Харлэн Карви

Криминалистическое исследование Windows

Глава 6

Анализ исполняемых файлов

Содержание этой главы:

- Статический анализ
- Динамический анализ

- Краткое изложение
- Быстрое повторение
- Часто задаваемые вопросы

Введение

Иногда, во время своей работы, эксперт может найти подозрительный исполняемый файл, который нужно проанализировать, чтобы получить представление о том, что он делает, и какие функции выполняет. Часто злоумышленники оставляют на компьютере скрипты или файлы конфигурации, и, как правило, это текстовые файлы, которые можно легко открыть и просмотреть. В случае со скриптами могут потребоваться некоторые знания программирования, чтобы полностью понять назначение файла.

В главе 5 мы рассматривали анализ сигнатур файлов – метод, который применяется, чтобы определить, имеет ли файл правильное расширение исходя из типа файла. Это один из самых простых способов, используемый злоумышленниками, чтобы скрыть или маскировать присутствие файлов на взломанном компьютере; изменяя имя и расширение файла, злоумышленник может (как правило, верно) предположить, что когда администратор обнаружит этот файл, он не сильно захочет открывать его и определять его настоящий тип, если файл имеет такое расширение, как .dll.

В этой главе мы рассмотрим способы, с помощью которых эксперт может попытаться определить тип исполняемого файла. Я расскажу о средствах и приемах, которые можно использовать, чтобы собрать информацию об исполняемом файле и получить представление о его назначении. Наше обсуждение будет касаться не просто анализа вредоносных файлов, а способов анализа исполняемых файлов в общем, так как вредоносные программы могут быть только одним из классов исполняемых файлов. В этой главе мы рассмотрим несколько способов анализа, но не будем затрагивать такие темы, как дизассемблирование кода или использование инструментов типа IDA Pro (www.hex-rays.com/idapro). Использование дизассемблеров – отдельная тема, которой можно посвятить целую книгу. В данной главе мы остановимся на методах и приемах, которые смогут применить большинство администраторов и судебных экспертов.

Однако, прежде чем мы начнем, вы можете спросить, зачем мы все это делаем. Для чего нужно анализировать вредоносные файлы? Разве этим не должны заниматься производители антивирусных программ? Недавние события показали, что это не совсем так. Например, в конце 2008 – начале 2009 года наблюдалось быстрое распространение червя Conficker (также известного как Downadup) в корпоративных сетях, что в основном было обусловлено отсутствием обновлений безопасности от Microsoft (а именно для уязвимости MS08-067). Организации, применяющие корпоративные антивирусные решения, оказались незащищенными от этого червя, когда появилась его новая, ранее неизвестная версия. Однако основная причина нарушения работы инфраструктур была связана с недостаточным пониманием этого типа вредоносных программ в общем и с

почти слепой верой в антивирусные решения. Когда лица, прибывшие первыми на место инцидента, могут сразу выполнить хотя бы часть анализа, это способствует более быстрому расследованию происшествия и восстановлению работоспособности организации, что в свою очередь может в дальнейшем помочь группе по расследованию инцидентов определить основную причину заражения. Кроме того, более быстрое расследование инцидента непосредственно способствует снижению вероятности того, что злоумышленник (посредством троянской программы, приложения скрытого удаленного администрирования или бота) или сама вредоносная программа получит доступ к конфиденциальным данным, так как анализ вредоносного исполняемого файла помогает получить представление о его векторе заражения (способе проникновения в систему), механизме сохраняемости (способе сохранения работоспособности в системе) и артефактах, которые он может оставить. Затем такие артефакты можно использовать, чтобы найти другие зараженные системы, особенно когда версия вредоносной программы не распознается антивирусными приложениями. Благодаря этим знаниям и навыкам можно немедленно принять решительные и обоснованные меры, а не ожидать несколько дней, пока появится представитель производителя антивирусного приложения, соберет образцы, а позднее предоставит обновленный файл сигнатур.

Итак, давайте посмотрим, как проводится анализ исполняемых файлов.

Статический анализ

Статический анализ состоит из сбора информации об исполняемом файле и из исполняемого файла, не выполняя или запуская этот файл никоим образом. Когда большинство людей открывает исполняемый файл в «Блокноте» (я часто делаю так, чтобы показать что-нибудь клиенту) или даже в шестнадцатеричном редакторе, то все, что они видят, – это набор двоичных данных, которые, на первый взгляд, лишены смысла. Иногда можно распознать какое-нибудь слово, но, как правило, оно не имеет контекста; это может быть все, что угодно. Нужно иметь в виду, что исполняемые файлы следуют определенным правилам в отношении своего формата, и есть отдельные элементы данных, которые эксперты могут ожидать увидеть в исполняемом файле в ОС Windows. Знание этих правил позволит нам тщательно исследовать непонятные данные исполняемых файлов и извлечь значащую информацию.

Прежде чем мы приступим к анализу исполняемых файлов, нужно обсудить еще несколько моментов.

Поиск файлов для анализа

Один из вопросов, который мне довольно часто задают, – «Как найти вредоносные или подозрительные файлы в системе или образе, полученном из системы?».

Как упоминалось в главе 3, один из способов определить местонахождение этих файлов – получить дамп памяти из системы, проанализировать его с помощью одного из инструментов, рассмотренных в той главе, и найти процесс, связанный с подозрительным действием, используя набор инструментов Volatility, созданный Аароном Уолтерсом (Aaron Walters). После того как вы найдете этот процесс и проанализируете блок EProcess, вы получите путь к исполняемому файлу. Затем, используя путь, можно найти этот файл в файловой системе в образе.

Еще один способ определить местонахождение подозрительных файлов – исследовать содержимое мест автозапуска в реестре Windows, как указано в главе 4. Если вы найдете подозрительную запись реестра, например, в разделе Run, то сможете затем просто определить местонахождение этого файла в образе системы.

Совет
Используя способы удаленного анализа работающего компьютера, рассмотренные в главе 1, можно получить доступ к удаленным системам, чтобы выполнить поиск в этих

местах автозапуска в реестре. Еще один способ сделать это – развернуть F-Response на удаленном компьютере и, подключив удаленный накопитель, использовать такой инструмент, как RegRipper, чтобы собрать информацию из реестра удаленной системы точно так же, как если бы вы работали с файлами кустов, извлеченными для анализа. Вы также можете автоматизировать процесс извлечения данных, добавив необходимые команды для реализации утилиты «rip.exe» (версия RegRipper с интерфейсом командной строки) в пакетный файл.

Другой способ определить местонахождение вредоносных или подозрительных файлов в образе системы – монтировать образ как накопитель в режиме только для чтения на компьютере, используемом для анализа данных, с помощью программы Smart Mount (www.asrdata.com/SmartMount) или Mount Image Pro, а затем просканировать этот накопитель с помощью антивирусного приложения. В действительности, с учетом того, что бывают случаи, когда фактические вредоносные файлы не распознаются тем или иным антивирусным приложением, вы, возможно, захотите проверить накопитель, используя несколько антивирусных решений.

Совет

Клаус Валка (Claus Valca) в блоге Grand Stream Dreams (http://grandstreamdreams.blogspot.com/) публикует время от времени статьи о различных антивирусных приложениях, доступных для использования. Некоторые из упоминаемых им приложений – бесплатны, а их полнофункциональные версии предоставляются за отдельную плату. Однако во многих случаях бесплатные версии позволяют проверить данные на предмет вредоносных программ, что на самом деле и интересует нас сейчас. Некоторые из антивирусных приложений, о которых рассказывает Клаус, настраиваются или специально созданы для запуска с флеш-накопителя, что позволяет вам загружать, обновлять и использовать эти приложения без необходимости устанавливать их в отдельной системе. Посетите его блог и выполните поиск по словам «*anti-virus software*» и «*malware tools*», чтобы увидеть список статей по этим темам.

Однако, даже с учетом современного уровня развития антивирусных приложений, они имеют один главный недостаток: так как эти приложения работают на основе анализа сигнатур, все, что нужно сделать разработчикам вредоносных программ, – внести небольшое изменение в свои программы, перекомпилировать, а затем заново развернуть их – и эти вредоносные программы, возможно, не будут обнаружены. Мне известны случаи, когда пользователи отправляли вредоносную программу для анализа на различные сайты (такие как VirusTotal.com), и этот исполняемый файл не могли обнаружить или распознать 35 (или более) антивирусных приложений. Поэтому нужно разрабатывать разные способы, с помощью которых можно определить вредоносные исполняемые файлы в системах или образах систем. Кроме уже упомянутых способов, есть еще один – провести более глубокий анализ сигнатур; то есть, вместо того чтобы просто искать буквы *MZ* в первых двух байтах потенциального исполняемого файла, а затем сравнивать их с расширением файла, надеясь найти «exe», «dll» или другое допустимое расширение, мы должны копнуть чуть глубже. Помимо начальной сигнатуры, нужно определить, имеет ли остальная часть файла структуру, соответствующую этому типу файла. Можно также проверить, имеет ли файл цифровую подпись, используя утилиту «sigcheck.exe» от Microsoft (http://technet.microsoft.com/en-us/sysinternals/bb897441.aspx), или применить утилиту WFPCheck, как описано в главе 5, чтобы попытаться найти файлы, защищенные функцией «Защита файлов Windows» (“Windows File Protection”, WFP), которые были заменены или изменены.

Независимо от того, какой способ вы используете, чтобы определить или найти исполняемые файлы, которые могут быть подозрительными или вредоносными (в идеале,

применяйте несколько вышеупомянутых приемов), вы должны удостовериться, что тщательно документируете свои действия, а также результаты проверок и поиска.

Документирование сведений о файле

Прежде чем анализировать или исследовать файл каким-либо образом, необходимо задокументировать информацию о нем. Однако существует широко распространенное мнение (которое уже почти превратилось в байку), что технари терпеть не могут документировать что-либо. Нужно признать, что это правда, по крайней мере частично. Не помню, сколько раз я прибывал на место инцидента, а администраторы говорили мне: «Мы нашли файл». Когда я спрашивал о том, где был найден файл, они смотрели на меня с широко раскрытыми от удивления глазами. Сведения о том, где был найден файл, могут быть чрезвычайно полезны, так как они добавляют контекст к уже имеющейся информации и помогают понять, что произошло.

Итак, первое, что нужно сделать, – задокументировать всю информацию о найденном файле: в какой системе он был найден, какой полный путь к нему, а также кто и когда обнаружил этот файл.

Предупреждение

Многие специалисты, похоже, не понимают, что файл в компьютерной системе (не только в Windows) может иметь любые имена. Просмотрите рассылки и форумы за определенный период времени и вы найдете несколько сообщений, где какой-нибудь пользователь пишет: «Я нашел такой-то файл в системе, и узнал в Google, что он безвредный». Выполняя поиск информации о файле только по его имени, можно найти много интересных и полезных сведений, но эти следования не следует рассматривать как окончание анализа. Однажды я расследовал инцидент, когда ИТ-специалисты компании нашли несколько файлов в зараженной системе и выполнили в Google поиск информации о каждом из этих файлов. Введя имя одного из найденных файлов, они увидели, что это надежный файл, предоставленный Microsoft, и закончили на этом свое расследование. Однако, исследовав файл более тщательно, используя способы, описанные в этой главе, я смог определить, что на самом деле этот файл является вредоносной программой.

В зависимости от того, каким способом был первоначально найден подозрительный файл, вы уже, возможно, будете иметь данные о нем. Если вы производите анализ работающего компьютера и используете способы анализа, упомянутые в главе 1, вероятно, у вас уже есть определенная документальная информация, например, полный путь к файлу. То же самое можно сказать о случае, когда вы обнаружили файл в образе системы, используя ProDiscover или другой инструмент для судебного анализа.

Еще один аспект файла, подлежащий документированию, – имя и версия операционной системы, в которой был найден файл. Операционные системы Windows отличаются в зависимости от версии и даже от пакета обновления в одной и той же версии. Результат работы вредоносной программы может зависеть от того, в какой версии Windows она находится. Например, в ложном сообщении электронной почты с вирусом Teddy Bear сообщалось, что файл «jdbgmgr.exe» является вредоносным, и рекомендовалось немедленно удалить этот файл. (Вирус назывался «Teddy Bear» (рус. *плюшевый мишка*), потому что на его значке был изображен медвежонок.) Если бы пользователь сделал это в ОС Windows NT 4.0, файл был бы удален. Однако в ОС Windows 2000 функция WFP немедленно бы заменила этот файл. Операционные системы Windows 2000 и Windows XP содержат разные наборы файлов, защищенных функцией WFP. В 2000 году хакеры под псевдонимами «Бенни» (Benny) и «Крысолов» (Ratter) выпустили пробный вирус W32.Stream, который использовал альтернативные потоки данных файловой системы NTFS (подробные сведения об этих потоках см. в главе 5).

Если вирус попадал в ОС Windows с файловой системой, отформатированной как FAT/FAT32, он вел себя по-другому, но только потому, что файловая система FAT не поддерживает альтернативные потоки данных

Помимо регистрации сведений о том, где был найден файл в файловой системе и в какой версии Windows, следует также собрать дополнительную информацию о файле, например, отметки времени изменения, доступа и создания, а также любые ссылки на этот файл в файловой системе (например, ярлыки в пользовательской папке «Автозагрузка» (“StartUp”)) или реестре, которые вы заметите во время первичного анализа.

Предупреждение

Экспертам нужно проявлять особую осторожность во время первичного исследования компьютера, особенно когда он еще работает. Ранее в этой книги мы говорили о принципе обмена Локара и о том, что поиск текста в кодировках ASCII и Unicode не всегда работает в реестре, так как некоторые значения хранятся в двоичном формате. Любое действие, выполняемое экспертом в системе, оставляет после себя артефакты, поэтому, если вы нашли необычный файл, ограничьте поиск дополнительной информации о нем в системе до минимума. Любые действия, которые вы решите выполнить, должны быть тщательно задокументированы.

Чем подробнее будет ваша документация, тем лучше. Рекомендуется постоянно придерживаться этого принципа во время любой экспертизы, так как это избавит вас от «головной боли» в будущем. Более того, этот принцип является частью передового метода работы.

Следующее действие, которое нужно выполнить во время документирования информации о файле, – вычислить криптографический хэш файла. Криптографические хэш-значения используются в области информационной безопасности и компьютерно-технической экспертизы, чтобы обеспечить целостность файла, то есть для того, чтобы убедиться, что файл не был изменен. Один из популярных алгоритмов хэширования – это MD5, принимающий входные данные произвольной длины и создающий 128-разрядное хэш-значение, которое обычно представляет собой последовательность из 32 шестнадцатеричных символов. Любые изменения во входных данных, даже переключение одного бита, приведут к созданию другого хэш-значения MD5. Несмотря на недостатки в алгоритме MD5, которые приводят к коллизиям хэш-значений (http://en.wikipedia.org/wiki/Md5), этот алгоритм все равно считается подходящим для компьютерно-технической экспертизы. Еще один распространенный алгоритм хэширования – SHA-1 (http://en.wikipedia.org/wiki/Sha-1). Такие организации, как библиотека National Software Reference Library (NSRL) при институте NIST, используют алгоритм SHA-1 при вычислении криптографических хэш-значений для компакт-дисков с эталонными наборами данных (Reference Data Set, RDS). Используя такие наборы, эксперты могут сократить объем обрабатываемых данных, отфильтровывая известные надежные и известные вредоносные файлы.

Совет

Вычислить MD5-хэш исполняемого файла, который, по вашему мнению, является вредоносным, можно посетить веб-сайт VirusTotal.com и опубликовать там либо сам файл, либо MD5-хэш для анализа. Если вы опубликуете для анализа исполняемый файл, он будет проверен примерно 35 различными антивирусными приложениями. Если вы отправите MD5-хэш, его сравнят с хэш-значениями из базы данных, хранящейся на сайте. Этот сайт – отличный ресурс для тех, кто имеет ограниченный доступ только к нескольким антивирусным приложениям, или для тех, кто хочет услышать 34 других мнения.

Еще один полезный алгоритм хэширования реализовал Джесси Корнблюм (Jesse Kornblum) в своем инструменте ssdeep (который основан на инструменте spamsum доктора Эндрю Триджелла (Andrew Tridgell)), доступном на сайте http://ssdeep.sourceforge.net/. Вместо того чтобы вычислять криптографический хэш всего файла, ssdeep вычисляет хэш-значения нескольких частей файла случайного размера (например, 4 Кб) одновременно (www.dfrws.org/2006/proceedings/12-Kornblum.pdf). Этот способ можно использовать не только для проверки целостности исходного файла, но и для того, чтобы увидеть, насколько похожими могут быть два файла. Например, если вычислить хэш документа Word с помощью «ssdeep.exe», затем немного изменить документ (добавив или удалив текст, изменив форматирование и т. д.), а после снова вычислить его хэш, то ssdeep сможет показать, насколько похожи эти файлы. Этот способ можно также использовать с файлами других типов, такими как изображения, аудио- и видеофайлы.

После того как вы задокументируете сведения о файле, можно начинать сбор информации из самого файла.

Анализ

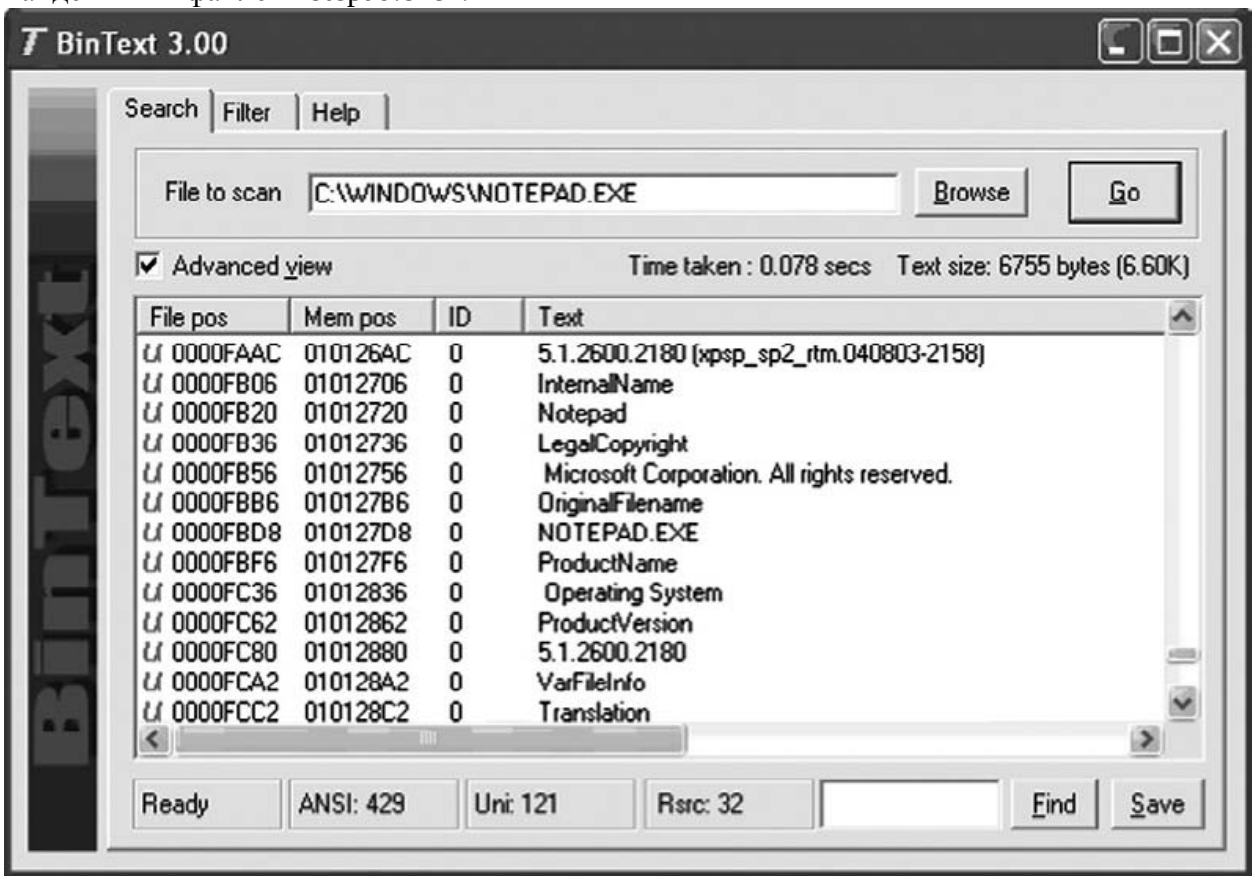
Одно из первых действий, которое выполняет большинство экспертов во время статического анализа, – проверка подозрительного файла с помощью антивирусного приложения. Это прекрасный способ начать анализ, но не удивляйтесь, если проверка антивирусным приложением не даст четких результатов. Новые вредоносные коды выпускаются постоянно. На самом деле одна компания, выпускающая антивирусные приложения, в январе 2007 года опубликовала отчет за предыдущий год и указала в нем в общей сложности 207 684 угроз, от которых мог защитить антивирусный продукт этой компании, а также 41 536 новых видов вредоносного кода, обнаруженных этим продуктом. Проверка подозрительного файла может дать вам представление о типе этого файла, но не нужно слишком беспокоиться, если вы получили ответ «вирусов не обнаружено». Проверка с использованием нескольких антивирусных приложений может также дать вам более полное представление о файле.

Следующее действие, которое выполняет большинство экспертов, – анализ подозрительного исполняемого файла с помощью программы «strings.exe» (http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx), чтобы извлечь все строки ASCII и Unicode заданной длины. Этот процесс помогает экспертам получить представление о типе файла на основе его строк. Последняя версия «strings.exe» (на момент написания этой книги) позволяет выполнять поиск строк в кодировках ASCII и Unicode, а также показывает смещение строки относительно начала файла. Это смещение подскажет вам, в каком разделе файла находится строка, и предоставит контекст к этой строке (мы рассмотрим разделы и заголовки разделов позднее в этой главе). Можно даже запустить программу «strings.exe» для поиска отдельных строк во всех файлах, используя примерную команду, показанную на веб-сайте программы.

Примечание

Когда-то я помогал исследовать файл, полученный из системы, которая отправляла большое количество трафика в Интернет из корпоративной инфраструктуры. Оказалось, что этот файл был вирусом IE0199 (www.f-secure.com/v-descs/antibtc.shtml), который заражал систему и начинал отправлять трафик в болгарскую телекоммуникационную инфраструктуру. Мы обнаружили ASCII-строки, составлявшие «манифест» разработчика, и к счастью один из членов нашей группы изучал русский язык во время службы в армии США и смог перевести найденный текст. Очевидно, автор расстроился из-за тарифов на доступ в Интернет в Болгарии и захотел вывести инфраструктуру из строя с помощью атаки типа «отказ в обслуживании» (DoS-атака).

Еще одна полезная утилита для поиска строк в двоичном файле – это BinText, которая раньше была доступна на сайте компании Foundstone (подразделение McAfee). BinText находит все строки в кодировке ASCII, Unicode и строки ресурсов в двоичном файле и показывает их в приятном графическом интерфейсе пользователя вместе со смещением строки относительно начала файла. На илл. 6.1 показаны несколько строк, найденных в файле «notepad.exe».



Илл. 6.1. Файл «notepad.exe», открытый в BinText .

Хотя строки, найденные в файле, не показывают полную картину того, для чего предназначен файл, они помогают это определить. Более того, строки могут не иметь никакого контекста, кроме их местонахождения. Например, на илл. 6.1, мы видим, что строки показаны в Unicode (буква «U» в левой части интерфейса) и что они, похоже, являются частью информации о версии файла (подробнее об этом будет рассказано позднее в этой главе). Другие строки могут не иметь такой же уровень контекста в файле. Строки в файле, которые кажутся странными или уникальными, можно использовать для поиска в других файлах, а также в Интернете (кроме шуток, однажды я нашел в файле строку «supercalifragilisticexpialidocious»). Результаты такого поиска могут предоставить вам информацию, которая поможет вам в дальнейшем анализе (статическом или динамическом) исполняемого файла.

Существует много веб-сайтов, предлагающих услуги по обратной разработке вредоносных или даже законных программ, и, как ни странно, все они используют одни и те же способы и инструменты для сбора информации из исполняемых файлов. Два таких инструмента, которые мы будем использовать в следующих разделах данной главы, – это «pedump.exe» и «peview.exe».

В феврале 2002 года была опубликована первая из двух статей Мэтта Питрека (Matt Pietrek), которая называется «An In-Depth Look into the Win32 Portable Executable File Format». В этих статьях Мэтт не только подробно описал различные аспекты формата переносимого исполняемого (PE) файла, но и предоставил инструмент командной строки «pedump.exe» (доступен на сайте www.wheaty.net), который можно использовать для извлечения детальной информации из заголовка PE-файла. Информация, извлеченная с

помощью «pedump.exe», отправляется на стандартное устройство вывода, чтобы ее можно было просмотреть или перенаправить в файл для дальнейшего анализа.

Совет
Первая часть статьи Мэтта Питрека находится по адресу http://msdn.microsoft.com/en-us/magazine/cc301805.aspx, вторая часть – http://msdn.microsoft.com/en-us/magazine/cc301808.aspx.

Еще один полезный инструмент для исследования содержимого PE-файлов Windows – это «peview.exe» (www.magma.ca/~wjr/) от Уэйна Редберна (Wayne Radburn). «Peview.exe» – инструмент с графическим интерфейсом пользователя, который позволяет просмотреть различные компоненты заголовка (а также другие части) PE-файла в удобном формате. На момент написания этой книги была доступна версия 0.96, в которой отсутствовала возможность сохранить результаты просмотра в файл.

Ни один из этих инструментов не предоставлен на DVD-носителе, который идет в комплекте с этой книгой, из-за проблем с лицензированием и распространением. Посетите соответствующие веб-сайты, чтобы загрузить эти инструменты, и таким образом вы гарантированно получите их последние доступные версии. Тем не менее, в состав DVD-диска включен Perl-код для доступа к структурам PE-файла. Скрипт «pedmp.pl» использует Perl-модуль File::ReadPE, чтобы получить доступ к содержимому заголовка PE-файла и проанализировать различные структуры. Perl-скрипт и модуль предоставлены в целях обучения, чтобы вы увидели, по какому принципу работают другие инструменты. Кроме того, Perl-код написан так, чтобы при необходимости его можно было выполнять на различных платформах; то есть когда значения байтов извлекаются из исполняемого файла, Perl-функция ***unpack( )*** используется вместе с распакованными строками, что позволяет перевести значения в формат с порядком следования байтов младшего к старшему. Таким образом, можно запускать скрипты и проводить анализ не только в ОС Windows, но и в Linux и даже в Mac OS X (что имеет свои преимущества, так как вряд ли вы сможете в Linux или Mac OS X «случайно» выполнить вредоносную программу для Windows и заразить систему).

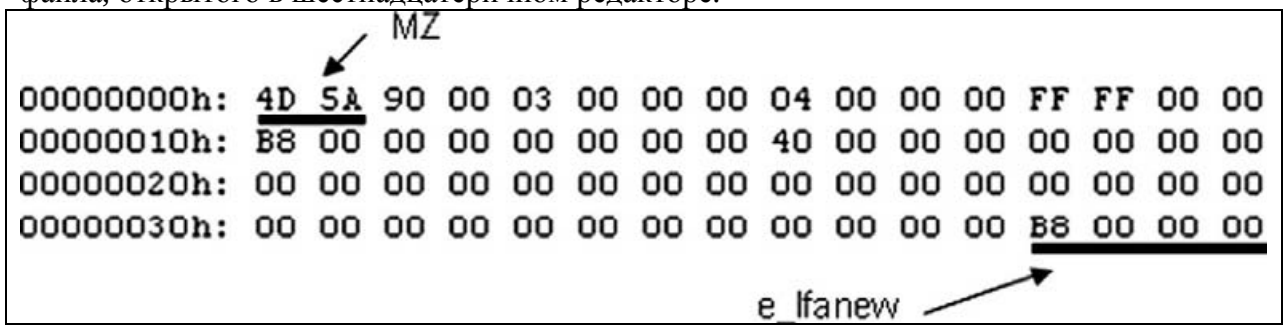
Заголовок PE-файла

По адресу www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx корпорация Microsoft предоставляет подробную документацию о формате PE-файлов (а также о формате COFF, который используется в системах VAX/VMS). Кроме того, Microsoft опубликовала информацию о большинстве структур, используемых в заголовках файлов, как часть документации для API-структур ImageHlp (http://msdn2.microsoft.com/en-gb/library/ms680198.aspx). Используя эти и другие источники, мы можем понять структуру PE-файла, исследовать его содержимое и извлечь информацию, которая пригодится нам во время анализа.

PE-файл можно разделить на несколько исследуемых областей (я не решаюсь сказать «разделов», так как этот термин мы будем использовать для особой цели во время обсуждения). Первая и, возможно, самая важная часть PE-файла – это сигнатура файла. В исполняемых файлах операционных систем Windows сигнатура состоит из букв ***MZ***, которые находятся в первых двух байтах файла. Как отмечалось ранее в этой книге, эти две буквы являются инициалами Марка Збиковского (http://en.wikipedia.org/wiki/Mark_Zbikowski), разработчика из Microsoft, который создавал формат исполняемых файлов. Однако, как вы увидите, для того чтобы файл считался исполняемым, нужно гораздо больше, чем эти две буквы и расширение «.exe» после имении файла.

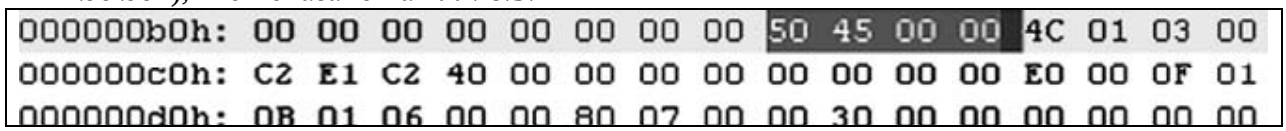
Инициалы Марка – это сигнатура для 64-байтовой структуры, которая называется IMAGE_DOS_HEADER. Важные элементы этой структуры – это первые два байта

(«магическое число» 0x5a4d в шестнадцатеричном формате с порядком следования байтов от младшего к старшему, или *MZ*) и последнее значение двойного слова (4 байта), которое называется *e_lfanew*. Это значение определено в заголовочном файле «ntimage.h» как файловый адрес (смещение) нового заголовка EXE-файла, то есть смещение, по которому мы должны найти сигнатуру для начала структуры IMAGE_NT_HEADERS. Значение *e_lfanew* указывает на местонахождение заголовка PE-файла, позволяя Windows правильно выполнить двоичный файл. На илл. 6.2 показаны эти значения из исполняемого файла, открытого в шестнадцатеричном редакторе.



Илл. 6.2. Структура IMAGE_DOS_HEADER, показанная в шестнадцатеричном редакторе.

В примере, показанном на илл. 6.2, структура IMAGE_NT_HEADERS должна быть расположена по смещению 0xB8 (184 в десятичной системе счисления) относительно начала файла. Структура IMAGE_NT_HEADERS состоит из сигнатуры и двух дополнительных структур, IMAGE_FILE_HEADER и IMAGE_OPTIONAL_HEADER. Сигнатура заголовка PE-файла – это символы «PE», за которыми следуют два нулевых значения (значение сигнатуры имеет длину двойного слова (4 байта) и отображается как «PE\00\00»), что показано на илл. 6.3.



Илл. 6.3. Значение сигнатуры IMAGE_NT_HEADERS.

Структура IMAGE_FILE_HEADER (http://msdn.microsoft.com/en-gb/library/ms680313.aspx) содержится в 20 байтах сразу после сигнатуры «PE\00\00» и включает в себя несколько значений, которые могут быть полезны экспертам. В таблице № 6.1 перечислены значения структуры IMAGE_FILE_HEADER и дано их описание.

Таблица № 6.1

Значения структуры IMAGE_FILE_HEADER

Размер	**Имя**	**Описание**
2 байта	*Machine*	Указывает тип архитектуры компьютера; программу можно запускать только в системе, которая эмулирует этот тип
2 байта	*Number of Sections*	Указывает, сколько разделов (IMAGE_SECTION_HEADERS) включено в PE-файл
4 байта	*TimeDateStamp*	Дата и время, когда компоновщик создал исполняемый файл, в формате времени UNIX (т. е. число секунд, прошедших с начала 1 января 1970 года). Это значение обычно указывает системное время на компьютере программиста, когда был скомпилирован исполняемый файл
4 байта	*Pointer to Symbol Table*	Смещение таблицы символов (0 – если таблица символов COFF отсутствует)
4 байта	*Number of Symbols*	Количество символов в таблице символов
2 байта	*Size of Optional Header*	Размер структуры IMAGE_OPTIONAL_HEADER;

		определяет архитектуру (32- или 64-разрядная), для которой предназначен файл
2 байта	*Characteristics*	Флаги, обозначающие различные характеристики файла

На илл. 6.4 показана структура IMAGE_FILE_HEADER тестового приложения, открытого в PEView.

pFile	Data	Description	Value
000000BC	014C	Machine	IMAGE_FILE_MACHINE_I386
000000BE	0003	Number of Sections	
000000C0	40C2E1C2	Time Date Stamp	2004/06/06 Sun 09:20:02 UTC
000000C4	00000000	Pointer to Symbol Table	
000000C8	00000000	Number of Symbols	
000000CC	00E0	Size of Optional Header	
000000CE	010F	Characteristics	
		0001	IMAGE_FILE_RELOCS_STRIPPED
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0004	IMAGE_FILE_LINE_NUMS_STRIPPED
		0008	IMAGE_FILE_LOCAL_SYMS_STRIPPED
		0100	IMAGE_FILE_32BIT_MACHINE

Илл. 6.4. Структура IMAGE_FILE_HEADER, показанная в программе PEView.

Значение ***TimeDateStamp*** может быть важным для судебных экспертов при исследовании исполняемого файла, так как оно показывает, когда компоновщик создал файл (эксперты должны также знать, что это значение можно изменить в шестнадцатеричном редакторе, что никак не повлияет на выполнение самого файла). Оно обычно указывает системное время на компьютере программиста, когда был скомпилирован исполняемый файл, и может дать представление о том, когда была создана эта программа. При выполнении анализа файла обратите внимание на то, что количество разделов, указанных в структуре IMAGE_FILE_HEADER, должно совпадать с количеством разделов в файле. Кроме того, если расширение файла было изменено, значение *Characteristics* предоставит вам сведения об истинном типе файла; например, если в значении *Characteristics*, показанном на илл. 6.4, установлен флаг IMAGE_FILE_DLL (т. е. значение равно 0x2000), то исполняемый файл является библиотекой динамической компоновки (DLL) и не может быть запущен напрямую. Один класс файлов, который обычно встречается в виде DLL-файлов, – это объекты модуля поддержки браузера (browser helper object, BHO; подробнее – в главе 4). Это DLL-файлы, которые загружаются вместе с браузером Internet Explorer и могут предоставить различные виды функциональных возможностей. В некоторых случаях эти DLL-файлы могут быть безопасными (как, например, объект BHO, используемый для загрузки программы Adobe Acrobat Reader, когда к PDF-файлу осуществляется доступ через браузер), но часто объекты BHO могут быть шпионскими или рекламными программами. На сайте MSDN есть страница о структуре IMAGE_FILE_HEADER, предоставляющая список возможных постоянных значений, которые могут содержаться в поле *Characteristics*.

Значение, указывающее размер структуры IMAGE_OPTIONAL_HEADER (http://msdn.microsoft.com/en-gb/library/ms680339.aspx), очень важно для анализа файла, так как оно показывает, используется ли заголовок для 32-рарядного или 64-разрядного приложения. Это значение совпадает с «магическим числом» структуры IMAGE_OPTIONAL_HEADER, которое находится в первых двух байтах структуры; значение 0x10b указывает на 32-разрядный исполняемый файл, значение 0x20b – на 64-разрядный исполняемый файл, а значение 0x107 – на ROM-образ. В нашем обсуждении мы сосредоточимся на структуре IMAGE_ OPTIONAL_HEADER32 для 32-разрядного

исполняемого файла. На илл. 6.5 показана структура IMAGE_OPTIONAL_HEADER тестового приложения, открытого в PEView.

pFile	Data	Description	Value
00000108	0007C000	Size of Image	
0000010C	00001000	Size of Headers	
00000110	0007F430	Checksum	
00000114	0002	Subsystem	IMAGE_SUBSYSTEM_WINDOWS_GUI
00000116	0000	DLL Characteristics	
00000118	00100000	Size of Stack Reserve	
0000011C	00001000	Size of Stack Commit	
00000120	00100000	Size of Heap Reserve	
00000124	00001000	Size of Heap Commit	
00000128	00000000	Loader Flags	
0000012C	00000010	Number of Data Directories	

Илл. 6.5. Структура IMAGE_OPTIONAL_HEADER, показанная в программе PEView.

Значения, показанные на илл. 6.5, указывают, что тестовое приложение было разработано для подсистемы графического интерфейса Windows, а значение ***DLL Characteristics***, установленное в 0000, указывает, что это приложение не является DLL-файлом.

Как вы видели ранее, значение размера структуры IMAGE_OPTIONAL_HEADER хранится в структуре IMAGE_FILE_HEADER, содержащей несколько значений, которые могут быть полезны для определенного, подробного анализа исполняемых файлов. Этот уровень анализа выходит за рамки этой главы.

Однако в структуре IMAGE_OPTIONAL_HEADER есть интересное значение ***Subsystem***, которая указывает операционной системе, какая подсистема необходима для запуска исполняемого файла. Microsoft даже предоставляет статью № 90493 из базы знаний (http://support.microsoft.com/kb/90493), в которой описан способ (и включен код), чтобы определить подсистему приложения. Обратите внимание, что страница о структуре IMAGE_OPTIONAL_HEADER на сайте MSDN предоставляет больше возможных значений для ***Subsystem***, чем статья из базы знаний.

Еще одно значение, которое может представлять интерес для экспертов, – это ***AddressofEntryPoint*** в структуре IMAGE_OPTIONAL_HEADER. Это указатель на функцию точки входа относительно базового адреса исполняемого файла. Именно здесь начинается код для приложения в исполняемых файлах. Важность этого значения станет очевидной позднее в этой главе.

Сразу после структуры IMAGE_OPTIONAL_HEADER находятся структуры IMAGE_DATA_DIRECTORY (http://msdn.microsoft.com/en-us/library/ms680305.aspx). Эти каталоги данных, показанные на илл. 6.6, выступают в роли структуры каталогов для такой информации в PE-файле, как таблицы имен и адресов DLL-функций, импортируемых в исполняемый файл и используемых им, таблица экспортируемых функций (для DLL-файлов), начальный адрес и размер каталога «Debug» (http://msdn.microsoft.com/en-us/library/ms680305.aspx), если таковой имеется, а также каталога «Resource» и других каталогов из 16 возможных. Для каждого каталога данных указывается относительный виртуальный адрес (англ. ***relative virtual address***, далее RVA) и значение размера.

00000138	00078004	RVA	IMPORT Table
0000013C	00000028	Size	
00000140	0007A000	RVA	RESOURCE Table
00000144	0000114C	Size	
00000148	00000000	RVA	EXCEPTION Table
0000014C	00000000	Size	
00000150	00000000	Offset	CERTIFICATE Table
00000154	00000000	Size	

Илл. 6.6. Пример структур IMAGE_DATA_DIRECTORY, показанных в PEView.

На илл. 6.6 показано четыре из 16 имеющихся каталогов данных в тестовом приложении. Указанные значения – это местонахождения или смещения данных в PE-файле. Например, в первой строке на илл. 6.6 показано, что таблица импорта (IMPORT) находится по смещению 0x138, также указано значение в этом месте (0x78004) и имя значения (RVA). Из информации на илл. 6.6 мы видим, что в тестовом приложении есть как таблица импорта (IMPORT), так и таблица ресурсов (RESOURCE).

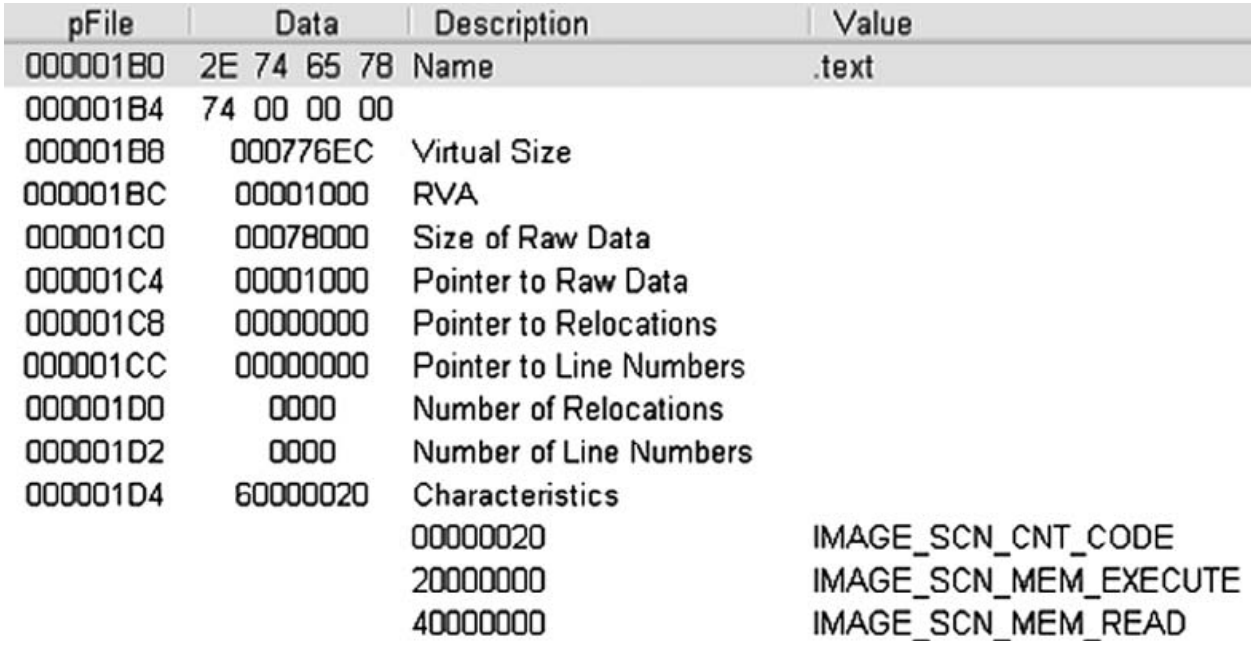
Совет

RVA используется в исполняемом файле, когда нужно указать адрес переменной (например), но нельзя использовать жестко закодированные адреса. Это объясняется тем, что исполняемый файл не будет загружаться в одно и то же место в памяти на каждом компьютере. Адреса RVA используются для того, чтобы указать места в памяти, которые не зависят от того места, куда загружается файл. RVA – это по существу смещение в памяти относительно места, куда загружается файл. Формула для вычисления RVA имеет следующий вид:

RVA = (целевой адрес) – (адрес загрузки)

Чтобы получить адрес памяти (также известный как виртуальный адрес; англ. *Virtual Address*, VA), просто сложите адрес загрузки и RVA.

Последняя часть PE-файла, представляющая для нас интерес, – это структуры IMAGE_SECTION_HEADER (http://msdn.microsoft.com/en-us/library/ms680341.aspx). Структура IMAGE_FILE_HEADER содержит значение, указывающее количество разделов в PE-файле, а следовательно количество структур IMAGE_SECTION_HEADER, которые должны быть считаны. Структуры IMAGE_SECTION_HEADER имеют размер 40 байт и содержат имя раздела (длиной восемь символов), информацию о размере раздела как на накопителе, так и в памяти (это упоминалось в главе 3), и свойства раздела (т. е. можно ли прочитать раздел, сделать в него запись, выполнить его и т. д.). На илл. 6.7 показана структура IMAGE_SECTION_HEADER.



pFile	Data	Description	Value
000001B0	2E 74 65 78	Name	.text
000001B4	74 00 00 00		
000001B8	000776EC	Virtual Size	
000001BC	00001000	RVA	
000001C0	00078000	Size of Raw Data	
000001C4	00001000	Pointer to Raw Data	
000001C8	00000000	Pointer to Relocations	
000001CC	00000000	Pointer to Line Numbers	
000001D0	0000	Number of Relocations	
000001D2	0000	Number of Line Numbers	
000001D4	60000020	Characteristics	
		00000020	IMAGE_SCN_CNT_CODE
		20000000	IMAGE_SCN_MEM_EXECUTE
		40000000	IMAGE_SCN_MEM_READ

Илл. 6.7. Структура IMAGE_SECTION_HEADER, показанная в программе PEView.

Совет

При просмотре имен разделов нужно иметь в виду, что не существует строго определенных требований относительно того, какими должны или могут быть эти имена. Имя раздела – это всего лишь последовательность любых символов (не более восьми). Вместо «.text» имя раздела может быть «timmy». Изменение имени не влияет на функциональность PE-файла. На самом деле некоторые авторы вредоносного ПО редактируют и изменяют имена разделов, возможно, чтобы сбить с толку неопытных специалистов по анализу таких программ. Большинство обычных программ имеет такие имена разделов, как «.code», «.data», «.rsrc» или «.text». В системных программах разделы имеют такие имена, как PAGE, PAGEDATA и т. д. Хотя эти имена являются обычными, злоумышленник может так же назвать разделы во вредоносной программе, чтобы они казались безвредными. Некоторые имена разделов могут быть напрямую ассоциированы с приложениями для сжатия или шифрования данных. Например, любая программа, в которой имя раздела начинается с символов *UPX*, была обработана с помощью одного из таких приложений. Позднее мы подробнее рассмотрим эту тему.

Всю информацию PE-файла можно также просмотреть, используя программу «pedump.exe». Информация, показанная на илл. 6.7, отображается в следующем формате при просмотре в «pedump.exe»:

```
01   .text VirtSize:    000776EC      VirtAddr:       00001000
     raw data offs:     00001000      raw data size:  00078000
     relocation offs:   00000000      relocations:    00000000
     line # offs:       00000000      line #'s:       00000000
     characteristics:   60000020
CODE EXECUTE            READ ALIGN_DEFAULT(16)
```

Как видите, эти инструменты показывают практически одинаковую информацию. Информация о виртуальном размере и об адресе определяет, как исполняемый файл будет «выглядеть» в памяти, а информация о необработанных данных (raw data) относится к исполняемому файлу, когда он находится на накопителе. Как вы поняли из главы 3, эта информация также предоставляет вам план действий при извлечении исполняемого файла из дампа памяти.

Таблицы импорта (IMPORT)

Сегодня приложения очень редко разрабатываются полностью с нуля. Большинство программ создается через доступ к интерфейсу прикладного программирования (API) Windows посредством различных функций, предоставленных в библиотеках (DLL-файлах) в системе. Microsoft предоставляет много DLL-файлов, которые предлагают доступ к стандартным функциям для создания окон, меню, диалоговых окон, сокетов и практически любых виджетов, объектов и конструкций в системе. Нет никакой необходимости разрабатывать эти элементы самому при создании приложения или программы.

Таким образом, когда программа разрабатывается, а затем компилируется и компонуется в исполняемый файл, информация о DLL-файлах и функциях, к которым получает доступ эта программа, должна быть доступна операционной системе при запуске программы. Эта информация содержится в таблице импорта и таблице адресов импорта в исполняемом файле.

Примечание

Некоторое время тому назад мне представилась возможность работать над проектом, одна из целей которого была определить, имеет ли исполняемый файл возможность подключения к сети. Я исследовал несколько приложений, чтобы определить имеют ли они возможность работать в качестве сетевого сервера (прослушивать соединения, как троянская программа скрытого удаленного администрирования) или клиента (устанавливать подключения к серверу, как чат-бот), но задачей этого проекта было автоматизировать этот процесс. Поэтому мы начали исследовать имеющиеся DLL-файлы, чтобы установить, какие из них предоставляют сетевую функциональность (т. е. wininet.dll, ws2_32.dll и т. д.), а затем определили, какие функции обеспечивают соответствующие основные функциональные возможности. После того как мы получили эту информацию, мы смогли автоматизировать процесс, анализируя структуры PE-файла, устанавливая местонахождение таблицы импорта и определяя, какие DLL-файлы и функции используются. Однако нужно иметь в виду, что чтение таблицы импорта вредоносного исполняемого файла может быть не таким простым, если данные файла были намеренно запутаны каким-либо образом.

Инструмент «pedump.exe» предоставляет легкий доступ к информации из таблицы импорта, устанавливая местонахождение каталога данных импорта и анализируя структуры, чтобы определить DLL-файлы и функции, которые использует приложение. Ниже показан фрагмент выходных данных утилиты «pedump.exe»:

Таблица импорта:

```
    ...
KERNEL32.dll
   OrigFirstThunk:    0000D114 (Unbound IAT)
   TimeDateStamp:     00000000 -> Wed Dec 31 19:00:00 1969
   ForwarderChain:    00000000
   First thunk RVA:   0000B000
   Ordn  Name
    448  GetSystemTimeAsFileTime
     77  CreateFileA
    393  GetNumberOfConsoleInputEvents
    643  PeekConsoleInputA
    571  LCMapStringW
    570  LCMapStringA
    443  GetSystemInfo
```

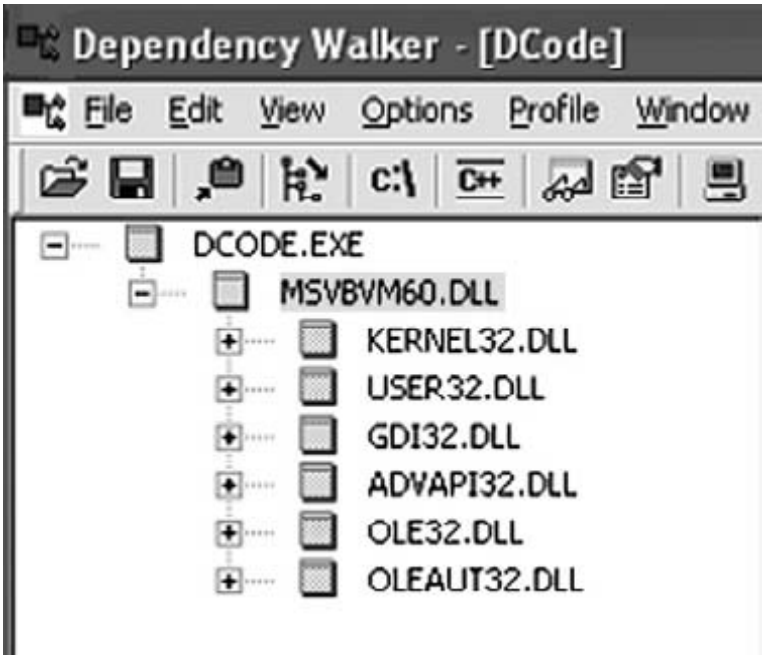
Как видите, тестовое приложение импортирует несколько функций из файла «kernel32.dll». Хотя данный DLL-файл фактически предоставляет несколько доступных для использования функций (см. раздел «Таблица экспорта» позднее в этой главе), этот тестовый исполняемый файл импортирует для использования такие функции, как *GetSystemTimeAsFileTime()* и *CreateFileA()*. Microsoft предоставляет на своем веб-сайте большое количество информации о различных доступных функциях, поэтому, выполнив поиск, вы сможете узнать, для чего предназначены многие из них. Например, функция *GetSystemTimeAsFileTime()* находит текущее системное время в виде 64-разрядного значения структуры *FILETIME*, и полученное значение представляет собой количество 100-наносекундных интервалов, прошедших с 1 января 1601 года, в формате всемирного координированного времени (UTC).

Совет

Сведения о функциях API Microsoft можно найти на сайте MSDN. Чтобы быстро получить к нему доступ, я создал на панели инструментов своего браузера закладку на страницу расширенного поиска Microsoft. Введя имя нужной мне функции, например, *GetSystemTimeAsFileTime*, я получу не только сведения об этой функции API, но и важную дополнительную информацию.

Имея информацию о функциях, импортируемых приложением, вы получите общее представление о том, что делает (и не делает) это приложение. Например, если приложение не импортирует ни один из DLL-файлов, содержащих сетевой код, либо в виде низкоуровневых функций сокета, либо высокоуровневых функций Internet API, то оно вряд ли может быть программой скрытого удаленного администрирования или использоваться для передачи информации из системы в Интернет. Это полезный прием, который я часто использовал, чтобы получить информацию о приложении. Однажды мне дали исполняемый файл и попросили узнать, может ли он быть программой скрытого удаленного администрирования или выполнять ее возможности. Задокументировав сведения о файле, я изучил таблицу импорта и увидел, что ни один из импортируемых DLL-файлов не предоставляет возможностей подключения к сети. Затем я провел более глубокий анализ, исследовав импортируемые функции, и обнаружил, что, хотя некоторые функции предоставляли математическую функциональность, ни одна из них не поддерживала возможность подключения к сети.

Еще один полезный инструмент для просмотра информации о DLL-файлах и функциях, необходимых приложениям, – это Dependency Walker (также известный как «depends.exe»), доступный на веб-сайте с таким же именем. На илл. 6.8 показана часть графического интерфейса программы Dependency Walker, в которой открыто тестовое приложение «dcode.exe».



Илл. 6.8. Часть графического интерфейса Dependency Walker.

Как показано на илл. 6.8, приложение «dcode.exe» основано на функциях из файла «MSVBVM60.DLL», который в свою очередь зависит от функций из шести других DLL-файлов (каждый DLL-файл поставляется с последними дистрибутивами Windows). На илл. 6.9 показана часть функций, экспортируемых файлом «MSVBVM60.DLL», в окне программы Dependency Walker.

E	Ordinal ^	Hint	Function	Entry Point
C	100 (0x0064)	60 (0x003C)	ThunRTMain	0x0000DE3E
C	101 (0x0065)	73 (0x0049)	VBDllUnRegisterServer	0x00018CFC
C	102 (0x0066)	70 (0x0046)	VBDllCanUnloadNow	0x0002C692
C	103 (0x0067)	72 (0x0048)	VBDllRegisterServer	0x000A4A8A
C	104 (0x0068)	71 (0x0047)	VBDllGetClassObject	0x00028FCA
C	105 (0x0069)	69 (0x0045)	UserDllMain	0x00018BA7
C	106 (0x006A)	13 (0x000D)	DllRegisterServer	0x000D3AD5
C	107 (0x006B)	14 (0x000E)	DllUnregisterServer	0x000D3CB3
C	108 (0x006C)	94 (0x005E)	__vbaAryLock	0x000E24D0

Илл. 6.9. Функции, экспортируемые файлом «MSVBVM60.DLL».

Dependency Walker позволяет вам увидеть не только DLL-файлы и функции, которые импортирует исполняемый файл (будь то файл с расширением .exe или .dll), но и функции, экспортируемые DLL-файлами. Мы подробнее поговорим о таблице экспорта в следующем разделе.

Dependency Walker также имеет полезную функцию профилирования, которая позволяет установить специальные параметры для того, как будет осуществляться профилирование модуля или приложения, а затем запустить приложение, чтобы увидеть, какие модули (DLL-файлы) будут загружены. Таким образом можно отследить различные вызовы функций DLL и возвращаемые значения во время работы приложения. Этот способ может быть полезен для обнаружения модулей, которые загружаются динамически, но не перечислены в таблицах импорта других модулей, или для определения причины, по которой появляется сообщение о том, что не удалось правильно инициализировать приложение. Однако этот способ не относится к статическому анализу, так как требует запуска файла.

Таблица экспорта (EXPORT)

Так как все DLL-файлы предоставляют функции, которые могут импортировать другие исполняемые файлы, сами DLL-файлы содержат список доступных функций в своих (как вы, наверно, уже догадались) таблицах экспорта. Это функции, которые могут импортировать или использовать другие исполняемые файлы (DLL-файлы, EXE-файлы и т. д.), чтобы авторам приложений не нужно было писать свой собственный код для всего, что они хотели бы сделать в системе. DLL-файлы выступают в роли библиотек или репозиториев предварительно написанного кода, которые можно использовать в системе. Утилита «pedump.exe» создает дамп таблиц экспорта DLL-файлов. Например, вот часть таблицы экспорта для файла «ws2_32.dll»:

```
Таблица экспорта:
  Name:            WS2_32.dll
  Characteristic   00000000
  s:
  TimeDateStamp:   41107EDA -> Wed Aug 04 02:14:50 2004
  Version:         0.00
  Ordinal base:    00000001
  #           of   000001F4
  functions:
  # of Names:      00000075

  Entry            Pt Ordn Name
  00011028         1 accept
  00003E00         2 bind
  00009639         3 closesocket
  0000406A         4 connect
  00010B50         5 getpeername
  0000951E         6 getsockname
  000046C9         7 getsockopt
  00002BC0         8 htonl
  00002B66         9 htons
  00004519         10 ioctlsocket
  00002BF4         11 inet_addr
```

Если у вас есть опыт работы с UNIX или программирования сокетов на Perl, вы увидите, что экспортируемые функции предоставляют основные возможности для передачи данных по сети. Например, функции *bind( )* и *accept( )* используются службами или демонами, прослушивающими соединения (программы скрытого удаленного администрирования и т. д.), а функция *connect( )* используется клиентскими приложениями (например, веб-браузерами и чат-ботами), которые подключаются к северам.

Следует отметить, что, помимо экспортирования своих собственных функций, DLL-файлы могут импортировать функции из других DLL-файлов. Например, используя «pedump.exe», чтобы просмотреть информацию для файла «ws2_32.dll», мы видим, что исполняемый файл импортирует функции из таких файлов, как «kernel32.dll», «ws2help.dll», «ntdll.dll» и других. Некоторые DLL-файлы импортируют функции из других DLL-файлов, чтобы усовершенствовать основные предоставляемые функциональные возможности. Такие инструменты, как Dependency Walker, покажут вам эти связанные или каскадные DLL-зависимости в удобном графическом интерфейсе.

Ресурсы

Часто PE-файл содержит раздел ресурсов с именем «.rsrc», а также каталог данных ресурсов. В разделе ресурсов может храниться информация о таких элементах, как диалоговые окна и значки, а также другие полезные данные, которые помогут

идентифицировать файл; однако, возможно, самая важная информация во время анализа исполняемого файла – сведения о версии файла.

Perl-скрипт «fvi.pl» (доступный на носителе, который идет в комплекте с этой книгой) использует модуль Win32::File::VersionInfo, чтобы извлечь сведения о версии из PE-файла, если таковые имеется. Скрипт «fvi.pl» принимает имя файла (с полным путем к нему) в качестве единственного аргумента и возвращает найденную информацию в формате, показанном ниже:

```
C:\Perl>fvi.pl c:\windows\system32\svchost.exe
Filename           : c:\windows\system32\svchost.exe
Type               : Application
OS                 : NT/Win32
Orig Filename      : svchost.exe
File Descriptoin   : Generic Host Process for Win32 Services
File Version       : 5.1.2600.2180 (xpsp_sp2_rtm.040803-2158)
Internal Name      : svchost.exe
Company Name       : Microsoft Corporation
Copyright          : • Microsoft Corporation. All rights reserved.
Product Name       : Microsoft« Windows« Operating System
Product Version    : 5.1.2600.2180
Trademarks :
```

Нужно учитывать пару моментов при использовании таких инструментов. Во-первых, модуль Win32::File::VersionInfo используется только на платформе Windows. Во-вторых, ни модуль, ни Perl-скрипт не предпринимают попыток проверить, что исследуемый файл на самом деле является PE-файлом. То есть если скрипт «fvi.pl» не возвращает никакой информации, это не значит, что исследуемый файл представляет собой вредоносную программу. Фактически многие авторы вредоносного ПО стараются, чтобы такая информация не компилировалась в их программы, тогда как другие включают в них фальшивые сведения о версии файла, чтобы сбить экспертов с толку. Некоторые даже сохраняют информацию о версии файла просто ради развлечения.

Записки из подполья…

Бот Russiantopz

Выполняя анализ чат-бота «russiantopz», я обнаружил интересные данные об этой программе (которая называлась «statistics.exe»). Оказалось, что этот чат-бот был написан кем-то, кто не живет в России! Не обращая внимания на имя файла и тщательно изучив сведения о версии, я обнаружил, что на самом деле этот файл является копией IRC-клиента «mirc32.exe» (www.mirc.com/get.html). IRC-клиент с графическим интерфейсом был скрыт на рабочем столе с помощью файла «team-scan.exe»», который в действительности был копией утилиты «hidewndw.exe» (http://premium.caribe.net/~adrian2/creations.html), написанной Адрианом Лопесом (Adrian Lopez).

Хотя поиск сведений о версии файла не всегда является определяющим средством анализа, он предоставляет новую информацию, которая дополняет общую картину исследования.

Запутывание кода

До сих пор мы использовали обычные, надежные исполняемые файлы, чтобы продемонстрировать различные структуры PE-файлов. Хотя вы можете использовать эти инструменты и приемы, чтобы идентифицировать файлы, создатели вредоносных программ часто пытаются замаскировать или запутать код своих файлов, не только для того чтобы они не были обнаружены администраторами и экспертами, но и чтобы скрыть их от антивирусных приложений и других программ обеспечения безопасности системы.

Часто создатели вредоносного программного обеспечения применяют инструменты сжатия и даже шифрования данных, чтобы замаскировать свои файлы, или просто создают новые версии своих программ.

Существуют различные утилиты для запутывания кода исполняемых файлов, такие как компоновщики, упаковщики и программы шифрования (шифраторы). Мы по очереди познакомимся с каждым типом этих инструментов.

Компоновщики

Компоновщики – это утилиты, позволяющие пользователю связать одно приложение с другим; часто используются при создании троянских программ. Смысл в том, чтобы у пользователя возникло желание запустить приложение-носитель (это может быть игра или другой исполняемый файл). После этого пользователь видит, что приложение выполняется, и ему кажется, что все в порядке. Однако в это же время запускается троянская программа, часто в фоновом режиме, без ведома потерпевшего. Одним из первых доступных компоновщиков был eLiTeWrap (http://homepage.ntlworld.com/chawmp/elitewrap/), но Silk Rope и SaranWrap (http://packetstormsecurity.org/trojans/bo/index3.html) стали популярными, когда группа хакеров Cult of the Dead Cow выпустила утилиту Back Orifice. Судя по обзорам и описаниям вредоносного ПО, опубликованным на антивирусных и других веб-сайтах, похоже, что создатели таких программ больше не считают компоновщики модными и «крутыми» инструментами. Это главным образом связано с тем, что компоновщики оставляют после себя сигнатуры, которые давно известны антивирусным приложениям.

Хотя многие компоновщики доступны под разными именами, все они выполняют одну основную функцию: связывают один исполняемый файл с другим. Утилита ELiTeWrap, возможно, уникальна тем, что позволяет пользователю сконфигурировать скрипт команд, которые будут запущены, или откликов, которые будут предоставлены, предлагая таким образом дополнительную функциональность связанным исполняемым файлам.

Предупреждение
Загрузив ELiTeWrap версии 1.04 на компьютер с ОС Windows XP Pro SP2, я несколько раз пытался создать пригодный для работы связанный пакет программ, но мне ни разу не удавалось этого сделать. Я пытался использовать ELiTeWrap в интерактивном режиме, а также применять скрипт. Все рано я получал выходной файл, который был меньше любого из входных файлов, а при попытке запустить выходной файл появлялось диалоговое окно с сообщением «Ошибка № 27 при чтении пакета» (“Error #57 reading package”).

Упаковщики

«Упаковщики» (англ. ***packer***) – еще одно название для программ, позволяющих пользователю сжимать свои приложения, чтобы сэкономить место на накопителе. Другое название таких инструментов – «программы сжатия данных». Хотя это не является сегодня большой проблемой из-за увеличения емкости накопителей, сжатый исполняемый файл действительно можно передать по сети быстрее, что теоретически позволит избежать того, что он будет обнаружен хостовыми или сетевыми антивирусами и системами обнаружения вторжений. Кроме того, применение упаковщиков усложняет анализ исполняемых файлов. Некоторые законные компании упаковывают свои программы, чтобы увеличить скорость их работы (уменьшить время загрузки с накопителя в ОЗУ) или защитить производственные секреты. Несмотря на то, что существует множество упаковщиков, наиболее популярными считаются такие программы, как ASPack (www.aspack.com) и UPX (http://upx.sourceforge.net).

ASPack сжимает исполняемый файл, записывая небольшую подпрограмму распаковки в его конец. Затем точка входа в исполняемый файл изменяется и указывает на начало подпрограммы распаковки, а исходная точка входа сохраняется. Когда исполняемый файл распаковывается в память, точка входа восстанавливается в исходное значение. Один из признаков применения программы ASPack – наличие разделов с такими именами, как «.adata», «.udata» и «.aspack» (но не забывайте, что имена разделов – это просто имена, и они могут быть изменены). Насколько мне известно, существуют инструменты, которые могут распаковывать файлы, сжатые программой ASPack.

UPX – еще один упаковщик, который можно использовать не только для сжатия, но и для распаковки файлов, сжатых UPX; то есть он также является распаковщиком для самого себя. Один из признаков того, что вы работаете с файлом, сжатым UPX, – это наличие разделов с именами UPX0 и UPX1, но не следует забывать, что эти имена могут быть легко изменены посредством редактирования PE-файла в шестнадцатеричном редакторе.

Это лишь несколько примеров утилит сжатия, используемых авторами вредоносных программ, но существует множество других подобных приложений. В зависимости от того, какая утилита сжатия использовалась, вы можете найти инструмент или подключаемый модуль, который предназначен для распаковки этого алгоритма. Возможно, вам придется потратить время на поиск в Интернете, чтобы узнать, является ли процесс распаковки опцией утилиты или для этого существует отдельное приложение.

Инструмент ProcDump32 (www.fortunecity.com/millenium/firemansam/962/html/procdump.html) имеет возможность распаковывать стандартные алгоритмы сжатия. На илл. 6.10 показано диалоговое окно «Выберите распаковщик» (“Choose Unpacker”) программы ProcDump32, в котором пользователь может указать алгоритм, использовавшийся для сжатия исполняемого файла.

Илл. 6.10. Диалоговое окно «Выберите распаковщик» (“Choose Unpacker”) программы ProcDump32.

Программа ProcDump32 также содержит другие функциональные возможности; например, она позволяет пользователю создавать на накопителе дамп запущенного процесса, распаковывать или расшифровывать PE-файл, используя стандартные алгоритмы, и редактировать заголовки PE-файлов. Вы уже встречали другие инструменты, позволяющие выполнять все эти операции, но программа ProcDump32 действительно предоставляет довольно полезные функциональные возможности, и ее следует включить в ваш набор инструментов для анализа вредоносных файлов.

Шифраторы

«Шифраторы» – приложения, позволяющие пользователю шифровать свои программы. Шифрование исполняемого файла – еще один способ, используемый создателями вредоносных программ для того, чтобы файл не был обнаружен хостовыми и сетевыми антивирусами и системами обнаружения вторжений. На самом деле это довольно популярный способ запутывания кода вредоносной программы, и в одних случаях может использоваться распространенный алгоритм шифрования или тот, который как минимум можно обнаружить (по какой-нибудь сигнатуре), а в других случаях алгоритм может быть совершенно неизвестен.

В качестве примера мы рассмотрим вредоносную программу, о которой нам известно, что ее код был некоторым образом запутан. Одно время на сайте Honeynet Project публиковались интересные задачи «Сканирование месяца» (“Scan of the Month”, SotM) (http://old.honeynet.org/scans/index.html), в которых предлагались различные данные и сценарии для пользователей, желающих попробовать свои силы в расшифровывании. Интересно, что через некоторое время ответы, предоставленные пользователями, оценивались и публиковались, чтобы можно было увидеть подробное решение задачи. Эд Скудис (Ed Skoudis) предлагает похожие задачи на своем сайте CounterHack.net.

Например, в задаче Honeynet SotM № 32 нужно было проанализировать вредоносный исполняемый файл «rada.exe». На илл. 6.11 показан значок этого файла.

Илл. 6.11. Значок вредоносного файла «rada.exe».

Используя утилиты «pedump.exe» и PEView для просмотра файла «rada.exe», мы видим, что он имеет обычный заголовок PE-файла, и, похоже, все интерпретируется нормально. Я имею в виду, что эти утилиты могут проанализировать заголовок PE-файла, и он понятен с точки зрения анализа. Если бы он был неправильным, указатели бы содержали адреса странных разделов файла или областей, которые находятся за пределами исполняемого файла.

Возможно самое интересное в этом файле то, что он имеет три раздела: JDR0, JDR1 и .rsrc. Мы уже знакомы с разделом «.rsrc», но другие два раздела нам не встречались в PE-файлах, которые мы анализировали до этого. Также мы должны обратить внимание на то, что в таблице импорта перечислены только два DLL-файла: KERNEL32.DLL и MSVBVM60.DLL, как показано ниже:

```
Таблица импорта:
 KERNEL32.DLL
 OrigFirstThunk:    00000000 (Unbound IAT)
 TimeDateStamp:     00000000 -> Wed Dec 31 19:00:00 1969
 ForwarderChain:    00000000
 First thunk RVA:   00010BE0
 Ordn  Name
    0  LoadLibraryA
    0  GetProcAddress
    0  ExitProcess
 MSVBVM60.DLL
 OrigFirstThunk:    00000000 (Unbound IAT)
 TimeDateStamp:     00000000 -> Wed Dec 31 19:00:00 1969
 ForwarderChain:    00000000
 First thunk RVA:   00010BF0
```

```
 Ordn  Name
  618
```

Это странно, так как мы знаем, что это вредоносная программа, а любая вредоносная программа, которая в действительности что-нибудь делает, должна импортировать больше двух DLL-файлов и определенно больше трех функций из файла KERNEL32.DLL.

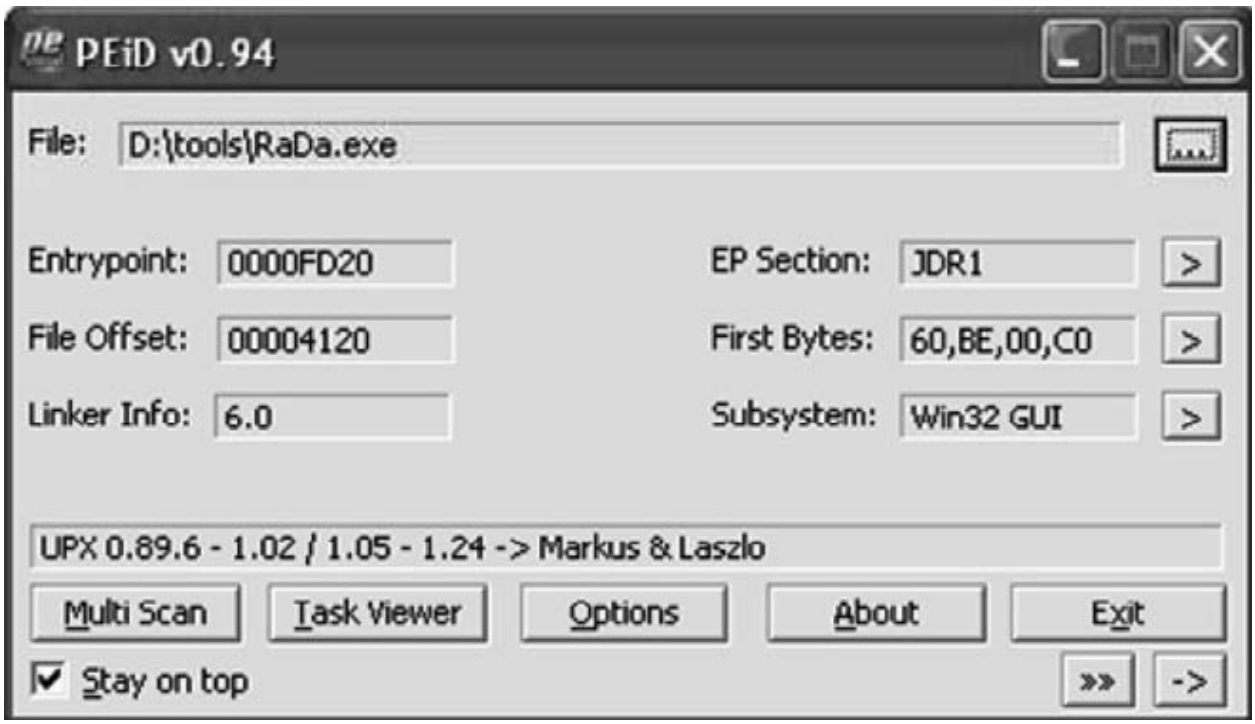
Совет
Это также отличный способ быстро определить вредоносную программу с запутанным кодом. Если в таблице импорта показан только файл KERNEL32.DLL (или может быть этот и пара других DLL-файлов) и несколько функций импортируемых из DLL-файла, в том числе *LoadLibraryA* и *GetProcAddress*, это свидетельствует о том, что код файла был запутан каким-либо образом.

Другой импортируемый модуль (MSVBVM60.DLL) – это файл среды выполнения Visual Basic. В выходных данных скрипта «fvi.pl» показано, что описание файла из раздела ресурсов этого DLL содержит строку «Visual Basic Virtual Machine». Таким образом, мы можем сделать вывод, что сама вредоносная программа была написана с использованием Visual Basic. Этот вывод также подтверждается в ответах к данной задаче, содержащихся на сайте Honeynet.

```
Filename           : d:\tools\rada.exe
Type               : Application
OS                 : Unknown/Win32
Orig Filename      : RaDa
File Description   :
File Version       : 1.00
Internal Name      : RaDa
Company Name       : Malware
Copyright          :
Product Name       : RaDa
Product Version    : 1.00
Trademarks         :
```

Интересно, что автор сообщает нам о том, что эта программа действительно является вредоносной. Не надейтесь, что такое будет часто происходить, если вообще будет.

Теперь, когда у нас есть явные признаки того, что код вредоносной программы запутан (хотя мы немного жульничали, так как уже знали, что в выбранной программе код запутан), мы бы хотел узнать, каким образом это было сделано. Использовался ли упаковщик? Применялось ли только сжатие или также шифрование? Мы можем использовать удобный инструмент PEiD (http://peid.has.it/), чтобы исследовать этот файл и попытаться определить способ запутывания кода. На илл. 6.12 показана программа «rada.exe», загруженная в PEiD.



Илл. 6.12. Программа «rada.exe», загруженная в PEiD.

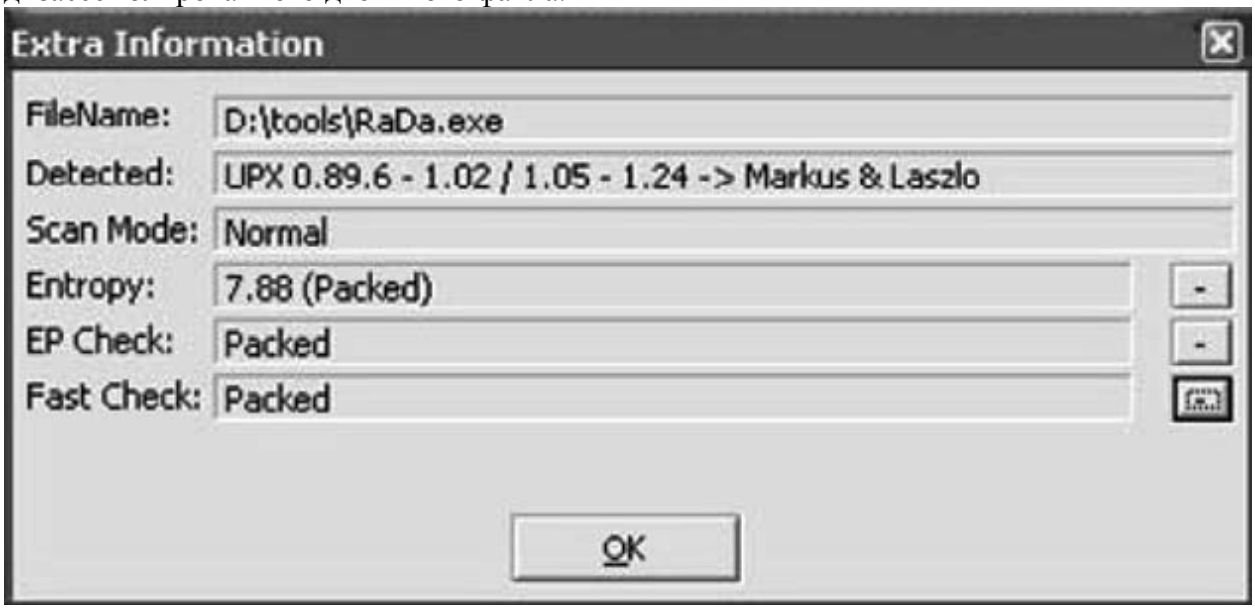
Обратите внимание, что инструмент PEiD определил, что для запутывания кода использовалась утилита сжатия UPX. Это интересно, так как разделы, указанные в программе PEView, назывались JDR0 и JDR1, а не UPX0 и UPX1. Как упоминалось выше, разделы с именами UPX0 и UPX1 указывают на утилиту сжатия UPX. Это свидетельствует о том, что если информация PEiD верна, то автор вредоносной программы использовал редактор, чтобы изменить эти имена разделов.

Совет

Все, что мы до сих пор узнали об исполняемых файлах, позволяет нам получить представление о различных аспектах файлов, которые мы можем исследовать, чтобы установить тип самого файла. Особенно это поможет нам понять, что сообщает антивирусные приложения, или что они не сообщают, когда не могут идентифицировать последнюю версию какой-нибудь вредоносной программы. В этих случаях может быть полезен такой инструмент, как Yara (http://code.google.com/p/yara-project/). Проект Yara предоставляет платформу, которая должна помочь специалистам по анализу вредоносных программ и расследованию инцидентов распознать и классифицировать различные аспекты вредоносных файлов. Yara работает в ОС Windows и Linux, а также доступна в виде модуля на Python. Дон Уэбер (Don Weber) расширил Python-модуль Yara (www.cutawaysecurity.com/blog/archives/422) до утилиты Yara-Scout Sniper или yara-ss, обладающей такими дополнительными возможностями, как например доступ к удаленным системам. Вы можете также использовать находящиеся в свободном доступе сигнатуры PEiD (файл «userdb.txt» на странице www.peid.info/BobSoft/Downloads.html) как часть правил Yara. Такой инструмент (особенно когда он поддерживается сообществом пользователей) может быть чрезвычайно полезен при определении и классифицировании новых версий вредоносных программ.

PEiD выявляет применение упаковщиков, шифраторов и компиляторов, определяя местонахождение точки входа приложения и анализируя байты в этом месте, чтобы установить метод запутывания кода. Создатели PEiD собрали сигнатуры для многих инструментов запутывания кода и включили их в программу PEiD. Также в PEiD было добавлено несколько полезных инструментов, в том числе средство просмотра запущенных процессов и модулей, которые они используют, диалоговое окно для

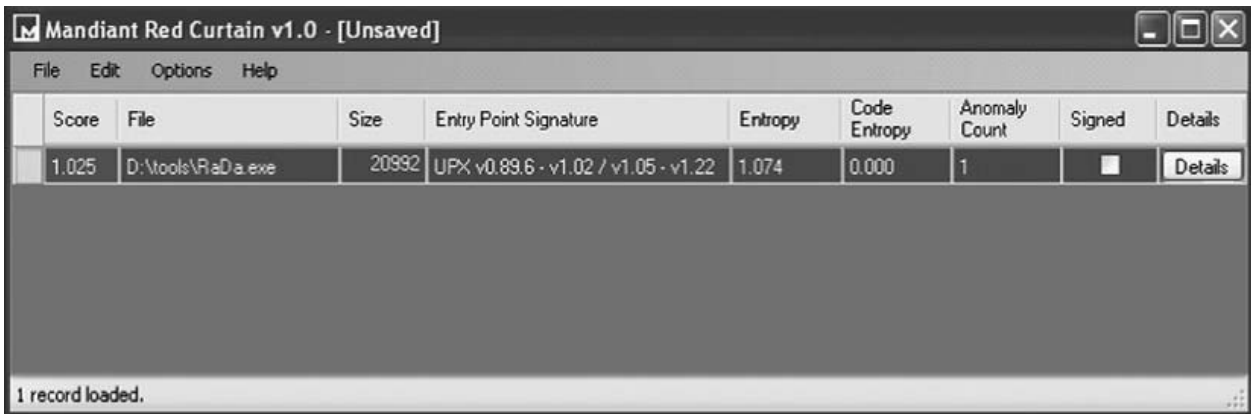
просмотра дополнительной информации о файле (см. илл. 6.13), диалоговое окно для просмотра заголовка PE-файла и даже диалоговое окно для просмотра дизассемблированного двоичного файла.



Илл. 6.13. Диалоговое окно с дополнительной информацией о файле «rada.exe», загруженном в PEiD.

Если у вас будет возможность загрузить программу PEiD и файл «rada.exe», запустите дизассемблер, щелкнув по кнопке со стрелкой справа от текстового поля «Первые байты» (“First Bytes”). Если вы знакомы с программированием на языке ассемблера (я не занимался этим с тех пор, как окончил школу, когда мы программировали для микропроцессора Motorola 68000), то сразу обратите внимание на команды перехода и множество команд сложения в списке. Если вам интересны подробности анализа этого исполняемого файла, изучите ответы, опубликованные на сайте Honeynet, особенно ответ, предоставленный Крисом Иглом (Chris Eagle). Крис – известный инструктор и докладчик на конференциях BlackHat (www.blackhat.com), а также старший преподаватель и заместитель председателя кафедры информатики в Высшей военно-морской школе (Naval Postgraduate School) в Монтерее (штат Калифорния).

Инструмент Red Curtain от компании Mandiant (www.mandiant.com/software/redcurtain.htm) предлагает более широкие функциональные возможности, чем рассмотренные ранее утилиты, в том числе PEiD. Этот инструмент анализирует заголовок исполняемого файла, пытаясь найти энтропию/беспорядочную информацию, признаки сжатия или запутывания кода, наличие цифровых сигнатур, а также другие характеристики исполняемого файла, и создает оценочный отчет об угрозах. Такой отчет предназначен для того, чтобы указать эксперту, нужно ли исследовать файл более тщательно. На илл. 6.14 показан файл «rada.exe», открытый в программе Red Curtain от Mandiant.



Илл. 6.14. Файл «rada.exe», загруженный в Red Curtain.

Б**о**льшая часть данных, предоставляемых программой Red Curtain, похожа на информацию, доступную в PEiD. Щелкните по кнопке «Подробности» (“Details”) в правой части интерфейса Red Curtain, чтобы открыть диалоговое окно, в котором показаны разделы PE-файла (JDRO, JDR1, .rsrc), информация о них, а также сведения об идентифицированной аномалии (в данном случае – «checksum_is_zero», что свидетельствует о возможном умышленном изменении файла). Дополнительные сведения о Red Curtain см. в руководстве пользователя, доступном в меню «Справка» (“Help”).

Совет

Если вас интересует более глубокое исследование исполняемых файлов вообще и вредоносных файлов в частности, рекомендуется познакомиться с ответами к задачам SotM № 32 и № 33 на сайте Honeynet. Вы не только увидите общность всех анализов, но и найдете информацию о других инструментах, которые можно использовать, чтобы получить лучшие результаты при исследовании файлов.

Динамический анализ

Динамический анализ включает в себя запуск исполняемого файла в контролируемой среде, чтобы можно было изучать и документировать его влияние на работу системы. Это чрезвычайно полезный механизм анализа, так как он дает вам более точное представление о том, что и в какой последовательности делает вредоносная программа в системе. Это особенно поможет вам в тех случаях, когда вредоносная программа сжата или зашифрована, так как исполняемый файл должен быть распакован и/или расшифрован в памяти, прежде чем он будет выполнен. Поэтому, помимо следов и артефактов вредоносной программы, вы сможете, используя способы сбора и анализа содержимого памяти (рассмотренные в главе 3), увидеть выполнение самой программы в режиме реального времени.

Условия проведения исследований

Если вы собираетесь проводить динамический анализ вредоносного файла, один из первых вопросов, которые нужно рассмотреть, – среда или условия проведения исследования. В конце концов, вряд ли вы захотите узнать, что делает вредоносная программа, запустив ее в сети предприятия и позволив ей выйти из-под контроля. Довольно неприятно, когда такое происходит случайно, и уж наверняка вы не захотите делать это специально.

Один из способов создания условий для проведения исследований – поместить компьютер в отдельной сети без возможности электрического соединения (обратите внимание, что я не говорю о логическом соединении или виртуальной локальной сети на основе коммутатора) с остальной частью вашей сети. Здесь необходим этот «воздушный промежуток»; я настоятельно рекомендую вам даже не думать о том, чтобы установить

какой-нибудь рубильник, который бы отделял «клетку» с вредоносной программой от остальной части сети, так мы все знаем, что однажды появится тот, кто забудет переключить рубильник и разделить сети. Кроме того, если вас проверяет какой-нибудь орган контроля, меньше всего вам захочется, чтобы у вас обнаружили способ, с помощью которого вредоносная программа (способная украсть конфиденциальные личные данные) может проникнуть в сеть, где хранятся такие данные. Если ваша лаборатория аккредитована или сертифицирована соответствующим органом, вы серьезно рискуете этим статусом, запуская ненадежные программы в работающей сети. Потеря этой аккредитации вряд ли сделает вас популярным среди тех, кто останется в вашей организации. Это касается не только лабораторий, аккредитованных для судебного анализа, но также и других регулирующих органов.

Один из недостатков работы с «одноразовыми» компьютерами состоит в том, что вам нужно переустанавливать операционную систему после каждого исследования; как еще вы сможете гарантировать, что собираете достоверные данные и что результаты не испорчены вредоносной программой? Один из способов добиться этого – использовать виртуализацию.

Виртуализация

Если у вас нет системы, которую вы можете постоянно переустанавливать и возвращать в первоначальное состояние (да и кто на самом деле захочет делать это?), виртуализация – еще один доступный для вас вариант. Существует несколько бесплатных и коммерческих инструментов виртуализации, например:

- **Bochs** Бесплатный инструмент с открытым исходным кодом, работающий на таких платформах, как Windows, Linux и даже Xbox (http://bochs.sourceforge.net/).
- **Parallels** Работает на платформах Mac, Windows и Linux (www.parallels.com).
- **Microsoft Virtual PC** Бесплатный инструмент, работающий на платформе Windows (используемой в качестве хостовой ОС); позволяет запускать DOS, Windows и OS/2 в качестве гостевых ОС (www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx).
- **Virtual Iron** Использует полную («встроенную») виртуализацию (т. е. не устанавливается на хостовую операционную систему) и может запускать Windows и Linux с производительностью, близкой к оригинальной (www.virtualiron.com/).
- **Win4Lin** Работает на Linux; позволяет запускать приложения Windows (http://win4lin.net/content/).
- **VMware** Работает на платформах Windows и Linux и позволяет запускать ряд гостевых ОС. Продукты VMware Server и VMware Player доступны бесплатно. VMware считается стандартом де-факто для продуктов виртуализации и подробно рассматривается в последующих разделах (www.vmware.com).

Совет

В январе 2009 года, когда я готовил рукопись этой книги для отправки издателю, мое внимание привлек интересный инструмент Zero Wine (http://zerowine.sourceforge.net/). Zero Wine – это виртуальная среда на основе программы QEMU с установленной гостевой ОС Debian Linux. Гостевая операционная система также имеет установленное программное обеспечение Wine и в начале работы она запускает веб-платформу для динамического анализа вредоносных программ. Фактически вы можете загрузить исполняемый файл вредоносной программы так же, как на веб-сайт VirusTotal.com, но в случае с Zero Wine вредоносная программа выполняется в виртуальной среде, и все действия в системе (доступ к функциям API, изменения в реестре), а также данные статического анализа, регистрируются и предоставляются эксперту.

Конечно, этот список не полный. Способ виртуализации, который вы выберите, во многом зависит от ваших потребностей, условий (т. е. имеющиеся системы, бюджет и т. д.) и уровня комфорта при работе с различными хостовыми и гостевыми ОС. Если вы не уверены, какой вариант больше всего подходит вам, посмотрите на страницу «Сравнение виртуальных машин» (“Comparison of virtual machines”, http://en.wikipedia.org/wiki/Comparison_of_virtual_machines) в Википедии. Это поможет вам сократить количество вариантов исходя из условий, бюджета и объема работ, необходимого для установки и запуска виртуальной платформы.

Преимущество использования виртуальной системы для анализа вредоносных файлов заключается в том, что вы можете создать снимок состояния системы, а затем «заразить» ее и выполнить все виды исследований. Собрав все необходимые данные, вы можете вернуть систему в первоначальное состояние (в котором система находилась до заражения). Таким способом можно не только более легко восстановить системы, но и исследовать несколько версий похожих вредоносных программ на одной и той же платформе для более точного сравнения.

Возможно, самая известная платформа виртуализации – это VMware. Компания VMware предоставляет несколько продуктов для виртуализации бесплатно, например, программы VMware Player, которая позволяет запускать виртуальные машины (но не создавать их), и VMware Server. Кроме того, ряд предварительно созданных образов виртуальных машин доступен пользователю для загрузки и использования. На момент написания этой книги, для загрузки были доступны образы виртуальных машин ISA Server и Microsoft SQL Server.

Нужно сделать одну оговорку относительно использования VMware, которая также относится к другим средам виртуализации. Не так давно начались разговоры о программном обеспечении, которое можно использовать, чтобы обнаружить присутствие среды виртуализации. Вскоре после этого эксперты начали встречать вредоносные программы, которые не только обнаруживали присутствие среды виртуализации, но также работали по-другому или совсем не функционировали. Девятнадцатого ноября 2006 года Ленни Зельцер (Lenny Zeltser) опубликовал статью на сайте ISC Handler’s Diary (http://isc.sans.org/diary.php?storyid=1871), в которой обсуждалось обнаружение вредоносными программами виртуальных машин посредством использования коммерческих инструментов. Нельзя забывать об этом при проведении динамического анализа вредоносных программ. Постарайтесь тщательно опросить всех пользователей, которые были свидетелями инцидента, и определить как можно больше возможных артефактов, прежде чем забирать образец вредоносной программы в лабораторию. Таким образом, если вы видите совершенно другое поведение вредоносного файла при его запуске в виртуальной среде, то, возможно, вы обнаружили вредоносную программу, содержащую этот код.

«Одноразовые» системы

Если виртуализация вам просто не подходит (из-за недостатка средств, опыта, удобства и т. д.), можно выбрать вариант работы с «одноразовыми» системами, то есть с системами, которые можно быстро клонировать и восстановить. Некоторые корпоративные организации применяют такие инструменты, как Norton Ghost от компании Symantec, чтобы создавать образы систем, установленных на одинаковом оборудовании. Таким образом, можно использовать стандартную сборку образа для установки систем, что позволяет облегчить управление ими. Другие организации используют подобный подход для учебного оборудования, позволяя системным администраторам быстро возвращать все системы в известное состояние. Например, однажды я проводил оценку уязвимости систем для организации, имеющей учебное оборудование. Сотрудники с гордостью говорили мне, что, используя Norton Ghost, они

могут полностью переустановить операционные системы на всех 68 учебных компьютерах с помощью одной дискеты.

Если вы решили выбрать этот вариант, нужно убедиться, что компьютеры никоим образом не подключены к сети корпорации или предприятия. Возможно, вы думаете, что это очевидно, но иногда функционирование сетей было нарушено из-за занятого администратора или неправильно сконфигурированной виртуальной локальной сети на основе коммутатора. Вы должны убедиться, что между платформой для проведения исследований и любыми другими сетями находится не просто логический промежуток. Рекомендуется обеспечить «воздушный промежуток».

После того как вы выберите платформу, можно следовать тем же процедурам сбора и анализа данных, которые вы бы использовали в виртуальной среде; эти процедуры действительно не отличаются. Однако при работе с «одноразовой» системой нужно будет добавить какой-нибудь способ для захвата содержимого памяти на вашей платформе (не забывайте, что сеансы VMware можно просто приостановить), особенно если вы анализируете вредоносную программу с запутанным кодом.

Инструменты

Можно использовать различные инструменты для наблюдения за системами при исследовании вредоносных файлов. Как правило, нужно иметь все необходимые инструменты под рукой, прежде чем запускать образец вредоносной программы. Кроме того, нужно знать, что эти инструменты умеют делать, и уметь их использовать.

Одно из основных отличий между анализом вредоносной программы и расследованием инцидента состоит в том, что во время анализа у вас есть возможность настроить испытательную систему перед тем, как она будет заражена. Несмотря на то, что теоретически системные администраторы имеют такую же возможность, вам не часто встретятся крупные серверные системы, тщательно сконфигурированные с учетом проблем безопасности и особенно расследования инцидентов.

Есть несколько дополнительных трудностей, о которых вы должны быть осведомлены при исследовании вредоносных файлов. Например, вы не знаете, что будет делать вредоносная программа после запуска. Понимаю, это кажется очевидным, но я неоднократно встречал людей, которые не учитывали этого. Я хочу сказать, что вы не знаете, как будет вести себя вредоносная программа: откроется и будет ждать, пока ее проанализируют, или быстро сделает свою работу и исчезнет. Я видел разные вредоносные программы: одни открывали порт и ожидали соединения (утилиты скрытого удаленного администрирования), другие пытались подключиться к компьютерам в Интернете (чат-боты), третьи за долю секунды внедряли код в другой выполняющийся процесс и исчезали. При проведении динамического анализа у вас есть возможность повторять «преступление» снова и снова, чтобы попытаться увидеть его подробности. Во время расследования инцидента мы главным образом делаем снимки места происшествия посредством инструментов для сбора информации о состоянии системы в отдельные моменты времени. Это похоже на ведение наблюдения с помощью фотоаппарата Polaroid. Во время динамического анализа мы наблюдаем за местом происшествия с помощью прямой видеосъемки, когда мы можем собирать данные за непрерывный период времени, а не в отдельные моменты. Таким образом, будем надеяться, мы сможем собрать и проанализировать данные о том, что происходит в течение всей жизни вредоносной программы.

Итак, какие инструменты нам нужно использовать? Во-первых, нам нужно зарегистрировать всю доступную информацию о сетевых подключениях, так как вредоносная программа может пытаться соединиться с удаленной системой и/или открыть порт и ожидать соединений. Один из способов сделать это – запустить в сети анализатор сетевых пакетов, например, Wireshark (ранее известный как Ethereal, доступный на сайте www.wireshark.org). Если вы используете автономную систему, вам нужно будет

установить анализатор пакетов на другом компьютере, а если вы работаете с VMware, вам потребуется запустить Wireshark в хостовой операционной системе и выполнять вредоносную программу в одной из гостевых операционных систем. Причина, по которой нужно сделать это, вскоре станет понятной.

Еще один инструмент, который потребуется установить на компьютере, – это Port Reporter (http://support.microsoft.com/kb/837243), свободно доступный на сайте Microsoft. Port Reporter запускается как служба в системах Windows и записывает данные об использовании портов TCP и UDP. В ОС Windows XP и Windows 2003 программа Port Reporter регистрирует используемые сетевые порты, процессы и службы, использующие эти порты, модули, загружаемые этими процессами, и учетные записи, от имени которых выполняются эти процессы. В ОС Windows 2000 регистрируется меньше информации. В Port Reporter есть различные опции конфигурации: например, можно указать, где будут создаваться файлы журналов, как будет запускаться служба (автоматически при загрузке системы или, по умолчанию, вручную), и т. д. Этими опциями можно управлять посредством параметров командной строки, добавляемых в диалоговое окно запуска службы, после установки Port Reporter. Перед установкой Port Reporter не забудьте прочитать статью из базы знаний, чтобы понимать, как работает эта программа и какую информацию она может предоставить.

Совет

Некоторые вредоноснее программы прекращают функционировать и просто завершают работу, если не могут подключиться к компьютеру (например, к серверу контроля и управления) в Интернете. Один из способов преодолеть эту проблему – проанализировать сетевой трафик, генерируемый процессом, и посмотреть, выполняется ли в нем поиск DNS для отдельного имени хоста. Затем можно изменить файл «hosts» (расположенный в каталоге %WinDir%\system32\drivers\etc), чтобы направить вашу систему на отдельный компьютер в вашей сети, а не на компьютер в Интернете. Дополнительную информацию о том, как системы Windows преобразовывают имена узлов в сетях TCP/IP, см. в статье № 172218 из базы знаний Microsoft (http://support.microsoft.com/kb/172218).

Port Reporter создает три типа файлов журнала: журнал инициализации (т. е. «PR-INITIAL-*.log», где звездочка заменяет дату и время создания журнала в 24-часовом формате), регистрирующий информацию о состоянии системы в момент запуска службы; журнал портов (т. е. «PR-PORTS-*.log»), предоставляющий информацию о сетевых подключениях и использовании портов, как утилита «netstat.exe»; и журнал идентификаторов процессов (т. е. «PR-PIDS-*.log»), содержащий информацию о процессах.

Microsoft также предоставляет веб-трансляцию (http://support.microsoft.com/kb/840832) об инструменте Port Reporter и его функциональных возможностях. Кроме того, на сайте Microsoft доступна программа Port Reporter Parser (http://support.microsoft.com/kb/884289), которая значительно облегчает анализ объемных журналов Port Reporter.

Имея под рукой такие инструменты наблюдения, вам, возможно, интересно, зачем нужно запускать анализатор сетевых пакетов в другой системе. Почему нельзя запустить его на той же платформе, используемой для динамического анализа, вместе с остальными инструментами наблюдения? Все дело в руткитах, которые мы будем рассматривать в главе 7. Если вкратце, руткиты позволяют вредоносным программам скрывать свое присутствие на компьютере, не позволяя операционной системе «видеть» определенные процессы, сетевые подключения и т. д. На момент написания этой книги, не проводилось тщательное исследование с использованием различных руткитов, поэтому мы должны быть уверены, что собираем максимально возможное количество информации. Запуская

анализатор сетевых пакетов на другой платформе, отдельной от платформы, используемой для исследования, мы гарантируем, что часть процесса наблюдения не будет подвергаться воздействию выполняющейся вредоносной программы.

Совет

Во время динамического анализа также рекомендуется выполнить сканирование «зараженной» системы с другого компьютера. Такое сканирование может показать программу несанкционированного удаленного администрирования, которая работает в системе, но скрыта с помощью других средств, например, руткитов (мы подробнее рассмотрим руткиты в главе 7). Можно использовать такие инструменты, как Nmap (http://nmap.org/) и PortQry (http://support.microsoft.com/kb/832919), чтобы быстро просканировать «зараженную» систему и даже попытаться определить тип службы, ожидающей соединения на отдельном порту. Хотя проблемы подключения по TCP/IP и использования метода «port knocking» не рассматриваются в этой книге, всегда существует возможность, что определенные запросы (или комбинации запросов), отправленные на открытый порт «зараженной» системы, могут стать причиной того, что процесс, связанный с этим портом, будет реагировать некоторым образом.

Не забывайте, что мы, как судебные эксперты, должны понимать, что факт отсутствия артефакта сам по себе является артефактом. В контексте динамического анализа это означает, что если мы видим, что из платформы, используемой для исследования, идет сетевой трафик в Интернет (или выполняется поиск других систем в локальной подсети), но не наблюдаем никаких признаков этих процессов с помощью инструментов наблюдения на этой платформе, то, возможно, мы имеем дело с руткитом.

Здесь я хотел бы подчеркнуть необходимость тщательного документирования процесса динамического анализа вредоносных программ. Мне встречались вредоносные программы, которые не имели возможностей руткита, но вместо этого внедряли код в область памяти другого процесса и выполнялись оттуда. Необходимо понимать это, так как предположение, что этот случай связан с руткитом, может привести к неправильному отчету, а также к неправильным действиям при расследовании этого инцидента. Суть в том, что если вы документируете процедуры и инструменты, которые вы используете, то кто-нибудь другой сможет проверить ваши результаты. В конце концов, применяя одни и те же инструменты и процедуры к одной и той же вредоносной программе, кто-нибудь другой сможет увидеть тот же результат, ведь так? Или же этот человек сможет ознакомиться с вашим процессом и поинтересоваться причиной отсутствия или использования отдельного инструмента, что позволит провести более тщательное изучение и исследование вредоносной программы.

Совет

При проведении динамического анализа вредоносной программы вы должны быть готовы к самым разным вариантам развития событий, но в то же время не нужно создавать себе лишние трудности, загружая в систему все доступные инструменты, иначе вы потратите столько времени на управление ими, что забудете о том, какой файл вы анализируете. Однажды я проводил проверку по просьбе клиента и обнаружил необычный файл. Первый признак работы этого файла был найден в реестре; при запуске он добавлял параметр в раздел Run пользователя, а также в раздел RunOnce. Интересно, что он добавлял параметр в раздел RunOnce, помещая перед именем файла символ *, что указывало операционной системе анализировать и запускать содержимое раздела, даже если система начинала работу в безопасном режиме (довольно ловко!). Мне пришлось прибегнуть к динамическому анализу, так как во время статического анализа было обнаружено, что вредоносный файл зашифрован, а программа PEiD не смогла определить используемый способ шифрования. Запустив вредоносную программу на своей платформе

и проанализировав собранные данные, я увидел, что эта программа запускала веб-браузер так, что процесс браузера выполнялся, а графический интерфейс не отображался, и затем добавлялась в область памяти процесса браузера. Исходя из этого я смог определить, что после запуска вредоносной программы нужно дополнительно анализировать процесс браузера. Это также объясняло, почему во время анализа энергозависимых данных я видел, что браузер отвечает за необычные сетевые подключения, и не было никаких признаков работы вредоносного процесса.

Также рекомендуется включить аудит отслеживания процессов в журнале событий как для успешных, так и для неудачных событий. Журнал событий может помочь вам отслеживать несколько различных действий в системе, в том числе, использование привилегий, события входа в систему, доступ к объектам (этот параметр требует, чтобы вы также сконфигурировали списки управлении доступом для объектов – файлов, каталогов, разделов реестра и т. д. – за которыми вы хотите специально вести наблюдение) и тому подобное. Так как во время динамического анализа вредоносных файлов нас интересуют процессы, включение аудита отслеживания процессов для успешных и неудачных событий может предоставить нам полезную информацию. Используя инструмент «auditpol.exe» из пакета Resource Kit (который мы рассматривали в главе 1), можно настроить политику аудита на компьютере, используемом для динамического анализа, а также подтвердить, что политика настроена правильно перед проведением исследования. Например, используйте следующую команду, чтобы убедиться, что включен правильный режим аудита:

```
C:\tools>auditpol /enable /process:all
```

Для того чтобы перед исследованием подтвердить, что правильный режим аудита все еще включен, просто запустите «auditpol.exe» из командной строки без аргументов.

Совет

Возможно, вы также захотите включить аудит системных событий, но не нужно активировать слишком много политик аудита. Существует такое понятие, как наличие избыточных данных, и это действительно может замедлить процесс анализа, особенно если эти данные не очень полезны. Некоторые считают, что им нужно отслеживать все события, чтобы ничего не пропустить, но существует ограничение на количество данных, которые мы можем эффективно использовать и анализировать. Тщательно оцените то, что вы планируете сделать, создайте конфигурацию для исследовательской платформы и придерживайтесь ее, если только у вас не будет серьезной причины ее изменить. Избыток данных может быть так же вреден для судебного эксперта, как и их недостаток.

Как упоминалось выше, один из способов отследить доступ к файлам и разделам реестра – включить аудит доступа к объектам, настроить списки управления доступом для всех объектов, которые вас интересуют, и, выполнив вредоносную программу, попытаться проанализировать содержимое журнала событий. В качестве альтернативы можно рассмотреть два способа отслеживании доступа к файлам и разделам реестра: 1) создание снимков состояния системы до и после выполнения вредоносной программы и сравнение их; 2) использование наблюдения в реальном времени. При проведении динамического анализа вредоносных файлов лучший вариант – использовать оба способа, а для этого вам понадобится несколько инструментов. Посетите веб-сайт Microsoft и загрузите утилиты FileMon и RegMon (позволяющие наблюдать за действиями в файловой системе и реестре в реальном масштабе времени) или программу Process Monitor. Преимущество использования инструментов наблюдения в реальном времени, а не программ создания снимков системы, состоит в том, что вы не только видите файлы и разделы реестра, которые создаются или изменяются, но также можете увидеть файлы и разделы реестра, поиск которых выполнялся, но не дал положительного результата. Более того, вы можете

увидеть временную шкалу действий с указанием порядка, в котором осуществлялся доступ к файлам и разделам реестра. Это может быть важной частью анализа вредоносных программ.

Совет
FileMon и RegMon – отличные инструменты наблюдения, доступные на веб-сайте Sysinternals от Microsoft (http://technet.microsoft.com/en-us/sysinternals/bb795535.aspx). Хотя оба этих инструмента все еще предоставляются отдельно, их функциональные возможности добавлены в программу Process Monitor, которая также доступна на том же сайте.

Мы рассмотрим некоторые инструменты для создания снимков системы в следующем разделе.

Процесс

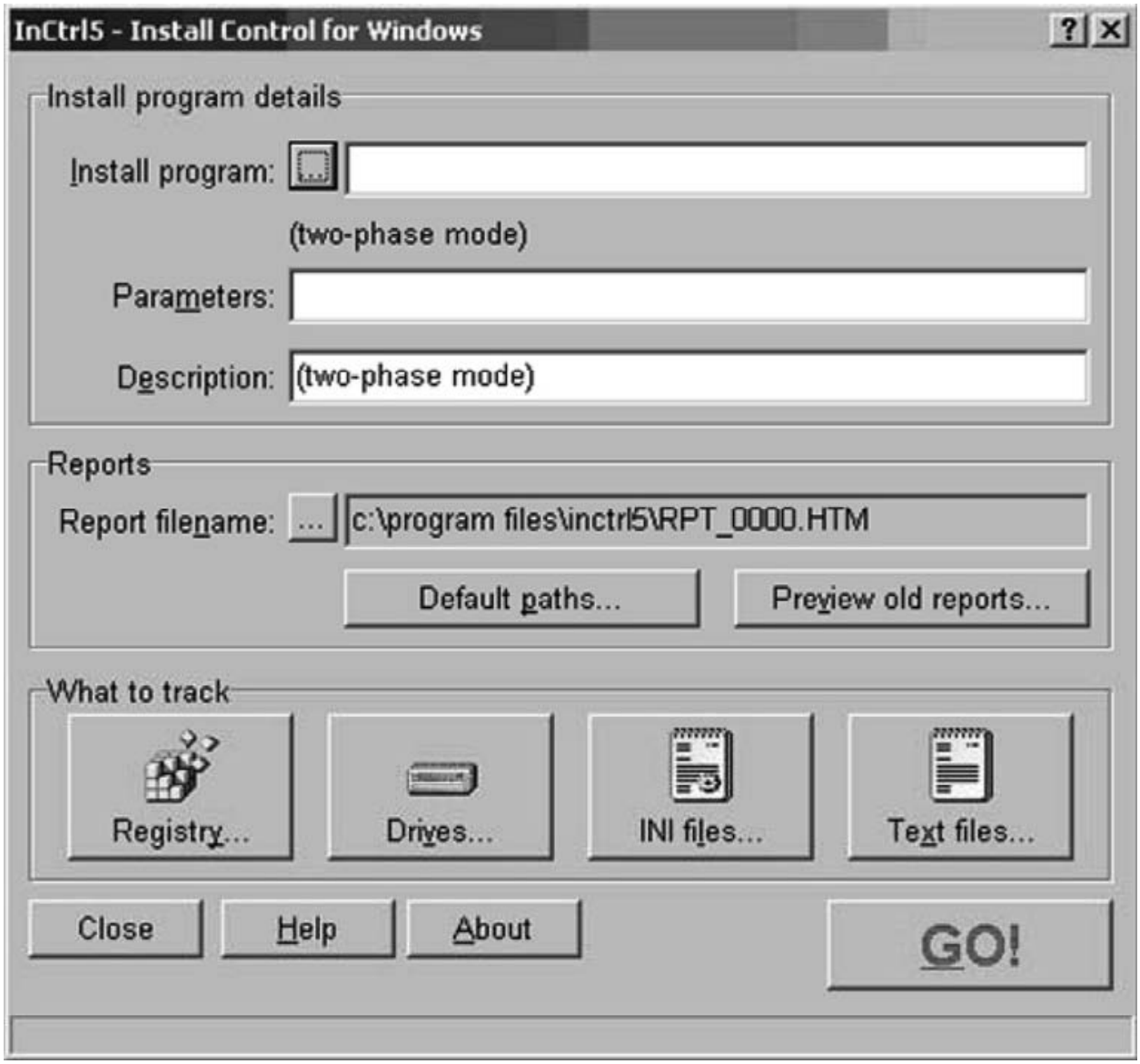
Процесс настройки платформы для проведения анализа вредоносных программ довольно простой, и самое главное – составить план работы или контрольный список действий. Как и во время сбора энергозависимых данных или судебной экспертизы, не стоит пытаться каждый раз проводить динамический анализ по памяти, так как иногда мы можем быть слишком загружены работой или просто можем забыть о каком-нибудь важном этапе этого процесса. Мы все можем ошибаться; в моей практике также были случаи, когда приходилось начинать анализ с самого начала из-за того, что я забыл активировать один из своих инструментов. Мне пришлось возвращаться назад и снова переустанавливать уже зараженную систему, а затем проверять, что все мои инструменты установлены и конфигурация системы верна. Я уверен, вам не нужно объяснять, как я был расстроен.

Первое, что нужно сделать, – убедиться, что вы определили, загрузили и установили все инструменты, которые вам могут понадобиться. В этой главе было рассмотрено достаточное количество программ, но, возможно, в будущем появятся другие инструменты, которые вы захотите использовать. Составьте список инструментов, используемых для динамического анализа, и обновляйте его при необходимости. Время от времени показывайте его другим экспертам, добавляйте новые инструменты, удаляйте старые и т. д.

Как только все инструменты будут установлены, убедитесь, что понимаете, как их использовать, а также удостоверьтесь, что знаете и понимаете необходимые параметры конфигурации. Большинство инструментов нужно запускать вручную, поэтому составьте контрольный список того, в каком порядке вы будете их запускать. Например, такие инструменты, как Regshot (http://sourceforge.net/projects/regshot/), см. илл. 6.15, и InControl5, см. илл. 6.16, создают снимки системы для сравнения, поэтому их нужно запускать на первом этапе (создание исходных снимков), а затем запускать инструменты наблюдения в реальном времени.

Илл. 6.15. Графический интерфейс программы Regshot.

Regshot сохраняет свои результаты в формате обычного текста или HTML. При использовании инструментов наблюдения и создания снимков (например, Regshot), имейте в виду, что большинство из них отслеживают изменения только в контексте их пользователя или ниже. Это означает, что если вы запустите инструменты от имени учетной записи администратора, то сможете отслеживать изменения, сделанные в контексте этого пользователя или ниже, но не изменения, сделанные от имени учетных записей системы.



Илл. 6.16. Графический интерфейс программы InControl5.

InControl5 предоставляет аккуратный отчет (в формате HTML, таблицы или обычного текста) о файлах или разделах реестра, которые были добавлены, изменены или удалены. InControl5 также отслеживает изменение в отдельных файлах, хотя список таких файлов ограничен. Кроме того, можно указать InControl5 выполнять наблюдение за программой установки, например, за MSI-файлом. Однако я не часто видел, чтобы троянские программы или черви распространялись в виде файлов установщика Microsoft.

После того как вы запустите вредоносную программу и соберете необходимые данные, нужно остановить работу инструментов наблюдения в реальном времени и во второй раз запустить инструменты создания снимков для сравнения. На данном этапе вы сами решаете, когда сохранять журналы инструментов наблюдения в реальном времени – до или после того, как вы запустите инструменты создания снимков во второй раз. Ваша испытательная платформа предназначена для ваших целей и не будет использоваться как улика, поэтому вы сами выбираете прядок заключительных этапов. Лично я сначала сохраняю данные, собранные инструментами наблюдения в реальном времени, а затем завершаю создание снимков. Я знаю, что увижу новые файлы, созданные инструментами наблюдения, в выходных данных инструментов для создания снимков, и я знаю, когда и как сохраняются эти файлы. Следовательно, я могу легко отделить эти данные от данных, сгенерированных вредоносной программой.

Кроме того, рекомендуется создавать отдельные каталоги для всех файлов журналов. Это позволяет легко разделять данные во время динамического анализа, а

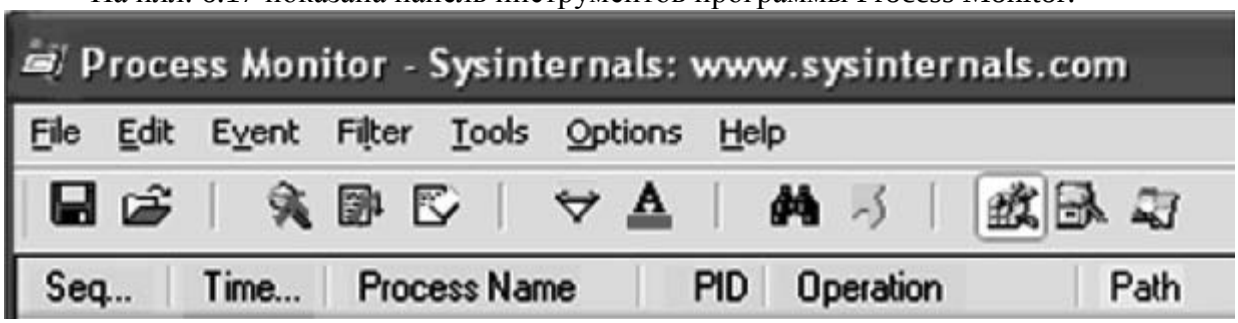
также упрощает сбор данных из системы после завершения наблюдения. Фактически можно даже подключать съемный USB-накопитель к системе и отправлять на него все файлы журналов.

Короче говоря, процесс динамического анализа выглядит приблизительно так:

- Убедитесь, что все инструменты наблюдения обновлены/установлены; сверьтесь с контрольным списком.
- Убедитесь, что все инструменты наблюдения правильно сконфигурированы.
- Выберите место для сохранения журналов (локальный НЖМД, съемный USB-накопитель и т. д.).
- Приготовьте вредоносный файл, который вы будете анализировать (скопируйте его в систему для исследования, документируйте сведения о его местонахождении в файловой системе).
- Выполните первый этап работы инструментов для создания снимков.
- Запустите инструменты наблюдения в реальном времени.
- Запустите вредоносный файл (документируйте способ запуска: назначенное задание, двойной щелчок через оболочку, запуск из командной строки и т. д.).
- Остановите работу инструментов наблюдения в реальном времени и сохраните их данные в ранее выбранном месте.
- Выполните второй этап работы инструментов для создания снимков и сохраните их данные в ранее выбранном месте.

Я знаю, что это выглядит довольно просто, но вы будете удивлены, сколько важных и полезных данных можно потерять, если *не* придерживаться этого процесса. Следует надеяться, что у вас есть общий план действий, которого можно придерживаться, и теперь можно уточнить его, добавляя названия нужных вам инструментов. Со временем эти инструменты могут изменяться. Например, довольно долгое время RegMon и FileMon с сайта Sysinternals.com были предпочтительными инструментами для наблюдения за тем, как процессы получают доступ к реестру и файловой системе.

На илл. 6.17 показана панель инструментов программы Process Monitor.



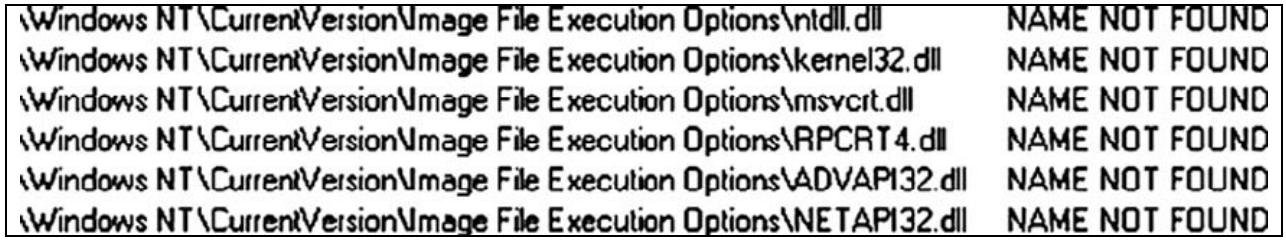
Илл. 6.17. Панель инструментов Process Monitor со знаками RegMon и FileMon.

Если вы работали с RegMon или FileMon раньше, то панель инструментов программы Process Monitor, показанная на илл. 6.17, покажется вам знакомой; большинство значков те же и расположены в том же порядке, что и в двух старых приложениях.

При использовании Process Monitor для сбора информации о доступе к реестру и файловой системе вы должны знать, что программа регистрирует *все* доступы, и поэтому вам придется обрабатывать достаточно большой объем данных. Например, щелкните по значку лупы с красным знаком X сверху и просто наблюдайте, не прикасаясь к клавиатуре или мыши. Данные о событиях начнут немедленно появляться в окне Process Monitor, несмотря на то, что вы совсем ничего не делаете! Очевидно, в ОС Windows каждую секунду происходит довольно много событий, которых вы не видите. При просмотре информации, собранной в окне Process Monitor, можно щелкнуть правой кнопкой мыши по записи и в контекстном меню выбрать пункт «**Исключить**» (“Exclude”), а затем – «**Имя процесса**» (“Process Name”), чтобы отфильтровать ненужные процессы и удалить лишние данные.

Совет

Если помните, в главе 4 мы говорили о разделе реестра Image File Execution Options. Process Monitor может показать, как ОС Windows получает доступ к этому разделу. Чтобы увидеть это, откройте командную строку и введите **net use**, но не нажимайте клавишу Enter. Откройте Process Monitor и начните сбор информации о доступе к реестру. Вернитесь в командную строку, нажмите Enter, и как только увидите, что команда завершилась, остановите сбор данных в Process Monitor, щелкнув по значку лупы, после чего над ним должен появиться красный знак X. На илл. 6.18 демонстрируется часть собранной информации, которая показывает, как процесс «net.exe» пытается определить, есть ли в разделе Image File Execution Options параметры выполнения для указанных DLL-файлов.



Илл. 6.18. Часть собранных данных в Process Monitor, которые показывают обращение к разделу реестра Image File Execution Options.

Можно также подумать об использовании других инструментов. Например, в статье «Malware Analysis Software Tools» (написанной Рассом Макри (Russ McRee) и опубликованной в 2007 году в июльском номере журнала ISSA Journal в рубрике *toolsmith*, http://holisticinfosec.org/toolsmith/docs/july2007.pdf) демонстрируется работа инструмента SysAnalyzer от компании iDefense (http://labs.idefense.com/software/malcode.php). SysAnalyzer позволяет вам отслеживать состояние работающей системы при выполнении вредоносной программы во время динамического анализа. SysAnalyzer наблюдает за различными аспектами системы во время выполнения вредоносной программы, поэтому само собой разумеется, что система будет заражена; однако использование виртуальной системы позволяет достаточно легко вернуться в первоначальное состояние.

Можно выполнить еще одно заключительное действие – создать дамп физической памяти (ОЗУ), используя один из способов, рассмотренных в главе 3. Так вы не только получите все данные динамического анализа, показывающие, какие изменения сделала вредоносная программа в системе, но и в случае с запутанным кодом будете иметь возможности извлечь исполняемый файл из дампа ОЗУ и получить представление о том, как на самом деле выглядит программа, что улучшит качество вашего анализа.

Совет

В этой главе представлено много полезной информации, которая поможет вам понять PE-файлы Windows и их структуру. Вместе с тем издательство Syngress Publishing летом 2008 года выпустило книгу *Malware Forensics: Investigating and Analyzing Malicious Code*, и считается, что ее авторы (Кэмерон Малин (Cameron Malin), Оуэн Кейси (Eoghan Casey) и Джеймс Аквилина (James Aquilina)) создали, возможно, самое полезное и исчерпывающее руководство по этой теме из доступных сегодня, в котором с различных точек зрения рассматриваются вредоносные программы как для Windows, так и для Linux. Одна из многих ценных особенностей этой книги – обзор большого количества бесплатных инструментов, которые можно использовать при проведении различных видов анализа.

Краткое изложение

В этой главе мы рассмотрели два способа, которые можно применять для сбора информации об исполняемых файлах. Понимая отдельные структуры исполняемого файла, вы будете знать, что искать и на что обращать внимание, особенно если злоумышленник выполнил определенные действия, чтобы попытаться защитить файл от анализа. Способы анализа, рассмотренные в данной главе, позволят вам определить, какое воздействие оказывает (вредоносная) программа на систему и какие артефакты свидетельствуют о ее присутствии. Эти знания будут особенно полезными для эксперта, когда антивирусная программа не может обнаружить вредоносный файл, а отчеты или описания производителя антивирусной программы составлены недостаточно подробно. Рассмотренные артефакты также помогут специалисту по расследованию инцидентов найти в сетевой инфраструктуре другие компьютеры, которые могут быть заражены. Эксперту эти артефакты предоставят более полное представление о заражении и о том, что вредоносная программа сделала в системе. В случае с троянскими программами и программами скрытого удаленного администрирования эти артефакты помогу вам создать временную шкалу действий в системе.

Каждый рассмотренный способ анализа имеет свои преимущества и недостатки, и, как и в случае с любым другим инструментом, его применение должно быть тщательно обосновано и документировано. Статический анализ позволяет вам увидеть, какие данные можно встретить во вредоносной программе, и дает представление о том, чего можно ожидать при проведении динамического анализа. Однако статический анализ часто предоставляет только ограниченный обзор вредоносной программы. Динамический анализ можно назвать «поведенческим», так при выполнении вредоносной программы в контролируемой среде вы можете увидеть, какое воздействие оказывает вредоносная программа на целевую систему, и в каком порядке. Но динамический анализ нужно применять с особой осторожностью, так как вы фактически запускаете вредоносную программу, и, если вы не будете внимательны, это может закончиться заражением всей сетевой инфраструктуры.

Даже если вы не будете проводить никаких анализов вредоносного файла, не забудьте полностью документировать сведения о нем, укажите, где он был обнаружен в файловой системе, какие другие файлы с ним связаны, вычислите его криптографический хэш и т. д. Авторы вредоносных программ не всегда присваивают своим приложениям имена, которые свидетельствуют о чем-то плохом, например, «syskiller.exe» (рус. *убийца системы*). Как правило, имя вредоносной программы безобидно или даже призвано ввести эксперта в заблуждение, поэтому очень важно документировать всю информацию о вредоносном файле.

Быстрое повторение

Статический анализ

- Документирование сведений о любом вредоносном приложении или файле, найденном во время экспертизы, – первый шаг для определения его функции и назначения.
- Содержимое подозрительного исполняемого файла может быть непонятным для большинства пользователей, но если вы разбираетесь в структурах, используемых для создания таких типов файлов, то сможете понять, как можно воспользоваться двоичной информацией из файла во время экспертизы.
- Не стоит опираться только на имена файлов при исследовании подозрительной программы. Известно, что даже опытные специалисты по анализу вредоносных программ попадаются на удочку злоумышленника, которому требуется только

несколько минут, чтобы попытаться «скрыть» свое приложение, присвоив ему безобидное имя.

Динамический анализ

- Процесс динамического анализа позволяет вам увидеть, какое воздействие оказывает вредоносная программа на систему.
- Применяя сочетание инструментов для создания снимков системы и для наблюдения в реальном времени, вы можете определить артефакты, образовавшиеся в результате заражения, и последовательность их создания.
- При проведении динамического анализа рекомендуется использовать инструменты наблюдения, не находящиеся на исследовательской платформе, чтобы сбор информации не подвергался влиянию вредоносной программы.
- После того как анализ вредоносной программы будет завершен, можно провести исследование, традиционными методами, как работающей так и выключенной исследовательской платформы. Это позволит эксперту не только совершенствовать свои навыки, но и провести дополнительную проверку артефактов вредоносной программы.

Часто задаваемые вопросы

Вопрос: При проведении расследования инцидента я обнаружил, что файл с именем «svchost.exe» отвечает за несколько соединений в системе. Заражена ли эта система вредоносной программой?
Ответ: Здесь вопрос не в том, заражена ли система, а в том, является ли файл «svchost.exe» вредоносным. Первое, что мне хотелось бы у вас спросить, – «Что вы сделали, чтобы просмотреть сетевые соединения?». Точнее, какое состояние соединений? Подключения только ожидаются или уже были установлены с другими системами? Во-вторых, какие порты учувствуют в сетевых соединениях? Связаны ли они обычно с файлом «svchost.exe»? Наконец, где в файловой системе вы обнаружили этот файл? Файл «svchost.exe» обычно находится в каталоге «system32» и защищен функцией «Защита файлов Windows» (“Windows File Protection”, WFP), которая автоматически выполняется в фоновом режиме. Если нет никаких признаков того, что работа функции WFP была нарушена, вы вычислили криптографический хэш файла «svchost.exe» и сравнили его с заведомо правильным значением? Недостаток знаний об операционной системе при расследовании инцидентов очень часто приводит эксперта к неправильным вводам.

Вопрос: Я обнаружил один PE-файл во время расследования и, открыв его в шестнадцатеричном редакторе, увидел в заголовке сигнатуру «MZ». Но я не увидел обычных имен разделов, таких как «.text», «.idata» и «.rsrc». Почему?
Ответ: Имена разделов в PE-файле не используются самим PE-файлом для какой-то конкретной цели и могут быть изменены без последствий для остальной части этого файла. Хотя обычные PE-файлы и некоторые инструменты сжатия имеют сигнатуры обычных имен разделов, это можно легко изменить. Имена разделов являются лишь одной маленькой частью информации, которую можно использовать, чтобы получить полное представление о файле.

Вопрос: Я провел как статистический, так и динамический анализ подозрительного исполняемого файла и достаточно хорошо понимаю, что он делает и какие артефакты оставляет в системе. Есть ли способ, с помощью которого я могу проверить свои данные?
Ответ: После того как вы завершите собственный анализ, попробуйте использовать антивирусную программу для проверки этого вредоносного файла. В большинстве случаев эксперт делает это в первую очередь, но это не всегда дает результат. Нередки случаи, когда специалист по расследованию инцидентов приходит на объект и

обнаруживает, что в сети свирепствует червь, несмотря на то, что базы антивирусных приложений на всех зараженных системах обновлены. Если проверка с помощью антивирусных приложений не даст результатов, попробуйте загрузить файл на сайт www.virustotal.com, где после проверки файла с использованием более чем двадцати антивирусных программ будет опубликован отчет. Если полученные результаты будут все равно неполными, отправьте файл со всей имеющейся документацией производителю антивирусного приложения, которое вы используете.

Вопрос: Я бы хотел узнать больше об анализе исполняемых и вредоносных файлов. Можете ли вы порекомендовать какие-нибудь источники?
Ответ: Лучший источник сегодня – книга *Malware Forensics: Investigating and Analyzing Malicious Code*, опубликованная издательством Syngress Publishing (авторы: Джеймс Аквилина (James Aquilina), Оуэн Кейси (Eoghan Casey) и Кэмерон Малин (Cameron Malin)). В этой книге рассматривается анализ вредоносных программ как для Windows, так и для Linux, и она предлагает, возможно, наиболее полную и исчерпывающую информацию по этой теме на сегодняшний день. В зависимости от количества времени, которое вы готовы потратить на углубленное изучение этой темы, можно также использовать несколько дополнительных источников по обратной разработке исполняемого кода. Многие приемы, рассматриваемые в этой теме, можно с таким успехов применять во время анализа вредоносных программ. К числу таких источников относятся сайты REblog (http://malwareanalysis.com/communityserver/blogs/geffner/default.aspx) и OpenRCE (www.openrce.org/articles/).

Содержание

http://computer-forensics-lab.org

Перевод:
Бочков Д.С.
Капинус О.В.
Михайлов И.Ю.