Betz) и команда, состоящая из Джорджа Гарнера-младшего (George M. Garner, Jr) и Роберта Яна Моры (Robert-Jan Mora), были выбраны совместными победителями этого конкурса, так как они предоставили подробные отчеты, в которых иллюстрируются их методики и показаны результаты разработанных ими инструментов. К сожалению, эти инструменты не были размещены в свободном доступе.

Через год после этого конкурса другие специалисты продолжили это исследование или провели свои собственные исследования в иных направлениях. Андреас Шустер (Andreas Schuster) начал публиковать в английском разделе своего блога (http://computer.forensikblog.de/en/topics/windows/memory_analysis) части проведенного им исследования вместе с форматом структур EProcess и EThread из различных версий ОС Windows, в том числе Windows 2000 и XP. Джо Стюарт (Joe Stewart) опубликовал Perl-скрипт «*pmodump.pl*», который входит в состав инструмента Truman Project (www.secureworks.com/research/tools/truman.html) и позволяет извлечь память, используемую процессом, из дампа ОЗУ (что важно для анализа вредоносных программ). Мариуш Бурдах (Mariusz Burdach) опубликовал материалы, касающиеся анализа памяти (сначала для систем Linux, а позднее специально для ОС Windows), в частности представил презентацию на конференции BlackHat Federal 2006. Джесси Корнблум (Jesse Kornblum) предложил несколько наработок в области анализа памяти, в том числе способ определения исходной операционной системы на основе содержимого дампа ОЗУ. Летом 2006 года Тим Видас (Tim Vidas; <http://nucia.unomaha.edu/tvidas/>), старший научный сотрудник Небрасского университета, опубликовал Perl-скрипт «*procloc.pl*», предназначенный для поиска процессов как в дампах ОЗУ, так и в файлах аварийного дампа памяти.

С тех пор область исследования, касающаяся сбора и анализа дампов памяти, развивается очень стремительно, и в некоторых случаях ведущие специалисты добились значительных результатов. Возможно, самый известный специалист в области анализа

памяти – Аарон Уолтерс (Aaron Walters), участвовавший в создании платформ FATKit (<http://4tphi.net/fatkit/>) и Volatility Framework (<https://www.volatilesystems.com/default/volatility>). Хотя эти инструменты позволяют собирать содержимое физической памяти из систем Windows XP и Vista (эта тема будет подробно рассмотрена в этой главе), первоначальной целью Аарона и его партнера, Ники Петрони-младшего (Nick L. Petroni Jr.), было предоставить платформу для анализа дампов памяти. Аарону и Нику помогали Брэндан Долан-Гавитт (Brendan Dolan-Gavitt) и другие специалисты, которые внесли значительный вклад в область анализа памяти, в частности в разработку платформы Volatility Framework. (Блог Брэндана можно найти по адресу <http://moyix.blog-spot.com/>, а страницу с его работами – по адресу www.cc.gatech.edu/~brendan/.) Маттье Свиш (Matthieu Suiche; www.msuiche.net/) разработал программу для создания содержимого физической памяти, а также опубликовал информацию и инструменты (которые были включены в состав Volatility Framework) для анализа файлов спящего режима Windows. Андреас Шустер (<http://computer.forensikblog.de/en/>) продолжил свою работу в области анализа дампов памяти Windows, выпустив в ноябре 2008 года скрипт PTFinder (рассматриваемый позднее в этой главе) для ОС Windows Vista, который включен в набор Perl-скриптов PTFinder.

В дополнение к бесплатным средствам с открытым исходным кодом для анализа дампов памяти компания Mandiant (веб-сайт – www.mandiant.com, блог – <http://blog.mandiant.com/>) выпустила программу Memoryze для сбора и анализа содержимого ОЗУ, а также инструмент Audit Viewer для лучшего представления результатов Memoryze. Кроме того, специалисты компании HBGary (www.hbgary.com) разработали свое собственное приложение для анализа памяти, которое называется Responder и выпускается в версиях Professional и Field. Веб-сайт компании HBGary содержит большое количество информации об этих инструментах, в том числе видеоролики, демонстрирующие способы их использования.

Сбор памяти процесса

Во время исследования системы вы, возможно, будете заинтересованы только в отдельных процессах, а не во всем списке процессов, и захотите получить намного больше данных, а не просто содержимое памяти процесса, доступное в файле дампа ОЗУ. Например, можно быстро идентифицировать процессы, которые не требуют дополнительного всестороннего исследования. Существует несколько способов для сбора памяти, используемой процессом, не только из физической, но и из виртуальной памяти или файла подкачки.

Для этого доступно несколько инструментов. Один из них – это «pmdump.exe» (www.ntsecurity.nu/toolbox/pmdump/), созданный Арне Видстромом (Arne Vidstrom) и доступный на сайте NTSecurity.nu. Однако, как указано на сайте NTSecurity.nu, инструмент «pmdump.exe» позволяет создавать дамп памяти процесса, не прекращая выполнение процесса. Как обсуждалось выше, это позволяет процессу продолжать работу, а содержимому памяти изменяться во время записи в файл, что приводит к созданию «размытой» памяти процесса. Кроме того, «pmdump.exe» не создает файл выходных данных, который можно проанализировать при помощи средств отладки (Debugging Tools) от Microsoft.

Тобиас Клейн (Tobias Klein) предложил еще один способ создания дампа содержимого памяти процесса в виде бесплатного инструмента (хотя и с закрытым исходным кодом), который называется Process Dumper (доступный как для версий Linux, так и для Windows по адресу www.trapkit.de/research/forensic/pd/index.html). Process Dumper («pd.exe») выводит содержимое всего пространства процесса, а также дополнительные метаданные и окружение процесса на консоль (стандартное устройство вывода), чтобы выходные данные можно было перенаправить в файл или сокет

(посредством netcat или других инструментов, некоторые из которых рассмотрены в главе 1). В документации, предоставленной Тобиасом для «pd.exe», ничего не говорится о том, что перед созданием дампа процесса выполняется его отладка, остановка или фиксация. Тобиас также предоставляет программу Memory Parser с графическим интерфейсом пользователя для анализа метаданных и содержимого памяти, собранных Process Dumper. Похоже, что эти инструменты являются приложением к работе Тобиаса Клейна по извлечению закрытых ключей и сертификатов алгоритма RSA из памяти процесса (www.trapkit.de/research/sslkeyfinder/index.html).

Совет

Джефф Брайнер (Jeff Bryner) разработал инструмент pdgmail (доступный по адресу www.jeffbryner.com/code/pdgmail) на основе языка Python, использующий выходные данные программы Process Dumper для поиска артефактов Gmail (например, контакты, записи о последнем доступе, имена учетных записей и т. д.). Джефф опубликовал статью о pdgmail в блоге SANS Forensic (<http://sansforensics.wordpress.com>), где указал, что не тестировал этот инструмент на платформе Windows. Впоследствии Джефф также выпустил инструмент pdymail (<http://jeffbryner.com/code/pdymail>) для извлечения почты Yahoo! из памяти процесса. В данном разделе этой главы мы рассмотрим создание дампа содержимого памяти процесса, а позднее поговорим о том, как эксперт может создавать дамп содержимого памяти процесса из полного дампа физической памяти. Можно применить любой из этих способов, чтобы получить данные для использования в pdgmail или pdymail.

Еще один инструмент, рекомендуемый во многих источниках, – «userdump.exe» от Microsoft. Он позволяет создавать дамп любого процесса во время его работы без подключения отладчика и остановки процесса после того, как создание дампа будет завершено. Кроме того, файл дампа, созданный посредством «userdump.exe», можно просмотреть при помощи средств отладки от Microsoft. Однако для работы «userdump.exe» необходимо установить драйвер, а, в зависимости от обстоятельств, такой вариант может вам не подойти.

Исходя из разговоров с Робертом Хенсингом (Robert Hensing), бывшим сотрудником группы безопасности из службы технической поддержки Microsoft, предпочтительный способ создания дампа памяти, используемой процессом, – применить скрипт «adplus.vbs», входящий в состав средств отладки, так как при этом отладчик подключается к процессу, а процесс приостанавливается на время создания дампа. Скрипт *adplus* (сокращенно от *Autodumpplus*) вначале был создан Робертом (в документации для этого скрипта говорится, что он написал версии 1–5, а, начиная с версии 6, его работу продолжил Израэль Бурман (Israel Burman)). В файле справки («debugger.chm») для средств отладки от Microsoft содержится много информации об этом скрипте, а также об инструменте отладки «cdb.exe», который скрипт использует для создания дампа необходимых процессов. Для работы средств отладки от Microsoft не требуется установка дополнительного драйвера, и их можно запустить с компакт-диска. Это значит, что инструменты («adplus.vbs» и «cdb.exe», а также вспомогательные библиотеки динамической компоновки, или DLL-файлы) можно записать на компакт-диск и использовать для создания дампа процесса и его сохранения на общий накопитель или запоминающее устройство, подключенное по USB. (Скрипт «adplus.vbs» использует компонент Windows Scripting Host (WSH) версии 5.6, также известный как «cscript.exe» и установленный в большинстве систем.) После того как файлы дампов будут созданы, для их анализа можно использовать бесплатные средства отладки от Microsoft. Кроме того, можно использовать другие инструменты, например BinText, чтобы извлечь строки ASCII, Unicode и строки ресурсов из дампа памяти. Более того, после создания дампа процесса можно также использовать другие инструменты (например, те, что рассматривались в

главе 1), чтобы быстро собрать дополнительную информацию о самом процессе из работающей системы. Инструмент «handle.exe» (доступный на странице <http://technet.microsoft.com/en-us/sysinternals/bb896655.aspx> и требующий для запуска наличия прав администратора) предоставит список дескрипторов (для файлов, каталогов и т. д.), открытых процессом, а «listdlls.exe» (версия 2.25 на момент написания этой книги доступна на странице <http://technet.microsoft.com/en-us/sysinternals/bb896656.aspx>) покажет полный путь к различным модулям, загруженных процессом, и номера версий этих модулей.

Подробные сведения об использовании скрипта «adplus.vbs» доступны не только в файле справки для средств отладки от Microsoft, но и в статье № 286350 из базы знаний Microsoft (<http://support.microsoft.com/kb/286350>). Скрипт «adplus.vbs» можно использовать, чтобы приостановить процесс на время создания его дампа (т. е. остановить его, создать дамп, а затем возобновить) или аварийно завершить процесс (остановить его, создать дамп, а затем завершить). Чтобы применить к процессу скрипт «adplus.vbs» в режиме «зависания», следует использовать следующую команду:

```
D:\debug>cscript adplus.vbs -quiet -hang -p <идентификатор процесса>
```

Эта команда создаст ряд файлов в каталоге «debug» в подкаталоге, имя которого начинается со строки *Hang_mode_* и содержит дату и время дампа. (Место сохранения выходных данных можно изменить, используя переключатель *-o*.) Вы увидите файл отчета скрипта «adplus.vbs», файл дампа для процесса (можно указать несколько процессов, используя несколько элементов с переключателем *-p*), список процессов (создаваемый по умолчанию при помощи «tlist.exe»; чтобы отключить эту возможность, используйте переключатель *-noTList*) и текстовый файл, в котором показаны все загруженные модули (DLL-файлы), используемые процессом.

Хотя вся информация, собранная о процессах при помощи «adplus.vbs», может быть чрезвычайно полезной во время расследования инцидента, этот инструмент можно применять только к тем процессам, которые доступны через интерфейс API. Если процесс недоступен (например, скрыт руткитом), этот инструмент нельзя будет использовать для сбора информации о процессе.

Можно использовать команду *volatility memdump* (мы рассмотрим платформу Volatility Framework позднее в этой главе), чтобы создать дамп адресуемой памяти для процесса из дампа памяти ОС Windows XP, как показано ниже:

```
D:\Volatility>python volatility memdump -f d:\hacking\xp-laptop1.img -p 4012
```

В результате этой команды создан файл «4012.dmp» размером 118 300 672 байт.

Создание дампа физической памяти

Итак, как получить полное содержимое физической памяти, а не только дамп процесса? Для этого существует несколько способов, каждый из которых имеет свои преимущества и недостатки. Цель этой главы – дать представление о различных доступных способах, а также о технических аспектах, связанных с каждым из способов. Таким образом, специалист по расследованию инцидентов или эксперт сможет сделать обоснованный выбор относительно того, какой из способов наиболее приемлемый, учитывая деловые потребности клиента (потерпевшего), а также проблемы, связанные с инфраструктурой.

DD

DD ([http://en.wikipedia.org/wiki/Dd_\(Unix\)](http://en.wikipedia.org/wiki/Dd_(Unix))) – короткое имя мощного инструмента, возникшего из мира UNIX и имеющего множество важных функций, в том числе

копирование файлов или даже всего содержимого накопителей. DD давно считается стандартом для создания судебных образов, и большинство основных инструментов для создания судебных образов или клонирования данных поддерживает формат dd. Компания GMG Systems создала модифицированную версию инструмента dd, который работает в системах Windows и который можно использовать для создания дампа содержимого физической памяти из ОС Windows 2000 и XP. Эта версия dd входила в состав пакета Forensic Acquisition Utilities, но больше недоступна. Данный инструмент мог собирать содержимое физической памяти посредством получения доступа к объекту *\Device\PhysicalMemory* из пользовательского режима. Можно было использовать следующую команду, чтобы сохранить содержимое ОЗУ в файле «ram.img» на сетевом ресурсе или USB флеш-накопителе, подключенному к компьютеру:

```
D:\tools>dd if=\\.\\PhysicalMemory of=F:\\ram.img bs=4096 conv=noerror
```

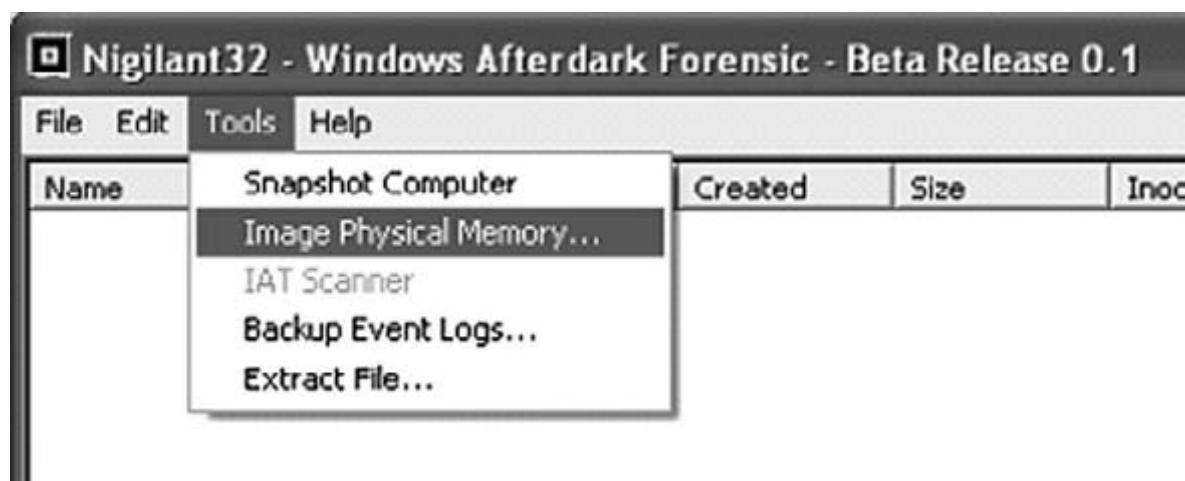
В упомянутой выше команде размер блока (значение *bs*) указан как 4 Кб (4096 байт), так как такой размер имеют по умолчанию страницы в памяти. Следовательно, эта команда указывает инструменту «dd.exe» захватывать одну страницу за раз. Эта версия «dd.exe» также позволяет сжимать данные и создавать криптографические хэши (контрольные суммы) для содержимого. Однако, из-за изменчивости ОЗУ (т. е. физическая память продолжает изменяться в то время, как создается ее дамп) не рекомендуется создавать контрольные суммы, пока память не запишется на накопитель, просто потому, что от этого нет никакой пользы. Если пользователь создаст образ памяти дважды, даже с минимальной задержкой, содержимое ОЗУ и соответственно последующие хэш значения будут другими. В данном случае имеет смысл создавать криптографический хэш только для обеспечения целостности собранного дампа всей памяти.

Предупреждение

Обсуждавшаяся выше версия «dd.exe», созданная Джорджем Гарнером-младшим, больше недоступна и не поддерживается, но, возможно, есть эксперты (например, я), у которых до сих пор осталась копия этого инструмента где-нибудь на компакт-диске или НЖМД. Этот инструмент рассматривается здесь для того, чтобы вы имели представление обо всех существующих возможностях и чтобы воздать должное Джорджу за его вклад в области клонирования и анализа памяти.

Nigilant32

Другие инструменты для сбора содержимого ОЗУ используют процедуры, подобные «dd.exe». Программа Nigilant32 (www.agilerm.net/publications_4.html), разработанная Мэттом Шенном (Matt Shannon) из компании Agile Risk Management, имеет графический интерфейс и позволяет эксперту клонировать содержимое физической памяти. На илл. 3.1., показана часть графического интерфейса Nigilant32 с параметрами для создания образа физической памяти.

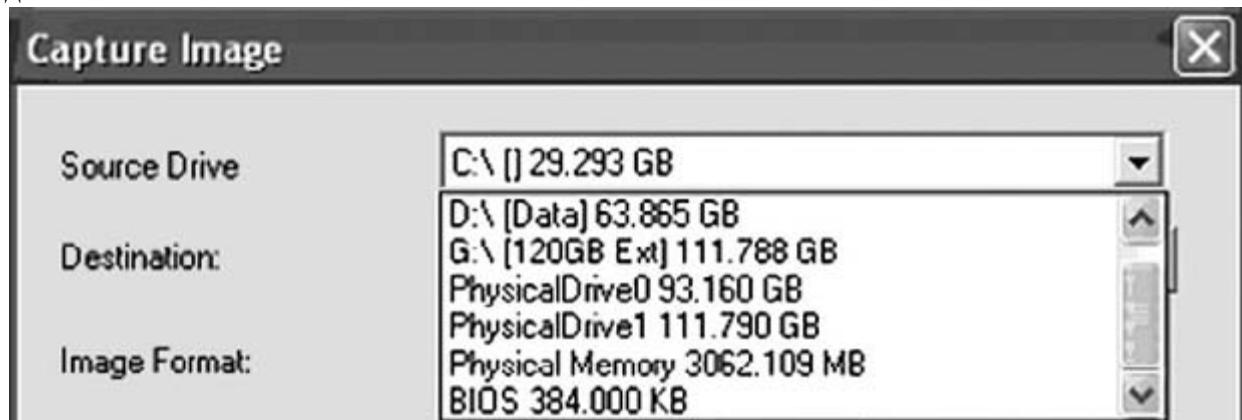


Илл. 3.1. Часть графического интерфейса программы Nigilant32.

Благодаря графическому интерфейсу программа Nigilant32 может быть более удобной в работе для некоторых экспертов. Можно записать программу Nigilant32 на компакт диск или флеш накопителе, вместе с другими инструментами и использовать ее для быстрого сбора содержимого физической памяти (главным образом из ОС Windows XP). Как и в случае с «dd.exe» и другими инструментами, Nigilant32 необходим локальный доступ к системе, из которой эксперт хочет получить дамп памяти. Кроме того, Nigilant32 можно развернуть до того, как произойдет фактический инцидент, а затем эксперты будут иметь возможность запустить эту программу локально или получить к ней доступ удаленно (используя такие приложения, как VNC или Remote Desktop Client), чтобы создать дамп памяти и сохранить его на флеш накопителе или общем сетевом ресурсе.

ProDiscover

Программа ProDiscover IR (версия 4.8 использовалась при написании этой книги, а версия 5.0 была выпущена летом 2008 года) также позволяет эксперту собирать содержимое физической памяти (и BIOS) через удаленное серверное приложение (сервлет), которое можно поместить в систему посредством съемного носителя (компакт-диск, флеш-накопитель и т. д.) или по сети. На илл. 3.2., показан интерфейс пользователя для этой возможности.



Илл. 3.2. Фрагмент диалогового окна «Захват образа» (“Capture Image”) из программы ProDiscover IR.

KnTDD

В то же время проблема использования таких инструментов, как «dd.exe» или Nigilant32, заключается в том, что, начиная с ОС Windows 2003 с пакетом обновления 1 (SP1), доступ к объекту `\Device\PhysicalMemory` запрещен из пользовательского режима (<http://technet.microsoft.com/en-us/library/cc787565.aspx>). То есть доступ к этому объекту

разрешен только драйверам, работающим в режиме ядра. Поэтому такие инструменты, как «dd.exe», Nigilant32 и ProDiscover (как упоминалось выше, в состав версии Incident Response программы ProDiscover входит сервлет, позволяющий эксперту получить доступ к физической памяти в удаленной системе Windows XP) не смогут собрать содержимое физической памяти из ОС Windows 2003 SP1 и более поздних версий, в том числе Vista. Чтобы решить эту проблему Джордж Гарнер-младший (из компании GMG Systems) создал новое приложение, которое называется KnTDD и входит в состав пакета инструментов KnTTools. Согласно условиям лицензии для пакета KnTTools, эти инструменты доступны для продажи по частному соглашению сотрудникам правоохранительных органов и специалистам в области информационной безопасности. KnTDD имеет следующие возможности:

- клонирование содержимого физической памяти различными способами, в том числе посредством доступа к объекту *|Device\PhysicalMemory*;
- работа в операционных системах корпорации Microsoft, с Windows 2000 по Vista, в том числе в версиях операционных систем для 64-битной аппаратной платформы, разработанной компанией AMD;
- преобразование образа необработанных данных памяти в формат дампа аварийной памяти Microsoft, чтобы полученные данные можно было проанализировать при помощи средств отладки от Microsoft;
- сохранение содержимого памяти на локальном съемном (USB-, FireWire-) накопителе или передача этого содержимого по сети, используя протокол TCP/IP;
- ведение журнала аудита и криптографические проверки целостности данных (специально для судебных экспертов).

Пакет KnTTools версии Enterprise Edition имеет следующие возможности:

- групповое шифрование выходных данных, используя сертификаты стандарта X.509, алгоритм aES-256 (по умолчанию) и понижение алгоритма до 3DES в ОС Windows 2000;
- клонирование памяти при помощи службы KnTDDSvc;
- удаленный модуль развертывания, позволяющий развернуть службу KnTDDSvc посредством передачи ее на удаленный общий ресурс ADMIN\$ или извлечения ее из веб-сервера по протоколу SSL с криптографической проверкой исполняемых файлов перед их выполнением.

Одной из особенностей использования «dd.exe» и других подобных инструментов является то, что необходимо учитывать принцип обмена Локара. Чтобы использовать эти инструменты для сбора содержимого ОЗУ, они должны быть загружены в физическую память как выполняющиеся процессы. Это означает, что они будут занимать пространство памяти, поэтому страницы памяти, используемые другими процессами, могут быть записаны в файл подкачки.

Еще одна особенность, которую нужно учитывать при использовании этих инструментов, заключается в том, что они не фиксируют состояние системы, как, например, при создании аварийного дампа памяти. Это означает, что при считывании содержимого ОЗУ это содержимое может изменяться; например, пока инструмент считывает тринадцатую страницу памяти, одиннадцатая страница может измениться, так как процесс, использующий эту страницу, продолжает выполняться. Количество времени, которое в конечном итоге требуется, чтобы закончить создание дампа, зависит от таких факторов, как частота процессора и скорости шины и ввода-вывода накопителя. Поэтому возникает вопрос: могут ли эти изменения, которые происходят в ограниченный период времени, повлиять на результаты анализа? В большинстве случаев, при расследовании инцидентов, такие инструменты, как «dd.exe», возможно, являются лучшим способом для получения содержимого физической памяти, особенно из ОС Windows 2000 и XP. Такие инструменты не требуют, чтобы работа системы была завершена, и не имеют ограничений относительно места сохранения содержимого физической памяти (например, используя

netcat, можно передать содержимое ОЗУ через сокет в другую систему, а не записывать его на локальный НЖМД). Позднее в этой главе будут рассмотрены бесплатные инструменты для анализа содержимого этих дампов ОЗУ и извлечения из них информации о процессах, сетевых соединениях и т. д. Более того, развитие пакета KnTTools и других подобных инструментов делает возможным поддержку этой методики для ОС Windows 2003 SP1 и более поздних версий.

Примечание

Главная проблема при использовании таких инструментов, как Forensic Acquisition Utilities или KnTTools, состоит в том, что система продолжает работать во время сбора содержимого памяти. Это значит, что, помимо того что страницы памяти используются просто во время выполнения инструментов (т. е. исполняемые образычитываются и загружаются в память), страницы, которые уже были считаны, могут измениться. То есть состояние системы не фиксируется на время, как в случае с созданием судебного образа НЖМД при применении традиционных судебных методик. Однако взаимодействие с работающей системой, возможно, является единственным способом, посредством которого можно получить определенную информацию. Не забывая о принципе обмена Локара, эксперты должны тщательно и понятно документировать свои действия во время исследования работающей системы. Такие вопросы, как необходимость проводить исследование работающей системы, а также «следы», оставляемые инструментами для исследования работающей системы, некоторое время были (и, вероятно, еще будут) предметом обсуждения.

MDD

В начале 2008 года было выпущено несколько новых инструментов для сбора содержимого физической памяти из систем Windows, в частности их ОС Windows 2003 SP1 и Vista. К счастью, эти инструменты одинаково хорошо работают в системах Windows XP, что делает их более универсальными, чем ранее доступные инструменты (например, «dd.exe», Nigilant32 и т. д.). Возможно, самым известным из них был инструмент «mdd.exe», выпущенный корпорацией ManTech International (www.mantech.com/msma/MDD.asp). «Mdd.exe» – простой инструмент с интерфейсом командной строки, позволяющий эксперту создавать дамп содержимого физической памяти, используя следующую команду:

```
E:\response>mdd.exe -o F:\system1\memory.dmp
```

Упомянутая выше команда показывает пример выполнения «mdd.exe» с компакт-диска (E:\) и отправки файла выходных данных (при помощи переключателя *-o*) на внешний USB-накопитель (F:\). Можно также запустить «mdd.exe» из пакетного файла, как описывалось в главе 1, используя такую команду:

```
mdd.exe -o %1\mdd-o.img
```

Кроме того, можно использовать доступные переменные из работающей среды Windows, чтобы отделить собранные энергозависимые данные, добавляя к имени дампа имя системы, из которой выполняется сбор данных:

```
mdd.exe -o %1\%ComputerName%-mdd-o.img
```

После того как «mdd.exe» завершит создание дампа памяти, будет показана контрольная сумма MD5 для полученного файла дампа:

```
took 108 seconds to write
MD5 is: 6fe975ee3ab878211d3be3279e948649
```

Эксперт может сохранить эту информацию и использовать ее для обеспечения целостности файла дампа памяти позднее.

Помимо того, что инструмент «mdd.exe» прост в использовании и развертывании, его выходные данные представляют собой то, что называется необработанным файлом дампа памяти, подобным дампу, получаемому при помощи других инструментов («dd.exe», VMware и т. д.). Однако прежде чем использовать этот инструмент, необходимо знать, что, согласно информации на веб-сайте ManTech International, «mdd.exe» не может собирать больше 4 Гб ОЗУ. Некоторые пользователи сообщали о сбоях в системах, имеющих 8 и более Гб ОЗУ, поэтому учитывайте эту особенность при принятии решения о сборе содержимого памяти из системы.

Совет

Если в рамках расследования инцидента вы собираетесь собрать содержимое ОЗУ из работающей системы, используя пакетный файл, рекомендую в первую очередь выполнить сбор содержимого ОЗУ. Таким образом, вы получите максимально «чистый» дамп, особенно если вы включили в состав пакетного файла другие сторонние утилиты («tlist.exe», «tcpvcon.exe»), а также встроенные инструменты Windows («netstat.exe»).

Win32dd

Маттье Свиш выпустил свой собственный инструмент для создания дампа содержимого физической памяти, который называется win32dd (<http://win32dd.msuiche.net/>). Win32dd описывается как «бесплатный инструмент режима ядра с полностью открытым исходным кодом»; то есть, как и «mdd.exe», «win32dd.exe» – это бесплатный инструмент, но в отличие от продукта корпорации ManTech, распространяется с открытым кодом. «Win32dd.exe» имеет несколько дополнительных возможностей, одна из которых – создание дампа памяти, совместимого с WinDbg (подобно аварийному дампу Windows), который затем можно проанализировать при помощи средств отладки от Microsoft (www.microsoft.com/whdc/DevTools/Debugging/default.mspx). Как и в случае с «mdd.exe» и другими инструментами командной строки, «win32dd.exe» можно включить в состав пакетного файла.

На момент написания этой главы была доступна версия 1.2.1.20090106 инструмента «win32dd.exe». Сведения о синтаксисе для «win32dd.exe» можно просмотреть, используя следующую команду:

```
D:\tools\win32dd>win32dd -h
Win32dd -v1.2.1.20090106 -Kernel land physical memory acquisition
Copyright (c) 2007 -2009, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2008 -2009, MoonSols <http://www.moonsols.com>
Usage:
  win32dd [option] [output path]
Option:
  -r Create a raw memory dump/snapshot. (default)
  -l Level for the mapping (with -r option only).
    l 0 Open \\Device\\PhysicalMemory device (default).
    l 1 Use Kernel API MmMapIoSpace()
  -d Create a Microsoft full memory dump file (WinDbg compliant).
  -t Type of MSFT dump file (with -d option only).
    t 0 Original MmPhysicalMemoryBlock, like BSOD. (default).
    t 1 MmPhysicalMemoryBlock (with PFN 0).
  -h Display this help.
Sample:
```

```
Usage: win32dd -d physmem.dmp
Usage: win32dd -l 1 -r C:\dump\physmem.bin
```

Как видите, в «win32dd.exe» есть несколько параметров для создания дампа содержимого физической памяти. Чтобы создать дамп необработанных данных памяти (как в dd), можно использовать следующую команду:

```
D:\tools\win32dd>win32dd -l 0 -r d:\tools\memtest\win32dd-10-xp.img
```

Включение параметров командной строки (например, *-l 0*) в имя файла выходных данных используется в качестве средства дополнительного документирования для эксперта, чтобы идентифицировать используемые переключатели команды. Как и в случае с другими инструментами командной строки, такую команду можно легко и просто добавить в пакетный файл (что также является способом самодокументирования).

Memoryze

Консалтинговая компания Mandiant выпустила свой собственный инструмент для сбора и анализа содержимого памяти, который называется Memoryze (www.mandiant.com/software/memoryze.htm). Программа Memoryze была разработана на основе продукта Mandiant Intelligent Response (MIR), и ее можно бесплатно загрузить с веб-сайта компании Mandiant, где также предоставлены примеры ее использования. После загрузки файла установщика (MSI-файла) и установки программы можно запустить Memoryze для сбора дампа памяти посредством пакетного файла «memorydd.bat», используя следующую команду:

```
D:\Mandiant\memorydd.bat
```

В результате будет создан дамп памяти в структуре каталогов внутри текущего каталога; при выполнении этой команды в моей системе была создана структура каталогов Audits\WINTERMUTE\20090103003442\, а в последний каталог команда записала несколько XML-файлов и файл-образ памяти. Выполняя командный файл с переключателем *-output*, можно указать другое место для сохранения файла дампа.

Кроме того, при выполнении в работающей системе программы Memoryze, по имеющимся данным, включает содержимое памяти из файла подкачки в процесс сбора данных. Это позволяет выполнить более полный сбор данных, так как содержимое памяти, которое было перемещено в файл подкачки, теперь доступно для анализа. В дополнение к этому, программа Memoryze версии 1.3.0 (выщенная 9 февраля 2009 года) имеет возможность создавать дамп памяти, доступной посредством программы F-Response, которая будет рассмотрена позднее в этой главе.

Winen

Компания Guidance Software также выпустила свой инструмент для сбора содержимого физической памяти, который называется «winen.exe». Хотя, как и некоторые другие инструменты, «winen.exe» является инструментом командной строки, он создает дамп памяти не в формате необработанных данных (как dd), а в патентованном формате образов EWF (Expert Witness Format), используемом программой клонирования данных EnCase. Большинство существующих инструментов для анализа данных требуют, чтобы дамп памяти был в формате dd, поэтому дампы памяти, созданные при помощи «winen.exe», необходимо перед анализом преобразовать в формат необработанных данных. Как указывает Ланс Мюллер (Lance Mueller) в своем блоге (www.forensickb.com/2008/06/new-version-of-encase-includes-stand.html), «winen.exe» можно запустить, просто указав различные параметры в консоли (т. е. ввести **winen** в

командной строке, а затем дать ответы на запросы) или предоставив посредством переключателя `-f` путь к файлу конфигурации с необходимой информацией. Для выполнения «winen.exe» необходимо в командной строке или файле конфигурации указать шесть параметров: путь к файлу выходных данных, степень сжатия, имя эксперта, имя исследуемого объекта, номер дела и номер исследуемого объекта. После загрузки и установки EnCase можно найти пример файла конфигурации в исходном каталоге программы.

В своем блоге ForensicZone (<http://forensiczone.blogspot.com/2008/06/winenexe-ram-imaging-tool-included-in.html>) Ричард МакКуон (Richard McQuown) предоставляет дополнительные сведения об использовании «winen.exe», а также ссылку на руководство пользователя для этого инструмента (чтобы получить PDF-документ по адресу <https://support.guidancesoftware.com/forum/downloads.php?do=file&id=478>, нужно зарегистрироваться на форуме EnCase).

Чтобы затем преобразовать полученный дамп памяти в формат необработанных данных, нужно открыть этот дамп в программе FTK Imager и выбрать пункт «**Создать образ накопителя**» (“Create Disk Image”) в меню «Файл» (“File”) (<http://windowsir.blogspot.com/2008/06/memory-collection-and-analysis-part-ii.html>) или открыть этот дамп памяти в программе EnCase и выбрать пункт «**Копировать/восстановить**» (“Copy/Unerase”) в меню EnCase.

Компания Guidance Software также предоставляет версию этого инструмента для 64-разрядных версий ОС Windows – «winen64.exe». Как и 32-разрядная версия инструмента, «winen64.exe» позволяет эксперту создавать дамп содержимого физической памяти в формате EWF.

Fastdump

Консалтинговая компания HBGary выпустила свой инструмент для сбора содержимого физической памяти, который называется fastdump (www.hbgary.com/download_fastdump.html). Хотя «fastdump.exe» (на момент написания этой книги доступна версия 1.2) является бесплатным инструментом, он, как и Nigilant32 и некоторые другие доступные инструменты, не может собирать содержимое памяти из ОС Windows 2003 SP1 и более поздних версий, в том числе Vista. Можно использовать этот инструмент только для сбора содержимого ОЗУ из систем Windows XP и Windows 2003 (без установленных пакетов обновлений).

Чтобы решить эту проблему, HBGary выпустила коммерческую версию инструмента, которая называется FastDump Pro («fdpro.exe») и доступна на той же веб-странице, что и «fastdump.exe», но за отдельную плату. Коммерческая версия поддерживает все, в том числе 32- и 64-разрядные, версии ОС Windows и по имеющимся данным может также создавать дамп памяти из систем, имеющих более 4 Гб ОЗУ. Кроме того, FastDump Pro обладает некоторыми дополнительными особенностями и преимуществами по сравнению с другими инструментами. Введите **fdpro** в командную строку, чтобы просмотреть сведения о синтаксисе для этого инструмента:

```
D:\HBGary\bin\FastDump>fdpro
-- FDPro v1.3.0.377 by HBGary, Inc --
***** Usage Help *****
General Usage: fdpro output dumpfile path [options] [modifiers]
To dump physical memory only to literal .bin format:
    fdpro mymemdump.bin [options] [modifiers]
To dump physical memory to an .hpak formatted file:
    fdpro mysysdump.hpak [options] [modifiers]
*** Valid .bin [options] Are: ***
-probe [all|smart|pid|help]      Pre-Dump Memory Probing
*** Valid .bin [modifiers] Are: ***
-nodriver                         Use old-style memory acquisition (XP/2k only)
```

-driver	Force driver based memory acquisition
*** Valid .hpak [options] Are: ***	
-probe [all smart pid help]	Pre-Dump Memory Probing
-hpak [list extract]	HPAK archive management
*** Valid .hpak [modifiers] Are: ***	
-nodriver	Use old-style memory acquisition (XP/2k only)
-driver	Force driver based memory acquisition
-compress	Create archive compressed
-nocompress	Create archive uncompressed

Как видите, «fdpro.exe» может по существу создавать дамп содержимого физической памяти в обычном формате необработанных данных двумя способами: используя переключатель *-nodriver*, чтобы выполнить традиционное клонирование памяти, или используя драйвер, чтобы получить доступ к физической памяти в тех версиях ОС Windows, где это необходимо (т. е. в Windows 2003 SP1, Vista и более поздних версиях).

«Fdpro.exe» также создает дамп в формате .hpak, который описан ниже:

«HPAK – патентованный формат компании HBGary, имеющий несколько отличительных особенностей, в том числе возможность сохранять содержимое ОЗУ и файла подкачки в одном архиве. Формат HPACK также поддерживает сжатие, используя формат gzip. Эта возможность полезна в случаях, когда свободное пространство на устройстве или накопителе, используемом для сбора данных, ограничено».

Один из недостатков большинства доступных инструментов для создания дампа ОЗУ состоит в том, что они позволяют получить доступ только к физической памяти и не включают в дамп файл подкачки. Инструмент «fdpro.exe», помимо того что обеспечивает доступ к различным (в том числе 64-разрядным) версиям ОС Windows и объемам памяти (т. е. более 4 ГБ), включает (как и программа Memoguze от компании Mandiant) файл подкачки в процесс сбора данных, что также позволяет впоследствии использовать это содержимое в процессе анализа (мы рассмотрим программу Responder от HBGary позднее в этой главе). В этой главе вы также узнаете, что содержимое памяти, которое было перемещено в файл подкачки, может содержать информацию, имеющую отношения к вашему исследованию и анализу.

Предупреждение

Следует отметить, что использование патентованных форматов во время сбора данных часто приводит к необходимости использовать отдельный инструмент для анализа данных. Дампы памяти, созданные, например, при помощи «winen.exe», необходимо преобразовать в формат необработанных данных, прежде чем выполнять анализ с использованием других инструментов (хотя для проведения ограниченного анализа файлов дампа доступны скрипты EnScript). То же верно в отношении патентованного формата .hpak; для анализа дампов памяти в этом формате можно использовать только программу Responder от компании HBGary. Следует учитывать эту особенность при выборе инструмента.

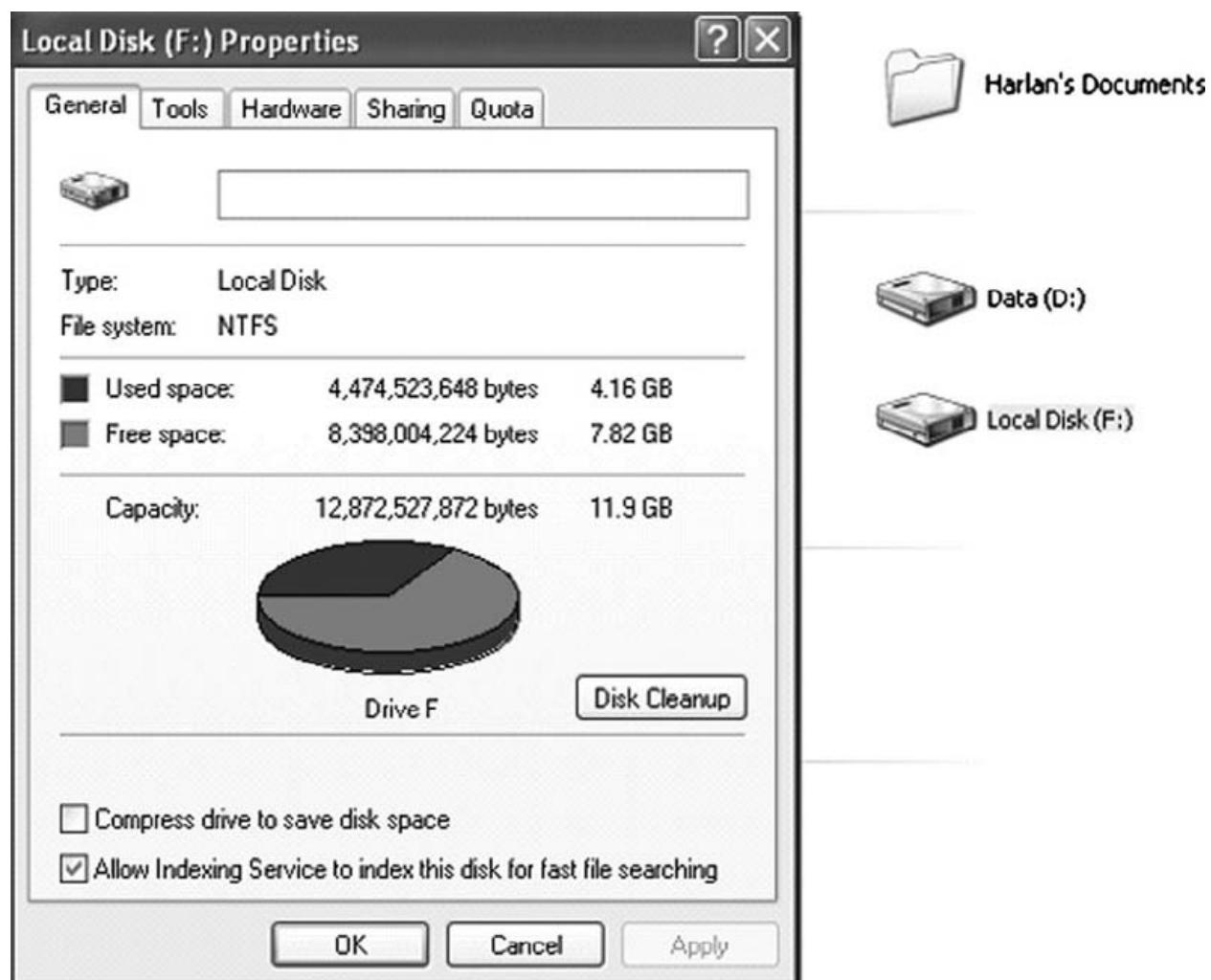
F-Response

Наконец, программа F-Response, выпущенная Мэттом Шенноном в 2008 году, ознаменовала начало новой эры в расследовании инцидентов. F-Response – платформа, которая не зависит от инструментов клонирования и использует протокол iSCSI для предоставления прямого доступа к накопителям по сети. F-Response разработана таким образом, что доступ предоставляется в режиме только для чтения, операции записи

помещаются в буфер и удаляются без предупреждения. Три версии F-Response (Field Kit, Consultant и Enterprise) развертываются по-разному, но каждая из них позволит вам «увидеть» накопители в удаленных системах (находящихся, например, в другой комнате, на другом этаже или даже в другом здании) как монтированные накопители в вашей собственной системе. F-Response не зависит от инструментов клонирования в том смысле, что, подключившись к удаленной системе и «увидев» ее накопитель, можно использовать любой инструмент по своему усмотрению («dd.exe», ProDiscover, FTK Imager и т. д.), чтобы создать образ этого накопителя. F-Response можно развертывать в системах Mac OS X, Linux и Windows, а в ОС Windows программа имеет дополнительное преимущество в виде предоставления удаленного доступа в режиме только для чтения к физической памяти. На саммите SANS Forensic Summit, в октябре 2008 года, во время совместного выступления с Аароном Уолтерсом, Мэтт объявил о выходе F-Response версии 2.03, которая умеет предоставлять удаленный доступ в реальном времени и только для чтения к физической памяти всех версий Windows, в том числе XP и Vista. Установив соединение с удаленной системой, эксперт может затем использовать такие инструменты, как «dd.exe» или FTK Imager, чтобы создать копию физической памяти.

Возможность программы F-Response (в этом примере использовалась версия 2.06 beta) получать копию физической памяти из удаленной системы Windows имеет огромное значение для расследования инцидентов, особенно при развертывании версии Enterprise Edition (дополнительные сведения об использовании этой возможности в среде предприятия см. во вставке «F-Response Enterprise Edition»). В блоге F-Response (www.f-response.com/index.php?option=com_content&task=blogsection&id=4&Itemid=9) Мэтт разместил несколько видеороликов, в которых демонстрируются способы использования и особенности F-Response, например, способ клонирования отдельных типов данных из работающего сервера Microsoft Exchange Server.

На саммите SANS Forensic Summit в октябре 2008 года Мэтт заявил, что F-Response будет обеспечивать удаленный доступ только для чтения к накопителям и физической памяти систем Windows. После установления успешного соединения удаленные накопители отображаются в системе эксперта с обычными значками накопителя, как показано на илл. 3.3.



Илл. 3.3. Накопитель удаленной ОС Windows, подключенный как диск F:\.

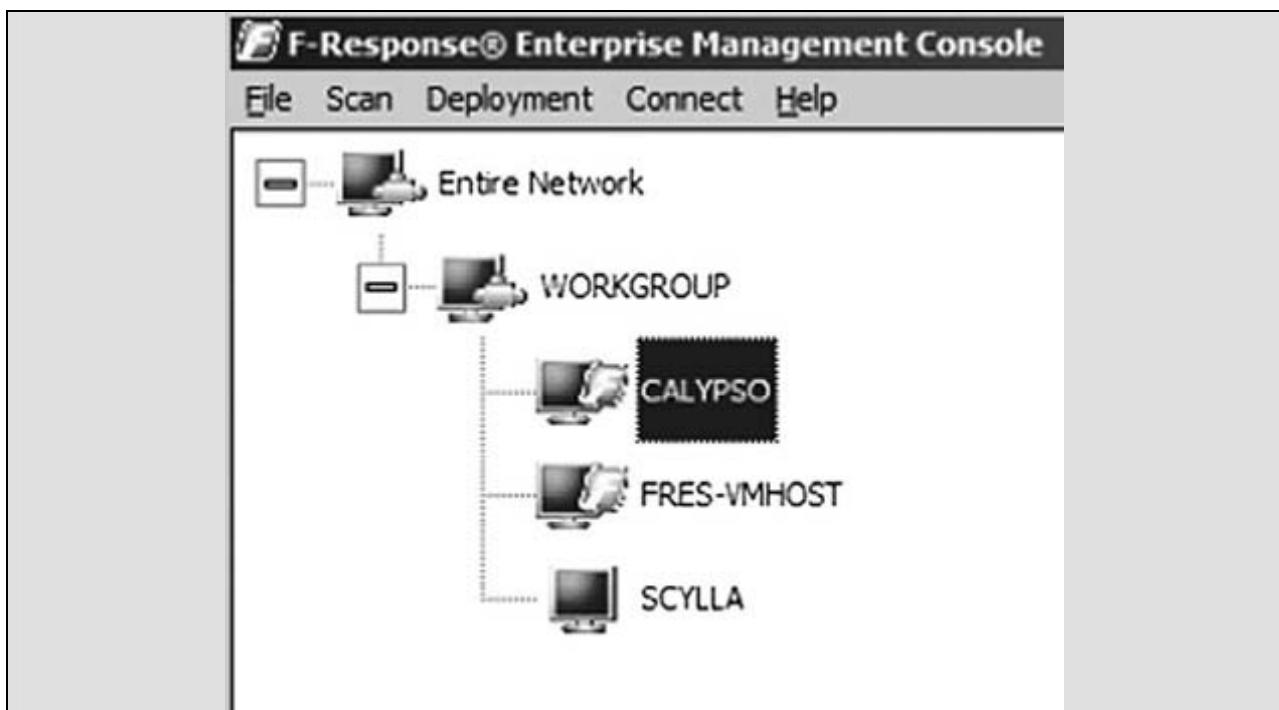
Инструменты и ловушки...

F-Response Enterprise Edition

Агенты программы F-Response Enterprise Edition (EE) можно развернуть заблаговременно. Разворачивание агентов EE до возникновения инцидента может значительно ускорить его расследование, так как агенты устанавливаются как службы Windows, но по умолчанию не запускаются автоматически при начальной загрузке системы.

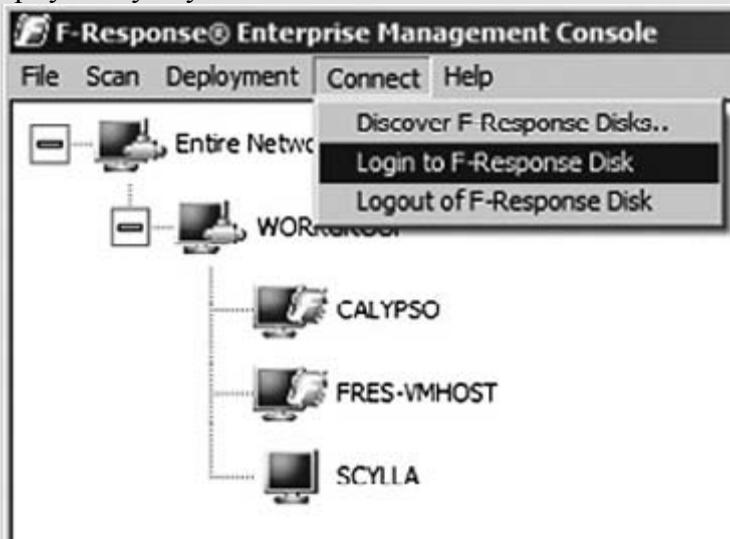
Агент можно также установить скрыто, используя такие инструменты как «*psexec.exe*» (<http://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>) или «*xcmd.exe*» (<http://feldkir.ch/xcmd.htm>). В PDF-документе «*Remote (Stealth) Deployment of F-Response EE on Windows Systems*», расположенном в каталоге «ch3» на носителе, который идет в комплекте с этой книгой, понятно описаны этапы, необходимые для развертывания F-Response EE удаленно (в скрытом режиме) по сети.

Весной 2009 года Мэтт Шенон выпустил программу F-Response Enterprise Management Console (FEMC). FEMC – программа с графическим интерфейсом пользователя, которая устраняет все сложности, связанные с установкой Enterprise Edition вручную. Используя интерфейс программы, эксперт (или консультант) может определить местонахождение систем в домене (или рабочей группе), где нужно установить F-Response EE, как показано на илл. 3.4.



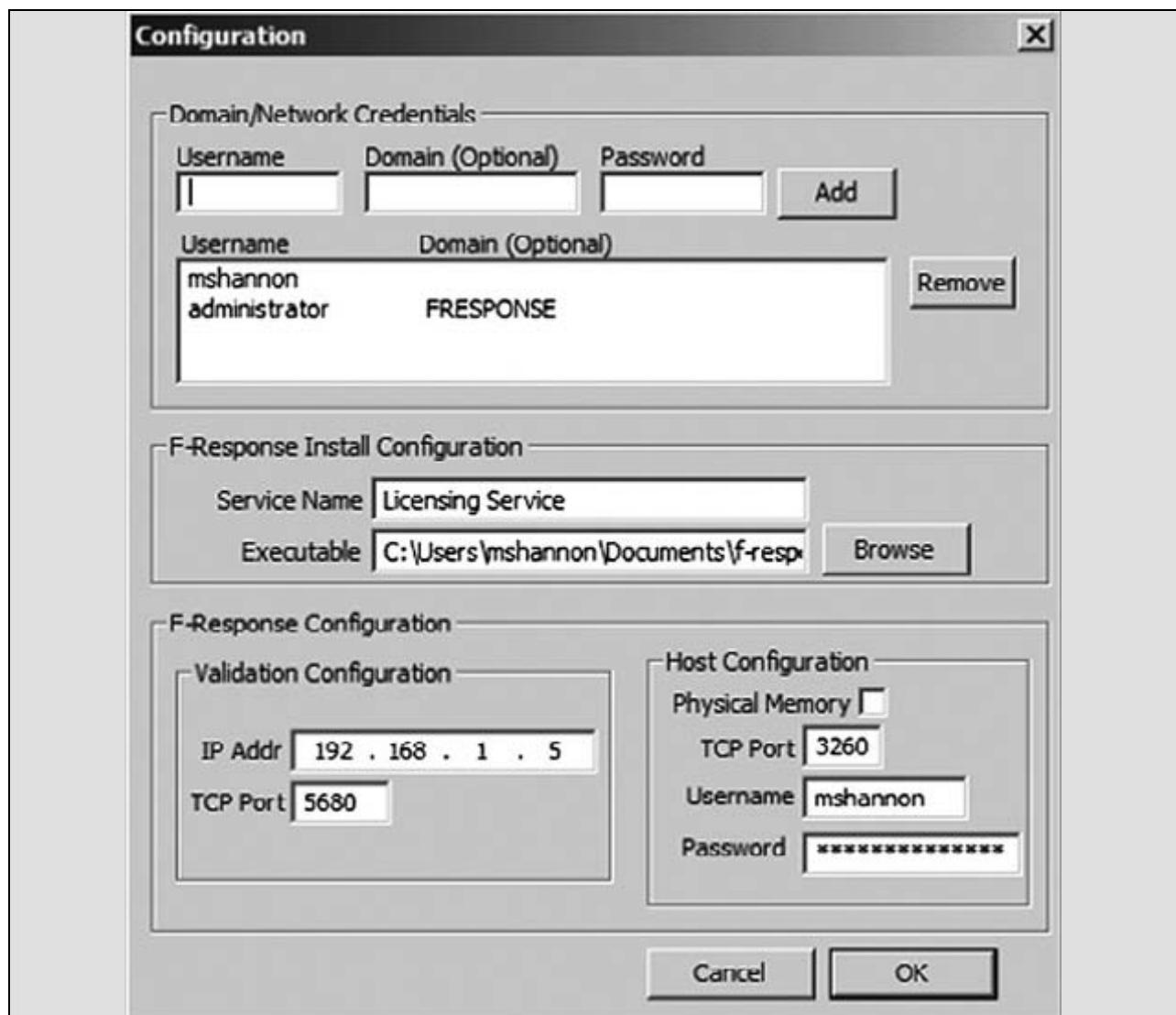
Илл. 3.4. Окно программы FEMC с перечнем доступных систем.

После определения местонахождения и выбора нужных систем развертывание и запуск службы F-Response выполняется с помощью всего нескольких щелчков мышью, как показано на илл. 3.5. Это действительно так независимо от того, в каком количестве систем нужно развернуть службу.



Илл. 3.5. Вход в F-Response EE через интерфейс программы FEMC.

Эксперту нужно только выбрать системы, в которых нужно развернуть F-Response, и сделать несколько щелчков мышью, чтобы агент установился автоматически. Через интерфейс пользователя можно даже редактировать параметры файла конфигурации (.ini), как показано на илл. 3.6.

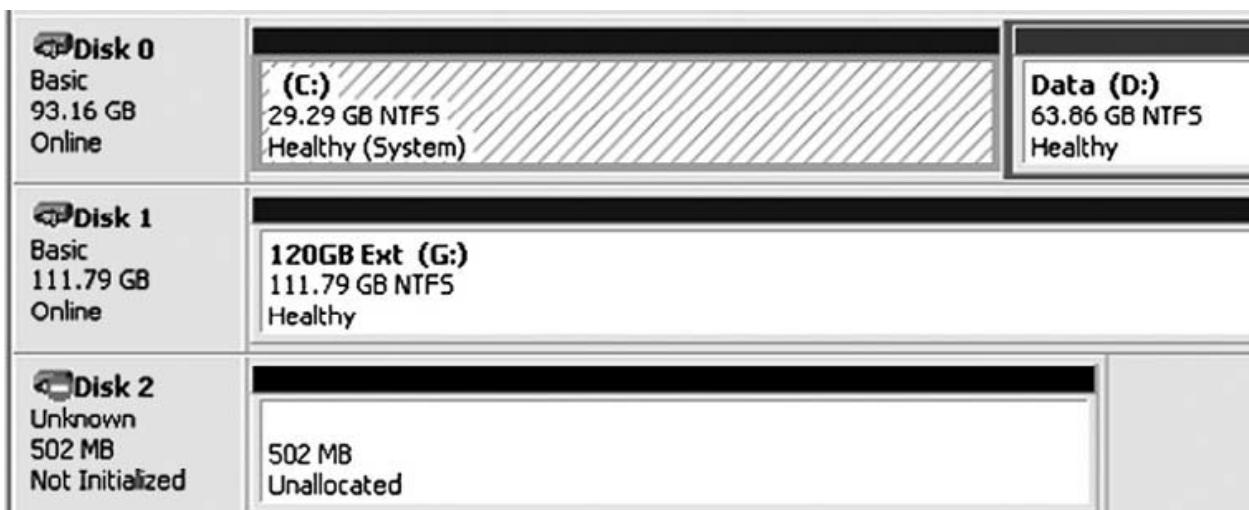


Илл. 3.6. Конфигурирование F-Response EE через интерфейс программы FEMC.

Информация, введенная в разделе «Учетные данные для входа в домен/сеть» (“Domain/Network Credentials”) диалогового окна «Конфигурация» (“Configuration”), показанного на илл. 3.6., доступна только во время сеанса работы, но никоим образом не сохраняется после закрытия FEMC.

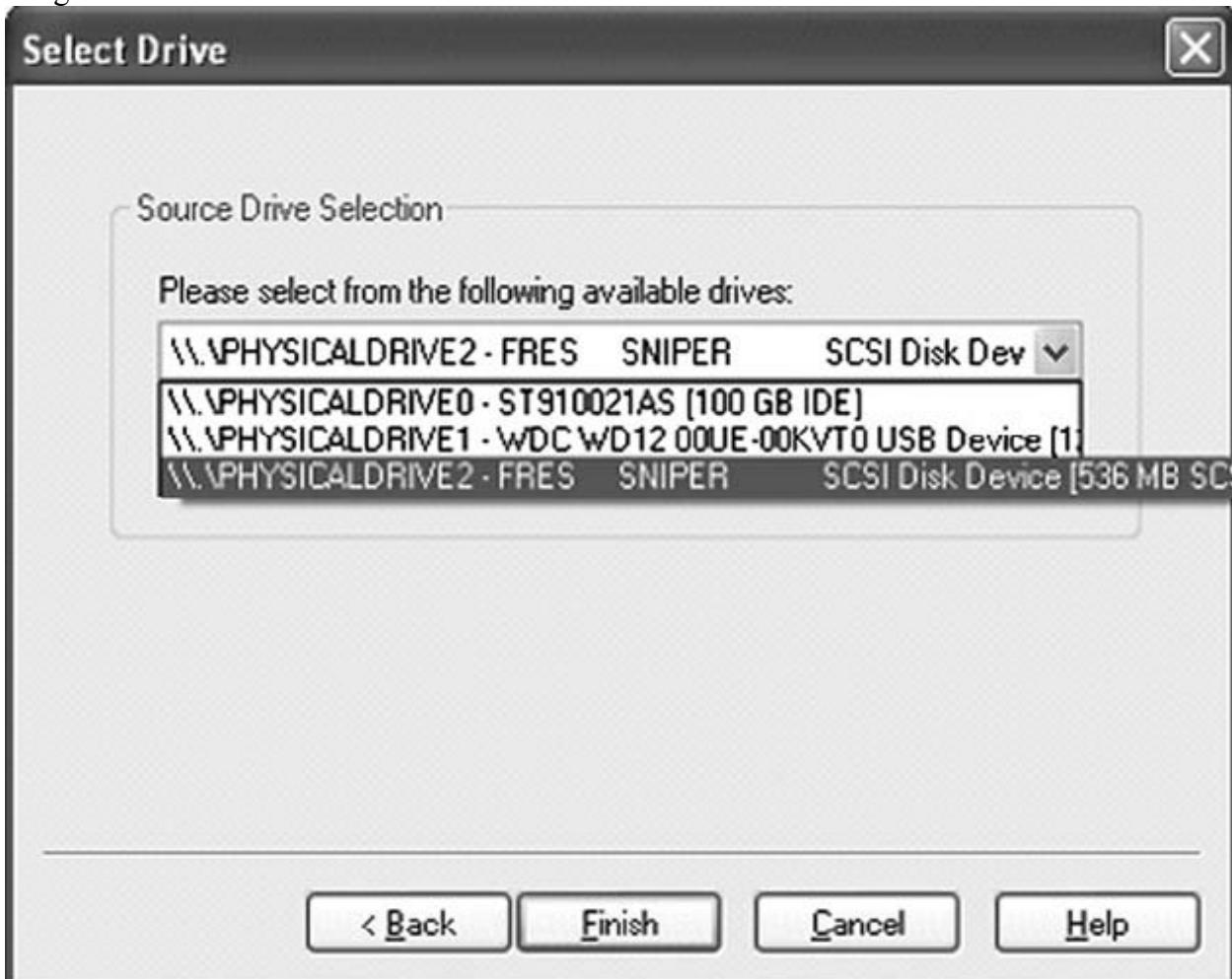
При работе с программой FEMC у меня больше сил ушло на то, чтобы играть в BrickBreaker на своем BlackBerry, чем на то, чтобы развернуть F-Response EE в нескольких системах, подключиться к каждой из них, а затем полностью удалить агент. FEMC выводит функциональные возможности чрезвычайно мощного и ценного инструмента на качественно новый уровень.

Так как в физической памяти (ОЗУ) нет таблицы разделов, компьютер эксперта при подключении к удаленной системе не будет распознавать физическую память как накопитель. Однако это подключение будет отображено в оснастке «Управление дисками» (“Disk Management”) консоли управления Microsoft, как показано на илл. 3.7.



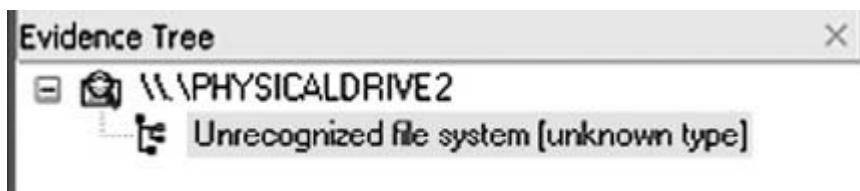
Илл. 3.7. Подключение F-Response EE к удаленному ОЗУ, отображаемому в оснастке «Управление дисками» (“Disk Management”) консоли управления Microsoft.

После того как подключение к ОЗУ удаленной системы будет проверено, можно использовать инструменты для создания дампа памяти, например FTK Imager. На илл. 3.8., показан процесс выбора ОЗУ удаленной системы посредством программы FTK Imager.



Илл. 3.8. Выбор ОЗУ удаленной системы в программе FTK Imager.

ОЗУ, выбранное на илл. 3.8., показано на илл. 3.9., в дереве исследуемых объектов в программе FTK Imager.



Илл. 3.9. Выбранное ОЗУ в дереве исследуемых объектов в FTK Imager.

Теперь ОЗУ удаленной системы можно клонировать, используя FTK Imager. Можно также использовать другие инструменты; F-Response – платформа, независимая от инструментов клонирования, так как она не требует, чтобы вы использовали какой-то конкретный инструмент. Эксперт имеет возможность применить EnCase, X-Ways Forensics или даже версии инструмента «dd.exe», чтобы создать дамп ОЗУ из удаленной системы. Как уже говорилось, подключение устанавливается в режиме только для чтения и, как показано на илл. 3.8. и 3.9., ОЗУ удаленной системы отображается на компьютере эксперта как объект \\.\PhysicalDrive (т. е. \\.\PhysicalDrive2 или \\.\PhysicalDrive3).

Примечание

Во время саммита SANS Forensic Summit в октябре 2008 года Аарон Уолтерс и Мэтт Шенон сделали презентацию некой платформы Voltage, объединяющей возможности инструментов F-Response и Volatility Framework (<http://volatilesys-tems.blogspot.com/2008/10/voltage-giving-investigators-power-to.html>). Хотя на момент написания этой книги Voltage была недоступна, эта платформа, судя по описанию, обеспечивает беспрецедентный уровень удаленного доступа (в реальном времени и в режиме только для чтения), автоматизации и виртуализации, чтобы получить недоступную ранее, важную информацию. Другими словами, Voltage предоставляет эксперту возможность быстро получить доступ к удаленным системам, определить системы, в которых, возможно, установлены и выполняются вредоносные программы, а затем собрать образец физической памяти, не изменяя при этом артефакты удаленной системы.

Краткое изложение раздела

Тринадцатого декабря 2008 года Джон Сойер (John Sawyer) опубликовал в блоге SANS Forensics статью (<http://sansforensics.wordpress.com/2008/12/13/windows-physical-memory-finding-the-right-tool-for-the-job/>) с перечнем и кратким описанием нескольких инструментов для сбора (а также анализа) дампов памяти из систем Windows. Помимо статьи Джона, существуют другие публикации (в блогах и электронных рассылках), в которых обсуждаются проблемы доступности и использования различных инструментов для создания дампа памяти из систем Windows. Во многих отношениях эта статья служит введением в тему инструментов, так как я не сомневаюсь, что эти инструменты изменятся еще до того, как книга с данной главой появится в продаже. Например, сразу после того, как я отправил эту главу на проверку техническому редактору и получил его комментарии, компания HBGary выпустила инструмент FastDump Pro и объявила о его доступности. Я твердо уверен, что состояние дел в области инструментов для создания (и анализа) дампа памяти изменится, и случится это относительно скоро.

В данном разделе вы познакомились с несколькими доступными инструментами для создания дампа содержимого физической памяти из систем Windows. Чтобы провести параллельное сравнение некоторых из этих инструментов, я выполнил небольшое собственное тестирование. Используя команды (в случае с программой Memoryze – пакетный файл «memorydd.bat») для каждого рассмотренного в этой главе инструмента, который можно запустить с компакт-диска или USB флеш-накопителя, я запустил каждый из них (за исключением «winen.exe») на компьютере Dell Latitude D820, в котором установлено 4 Гб ОЗУ и выполняется ОС Windows XP Service Pack 3. Процесс

тестирования состоял из загрузки системы, запуска одного из инструментов и, после того как этот инструмент закончит создание дампа памяти, перезагрузки системы и запуска следующего инструмента. При этом я сохранял дампы памяти, чтобы увидеть размер каждого из них относительно дампов, созданных другими инструментами; ниже показаны окончательные результаты в виде списка, содержащего размер в байтах и имя каждого файла выходных данных:

```
3 210 854 400 win32dd-10-xp.img
3 210 854 400 win32dd-11-xp.img
3 210 854 400 mdd-xp-2.img
3 210 854 400 mdd-xp.img
3 211 329 536 dd_xp.img
3 211 329 536 nigelant_xp.img
3 211 333 632 fdpro-xp.bin
3 211 334 904 fdpro-xp.hpak
3 211 329 536 memory.1e1d2b07.img (Memoryze)
```

Как видите, я включил имя инструмента и некоторую дополнительную идентифицирующую информацию из командной строки в имя файла выходного дампа. Кроме того, видно, что присутствует незначительный разброс в размерах полученных дампов, вероятно, из-за разных принципов работы инструментов. Как и ожидалось, исключением является файл дампа в формате .hpak, созданный инструментом FastDump Rго, так как использовался переключатель *-page*, чтобы добавить файл подкачки в процесс создания дампа. Размер файла был бы значительно больше, если бы в системе выполнялось больше операций, а система работала дольше.

На данный момент должно быть понятно, что выбор инструмента для сбора содержимого ОЗУ на самом деле зависит от нескольких факторов, в том числе от версии Windows (XP или Vista, 32- или 64-разрядная), объема физической памяти в компьютере и инструментов анализа данных.

Предупреждение

Как и в случае с любыми другими инструментами, загружаемыми из Интернета, обязательно протестируйте инструменты и ознакомитесь с лицензионным соглашением об их использовании. Иногда консультантам запрещается использовать некоторые (чрезвычайно эффективные и полезные) инструменты. Кроме того, необходимо знать об артефактах, оставляемых разными инструментами. Как упоминалось в главе 1, многие инструменты, доступные на сайте Microsoft (ранее на сайте Sysinternals), при запуске создают записи в реестре. Инструмент «winen.exe» делает что-то похожее, в том отношении, что он записывает свой драйвер («winen_.sys») в тот же каталог, из которого выполняется файл, и создает подраздел Services в реестре. Здесь, как и на других этапах расследования инцидента, самое главное – не отказываться от использования инструмента из-за того, что он оставляет «следы» или артефакты своей работы, а понять этот факт и учитывать его в своей методике и документации.

Альтернативные подходы к созданию дампа физической памяти

Программы, обсуждавшиеся до настоящего момента, – не единственные средства для создания дампа содержимого физической памяти из работающей системы; в прошлом было разработано несколько альтернативных способов. Некоторые из этих способов используют функциональные возможности, свойственные операционной системе (например, аварийные дампы памяти) или платформе виртуализации (например, VMware). Как мы увидим, альтернативные способы для создания дампа физической памяти из системы основаны, как правило, на использовании аппаратных средств.

Аппаратные устройства

В феврале 2004 года в журнале *Digital Investigation Journal* была опубликована статья Брайана Кэрриера (Brian Carrier) и Джо Гранда (Joe Grand), которая называлась «A Hardware-Based Memory Acquisition Procedure for Digital Investigations». В этой статье Брайан и Джо представили концепцию платы расширения, прозванной Tribble (возможно, в честь той незабываемой серии из сериала «Звездный путь» (*Star Trek*)), которую можно было бы использовать для создания дампа физической памяти и сохранения его на внешнем носителе. Это позволило бы эксперту получить энергозависимую память из системы, не используя для этого новых или потенциально ненадежных инструментов. Авторы статьи утверждали, что разработали экспериментальное устройство в виде платы расширения PCI, которую можно подключить к шине компьютера. Существуют другие аппаратные устройства, позволяющие получить содержимое физической памяти и предназначенные в основном для отладки аппаратных систем. Эти устройства также можно использовать для компьютерно-технической экспертизы.

Как было показано в задаче на анализ памяти (упоминавшееся ранее в этой главе) с конференции DFRWS 2005, один из недостатков программного подхода к получению энергозависимой памяти заключается в том, что программа, которую использует эксперт, должна быть загружена в память. Следовательно, особенно в системах Windows, программа может зависеть от ненадежного кода или библиотек (DLL-файлов), измененных злоумышленником. Ниже рассматриваются преимущества и недостатки аппаратных устройств.

Устройства, подобные Tribble, не оказывают воздействия на работу системы и легкодоступны. При создании дампа содержимого физической памяти таким способом в систему не устанавливаются новые или дополнительные программы, что сводит возможность какого-либо искажения данных к минимуму. Однако главный недостаток подхода, основанного на применении аппаратных средств, заключается в том, что аппаратные средства нужно устанавливать до инцидента. На данный момент устройства Tribble слабо распространены. Доступны другие устройства, предназначенные для отладки аппаратных средств, но их также нужно устанавливать до возникновения инцидента.

FireWire

Благодаря техническим особенностям устройств и протокола FireWire, существует возможность, что при использовании подходящего программного обеспечения эксперт может собрать содержимое физической памяти из системы. Устройства FireWire используют прямой доступ к памяти (Direct Memory Access, DMA), то есть они могут получить доступ к памяти системы без использования процессора. Эти устройства могут выполнять чтение из памяти и запись в память намного быстрее, чем системы, не использующие DMA. Эксперту необходим контроллер устройства, содержащий соответствующее программное обеспечение и способный записывать команды в специальную область пространства устройства FireWire. Распределение памяти выполняется в устройстве без использования операционной системы компьютера, что позволяет передавать данные с высокой скоростью и малой задержкой.

Адам Буало (Adam Boileau) предложил способ для извлечения физической памяти из системы, используя ОС Linux и язык программирования Python (см. www.storm.net.nz/projects/16). Программа, используемая для этого способа сбора данных, работает в ОС Linux и основывается на поддержке устройства /dev/raw1394, а также на библиотеке Адама pythonraw1394, библиотеки libraw1394 и инструмента Swig (который делает заголовочные файлы C/C++ доступными для других языков, создавая код оболочки). Во время демонстрации этого способа Адам даже использовал инструмент,

который собирает содержимое ОЗУ из ОС Windows с заблокированным экраном, затем путем анализа памяти получает пароль, после чего Адам входил в систему.

Джон Эванс (Jon Evans), сотрудник полицейского управления в графстве Гент в Великобритании, установил инструмент Адама и успешно собрал содержимое физической памяти как из ОС Windows, так и из различных версий ОС Linux. В рамках своей диссертации Джон описал этапы установки, настройки и использования этого инструмента в различных дистрибутивах Linux, в том числе Knoppix версии 5.01, Gentoo Linux версии 2.6.17 и BackTrack (с сайта Remote-exploit.org). После описания этапов загрузки и установки всех необходимых пакетов (в том числе инструментов Адама) Джон подробно объясняет, как определить порты FireWire и заставить целевую ОС Windows «думать», что система Linux – это iPod (используя команду Linux *romtool*, чтобы загрузить файл данных, содержащий регистр управления состоянием (Control Status Register, CSR) для iPod (CSR-файл предоставляется вместе с инструментами Адама)). Ниже показаны достоинства и недостатки такого подхода.

Многие современные компьютеры выпускаются с интерфейсом IEEE 1394 (FireWire), встроенным непосредственно в системную плату, что увеличивает возможность доступа к физической памяти при помощи этого способа. Однако Арне Видстром (Arne Vidstrom) обратил внимание на то, что при создании дампа физической памяти через FireWire возможны технические проблемы, например, зависание системы или пропуск некоторых областей памяти (см. <http://ntsecurity.nu/onmymind/2006/2006-09-02.html>). Джордж Гарнер-младший отметил в одной из рассылок в октябре 2006 года, что в ограниченных испытаниях наблюдались заметные различия в важных смещениях между дампом ОЗУ, собранным при помощи способа с использованием FireWire, и дампом ОЗУ, собранным при помощи программы, разработанной Джорджем. Это различие можно было объяснить только ошибкой в данном способе, используемом для сбора памяти. Более того, этот способ был причиной ошибок BSOD (англ. *Blue Screen of Death* – синий экран смерти; см. следующую главу) в нескольких целевых системах Windows, возможно, из-за типа устройств FireWire в системе.

Аварийные дампы памяти

Время от времени мы все сталкиваемся с аварийными дампами памяти, которые обнаруживаются после ошибок BSOD (более подробную информацию см. на странице http://en.wikipedia.org/wiki/Blue_Screen_of_Death). В большинстве случаев такие ошибки служат поводом для беспокойства или даже свидетельствуют о серьезных проблемах. Однако если необходимо получить копию первоначального состояния ОЗУ из системы Windows, то, возможно, единственный способ для этого – создать полный аварийный дамп памяти. Это объясняется тем, что при создании аварийного дампа памяти состояние системы фиксируется, а содержимое ОЗУ (вместе с примерно 4 Кб информации заголовка) записывается на накопитель. Таким образом сохраняется состояние системы и обеспечивается целостность данных системы с того момента, как начинается создание дампа.

Эти данные могут быть чрезвычайно ценными для эксперта. Во-первых, содержимое дампа памяти – снимок состояния системы на момент времени. Я участвовал в нескольких расследованиях, во время которых обнаруженные дампы памяти использовались для определения основных причин инцидентов, например, маршрутов, использовавшихся для заражения или взлома системы. Во-вторых, Microsoft предоставляет инструменты для анализа дампов памяти – не только в средствах отладки от Microsoft (www.microsoft.com/whdc/devtools/debugging/default.mspx), но и в программе Kernel Memory Space Analyzer (для поиска этой программы рекомендуется воспользоваться поисковой системой из-за удивительно длинного URL-адреса ее загрузки), основанной на средствах отладки.

Звучит многообещающе, не так ли? Ведь, кроме того, что на НЖМД сохраняется файл размером 1 Гб, перезаписывая при этом потенциальные улики (и не уменьшая на самом деле воздействия исследования на систему), у этого подхода нет недостатков, правда? В зависимости от обстоятельств вы, возможно, будете готовы смириться с этим условием. Однако существует еще несколько проблем. Во-первых, не все системы по умолчанию создают аварийные дампы памяти. Во-вторых, по умолчанию системы Windows не создают аварийные дампы памяти по команде.

Первую проблему решить относительно просто, согласно статье № Q254649 из базы знаний Microsoft (<http://support.microsoft.com/kb/254649>). В этой статье описывается три типа аварийного дампа памяти: малый дамп (64 Кб), дамп памяти ядра и полный дамп. Нас интересует полный аварийный дамп, так как он содержит полное одержимое ОЗУ. В статье также говорится, что ОС Windows 2000 Pro и Windows XP (как версии Pro, так и Home) создают малый аварийный дамп памяти, а ОС Windows 2003 (всех версий) создает полный дамп памяти. Мой опыт работы с ОС Windows Vista RC1 показывает, что по умолчанию она создает малый дамп памяти.

Примечание

В статье № 235496 (<http://support.microsoft.com/kb/235496>) из базы знаний Microsoft указаны записи реестра, содержащие сведения о конфигурации для систем Windows в отношении файла «memory.dmp», который является результатом создания аварийного дампа памяти. Место расположения этого файла по умолчанию – %SystemRoot%\memory.dmp, но администратор может указать другое место в параметре реестра *DumpFile*.

Вместе с тем в статье № Q274598 (<http://support.microsoft.com/kb/274598>) из базы знаний Microsoft говорится, что полный дамп аварийной памяти нельзя создать в системах, имеющих более 2 Гб ОЗУ. Согласно статье, это обусловлено в основном требованиями к свободному пространству на накопителе (т. е. для систем с включенной возможностью создавать полный дамп аварийной памяти файл подкачки должен иметь такой же размер, как содержимое ОЗУ, плюс 1 Мб), а также количеством времени, которое потребуется на создание аварийного дампа памяти.

В статье № Q307973 (<http://support.microsoft.com/kb/307973>) из базы знаний Microsoft описывается, как настроить все разнообразные параметры обработки сбоев и восстановления системы. Эти параметры больше необходимы системным администраторам и руководителям ИТ-отделов, которые настраивают и конфигурируют системы до возникновения инцидентов, однако параметры разделов реестра могут предоставить эксперту важную информацию. Например, если система настроена (по умолчанию или другим способом) на создание полного дампа оперативной памяти, а администратор сообщил об ошибке BSOD, то эксперт должен ожидать увидеть в системе файл полного аварийного дампа памяти.

Примечание

Эксперты должны быть очень внимательны при работе с файлами аварийного дампа памяти, особенно если они были получены из систем, в которых обрабатываются, но не обязательно хранятся, конфиденциальные данные. В некоторых случаях аварийные дампы были созданы в системах, обрабатывающих такую информацию, как номера кредитных карт, номера социального страхования и т. д., и было обнаружено, что в этих дампах содержатся эти конфиденциальные данные. Даже несмотря на то, что программисты специально разрабатывают приложения так, чтобы никакие конфиденциальные данные не сохранялись локально на НЖМД, при создании аварийного дампа содержимое физической памяти записывается на накопитель.

Итак, предположим, что параметры обработки сбоев и восстановления системы настроены заранее (в рамках политики конфигурации для систем) на создание полного аварийного дампа памяти. Как эксперту «заставить» систему создавать аварийный дамп памяти по команде, когда это необходимо? Оказывается, есть раздел реестра, в котором можно задать параметры, чтобы аварийный дамп памяти создавался при удерживании нажатой клавиши «**Ctrl**» и двойном нажатии клавиши «**Scroll Lock**» (см. статью № Q244139 (<http://support.microsoft.com/kb/244139>) из базы знаний Microsoft). Однако для того, чтобы заданный параметр вступил в силу, систему необходимо перезагрузить. Давайте рассмотрим преимущества и недостатки этого способа.

Создание аварийного дампа памяти – возможно, единственный правильный с технической (хотя и не с криминалистической) точки зрения способ получения образа одержимого ОЗУ. Это объясняется тем, что при вызове API-функции *KeBugCheck* вся работа системы приостанавливается, а содержимое ОЗУ записывается в файл подкачки, а затем – в файл на системном НЖМД (перезаписывая данные). Кроме того, Microsoft предоставляет средства отладки, а также программу Kernel Memory Space Analyzer (состоящую из основной части, подключаемых модулей и интерфейса пользователя) для анализов файлов аварийного дампа памяти. Некоторые системы Windows не создают полные дампы аварийной памяти по умолчанию (например, Vista RC1; у меня была проблема с одним из драйверов, когда я впервые установил ОС Windows Vista RC1, и при каждой попытке выключить компьютер возникала ошибка BSOD, в результате чего создавались файлы малого дампа памяти).

Кроме того, при изменении настроек системы для того, чтобы аварийный дамп памяти создавался по нажатию сочетания клавиш, требуется перезагрузка компьютера, поэтому данные изменения необходимо выполнить заранее, чтобы этот способ можно было эффективно использовать при расследовании инцидентов. Даже если эти изменения будут выполнены заранее, при создании аварийного дампа на НЖМД будет все равно сохранен файл, размер которого равен физической памяти. Для этого, как указано в статье № Q274598 из базы знаний, файл подкачки необходимо настроить так, чтобы его размер как минимум на 1 Мб превышал размер физической памяти. Это дополнительный параметр, который необходимо задать, чтобы использовать данный способ для сбора содержимого физической памяти, и о котором часто забывают.

Совет

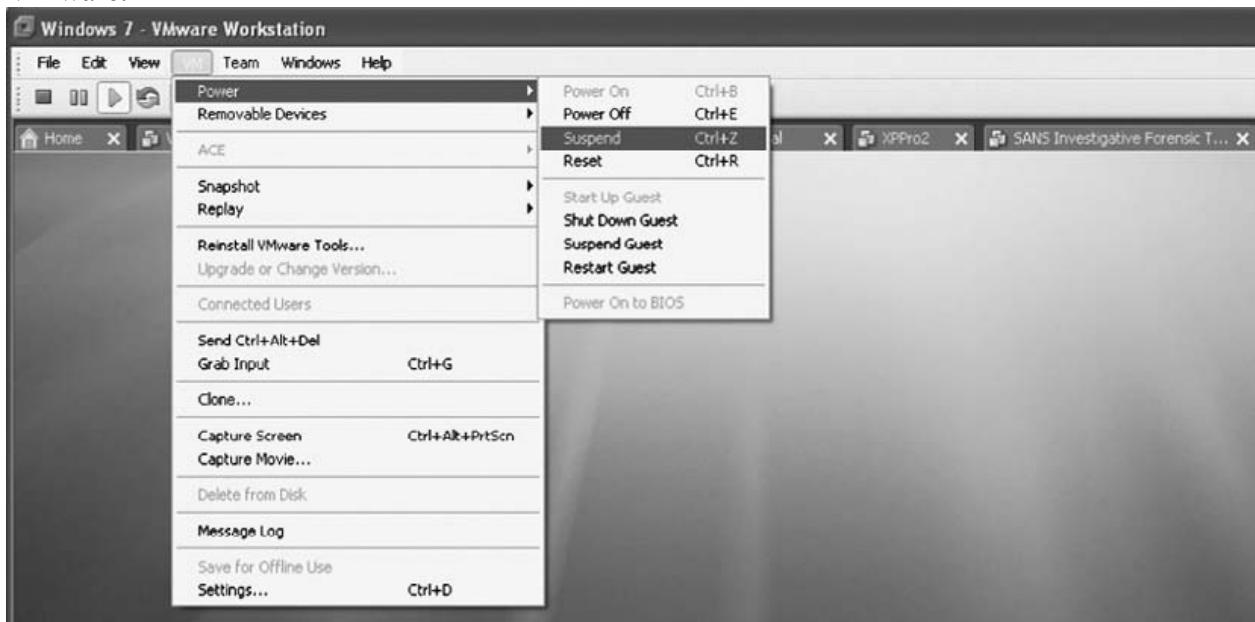
В справочной статье на веб-сайте компании Citrix (<http://support.citrix.com/article/CTX107717&parentCategoryID=617>) описывается способ использования инструмента «*livekd.exe*» (<http://technet.microsoft.com/en-us/sysinternals/bb897415.aspx>) и средств отладки от Microsoft Debugging для создания полного дампа физической памяти. После запуска «*livekd.exe*» используется команда *.dump /f <путь к файлу>* для создания файла дампа. Справочная статья также содержит предупреждение о том, что дампы ОЗУ, созданные таким способом, могут быть некорректны из-за того, что на создание дампа памяти может потребоваться значительное количество времени, а система продолжает работать во время создания дампа.

Виртуализация

VMware – популярная программа виртуализации (при написании данной книги активно использовалась программа VMware Workstation версии 5.5.2), которая, прежде всего, позволяет создавать псевдосети, используя аппаратные средства одного компьютера. Данная возможность имеет много преимуществ. Например, можно установить гостевую операционную систему и создать снимок состояния системы после того, как она будет настроена для ваших потребностей. Затем можно выполнять любые виды тестирований, в том числе установку вредоносных программ и наблюдение за ними, и вы всегда будете иметь возможность вернуться к предыдущему состоянию системы и

начать все заново. Я даже видел активные производственные системы, выполняющиеся из сеансов VMware.

Выполняющийся сеанс VMware можно приостановить, временно зафиксировав состояние системы. На илл. 3.10., показаны элементы меню для приостановки сеанса VMware.



Илл. 3.10. Элементы меню для приостановки сеанса в VMware Workstation 6.5.

Когда сеанс VMware приостановлен, содержимое физической памяти содержится в файле с расширением .vmem. Формат этого файла очень похож на формат необработанных данных памяти (как в dd), и фактически этот файл можно проанализировать при помощи тех же инструментов, что были рассмотрены ранее в этой главе.

VMware – не единственная программа виртуализации. К другим программам относятся VirtualPC от Microsoft, а также бесплатная Bochs (<http://bochs.sourceforge.net/>). Ни одна из этих программ виртуализации не была проверена на предмет того, может ли она создавать дампы физической памяти; однако, если вам доступна такая возможность, приостановка сеанса VMware, чтобы получить дамп физической памяти, – быстрый и легкий способ, сводящий к минимуму ваше взаимодействие с системой и воздействие на систему.

Совет

В мае 2006 года Брет Шэйверс (Brett Shavers) написал отличную статью для веб-сайта ForensicFocus, которая называется «VMware as a forensic tool» (www.forensicfocus.com/vmware-forensic-tool). Позднее, в 2008 году, Бред написал статью «Virtual Forensics: A Discussion of Virtual Machines Related to Forensics Analysis» (www.forensicfocus.com/downloads/virtual-machines-forensics-analysis.pdf), представляющую собой лучшее исследование по этой теме, которое мне удалось найти.

Файл спящего режима

Когда ОС Windows (Windows 2000 или более поздней версии) переходит в спящий режим, диспетчер питания сохраняет сжатое содержимое физической памяти в файле «hiberfil.sys» в корневом каталоге системного тома. Этот файл имеет достаточно большой размер, чтобы вместить несжатое содержимое физической памяти, однако сжатие используется для уменьшения количества операций дискового контроллера ввода-вывода и улучшения выхода из спящего режима. Если во время загрузки обнаруживается допустимый файл «hiberfil.sys», загрузчик NT Loader (NTLDR) загружает содержимое

файла в физическую память и передает управление коду в ядре, который обрабатывает системные операции, возобновляемые после спящего режима (загрузка драйверов и т. д.). Данная функциональная возможность чаще всего используется в ноутбуках. Ниже рассматриваются преимущества и недостатки этого способа.

Анализ содержимого файла спящего режима может предоставить вам информацию о том, что происходило в системе в определенный момент времени в прошлом. Мэттье Свиш декодировал формат файла спящего режима и показал свои результаты на конференции BlackHat USA 2008 (www.blackhat.com/presentations/bh-usa-08/Suiche/BH_US_08_Suiche_Windows_hibernation.pdf). Данной презентации предшествовала его статья на эту тему в феврале 2008 года, которая называлась «Sandman Project» (http://sandman.msuiche.net/docs/SandMan_Project.pdf), и работа с Николасом Раффом (Nicolas Ruff). Инструмент Sandman доступен на странице www.msuiche.net/category/sandman/, а его функциональные возможности были включены в состав платформы Volatility Framework (которая будет рассматриваться позднее в этой главе).

Следует также иметь в виду, что возможность перехода в спящий режим может быть единственным доступным вариантом для сбора полного содержимого физической памяти. Некоторые существующие инструменты для сбора содержимого физической памяти могут иметь проблемы (точнее, могут быть причиной сбоя систем, имеющих более 4 Гб ОЗУ) или быть недоступными или неприемлемыми для использования в некоторых средах. Применение инструмента «powercfg.exe», чтобы разрешить использование спящего режима (если это еще не разрешено), а затем какого-нибудь другого механизма или инструмента, чтобы заставить систему перейти в спящий режим, может быть единственной возможностью получить дамп памяти. Можно заставить систему (с разрешенным спящим режимом) перейти в спящий режим, используя инструмент «psshutdown.exe» от Microsoft (<http://technet.microsoft.com/en-us/sysinternals/bb897541.aspx>) с параметром *-h* или создав пакетный файл, содержащий следующую строку:

```
%windir%\System32\rundll32.exe powrprof.dll,SetSuspendState Hibernate
```

Файл спящего режима сжимается и в большинстве случаев не будет содержать *текущего* содержимого памяти. Благодаря работе, выполненной Мэттье и Николасом, к файлу спящего режима можно получить доступ так же, как к дампу памяти из работающей системы, поэтому на самом деле я не могу найти недостатки этого способа. Имея дамп спящего режима, вы сможете получить доступ к содержимому памяти на определенный момент времени в прошлом, которое можно сравнить с содержимым дампа памяти из работающей системы.

Анализ дампа физической памяти

Теперь, когда вы познакомились со способами клонирования содержимого ОЗУ, полученного из системы (как локально, так и удаленно), что можно сделать с дампом памяти? До лета 2005 года для большинства специалистов, которые потрудились собрать дамп памяти (обычно при помощи версии инструмента «dd.exe» из пакета Forensic Acquisition Utilities, разработанного Джорджем Гарнером-младшим), стандартной рабочей процедурой было применение инструмента «strings.exe» к дампу или выполнение поиска (адресов электронной почты, IP-адресов и т. д.) при помощи утилиты *grep*, или и то и другое. Хотя такой способ позволял получить информацию, которая часто приводила к каким-то определенным результатам (например, обнаруженные данные, похожие на пароль и, возможно, связанные с именем пользователя, могли указать эксперту на последующие этапы расследования), он не предоставлял полного контекста к найденной информации. Например, была ли найденная строка частью текстового документа, или она

была скопирована в буфер обмена системы? Какой процесс использовала память, где была найдена строка или IP-адрес?

Начиная с задачи на анализ памяти, предложенной на конференции DFRWS 2005, были предприняты шаги, чтобы добавить контекст к информации, найденной в ОЗУ. Найдя отдельные процессы (или другие объекты, хранящиеся в памяти) и страницы памяти, используемые этими процессами, эксперт может получить большее представление об обнаруженной информации, а также значительно сократить объем этой информации, отфильтровав заведомо допустимые процессы и данные и сосредоточившись на данных, которые кажутся «необычными». Несколько специалистов разработали инструменты, которые можно использовать для анализа дампов ОЗУ и поиска подробной информации о процессах и других структурах.

В 2007 году Аарон Уолтерс выпустил Volatility Framework (<https://www.volatilesystems.com/>), платформу с открытым исходным кодом на основе языка Python для анализа дампов памяти из систем Windows XP (на момент написания этой книги). Летом 2008 года непосредственно перед конференцией DFRWS 2008 Аарон провел первый семинар Open Memory Forensics Workshop (<https://www.volatilesystems.com/default/omfw>), во время которого исследователи, аналитики и специалисты-практики могли общаться между собой и обсуждать вопросы, связанные с клонированием и анализом памяти. Версия 1.3 платформы Volatility Framework была доступна на конференции DFRWS и даже использовалась во время конкурса «Судебное rodeo» (“Forensic Rodeo”).

В оставшейся части данной главы мы рассмотрим эти инструменты для проведения анализа дампов памяти. Вначале мы будем использовать дампы из задачи на анализ памяти, предложенной на конференции DFRWS 2005, в качестве примеров и демонстрации инструментов и приемов для анализа дампов памяти из ОС Windows 2000. Вероятно, вы задаетесь вопросом: «Для чего вообще это нужно?» Ведь ОС Windows 2000 – это обновленная MS-DOS, не так ли? Ну, вероятно, вы не далеки от истины, но эти дампы на самом деле служат отличной основой для примеров, потому что их уже исследовали очень подробно. Кроме того, они свободно доступны для загрузки и анализа.

Определение операционной системы на основе дампа памяти

Давали ли вам когда-нибудь образ системы, а когда вы спрашивали, из какой операционной системы он получен, то просто слышали «Windows» в ответ? «Все так же сладок розы аромат, когда другое имя ей дано», – говорил Шекспир, но при анализе дампа памяти, друзья мои, все по-другому. Когда вы работаете с образом системы, версия ОС Windows, из которой он получен, *имеет значение*, и в зависимости от решаемой проблемы версия может иметь большое значение. Это же справедливо (фактически, даже в большей степени) в случаях, когда вы имеете дело с файлом дампа ОЗУ. Как я уже говорил, структуры, используемые для определения потоков и процессов в памяти, отличаются не только в зависимости от основных номеров версий операционной системы, но и в зависимости от установленных пакетов обновления в одной и той же версии.

Поэтому, когда кто-либо дает вам дамп ОЗУ и говорит: «Windows», вы, вероятно, захотите узнать, о какой именно версии идет речь, не так ли? Ведь вы не хотите потратить уйму времени, обрабатывая дамп памяти при помощи всех известных вам инструментов, пока один из них не начнет выдавать успешные результаты по процессам, правда? Некоторое время тому назад в личной корреспонденции (это изысканное выражение для обозначения электронной почты) Андреас Шустер навел меня на мысль, что ядро Windows, возможно, загружается в одно и то же место в памяти при каждой загрузке операционной системы. Это место, вероятно, будет изменяться (и на самом деле изменяется) для разных версий Windows, но пока оно, похоже, неизменно для каждой отдельной версии. Самый простой способ найти это место – запустить инструмент

LiveKD, как описано ранее в этой главе, и обратить внимание на информацию, отображаемую при его запуске (здесь показаны данные для ОС Windows XP SP2):

```
Windows XP Kernel Version 2600 (Service Pack 2) MP (2 procs) Free x86
compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpssp.050329-1536
Kernel base = 0x804d7000 PsLoadedModuleList = 0x8055c700
```

Нас больше всего интересует информация, которую я выделил полужирным, – адрес базы ядра. Мы отнимем 0x80000000 от этого адреса, а затем перейдем к полученному физическому расположению в файле дампа. Если первые два байта, расположенные по этому адресу, – это *MZ*, то в этом месте у нас есть полнофункциональный переносимый исполняемый (PE) файл Windows, и, возможно, у нас есть ядро. С этого момента мы можем использовать код, похожий на тот, что содержится в скрипте «lspi.pl», чтобы проанализировать заголовок PE-файла и найти различные разделы в PE-файле. Так как ядро Windows – это допустимый файл приложения Microsoft, мы можем быть уверены, что в нем есть раздел ресурсов, содержащий раздел *VS_VERSION_INFO*. Следяя информации, предоставленной Microsoft, о различных структурах, составляющих этот раздел, мы затем можем проанализировать его на наличие строки описания файла.

На сопроводительном DVD-носителе вы найдете файл «osid.pl», выполняющий такой анализ. Скрипт «osid.pl» возник как «kern.pl», а затем оказался в программе PTFinderFE Ричарда МакКуона (Richard McQuown). Однажды Ричард спросил меня по электронной почте, возможно ли сделать выходные данные скрипта короче и понятнее, поэтому я внес некоторые изменения в файл (изменил выходные данные, добавил некоторые переключатели и т. д.) и дал ему другое имя.

Скрипт «osid.pl», в его простейшем виде, можно запустить из командной строки, указывая путь к файлу-образу в качестве единственного аргумента:

```
C:\Perl\memory>osid.pl d:\hacking\xp-laptop1.img
```

Или же можно указать отдельный файл, используя переключатель *-f*:

```
C:\Perl\memory>osid.pl -f d:\hacking\xp-laptop1.img
```

Обе эти команды выдадут одинаковые выходные данные; в данном случае дамп ОЗУ был получен из системы Windows XP SP2, поэтому скрипт вернет строку *XPSP2*. Если этой информации недостаточно, и вы хотите узнать больше, можно добавить переключатель *-v* (от англ. *verbose* – подробный), и скрипт вернет следующие данные для файла «xp-laptop1.img»:

```
OS      : XPSP2
Product : Microsoft® Windows® Operating System ver 5.1.2600.2622
```

Как видите, строки внутри структуры *VS_VERSION_INFO*, которые относятся к названию и версии продукта, были объединены, чтобы получить дополнительные выходные данные. Если мы применим этот скрипт с переключателями *-v* и *-f* к первому файлу дампа ОЗУ из задачи на анализ памяти с конференции DFRWS 2005, то получим:

```
OS      : 2000
Product : Microsoft® Windows® 2000 Operating System ver 5.00.2195.1620
```

Запуск этого скрипта для анализа других дампов памяти, упомянутых ранее в этой главе, покажет, насколько хорошо он работает в отношении различных версий ОС Windows. Например, применив скрипт к дампу памяти, использовавшемуся в примере с программой Memoryze, мы увидим следующее:

```
C:\Perl\memory>osid.pl -f d:\hacking\boomer-win2003.img -v
OS      : 2003
Product : Microsoft® Windows® Operating System ver 5.2.3790.0
```

Запустив этот скрипт для анализа другого дампа памяти из ОС Windows 2003, мы получим немного другие результаты:

```
C:\Perl\memory>osid.pl -f d:\hacking\win2k3sp1_physmem.img -v
OS      : 2003SP1
Product : Microsoft® Windows® Operating System ver 5.2.3790.1830
```

Этот скрипт так же хорошо работает с файлами .vmem программы VMware. Я применил этот скрипт к vmem-файлу из сеанса VMware с ОС Windows 2000 и получил следующие выходные данные:

```
OS      : 2000
Product : Microsoft (R) Windows (R) 2000 Operating System ver 5.00.2195.7071
```

Я думаю, что эти примеры достаточно наглядно показывают полезность подобных скриптов или инструментов, так как с их помощью эксперт получает возможность более подробно документировать различные исследуемые объекты, особенно те из них, которые, возможно, не были полностью задокументированы во время первоначального сбора данных.

Совет

Большинство инструментов анализа, обсуждаемых позднее в этой главе, могут выполнять ту же функцию, которая была только что рассмотрена. Например, можно использовать команду *ident* платформы Volatility, чтобы определить операционную систему на основе дампа памяти.

Основы процессов

В данной главе мы в основном сосредоточим наше внимание на анализе информации о процессах из дампа памяти. Частично это связано с тем, что большинство общедоступных исследований и инструментов ориентировано на процессы как на источник судебной информации. Это не значит, что другие объекты следует исключать из анализа, просто большая часть исследователей, похоже, делает упор на процессы. Позднее в этой главе мы рассмотрим другие средства получения информации из дампа ОЗУ, но пока мы сосредоточим усилия на процессах. Исходя из этого, мы должны достаточно хорошо понимать, что представляет собой процесс в памяти. Последующий раздел посвящен процессам в памяти ОС Windows 2000, но большинство понятий не изменяются для всех версий Windows. Самое большое отличие – в фактической структуре самого процесса, а подробное изучение подробностей структур процессов во всех версиях Windows выходит за рамки этой книги.

Структура EProcess

Каждый процесс в системе Windows представлен блоком процесса, создаваемым исполнительной системой, или EProcess. Блок EProcess – это структура данных, в которой хранятся различные атрибуты процессов, а также указатели на ряд других атрибутов и

структур данных (потоки, блок переменных окружения процесса), относящихся к процессу. Так как эта структура данных представляет собой последовательность байтов, а каждая последовательность имеет смысл и цель, эксперт может прочитать и проанализировать эти структуры данных. Однако следует иметь в виду, что единственная неизменная особенность, касающаяся этих структур, в различных версиях ОС Windows заключается в том, что они изменяются. Вы не ослышались. Размер и даже значения структур изменяются не только в зависимости от версий операционной системы (например, Windows 2000 и XP), но и в зависимости от пакетов обновления в одной и той же версии операционной системы (Windows XP и XP SP2).

Андреас Шустер выполнил потрясающую работу, опубликовав структуры блока EProcess в своем блоге (http://computer.forensikblog.de/en/topics/windows/memory_analysis). Как бы то ни было, просмотреть содержимое структуры EProcess (или любой другой структуры, имеющейся в Windows) – относительно легко. Сначала загрузите и установите средства отладки от Microsoft и нужные символы для операционной системы и пакета обновления. Затем загрузите инструмент «livekd.exe» с сайта [Sysinternals.com](http://www.sysinternals.com) и для удобства скопируйте его в тот же каталог, где находятся средства отладки (при вводе [sysinternals.com](http://www.sysinternals.com) в адресную строку браузера вы будете автоматически перенаправлены на сайт Microsoft, так как теперь Марк Руссинович является сотрудником Microsoft). После этого откройте командную строку, перейдите к каталогу, где установлены средства отладки, и введите следующую команду:

```
D:\debug>livekd -w
```

Данная команда откроет WinDbg, графический интерфейс пользователя для средств отладки. Чтобы увидеть, как выглядит полное содержимое блока EProcess (со всеми подструктурами, составляющими структуру EProcess), введите **dt -a -b -v _EPROCESS** в окно командной строки и нажмите «Enter». Флаг **-a** показывает каждый элемент массива с новой строки, а переключатель **-b** рекурсивно показывает блоки. Флаг **-v** создает более подробные выходные данные, указывая, например, полный размер каждой структуры. В некоторых случаях полезно добавлять флаг **-r** для получения рекурсивных выходных данных. Ниже показан краткий фрагмент выходных данных этой команды, запущенной в системе Windows 2000:

```
kd> dt -a -b -v _EPROCESS
struct _EPROCESS, 94 elements, 0x290 bytes
+0x000 Pcb : struct _KPROCESS, 26 elements, 0x6c bytes
+0x000 Header : struct _DISPATCHER_HEADER, 6 elements, 0x10 bytes
    +0x000 Type : UChar
    +0x001 Absolute : UChar
    +0x002 Size : UChar
    +0x003 Inserted : UChar
    +0x004 SignalState : Int4B
    +0x008 WaitListHead : struct _LIST_ENTRY, 2 elements, 0x8 bytes
        +0x000 Flink : Ptr32 to
        +0x004 Blink : Ptr32 to
    +0x010 ProfileListHead : struct _LIST_ENTRY, 2 elements, 0x8 bytes
        +0x000 Flink : Ptr32 to
        +0x004 Blink : Ptr32 to
+0x018 DirectoryTableBase : (2 elements) UInt4B
```

Полные выходные данные намного длиннее (согласно заголовку, длина всей структуры – 0x290 байт), но не волнуйтесь, мы рассмотрим важные (с судебной точки зрения) структуры на протяжении этой главы.

Примечание

Ядро Windows отслеживает активные процессы посредством двусвязного списка;

это означает, что каждый процесс «указывает» как на последующий, так и на предыдущий процесс в циклическом списке. Операционная система перечисляет активные процессы, просматривая элемент *PsActiveProcessList* и составляя список известных активных процессов. В структуре EProcess есть элемент *LIST_ENTRY*, который называется *ActiveProcessLinks*. У данного элемента есть два значения, *flink* и *blink*, которые являются указателями на следующий и предыдущий процессы соответственно. Многие инструменты анализа памяти делают то же самое (т. е. просматривают список активных процессов) в файле дампа памяти, тогда как другие перечисляют объекты процесса методом полного перебора (например, скрипт «*lsproc.pl*» и платформа Volatility), указывая даже процессы, завершившие работу. Это очень важно, так как некоторые руткиты (см. главу 7) скрывают процессы путем удаления связей своих процессов из этого двусвязного списка.

Важный элемент процесса, на который указывает структура EProcess, – это блок переменных окружения процесса, или PEB (англ. *process environment block*). Данная структура содержит большое количество информации, но для судебных экспертов важны следующие элементы:

- указатель на структуру данных загрузчика (которая называется *PPEB_LDR_DATA*), содержащую указатели или ссылки на модули (DLL-файлы), используемые процессом;
- указатель на базовый адрес образа, где можно ожидать обнаружить начало исполняемого файла;
- указатель на структуру параметров процесса, которая содержит путь к DLL-файлу, путь к исполняемому файлу и команду, использовавшуюся для запуска процесса.

Эта информация, извлеченная из файла дампа, может оказаться чрезвычайно полезной для эксперта, как будет показано в оставшейся части данной главы.

Механизм создания процесса

Теперь, когда вы немного знаете о различных структурах, связанных с процессами, будет полезно узнать о том, как операционная система использует эти структуры, особенно когда дело касается создания фактических процессов.

При создании процесса выполняется несколько действий. Эти действия можно разделить на шесть этапов (из книги «Внутреннее устройство Microsoft Windows» (*Microsoft Windows Internals*), 4-е издание, глава 6, М. Руссинович и Д. Соломон):

1. Открытие файла образа (.exe), который будет выполнен. Во время этого этапа определяются соответствующие подсистемы (POSIX, MS-DOS, Win 16 и т. д.). Кроме того, проверяется раздел реестра Image File Execution Options (см. главу 4) на наличие параметра Debugger, и, если этот параметр там есть, создание процесса начинается.
2. Создание объекта EProcess. Кроме того, создается блок процесса ядра (KProcess), блок PEB и начальное адресное пространство процесса.
3. Создание первичного потока.
4. Отправка сообщения подсистеме Windows о создании нового процесса и потока; сообщение также содержит идентификатор родительского процесса и флаг, указывающий, относится ли данный процесс к приложениям Windows.
5. Начало выполнения первичного потока. На данном этапе окружение процесса уже определено, а потокам процесса выделены ресурсы.
6. Завершение инициализации адресного пространства в контексте нового процесса и потока.

На данном этапе процесс уже использует пространство в памяти в соответствии со структурой EProcess (которая включает в себя структуру KProcess) и структурой PEB. Процесс имеет как минимум один поток и может начать потреблять дополнительные

ресурсы памяти во время выполнения. Если процесс или память на данном этапе приостановить и создать его или ее дамп, у эксперта будут данные для анализа.

Анализ содержимого дампа памяти

Инструменты, применяющиеся для решения задачи на анализ памяти во время конференции DFRWS 2005, использовали методику, основанную на поиске и перечислении активных процессов. При этом использовались специальные значения/смещения (полученные из системных файлов), чтобы идентифицировать начало списка, а затем просмотреть двусвязный список, пока не будут определены все активные процессы. Местонахождение смещения для начала списка активных процессов было получено из важного системного файла «ntoskrnl.exe».

Андреас Шустер применял другой подход в своем Perl-скрипте «ptfinder.pl». Для решения проблемы он использовал метод перебора всех возможных вариантов посредством определения характеристик процессов в памяти, а затем перечисления блоков EProcess, а также другой информации о процессах, основываясь на этих характеристиках. Андреас начал с перечисления структуры *DISPATCHER_HEADER*, которая расположена по смещению 0 для каждого блока EProcess (фактически, она находится в структуре, известной как блок KProcess). Используя LiveKD, мы видим, что перечисленная структура из ОС Windows 2000 имеет следующие элементы:

+0x000 Header	:	struct _DISPATCHER_HEADER, 6 elements, 0x10 bytes
+0x000 Type	:	UChar
+0x001 Absolute	:	UChar
+0x002 Size	:	UChar
+0x003 Inserted	:	UChar
+0x004 SignalState	:	Int4B

Вкратце, Андреас обнаружил, что некоторые элементы структуры *DISPATCHER_HEADER* неизменны во всех процессах системы. Он исследовал элементы структуры *DISPATCHER_HEADER* для процессов (и потоков) в версиях ОС Windows – с 2000 до первых бета-версий Vista – и обнаружил, что значение *Type* не изменяется во всех версиях операционной системы. Он также обнаружил, что значение *Size* оставалось неизменным во многих версиях операционной системы (например, все процессы в Windows 2000 или XP имели одно и то же значение *Size*), но изменялось в других версиях (например, для Windows 2000 значение *Size* было равно 0x1b, но для первых версий Vista – 0x20).

Используя эту информацию, а также общий размер структуры и то, как можно подразделить саму структуру, Андреас написал Perl-скрипт «ptfinder.pl», который перечислял процессы и потоки в дампе памяти. На конференции DFRWS 2006 он также представил статью «Searching for processes and threads in Microsoft Windows memory dumps» (www.dfrws.org/2006/proceedings/2-Schuster.pdf), в которой рассматривались не только структуры данных, составляющие процессы и потоки, но и различные правила, позволяющие определить, являются ли найденные данные допустимой структурой или просто набором байтов в файле.

Примечание

Осенью 2006 года Ричард МакКуон (<http://forensiczone.blogspot.com/>) разработал внешний графический интерфейс для инструментов PTFinder Андреаса Шустера. Инструменты PTFinder – это набор Perl-скриптов, для выполнения которых необходимо установить интерпретатор языка Perl в системе. (Интерпретатор Perl по умолчанию

устанавливается во многих дистрибутивах Linux, а для платформ Windows его можно бесплатно загрузить с сайта ActiveState.com.)

Инструмент, разработанный Ричардом, может не только определить операционную систему из дампа ОЗУ (избавляя пользователя от необходимости вводить имя системы вручную) при помощи кода, который будет рассмотрен позднее в этой главе, но и обеспечивает графическое представление выходных данных. PTFinderFE имеет несколько интересных применений, особенно в отношении виртуализации.

Весной 2006 года я разработал несколько собственных инструментов, предназначенных для анализа файлов дампа ОЗУ Windows. Так как в то время были доступны примеры дампов памяти для ОС Windows 2000, использовавшиеся в задаче на анализ памяти на конференции DFRWS 2005, я сначала сосредоточил свои усилия на создании кода, который работал для данной платформы. Это позволило мне решить различные проблемы разработки кода, не углубляясь в изучение множества различий между разнообразными версиями ОС Windows. В результате было создано четыре отдельных Perl-скрипта, каждый из которых запускался из командной строки. Все эти скрипты предоставлены на носителе, который идет в комплекте с данной книгой, и мы рассмотрим их здесь.

Примечание

Следующие инструменты («lsproc.pl», «lspd.pl», «lspi.pl» и «lspm.pl») предназначены для использования только с дампами памяти ОС Windows 2000. Как мы уже говорили, существуют значительные различия в формате структуры EProcess между разными версиями Windows (2000, XP, 2003, Vista и т. д.). В связи с этим необходимо выполнить значительную работу, чтобы создать отдельное приложение, которое позволит анализировать дампы памяти из всех версий.

Lsproc.pl

LProc (сокращенно от англ. *list processes* – перечислить процессы) похож на скрипт «ptfinder.pl», разработанный Андреасом Шустером; однако «lsproc.pl» не умеет находить потоки и находит только процессы. В качестве аргумента «lsproc.pl» принимает путь к файлу дампа ОЗУ вместе с его именем:

```
c:\perl\memory>lsproc.pl d:\dumps\drfws1-mem.dmp
```

Выходные данные «lsproc.pl» появляются на консоли (т. е. в стандартном устройстве вывода) в шести столбцах: слово *Proc* (позднее я собираюсь добавить потоки), идентификатор родительского процесса (PPID), идентификатор процесса (PID), имя процесса, смещение структуры процесса в дампе памяти и время создания процесса. Вот фрагмент выходных данных скрипта «lsproc.pl»:

Proc	820	324	helix.exe	0x00306020	Sun	Jun 5	14:09:27	2005
Proc	0	0	Idle	0x0046d160				
Proc	600	668	UMGR32.EXE	0x0095f020	Sun	Jun 5	00:55:08	2005
Proc	324	1112	cmd2k.exe	0x00dcc020	Sun	Jun 5	14:14:25	2005
Proc	668	784	dfrws2005.exe (x)	0x00e1fb60	Sun	Jun 5	01:00:53	2005
Proc	156	176	winlogon.exe	0x01045d60	Sun	Jun 5	00:32:44	2005
Proc	156	176	winlogon.exe	0x01048140	Sat	Jun 4	23:36:31	2005
Proc	144	164	winlogon.exe	0x0104ca00	Fri	Jun 3	01:25:54	2005
Proc	156	180	csrss.exe	0x01286480	Sun	Jun 5	00:32:43	2005
Proc	144	168	csrss.exe	0x01297b40	Fri	Jun 3	01:25:53	2005
Proc	8	156	smss.exe	0x012b62c0	Sun	Jun 5	00:32:40	2005
Proc	0	8	System	0x0141dc60				
Proc	668	784	dfrws2005.exe (x)	0x016a9b60	Sun	Jun 5	01:00:53	2005

Proc	1112	1152	dd.exe(x)	0x019d1980	Sun	Jun 5	14:14:38	2005
Proc	228	592	dfrws2005.exe	0x02138640	Sun	Jun 5	01:00:53	2005
Proc	820	1076	cmd.exe	0x02138c40	Sun	Jun 5	00:35:18	2005
Proc	240	788	metasploit.exe(x)	0x02686cc0	Sun	Jun 5	00:38:37	2005
Proc	820	964	Apoint.exe	0x02b84400	Sun	Jun 5	00:33:57	2005
Proc	820	972	HKserv.exe	0x02bf86e0	Sun	Jun 5	00:33:57	2005
Proc	820	988	DragDrop.exe	0x02c46020	Sun	Jun 5	00:33:57	2005
Proc	820	1008	alogserv.exe	0x02e7ea20	Sun	Jun 5	00:33:57	2005
Proc	820	972	HKserv.exe	0x02f806e0	Sun	Jun 5	00:33:57	2005
Proc	820	1012	tgcmd.exe	0x030826a0	Sun	Jun 5	00:33:58	2005
Proc	176	800	userinit.exe(x)	0x03e35020	Sun	Jun 5	00:33:52	2005
Proc	800	820	Explorer.Exe	0x03e35ae0	Sun	Jun 5	00:33:53	2005
Proc	820	1048	PcfMgr.exe	0x040b4660	Sun	Jun 5	00:34:01	2005

Первый процесс, перечисленный в выходных данных скрипта «lsproc.pl», – это «helix.exe». Согласно информации, предоставленной на веб-сайте с условиями задачи на анализ памяти с конференции DFRWS 2005, для создания дампа памяти использовался загрузочный компакт-диск (Live CD) Helix.

В предыдущем списке показана только часть выходных данных «lsproc.pl». Всего в файле дампа памяти было найдено 45 процессов. Обратите внимание, что после имени некоторых процессов в выходных данных стоит символ *x* в скобках. Это означает, что процесс завершился.

Примечание

Присмотревшись, вы заметите несколько интересных особенностей в выходных данных скрипта «lsproc.pl». Первая из них заключается в том, что, согласно дате создания, процесс «csrss.exe» (с идентификатором 168) был создан раньше других процессов из списка. Изучив данные еще тщательнее, вы увидите, что-то похожее для процессов «winlogon.exe» (с идентификаторами 164 и 176). Андреас Шустер также обратил на это внимание, и, согласно статье о сохраняемости данных в его блоге (http://computer.forensikblog.de/en/2006/04/persistence_through_the_boot_process.html), было определено, что загрузка системы, из которой получен дамп памяти, была выполнена в воскресенье, 5 января 2005 года, приблизительно в 00:32:27. Итак, откуда взялись эти процессы?

Как указывает Андреас в своем блоге, не имея более определенной информации о состоянии испытательной системы до сбора данных для задачи на анализ памяти, тяжело полностью ответить на этот вопрос. Однако характеристики системы были известны и задокументированы, и было отмечено, что во время сбора данных произошел сбой системы.

Вполне возможно, что данные сохранились после перезагрузки. Кажется, нет документов, в которых указано, что после выключения или сбоя системы Windows, содержимое физической памяти должно обнуляться или стираться тем или иным способом. Поэтому, возможно, что содержимое физической памяти остается в предыдущем состоянии, и, если оно не перезаписывается после перезагрузки системы, данные все еще доступны для анализа. Во многих версиях BIOS есть возможность перезаписывать память во время загрузки в рамках проверки ОЗУ, но эта возможность обычно отключена, чтобы ускорить процесс загрузки.

Несомненно, это та область, которая требует дальнейшего исследования. Как утверждает Андреас (http://computer.forensikblog.de/en/2006/04/data_lifetime.html), эта область исследования имеет «блестящее будущее».

Lspd.pl

«Lspd.pl» – это Perl-скрипт, позволяющий перечислять подробности о процессах. Как и другие рассматриваемые здесь инструменты, «lspd.pl» – это Perl-скрипт,

запускаемый из командной строки, который получает свою информацию на основе выходных данных скрипта «lsproc.pl». В частности, «lspd.pl» принимает два аргумента: полный путь к файлу дампа и физическое смещение процесса, который представляет для вас интерес (последний аргумент можно получить из выходных данных скрипта «lsproc.pl»). Хотя скрипту «lsproc.pl» требуется некоторое время, чтобы проанализировать содержимое файла дампа, «lspd.pl» работает намного быстрее, потому что вы точно указываете ему место в файле, в которое нужно перейти, чтобы получить информацию.

Давайте рассмотрим отдельный процесс. В данном случае мы исследуем «dd.exe», процесс с идентификатором 284. Чтобы получить подробную информацию об этом процессе при помощи скрипта «lspd.pl», используйте следующую команду:

```
c:\perl\memory>lspd.pl d:\dumps\dfrws1-mem.dmp 0x0414dd60
```

Примечание

Выше был показан только фрагмент выходных данных скрипта «lsproc.pl»; я не стал перечислять выходные данные полностью просто потому, что данный фрагмент показывает достаточно информации, чтобы я мог изложить свою точку зрения. Процесс, указанный в командной строке «lspd.pl» (т. е. находящийся по смещению 0x0414dd60), не перечислен в данном фрагменте, однако он присутствует в полных выходных данных скрипта «lsproc.pl».

Обратите внимание, что, используя «lspd.pl», мы указываем два аргумента: путь к файлу дампа (вместе с именем этого файла) и физическое смещение в файле дампа, где мы обнаружили процесс при помощи «lsproc.pl». Мы будем рассматривать выходные данные скрипта «lspd.pl» частями, начиная с полезной информации, извлеченной непосредственно из самой структуры EProcess:

Process Name	:	dd.exe
PID	:	284
Parent PID	:	1112
TFLINK	:	0xffff2401c4
TBLINK	:	0xffff2401c4
FLINK	:	0x8046b980
BLINK	:	0xffff1190c0
SubSystem	:	4.0
Exit Status	:	259
Create Time	:	Sun Jun 5 14:53:42 2005
Exit Called	:	0
DTB	:	0x01d9e000
ObjTable	:	0xffff158708 (0x00eb6708)
PEB	:	0x7ffd000 (0x02c2d000)
InheritedAddressSpace	:	0
ReadImageFileExecutionOptions	:	0
BeingDebugged	:	0
CSDVersion	:	Service Pack 1
Mutant	=	0xffffffff
Img Base Addr	=	0x00400000 (0x00fee000)
PEB_LDR_DATA	=	0x00131e90 (0x03a1ee90)
Params	=	0x00020000 (0x03a11000)

Примечание

Ранее в этой главе я говорил, что список активных процессов в работающей системе хранится в двусвязном списке. Значения *flink* и *blink*, отображаемые в выходных данных скрипта «lspd.pl» выше, указывают на следующий и предыдущий процессы соответственно. Как показано в выходных данных скрипта «lspd.pl», это указатели на

адреса в памяти, а не на физические адреса или смещения в файле дампа.

Скрипт «lspd.pl» также следует указателям, предоставленным структурой EProcess, чтобы в добавок собрать другие данные. Например, мы видим путь к исполняемому файлу и команду, использовавшуюся для запуска процесса (выделено полужирным):

```

Current Directory Path      = E:\Shells\
DllPath                   = E:\Acquisition\FAU;.;C:\WINNT\System32;C:\WINNT\system;
                             C:\WINNT;E:\Acquisition\FAU\;E:\Acquisition\GNU\;
                             E:\Acquisition\CYGWIN\;E:\IR\bin\;E:\IR\WFT;E:\IR\
                             windbg\;E:\IR\Foundstone\;E:\IR\Cygwin;E:\IR\
                             somarsoft\;E:\IR\sysinternals\;E:\IR\ntsecurity\;
                             E:\IR\perl\;E:\Static-Binaries\gnu_utils_win32\;C:\WINNT\
                             system32;C:\WINNT;C:\WINNT\System32\Wbem
ImagePathName             = E:\Acquisition\FAU\dd.exe
Command Line               = ..\Acquisition\FAU\dd.exe if=\\.\\PhysicalMemory of=F:\\
                             intrusion2005\physicalmemory.dd conv=noerror --md5sum
                             --verifymd5 --md5out=F:\\intrusion2005\physicalmemory.dd.
                             md5 --log=F:\\intrusion2005\audit.log
Environment Offset         = 0x00000000 (0x00000000)
Window Title               = ..\Acquisition\FAU\dd.exe if=\\.\\PhysicalMemory of=F:\\
                             intrusion2005\physicalmemory.dd conv=noerror --md5sum
                             --verifymd5 --md5out=F:\\intrusion2005\physicalmemory.dd.
                             md5 --log=F:\\intrusion2005\audit.log
Desktop Name               = WinSta0\Default

```

Кроме того, скрипт «lspd.pl» находит список имен различных модулей, используемых процессом, и имеющиеся дескрипторы (дескрипторы файлов и т. д.) в памяти. Например, «lspd.pl» обнаружил, что с процессом «dd.exe» связан следующий открытый дескриптор файла:

```

Type : File
Name = \intrusion2005\audit.log

```

Как видно из данных выше, файл \intrusion\audit.log находится на накопителе F:\\ и представляет собой файл выходных данных для журнала действий, выполненных инструментом «dd.exe», что объясняет, почему он перечислен как открытый дескриптор файла, используемый этим процессом. Используя такую информацию, извлеченную из других процессов, вы можете получить представление о файлах, на которые нужно обратить внимание во время расследования. В данном конкретном случае можно предположить, что накопитель E:\\, указанный в значении *ImagePathName*, – это дисковод CD-ROM, так как Helix можно запустить с компакт-диска. Это можно подтвердить, проверив параметры реестра в образе системы, о которой идет речь (однако образ системы не предоставляется в рамках задачи на анализ памяти). Можно также использовать подобную информацию, чтобы узнать немного больше о накопителе F:\\. Эта информация будет рассматриваться в главе 4.

В завершение анализа скрипт «lspd.pl» переходит к месту, указанному в значении *Image Base Addr* (после того как это значение будет преобразовано из виртуального адреса в физическое смещение в файле дампа памяти), и проверяет, находится ли по этому адресу действительный исполняемый образ. Эта проверка очень простая: скрипт всего лишь считывает первые два байта начиная с преобразованного адреса, чтобы увидеть, содержат ли они символы *MZ*. Эту проверку нельзя считать окончательной, но PE-файлы (с расширениями .exe, .dll, .ocs, .sys и другими подобными) начинаются с инициалов Марка Збиковского (Mark Zbikowski), одного из первых разработчиков MS-DOS и Windows NT. Формат PE-файла и его заголовка более подробно рассматривается в главе 6.

Совет

Если вы создали дамп содержимого физической памяти из ОС Windows 2000 или XP при помощи «winen.exe», и у вас есть электронный ключ EnCase, можно проанализировать информацию о процессах из дампа памяти при помощи скриптов EnScript, написанных пользователем TK_Lane и доступных в блоге «EDD and Forensics» (<http://eddandforensics.blogspot.com/2008/04/windows-memory-analysis.html>).

Volatility Framework

Аарон Уолтерс предоставляет ценную информацию о платформе Volatility Framework в своей презентации с семинара OMFW, доступной по адресу https://www.volatilesystems.com/volatility/omfw/Walters_OMFW_2008.pdf.

Файл «readme.txt», являющийся частью дистрибутива Volatility (версии 1.3 beta на момент написания этой книги), содержит много информации о способах использования платформы, доступных типах команд и имеющихся возможностях, а также примеры запуска различных команд. Аарон разработал Volatility, чтобы использовать некоторые команды, которые обычно применяются при расследовании инцидентов. Например, чтобы получить список выполняющихся процессов из дампа памяти, Volatility использует команду *pslist*. Прежде чем начать работу с Volatility, обязательно прочитайте файл «readme.txt», чтобы узнать, какие виды информации можно получить из дампа памяти ОС Windows XP SP2 или SP3.

Чтобы показать, какая информация доступна в дампе необработанных данных памяти, давайте рассмотрим пример; в данном случае мы имеем дело с дампом памяти, который имеет размер 512 Мб и получен из ноутбука с ОС Windows XP SP2. Мы можем узнать основные сведения о дампе памяти, используя команду *ident*:

```
D:\Volatility>python volatility ident -f d:\hacking\xp-laptop1.img
      Image Name      : d:\hacking\xp-laptop1.img
      Image Type     : Service Pack 2
      VM Type       : nopsae
      DTB           : 0x39000
      Datetime      : Sat Jun 25 12:58:47 2005
```

Эта информация может быть очень полезна при документировании анализа дампа памяти. Используя команду *pslist*, можно получить список активных процессов из дампа памяти в формате, похожем на тот, что мы видели при выполнении инструмента «*pslist.exe*» в работающей системе:

```
D:\Volatility>python volatility pslist -f d:\hacking\xp-laptop1.img
      Name        Pid   PPid   Thds   Hnds   Time
      System      4      0      61    1140   Thu Jan  1  00:00:00 1970
      smss.exe    448     4      3     21    Sat Jun 25 16:47:28 2005
      csrss.exe   504     448    12    596   Sat Jun 25 16:47:30 2005
      winlogon.exe 528     448    21    508   Sat Jun 25 16:47:31 2005
      services.exe 580     528    18    401   Sat Jun 25 16:47:31 2005
      lsass.exe   592     528    21    374   Sat Jun 25 16:47:31 2005
      svchost.exe 740     580    17    198   Sat Jun 25 16:47:32 2005
      svchost.exe 800     580    10    302   Sat Jun 25 16:47:33 2005
      svchost.exe 840     580    83   1589   Sat Jun 25 16:47:33 2005
      Smc.exe     876     580    22    423   Sat Jun 25 16:47:33 2005
      svchost.exe 984     580     6     90    Sat Jun 25 16:47:35 2005
      svchost.exe 1024    580    15    207   Sat Jun 25 16:47:35 2005
      spoolsv.exe 1224    580    12    136   Sat Jun 25 16:47:39 2005
      ssonsvr.exe 1632    1580    1     24    Sat Jun 25 16:47:46 2005
      explorer.exe 1812    1764    22    553   Sat Jun 25 16:47:47 2005
      Directcd.exe 1936    1812    4     40    Sat Jun 25 16:47:48 2005
      TaskSwitch.exe 1952    1812    1     21    Sat Jun 25 16:47:48 2005
```

Fast.exe	1960	1812	1	22	Sat	Jun 25	16:47:48	2005
VPTray.exe	1980	1812	2	89	Sat	Jun 25	16:47:49	2005
atiptaxx.exe	2040	1812	1	51	Sat	Jun 25	16:47:49	2005
jusched.exe	188	1812	1	22	Sat	Jun 25	16:47:49	2005
EM_EXEC.exe	224	112	2	74	Sat	Jun 25	16:47:50	2005
ati2evxx.exe	432	580	4	38	Sat	Jun 25	16:47:55	2005
Crypserv.exe	688	580	3	34	Sat	Jun 25	16:47:55	2005
DefWatch.exe	864	580	3	27	Sat	Jun 25	16:47:55	2005
msdtc.exe	1076	580	14	166	Sat	Jun 25	16:47:55	2005
Rtvscan.exe	1304	580	37	300	Sat	Jun 25	16:47:58	2005
tcpsvcs.exe	1400	580	2	94	Sat	Jun 25	16:47:58	2005
snmp.exe	1424	580	5	192	Sat	Jun 25	16:47:58	2005
svchost.exe	1484	580	6	119	Sat	Jun 25	16:47:59	2005
wdfmgr.exe	1548	580	4	65	Sat	Jun 25	16:47:59	2005
Fast.exe	1700	580	2	32	Sat	Jun 25	16:48:01	2005
mqsvc.exe	1948	580	23	205	Sat	Jun 25	16:48:02	2005
mqtgsvc.exe	2536	580	9	119	Sat	Jun 25	16:48:05	2005
alg.exe	2868	580	6	108	Sat	Jun 25	16:48:11	2005
wuauctl.exe	2424	840	4	160	Sat	Jun 25	16:49:21	2005
firefox.exe	2160	1812	6	182	Sat	Jun 25	16:49:22	2005
PluckSvr.exe	944	740	9	227	Sat	Jun 25	16:51:00	2005
iexplore.exe	2392	1812	9	365	Sat	Jun 25	16:51:02	2005
PluckTray.exe	2740	944	3	105	Sat	Jun 25	16:51:10	2005
PluckUpdater.ex	3076	1812	0	-1	Sat	Jun 25	16:51:15	2005
PluckUpdater.ex	1916	944	0	-1	Sat	Jun 25	16:51:40	2005
PluckTray.exe	3256	1812	0	-1	Sat	Jun 25	16:54:28	2005
cmd.exe	2624	1812	1	29	Sat	Jun 25	16:57:36	2005
wmiprvse.exe	4080	740	7	0	Sat	Jun 25	16:57:53	2005
PluckTray.exe	3100	1812	0	-1	Sat	Jun 25	16:57:59	2005
dd.exe	4012	2624	1	22	Sat	Jun 25	16:58:46	2005

Используя похожие команды (*psscan* или *psscan2*), можно получить информацию обо всех объектах процессов, в том числе о завершенных процессах. Команды для получения информации обо всех объектах (сетевые соединения, процессы и т. д.) работают медленнее, так как они используют способ линейного сканирования, чтобы полностью просмотреть файл дампа памяти, исследуя все возможные объекты, вместо того чтобы использовать отдельные смещения, предоставленные операционной системой (см. обсуждение программы LiveKD выше).

При расследовании случаев вторжения или взлома большинство экспертов считает целесообразным изучить сведения о сетевых соединениях. Список активных сетевых соединений (как при запуске команды *netstat -ano*) из дампа памяти можно получить, используя команду *connections*, как показано ниже:

```
D:\Volatility>python volatility connections -f d:\hacking\xp-laptop1.img
Local Address      Remote Address      Pid
127.0.0.1:1056    127.0.0.1:1055    2160
127.0.0.1:1055    127.0.0.1:1056    2160
192.168.2.7:1077  64.62.243.144:80   2392
192.168.2.7:1082  205.161.7.134:80   2392
192.168.2.7:1066  199.239.137.200:80  2392
```

Более того, можно исследовать файл дампа памяти на наличие признаков объектов сетевых соединений, в частности на наличие сетевых соединений, которые, возможно, были закрыты на момент создания дампа памяти:

```
D:\Volatility>python volatility connscan2 -f d:\hacking\xp-laptop1.img
Local Address      Remote Address      Pid
-----
192.168.2.7:1115  207.126.123.29:80  1916
3.0.48.2:17985    66.179.81.245:20084 4287933200
```

192.168.2.7:1164	66.179.81.247:80	944
192.168.2.7:1082	205.161.7.134:80	2392
192.168.2.7:1086	199.239.137.200:80	1916
192.168.2.7:1162	170.224.8.51:80	1916
127.0.0.1:1055	127.0.0.1:1056	2160
192.168.2.7:1116	66.161.12.81:80	1916
192.168.2.7:1161	66.135.211.87:443	1916
192.168.2.7:1091	209.73.26.183:80	1916
192.168.2.7:1151	66.150.96.111:80	1916
192.168.2.7:1077	64.62.243.144:80	2392
192.168.2.7:1066	199.239.137.200:80	2392
192.168.2.7:1157	66.151.149.10:80	1916
192.168.2.7:1091	209.73.26.183:80	1916
192.168.2.7:1115	207.126.123.29:80	1916
192.168.2.7:1155	66.35.250.150:80	1916
127.0.0.1:1056	127.0.0.1:1055	2160
192.168.2.7:1115	207.126.123.29:80	1916
192.168.2.7:1155	66.35.250.150:80	1916

Volatility – мощная платформа с открытым исходным кодом, возможности которой можно расширять посредством разработки дополнительных модулей (важным требованием является знание языка программирования Python). Брендан Долан-Гавитт (известный также как Moyix) создал модуль Volatility для поиска оконных сообщений (<http://moyix.blogspot.com/2008/09/window-messages-as-forensic-resource.html>), которые представляют различные события, созданные приложениями с графическим интерфейсом в ОС Windows, и обрабатываются в очереди сообщений. Как отмечает Брендан, приложение может быть написано плохо и может неправильно обрабатывать свои собственные сообщения. В таком случае вы, возможно, сможете найти остатки этих сообщений в дампе памяти. Эта информация может быть очень полезна во время судебной экспертизы.

Совет

Брендан также разработал несколько подключаемых модулей для Volatility, позволяющих получать доступ к данным реестра, находящимся в дампах памяти Windows (статья в его блоге – <http://moyix.blogspot.com/2009/01/memory-registry-tools.html>, обновления для кода – <http://moyix.blogspot.com/2009/01/registry-code-updates.html>). В своем блоге (<http://forensiczone.blogspot.com/2009/01/using-volatility-1.html>) Ричард МакКуон показывает, как использовать эти модули для извлечения паролей из файла куста реестра SAM (англ. *Security Account Manager* – диспетчер учетных записей безопасности), находящегося в памяти, чтобы можно было взломать эти пароли, используя одну из предназначенных для этого программ.

Чтобы использовать платформу Volatility и модули, разработанные Бренданом, для извлечения паролей из файлов кустов реестра, находящихся в памяти, нужно установить модули PyCrypto (доступные в виде предварительно собранных исполняемых файлов Windows на странице www.voidspace.org.uk/python/modules.shtml#pycrypto).

Кроме того, Джесси Корнблюм разработал два модуля: *suspicious*, выполняющий поиск подозрительных записей в командах для запуска процессов, и *cryptoscan*, выполняющий поиск парольных фраз TrueCrypt. Последний модуль может быть чрезвычайно полезным для эксперта, так как TrueCrypt (www.truecrypt.org/) – мощное бесплатное приложение, которое можно использовать для шифрования томов и накопителей.

Платформа Volatility обладает множеством других функций, помимо простого анализа необработанных данных из дампов памяти. Благодаря усилиям Маттье Свиша (www.msuiche.net/) Volatility также имеет возможность обрабатывать файлы спящего

режима. Эта возможность впервые использовалась в рамках проекта Sandman, а затем стала неотъемлемой частью платформы Volatility. В декабре 2008 года Мэттье выпустил отдельную альфа-версию оболочки для работы файлами спящего режима, которая называлась hibrshell (www.msuiche.net/hibrshell/). По имеющимся данным, эта версия hibrshell работает с файлами спящего режима из ОС Windows XP, 2003, Vista и 2008.

Совет

Независимо от того, какая платформа используется для анализа файла спящего режима, этот файл предоставляет эксперту несколько недоступных ранее возможностей. Во-первых, его можно использовать как источник данных о состоянии работающей системы на определенный момент времени в прошлом. Эта возможность может быть чрезвычайно полезной во время анализа вредоносных программ, а также при составлении временной шкалы для вторжения, особенно если у эксперта есть дамп текущей памяти для анализа. В условиях, когда упомянутые выше инструменты (например, «mdd.exe» и т. д.) нельзя использовать для создания дампа содержимого физической памяти из системы, эксперт, возможно, сможет перевести систему в спящий режим, чтобы создать дамп памяти, который впоследствии можно проанализировать.

Кроме того, Volatility можно использовать для анализа файлов аварийного дампа памяти, а также для преобразования дампа необработанных данных памяти в формат аварийного дампа, чтобы эксперт мог использовать средства отладки от Microsoft.

К настоящему моменту вам должно быть понятно, какими чрезвычайно мощными возможностями обладает платформа Volatility и какое количество информации можно извлечь из дампа памяти. Чтобы помочь в сопоставлении некоторых данных, которые можно получить при использовании Volatility, Джейми Леви (Jamie Levy) написала Perl-скрипт «vol2html.pl» (<http://gleeda.blogspot.com/2008/11/vol2html-perl-script.html>). Данный скрипт принимает выходные данные команд *pslist*, *files* и *dlllist* платформы Volatility и сопоставляет их в HTML-отчете, пример которого можно посмотреть на странице <http://venus.cs.qc.edu/~jlevy/code/report/index.html>. Как и знакомый инструмент «listdlls.exe» с веб-сайта Microsoft (Sysinternals), команда *dlllist* платформы Volatility включает командную строку процесса в выходные данные; эта командная строка также появляется в HTML-отчете скрипта «vol2html.pl».

Примеры файлов дампов памяти из ОС Windows XP доступны на странице конкурса «Судебное rodeo», проводившегося во время конференции DFRWS 2008 (www.dfrws.org/2008/rodeo.shtml), а также на странице сайта Национального института стандартов и технологий (www.cfreds.nist.gov/mem/Basic_Memory_Images.html).

Майкл Хейл Лай (Michael Hale Ligh) написал две статьи о своем успешном опыте использования платформы Volatility Framework, особенно в отношении анализа вредоносных программ; см. «Recovering CoreFlood Binaries with Volatility» (<http://mnin.blogspot.com/2008/11/recovering-coreflood-binaries-with.html>) и «Locating Hidden Clampi DLLs (VAD-style)» (<http://mnin.blogspot.com/2008/11/locating-hidden-clampi-dlls-vad-style.html>). Обе эти статьи являются отличным примером того, как Volatility Framework может увеличить возможности эксперта.

Memoryze

Программа Memoryze от компании Mandiant предоставляет эксперту возможность анализировать дампы памяти, полученные из нескольких версий ОС Windows. Чтобы установить Memoryze, загрузите и запустите MSI-файл с веб-сайта компании Mandiant (упоминавшегося ранее в этой главе). Я выбрал для установки каталог D:\Mandiant. Затем, чтобы установить Audit Viewer, загрузите zip-архив этого инструмента, а также зависимые компоненты (т. е. Python 2.5 или 2.6, расширение для графического интерфейса wxPython)

как описано на веб-сайте компании Mandiant (если вы ранее установили и запускали платформу Volatility, ваша система уже поддерживает язык программирования Python). Я распаковал файлы инструмента Audit Viewer в каталог D:\Mandiant\AV.

Для демонстрации использования Memoryze и Audit Viewer мы сначала выберем дамп памяти из ОС Windows 2003: «boomer-win2003.img». Чтобы начать анализ дампа памяти, мы извлечем из него различные данные, запустив Memoryze:

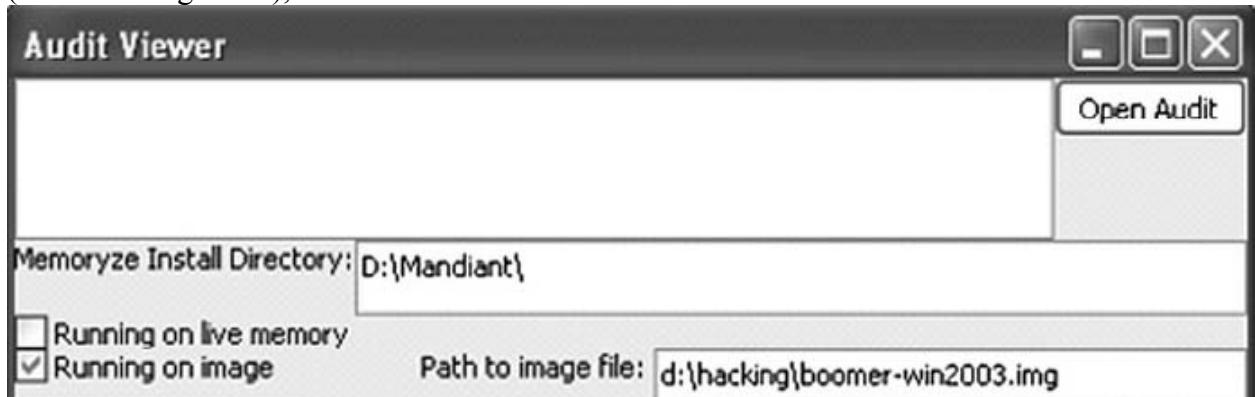
```
D:\mandiant>process.bat -input d:\hacking\boomer-win2003.img -ports true -handles true -sections true
```

Предыдущая команда указывает программе Memoryze проанализировать информацию о процессах и получить сведения о портах, дескрипторах и разделах памяти (я специально решил не извлекать строки из каждого процесса) для процессов из списка активных процессов. Ниже показан полный набор параметров для файла «process.bat»:

```
Usage: process.bat
-input      name of snapshot. Exclude for live memory.
-pid        PID of the process to inspect. Default: 4294967295 = All
-process    optional name of the process to inspect. Default: Excluded
-handles    true|false inspect all the process handles. Default: false
-sections   true|false inspect all process memory ranges. Default: false
-ports      true|false inspect all the ports of a process. Default: false
-strings    true|false inspect all the strings of a process. Default: false
-output     directory to write the results. Default .\Audits
```

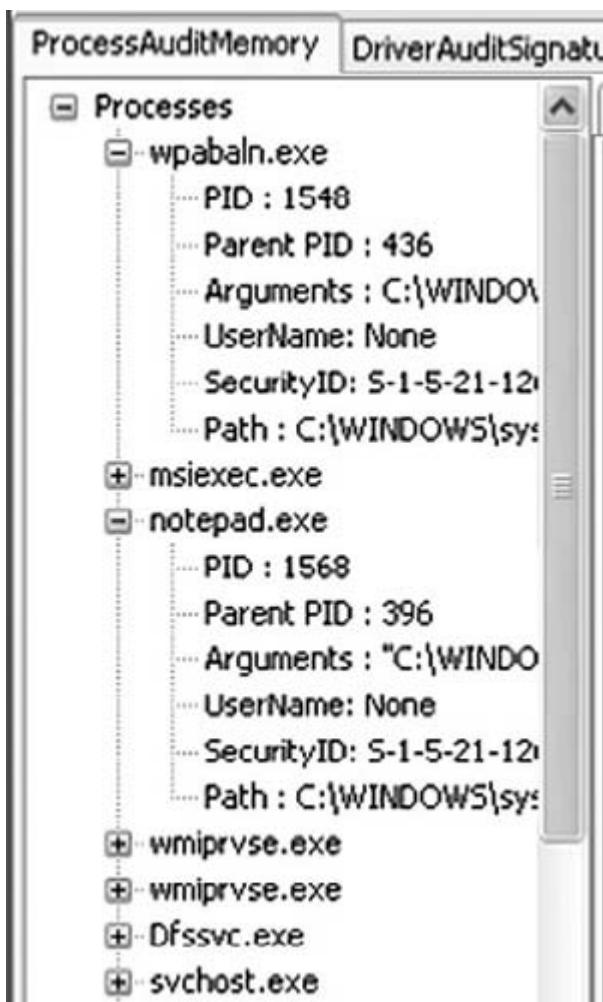
Предыдущая команда по умолчанию помещает итоговый XML-файл в каталог .\Audit. В данном случае полный путь будет D:\mandiant\Audits\WINTERMUTE\20090103134554.

Audit Viewer от компании Mandiant – инструмент с графическим интерфейсом пользователя для исследования XML-файлов, созданных программой Memoryze. Чтобы запустить Audit Viewer, дважды щелкните по файлу «**AuditViewer.py**» в каталоге, где был распакован архив, загруженный с веб-сайта Mandiant. Так как вы установили модули wxPython, вы увидите графический интерфейс запустившегося инструмента Audit Viewer. После этого нужно настроить инструмент, изменив при необходимости путь в поле «**Каталог установки Memoryze**» (“Memoryze Install Directory”), включив кнопку-флажок «**Работа с образом**» (“Running on image”) и указав путь в поле «**Путь к файлу-образу**» (“Path to image file”), как показано на илл. 3.11.



Илл. 3.11. Интерфейс Audit Viewer с измененными параметрами конфигурации.

Сделав необходимые изменения, нажмите кнопку «**Открыть аудит**» (“Open Audit”) в верхней части интерфейса Audit Viewer и перейдите к каталогу, где были созданы XML-файлы аудита. После того как будет выбран каталог, инструмент Audit Viewer проанализирует имеющиеся файлы и заполнит данными дерево процессов в пользовательском интерфейсе, как показано на илл. 3.12.



Илл. 3.12. Интерфейс Audit Viewer с деревом процессов.

Для каждого из двух развернутых процессов на илл. 3.12., показаны идентификатор процесса (PID), идентификатор родительского процесса (PPID), аргументы (или командная строка), а также другие сведения. Чтобы подробнее исследовать каждый процесс, дважды щелкните по имени процесса в дереве, а затем просмотрите содержимое различных вкладок («Файлы» (“Files”), «Каталоги» (“Directories”) и т. д.), отображаемых в окне инструмента Audit Viewer, как показано на илл. 3.13.

Files	Directories	Processes	Keys	Mutants	Events	Dlls	Strings	Memory Sections	Ports
Object Ad...									
	File								
0x85b786c8L	\Documents and Settings\Administrator								
0x85ef7890L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85b80398L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85e9f360L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85b816d0L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85faece8L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85bb5f90L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85b6fa40L	\WINDOWS\WinSxS\x86.Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.100.0_x-ww_84174508								
0x85b5e158L	\Isarp								
0x85b58028L	\Documents and Settings\All Users\Desktop								
0x85b59d98L	\Documents and Settings\Administrator\Desktop								

Илл. 3.13. Интерфейс Audit Viewer с вкладками подробностей о процессах.

Memoryze и Audit Viewer предоставляют ряд дополнительных возможностей для эксперта. Например, исходя из полученных данных в Audit Viewer, вы, возможно, решите, что нужно создать образ исполняемого файла процесса из файла-образа. Для этого используйте пакетный файл «processdd.bat», как показано ниже:

```
D:\mandiant\processdd.bat -pid PID -input d:\hacking\boomer-win2003.img
```

Можно также использовать другие пакетные файлы, предоставляемые с программой Memoryze, чтобы обнаружить руткиты и перехваты, а также выполнить поиск драйверов (www.mandiant.com/software/usememoryze.htm). В блоге M-union компании Mandiant (<http://blog.mandiant.com/>) показаны дополнительные примеры использования Memoryze и Audit Viewer, например, способ использования этих инструментов в приложении для судебного анализа EnCase от компании Guidance Software.

Responder

Responder от компании HBGary – это коммерческая программа для анализа дампа памяти, которая имеет графический интерфейс. На веб-сайте компании (www.hbgary.com) она описывается как «программный пакет анализа памяти и среды выполнения, используемый для обнаружения, диагностики и расследования современных компьютерных угроз». Как и другие инструменты анализа, программа Responder (версий Professional и Field) позволяет эксперту анализировать дамп памяти, не используя API взломанной или зараженной системы. Хотя программа Responder была написана для анализа вредоносных программ, она также предоставляет богатые возможности для расследования инцидентов, позволяя специалистам быстро и легко получить необходимую информацию.

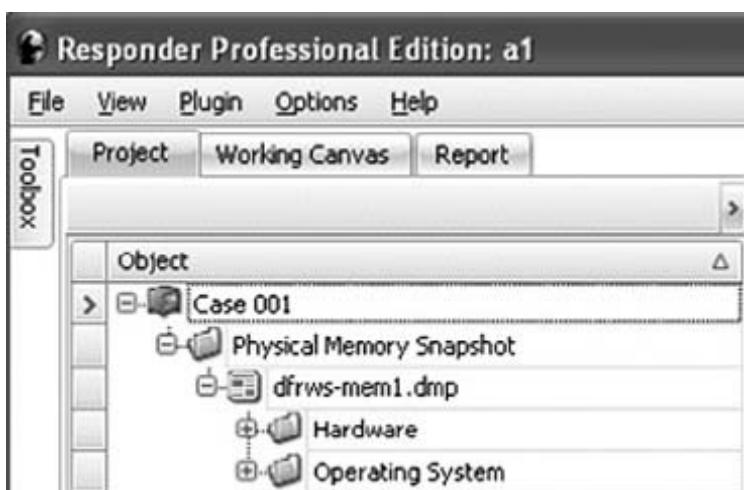
Примечание

В примерах, перечисленных в этом разделе главы, использовалась пробная версия Responder Professional Edition 1.3.0.377. Однако функциональные возможности, представленные и наблюдаемые в этом разделе, присущи обеим версиям программы (Professional и Field). Мы не будем делать всесторонний обзор Responder Professional (к возможностям данной версии относятся дизассемблирование исполняемых файлов, анализ графа потока управления и обратная разработка), так это выходит за рамки данной книги, и сосредоточим наше внимание на тех особенностях программы, которые непосредственно относятся к анализу дампов памяти в рамках расследования инцидента.

Для того чтобы начать анализ дампа памяти нужно всего лишь создать дело и импортировать снимок физической памяти: выбрать элемент «Файл» (‘File’) в строке меню, щелкнуть по пункту «Импортировать» (“Import”), а затем – «Импортировать снимок физической памяти» (“Import a Physical Memory Snapshot”). В данном примере мы будем использовать дамп памяти ОС Windows 2000 из задачи на анализ памяти, предложенной на конференции DFRWS 2005; однако, как и программа Memoryze, Responder работает с дампами памяти из всех, в том числе последних, версий ОС Windows.

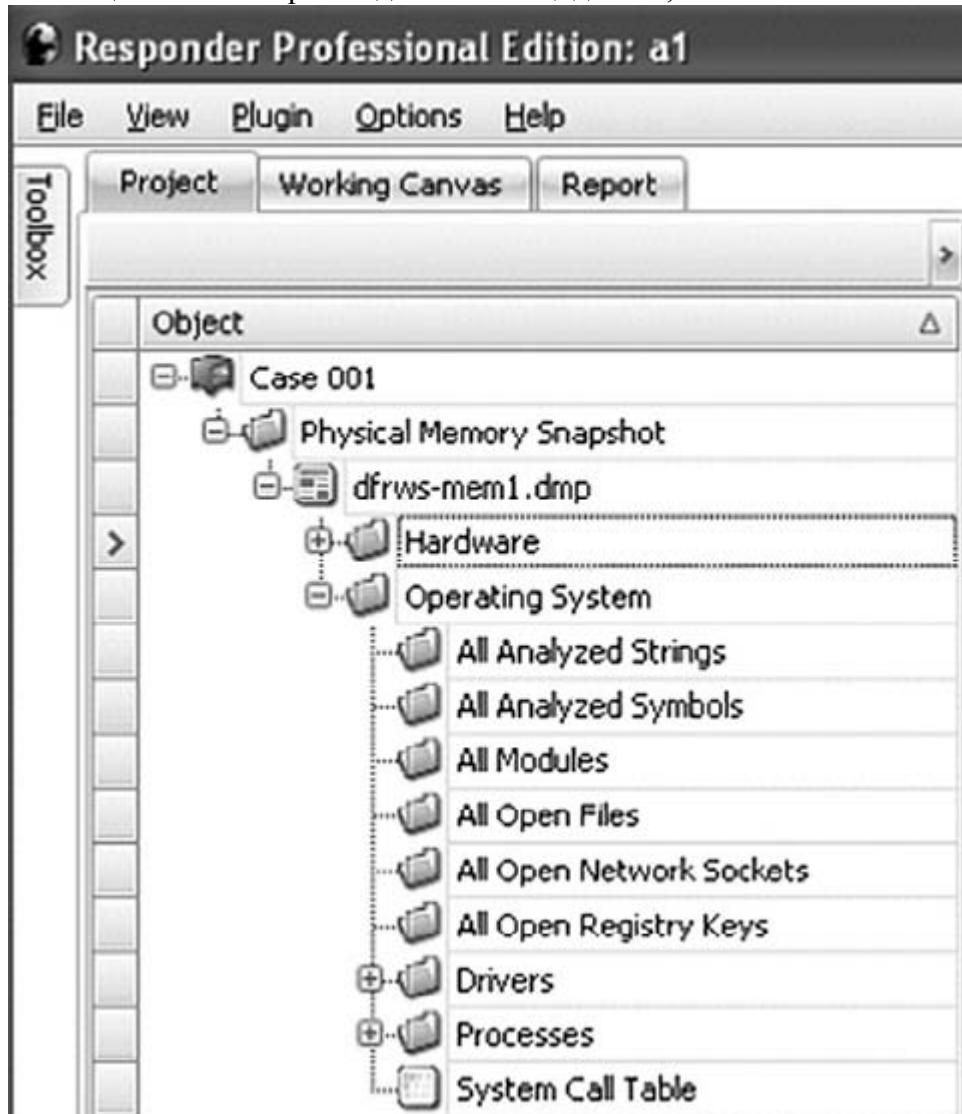
При импортировании «снимка» физической памяти в дело программы Responder пользователю доступна возможность извлечь и анализировать все подозрительные исполняемые файлы. Правила, используемые программой Responder для определения подозрительных объектов, находятся в текстовом файле, который можно открыть и просмотреть и в котором можно даже добавлять и удалять комментарии, уменьшая количество ошибочных результатов. Кроме того, в файл можно добавлять записи, исходя из опыта, увеличивая таким образом эффективность программы.

После того как файл дампа памяти будет импортирован и проанализирован, Responder покажет в левой панели этот файл вместе с двумя папками, «Аппаратные средства» (“Hardware”) и «Операционная система» (“Operating System”), см. илл. 3.14.



Илл. 3.14. Файл дампа памяти, импортированный в проект программы Responder.

Развернув папку «Аппаратные средства» («Hardware»), вы увидите таблицу прерываний. Развернув папку «Операционная система» («Operating System»), вы увидите много подпапок с дополнительной информацией, в том числе «Процессы» («Processes ») и «Все открытые сетевые сокеты» («All Open Network Sockets»), которые интересуют большинство специалистов по расследованию инцидентов, как показано на илл. 3.15.



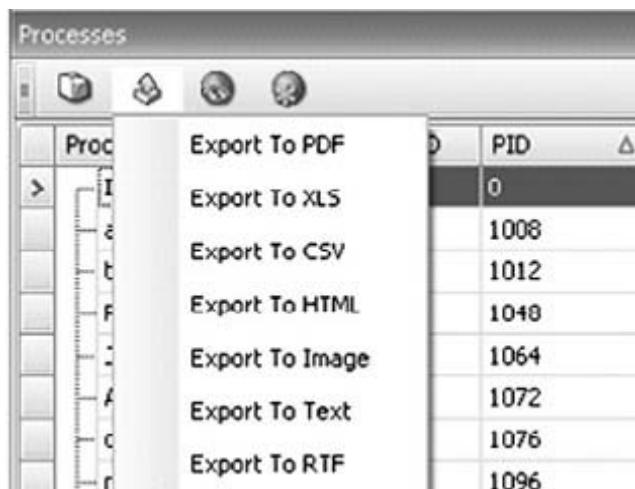
Илл. 3.15. Развернутая папка «Операционная система» («Operating System») в программе Responder.

Затем можно развернуть папку «Процессы» (“Processes”), чтобы увидеть все активные процессы, извлеченные из дампа памяти, или дважды щелкнуть по этой папке, чтобы отобразить все процессы и подробную информацию о них в правой панели программы Responder, как показано на илл. 3.16.

nc.exe	592	1096	"c:\winnt\system32\nc.exe" -L -p 3000 -t -e cmd.exe
cmd2k.exe	324	1112	"E:\Shells\cmd2k.exe" /D /T:80 /F:ON /K cmdenv.bat
cmd2k.exe	324	1132	"E:\Shells\cmd2k.exe" /D /T:80 /F:ON /K cmdenv.bat
smss.exe	8	156	\SystemRoot\System32\smss.exe
winlogon.exe	156	176	winlogon.exe
csrss.exe	156	180	C:\WINNT\system32\csrss.exe ObjectDirectory= Windows SharedSection=1024,3072,51...
services.exe	176	228	C:\WINNT\system32\services.exe
lsass.exe	176	240	
dd.exe	1112	284	..\Acquisition\FAU\dd.exe if= \.\PhysicalMemory of=F:\intrusion2005\physicalmemory.dd ...
helix.exe	820	324	E:\helix.exe

Илл. 3.16. Фрагмент подробной информации о процессах в программе Responder.

Пользовательский интерфейс программы Responder предоставляет множество полезной информации. Вы можете настраивать столбцы, выбирая и перетаскивая их в новое место в той же самой панели просмотра. Кроме того, можно экспорттировать информацию (эта функции не поддерживается в пробной версии) из панели просмотра в известные форматы, щелкнув по соответствующему значку в верхней части панели, как показано на илл. 3.17.



Илл. 3.17. Выбор функции экспорта данных в программе Responder.

Дважды щелкните по папке «Все открытые сетевые сокеты» (“All Open Network Sockets”), чтобы открыть в правой панели вкладку «Сеть» (“Network”), сведения в которой похожи на выходные данные инструмента «netstat.exe», см. илл. 3.18.

Source	Destination	Type	Process
0.0.0.0:1033	192.168.0.5:4321	TCP	lsass.exe (240)
0.0.0.0:1055	192.168.0.5:4321	TCP	lsass.exe (240)
0.0.0.0:1025	0.0.0.0:0	TCP	MSTask.exe (552)
0.0.0.0:3000	0.0.0.0:0	TCP	nc.exe (1096)
0.0.0.0:1026	0.0.0.0:0	UDP	services.exe (228)
0.0.0.0:135	0.0.0.0:0	UDP	svchost.exe (408)
0.0.0.0:135	0.0.0.0:0	TCP	svchost.exe (408)
0.0.0.0:641	0.0.0.0:0	TCP	tgcmd.exe (1012)
0.0.0.0:653	0.0.0.0:0	TCP	tgcmd.exe (1012)
0.0.0.0:44444	0.0.0.0:0	TCP	UMGR32.EXE (668)

Илл. 3.18. Сведения об открытых сетевых сокетах, полученные из дампа памяти, в программе Responder.

Можно также просмотреть открытые дескрипторы файлов для всех процессов, дважды щелкнув по папке «**Все открытые файлы**» (“All Open Files”), как показано на илл. 3.19.

<code>audit.log</code>	<code>\intrusion2005\audit.log</code>	<code>dd.exe (284)</code>
<code>physicalmemory.dd</code>	<code>\intrusion2005\physicalmemory.dd</code>	<code>dd.exe (284)</code>
<code>shells</code>	<code>\shells</code>	<code>dd.exe (284)</code>
<code>hxdef-rk100sb4d1ba5d</code>	<code>\hxdef-rk100sb4d1ba5d</code>	<code>dfrws2005.exe (592)</code>
<code>hxdef-rk100sb4d1ba5d</code>	<code>\hxdef-rk100sb4d1ba5d</code>	<code>dfrws2005.exe (592)</code>
<code>ntcontrolpipe9</code>	<code>\net\ntcontrolpipe9</code>	<code>dfrws2005.exe (592)</code>
<code>svctrl</code>	<code>\svctrl</code>	<code>dfrws2005.exe (592)</code>

Илл. 3.19. Фрагмент списка открытых файлов из программы Responder.

Можно просмотреть открытые дескрипторы файлов для отдельного процесса, развернув дерево процесса и выбрав «**Открытые файлы**» (“Open Files”). Можно сделать то же самое для открытых сетевых сокетов и открытых разделов реестра.

Помимо возможности быстро просмотреть всю эту информацию как в формате всего дампа памяти, так и в формате отдельного процесса, можно проанализировать исполняемые образы (файлы .exe и .dll) на наличие строк, а также выполнить поиск отдельных элементов в дампе памяти, используя подстоки и регулярные выражения, или извлечь совпадения. Возможность сортировать элементы, отображаемые в столбцах, также позволяет намного быстрее определять подозрительные процессы или модули (т. е. DLL-файлы).

Анализ памяти процесса

Мы говорили о потребности в контексте для исследуемых объектов, и частично этого можно достичь посредством извлечения памяти, используемой процессом. В прошлом эксперты использовали такие инструменты, как «*strings.exe*» или *grep*, чтобы проанализировать содержимое памяти ОЗУ и выполнить поиск интересных строк (паролей), IP-адресов или адресов электронной почты, URL-адресов и т. д. Однако при анализе файла, размер которого примерно полмегабайта, трудно найти большое количество контекста. Бывало, эксперт открывал файл дампа в шестнадцатеричном редакторе, находил интересную строку, и, если рядом он видел что-то похожее на имя пользователя, он мог предположить, что эта строка является паролем. Но исследование файла дампа ОЗУ таким способом не позволяет эксперту сопоставить эту строку с отдельным процессом. Помните пример принципа обмена Локара из главы 1? Если бы мы собрали содержимое физической памяти во время этого примера, мы бы ни при каких обстоятельствах не смогли определенно сказать, что отдельный IP-адрес или другие данные, например список каталога, привязаны к отдельному событию или процессу. Однако если мы используем информацию, предоставленную в структуре процесса в памяти, и находим все страницы, используемые процессом, которые все еще находились в памяти во время создания дампа ее содержимого, мы бы смогли затем выполнить поиск и определить, какой процесс использовал эту информацию.

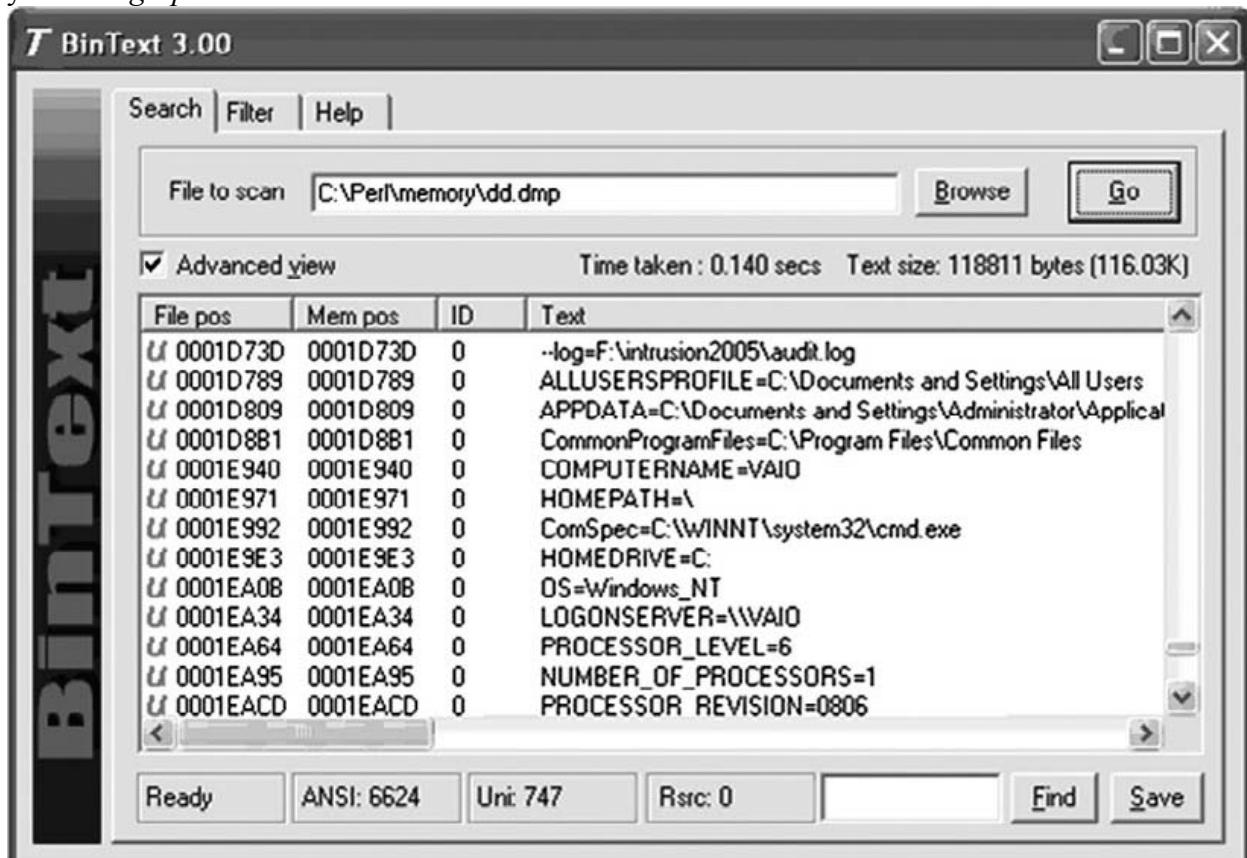
Скрипт «*lspm.pl*» позволяет сделать это автоматически при работе с дампами памяти из ОС Windows 2000. Данный скрипт принимает те же аргументы, что и «*lspd.pl*» (путь к файлу дампа вместе с его именем и смещение структуры процесса в файле) и извлекает имеющиеся страницы из дампа, записывая их в файл в текущем рабочем каталоге. Чтобы применить «*lspm.pl*» к процессу «*dd.exe*», используйте следующую команду:

```
c:\perl\memory>lspm.pl d:\dumps\dfrws1-mem.dmp 0x0414dd60
```

Выходные данные будут выглядеть так:

```
Name : dd.exe -> 0x01d9e000
There are 372 pages (1523712 bytes) to process.
Dumping process memory to dd.dmp...
Done.
```

Теперь у вас есть файл «dd.dmp», который имеет размер 1 523 712 байт и содержит все (всего 372) страницы памяти для этого процесса, которые были еще доступны во время создания файла дампа. Можно запустить инструмент «strings.exe» или использовать программу BinText (см. илл. 3.20.) с сайта Foundstone.com, чтобы проанализировать этот файл на наличие Unicode- и ASCII-строк, или выполнить поиск IP-адресов или адресов электронной почты и номеров кредитных карт или социального страхования при помощи утилиты grep.



Илл. 3.20. Содержимое памяти процесса в программе BinText.

На илл. 3.20., можно увидеть несколько Unicode-строк, содержащихся в памяти, используемой процессом «dd.exe», в том числе имя системы и имя *LogonServer* для сеанса. Вся эта информация может в дальнейшем способствовать пониманию дела; важная особенность этой возможности состоит в том, что теперь вы можете сопоставить найденный данные с отдельным процессом.

Платформа Volatility содержит такую же возможность в команде *memdump*. Как упоминалось выше в данной главе, можно использовать команду *volatility memdump*, чтобы создать дамп адресуемой памяти для процесса из дампа памяти ОС Windows XP следующим образом:

```
D:\Volatility>python volatility memdump -f d:\hacking\xp-laptop1.img -p 4012
```

Совет

Платформу Volatility можно использовать для сбора памяти процессов, скрытых руткитами, даже если они скрыты путем непосредственного манипулирования объектами

ядра (Direct Kernel Object Manipulation, DKOM; см. главу 7). Говоря точнее, при применении DKOM связь блока EProcess скрытого процесса удаляется из двусвязного списка активных процессов, который «видит» операционная система. Однако, используя Volatility для анализа дампа необработанных данных памяти или файла спящего режима из ОС Windows XP, можно выполнить поиск процессов, не являющихся частью этого двусвязного списка (обсуждалось выше в данной главе), а затем воспользоваться командой *memdump*, чтобы получить память, используемую процессом, из файла дампа памяти.

Извлечение образа процесса

Как вы видели ранее в этой главе, при запуске процесса исполняемый файл считывается в память. Один из элементов данных, который можно получить из подробностей о процессе (посредством скрипта «lspd.pl»), – это смещение в файле дампа памяти ОС Windows 2000 относительно базового адреса образа. Скрипт «lspd.pl» выполняет быструю проверку, чтобы узнать находится ли исполняемый образ в этом месте. Чтобы получить дополнительную информацию, необходимо проанализировать заголовок PE-файла (содержимое которого мы будем подробно рассматривать в главе 6) и увидеть, можно ли извлечь все содержимое исполняемого образа из файла дампа памяти ОС Windows 2000. Скрипт «lspi.pl» позволяет сделать это автоматически.

Perl-скрипт «lspi.pl» принимает те же аргументы, что скрипты «lspd.pl» и «lspm.pl», и определяет местонахождение начала исполняемого образа для этого процесса. Если смещение базового адреса образа действительно ведет к исполняемому файлу, скрипт «lspi.pl» анализирует значения, содержащиеся в заголовке PE-файла, чтобы найти страницы, составляющие оставшуюся часть исполняемого файла.

Итак, можно запустить «lspi.pl» для анализа процесса «dd.exe» (с идентификатором 284), используя следующую команду:

```
c:\perl\memory>lspi.pl d:\dumps\dftrws1-mem.dmp 0x0414dd60
```

Выходные данные этой команды будут иметь следующий вид:

```
Process Name : dd.exe
PID : 284
DTB : 0x01d9e000
PEB : 0x7ffd000 (0x02c2d000)
ImgBaseAddr : 0x00400000 (0x00fee000)
e_lfanew = 0xe8
NT Header = 0x4550
Reading the Image File Header
Sections = 4
Opt Header Size = 0x000000e0 (224 bytes)
Characteristics:
    IMAGE_FILE_EXECUTABLE_IMAGE
    IMAGE_FILE_LOCAL_SYMS_STRIPPED
    IMAGE_FILE_RELOCS_STRIPPED
    IMAGE_FILE_LINE_NUMS_STRIPPED
    IMAGE_FILE_32BIT_MACHINE
Machine = IMAGE_FILE_MACHINE_I860
Reading the Image Optional Header
Opt Header Magic = 0x10b
Subsystem : IMAGE_SUBSYSTEM_WINDOWS_CUI
Entry Pt Addr : 0x00006bda
Image Base : 0x00400000
File Align : 0x00001000
Reading the Image Data Directory information
Data Directory      RVA          Size
```

```

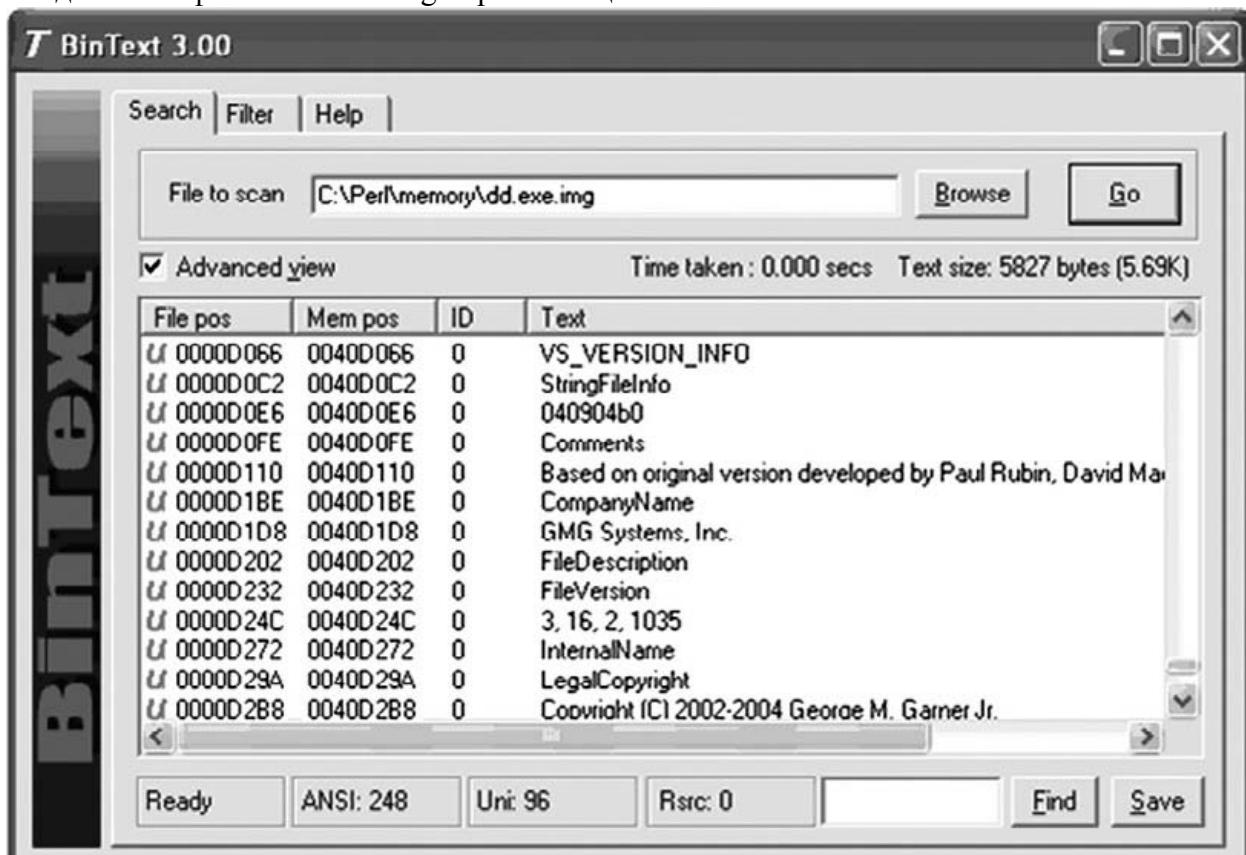
----- ----- -----
ResourceTable      0x0000d000  0x00000430
DebugTable         0x00000000  0x00000000
BaseRelocTable     0x00000000  0x00000000
DelayImportDesc    0x0000af7c  0x000000a0
TLSTable           0x00000000  0x00000000
GlobalPtrReg       0x00000000  0x00000000
ArchSpecific       0x00000000  0x00000000
CLHeader           0x00000000  0x00000000
LoadConfigTable    0x00000000  0x00000000
ExceptionTable     0x00000000  0x00000000
ImportTable         0x0000b25c  0x000000a0
unused              0x00000000  0x00000000
BoundImportTable   0x00000000  0x00000000
ExportTable         0x00000000  0x00000000
CertificateTable   0x00000000  0x00000000
IAT                 0x00007000  0x00000210
Reading Image Section Header Information
Name      Virt Sz   Virt Addr   rData Ofs   rData Sz   Char
----      -----   -----      -----      -----      -----
.text     0x00005ee0  0x00001000  0x00001000  0x00006000  0x60000020
.data     0x000002fc  0x0000c000  0x0000c000  0x00001000  0xc0000040
.rsrc     0x00000430  0x0000d000  0x0000d000  0x00001000  0x40000040
.rdata    0x00004cf8  0x00007000  0x00007000  0x00005000  0x40000040
Reassembling image file into dd.exe.img
Bytes written = 57344
New file size = 57344

```

Как видите, выходные данные скрипта «lspi.pl» довольно подробные, а большая часть этой информации, возможно, вовсе не полезна (или понятна) эксперту, если только он не интересуется анализом вредоносных программ. Мы подробно рассмотрим эту информацию в главе 6. На данном этапе важными элементами данных являются таблица, следующая за строкой «Reading Image Section Header Information», и имя файла, в который был повторно собран исполняемый образ. Информация заголовка раздела предоставляет вам план для повторной сборки исполняемого образа, так как позволяет узнать, где найти страницы, которые составляют этот файл образа. Скрипт «lspi.pl» использует этот план и пытается повторно собрать исполняемый образ в файл. Если это удается выполнить, он сохраняет файл с именем процесса, добавляя к нему расширение .img (чтобы предотвратить случайное выполнение файла). Скрипт не будет повторно собирать файл, если какая-нибудь страница памяти отмечена как недействительная и больше не расположена в памяти (например, страницы были перемещены в файл подкачки, «pagefile.sys»). Вместо этого «lspi.pl» сообщит, что не может повторно собрать полный файл, так как некоторые страницы недоступны в памяти (даже если это будет относиться к одной странице).

Файл, который вы извлекаете из памяти, не будет точно таким же, как исходный исполняемый файл. Это объясняется тем, что разделы файла перезаписываемые, и они будут изменяться во время выполнения процесса. По мере выполнения процесса различные элементы исполняемого кода (адреса, переменные и т. д.) будут изменяться в соответствии со средой и стадией выполнения. Однако есть несколько способов, с помощью которых можно определить тип файла и получить информацию о его назначении. Один из этих способов – узнать, компилированы ли в файл сведения о версии, как в большинстве файлов, созданных легальными компаниями-разработчиками программного обеспечения. Как видно в заголовках разделов файла образа, есть раздел с именем «.rsrc»; это имя часто используется для раздела ресурсов PE-файла. Этот раздел содержит различные ресурсы, например, диалоговые окна и строки версий, и организован как определенная файловая система. Используя программу BinText, можно выполнить поиск Unicode-строки *VS_VERSION_INFO* и узнать, есть ли в исполняемом файле какая-

нибудь идентифицирующая информация. На илл. 3.21., показано несколько строк, найденных в файле «dd.exe.img» при помощи BinText.



Илл. 3.21. Строки с информацией о версии, найденные в файле «dd.exe.img» при помощи BinText.

Еще один способ определить тип файла – использовать хэширование. Вероятно, вы подумали: «Погодите! Вы только что сказали, что файл, созданный при помощи «lspci.pl», не будет точно таким же, как исходный файл, так как же можно использовать хэширование?» Что ж, вы правы... Но не совсем. Мы не можем использовать для сравнения хэши MD5, потому что, как известно, изменение даже одного бита – из значения 1 в 0 – приведет к вычислению совершенно другого хэша. Так что же делать?

Летом 2006 года Джесси Корнблум разработал инструмент ssdeep (<http://ssdeep.sourceforge.net>), который вычисляет хэш частей файлов, или выполняет нечеткое хэширование. Чтобы получить подробную информацию об этом способе, прочитайте статью Джесси с конференции DFRWS 2006 (<http://dfrws.org/2006/proceedings/12-Kornblum.pdf>). Вкратце, Джесси реализовал алгоритм, который показывает взвешенный процент идентичных последовательностей битов, общих для файлов, основанный на хэш-значениях и вычисленный инструментом ssdeep. Так как мы знаем, что для создания дампа содержимого ОЗУ из OC Windows 2000 (для задачи на анализ памяти, предложенной на конференции DFRWS 2005) использовалась версия «dd.exe», созданная Джорджем Гарнером-младшим, мы можем сравнить файл «dd.exe.img» с исходным файлом «dd.exe», который случайно оказался у нас под рукой.

Сначала мы запустим «ssdeep.exe» для вычисления хэша нашего файла образа:

```
D:\tools>ssdeep c:\perl\memory\dd.exe.img > dd.sdp
```

Теперь мы создали хэш и сохранили эту информацию в файле «dd.sdp». Используя переключатели, доступные для инструмента «ssdeep.exe», мы можем быстро сравнить img-файл с исходным исполняемым образом:

```
D:\tools>ssdeep -v -m dd.sdp d:\tools\dd\old\dd.exe
d:\tools\dd\old\dd.exe matches c:\perl\memory\dd.exe.img (97)
```

То же можно сделать с помощью одной команды, используя переключатель *-d* или *-p*:

```
D:\tools\> ssdeep -d c:\perl\memory\dd.exe.img d:\tools\dd\old\dd.exe
C:\perl\memory\dd.exe.img matches d:\tools\dd\old\dd.exe (97)
```

Мы видим, что файл образа, созданный посредством скрипта «lspi.pl», с 97-процентной вероятностью совпадает с исходным файлом «dd.exe».

Не забывайте, что для правильного сравнения хэш значений нам необходимы объекты, с которыми мы можем сравнить файлы, созданные скриптом «lspi.pl». «Ssdeep.exe» – относительно новый, но очень мощный инструмент, и, вероятно, пройдет некоторое время, прежде чем с его помощью будут созданы наборы хэшей или наборы хэшей будут включать в себя хэш значения, вычисленные с использованием этого инструмента.

Можно использовать платформу Volatility Framework, чтобы попытаться извлечь исполняемый образ из файла дампа памяти ОС Windows XP, используя команду *procdump*: Синтаксис команды *procdump* (из файла «readme.txt» платформы Volatility) выглядит так:

```
procdump
-----
For each process in the given image, extract an executable sample.
If -t and -b are not specified, Volatility will attempt to infer
reasonable values.
Options:
-f      <Image>    Image file to load
-b      <base>     Hexadecimal physical offset of valid Directory Table Base
-t      <type>     Image type (pae, nopae, auto)
-o      <offset>   Hexadecimal physical offset of EPROCESS object
-p      <pid>      Pid of process
-m      <mode>     Strategy to use when extracting executable sample.
                  Use "disk" to save using disk-based section sizes or "mem"
                  for memory based sections (default": "mem") .
```

Продолжая пример использования платформы Volatility, который мы начали ранее в этой главе, можно извлечь файл образа для процесса «dd.exe» (процесс с идентификатором 4012 в файле дампа памяти «xp-laptop1.img»), используя следующую команду:

```
D:\Volatility>python volatility procdump -f d:\hacking\xp-laptop1.img -p 4012
*****
Dumping dd.exe, pid: 4012 output: executable.4012.exe
D:\Volatility>dir exe*.exe
Volume in drive D is Data
Volume Serial Number is 8049-F885
Directory of D:\Volatility
01/01/2009 12:45 PM 57,344 executable.4012.exe
1 File(s) 57,344 bytes
```

Хотя команда *volatility procdump* не предоставляет такие же подробные выходные данные, как Perl-скрипт «lspi.pl» при работе с дампами памяти из ОС Windows 2000, она извлекает исполняемый образ для процесса. Эта возможность может быть чрезвычайно полезной во время анализа вредоносных программ, так как много современных вредоносных программ имеют запутанный (посредством шифрования, сжатия или

использования обоих этих способов) код, когда находится на накопителе, что затрудняет статический анализ (см. главу 6). Кроме того, некоторые вредоносные программы могут находиться только в памяти и не записываться на НЖМД; возможность извлечь файл образа из дампа памяти может быть единственным способом получить копию этого файла для анализа.

Совет

Специалисты по расследованию инцидентов иногда имеют дело с системами, в которых применяется шифрование отдельного тома или всего накопителя. Я сталкивался с несколькими подобными системами, и, когда мне нужно было выполнять клонирование данных с целью проведения анализа (в отличие от простого создания образа), я выбирал клонирование данных НЖМД работающей системы. В мае 2007 года Брайан Каплан (Brian Kaplan) написал статью «RAM is Key: Extracting Disk Encryption Keys from Volatile Memory». Вместе со статьей Брайан также выпустил экспериментальный инструмент для извлечения ключей программы шифрования PGP Whole Disk Encryption из дампа памяти. Статья и инструмент доступны по адресу www.andrew.cmu.edu/user/bfkaplan/#KeyExtraction.

Файл подкачки и анализ дампа памяти

До настоящего времени мы рассматривали анализ дампа содержимого ОЗУ обособленно, то есть, не используя дополнительную информацию. Это означает, что инструменты типа скрипта «lspm.pl», которые опираются исключительно на дамп содержимого ОЗУ, выполняют неполный анализ дампа памяти, так как страницы памяти, перемещенные в файл подкачки («pagefile.sys» в системах Windows), не включены в итоговый дамп памяти. Чтобы преодолеть это ограничение, Николас Пол МакЛин (Nicholas Paul Maclean) весной 2006 года опубликовал свою диссертацию «Acquisition and Analysis of Windows Memory» (на момент написания данной книги я не смог найти действующую ссылку на эту работу), в которой объясняется внутренний механизм системы управления памятью ОС Windows и предоставляется инструмент vtop (написанный на языке Python) с открытым исходным кодом для восстановления виртуального адресного пространства процесса.

В начале 2007 года в журнале *Journal of Digital Investigation* была опубликована статья Джесси Корнблюма, которая называлась «Using Every Part of the Buffalo in Windows Memory Analysis», и издатель разрешил Джесси разместить копию статьи на его веб-сайте.

В этой статье показаны нюансы преобразования адресов страниц и способ включения файла подкачки в процесс анализа памяти, что позволяет получить более полное (и точное) представление о доступной информации.

Выделение памяти из пула

Диспетчер памяти ОС Windows обычно выделяет память страницами размером 4 Кб (4096 байт). Однако выделение целой 4-килобайтовой страницы для, например, предложения, скопированного в буфер обмена, было бы ненужной тратой памяти. Поэтому диспетчер памяти выделяет несколько страниц заранее, создавая доступный пул памяти. Андреас Шустер провел исчерпывающее исследование в этой области, так как, хотя Microsoft предоставляет список заголовков, назначаемых наиболее часто используемым пулам, документация для любого содержательного анализа этих пулов просто недоступна. Многие часто используемые заголовки пулов перечислены в файле «pooltag.txt» (www.microsoft.com/whdc/driver/tips/PoolMem.mspx), распространяемом вместе со средствами отладки от Microsoft, и Microsoft предоставляет статью из базы знаний, в которой описывается, как найти теги (заголовки) пулов, используемые сторонними приложениями (<http://support.microsoft.com/default.aspx?scid=kb;en-us;298102>).

Андреас использовал похожий способ, чтобы определить формат пулов памяти, используемых для сохранения информации о сетевых соединениях, в дампах памяти ОС Windows 2000

(http://computer.forensikblog.de/en/2006/07/finding_network_socket_activity_in_pools.html); он выполнил поиск заголовка пула в драйвере «tcpip.sys» в ОС Windows 2000 и смог определить формат информации о сетевых соединениях в пульке памяти.

Недостаток поиска выделенной памяти из пула состоит в том, что, хотя заголовки пулов, похоже, не меняются в зависимости от версий Windows, формат данных, находящихся в пульке памяти, изменяется, а документация о формате этих пулов памяти недоступна.

Краткое изложение

К этому времени вам должно быть понятно, что существует несколько возможностей для сбора физической памяти или памяти процесса из системы во время расследования инцидента. В главе 1 мы изучили несколько инструментов для сбора различных элементов (процессов, сетевых соединений и т. д.) энергозависимой памяти во время исследования работающей системы, не забывая о том, что всегда существует вероятность того, что безопасность интерфейсов Windows API (от которых зависят эти инструменты) может быть нарушена злоумышленником. Это относится ко всем случаям, когда проводится исследование работающей системы, и, следовательно, мы должны использовать несколько разных способов сбора энергозависимых данных. Руткит может скрывать существование процесса от большинства инструментов, перечисляющих активные процессы («tlist.exe», «pslist.exe»), но посредством создания дампа содержимого ОЗУ эксперт может получить список активных и завершившихся процессов, а также процессов, скрытых при помощи руткитов режима ядра (дополнительную информацию о руткитах см. в главе 7).

Быстрое повторение

Сбор памяти процесса

- Специалист по расследованию инцидентов может столкнуться с ситуацией, когда не нужно собирать все содержимое физической памяти, а достаточно выполнить сбор памяти, используемой отдельным процессом.
- Выполнить сбор памяти отдельного процесса можно только для процессов, которые доступны операционной системе и инструментам эксперта в списке активных процессов. Процессы, скрытые определенным способом (см. главу 7), могут быть не видны в этом списке, и эксперт не сможет указать идентификатор процесса в инструментах, которые применяются для сбора памяти, используемой процессом.
- Создание дампа памяти позволяет эксперту собрать не только память, используемую процессом, который можно найти в ОЗУ, но и память, находящуюся в файле подкачки.
- После сбора памяти процесса можно собрать дополнительную информацию о процессе, например, сведения об открытых дескрипторах или о загруженных модулях.

Создание дампа физической памяти

- Существует несколько методик для создания дампа содержимого физической памяти. Специалист по расследованию инцидентов должен знать о них, а также о преимуществах и недостатках каждой из них, чтобы уметь сделать правильный выбор относительно того, какую методику следует использовать.

- Создание дампа содержимого физической памяти из работающей системы может сопровождаться проблемами, касающимися целостности данных, так как во время создания дампа памяти система все еще работает и обрабатывает информацию.
- При создании дампа содержимого физической памяти специалисты по расследованию инцидентов и эксперты должны помнить о принципе обмена Локара.

Анализ дампа физической памяти

- В зависимости от способа, использовавшегося для сбора содержимого физической памяти, доступны различные инструменты для извлечения полезной информации из дампа памяти. В прошлом популярностью пользовались такие инструменты, как «strings.exe», BinText и grep с разными регулярными выражениями, а исследования, которые начали проводиться весной 2005 года, показали, как извлекать отдельные процессы.
- Дампы физической памяти содержат полезную информацию и такие объекты, как процессы, содержимое буфера обмена и сетевые соединения.
- Продолжающиеся исследования в этой области продемонстрировали, как использовать файл подкачки вместе дампом ОЗУ, чтобы получить более полный набор информации.

Часто задаваемые вопросы

Вопрос: Зачем нужно создавать дамп содержимого ОЗУ из работающей системы? Какой толк от этого и какая потенциально полезная или важная информация будет мне доступна?

Ответ: Как отмечалось в главе 1, значительное количество информации, имеющейся в работающей системе, может быть чрезвычайно полезной для расследования инцидента. Эта энергозависимая информация существует в памяти, или ОЗУ, во время работы системы. Для сбора этой информации можно использовать различные инструменты сторонних разработчиков (рассмотренные в главе 1), но иногда важно собрать полное содержимое памяти, чтобы не только иметь все имеющиеся данные, но и «видеть» объекты, которые могут быть не «видны» посредством традиционных средств (например, объекты, скрытые руткитом; дополнительную информацию о руткитах см. в главе 7). Кроме того, можно найти информации о завершившихся процессах, а также остаточные данные процессов, сохранившиеся после перезагрузки системы.

Вопрос: Как проводить анализ после создания дампа содержимого ОЗУ?

Ответ: В прошлом для «анализа» дампов ОЗУ эксперты использовали обычные инструменты для поиска в файлах. Например, инструменты «strings.exe» и grep применялись для поиска паролей, адресов электронной почты, URL-адресов и других подобных элементов в дампах ОЗУ. Сегодня существуют инструменты, позволяющие проанализировать дампы ОЗУ и найти процессы, подробности о процессах (команды, дескрипторы и т. д.), потоки и другие объекты, а также извлечь исполняемые образы, что особенно полезно как для анализа вредоносных программ (дополнительные сведения по этой теме см. в главе 6), так и для традиционной компьютерно-технической экспертизы.

Вопрос: Я расследую дело о пропавшем без вести человеке. Исследуя его домашний компьютер, я обнаружил на рабочем столе открытое окно приложения для обмена мгновенными сообщениями. После просмотра содержимого переписки в окне мне стало ясно, что из этого процесса можно получить полезную информацию. Как это сделать?

Ответ: Если задача, которую вам нужно решить, связана с одним видимым процессом, возможно, нет необходимости создавать дамп полного содержимого физической памяти. Рекомендуется создать дамп содержимого памяти процесса, затем использовать другие инструменты для извлечения отдельной информации о процессе, например, сведений о загруженных модулях, о команде, использовавшейся для запуска процесса, или об открытых дескрипторах. После сбора всей этой информации можно сохранить содержимое переписки из приложения для обмена мгновенными сообщениями. После того, как вся относящаяся к делу информация будет собрана, поиск фрагментов предыдущих переписок или других данных в содержимом памяти процесса может предоставить вам полезные сведения.

Содержание

Глава 1 Исследование работающей системы: Сбор энергозависимых данных	3
Введение	3
Исследование работающей системы	4
Принцип обмена Локара	5
Порядок изменяемости энергозависимых данных	8
Когда проводить исследование работающей системы	9
Какие данные нужно собрать	13
Системное время	14
Пользователи, вошедшие в систему	16
PsLoggedOn	16
Net Sessions	16
LogonSessions	17
Открытые файлы	18
Сетевая информация (таблица кэшированных имен NetBIOS)	18
Сетевые соединения	19
Netstat	20
Информация о процессах	21
Tlist	22
Tasklist	23
PsList	23
ListDLLs	24
Handle	24
Сопоставление процесса с портом	27
Netstat	27
Fport	27
Tcpvcon	28
Память процесса	29
Состояние сети	29
Ipconfig	29
PromiscDetect and Promqry	30
Содержимое буфера обмена	32
Информация о службах/драйверах	33
История командной строки	34
Подключенные сетевые накопители	35
Общие ресурсы	36
Энергонезависимые данные	36
Параметры реестра	36
ClearPageFileAtShutdown	37
DisableLastAccess	37
Autoruns	38
Журналы регистрации событий	40
Устройства и другая информация	41
Несколько слов о выборе инструментов	41
Методики исследования работающей системы	44
Методика локального исследования	44
Методика удаленного исследования	46
Комбинированный подход (Использование FSP)	48
Краткое изложение	53
Быстрое повторение	53
Часто задаваемые вопросы	54

Глава 2 Исследование работающей системы: Анализ данных	56
Введение	56
Анализ данных	56
Пример 1	58
Пример 2	62
Пример 3	65
Гибкий анализ	67
Повышение компетентности	69
Реагирование	71
Предотвращение	72
Краткое изложение	73
Быстрое повторение	74
Часто задаваемые вопросы	74
Глава 3 Анализ памяти Windows	75
Введение	75
Краткая история	76
Сбор памяти процесса	77
Создание дампа физической памяти	79
DD	79
Nigilant32	80
ProDiscover	81
KnTDD	81
MDD	83
Win32dd	84
Memoryze	85
Winen	85
Fastdump	86
F-Response	87
Краткое изложение раздела	93
Альтернативные подходы к созданию дампа физической памяти	94
Аппаратные устройства	95
FireWire	95
Аварийные дампы памяти	96
Виртуализация	98
Файл спящего режима	99
Анализ дампа физической памяти	100
Определение операционной системы на основе дампа памяти	101
Основы процессов	103
Структура EProcess	103
Механизм создания процесса	105
Анализ содержимого дампа памяти	106
Lsproc.pl	107
Lspd.pl	108
Volatility Framework	111
Memoryze	114
Responder	117
Анализ памяти процесса	120
Извлечение образа процесса	122
Файл подкачки и анализ дампа памяти	126
Выделение памяти из пула	126

Краткое изложение	127
Быстрое повторение	127
Часто задаваемые вопросы	128



<http://computer-forensics-lab.org>

Перевод:
Бочков Д.С.
Капинус О.В.
Михайлов И.Ю.