

Харлэн Карви

**Криминалистическое исследование
Windows**



Глава 4

Анализ системного реестра

Содержание этой главы:

- § Внутри системного реестра
- § Анализ системного реестра

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

Введение

Для большинства администраторов и судебных экспертов системный реестр, вероятно, выглядит как вход в мрачную, зловещую пещеру в ландшафте операционной системы Windows. Другие, возможно, представляют себе реестр как темную дверь в конце длинного коридора, на которой каракулями написано: «Оставь надежду всяк сюда входящий». Правда состоит в том, что системный реестр – это неиссякаемый источник ценной информации как для администраторов, так и для судебных экспертов. Программное обеспечение, используемое злоумышленниками, во многих случаях оставляет после себя следы в реестре, таким образом предоставляя эксперту сведения об инциденте. Если вы знаете, где искать информацию в реестре и как интерпретировать найденные данные, то сможете понять, какие действия были совершены в системе.

Эта глава поможет вам получить более полное представление о системном реестре и содержащейся в нем ценной информации. Помимо сведений о конфигурации, в реестре Windows хранится информация о недавно открывавшихся файлах и большое количество данных о действиях пользователя. Все это может иметь огромную ценность в зависимости от характера дела, над которым вы работаете, и вопросов, на которые вам необходимо ответить. Большая часть анализа реестра, рассматриваемого в этой главе, будет проводиться «посмертно» – другими словами, после того, как вы создадите образ данных системы. Однако в некоторых случаях будет описан анализ работающего компьютера, а также будут предоставлены примеры того, как выглядят разделы и параметры реестра в работающем компьютере. Во время анализа работающего компьютера необходимо также учитывать несколько второстепенных моментов, на которые мы обратим внимание во время изучения этой темы.

В этой главе мы рассмотрим различные разделы и параметры реестра, которые могут представлять для нас интерес во время экспертизы. Однако не следует рассматривать эту главу как исчерзывающий список всех важных параметров системного реестра. Несмотря на то, что разделы реестра, связанные непосредственно с операционной системой, как правило, остаются постоянными после ее инсталляции, они могут изменяться после установки пакетов обновлений и исправлений, а также в зависимости от версий самой операционной системы. Кроме того, имеется много приложений, каждое из которых добавляет свои собственные записи в реестр, и эти записи также могут изменяться в зависимости от версий этого приложения. Рассматривайте эту главу как справочное руководство, но учитывайте, что оно далеко не полное. Моя задача в этой главе – описать формат файлов реестра, способы исследования реестра, несколько важных разделов в реестре и способы их анализа. Дойдя до конца этой главы, вы должны понимать, что содержится в системном реестре, и уметь находить и анализировать информацию в реестре.

Внутри системного реестра

Итак, что такое системный реестр? Если вы помните времена DOS и первых версий Windows (3.1, 3.11 и т. д.), сведения о конфигурации (драйверах, параметрах) для этих систем в основном содержались в нескольких файлах, а именно в «autoexec.bat», «config.sys», «win.ini» (в Windows) и «system.ini». Различные параметры в этих файлах определяли, какие программы загружались, как выглядела система, и как она отвечала на ввод данных пользователем.

В более поздних версиях ОС Windows эти файлы были заменены реестром, который представляет собой иерархическую централизованную базу данных (<http://support.microsoft.com/kb/256986>), содержащую сведения о конфигурации для приложений, аппаратных устройств и пользователей. Эта «база данных» (я использую этот термин в широком смысле) заменяет текстовые файлы конфигурации, используемые предыдущими версиями операционных систем Microsoft.

Администраторы взаимодействуют с системным реестром в основном через промежуточные приложения, среди которых наиболее распространены редакторы реестра с графическим интерфейсом пользователя, входящие в состав Windows, – «regedit.exe» или «regedt32.exe». Во многих случаях взаимодействие пользователя или администратора с реестром осуществляется посредством программ установки (приложений, исправлений), которые не позволяют пользователю напрямую воздействовать на отдельные разделы и параметры в реестре. В состав ОС Windows XP и 2003 входит инструмент «reg.exe», с которым можно работать как из командной строки, так и используя скрипты. Дополнительную информацию о программах редактирования реестра с графическим интерфейсом пользователя см. во вставке «RegEdit и RegEdt32».

Инструменты и ловушки...

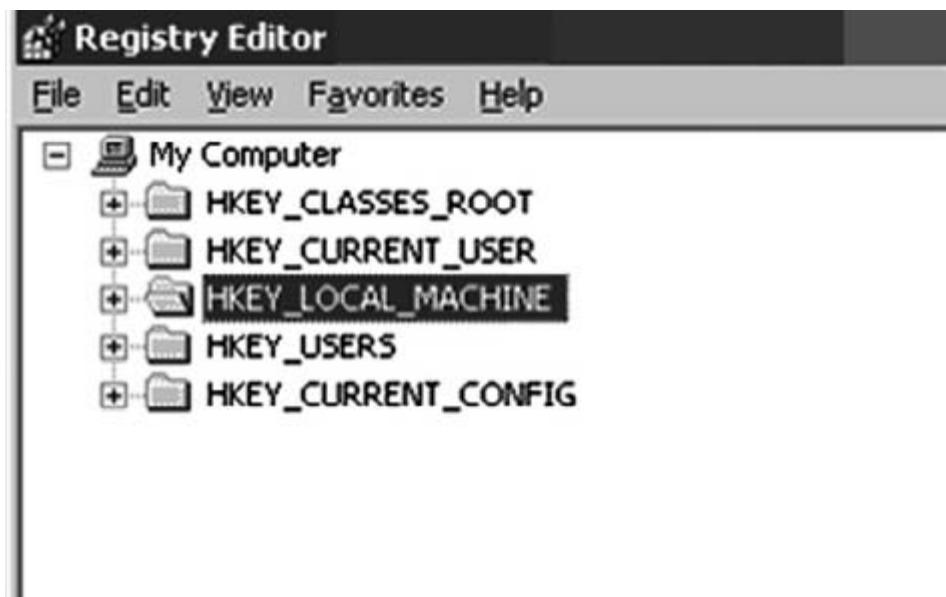
RegEdit и RegEdt32

Интересно, что две служебные программы редактирования реестра, поставляемые с ОС Windows, выполняют разные функции во всех версиях этих систем (<http://support.microsoft.com/kb/141377>).

Например, в ОС Windows NT 2000 редактор «regedit.exe» используется в основном для поиска и не поддерживает изменение списков управления доступом (ACL) в разделах реестра. В «regedit.exe» есть дополнительные ограничения на редактирование типов данных *REG_EXPAND_SZ* (строка переменной длины) или *REG_MULTI_SZ* (несколько строк). Если значения в любом из этих типов данных будут изменены, то данные сохраняются как тип *REG_SZ* (строка фиксированной длины), и функциональность этого раздела будет утрачена. Программа «regedt32.exe», в свою очередь, не позволяет импортировать или экспортить файлы кустов реестра (с расширением .reg).

В ОС Windows XP и 2003 редактор «regedit.exe» позволяет выполнять все эти операции, а «regedt32.exe» – просто небольшая программа, запускающая редактор «regedit.exe».

Когда администратор открывает «regedit.exe», то в области навигации графического интерфейса пользователя он видит пять корневых папок («кустов»), как показано на илл. 4.1. Структура в виде папок позволяет администратору перемещаться по системному реестру так же легко, как и в файловой системе.



Илл. 4.1. Окно программы «regedit.exe», в котором показаны пять корневых кустов.

Каждый из этих кустов играет важную роль в функционировании системы. Куст HKEY_USERS содержит все активные загруженные профили пользователей для данной системы. HKEY_CURRENT_USER – активный загруженный профиль для пользователя, вошедшего в систему в настоящий момент. Куст HKEY_LOCAL_MACHINE содержит множество сведений о конфигурации для компьютера, в том числе, параметры оборудования и программного обеспечения. Куст HKEY_CURRENT_CONFIG содержит профиль оборудования, используемый системой при запуске. Наконец, куст HKEY_CLASSES_ROOT содержит сведения о конфигурации, относящиеся к программам, которые используются для открытия различных файлов в системе. Этот куст является подразделом кустов HKEY_CURRENT_USER\Software\Classes (параметры пользователя) и HKEY_LOCAL_MACHINE\Software\Classes (параметры системы).

Кроме того, не будет лишним знать, откуда берутся кусты и где они находятся на накопителе в файловой системе. Содержимое большей части системного реестра, отображаемое в редакторе реестра, доступно в нескольких файлах, перечисленных в таблице 4.1.

Таблица 4.1

Пути к кустам реестра и соответствующие им файлы

Путь к кусту реестра	Путь к файлу
HKEY_LOCAL_MACHINE\System	%WINDIR%\system32\config\System
HKEY_LOCAL_MACHINE\SAM	%WINDIR%\system32\config\Sam
HKEY_LOCAL_MACHINE\Security	%WINDIR%\system32\config\Security
HKEY_LOCAL_MACHINE\Software	%WINDIR%\system32\config\Software
HKEY_LOCAL_MACHINE\Hardware	Энергозависимый куст
HKEY_LOCAL_MACHINE\System\Clone	Энергозависимый куст
HKEY_USERS\SID пользователя	Профиль пользователя (NTUSER.DAT); Documents and Settings\имя пользователя (в ОС Vista путь изменен на Users\имя пользователя)
HKEY_USERS\Default	%WINDIR%\system32\config\default

Совет

В операционных системах Windows Vista и Windows 7 есть дополнительные файлы кустов реестра: файл «COMPONENTS» (расположенный в каталоге system32\config) и файл «usrclass.dat» (расположенный в каталоге C:\Users\имя_пользователя\AppData\Local\Microsoft\Windows). Мы рассмотрим эти файлы

кустов, в частности «usrclass.dat», позднее в этой главе.

Как видите, некоторые кусты реестра энергозависимы и не существуют в виде файлов на НЖМД. Эти кусты создаются во время запуска системы и недоступны после выключения компьютера. Необходимо помнить об этом не только при проведении судебного анализа образа данных, но и во время исследования работающего компьютера. Если данные в энергозависимых кустах важны для поиска и устранения неисправностей в системе или для расследования инцидента, следует либо экспорттировать весь энергозависимый куст в файл с расширением .reg при помощи программы «regedit.exe», либо использовать другие механизмы для сбора отдельных данных из этого куста перед выключением компьютера.

Помимо разных разделов или кустов, системный реестр поддерживает различные типы данных для параметров, которые он содержит. В таблице 4.2 перечислены эти различные типы данных и их описание.

Таблица 4.2

Типы данных реестра и их описание

Тип данных	Описание
<i>REG_BINARY</i>	Необработанные двоичные данные
<i>REG_DWORD</i>	Данные, представленные в виде 32-разрядного целого числа (длина – 4 байта).
<i>REG_SZ</i>	Текстовая строка фиксированной длины
<i>REG_EXPAND_SZ</i>	Строка данных переменной длины
<i>REG_MULTI_SZ</i>	Несколько строк, разделенных пробелом, запятой или другим разделителем
<i>REG_NONE</i>	Тип данных отсутствует
<i>REG_QWORD</i>	Данные, представленные в виде 64-разрядного целого числа (длина – 8 байт)
<i>REG_LINK</i>	Строка в формате Unicode, обозначающая символьную ссылку
<i>REG_RESOURCE_LIST</i>	Последовательность вложенных массивов, предназначенная для хранения списка ресурсов
<i>REG_RESOURCE_REQUIREMENTS_LIST</i>	Последовательность вложенных массивов, предназначенная для хранения списка возможных аппаратных ресурсов, принадлежащего драйверу устройства
<i>REG_FULL_RESOURCE_DESCRIPTOR</i>	Последовательность вложенных массивов, предназначенная для хранения списка ресурсов, используемого физическим устройством

Как видите, в реестре находятся различные типы данных. Похоже, что нет никаких правил или никакой последовательности для параметров, хранящихся в различных разделах; параметры, служащие для одной и той же цели, могут иметь разные типы данных, что позволяет форматировать и сохранять их данные по-разному. Это может затруднить текстовый поиск данных в системном реестре. Там, где одно приложение сохраняет список недавно использовавшихся документов в виде текстовых строк ASCII, другое может сохранять такой же список как Unicode-строку в виде двоичных данных, и в таком случае эти данные будут пропущены при проведении поиска ASCII-текста. В действительности, в статье из базы знаний Microsoft (<http://support.microsoft.com/kb/161678>) однозначно сказано, что средство поиска в редакторе реестра можно использовать только для нахождения строковых данных ASCII, а не параметров типа *DWORD* или двоичных данных.

Структура реестра в файле куста

Теперь, когда вы знаете, где находятся файлы кустов, давайте заглянем внутрь этих файлов и посмотрим на саму структуру реестра на более низком уровне. Вероятно, сейчас вам интересно, зачем нам это делать. Дело в том, что, понимая основные компоненты реестра, мы сможем собрать дополнительную информацию, используя поиск по ключевым словам в других местах или источниках, таких как файл подкачки, физическая память или даже свободное пространство накопителя. Если знать, что и где искать, можно извлечь дополнительные части информации. Кроме того, имея дополнительные знания о данных, доступных в системном реестре, мы будем лучше понимать, что и где можно найти.

Марк Руссинович (Mark Russinovich) написал отличную статью для журнала *Windows NT Magazine*, которая называется «Inside the Registry» (<http://technet.microsoft.com/en-us/library/cc750583.aspx>) и в которой описываются различные компоненты, или ячейки, реестра. Та же тема рассматривается в книге *Windows Internals*, которую Марк написал вместе с Дэвидом Соломоном (David Solomon).

Каждый тип ячеек имеет особую структуру и содержит особый тип информации. Существуют такие типы ячеек:

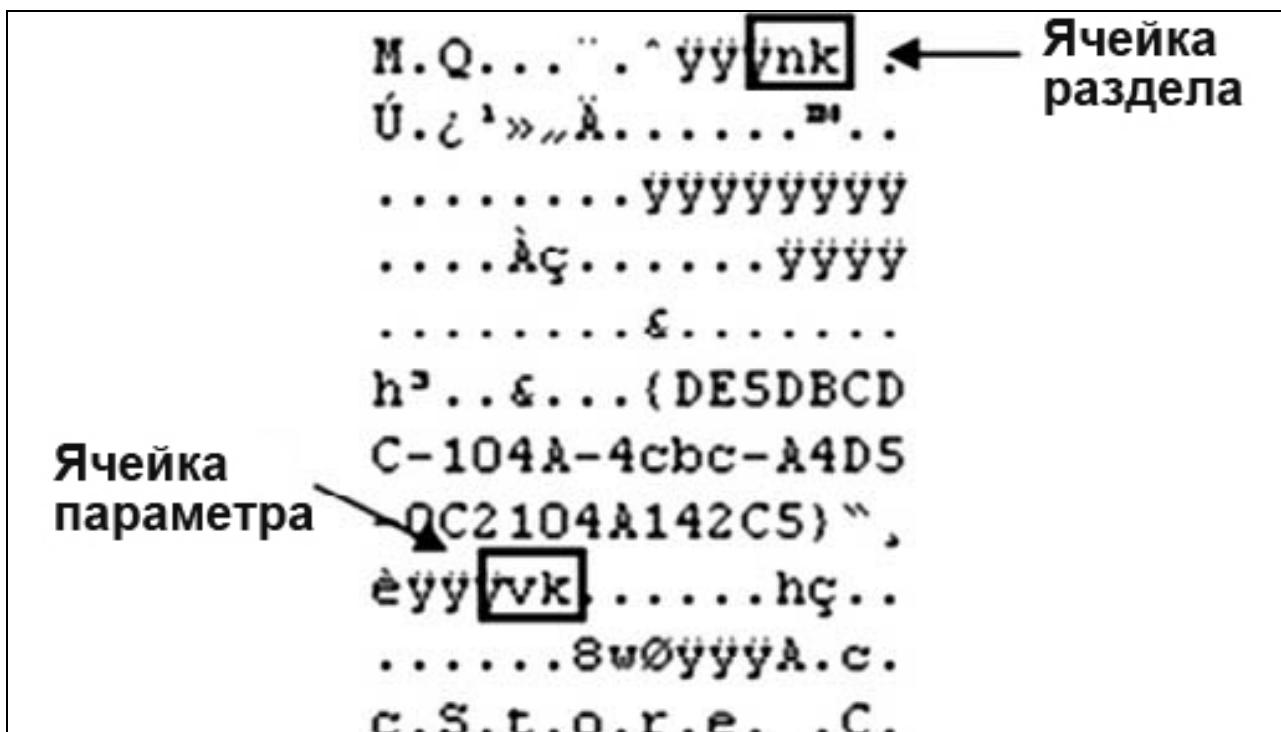
- § **Ячейка раздела.** Содержит информацию о разделе реестра, в том числе смещения по отношению к другим ячейкам, а также время *LastWrite* (время последней записи) для этого раздела (сигнатурой: *kn*).
- § **Ячейка параметра.** Содержит параметр и его данные (сигнтура: *kv*).
- § **Ячейка списка подразделов.** Состоит из ряда индексов (или смещений), указывающих на ячейки разделов; все они являются подразделами ячейки родительского раздела.
- § **Ячейка списка параметров.** Состоит из ряда индексов (или смещений), указывающих на ячейки параметров; все они являются параметрами ячейки общего раздела.
- § **Ячейка дескриптора безопасности.** Содержит информацию о дескрипторе безопасности для ячейки раздела (сигнтура: *ks*).

На илл. 4.2 показаны различные типы ячеек (за исключением ячейки дескриптора безопасности) в том виде, в котором они отображаются в редакторе реестра.



Илл. 4.2. Пример реестра Windows с изображением разделов, параметров и данных.

На илл. 4.3 изображен примерный файл реестра, открытый в шестнадцатеричном редакторе. Здесь показаны сигнатуры ячейки раздела и ячейки параметра. Обратите внимание, что из-за обратного порядка записи байтов сигнтура для ячейки раздела (т. е. *kn*) отображается как *nk*, а для ячейки параметра (т. е. *kv*) – как *vk*.



Илл. 4.3. Фрагмент необработанных данных из файла реестра, в которых показаны сигнатуры ячейки раздела и ячейки параметра.

Как отмечалось ранее, эти сигнатуры могут предоставить нам чрезвычайно ценную информацию во время экспертизы. Используя эти сигнатуры, можно извлечь информацию о разделах и параметрах реестра из свободных кластеров созданного образа данных или даже из дампов ОЗУ (дополнительную информацию о дампах ОЗУ см. в главе 3.)

Совет

Понимание двоичной структуры разделов и параметров реестра может быть чрезвычайно полезным при проведении экспертизы. Большая часть исследований реестра связана с анализом самих файлов реестра при помощи инструментов, рассматриваемых в этой главе. Однако, исследуя свободное пространство накопителя или содержимое дампа памяти, можно также найти сигнатуры, показанные на илл. 4.3. Зная формат структуры разделов и параметров в реестре, можно при необходимости извлечь эти данные и преобразовать их в понятный формат. Например, можно найти раздел реестра в памяти и извлечь время *LastWrite*. Понимание структуры разделов и параметров реестра предоставляет контекст для информации, полученной из дампа ОЗУ или свободного пространства в созданном образе накопителя, а также позволяет провести анализ удаленных ключей в «свободном пространстве» самих файлов кустов. Мы рассмотрим тему поиска разделов реестра в свободном пространстве файлов кустов позднее в этой главе.

Полный справочный материал о фактической программной двоичной структуре различных ячеек в файлах кустов реестра нельзя получить от производителя. Как ни странно, этой информацией владеет Питер Нордал-Хаген (Peter Nordahl-Hagen), создатель утилиты Offline Registry and Password Editor (<http://home.eunet.no/pnordahl/ntpasswd/>). Эта утилита позволяет загрузить компьютер (при условии, что у вас есть к нему физический доступ) с загрузочного диска Linux и редактировать реестр, а также изменять пароли в ОС Windows. Питер предоставляет исходный код для своей утилиты, и если мы загрузим ее, то больше всего нас будет интересовать файл с именем «*ntreg.h*». Это заголовочный файл языка программирования С, определяющий структуры для различных типов ячеек, из которых состоит реестр.

Совет

Тим Морган (Tim Morgan) написал отличную статью, в которой дано описание структуры реестра Windows, в том числе различных типов ячеек и способов их взаимосвязи. Статья доступна на веб-сайте Sentinelchicken.com по адресу www.sentinelchicken.com/data/TheWindowsNTRegistryFileFormat.pdf.

Изучив исходный код утилиты Питера, мы увидим, что длина ячейки раздела равна 76 байт, к которым добавляется имя переменной длины. Следовательно, если мы найдем сигнатуру для ячейки раздела (как показано на илл. 4.3) в дампе ОЗУ или свободных кластерах, то сможем получить содержимое этой ячейки. Отсюда мы можем проанализировать следующие 74 байта (длина сигнатуры составляет два байта – в шестнадцатеричном формате с порядком байтов от младшего к старшему, 0x6B6E) и увидеть такие данные, как имя раздела и время *LastWrite*.

Например, предположим, что у вас есть образ со свободными кластерами или дамп оперативной памяти, и вы нашли там ячейку раздела. Принимая во внимание переменную *\$offset*, обозначающую смещение в файле по отношению к ячейке раздела, которую вы нашли, код на языке Perl для анализа этой ячейки раздела и извлечения из нее данных будет выглядеть так:

```
seek(FH,$offset,0);
read(FH,$data,76);
my %nk;
my (@recs)      = unpack("vvV17vv", $record);
$nk{id}          = $recs[0];
$nk{type}        = $recs[1];
$nk{time1}       = $recs[2];
$nk{time2}       = $recs[3];
$nk{time3}       = $recs[4];
$nk{no_subkeys} = $recs[6];
$nk{ofs_lf}      = $recs[8];
$nk{no_values}  = $recs[10];
$nk{ofs_vallist} = $recs[11];
$nk{ofs_sk}      = $recs[12];
$nk{ofs_classname}= $recs[13];
$nk{len_name}    = $recs[19];
$nk{len_classname}= $recs[20];
# Get the name
seek(FH,$offset + 76,0);
read(FH,$data,$nk{len_name});
```

В этом коде *FH* – это Perl-дескриптор для файла, содержащего данные, которые мы исследуем, а *\$data* – это скалярная переменная, в которой должно быть содержимое того, что мы только что прочитали из файла. Этот код предоставляет простой способ для анализа важной информации из ячейки раздела: значения идентификатора ячейки раздела (т. е. *\$nk{id}*), как показано на илл. 4.3. На илл. 4.3 в первой половине строки сразу под идентификатором *nk* можно также увидеть шестнадцатеричное представление времени *LastWrite* этого раздела. Вышеуказанный код считывает двоичные данные и разбивает их на различные компоненты, основываясь на структуре данных для ячейки раздела.

Мы можем увидеть значения, которые помогут нам получить время *LastWrite*, а также номера подразделов и параметров, связанных с ячейкой раздела. Мы также можем получить смещения по отношению к дополнительным структурам данных, таким как ячейки списков других разделов (т. е. подразделов) или списков параметров (т. е. параметров для этого раздела).

Совет

Значение `$offset` очень важно при работе с разделами реестра, особенно с теми, что находятся внутри файлов кустов реестра. Четырехбайтовое значение (двойное слово), предшествующее сигнатуре раздела, которая находится в `$offset`, – это размер раздела. Когда это значение интерпретируется как длинное целое без знака, оно во многих случаях будет отрицательным числом. Было обнаружено, что отрицательное значение размера означает, что сам раздел используется, а положительное значение размера означает, что раздел удален. Позднее в этой главе мы подробнее рассмотрим эту тему.

Длина ячейки параметра составляет всего 18 байт с именем переменной длины и данными переменной длины. Этот тип данных хранится в 4-байтовом значении (двойное слово), которое расположено в самой ячейке параметра. Используя тот же способ, что и для ячейки раздела, мы можем проанализировать информацию о ячейке параметра из любого имеющегося источника и увидеть фактические значения. Perl-код для анализа ячейки параметра выглядит следующим образом:

```
seek(FH,$offset,0);
my $bytes      = read(FH,$data,20);
my (@recs)    = unpack("vvVVVVvv",$data);
$vk{id}       = $recs[0];
$vk{len_name} = $recs[1];
$vk{len_data} = $recs[2];
$vk{ofs_data} = $recs[3];
$vk{val_type} = $recs[4];
$vk{flag}     = $recs[5];
if ($vk{len_name} == 0) {
$vk{valname} = "Default";
}
else {
seek(FH,$offset + 20,0);
read(FH,$data,$vk{len_name});
$vk{valname} = $data;
}
```

Как и в случае с ячейкой раздела, идентификатор ячейки параметра (т. е. `$vk{id}`) был показан на илл. 4.3.

Хотя мы можем проанализировать данные параметров реестра из различных источников, информации может быть недостаточно, чтобы узнать, какому разделу изначально принадлежал этот параметр, когда он был создан или как он использовался. Запомните, что в разделах реестра есть связанная с ними отметка времени (т. е. время *LastWrite*), а в параметрах реестра – нет. Извлечение информации о разделах и параметрах из дампов ОЗУ или свободного пространства может иметь ограниченную ценность в зависимости от потребностей исследования. Некоторое время тому назад (на основе структур, предоставленных Питером) я написал Perl-скрипт, который назвал Offline Registry Parser, или сокращенно «regp.pl». В него включен Perl-код, который вы только что видели. Этот скрипт позволяет проанализировать необработанные данные из файла реестра, извлечь информацию о разделах и параметрах и вывести ее на консоль в формате ASCII. Выходные данные можно сохранить в файле, чтобы позднее выполнить по ним поиск или сравнить с другими файлами. Использование такого скрипта дает несколько преимуществ. Во-первых, он является примером с открытым исходным кодом для анализа необработанных данных из файлов реестра; любой, кто разбирается в программировании на языке Perl, может легко расширить возможности скрипта, например, добавить запись результатов в таблицу или базу данных, чтобы упростить их обработку. Во-вторых, скрипт зависит только от функциональных возможностей языка Perl: он не использует различные модули для отдельных платформ и не зависит от интерфейса прикладного

программирования (API) Microsoft. Это означает, что скрипт можно запускать в любой системе, поддерживающей Perl, в том числе в Linux. Если эксперт использует такие инструменты, как The Sleuth Kit (www.sleuthkit.org) или PyFlag (www.pyflag.net/cgi-bin/moin.cgi), на компьютере Linux, который служит платформой для анализа данных, то он также может анализировать и просматривать содержимое файлов реестра. Хотя этот скрипт не добавляет новые функции в любой из доступных судебных инструментов, он предоставляет дополнительные возможности для сокращения объема и более быстрой и эффективной обработки данных. Когда я только собирался сделать этот скрипт более функциональным и гибким, Джеймс МакФарлэн (James MacFarlane) выпустил свой Perl-модуль Parse::Win32Registry, который предоставляет объектно-ориентированный доступ к файлам кустов реестра, в том числе в ОС Windows 98 и ME. Как и файл «regp.pl», код модуля Джеймса частично основан на описании различных компонентов реестра, предоставленном Питером.

Совет

Утилиты PlainSight (www.plainsight.info/) и SANS Investigative Forensic Toolkit (SIFT) Workstation (<http://forensics.sans.org/community/downloads/>) основаны на ОС Linux и включают в себя RegRipper (www.regripper.net), программу, написанную на Perl, для обработки файлов кустов реестра во время анализа образа данных. RegRipper основана на модуле Parse::Win32Registry от Джеймса МакФарлэна и легко устанавливается в ОС Windows, Linux и Mac OS X.

Скрипт «reg.pl», написанный на Perl, а также автономный исполняемый файл для Windows, скомпилированный из скрипта с помощью программы Perl2Exe, доступны на носителе, который идет в комплекте с этой книгой.

Еще один интересный файл, предоставленный Питером в исходном коде для своей утилиты, – это «sam.h». Этот файл содержит очень ценную информацию о структурах в разделе SAM системного реестра. Мы рассмотрим эти структуры и способы их использования позднее в этой главе в разделе «Поиск сведений о пользователях».

Системный реестр как файл журнала

Ячейки разделов в реестре состоят из разделов или папок, которые вы видите, когда открываете редактор реестра. Это единственная структура, содержащая значение времени, которое называется время *LastWrite* (рус. время последней записи). Время *LastWrite* – это 64-разрядное значение структуры *FILETIME*, похожее на время последнего изменения в файле (дополнительные сведения см. во вставке «Время LastWrite раздела реестра»). Эти данные не только указывают на период времени, в течение которого были совершены определенные действия пользователя в системе, но и в некоторых случаях показывают, когда отдельный параметр был добавлен в раздел или изменен.

Записки из подполья...

Время *LastWrite* раздела реестра

С разделами реестра связано несколько свойств, одно из которых – это время *LastWrite*, похожее на время последнего изменения в файлах и каталогах. Время *LastWrite* хранится в структуре *FILETIME*, которая содержит значение, представляющее количество 100-наносекундных интервалов, прошедших с 1 января 1601 года (на основе григорианского календаря). Для того чтобы преобразовать 64-разрядное значение в понятный формат, необходимо перевести его в структуру *SYSTEMTIME* (<http://msdn.microsoft.com/en-us/library/ms724280.aspx>). Перевод структуры *FILETIME* непосредственно в структуру *SYSTEMTIME*, используя интерфейс API *FileTimeToSystemTime()*, позволит отобразить дату в формате времени UTC, которое

иногда называют средним временем по Гринвичу (GMT). Чтобы перевести эти значения в формат, который точно представляет текущее системное время, необходимо сначала пропустить структуру *FILETIME* через интерфейс API *FileTimeToLocalFileTime()*. API учитывает информацию локальной системы о летнем времени и часовом поясе. Структуру *FILETIME* можно интерпретировать скриптом Perl как два 32-разрядных числа, а затем перевести в 32-разрядное значение времени UNIX (http://en.wikipedia.org/wiki/Unix_time) и отобразить, используя встроенные функции Perl.

Эта информация может оказаться особенно полезной во время расследования вторжений. Я люблю открыть приложение Registry Viewer в программе ProDiscover и найти область, в которой хранятся разделы реестра, относящиеся к службам ОС Windows. Здесь я могу отсортировать первый уровень разделов по их времени *LastWrite* и легко найти установленные во время вторжения объекты, например, программы удаленного администрирования или даже руткиты.

Совет

Мы подробнее поговорим о руткитах в главе 7. Однако червь, определенный компанией Network Associates как W32/Opanki.worm!MS06-040 (http://vil.nai.com/vil/Content/v_140546.htm), можно рассмотреть здесь в качестве полезного примера. Червь устанавливает компонент руткита, создавая, помимо ряда других артефактов, две службы Windows. Так как большая часть вторжений происходит через некоторое время после установки операционной системы, сортировка разделов служб (Services) по их времени *LastWrite*, как правило, поможет вам быстро закончить расследование. Этот способ очень полезен для обнаружения разнообразных вредоносных программ, создающих службы Windows после проникновения в систему.

Помимо времени *LastWrite*, реестр содержит информацию об отметках времени в данных, связанных с отдельными параметрами. Мы поговорим о некоторых из этих параметрах позднее, а пока достаточно знать, что есть разделы реестра, значения которых содержат 8 байт двоичных данных, составляющих структуру *FILETIME*.

Предупреждение

Одно из свойств, характеризующее реестр ОС Windows, – отсутствие последовательности. Я знаю, что вы, вероятно, сейчас читаете это и думаете, что я не выспался или выпил слишком много кофе. Но, потратив немного времени на изучение реестра, вы поймете, что я имею в виду. Некоторые параметры в реестре содержат двоичные данные, которые после анализа или преобразования можно интерпретировать как 64-разрядное значение структуры *FILETIME*. Такое, например, возможно в некоторых параметрах раздела UserAssist и в параметре *ShutdownTime*. Однако в остальных случаях отметка времени хранится как 32-разрядное значение UNIX.

Итак, почему я сравниваю реестр с файлом журнала? Потому что файлы журналов обычно содержат записи о действиях или событиях, связанном с временем, и во многих случаях то же можно сказать о реестре. Хотя в реестре может храниться буквально тысячи записей, не все из них представляют интерес для эксперта. Во время различных экспертиз вас будут интересовать отдельные разделы, и если вы знаете, как система создает, изменяет и использует эти разделы и параметры, то поймете, как в реестре записывается информация о различных событиях и связанных с ними отметках времени.

Наблюдение за изменениями в реестре

Не существует отдельного единого ресурса о разделах реестра, которые могут быть полезными в какой-нибудь определенной ситуации. Электронная таблица со многими из

разделов, содержащими, по моему мнению и мнению моих коллег, важную информацию для эксперта, включена в каталог «ch4» на диске, который идет в комплекте с этой книгой. Однако она не является самым исчерпывающим источником, потому что такого просто не существует. В некоторых случаях разделы реестра, которые создаются при установке программы и используются во время ее работы, могут изменяться в зависимости от версии этой программы. Вскоре после того, как эта книга пойдет в печать, будет опубликована и поступит в продажу, несомненно, появится ранее неизвестное приложение, записывающее сведения о конфигурации и параметрах в системный реестр.

Итак, как определить важные разделы и параметры реестра? Один из способов – сделать снимок реестра, выполнить простое действие, снова сделать снимок реестра и сравнить эти два снимка на предмет различий. Один из полезных инструментов для этой задачи – это InControl5, который журнал PC Magazine (распространитель этого приложения) называет как Inctrl5.

Предупреждение

Программа Inctrl5 доступна по адресу www.pc当地.com/article2/0,4149,9882,00.asp, однако она не бесплатна. Разработчик запрашивает небольшую сумму за загрузку программы, а в лицензионном соглашении однозначно запрещено повторное распространение этого ПО.

InControl5 – отличная утилита для выполнения различных видов анализа. Я обычно использую InControl5 следующим образом: открываю программу и после того, как она сделает снимок системы, я делаю все, что собирался (например, устанавливаю приложения одноранговой сети для обмена файлами или какое-нибудь вредоносное ПО), а затем снова открываю InControl5 и жду, пока она закончит свою работу. Программа создает отчет в формате HTML, где я могу точно увидеть, какие разделы реестра (и файлы) были добавлены, изменены или удалены во время установки.

Еще один способ обнаружить полезные или важные разделы – использовать утилиту RegMon от корпорации Microsoft, чтобы наблюдать за реестром в реальном масштабе времени. RegMon доступна по адресу <http://technet.microsoft.com/en-us/sysinternals/bb896652.aspx>, но она также включена в состав программы Process Explorer от Microsoft. Когда я пытался узнать, где в реестре хранится информация о пользователях, я обнаружил, что, запустив RegMon и выполнив затем команду net user, я смог определить, к каким разделам и параметрам происходит обращение во время поиска данных о пользователях, см. илл. 4.4.

lsass.exe:1052	QueryValue	HKLM\SAM\SAM\C
lsass.exe:1052	QueryValue	HKLM\SAM\SAM\Domains\Account\
lsass.exe:1052	OpenKey	HKLM\SAM\SAM\DOMAINS\Account\Users\Names
lsass.exe:1052	Enumerate...	HKLM\SAM\SAM\DOMAINS\Account\Users\Names
lsass.exe:1052	OpenKey	HKLM\SAM\DOMAINS\Account\Users\Names\Administrator
lsass.exe:1052	QueryValue	HKLM\SAM\DOMAINS\Account\Users\Names\Administrator\{Default}
lsass.exe:1052	CloseKey	HKLM\SAM\DOMAINS\Account\Users\Names\Administrator
lsass.exe:1052	Enumerate...	HKLM\SAM\DOMAINS\Account\Users\Names
lsass.exe:1052	OpenKey	HKLM\SAM\DOMAINS\Account\Users\Names\Guest
lsass.exe:1052	QueryValue	HKLM\SAM\DOMAINS\Account\Users\Names\Guest\{Default}
lsass.exe:1052	CloseKey	HKLM\SAM\DOMAINS\Account\Users\Names\Guest
lsass.exe:1052	Enumerate...	HKLM\SAM\DOMAINS\Account\Users\Names

Илл. 4.4. Фрагмент выходных данных программы RegMon.

На илл. 4.4 показан фрагмент выходных данных программы RegMon, которая использовалась для наблюдения за командой net user. Я начал сбор данных, запустил команду, а затем остановил сбор и выполнил в полученных данных поиск куста SAM. Таким образом, несмотря на то, что я использовал команду «net.exe», чтобы запросить информацию, она передала запрос процессу «lsass.exe», который обратился к реестру, чтобы получить информацию об именах пользователей в системе.

Тест-драйв...

Наблюдение за реестром

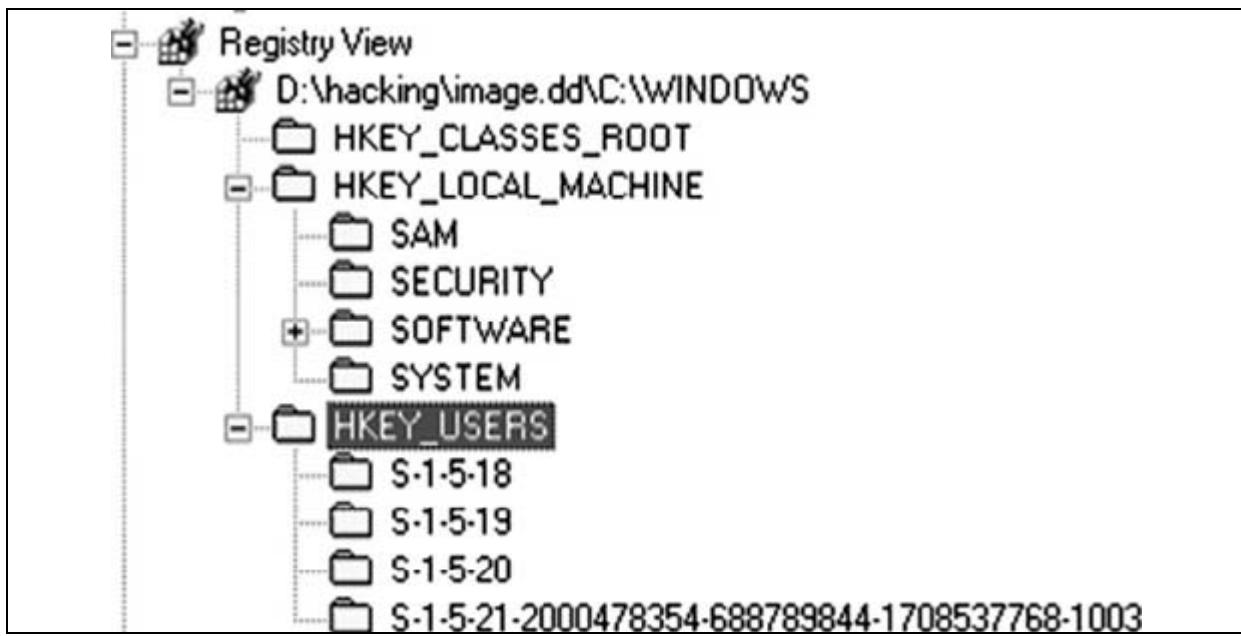
Попробуйте выполнить следующее упражнение. Загрузите на свой компьютер копию программы RegMon, откройте ее и остановите сбор данных (щелкните по значку лупы, чтобы сверху него появился красный символ X). Затем откройте командную строку и введите **net accounts**, но не нажмайте клавишу Enter. Вернитесь в окно программы RegMon и запустите сбор данных, затем снова перейдите в командную строку и незамедлительно нажмите Enter. Как только в командной строке начнет появляться информация о политиках учетных записей, вернитесь в RegMon и остановите сбор данных. Для того чтобы облегчить анализ, отфильтруйте процесс «lsass.exe» и посмотрите, к каким разделам реестра было выполнено обращение, чтобы получить информацию, которая отображается в командной строке. Теперь при исследовании образа данных вы будете знать, в каких разделах реестра нужно искать информацию, относящуюся к политикам учетных записей.

В оставшейся части главы будет показано несколько примеров применения программ InControl5 и RegMon. Рекомендуется иметь эти инструменты в наличии (в условиях лицензионного соглашения для этих программ запрещено распространять их на носителе, который идет в комплекте с этой книгой), особенно если вы хотите опробовать различные методы работы, упоминаемые в данной главе.

Анализ системного реестра

Теперь, когда вы знакомы со структурой реестра, давайте рассмотрим способы получения и анализа информации системного реестра. Большая часть этих данных будет доступна вам во время анализа работающего компьютера (за исключением тех разделов, к которым вы не можете получить доступ из-за отсутствия необходимых разрешений), и все они (за исключением энергозависимой части реестра) будут доступны во время анализа образа данных.

Программа ProDiscover предоставляет простой интерфейс (фактически, API) для доступа к реестру во время анализа образа данных. После того как дело будет загружено в ProDiscover, вам нужно только щелкнуть правой кнопкой мыши по каталогу «Windows» в разделе «Просмотр содержимого» (“Content View”) и выбрать пункт «Добавить в Registry Viewer» (“Add to Registry Viewer”). Затем ProDiscover найдет необходимые файлы и отправит данные в Registry Viewer, как показано на илл. 4.5.

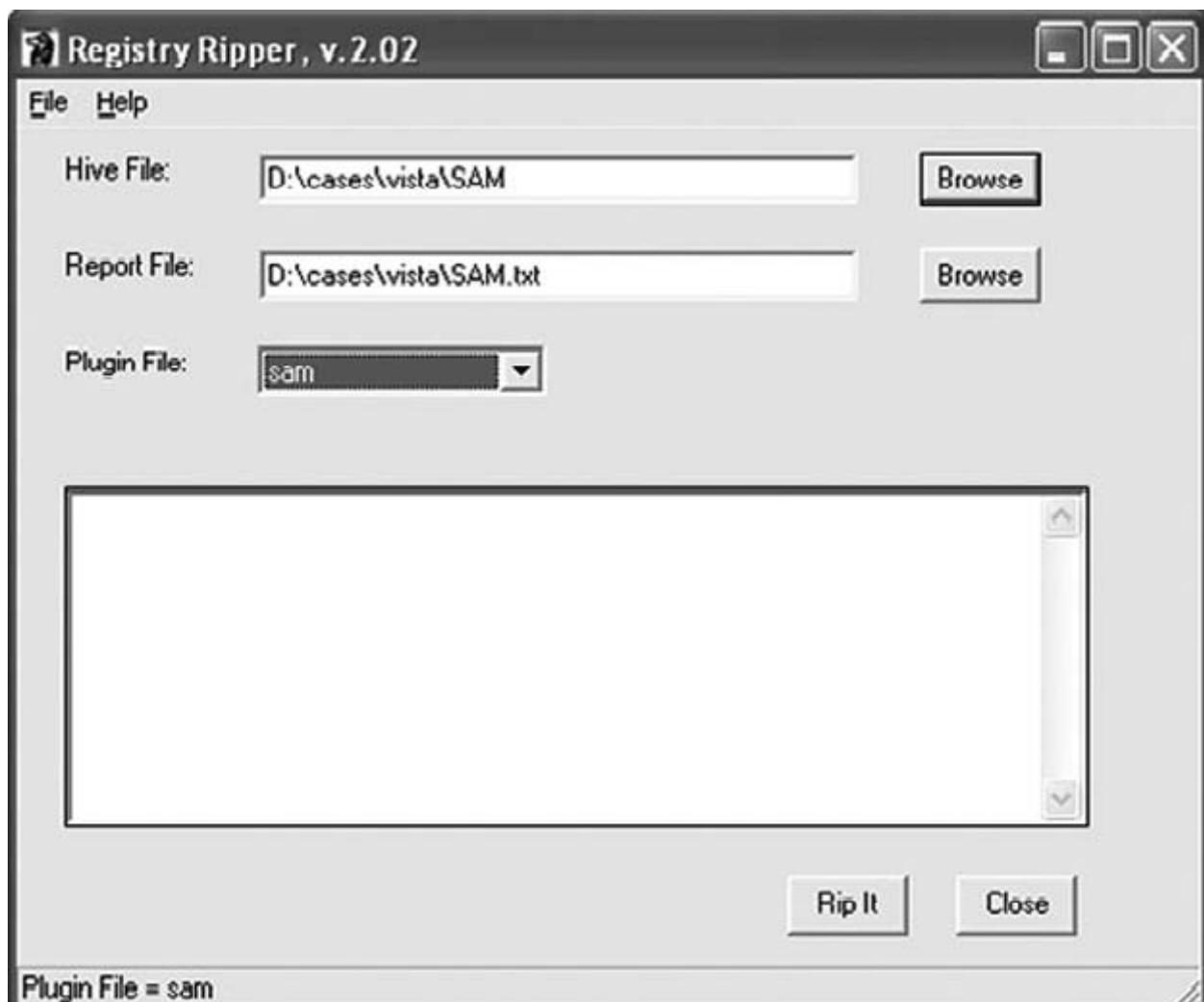


Илл. 4.5. Окно приложения Registry Viewer в программе ProDiscover IR.

Скриптовый язык ProScript в программе ProDiscover основан на языке Perl и предоставляет отличные возможности для извлечения информации из системного реестра в автоматическом режиме. Модуль «ProScript.pm» упрощает интерфейс для создания Perl-скриптов в ProDiscover; практически все, что вы можете сделать посредством интерфейса программы ProDiscover, можно также выполнить с помощью скриптов ProScript. Что касается реестра Windows и судебного анализа образа данных, вы можете использовать скрипты ProScript, чтобы полностью автоматизировать множество трудоемких операций по сбору и форматированию данных. В этой главе я опишу скрипты ProScript, необходимые для выполнения таких задач, как извлечение и анализ информации из реестра во время исследования образа данных. Все эти скрипты можно найти в каталоге ch4\code\Proscripts на носителе, который идет в комплекте с этой книгой.

RegRipper

В первое издание этой книги я включил ряд Perl-скриптов, которые можно было использовать для извлечения отдельных данных из файлов кустов реестра (все эти скрипты сохранены в каталоге ch4\code\old на носителе, который идет в комплекте с этой книгой). Занимаясь расследованием инцидентов и судебным анализом, я обнаружил, что продолжаю искать те же разделы реестра, но в то же время добавляю дополнительные разделы в свой быстро увеличивающийся список артефактов системного реестра. Я понял, что мне нужно средство для записи большого количества информации; например, одно дело – извлечь определенный параметр из раздела реестра, но добавление справочных сведений (к примеру, статей из базы знаний Microsoft и т. д.) обеспечило бы большую полноту этих данным, особенно при включении их в отчет. Затем я подумал, что мог бы разработать платформу, которая облегчила бы добавление всех этих аспектов при обнаружении разделов и параметров реестра, относящихся к экспертизе. Кроме того, я мог бы предоставить эту платформу другим пользователям. Платформа должна была быть мощной, но достаточно простой, чтобы любой пользователь мог легко научиться обновлять и расширять ее. Инструменты на основе подключаемых модулей, такие как Nessus (www.nessus.org), казалось, хорошо для этого подходили, поэтому я создал графический интерфейс пользователя и начал разрабатывать средство, которое назвал RegRipper. На илл. 4.6 показан графический интерфейс для RegRipper версии 2.02.



Илл. 4.6. Графический интерфейс программы RegRipper версии 2.02.

Полная, обновленная копия RegRipper 2.02 доступна в каталоге ch4\RegRipper на носителе, который идет в комплекте с этой книгой.

RegRipper написана на языке Perl, поэтому вы можете получить доступ к исходному коду, чтобы узнать, что делает эта программа. Графический интерфейс пользователя основан на модуле Win32::GUI (что означает, что интерфейс будет работать только в ОС Windows), а RegRipper использует модуль Parse::Win32Registry Джеймса Макфарлейна (James MacFarlane), чтобы выполнять большинство трудных задач, связанных с доступом к файлам кустов системного реестра. Я также предоставляю RegRipper и другие связанные с ней инструменты в виде исполняемых файлов, скомпилированных с помощью Perl2Exe, чтобы их можно было запускать на компьютере с ОС Windows без необходимости устанавливать интерпретатор языка Perl. Утилита Perl2Exe доступна на странице www.indigostar.com/perl2exe.htm; для тех же целей можно использовать другие инструменты, такие как PerlApp от компании ActiveState или PAR (Perl Archiver).

Для того чтобы RegRipper (rr.exe) работала на других платформах, например, в ОС Linux, ее необходимо запускать под управлением программного обеспечения Wine (www.winehq.org/) с небольшими изменениями в коде Perl, чтобы решить проблему с разделителями путей. Версии инструментов RegRipper на основе командной строки можно запускать в Linux или Mac OS X. В следующих версиях этих инструментов будет уделено большее внимание совместимости между разными платформами.

Основополагающая идея программы RegRipper довольно проста. Я начал думать о том, что мне не хочется вести список всех возможных разделов реестра, которые мне нужно было бы исследовать при множестве различных обстоятельств. Я составлял списки

разделов и параметров, записывал справочную информацию о том, как эти разделы изменяются или почему они важны в тех или иных условиях, и даже делал заметки о том, как сопоставлять информацию из различных разделов реестра, чтобы создать более полную картину действий пользователей в системе. Я начал писать различные коды для выполнения отдельных задач, и большая часть этой главы в первом издании книги состояла из этих кодов. Но я также понял, что необходим больший уровень гибкости и практичности. У меня появилась идея создать платформу, которая, так же как Nessus, использовала бы подключаемые текстовые модули для выполнения всех трудных операций по извлечению данных реестра и даже для проведения отдельных видов анализа. Так как модули основаны на тексте, любой может воспользоваться текстовым редактором, чтобы прочитать или модифицировать их; все, что для этого требуется, – немного знаний о программировании на Perl. Я обнаружил, что могу легко добавлять такие сведения, как ссылки на статьи из базы знаний Microsoft, короткие заметки по анализу, в которых объясняется, как интерпретировать извлеченную информацию, и другие похожие данные. Я надеялся, что, имея достаточное количество разных подключаемых модулей, используемых в качестве образцов кода, эксперты рано или поздно начнут расширять функциональность этих модулей или даже создавать свои собственные модули. Именно так и произошло.

Необходимо понимать несколько принципов, лежащих в основе RegRipper (это также относится к другим инструментам из семейства RegRipper, которые мы рассмотрим в скором времени), чтобы правильно и эффективно запускать эту программу. Во-первых, к файлам кустов реестра можно получить доступ несколькими способами: посредством накопителя удаленной системы, монтированного с помощью F-Response, посредством извлечения файла куста из созданного образа данных (используя FTK Imager или похожие программы) или посредством монтирования образа данных в режиме только для чтения с помощью таких приложений, как Smart Mount или Mount Image Pro. Некоторые эксперты даже добавляют инструменты RegRipper (как внешнее средство просмотра) в программу EnCase от Guidance Software.

Предупреждение

Не следует запускать RegRipper для анализа файлов кустов реестра на работающем компьютере. RegRipper предназначена для других целей. Кроме того, некоторые пользователи запускали RegRipper для анализа файлов кустов из каталога «systemprofile» или «Default Users» и не находили полезной информации; во многих случаях эти файлы являются заполнителями и содержат нули.

Как указывалось выше, при необходимости можно запустить RegRipper, чтобы исследовать реестр работающего компьютера, но для этого нужно монтировать НЖМД удаленного компьютера с помощью F-Response в режиме только для чтения на вашем компьютере, используемом для анализа данных. В главе 3 упоминается PDF-документ, в котором подробно описаны этапы удаленной установки и удаления F-Response Enterprise Edition по сети. Документ находится на носителе, который идет в комплекте с этой книгой.

RegRipper выступает в качестве платформы или средства для запуска подключаемых модулей с целью обработки файлов кустов реестра. Подключаемые модули – это текстовые Perl-скрипты, содержащиеся в подкаталоге «plugins» в папке, где хранится RegRipper. Модули имеют расширение .pl и содержат все фактические кодированные команды для получения доступа к данным из файлов кустов реестра, а также их извлечения и отображения.

RegRipper может также работать со списками из файлов модулей. Файл модулей – это текстовый файл, содержащий списки подключаемых модулей. Несмотря на то, что модули имеют расширение .pl, в файле модулей расширения нет. В файле модулей

каждый подключаемый модуль, который хочет запустить эксперт, перечислен в том порядке, в котором он должен быть запущен, без расширения. Например, модуль «appinitdlls.pl» был создан для обработки файла куста Software, чтобы извлечь данные из параметра реестра *AppInit_DLLs*. В дистрибутиве RegRipper в каталоге «plugins» вы также найдете файл с именем «software». Этот файл содержит список подключаемых модулей, которые можно применить к файлу куста Software, чтобы извлечь разнообразную информацию. В файле «software» вы увидите, что модуль «appinitdlls» указан без расширения. Если вы выберите этот файл через интерфейс RegRipper или из командной строки, программа применит все перечисленные модули к указанному файлу куста.

Итак, вместо того чтобы запускать один модуль для анализа одного файла куста, можно составить собственный список модулей в отдельном файле, расположить эти модули в нужном порядке и даже создать свой собственный модуль для исследования отдельных файлов кустов. Дистрибутив RegRipper распространяется с несколькими стандартными файлами модулей, основанными на именах файлов кустов, для которых они предназначены; сами подключаемые модули содержат похожую информацию. Однако ни RegRipper, ни подключаемые модули не проверяют, запущены ли они для обработки правильного типа файла куста; если эксперт применит модуль, предназначенный для анализа файла куста System, к файлу куста SAM, то он не должен удивляться, когда не найдет полезной информации.

RegRipper распространяется как в виде скрипта «rr.pl», так и в виде файла «rr.exe», который представляет собой Perl-скрипт, скомпилированный с помощью утилиты Perl2Exe, чтобы вам не нужно было устанавливать интерпретатор Perl для запуска скрипта. Для того чтобы запустить RegRipper, просто дважды щелкните по файлу «rr.exe». Откроется графический интерфейс, показанный на илл. 4.6, и отсюда вы сможете перейти к файлу куста, который нужно обработать, указать место для сохранения файла отчета, а затем выбрать файл модулей, который нужно применить к файлу куста, из раскрывающегося списка (список заполняется автоматически при запуске RegRipper). Нажав кнопку «Получить данные» (“Rip It”), вы запустите подключаемые модули, перечисленные в файле модулей, для анализа выбранного файла куста.

Еще одна интересная особенность программы RegRipper состоит в том, что она автоматически создает журнал своих действий, записывая отдельные события, связанные с работой модулей. Когда вы выбираете место для сохранения файла отчета, RegRipper автоматически создает свой файл журнала, используя то же имя (но с другим расширением), в той же папке, где сохраняется отчет. Файл журнала будет как минимум содержать путь к исследуемому файлу куста, имя и версию каждого модуля (в порядке запуска), а также время запуска каждого модуля. RegRipper работает очень быстро, поэтому отметки времени в журнале будут иметь очень близкие значения; тем не менее, этот файл журнала предоставляет исчерпывающую документацию, которую можно добавить в примечания к делу.

Дистрибутив RegRipper доступен на носителе, который поставляется в комплекте с этой книгой, а также на веб-сайте RegRipper.net, созданном и поддерживаемом Бретом Шэйверсом (Brett Shavers). На форуме и странице загрузки сайта RegRipper.net можно оставлять запросы на модули, тестировать модули, созданные другими пользователями, или распространять новые модули; некоторые размещают свои модули на сайте, другие присыпают свои работы непосредственно мне по электронной почте. Джейсон Коппе (Jason Koppe) даже написал генератор модулей (<http://nssadoc.blogspot.com/2008/10/regripper-regview-and-bluetooth.html>), чтобы облегчить создание простых подключаемых модулей для RegRipper.

Rip

Работая с RegRipper, я обнаружил, что в некоторых случаях мне нужно просто выполнить быструю проверку, скажем, применить один модуль к файлу куста. Например,

одна из первых операций, которую я выполняю при анализе журналов событий ОС Windows, – это извлечение политики аудита из файла куста Security. Возможно, для этого мне не нужно запускать графический интерфейс пользователя, мне нужно только просмотреть эту информацию в консоли, чтобы я мог провести анализ, а затем вставить полученные данные в свои примечания к делу. Поэтому я написал версию RegRipper на основе командной строки и назвал ее «rip» или «rip.pl». Как и RegRipper, утилита «rip.pl» также предлагается в виде исполняемого файла, чтобы вам не нужно было устанавливать на компьютере интерпретатор Perl для ее использования. Rip основана на той же платформе, что RegRipper, и использует те же подключаемые модули.

Синтаксис для файлов «rip.pl» и «rip.exe» имеет следующий вид:

```
Rip v.20080419 - CLI RegRipper tool
Rip [-r Reg hive file] [-f plugin file] [-p plugin module] [-l] [-h]
Parse Windows Registry files, using either a single module, or a plugins
file. All plugins must be located in the "plugins" directory; default plugins
file used if no other filename given is "plugins\plugins".
    -r Reg hive file....Registry hive file to parse
    -g .....Guess the hive file (experimental)
    -f [plugin file]....use the plugin file (default: plugins\plugins)
    -p plugin module....use only this module
    -l .....list all plugins
    -c .....Output list in CSV format (use with -l)
    -h .....Help (print this information)
Ex: C:\>rr -r c:\case\system -f system
    C:\>rr -r c:\case\NTUSER.DAT -p userassist
    C:\>rr -l -c
All output goes to STDOUT; use redirection (ie, > or >>) to output to a file.
```

Как видите, «rip.pl» имеет множество полезных функций. Я добавил возможность выводить список доступных модулей и перечислять их в формате значений, разделенных запятыми (.csv), чтобы я мог открыть этот список в Excel и сортировать его на основе таких параметров, как дата модуля или файл куста, для которого предназначен модуль. Эта возможность доступна при помощи переключателей *-l* и *-c* в утилите «rip.pl». Переключатель *-l* указывает «rip.pl» проанализировать доступные подключаемые модули (которые должны быть расположены в каталоге «plugins») и вывести их список. Переключатель *-l* перечисляет различные модули в алфавитном порядке, нумеруя каждый пункт списка, как показано в следующем фрагменте выходных данных:

9. auditpol v.20080327 [Security]
 - Get audit policy from the Security hive file
10. autoendtasks v.20081128 [NTUSER.DAT]
 - Automatically end a non-responsive task
11. autorun v.20081212 [NTUSER.DAT]
 - Gets autorun settings
12. banner v.20081119 [Software]
 - Get HKLM\SOFTWARE.. Logon Banner Values
13. bho v.20080418 [Software]
 - Gets Browser Helper Objects from Software hive
14. bitbucket v.20080418 [Software]
 - Get HKLM\..\BitBucket keys\values

Как видите, каждый модуль перечислен в алфавитном порядке с указанием версии (которая представляет собой дату в формате ГГГГММДД), куста, для анализа которого предназначен этот модуль, и содержимого поля с кратким описанием в самом модуле. Добавление переключателя *-c* выводит тот же список только без порядковых номеров и в формате значений, разделенных запятыми (.csv). Модули можно отфильтровать в командной строке, используя команду *find* (например, *C:\Perl>rip.pl -l -c / find*

“Software”), чтобы найти отдельные модули, или можно перенаправить выходные данные в файл (например, *C:\Perl>rip.pl -l -c > plugins.csv*), а затем открыть и отсортировать список в программе Excel. Следует отметить, что модуль «banner.pl» был создан одной сотрудницей правоохранительных органов для своих целей; она любезно согласилась предоставить мне копию этого модуля для распространения вместе с RegRipper.

Еще один полезный, хотя экспериментальный, переключатель (-g) предоставляет возможность приблизительно определять тип файла куста (System, SAM, Software, NTUSER.DAT, Security), для которого применяется запущенный модуль. Переключатель доступен из командной строки, но я собираюсь включить его в код утилиты, чтобы он работал незаметно для пользователя; эта функциональная возможность, вероятно, будет добавлена в будущие версии RegRipper.

Утилита «rip.pl» обладает очень мощными и гибкими функциями. Самый простой способ использования «rip.pl» – применить отдельный подключаемый модуль к файлу куста:

```
C:\Perl>rip.pl -r d:\case1\Security -p auditpol
Plugins Dir = C:\Perl\forensics\rr\plugins/
Launching auditpol v.20080327
auditpol
Policy\PolAdtEv
LastWrite Time Fri Sep 9 01:11:43 2005 (UTC)
Auditing is enabled.
      Audit System Events      = S/F
      Audit Logon Events       = S/F
      Audit Object Access      = N
      Audit Privilege Use      = N
      Audit Process Tracking   = N
      Audit Policy Change      = S/F
      Audit Account Management = S/F
      Audit Dir Service Access = N
      Audit Account Logon Events = S/F
```

В этом примере я запустил модуль «auditpol» для анализа файла куста Security, что позволило мне (как видите) просмотреть политику аудита для данной системы. Этот модуль показывает некоторую полезную для эксперта информацию о предпринятых действиях и проанализированном разделе (например, время *LastWrite* радела), а затем выводит саму политику аудита. В статье № 246120 из базы знаний Microsoft (<http://support.microsoft.com/kb/246120>) описано, как перевести двоичные данные параметра реестра в информацию, которую эксперт сможет легко прочитать и понять; модуль «auditpol.pl» сделает такое преобразование автоматически.

Версии операционной системы Windows, выпущенные после XP, имеют свои отличия, но в том, что касается файлов кустов реестра, ни структура файлов, ни их фактическое содержимое не претерпели значительных изменений. В качестве примера я запустил модуль «winver.pl», чтобы проанализировать файлы куста Software из систем Windows Vista и Windows 7 (бета-версия на момент написания этой книги) и увидел следующие результаты (выделения сделаны автором):

```
C:\Perl\forensics\rr>rip.pl -p winver -r d:\cases\vista\software
Plugins Dir = C:\Perl\forensics\rr\plugins/
Launching winver v.20081210
ProductName    = Windows Vista (TM) Home Premium
CSDVersion     = Service Pack 1
InstallDate    = Fri Aug 31 15:21:10 2007

C:\Perl\forensics\rr>rip.pl -p winver -r d:\cases\win7\software
Plugins Dir    = C:\Perl\forensics\rr\plugins/
Launching winver v.20081210
```

```
ProductName      = Windows 7 Ultimate
InstallDate     = Fri Dec 12 20:52:50 2008
```

Что касается фактического содержимого файлов кустов реестра в различных версиях операционной системы Windows, то существуют различия в том, какие данные сохраняются в реестре и как они хранятся; мы рассмотрим эти различия в данной главе.

Используя переключатель `-f`, можно запустить файл модулей, а не каждый модуль по отдельности, для обработки файла куста. Или, так как «`rip.pl`» и «`rip.exe`» – это инструменты на основе командной строки, можно создать пакетные файлы, запускающие только отдельные модули для анализа файлов кустов. Например, файл «`auditpol.bat`» предоставляется как часть дистрибутива RegRipper и содержит следующее:

```
@echo off
REM Batch file to automate running of auditpol.pl plugin file
REM in order to enumerate the audit policy from the Security hive
REM file.
REM
REM Usage: auditpol <path_to_Security_hive_file>
REM
REM copyright 2008 H. Carvey, keydet89@yahoo.com
rip.exe -r %1 -p auditpol
```

Как видите, это обычный пакетный файл DOS, содержащий комментарии для удобочитаемости (и как средство документации). Фактический исполняемый код находится в последней строке и выделен полужирным шрифтом.

Запуск модуля для обработки файла куста, экспортированного из образа с помощью FTK Imager или ProDiscover, – не единственный способ использования этих инструментов. После монтирования образа данных с помощью программы Smart Mount или ImDisk (информация об этих инструментах и их применении доступна в главе 5) можно использовать «`rip.pl`» (или пакетные файлы при помощи «`rip.pl`» или «`rip.exe`»), как показано ниже (образ монтирован в режиме только для чтения как символ накопителя H:):

```
C:\Perl>auditpol h:\windows\system32\config\security
Launching auditpol v.20080327
auditpol
Policy\PolAdtEv
LastWrite Time Wed Jan 30 05:31:12 2008 (UTC)
Auditing is enabled.
      Audit System Events      = S/F
      Audit Logon Events       = S/F
      Audit Object Access      = N
      Audit Privilege Use      = N
      Audit Process Tracking   = N
      Audit Policy Change      = S/F
      Audit Account Management = S/F
      Audit Dir Service Access = N
      Audit Account Logon      = S/F
      Events
```

Как видите, RegRipper, и в частности «`rip.pl`», – универсальные и гибкие инструменты. Вместо того чтобы открывать файлы кустов реестра в программе просмотра (например, в редакторе реестра), подобные инструменты предоставляют эксперту автоматизированное средство для извлечения, а во многих случаях и для преобразования, отдельных данных из файлов кустов. Таким образом можно выполнить более полное и тщательное исследование намного быстрее, что снижает количество ошибок и позволяет эксперту сосредоточиться на анализе данных.

Как упоминалось во введении к этой главе, мы рассмотрим ряд важных для эксперта разделов и параметров реестра, находящихся в различных кустах. При этом мы познакомимся с несколькими подключаемыми модулями, используемыми инструментами RegRipper: самой программой RegRipper, утилитой «*grip.pl*» на основе командной строки, а также утилитой «*gripxp.pl*». (Утилита «*gripxp.pl*» будет описана в следующем разделе, а ее работа будет продемонстрирована позднее в этой главе). Однако мы не будем рассматривать все доступные модули, а также не сможем обсудить все имеющиеся разделы реестра.

RipXP

Семейство инструментов RegRipper содержит еще одну полезную утилиту, которая представляет собой модификацию «*grip.pl*» и называется «*gripxp.pl*». Я выбрал это имя, потому что это утилита командной строки, основанная на «*grip.pl*» и предназначенная для ОС Windows XP. ОС Windows XP сохраняет точки восстановления системы (эта тема подробно рассмотрена в главе 5), которые содержат части файлов кустов системного реестра. Для того чтобы запустить «*gripxp.pl*», необходимо указать путь к файлу куста, который нужно проанализировать, путь к каталогу с точками восстановления, а затем выбрать модуль, который вы хотите запустить.

Примечание

«*Gripxp.pl*» не предоставляется на носителе, который идет в комплекте с этой книгой. На момент написания книги утилита была еще слишком недоработанной, чтобы выпускать ее для всеобщего пользования. Я продемонстрировал способы ее применения на саммите SANS Forensic Summit в Лас-Вегасе в октябре 2008 года и предоставил ознакомительную версию утилиты своему другу, который с ее помощью достиг больших результатов в своей работе. Для того чтобы запустить «*gripxp.pl*», следует выполнить несколько своеобразных действий, поэтому нужно не только разработать новый учебный материал, но и проделать дополнительную работу, чтобы сделать утилиту более функциональной и дружелюбной для пользователя.

Утилита «*gripxp.pl*» «угадывает» тип файла куста, а затем определяет, можно ли запустить подключаемый модуль для обработки этого файла. Она применяет модуль к файлу куста, затем переходит к каталогу с точками восстановления, определяет местонахождение каждой точки восстановления, определяет и отображает дату создания каждой точки, а затем запускает модуль для анализа отдельного файла куста. Все это выполняется в автоматическом режиме.

Возможность таких инструментов, как «*gripxp.pl*», извлекать данные за прошедшие периоды времени из точек восстановления Windows XP очень эффективна. Как и в случае со многими инструментами, которые я разработал, потребность в таких функциональных возможностях возникала либо по мере необходимости, либо когда кто-нибудь говорил: «Послушай, а было бы неплохо, если бы можно было...». На самом деле, идея создания «*gripxp.pl*» принадлежит Робу Ли (Rob Lee), так как весной 2008 года он сказал следующее: «Было бы классно, если бы можно было автоматически запускать инструмент типа RegRipper для анализа точек восстановления Windows XP». Я не мог с ним не согласиться. Способы использования «*gripxp.pl*» буду продемонстрированы позднее в этой главе.

Информация о системе

Исследуя образ данных ОС Windows, можно найти много основной информации о системе. Большую часть этой информации относительно легко получить во время исследования работающего компьютера; например, во многих случаях версию операционной системы (например, Windows 2000, XP, 2003 или Vista) можно определить, просто взглянув на пользовательский интерфейс. Или можно щелкнуть правой кнопкой

мыши по значку «**Мой компьютер**» (“My Computer”) и выбрать пункт «**Свойства**» (“Properties”), чтобы просмотреть такую информацию, как версия операционной системы, версия пакета обновления и имя компьютера. Эта информация также доступна в системном реестре, где к ней можно легко получить доступ во время анализа образа данных. Более того, информация о самой системе хранится главным образом в файлах кустов System и Software (некоторая часть – в файле куста Security), а информация о пользователях содержится в файле куста SAM. Информация о пользователях и их действиях хранится в пользовательском файле «NTUSER.DAT». Этот раздел главы в основном посвящен обзору файлов кустов System, Software и Security.

Если помните, *CurrentControlSet* – это энергозависимый раздел реестра, и вы не найдете его в образе клонированных данных. Во вставке «Поиск CurrentControlSet» показаны способы, с помощью которых можно определить, какой раздел *ControlSet* отмечен как «текущий» в работающей системе. Определив, какой *ControlSet* является текущим, можно сосредоточиться на исследовании подразделов данного *ControlSet*.

Искусство судебной экспертизы...

Поиск CurrentControlSet

Часто при работе образом данных вам будет интересно узнать, какой из двух разделов *ControlSet*, отображаемых в приложении Registry Viewer (в ProDiscover), используется операционной системой как *CurrentControlSet*. Для этого перейдите в раздел HKEY_LOCAL_MACHINE\System\Select, где вы найдете несколько параметров, как показано на илл. 4.7.

Дистрибутив RegRipper распространяется вместе с файлом шаблона, который можно использовать, чтобы быстро создать подключаемый модуль; этот шаблон содержит код для определения текущего набора управления (*CurrentControlSet*) в файле куста System. Кроме того, можно использовать любой модуль, который обращается к кусту System, из каталога «plugins» как основу для создания собственных модулей.

		(Default)	REG_SZ	(value not set)
		Current	REG_DWORD	0x00000001 (1)
		Default	REG_DWORD	0x00000001 (1)
		Failed	REG_DWORD	0x00000000 (0)
		LastKnownGood	REG_DWORD	0x00000002 (2)

Илл. 4.7. Определение местонахождения *CurrentControlSet* в образе данных.

Как показано на илл. 4.7, разделу *ControlSet*, который операционная система использует как *CurrentControlSet* во время своей работы, присвоен номер 1. В приложении Registry Viewer показано два раздела *ControlSet*: *ControlSet001* и *ControlSet002*. (В некоторых случаях можно найти другие номера, в том числе *ControlSet03* и *ControlSet04*, но обычно вы увидите только два раздела *ControlSet*.)

Имя компьютера (ComputerName)

Для того чтобы узнать основные сведения о компьютере, можно использовать большую часть информации, имеющейся в кустах System и Software. Эти файлы кустов содержат много информации, которую можно использовать, чтобы получить основные сведения о системе (т. е. имя компьютера, установленный часовой пояс и информацию о подключенных устройствах и имеющихся общих ресурсах), а также чтобы провести анализ, например, определить конфигурацию аудита в системе, как было продемонстрировано выше.

Имя компьютера можно найти в нижеупомянутом разделе в параметре *ComputerName*:

SYSTEM\CurrentControlSet\Control\ComputerName\ActiveComputerName

Модуль «compname.pl» возвращает следующую информацию:

```
C:\Perl\forensics\rr>rip.pl -p compname -r d:\cases\system
Plugins Dir      = C:\Perl\forensics\rr\plugins/
Launching compname v.20080324
ComputerName     = PETER
```

```
C:\Perl\forensics\rr>rip.pl -p compname -r d:\cases\win7\system
Plugins Dir      = C:\Perl\forensics\rr\plugins/
Launching compname v.20080324
ComputerName     = TUXDISTRO-PC
```

В следующем разделе можно найти время последнего завершения работы компьютера:

SYSTEM\ControlSet00x\Control\Windows

Параметр *ShutdownTime* в данном разделе – это значение структуры *FILETIME*, которое можно соотнести с другими отметками времени в системе, например, с записями журнала регистрации событий (подробнее – в главе 6) и прочими подобными значениями, чтобы создать временную шкалу действий пользователей и использования системы.

Следующий раздел также может быть важен во время анализа:

SOFTWARE\Microsoft\Windows NT\CurrentVersion

Он содержит несколько параметров, предоставляющих сведения о системе. Параметры *ProductName*, *CurrentBuildNumber* и *CSDVersion* подскажут вам, с какой операционной системой и версией (в том числе пакета обновлений) вы работаете. Параметры *RegisteredOrganization* и *RegisteredOwner*, которые часто бывают незаполненными, можно использовать для дальнейшего распознавания системы. Параметры *ProductId* и *InstallDate* также могут быть полезными. Модуль «winnt_cv.pl» извлекает все параметры и их данные из этого раздела:

```
C:\Perl\forensics\rr>rip.pl -p winnt_cv -r d:\cases\vista\software
Plugins Dir = C:\Perl\forensics\rr\plugins/
Launching winnt_cv v.20080609
WinNT_CV
Microsoft\Windows NT\CurrentVersion
LastWrite Time Fri Dec 12 18:26:31 2008 (UTC)
RegisteredOrganization :
    CurrentVersion      : 6.0
    CurrentBuildNumber   : 6001
    CurrentBuild        : 6001
    CSDBuildNumber      : 1616
    SoftwareType         : System
    RegisteredOwner       : Harlan
    SystemRoot           : C:\Windows
    PathName              : C:\Windows
    EditionID             : HomePremium
    CSDVersion            : Service Pack 1
    CurrentType            : Multiprocessor Free
    ProductId              : 89578-OEM-7332157-00204
    BuildLab                : 6001.vistaspl_gdr.080917-1612
```

```
InstallDate      : Fri Aug 31 15:21:10 2007 (UTC)
ProductName     : Windows Vista (TM) Home Premium
BuildGUID       : 4c600e9b-ab0a-4f8e-ac60-b42c6428d3e9
BuildLabEx      : 6001.18145.x86fre.vistaspl_gdr.080917-1612
```

Модуль «*winnt_cv.pl*» создан для сортировки и отображения различных данных параметров по размеру самих данных. Всю эту информацию можно использовать для распознавания и документального оформления исследуемой системы, а также для проведения последующего анализа.

Сведения о часовом поясе (TimeZoneInformation)

Информацию о параметрах часового пояса можно найти в нижеуказанном разделе:

```
SYSTEM\CurrentControlSet\Control\TimeZoneInformation
```

Эта информация может быть чрезвычайно полезна для установления временной шкалы действий, совершенных в системе. В остальной части этой главы мы рассмотрим различные скрипты, которые можно использовать для получения информации из системного реестра; в других главах мы обсудим файлы, записи журнала регистрации событий и т. п. Большинство имеющихся инструментов извлекает информацию, связанную с временем и датами, в формате UTC (всемирное координированное время) или GMT (среднее время по Гринвичу), и вы можете использовать значение параметра *ActiveTimeBias* (указанное в минутах) из раздела *TimeZoneInformation*, чтобы перевести или стандартизировать отметки времени в соответствии с другими источниками в системе, например, с записями в файлах журнала. Модуль «*timezone.pl*» программы RegRipper отобразит эту информацию для вас.

Сетевые интерфейсы

Информация о сетевых интерфейсах, или сетевых интерфейсных платах, хранится в файлах кустов System и Software. Нижеуказанный раздел в файле куста Software содержит сведения о сетевых платах:

```
Microsoft\Windows NT\CurrentVersion\NetworkCards
```

В кусте System сведения о сетевых интерфейсах хранятся в следующем разделе реестра:

```
ControlSet00n\Services\Tcpip\Parameters\Interfaces
```

Как RegRipper, так и «*rip.pl*» могут использовать модули «*networkcards.pl*» и «*nic_mst2.pl*» для извлечения информации о сетевых интерфейсах из соответствующих файлов кустов. Например, модуль «*networkcards.pl*» получает следующую информацию из ОС Windows Vista:

```
C:\Perl\forensics\rr>rip.pl -r d:\cases\vista\software -p networkcards
Launching networkcards v.20080325
NetworkCards
Microsoft\Windows NT\CurrentVersion\NetworkCards
Broadcom 440x 10/100 Integrated Controller [Fri Aug 31 15:19:33 2007]
Intel(R) PRO/Wireless 3945ABG Network Connection [Fri Aug 31 15:19:37 2007]
```

Из той же ОС Windows Vista модуль «*nic_mst2.pl*» извлекает следующее:

```
C:\Perl\forensics\rr>rip.pl -r d:\cases\vista\system -p nic_mst2
```

```
Launching nic_mst2 v.20080324
Network key
ControlSet001\Control\Network\{4D36E972-E325-11CE-BFC1-08002BE10318}
ControlSet001\Services\Tcpip\Parameters\Interfaces
LastWrite time Sun Dec 28 04:44:51 2008 (UTC)
Interface {EE564486-60AE-4868-BB0B-E1A906CC2B44}
Name: Local Area Connection
Control\Network key LastWrite time Tue Sep 11 14:33:17 2007 (UTC)
Services\Tcpip key LastWrite time Sun Dec 28 04:44:51 2008 (UTC)
    DhcpDomain      = us.dell.com
    DhcpIPAddress   = 10.12.138.119
    DhcpSubnetMask   = 255.255.248.0
    DhcpNameServer   = 143.166.95.37 143.166.99.14
    DhcpServer       = 143.166.165.254
Interface {D2B6079F-D864-4E0A-A852-4EB72B87E87B}
Name: Wireless Network Connection
Control\Network key LastWrite time Tue Sep 11 14:33:17 2007 (UTC)
Services\Tcpip key LastWrite time Mon Jan 12 12:46:24 2009 (UTC)
    DhcpDomain      =
    DhcpIPAddress   = 192.168.2.102
    DhcpSubnetMask   = 255.255.255.0
    DhcpNameServer   = 192.168.0.1
    DhcpServer       = 192.168.2.1
```

Вышеуказанная информация может быть чрезвычайно полезна не только при распознавании системы, но и при сопоставлении данных ОС Windows с другими данными, такими как файлы журналов сетевых устройств или сетевые пакеты.

MAC-адрес

Как видите, множество информации о сетевых настройках Windows (т. е. сетевых интерфейсах и их конфигурациях) хранится в реестре. Однако MAC-адрес, жестко запрограммированный, или «зашитый», в сетевой интерфейсной плате, обычно не сохраняется как часть этих конфигурационных данных. Когда операционной системе Windows нужно определить MAC-адрес, она сначала проверяет раздел реестра, и, если не может найти там адрес, обращается к самой сетевой интерфейсной плате. ОС Windows ищет параметр *NetworkAddress* в следующем разделе:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet00x\Control\Class\{4D36E972-E325-11CE-BFC1-08002BE10318}\000n
```

Здесь «000n» обозначает номер сетевой платы. Опять же, MAC-адрес не хранится в этом разделе по умолчанию; те, кто разрабатывают такие инструменты, как *Technitium MAC Address Changer* (<http://tmac.technitium.com/tmac/index.html>), позволяющие изменять или «подделывать» MAC-адреса (иногда в незаконных целях), знают об этом.

Еще одно место, где вы, *возможно*, найдете MAC-адрес (слово «возможно» выделено, потому что это место встречается не во всех системах), находится в следующем разделе:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Genuine Advantage
```

В этом разделе я случайно нашел параметр с именем *MAC*, содержащий несколько MAC-адресов, перечисленных в виде строк (в отличие от двоичных данных, которые нужно анализировать и преобразовывать). Например, в одном из моих компьютеров параметр *MAC* показан в указанном выше разделе и содержит следующие данные (скопировано из реестра):

```
00-15-C5-1B-97-12;00-16-CE-74-2C-B3;00-50-56-C0-00-01;00-50-56-C0-00-08;
```

Каждый перечисленный MAC-адрес соответствует значениям, которые можно увидеть в результатах команды `ipconfig /all`; адреса, начинающиеся с цифр «00-50-56-C0», соответствуют интерфейсам установленной у меня программы VMware.

Скрипт «macaddr.pl» из дистрибутива RegRipper представляет собой что-то вроде гибридного модуля, так как, вместо того, чтобы быть привязанным к какому-то одному файлу куста (Software или System), он содержит код, позволяющий ему определить, для анализа какого файла куста он запущен, и запросить нужную информацию в соответствующем разделе реестра. Если нужный параметр находится в вышеупомянутых разделах в соответствующих файлах кустов, модуль «macaddr.pl» извлечет эту информацию. Интересно, что при исследовании систем Windows Vista (32- и 64-разрядных версий) и Windows 7 некоторые разделы в файле куста System содержали параметр `NetworkAddress`, однако в нем не было данных.

Определение местонахождения MAC-адреса в образе может помочь эксперту в различных ситуациях. MAC-адреса могут быть внедрены в файлы ярлыков (*.lnk) Windows (см. главу 5), или у эксперта могут быть данные захвата сетевого трафика, и ему нужно однозначно определить систему, основываясь на MAC-адресах, которые были обнаружены в собранных данных. Информация о MAC-адресах поможет по меньшей мере однозначно определить систему, даже если используется DHCP-протокол, чтобы назначить другой IP-адрес, и пользователь изменяет имя компьютера.

Общие ресурсы

Очень часто в ОС Windows есть общие ресурсы, чтобы пользователи могли удаленно получать доступ к компьютеру. Во многих случаях это относится к файловым серверам, но то же можно сказать в отношении рабочих станций, ноутбуков и т. д. По умолчанию операционные системы Windows 2000, XP, 2003 и Vista также создают скрытые административные общие ресурсы. Помимо прочих, это общий ресурс IPC\$ (межпроцессное взаимодействие), ADMIN\$, а также ресурсы, относящиеся к корневому каталогу накопителя (накопителей) в системе (C\$, D\$ и т. д.). Если пользователь создает дополнительный ресурс, например, с помощью команды `net share`, этот ресурс будет показан в следующем разделе (если не указано иное, все разделы реестра, упомянутые в этой части главы, находятся в кусте HKEY_LOCAL_MACHINE):

```
SYSTEM\CurrentControlSet\Services\lanmanserver\Shares
```

Можно использовать модуль «shares.pl», чтобы извлечь из файла куста System всю информацию об имеющихся общих ресурсах, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -p shares -r d:\cases\local\system
Plugins Dir = C:\Perl\forensics\rr\plugins/
Launching shares v.20090112
print$
  Path=C:\WINDOWS\system32\spool\drivers
  Remark=Printer Drivers
  Type=0
SharedDocs
  Path=C:\DOCUMENTS AND SETTINGS\ALL USERS\DOCUMENTS
  Remark=
  Type=0
Printer2
  Path=hp deskjet 5550 series,LocalsplOnly
  Remark=hp deskjet 5550 series
  Type=1
```

Как видите, модуль находит сведения об общем ресурсе, включая имя ресурса, путь к ресурсу, любые примечания и тип ресурса. Согласно классу инструментария WMI (Windows Management Instrumentation) `Win32_Share` ([http://msdn.microsoft.com/en-us/library/aa394435\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa394435(VS.85).aspx)), `Type=0` означает, что общий ресурс – это накопитель, а `Type=1` означает, что общий ресурс – это принтер.

Возможно, вы не увидите списка общих ресурсов в разделе Shares. Это означает, что пользователь не создал ни одного общего ресурса. Однако ОС Windows автоматически создает несколько административных общих ресурсов. Пользователи или администраторы могут *отключить* возможность создания этих скрытых административных общих ресурсов, что будет указано в следующем разделе:

SYSTEM\CurrentControlSet\Services\lanmanserver\parameters

Если в этом разделе есть параметр *AutoShareServer* (<http://support.microsoft.com/kb/288164>), и его значение равно 0, это свидетельствует о том, что настройки системы были специально изменены, чтобы исключить возможность создания скрытых административных общих ресурсов. Модуль «shares.pl» проверяет наличие параметра *AutoShareServer*.

Предупреждение

В разных версиях ОС Windows имена некоторых подразделов могут быть написаны по-разному. Например, в Windows XP раздел Lanmanserver пишется как «lanmanserver», тогда как в Vista он пишется как «LanmanServer». В Windows XP подраздел Parameters (в разделе Lanmanserver) пишется как «parameters», а в Vista – как «Parameters». Модуль «shares.pl» содержит код, учитывающий эти различия.

Политика аудита и журналы регистрации событий

В ОС Windows 2000, XP и 2003 политика аудита системы (<http://support.microsoft.com/kb/246120>) содержится в кусте Security в разделе Policy\PolAdtEv. Параметр «По умолчанию» (Default) представляет собой тип данных REG_NONE и содержит двоичные данные, в которых закодирована политика аудита. Ниже показаны сведения о политике аудита, извлеченные из тестового образа с помощью утилиты Offline Registry Parser:

Первое двойное слово (четыре байта) двоичных данных (на самом деле, первый байт) показывает, включен ли аудит. В данном случае значение равно 01, поэтому аудит включен (00 означает, что он выключен). В ОС Windows 2000 и XP есть девять типов событий, для которых может производиться аудит, и каждое из них представлено значением двойного слова в последовательности байтов. Последнее значение двойного слова не используется.

Предупреждение

Информация в этом разделе относится только к ОС Windows 2000, XP и 2003. Механизмы аудита и ведения журналов для операционных систем Windows изменились с появлением версии Vista; к изменениям относятся появление большего количества журналов и другой формат журналов (XML, а не формат двоичных данных).

Чтобы расшифровать эту информацию, необходимо разобраться в формате ее записи. Сопоставьте следующий шаблон с данными полученными из раздела PolAdtEv:

```
0Z XX XX AA 00 00 00 BB 00 00 00 CC 00 00 00 DD 00 00 00 EE 00 00 00  
FF 00 00 00 GG 00 00 00 HH 00 00 00 II 00 00 00 XX 00 00 00
```

Значение для *Z* определяет состояние аудита (1 – включен, 0 – выключен). Остальные значения соответствуют следующему списку (значения для *X* нас не интересуют):

AA	Аудит системных событий
BB	Аудит входа в систему
CC	Аудит доступа к объектам
DD	Аудит использования привилегий
EE	Аудит отслеживания процессов
FF	Аудит изменения политики
GG	Аудит управления учетными записями
HH	Аудит доступа к службе каталогов
II	Аудит событий входа в систему для учетных записей

Для каждой пары букв 00 означает, что аудит не выполняется, 01 – аудит выполняется для успешных событий, 02 – аудит выполняется для неудачных событий, 03 – аудит выполняется как для успешных, так и неудачных событий.

Мы видим, что в тестовом образе был включен аудит успешных и неудачных действий для системных событий, событий входа в систему, событий изменения политики, событий управления учетными записями и событий входа в систему для учетных записей.

Эта информация может быть полезна во время исследования систем Windows 2000, XP и 2003, так как она укажет нам, какие события следует ожидать увидеть в журнале регистрации событий. (Мы рассмотрим анализ журналов регистрации событий в главе 5.) Модуль «auditpol.pl» программы RegRipper извлечет эти данные из файла куста Security и преобразует их так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -p auditpol -r d:\cases\local\security  
Plugins Dir = C:\Perl\forensics\rr\plugins/  
Launching auditpol v.20080327  
auditpol  
Policy\PolAdtEv  
LastWrite Time Mon Aug 7 16:14:22 2006 (UTC)  
**Auditing is NOT enabled.  
  
Plugins Dir = C:\Perl\forensics\rr\plugins/  
Launching auditpol v.20080327  
auditpol  
Policy\PolAdtEv  
LastWrite Time Fri Sep 9 01:11:43 2005 (UTC)  
Auditing is enabled.  
Audit System Events      = S/F  
Audit Logon Events       = S/F  
Audit Object Access      = N  
Audit Privilege Use      = N  
Audit Process Tracking   = N  
Audit Policy Change      = S/F  
Audit Account Management = S/F  
Audit Dir Service Access = N  
Audit Account Logon Events = S/F
```

Как видите, в одном случае модуль «auditpol.pl» указал, что аудит не включен, а в другом – что аудит выполняется для различных событий (N = аудит не выполняется, S = аудит успешных событий, F = аудит неудачных событий и S/F = аудит успешных и неудачных событий).

Информация о самих файлах журнала событий хранится в разделе HKLM\System\ControlSet00x\Services\EventLog. Это касается всех операционных систем Windows, в том числе Windows 2000, XP, 2003, Vista и Windows 7. Отличие между версиями Windows заключается в том, сколько подразделов находится в разделе EventLog, так как число подразделов равняется количеству различных имеющихся журналов регистрации событий. Например, в ОС Windows XP чаще всего можно найти подразделы Application, System, Security и, возможно, даже Internet Explorer (если браузер Internet Explorer был обновлен до версии 7). В случае с Windows 2003 можно также найти журнал событий DNS-сервера, а кроме того мне встречались журналы регистрации событий для отдельных приложений. В большинстве случаев (особенно со стандартными журналами регистрации событий Microsoft) вы найдете параметры, относящиеся к имени и местонахождению файла журнала событий, его максимальному размеру и количеству дней, в течение которых будут сохраняться записи журнала. (Хотя статья № 102998 из базы знаний Microsoft, <http://support.microsoft.com/kb/102998>, устарела, информация в ней все еще имеет отношение к этой теме.) Модуль «eventlog.pl» извлекает эту информацию, преобразовывает разнообразные параметры в пригодный для чтения формат и отображает данные так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -p eventlog -r d:\cases\local\system
Launching eventlog v.20090112
Application \ Fri Dec 12 12:06:13 2008Z
  File          = %SystemRoot%\system32\config\AppEvent.Evt
  DisplayNameFile = %SystemRoot%\system32\els.dll
  MaxSize        = 512.00KB
  Retention      = 7.00 days
  AutoBackupLogFile = 0
Internet Explorer \ Fri Aug 29 00:36:11 2008Z
Security \ Fri Aug 29 00:36:11 2008Z
  File          = %SystemRoot%\System32\config\SecEvent.Evt
  DisplayNameFile = %SystemRoot%\System32\els.dll
  MaxSize        = 512.00KB
  Retention      = 7.00 days
System \ Mon Oct 6 23:15:13 2008Z
  File          = %SystemRoot%\system32\config\SysEvent.Evt
  DisplayNameFile = %SystemRoot%\system32\els.dll
  MaxSize        = 512.00KB
  Retention      = 7.00 days
```

Как видите модуль «eventlog.pl» извлекает много информации о параметрах журнала регистрации событий. Имя каждого журнала регистрации событий отображается вместе с отметкой времени *LastWrite*, указывающей дату последнего изменения содержимого в разделе. Помимо пути к самому файлу журнала, отображаются максимальные размеры файлов и сроки сохранения записей (которые представлены в реестре в виде секунд, а модуль отображает их в виде количества дней). Как видно из предыдущей записи журнала регистрации событий приложения, также отображается значение *AutoBackupLogFile* (которое описано в статье № 312571 из базы знаний Microsoft, <http://support.microsoft.com/kb/312571/>), при наличии такого. Еще одно значение, характерное для журнала событий безопасности, – это *WarningLevel* (описанное в статье № 945463 из базы знаний Microsoft, <http://support.microsoft.com/kb/945463>), которое заставляет журнал безопасности создавать событие с идентификатором 523, когда размер журнала достигает заданного значения. Модуль «eventlog.pl» также отображает это значение, если такое присутствует.

Совет

Примерно в то же время, когда я создавал модуль «eventlog.pl», Дон Уэбер (Don Weber) из компании Cutaway Security (www.cutawaysecurity.com/) написал свой модуль, который он назвал «eventlogs.pl» и разместил на форуме сайта RegRipper.net.

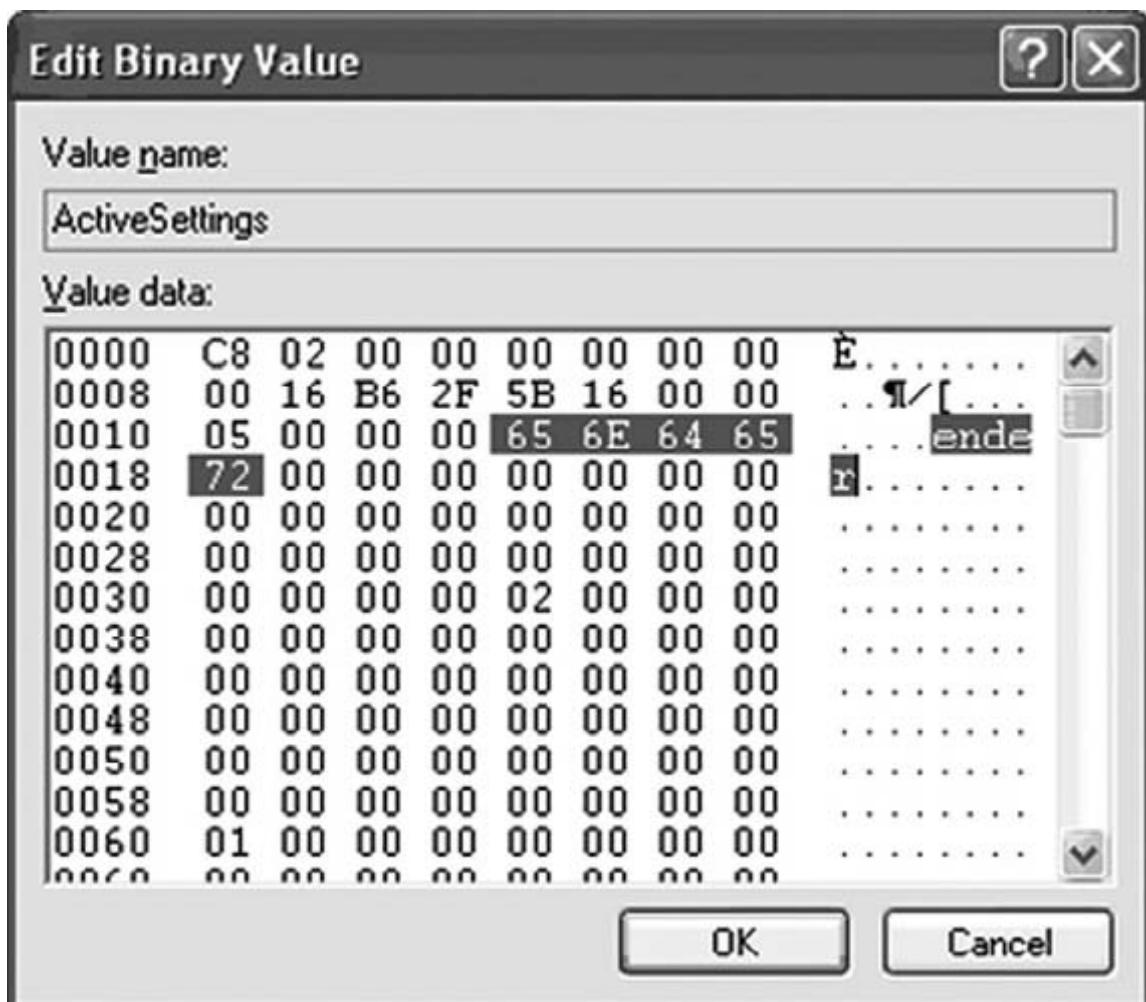
Кроме того, корпорация Microsoft выпустила (как часть пакета Windows 2000 Resource Kit) Perl-скрипт «eventlog.pl» для управления журналами событий. Скрипт описан в статье № 318763 из базы знаний Microsoft, <http://support.microsoft.com/kb/318763>. Он предназначен для запуска на работающих компьютерах и позволяет администратору получить большую часть той же информации, что можно получить с помощью модуля «eventlog.pl» программы RegRipper, в частности, свойства журналов событий.

Идентификаторы SSID беспроводной сети

В работающем компьютере (чаще всего в ноутбуках) операционная система Windows хранит список идентификаторов SSID беспроводных сетей, к которым она подключалась. Если беспроводные подключения управляются службой настройки беспроводной связи (Wireless Zero Configuration Service, WZCSVC), то этот список содержится в следующем разделе реестра:

```
SOFTWARE\Microsoft\WZCSVC\Parameters\Interfaces\{GUID}
```

В данном случае *GUID* – это глобальный уникальный идентификатор для беспроводного интерфейса. В этом разделе вы, возможно, увидите параметр *ActiveSettings* и несколько других параметров с именами *Static#000x*, где *x* – это целое число, начиная с нуля. Эти параметры полностью состоят из двоичных данных, и все идентификаторы SSID для точек доступа беспроводной сети, к которым выполнялось подключение, включены в эти двоичные данные. В двоичных данных, по смещению 0x10, находится значение двойного слова, содержащее длину идентификатора SSID. Имя идентификатора SSID, длина которого равна количеству указанных байтов/символов, следует сразу после этого значения двойного слова. На илл. 4.8 показано двоичное содержимое параметра *ActiveSettings*, полученного из работающей системы. Идентификатор SSID выделен.



Илл. 4.8. Параметр *ActiveSettings*, в котором выделен идентификатор SSID.

Время от времени возникает вопрос, касающийся способов определения идентификаторов SSID сетей, к которым была подключена система. Это может быть полезным в случаях, когда вы имеете дело с несанкционированным доступом или когда нужно отследить IP-адрес, принадлежащий пользователю. Модуль «ssid.pl» может извлечь данные из куста Software, включая идентификатор SSID беспроводной сети, к которой было выполнено подключение, и дату последнего подключения к этой сети, как показано ниже:

```
C:\Perl\forensics\rri>rip.pl -p ssid -r d:\cases\test\software  
Plugins Dir = C:\Perl\forensics\rri\plugins/  
Launching ssid v.20080327  
SSID  
Microsoft\WZC SVC\Parameters\Interfaces  
NIC: 11a/b/g Wireless LAN Mini PCI Express Adapter  
    Static#0000 SSID : tmobile [Wed Oct 3 16:44:25 2007]  
    Static#0001 SSID : ender [Mon Oct 8 10:12:46 2007]
```

Из этого примера мы видим, что компьютер (в данном случае ноутбук) подключался точке доступа беспроводной сети с SSID-идентификатором «tmobile» (обычно предлагается в сети кофеен Starbucks, помимо других мест) в октябре 2007 года и к сети с SSID-идентификатором «ender» в том же месяце. Эта информация может быть чрезвычайно полезна для установления временной шкалы действий, совершенных в системе, особенно при сопоставлении с другими данными. Однако имейте в виду, что информация о времени из этого модуля (а также из других модулей) представлена в

формате UTC (всемирное координированное время), и, возможно, ее придется корректировать для соответствующего часового пояса.

Места автоматического запуска

Места автозапуска в реестре – это места, из которых приложения могут запускаться без взаимодействия с пользователем и очень часто без ведома пользователя, пока он делает... что-нибудь в системе. Изначально эти места предоставлялись для удобства пользователя. Некоторые программы, такие как драйверы сенсорной панели и приложения на ноутбуке, а также антивирусное ПО и брандмауэры наиболее эффективны, когда они запускаются автоматически. В других случаях пользователям удобно, когда приложения, например, клиенты обмена мгновенными сообщениями, запускаются автоматически при входе пользователя в систему.

В 2004 году на конференции USENIX члены научно-исследовательского центра CyberSecurity and Systems Management Research Group корпорации Microsoft представили статью «Gatekeeper: Monitoring Auto-Start Extensibility Points for System Management». Авторы статьи называют места автозапуска точками расширения для автоматического запуска (англ. *auto-start extensibility points*, ASEP). В этой статье точки ASEP распределены по категориям не так, как у меня, и представлена схема, в которой показано, что наиболее популярная точка ASEP, используемая шпионским ПО (на момент написания статьи), – объект модуля поддержки браузера (англ. *browser helper object*, далее – ВНО). Объект ВНО – это по существу динамическая библиотека (DLL), которая автоматически загружается при запуске браузера Internet Explorer, после чего она может отслеживать действия пользователя. В анализе конкретной ситуации «Объекты ВНО» показан пример использования мест автозапуска (точек ASEP).

Ваш компьютер защищен?

Анализ конкретной ситуации: Объекты ВНО*

Я работал администратором системы безопасности в фирме, предлагающей финансовые услуги, когда мне случайно встретился интересный способ использования объектов ВНО. Мой работодатель предоставлял услуги по контролю кредитов и защите от кражи персональных данных, которые включали в себя различные уровни мониторинга: от отправки квартальных отчетов до практически мгновенного создания веб-страниц и/или сообщений электронной почты по мере поступления запросов. И, как и у большинства фирм, у моего работодателя были конкуренты. Однажды мне позвонили ребята из отдела коммерческого развития нашей фирмы относительно возможной проблемы в системе безопасности. Когда пользователи заходили на наш веб-сайт и загружали главную страницу, каждый элемент названия компании выделялся. Щелкнув по такому названию, пользователь перенаправлялся на веб-сайт конкурента!

Оказалось, что на компьютере пользователей (были «заражены» компьютеры некоторых клиентов и сотрудников отдела коммерческого развития) установлен объект ВНО, который отслеживал страницы, посещенные пользователем. Различные компании подписывались на услуги поставщика этого рекламного ПО/объекта ВНО, и когда название их конкурента появлялось на веб-странице, объект ВНО автоматически преобразовывал это название в гиперссылку на веб-сайт абонента. Предположим, компания бытовой электроники «А» подписалась на услуги поставщика этой рекламной программы. Когда рекламное ПО заражало компьютер пользователя, то при каждой загрузке веб-страницы, содержащей названия конкурентов компании «А», эти имена будут преобразованы в гиперссылки на веб-сайт компании «А».

К счастью, корпорация Microsoft любезно предоставляет из своей базы знаний статью № 298931 (<http://support.microsoft.com/kb/298931>), в которой объясняется, как отключить объекты ВНО. В этой статье предоставляются ссылки на более подробную информацию об объектах ВНО, перечисленных в разделе реестра

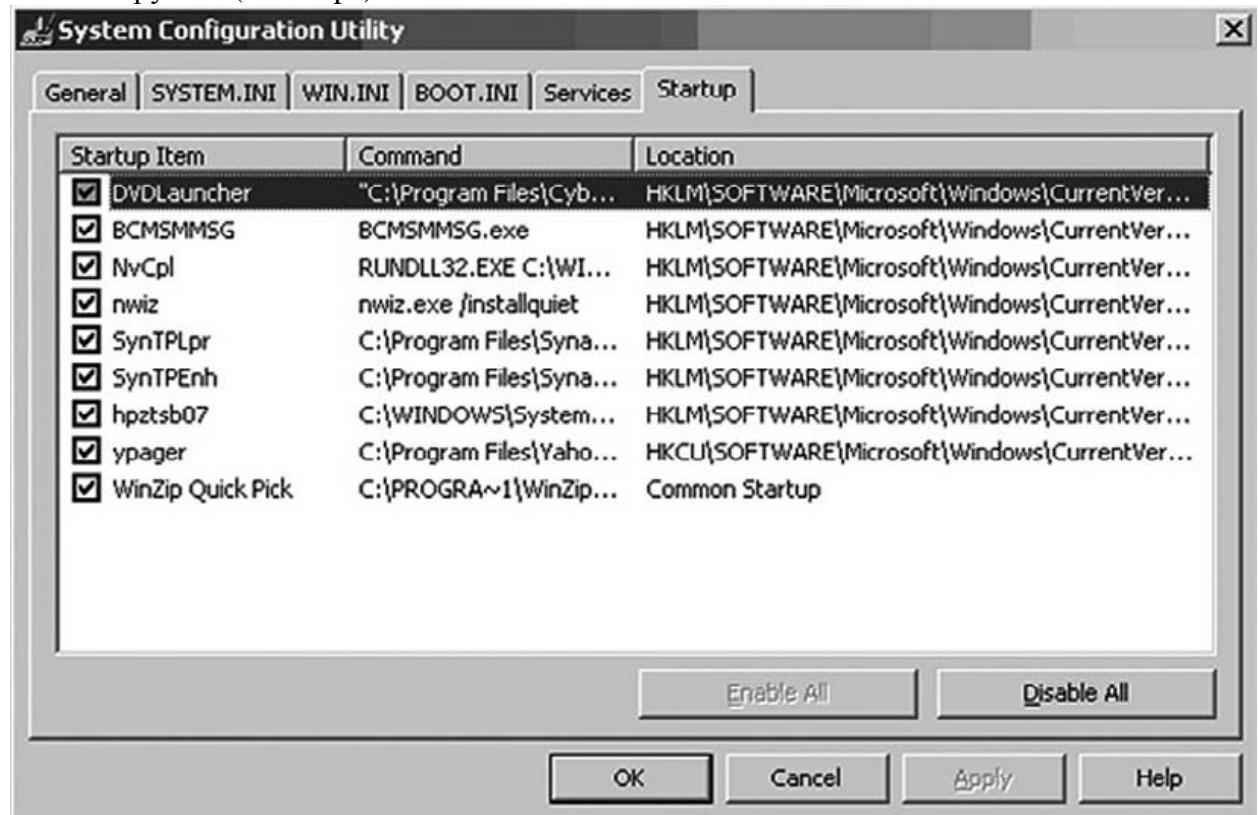
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects.
Модуль «bho.pl» программы RegRipper извлечет и отобразит эту информацию для вас.

Обратите внимание, что этот раздел находится в кусте HKEY_LOCAL_MACHINE, что означает, что объекты ВНО загружаются при каждом открытии браузера независимо от желания пользователя.

Perl-скрипт «bho.pl», расположенный (в каталоге ch4\code\old) на носителе, который идет в комплекте с этой книгой, позволяет узнать, какие объекты ВНО установлены на работающем компьютере.

* ВНО - объекты модуля поддержки браузера.

Существует мнение, что большинство пользователей и администраторов совсем не знакомы с местами автозапуска в реестре и считают, что очень немногие разделы (в частности, распространенные Run и RunOnce) можно использовать для этих целей. Это частично подтверждается документацией и приложениями, предоставляемыми производителем операционной системы. Например, на работающем компьютере с ОС Windows XP команда *MSConfig* запускает служебную программу «Настройка системы» («System Configuration») (<http://support.microsoft.com/kb/310560>). Для того чтобы выполнить эту команду, щелкните по кнопке «Пуск» («Start») в панели задач, выберите кнопку «Выполнить» («Run»), в текстовом поле введите команду **msconfig** и нажмите клавишу **Enter**. На илл. 4.9 показано открытое окно служебной программы «Настройка системы» («System Configuration»), в котором отображено содержимое вкладки «Автозагрузка» («Startup»).



Илл. 4.9. Вкладка «Автозагрузка» («Startup») в программе «Настройка системы» («System Configuration»).

Выполните эту команду в ОС Windows XP и внимательнее посмотрите на столбец «Расположение» («Location»). Вы увидите, что в реестре анализируются только разделы Run из кустов HKEY_CURRENT_USER и HKEY_LOCAL_MACHINE. К сожалению, похоже, что это единственные разделы, в которых выполняется проверка. Существует класс инструментария WMI с именем *Win32_StartupCommand*, который позволяет программисту автоматически находить содержимое этих разделов, но, как и в случае с

программой «Настройка системы» (“System Configuration”), он проверяет только ограниченное число мест автозапуска в реестре.

Согласно другой точке зрения, число мест автозапуска в реестре настолько велико, что исследование этих мест лучше оставить специалистам и/или специальным приложениям, таким как коммерческие антивирусные и антишпионские программы.

Истина находится где-то посередине. Да, в реестре существует много мест автозапуска, но их число ограничено. В следующих частях этой главы я разобью эти места на три категории и опишу несколько разделов реестра, к которым происходит обращение при начальной загрузке системы, при входе в систему и при выполнении некоторых действий в системе. Затем мы рассмотрим способы перечисления записей в этих местах. Однако перечисленные разделы не будут представлять собой полный и исчерпывающий список. На носителе, который идет в комплекте с этой книгой, есть таблица «regref.xls», содержащая несколько листов. Каждый лист включает в себя различные разделы системного реестра, относящиеся к определенной категории, а также краткое описание раздела и, если необходимо, достоверную справочную информацию, описывающую функциональные возможности этого раздела. Эту таблицу следует рассматривать отправным пунктом для анализа реестра, но, так как существует большое количество приложений и постоянно появляются их новые версии, ее нельзя считать исчерпывающей.

Начальная загрузка системы

Разделы автозагрузки в реестре, к которым происходит обращение при начальной загрузке системы, чаще всего используются создателями вредоносных программ, так как эти места позволяют их программам запускаться без взаимодействия с пользователем (не требуется даже вход пользователя в систему). Одно из таких мест в ОС Windows – это раздел *Services*, а точнее:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
```

При загрузке системы определяется значение для текущего раздела *ControlSet* и используются параметры этого раздела. Осуществляется поиск служб в разделе *ControlSet*, и загружаются службы, в настройках которых указан автоматический запуск (параметр *Start* установлен в значение 0x02).

Во время анализа вторжения в систему чаще всего приходиться иметь дело только с образом клонированных данных и отчетом об инциденте, который, как правило, представляет собой запись типа: «Мы видели, как это произошло в такое-то время». Столкнувшись с такой ситуацией, лучше всего сначала открыть образ в программе ProDiscover, заполнить данными программу просмотра реестра (Registry Viewer), определить местонахождение раздела *ControlSet*, который отмечен как текущий (*Current*), а затем отсортировать его подразделы по отметкам времени *LastWrite*. Большая часть этих отметок времени *LastWrite* обычно будет перечислена в соответствии с временем последней загрузки системы, так как многие разделы *Services* изменяются, а не просто считаются, во время начальной загрузки системы. Однако в некоторых случаях я использовал этот способ, чтобы найти установленные злоумышленником службы и драйверы, в том числе руткиты и драйвер «rdrv.sys» (который представляет собой драйвер руткита, используемый некоторыми троянскими программами и чат-ботами). Иногда я находил эти службы, но ни имена служб, ни отметки времени *LastWrite* не были взаимосвязаны с данными из отчета об инциденте, который я получил. По существу, я расследовал одно вторжение, а находил другое!

Можно найти и другие интересные артефакты, используя тот же способ. Например, я находил драйвер «npf.sys», установленный вместе с утилитами WinPcap, которые позволяют выполнять анализ сетевых пакетов в системе. Этот драйвер устанавливается

такими инструментами, как Wireshark или анализаторами пакетов, имеющимися в утилитах WinPcap, но он также может быть установлен злоумышленником.

Программа RegRipper может использовать два модуля, «svc.pl» и «svc2.pl», для анализа информации, доступной в разделе Services. В обоих случаях информация извлекается и сортируется по времени *LastWrite* каждого подраздела. Модуль «svc.pl» показывает суммарную информацию, тогда как «svc2.pl» предоставляет более подробные данные (в том числе отображаемое имя и тип служб или драйверов, путь к ним и т. д.) в формате значений, разделенных запятыми (.csv). Запустив модуль «svc2.pl» с помощью «rip.pl», его выходные данные можно перенаправить в файл и открыть в программе Excel для быстрого анализа.

В некоторых случаях специалисту по расследованию инцидентов, возможно, потребуется получить доступ к информации раздела Services на работающем компьютере. Например, существует вредоносная программа, использующая службы Windows в качестве своего механизма сохраняемости и устанавливающаяся как DLL-файл, который запускается под видом процесса «svchost.exe» (в статье № 314056 из базы знаний Microsoft, <http://support.microsoft.com/kb/314056>, описывается процесс «svchost.exe» в ОС Windows XP Professional). Так как вредоносную программу, использующую этот механизм сохраняемости, непросто обнаружить (например, червь Conficker создает случайные записи, поэтому найти его еще сложнее), я создал инструмент «regscan.pl» на основе командной строки. Этот инструмент автономный и не является частью программы RegRipper; он доступен на носителе, который идет в комплекте с этой книгой, в каталоге the ch4\code\Regscan. Он анализирует информацию в разделе Services и выводит основные данные о каждой службе, в том числе время *LastWrite* раздела и параметры *ImagePath* и *Parameters\ServiceDll* (при наличии таковых). Эти данные разделены вертикальной чертой и перенаправляются на стандартное устройство вывода. Специалист по расследованию инцидентов может запустить модуль «regscan.pl» (или соответствующий скомпилированный исполняемый файл) на локальном компьютере или предоставить IP-адрес или имя компьютера, к которому у него есть доступ с правами администратора, и извлечь те же данные. Найти вредоносную программу, установленную с помощью этого способа, так же просто, как ввести следующую команду:

```
C:\Perl\regscan>regscan.pl | find "svchost.exe -k netsvcs"
```

Это команда извлекает всю информацию о службах на локальном компьютере и сокращает количество данных, направляя результаты через команду *find*, чтобы показать только те записи, которые запущены через процесс «svchost.exe». Ниже показан пример выходных данных этой команды:

```
Wed Jan 14 01:16:37 2009Z|BITS|%SystemRoot%\system32\svchost.exe -k  
netsvcs|C:\WINDOWS\system32\qmgr.dll  
Wed Jan 14 01:16:37 2009Z|Browser|%SystemRoot%\system32\svchost.exe -k  
netsvcs|%SystemRoot%\System32\browser.dll  
Wed Jan 14 01:16:37 2009Z|CryptSvc|%SystemRoot%\system32\svchost.exe -k  
netsvcs|%SystemRoot%\System32\cryptsvc.dll  
Wed Jan 14 01:16:37 2009Z|Dhcp|%SystemRoot%\system32\svchost.exe -k  
netsvcs|%SystemRoot%\System32\dhcpcsvc.dll
```

Вертикальная черта (|) между каждым сегментом данных упрощает их анализ и позволяет расположить данные каждого раздела в одной строке, чтобы можно было использовать другие инструменты, например, команду *find*, для сокращения объема отображаемых данных, которые необходимо обработать. Данные перечислены по времени *LastWrite* каждой службы, поэтому, если эксперт приблизительно знает, в какое время или какой период времени произошел инцидент, он может еще больше сократить объем данных.

Вход пользователя в систему

Согласно документации Microsoft, процесс начальной загрузки системы считается неполным, пока пользователь не войдет в систему. Когда пользователь входит в систему, обрабатываются определенные разделы реестра, чтобы можно было запустить перечисленные в них приложения. Ниже указаны эти разделы (по порядку):

1. HKLM\Software\Microsoft\Windows\CurrentVersion\Runonce
2. HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
3. HKLM\Software\Microsoft\Windows\CurrentVersion\Run
4. HKCU\Software\Microsoft\Windows NT\CurrentVersion\Windows\Run
5. HKCU\Software\Microsoft\Windows\CurrentVersion\Run
6. HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce

Для краткости, символы *HKLM* обозначают куст HKEY_LOCAL_MACHINE, *HKCU* – куст HKEY_CURRENT_USER.

Каждый раз, когда новый пользователь входит в систему, выполняется анализ разделов 1, 3, 5 и 6, и запускаются программы, перечисленные в этих разделах (<http://support.microsoft.com/kb/137367>). По умолчанию эти разделы автозапуска игнорируются при загрузке компьютера в безопасном режиме. Но если в ОС Windows XP и 2003 поставить звездочку (*) перед параметрами *RunOnce* (разделы 1 и 6), можно запустить программы, связанные с этими разделами, даже когда компьютер загружается в безопасном режиме (<http://support.microsoft.com/kb/314866>). Кроме того, разделы 1, 3, 5 и 6 в ОС Windows XP предоставлены для устаревших программ и обратной совместимости, чтобы можно было использовать программы, написанные для предыдущих версий Windows (до выпуска XP).

Действия пользователей

Места автозапуска в реестре, относящиеся к этой категории, – это разделы, к которым происходит обращение, когда пользователь выполняет какое-нибудь действие, например, открывает приложение Explorer или Outlook. Если вы запустите программу RegMon в системе и просто пошевелите мышью или откроете какое-нибудь приложение (или совсем ничего не будете делать), то увидите, что выполняется много обращений к реестру, даже когда пользователь не взаимодействует с системой. Как и в случае с местами автозапуска, создатели вредоносных программ (вирусов, троянов, червей и т. д.) считают такие разделы чрезвычайно полезными для поддержания сохраняемости и работоспособности своих продуктов.

Один из таких известных разделов – это:

HKEY_LOCAL_MACHINE\Software\Classes\Exefile\Shell\Open\command

Этот раздел реестра, а также разделы для других классов файлов (batfile, comfile и т. д.) управляют действиями, которые происходят при открытии этого класса файла. Например, в проводнике Windows щелкните правой кнопкой мыши по любому файлу, поле чего откроется контекстное меню, в котором слово «Открыть» (“Open”) вверху выделено полужирным шрифтом. Действие, выделенное полужирным шрифтом, – это то, что чаще всего происходит при выполнении двойного щелчка мышью по файлу. После того как вы дважды щелкните по файлу, ОС Windows выполнит сканирование реестра на наличие параметров для этого класса файла и, исходя из них, определит, какое действие предпринять. Вредоносные программы, такие как черви SirCam (<http://support.microsoft.com/kb/311446>) и Pretty Park (<http://support.microsoft.com/kb/310585>), используют эти разделы реестра для поддержания сохраняемости в зараженной системе.

Предположим, что вы хотите сыграть на компьютере в игру «Косынка» (“Solitaire”). Вы переходите к командной строке и вводите команду:

```
C:\>dir /s sol.exe
```

В выходных данных этой команды будет указано, где находится исполняемый файл игры «Косынка» в файловой системе. (На моем компьютере с ОС Windows XP Home файл «sol.exe» расположен в каталоге C:\Windows\system32.) Если, просто из любопытства, вам интересно, что происходит при двойном щелчке по значку файла «Косынка», введите следующую команду:

```
C:\>ftype exefile
```

В выходных данных команды будет показано: *exefile=%1 %**. Эти значения указывают системе запустить файл с первым аргументом (имя файла) и любыми последующими аргументами. Однако запись реестра *shell\open\command* можно дополнить, чтобы при открытии отдельного класса файла запускались другие файлы. По умолчанию записи в этом и других подобных разделах (которые сейчас будут описаны) должны содержать только значения “%1” %*. Любые другие данные в этом параметре должны считаться подозрительными, и их нужно немедленно изучить.

Еще одна запись, содержащая похожую информацию, – это:

```
HKEY_CLASSES_ROOT\Exefile\Shell\Open\Command
```

Эта функция применяется не просто к записи Exefile в кусте HKEY_CLASSES_ROOT. Некоторые вредоносные программы изменяют другие похожие записи, чтобы обеспечить свою сохраняемость в системе. Например, программы удаленного администрирования (backdoor) иногда изменяют записи в следующем разделе:

```
HKEY_CLASSES_ROOT\Word.Document.x\shell\open\command
```

В данном случае *x* – это номер версии (8, 9 и т. д.) программы Word. Это значение указывает системе, что при выполнении команды «Открыть» (“Open”) для документа Microsoft Word через оболочку (проводник Windows), например, при двойном щелчке по документу, будет запущена вредоносная программа.

Еще одно место, которое можно использовать таким же образом, находится в следующем разделе:

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Command Processor\AutoRun
```

Подробную информацию об этом параметре AutoRun можно найти по адресу <http://technet.microsoft.com/en-us/library/bb490880.aspx>.

В этом параметре реестра перечислены команды, исполняемые каждый раз при запуске обработчика команд (cmd.exe). Например, этот параметр может запустить какую-либо программу при открытии командной строки. По умолчанию данный параметр пустой. Можно сделать записи в том же параметре в кусте HKEY_CURRENT_USER; эти записи имеют преимущество перед записями в кусте HKEY_LOCAL_MACHINE.

Чтобы продемонстрировать, насколько это просто, откройте редактор реестра и перейдите к разделу Command Processor. В моей ОС Windows XP Pro SP2 параметр AutoRun отображается, но с ним не связано никаких данных. Щелкните по параметру правой кнопкой мыши, выберите пункт «Изменить» (“Modify”) и введите необходимые данные. На илл. 4.10 показан пример данных, которые можно добавить в этот параметр реестра.



Илл. 4.10. Добавление данных в параметр AutoRun в разделе Command Processor.

После изменения этого параметра нажмите кнопку «**OK**». Затем щелкните по кнопке «**Пуск**» (“Start”), выберите пункт «**Выполнить**» (“Run”), введите **cmd** и нажмите **Enter**. Откроется командная строка, а также приложение, которое вы выбрали. Как показано на илл. 4.10, я выбрал приложение с графическим интерфейсом пользователя, чтобы при запуске командной строки было заметно, что также открылось другое приложение. Таким образом можно выполнить ряд других действий, например, добавить учетную запись пользователя, установить службу или запустить вредоносную программу без графического интерфейса.

Что касается графического интерфейса пользователя, есть малоизвестный параметр реестра, который можно использовать, чтобы загружать DLL в память всякий раз, когда запускается графический интерфейс. Этот параметр указан ниже:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs
```

В статье № 197571 из базы знаний Microsoft (<http://support.microsoft.com/kb/197571>) указано, что библиотеки DLL, перечисленные в этом параметре, загружаются в память с помощью функции *LoadLibrary()* во время процесса **DLL_PROCESS_ATTACH** файла «*user32.dll*»; в этом файле библиотеки хранятся команды для различных элементов графического интерфейса, таких как окна и диалоговые окна.

В 2000 году на конференциях Black Hat и USENIX Дж. Д. Глейсер (J. D. Glaser), который раньше работал в компании Foundstone, а сегодня управляет NT OBJECTives, показал презентацию, касающуюся отслеживания взлома сервера и поиска записи в разделе с параметром **AppInit_DLLs**.

Параметр **AppInit_DLLs** очень эффективен в качестве укрытия для вредоносных программ. Известно, например, что его использует шпионская программа CoolWebSearch. Почему этот параметр так эффективен? Когда я проводил практические курсы по расследованию инцидентов, одно из первых упражнений, которое я давал, было простое упражнение на демонстрацию заражения, предназначенное для того, чтобы наблюдать за действиями слушателей курсов, а не для того, чтобы определить, кто первым сможет найти заражение. Я читал этот курс не только начинающим и опытным администраторам Windows, но и квалифицированным администраторам UNIX и Linux, которые также отвечали за работу операционных систем Windows. Неизменно, во всех случаях, первое, что делает каждый слушатель, – открывает на рабочем столе инструмент с графическим интерфейсом пользователя. Это может быть программа «Просмотр событий» (“Event Viewer”), диспетчер задач или даже проводник Windows. Однако их первый инстинкт неизменен – открыть приложение с графическим интерфейсом.

Операционная система Windows предоставляет возможность уведомлять внешние функции, когда в системе совершаются определенные действия, например, когда пользователь входит в систему или выходит из нее или когда запускается экранная заставка. Эти уведомления управляются следующим разделом реестра:

```
HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\
```

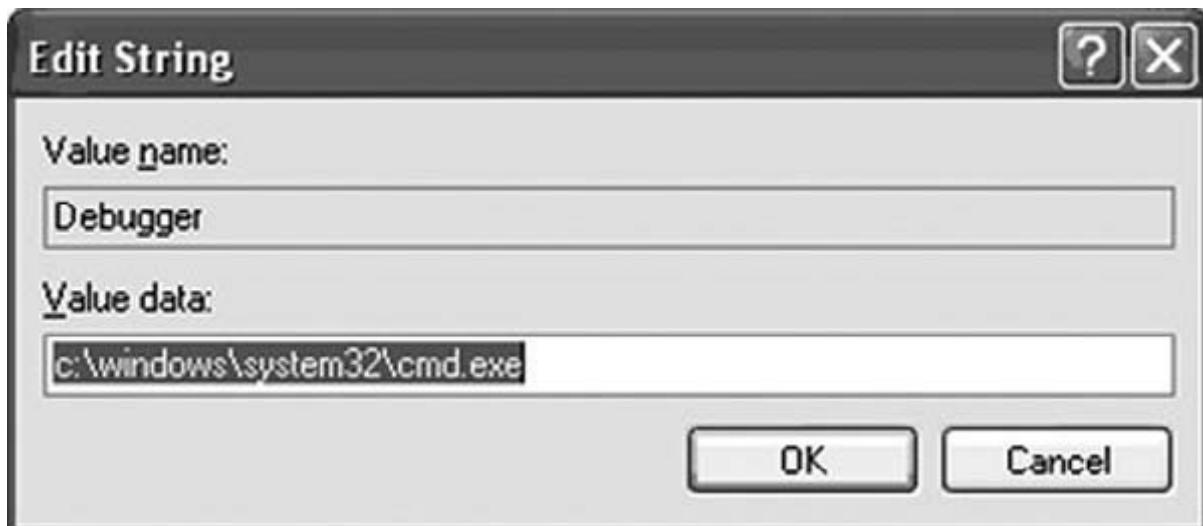
Записи в этом разделе указывают на библиотеки DLL, которые получают уведомления об определенных событиях. Выполнив в Google поиск по словам «*Winlogon\Notify*», вы найдете множество сведений о вредоносных программах, использующих функциональность этого раздела. При проведении судебного анализа системы рекомендуется сортировать подразделы в разделе *Notify* по их отметкам времени *LastWrite* и обратить особое внимание на любые записи, время которых приблизительно совпадает с предполагаемым временем инцидента, а также на записи с указанием DLL-файлов в параметре *DLLName*, содержащие информацию о подозрительных версиях файлов или совсем не содержащие информации. (В главе 6 рассматривается тема получения сведений о версии файла из исполняемого файла.)

В разделе WinLogon, указанном выше, есть параметр *TaskMan*, который может представлять интерес для эксперта, так как он позволяет пользователю выбирать приложение, заменяющее диспетчер задач. Этого параметра нет по умолчанию, но его можно добавить. Фактически, при установке программы Process Explorer от компании Sysinternals можно выбрать эту программу в качестве замены стандартного диспетчера задач. Если в разделе WinLogon есть параметр *TaskMan*, следует отнестись к этому факту с подозрением и тщательно исследовать приложение, указанное в этих данных.

В реестре есть интересный раздел, позволяющий пользователю (как правило, разработчику приложения или тому, кто выполняет отладку приложений) указать отладочную программу, которая будет запускаться при выполнении приложения. Этот раздел реестра указан ниже:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
```

В базе знаний Microsoft есть несколько статей, в которых обсуждается использование этого раздела для отладки CGI-приложений (<http://support.microsoft.com/kb/238788>), а также для выключения функции обновления Windows Update в ОС Windows XP (<http://support.microsoft.com/kb/892894>). Создание подраздела для приложения, которые вы хотите, например, заблокировать, и добавление параметра *Debugger* с данными *ntsd --* (значение *ntsd*, после которого следует пробел и два тире) позволит отладчику подключиться к процессу, а затем немедленно выйти. Однако Дана Эпп (Dana Epp) установила (<http://silverstr.ufies.org/blog/archives/000809.html>) способ использования этого раздела в качестве «вектора атаки» или, точнее, метода сохраняемости для вредоносных программ. Чтобы увидеть, как этот способ работает, сначала добавьте в раздел File Execution Options подраздел с именем исполняемого файла, который вы хотите обойти (например, notepad.exe). Введите только имя исполняемого файла, не нужно указывать путь к нему. Затем добавьте в этот раздел строковой параметр с именем *Debugger* и укажите в нем путь к командной строке, как показано на илл. 4.11.



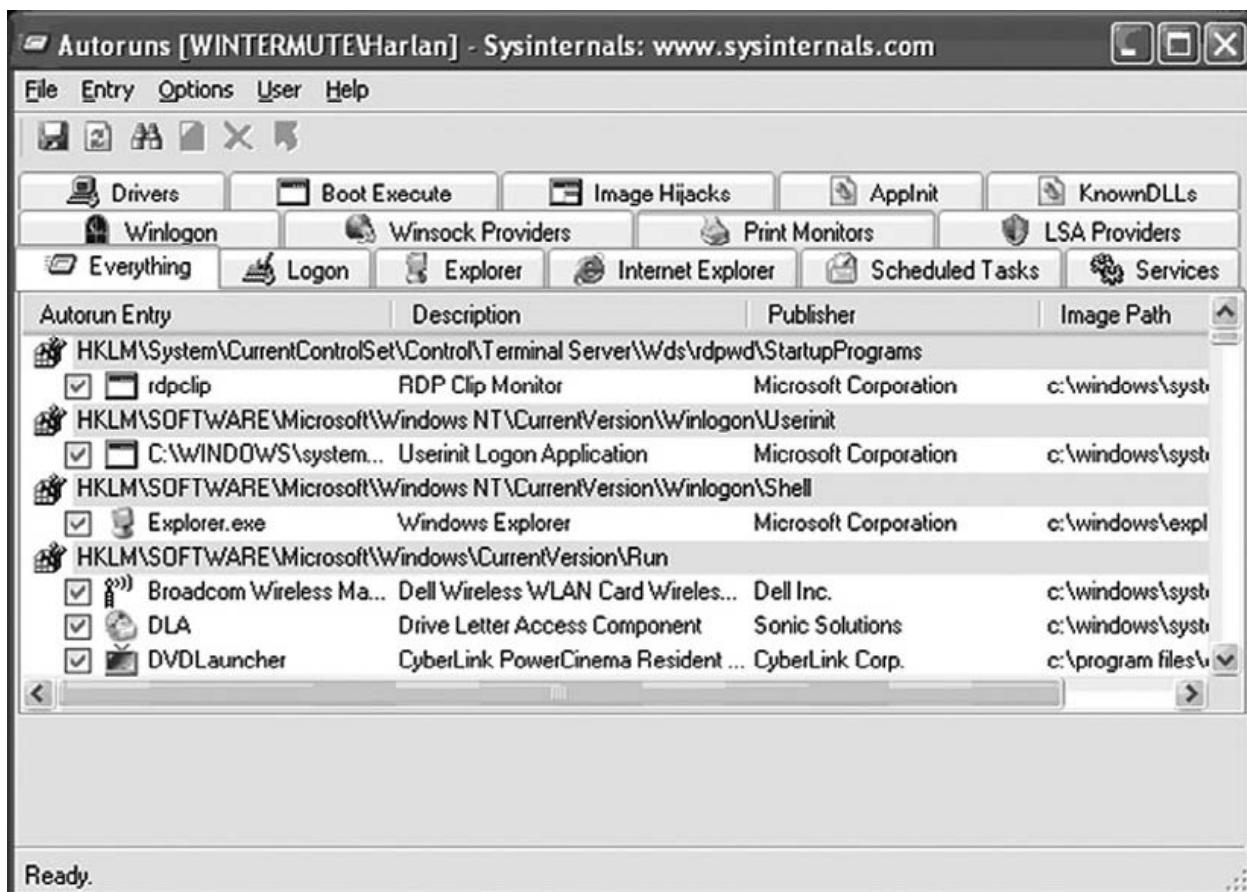
Илл. 4.11. Добавление параметра Debugger в раздел Image File Execution Options.

Нажмите «OK», затем щелкните по кнопке «Пуск» (“Start”), выберите пункт «Выполнить» (“Run”) и введите команду **notepad**. Вы увидите, что вместо программы «Блокнот» (“Notepad”) откроется командная строка. Для того чтобы это изменение вступило в силу, не нужно выполнять перезагрузку компьютера

На данном этапе я могу показать несколько интересных способов использования такого метода, например, указать в параметре *Debugger* зараженный файл «notepad.exe», который не только открывает программу «Блокнот», но также запускает программу удаленного администрирования, чат-бот или какой-нибудь червь. Однако сегодня существует достаточно примеров вредоносных программ, использующих этот раздел реестра в качестве исходной позиции для атаки, чтобы ясно понять, что данный раздел, несомненно, стоит исследовать. Фактически, во время проведения экспертиз, в том числе при расследовании вторжений, я видел, как этот способ сохраняемости использовался вредоносными программами. Просто просмотрите все подразделы и исследуйте исполняемый файл, на который указывает параметр *Debugger* (в некоторых разделах этот параметр может отсутствовать); такой же поиск можно автоматически выполнить в файле куста Software с помощью модуля «imagefile.pl».

Перечисление мест автозапуска в реестре

Один из лучших инструментов, имеющихся сегодня, для извлечения информации из множества мест автозапуска на работающем компьютере – это Autoruns от корпорации Microsoft (версия 9.39 доступна на момент написания этой книги по адресу <http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx>). Это обновленный инструмент, который распространяется как с графическим интерфейсом, так и на основе командной строки. На илл. 4.12 показана версия Autoruns с графическим интерфейсом пользователя.



Илл. 4.12. Autoruns (с графическим интерфейсом).

Как показано на илл. 4.12, Autoruns извлекает записи из многих разделов реестра и выводит результаты на экран. Autoruns также извлекает описание и информацию об издателе из исполняемого файла, на который указывает каждый параметр реестра, и путь к которому отображается в столбце Image Path (показан частично). Эта информация позволяет эксперту быстро определить, запущено ли что-нибудь подозрительное в одном из этих разделов и нужно ли проводить исследование.

Существуют и другие инструменты, перечисляющие содержимое разделов автозапуска в реестре. Один из них – скрипт Silent Runners, написанный на Visual Basic и доступный на сайте www.silentrunners.org. Веб-сайт содержит полный список точек запуска, перечисляемых скриптом, который впервые появился в широком доступе на сайте NTBugTraq (www.ntbugtraq.com) 12 мая 2004 года. Скрипт предназначен для запуска во многих версиях ОС Windows, включая Windows 98, которая не рассматривается в этой книге. Не забывайте об этом при запуске скрипта, так как некоторые перечисленные места автозапуска (в реестре и файловой системе) относятся только к определенным версиям Windows. Эти места указаны в перечне точек запуска.

Для исследования образа данных эксперту необходимы инструменты, позволяющие не только просматривать, но и перечислять разделы реестра, восстановленного из компонентов образа. Функция ProScript в программе ProDiscover позволяет эксперту использовать Perl-скрипты, подобные тем, что написаны для анализа работающих компьютеров, чтобы выполнять поиск в реестре во время исследования образа накопителя. Такие инструменты, как Autoruns, постоянно обновляются и предоставляют полный перечень разделов реестра, связанных с автозапуском.

Функция автоматического запуска носителей

Помимо мест автозапуска в реестре, операционные системы Windows предоставляют возможность автоматического запуска носителей. Чаще всего вы встречаетесь с этой функцией, когда вставляете музыкальный компакт-диск в дисковод:

как только диск начинает вращаться, автоматически запускается соответствующее приложение (например, Windows Media Player или RealPlayer) и воспроизводится первая дорожка компакт-диска. Еще один пример функции автоматического запуска носителей: как только вы вставляете установочный компакт-диск в дисковод, автоматически запускается мастер установки программного обеспечения.

Данная функциональная возможность работает следующим образом: ОС Windows ищет файл «autorun.inf» в корневом каталоге на определенных типах накопителей и, если такой файл присутствует, анализирует и выполняет команды в его строках *load=* или *run=*. Согласно документации Microsoft, эта функция управляется следующим разделом реестра:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\
```

Параметр *NoDriveTypeAutoRun* в этом разделе позволяет отключить функцию автозапуска на отдельных типах накопителей. Каждый бит данного параметра соответствует отдельному типу накопителя, а значение «1» в любом бите отключает функцию автозапуска на накопителе соответствующего типа. Следовательно, значение 0x95 (149 в десятичной системе счисления) отключает функцию автозапуска на съемных и сетевых накопителях, а значение 0xFF – на накопителях всех типов.

Предупреждение

Параметр *NoDriveTypeAutoRun* можно найти в кусте HKLM или HKCU. Согласно документации Microsoft, если этот параметр есть в кусте HKLM, то параметр в кусте HKCU игнорируется.

Одннадцатого сентября 2008 года корпорация Microsoft опубликовала статью № 953252 (<http://support.microsoft.com/kb/953252>), в которой говорится, что для корректной работы параметра *NoDriveTypeAutoRun* в Windows 2000, XP, 2003 и Vista (не касается Windows 2008) пользователи должны установить дополнение, добавляющее параметр *HonorAutorunSetting* в раздел куста HKLM и устанавливающее его в значение 0x1. Статья содержит полную интерпретацию параметров *NoDriveTypeAutoRun* и *HonorAutorunSetting*. Этот вопрос был актуальным в то время, когда статья была опубликована, так как версии вируса Sality (а затем и Conficker) распространялись частично за счет того, что записывали вредоносную программу и файл «autorun.inf», ссылающийся на эту вредоносную программу, в корневой каталог каждого накопителя, имеющегося в зараженной системе. Фактически было обнаружено, что некоторые версии вируса Sality заражали съемные накопители, которые подключались к системе после того, как она была первоначально заражена.

Инструменты и ловушки...

Приводы компакт-дисков

Кроме изменения параметра *NoDriveTypeAutoRun*, функцию автоматического запуска для приводов компакт-дисков можно включить, установив параметр *Autorun* в разделе HKLM\System\CurrentControlSet\Services\CDRom в значение 1 (чтобы отключить функцию, установите этот параметр в значение 0). Согласно статье № 319287 из базы знаний Microsoft (<http://support.microsoft.com/kb/319287>), настоящее назначение параметра *Autorun* – включать или отключать уведомления о смене носителей. Функция автозапуска также будет отключена для компакт-дисков, если параметр *NoDriveTypeAutoRun* установлен в значение 0xb5. Подробности об этих настройках, а также дополнительную информацию можно найти в статье № 330135 из базы знаний Microsoft (<http://support.microsoft.com/kb/330135>).

Параметр NtfsDisableLastAccessUpdate

Многие эксперты знают, что ОС Windows, как и многие другие системы, сохраняет в файле отметки времени (время последнего изменения, доступа и создания); эта тема будет подробно рассмотрена в главе 5. Но немногие знают, что, используя один параметр реестра, ОС Windows можно настроить так, чтобы она не обновляла в файле дату последнего обращения к этому файлу. Параметр *NtfsDisableLastAccessUpdate* можно использовать, чтобы отключить эту функцию для увеличения быстродействия высокопроизводительных серверов, как описано в статье № 894372 из базы знаний Microsoft (<http://support.microsoft.com/kb/894372>). Однако этот параметр включен по умолчанию в ОС Windows Vista и Windows 7, что видно в показанных ниже результатах модуля «disablelastaccess.pl» программы RegRipper:

```
C:\Perl\forensics\rr>rip.pl -r d:\cases\vista\system -p disablelastaccess
Launching disablelastaccess v.20090118
DisableLastAccess
ControlSet001\Control\FileSystem
NtfsDisableLastAccessUpdate = 1

C:\Perl\forensics\rr>rip.pl -r d:\cases\win7\system -p disablelastaccess
Launching disablelastaccess v.20090118
NtfsDisableLastAccessUpdate
ControlSet001\Control\FileSystem
NtfsDisableLastAccessUpdate = 1
```

Эти данные могут быть очень важны, особенно при проведении анализа ОС Windows Vista и Windows 7, так как отсутствие отметок о времени последнего доступа к файлу может стать значительной проблемой во время экспертизы. Этот параметр не включен по умолчанию в системах Windows XP и 2003, поэтому его наличие (а также значение 1 в нем) может сказать о многом во время экспертизы. Даже если этот параметр включен, в реестре, как вы уже поняли, существует много мест, где можно найти сведения о доступе к файлам и типе этого доступа.

Параметр NukeonDelete

Когда пользователи удаляют файлы в ОС Windows XP или 2003, проводник по умолчанию отправляет эти файлы в корзину. Пользователи могут удалить файлы в обход корзины, нажав клавишу Shift при удалении или добавив параметр *NukeonDelete* в указанном ниже разделе реестра в файле куста Software:

```
Microsoft\Windows\CurrentVersion\Explorer\BitBucket
```

После добавления параметра *NukeonDelete* и установления его в значение 1, файлы будут удаляться в обход корзины. Следовательно, если во время экспертизы вы обнаружили, что в корзине нет файлов, проверьте реестр на предмет наличия этого параметра и его значения 1.

Как упоминалось выше, несмотря на то, что двоичная структура реестра, начиная с Windows NT до Windows 7, остается прежней, разделы и параметры меняются в зависимости от версии операционной системы. Например, только что обсуждавшееся местонахождение параметра *NukeonDelete* характерно для ОС Windows XP и 2003; оно изменилось с появлением Windows Vista. В ОС Windows XP и 2003 раздел BitBucket находится в файле куста Software, поэтому его параметры применяются ко всем пользователям в системе. В ОС Vista и Windows 7 раздел BitBucket был перемещен в пользовательский файл куста (NTUSER.DAT) и находится по адресу:

Software\Microsoft\Windows\CurrentVersion\Explorer\BitBucket\Volume\{GUID}

Операционные системы Windows XP и 2003 применяют параметр *NukeOnDelete* для всех пользователей в системе, а Windows Vista и Windows 7 предоставляют эту функцию отдельно для каждого пользователя и каждого тома.

Съемные USB-носители

Одна из самых популярных тем, освещаемых мной на различных конференциях, связана со способами отслеживания съемных USB-накопителей в операционных системах Windows. Когда я впервые рассматривал эту тему, многие слушатели даже не представляли себе, что такое отслеживание возможно. С тех пор я ответил на множество вопросов, касающихся этой темы.

Когда съемное USB-устройство, например, флеш-накопитель, подключается к ОС Windows, в системном реестре остаются следы или артефакты. (Кроме того, артефакты остаются в файле «setupapi.log».) Когда устройство подключается к компьютеру, диспетчер PnP (Plug and Play) получает уведомление об этом событии и запрашивает в дескрипторе устройства информацию о нем, например, сведения о производителе; эта информация находится во встроенным программном обеспечении, а не в области памяти устройства. Затем диспетчер PnP использует эту информацию, чтобы найти подходящий драйвер для устройства (исходя из содержимого inf-файлов) и, при необходимости, загружает этот драйвер. (Эта информация записывается в файл «setupapi.log».) Как только устройство будет распознано, происходят изменения в вышеупомянутом разделе реестра:

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\USBSTOR

В этом разделе создаются подразделы, которые выглядят следующим образом:

Disk&Ven_###&Prod_###&Rev_###

Такой подраздел представляет идентификатор класса устройства, так как он определяет отдельный класс устройства. Поля ### заполняются диспетчером PnP на основе информации, найденной в дескрипторе устройства. Например, у меня есть флеш-накопитель Geek Squad емкостью 1 Гб, который я купил в магазине Best Buy; идентификатор класса для этого устройства выглядит так:

Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15

Для просмотра содержимого дескриптора устройства можно использовать утилиту UVCView. На илл. 4.13 показана часть информации из дескриптора устройства для упомянутого выше флеш-накопителя Geek Squad.

iManufacturer:	0x01
English (United States)	"Best Buy"
iProduct:	0x02
English (United States)	"Geek Squad U3"
iSerialNumber:	0x03
English (United States)	"0C90195032E36889"

Илл. 4.13. Часть данных дескриптора устройства в утилите UVCView.

Как видно на илл. 4.13, информация о производителе (iManufacturer) и продукте (iProduct) сопоставляется с идентификатором класса устройства.

Примечание

Раньше утилиту UVCView можно было бесплатно загрузить с сайта Microsoft, но теперь она включена в набор драйверов Windows (Windows Driver Kit, WDK), согласно <http://msdn.microsoft.com/en-us/library/aa469207.aspx>. UVCView – утилита, позволяющая эксперту просматривать информацию дескриптора USB-устройства; эта информация не содержится в области памяти флеш-накопителя USB, и следовательно ее нельзя получить при создании образа такого устройства.

После того как создан идентификатор класса устройства, должен быть создан уникальный идентификатор экземпляра для этого отдельного устройства. Обратите внимание, что на илл. 4.13 есть значение, которое называется *iSerialNumber*. Это уникальный идентификатор экземпляра для устройства, похожий на MAC-адрес сетевой интерфейсной платы. Это значение используется как уникальный идентификатор экземпляра для устройства, чтобы можно было однозначно определить несколько устройств одного класса (например, два флеш-накопителя Geek Squad емкостью 1 ГБ) в системе. Из статьи «USB FAQ – Intermediate» (www.microsoft.com/whdc/connect/usb/usbfaq_intermed.mspx):

«Если у устройства есть серийный номер, Microsoft требует, чтобы этот серийный номер однозначно определял каждый экземпляр одного и того же устройства. Например, если два дескриптора устройства имеют одинаковые значение для полей *idVendor*, *idProduct* и *bcdDevice*, то поле *iSerialNumber* будет отличать одно устройство от другого».

На илл. 4.14 показаны идентификатор класса устройства и подчиненный ему уникальный идентификатор экземпляра в редакторе реестра.



Илл. 4.14. Часть данных редактора реестра, в которых показаны идентификатор класса устройства и уникальный идентификатор экземпляра.

Если производитель хочет, чтобы его устройство отвечало требованиям программы Windows Logo, оно должно иметь серийный номер (www.microsoft.com/whdc/archive/usbfaq.mspx#ERCAC). Тем не менее, не все устройства содержат такой номер. Для устройств без серийного номера диспетчер PnP создает уникальный идентификатор экземпляра, похожий на следующую строку:

6&26c97b61&0

Обратите внимание, что второй символ – амперсанд (&). Если в разделе USBSTOR есть уникальный идентификатор экземпляра, который выглядит таким образом, то устройство, подключенное к системе, не содержит серийного номера в дескрипторе устройства.

Значит, если второй символ в уникальном идентификаторе экземпляра – не знак &, то можно определить уникальное устройство, которое было подключено к системе. В

делах, связанных с несколькими системами и накопителями, экспертам следует использовать утилиту UVCView, чтобы позднее устройства можно было связать с системой не только посредством артефактов реестра в разделе USBSTOR, но и посредством записей в разделе MountedDevices, а также в ярлыках и других ссылках на файлы, находящиеся в системе.

После того как создан раздел для уникального идентификатора экземпляра, он заполняется различными параметрами, в том числе *FriendlyName*. Параметр, представляющий интерес для экспертов, – *ParentIdPrefix*. Microsoft не предоставляет информацию о том, как создается этот параметр, или является ли он уникальным в системах Windows. Однако параметр *ParentIdPrefix* можно использовать, чтобы установить взаимосвязь с дополнительной информацией из реестра.

Например, используя уникальный идентификатор экземпляра и параметр *ParentIdPrefix*, можно определить, когда USB-устройство последний раз подключалось к ОС Windows. В работающем компьютере перейдите к следующему разделу:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\DeviceClasses
```

Здесь вы увидите ряд подразделов, имена которых представляют собой глобально уникальные идентификаторы (GUID). Отдельные классы устройств, представляющие для нас интерес, – это {53f56307-b6bf-11d0-94f2-00a0c91efb8b} и {53f5630d-b6bf-11d0-94f2-00a0c91efb8b}. Эти два класса определены в заголовочном файле «ntddstor.h» (пример заголовочного файла можно увидеть на странице www.reactos.org/generated/doxygen/d1/d09/ntddstor_8h.html) как идентификаторы GUID для интерфейсов дискового устройства и устройства тома соответственно. Перейдя к идентификатору дискового устройства, мы увидим ряд подразделов с длинными именами; для устройства, показанного на илл. 4.13, мы увидим подраздел с таким именем:

```
USBSTOR#Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15#0C90195032E36889&0
#{53f56307-b6bf-11d0-94f2-00a0c91efb8b}
```

В этом примере я выделил уникальный идентификатор экземпляра (в данном случае это серийный номер устройства), чтобы продемонстрировать, где находится этот идентификатор в имени раздела. Время *LastWrite* этого раздела соответствует времени, когда дисковое устройство подключалось к системе последний раз. Это обусловлено тем, что при подключении устройства к ОС Windows создается подраздел с именем Control; после отключения устройства этот подраздел удаляется. Оба эти действия приводят к тому, что время *LastWrite* раздела DeviceClasses для отдельного устройства изменяется.

Мы также можем провести такое же сопоставление в случае с идентификатором GUID устройства тома, используя параметр *ParentIdPrefix* для устройства, как показано ниже

```
##?#STORAGE#RemovableMedia#7&326659cd&0&RM#{53f5630d-b6bf-11d0-94f2-
00a0c91efb8b}
```

Здесь я снова выделил значение параметра *ParentIdPrefix*, чтобы показать, где оно находится в имени подраздела устройства. Время *LastWrite* этого раздела соответствует времени, когда устройство тома подключалось к системе последний раз. Как и в случае с идентификатором GUID дискового устройства, в разделе с именем идентификатора устройства тома создается подраздел Control при подключении тома; этот подраздел удаляется после отключения тома (т. е. устройства), что приводит к соответствующему изменению времени *LastWrite* в разделе.

Позднее в этой главе, в разделе «Монтированные устройства», мы подробнее рассмотрим тему использования значения *ParentIdPrefix* для сопоставления информации из реестра.

В состав RegRipper входит несколько подключаемых модулей для извлечения данных о съемных USB-носителях из файла куста System. Первый модуль, который называется «usbstor.pl», анализирует содержимое раздела USBSTOR и перечисляет все разделы классов устройств и все уникальные экземпляры устройств в этих разделах, как показано ниже:

```
CdRom&Ven_SanDisk&Prod_U3_Cruzer_Micro&Rev_3.27 [Fri Aug 29 00:36:11 2008]
S/N: 0000161511737EFB&1 [Fri Aug 29 00:36:11 2008]
  FriendlyName : SanDisk U3 Cruzer Micro USB Device
  Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15 [Fri Aug 29 00:36:11 2008]
  S/N: 0C90195032E36889&0 [Fri Dec 5 14:02:08 2008]
    FriendlyName : Best Buy Geek Squad U3 USB Device
    ParentIdPrefix: 7&326659cd&0
Disk&Ven_Generic&Prod_STORAGE_DEVICE&Rev_0026 [Fri Aug 29 00:36:11 2008]
S/N: 0000082509&0 [Fri Aug 29 00:36:11 2008]
  FriendlyName : Generic STORAGE DEVICE USB Device
  ParentIdPrefix: 7&2a5b4c66&0
Disk&Ven_Hitachi&Prod_Easy_Device&Rev_ [Fri Aug 29 00:36:11 2008]
S/N: 200718900AEA&0 [Fri Aug 29 00:36:11 2008]
  FriendlyName : Hitachi Easy Device USB Device
  S/N: 200718900CDC&0 [Fri Aug 29 00:36:11 2008]
    FriendlyName : Hitachi Easy Device USB Device
Disk&Ven_Maxtor&Prod_OneTouch&Rev_0121 [Fri Aug 29 00:36:11 2008]
S/N: 2HAA06KR____&0 [Fri Aug 29 00:36:11 2008]
  FriendlyName : Maxtor OneTouch USB Device
  S/N: 2HAA07XG____&0 [Fri Aug 29 00:36:11 2008]
    FriendlyName : Maxtor OneTouch USB Device
```

В предыдущем фрагменте выходных данных модуля «usbstor.pl» мы видим несколько перечисленных устройств, первое из которых – это устройство U3 (мы подробно рассмотрим эти устройства в следующем разделе). Мы также видим в списке два устройства Maxtor OneTouch и можем предположить, что, так как ни в одном из них не указано значение *ParentIdPrefix*, это – внешние НЖМД, а не флеш-накопители (что на самом деле так и есть). Модуль «devclass.pl» извлекает описанные выше данные из раздела DeviceClasses для дисковых устройств и устройств тома; ниже показан фрагмент выходных данных для раздела, имя которого представляет собой идентификатор дискового устройства:

```
Wed Dec 17 13:18:41 2008 (UTC)
Disk&Ven_SanDisk&Prod_U3_Cruzer_Micro&Rev_3.27,0000161511737EFB&0
Wed Dec 17 13:16:36 2008 (UTC)
Disk&Ven_USB_2.0&Prod_USB_Flash_Drive&Rev_0.00,3de63180700745&0
Fri Dec 5 14:02:11 2008 (UTC)
Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15,0C90195032E36889&0
```

Обратите внимание, что в выходных данных модуля «devclass.pl» различные устройства отсортированы в списке по отметкам времени *LastWrite* соответствующих разделов. Как уже говорилось, когда устройство подключается к системе Windows, к разделам этого устройства в разделах DeviceClasses добавляется подраздел Control, а при отключении устройства этот подраздел удаляется. Эти действия приводят к тому, что время *LastWrite* для самого раздела устройства изменяется.

Еще один модуль, созданный мной, называется «usbstor2.pl». Как и другие модули, его можно запускать в программе RegRipper, но он предназначен для работы в «grpl.pl» как часть пакетного файла. Ко мне обратился один эксперт, который сказал, что создал образы

30 систем и ему нужно сопоставить данные о съемных USB-накопителях во всех 30 системах, чтобы найти особенности использования этих устройств. Для того чтобы облегчить его задачу, я написал модуль «usbstor2.pl», который работает как модуль «usbstor.pl», но также извлекает дополнительную информацию и предоставляет ее в формате значений, разделенных запятыми (.csv). Вот часть выходных данных модуля «usbstor2.pl»:

```
WINTERMUTE,CdRom&Ven_SanDisk&Prod_U3_Cruzer_Micro&Rev_3.27,0000161511737EFB&1,1219970171,SanDisk U3 Cruzer Micro USB Device
WINTERMUTE,Disk&Ven_&Prod_USB_DISK&Rev_1.13,0738015025AC&0,1219970171,USB DISK USB Device,7&2713a8a1&0
WINTERMUTE,Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15,0C90195032E36889&0,1228485728,Best Buy Geek Squad U3 USB Device,7&326659cd&0
```

Как видите, в выходных данных содержится имя системы, идентификатор класса устройства, уникальный идентификатор экземпляра (или серийный номер), отметка времени *LastWrite* для раздела в виде 32-разрдного значения в формате времени UNIX и понятное имя устройства. Последнее значение – это *ParentIdPrefix*, если таковое присутствует. Итак, я предложил своему другу запустить модуль «usbstor2.pl» в утилите «rip.pl» и выводить каждое повторение команды в отдельный файл, используя перенаправление:

```
C:\Perl>rip.pl -p usbstor2.pl -r path\system >> devices.csv
```

После того как модуль обработал все файлы кустов System, мой друг смог открыть файл «devices.csv» в программе Excel и сортировать данные по различным устройствам, именам систем или времени. Если бы он работал с программами просмотра реестра, даже с теми, что встроены в судебные приложения для анализа, у него бы ушло несколько дней, чтобы закончить то, что он смог сделать в «rip.pl» за несколько минут.

Проблемы, связанные с USB-устройствами

Всем (особенно специалистам по информационной безопасности) известно, что съемные USB-накопители представляют угрозу для безопасности, в частности в корпоративной инфраструктуре. С тех пор, как появились первые накопители на гибких магнитных дисках (тогда эти штуки были действительно на гибких дисках!), емкость запоминающих устройств увеличилась, а их размер («форм-фактор») уменьшился. Сегодня, когда я пишу эту книгу, в продаже имеется множество флеш-накопителей емкостью от двух до четырех гигабайт по доступным ценам. С помощью таких устройств можно запросто украсть из организации файл или целую базу данных. Особенностью флеш-накопителей являются их небольшие размеры: сняв корпус устройства и оставив одну микросхему, вы получите накопитель размером буквально с ноготь большого пальца.

Еще хуже то, что эти устройства сильно распространены. Раньше любой, у кого был флеш-накопитель емкостью 64 Мб, считался своего рода суперхакером. Сегодня кто угодно может приобрести такие устройства и использовать их для хранения фотографий, презентаций и других данных. То же относится к устройствам iPod и MP3-плеерам. Мы видим их в спортзале, в офисе, в автобусе – везде. По сути, мы привыкли к ним, поэтому, когда мы видим на столе такое устройство, подключенное к ноутбуку, нам совсем не кажется это необычным. Прямо сейчас можно купить плеер iPod Nano примерно за 200 долларов США. Если пользователь присоединит подобное устройство к ноутбуку, подключенному к локальной сети, кто знает, слушает ли он музыку или загружает на накопитель финансовые прогнозы, схемы оплаты труда, заявки на получение подряда и другую конфиденциальную информацию.

Другая проблема имеет отношение к функции автозапуска, упоминавшейся ранее в главе. Когда CD- или DVD-диск вставляется в дисковод компьютера с ОС Windows, происходит обнаружение нового носителя. Если в корневой папке носителя есть файл «autorun.inf», этот файл анализируется, и выполняются строки *run=* и *load=*. Эта возможность является частью расширенного взаимодействия с пользователем, предлагаемого системой Windows. По умолчанию функция автозапуска отключена для съемных запоминающих устройств, например, для флеш-накопителей; однако она по умолчанию включена для носителей, распознаваемых системой как диски CD-ROM, на которые, из-за способа форматирования, частично похожи устройства UЗ. В Интернете доступны инструкции, где объясняется, как создавать собственные данные и программы, которые будут записаны поверх существующего раздела UЗ, что делает такие устройства чрезвычайно опасным механизмом получения доступа к системе.

Компания UЗ предоставляет служебную программу, обеспечивающую пользователям большую мобильность для их приложений. Эта программа создает небольшой раздел в начале флеш-накопителя и обозначает его как раздел CDFS (файловая система компакт-диска), чтобы ОС Windows распознавала этот раздел как компакт-диск, а не как съемное запоминающее устройство. Затем различные приложения запускаются из этого раздела CDFS, а остальная часть накопителя используется для хранения данных. Но это означает, что, даже если функция автозапуска отключена (по умолчанию) для съемных накопителей, она *включена* (по умолчанию) для раздела CDFS.

Когда я подключил флеш-накопитель с разделом UЗ к компьютеру с ОС Windows XP, то обнаружил, что для одного и того же устройств были созданы две записи идентификатора класса устройства:

CdRom&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15

и

Disk&Ven_Best_Buy&Prod_Geek_Squad_U3&Rev_6.15

Оба эти идентификатора класса устройства содержали подраздел с одинаковым уникальным идентификатором экземпляра. При проведении судебного анализа образа данных необходимо, несомненно, выполнять поиск таких данных, так как они могут определить вектор заражения или способ взлома. Злоумышленник может создать специальный ISO-образ, который будет установлен в раздел CDFS, а затем удалить все признаки служебных программ или логотип UЗ в устройстве. Если кто-нибудь вставит такой накопитель в компьютер, будет активирована функция автозапуска для раздела CDFS, и все, что задумал злоумышленник (установка троянских программ, сбор информации или паролей из защищенного хранилища), будет выполнено автоматически.

Вставка «USBdump» содержит дополнительную информацию об угрозах, представляемых съемными накопителями. Несмотря на то, что эти угрозы не связаны непосредственно с анализом реестра, они ставят трудные задачи перед специалистами по информационной безопасности.

Инструменты и ловушки...

USBdump

Утилита USBdump создает угрозу нарушения информационной безопасности при работе со съемными USB-накопителями (http://wiki.hak5.org/wiki/USB_Hacksaw). USBdump устанавливается в ОС Windows и, когда съемный USB флеш-накопитель подключается к компьютеру, незаметно копирует содержимое этого устройства. Кроме того, поговаривают, что существует программа, которая незаметно создает образ флеш-накопителя, когда он подключается к системе, чтобы можно было найти не только

активные, но также и удаленные файлы. Обе эти проблемы упоминались в блоге Брюса Шнайера (Bruce Schneier) «Schneier on Security» 25 августа 2006 года.

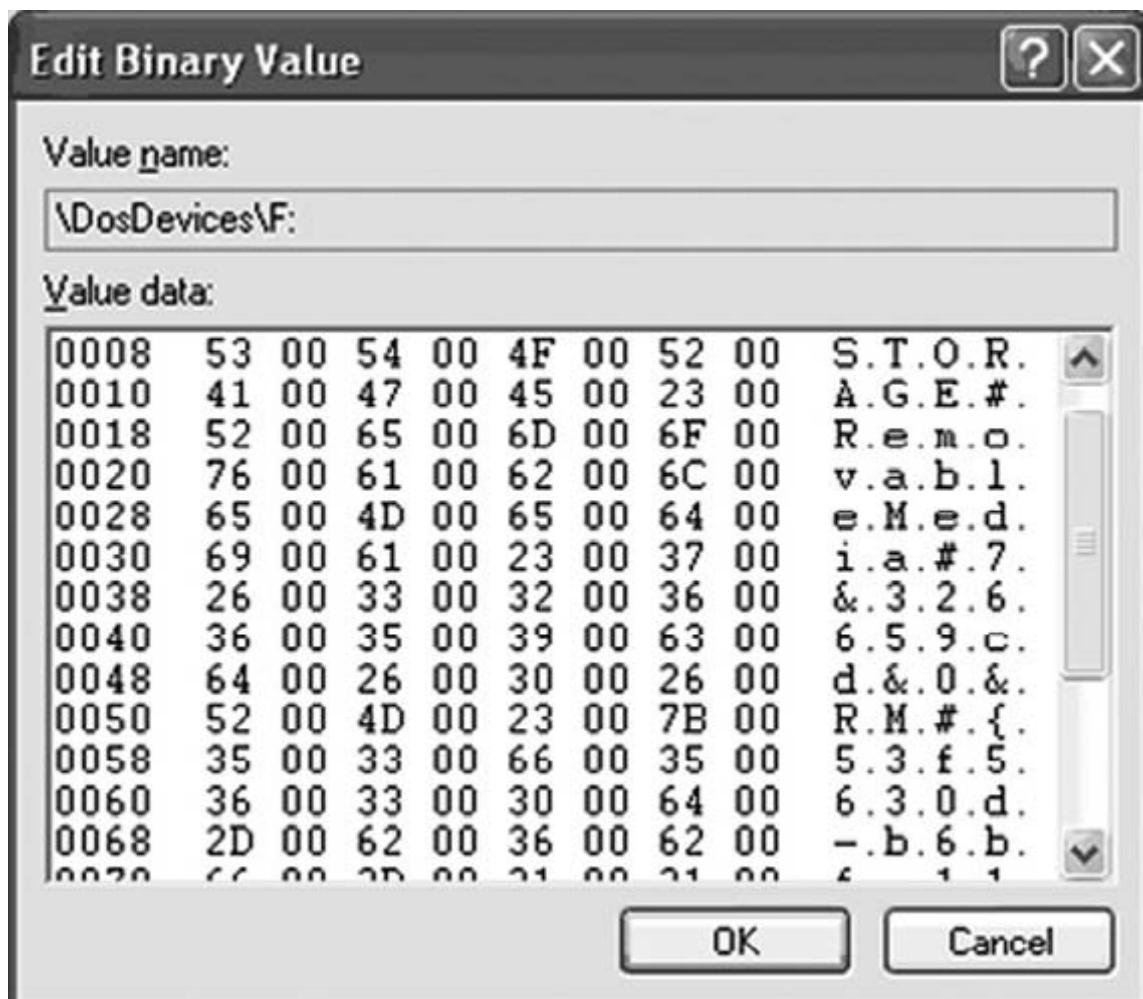
Монтированные устройства

В разделе MountedDevices хранится информация о различных устройствах и томах, монтируемых в файловой системе NTFS: Полный путь к разделу выглядит так:

```
HKEY_LOCAL_MACHINE\System\MountedDevices
```

Например, когда съемный USB-накопитель подключается к системе Windows, ему назначается буква накопителя; эта буква отображается в разделе MountedDevices. Если устройству назначена буква F:\, параметр в разделе MountedDevices будет показан как \DOSDevices\F:. Мы можем сопоставить запись из раздела USBSTOR с разделом MountedDevices, используя параметр *ParentIdPrefix*, который находится в разделе уникального идентификатора экземпляра для устройства. Параметр *ParentIdPrefix* для USB-устройства, обсуждавшегося в предыдущем разделе, имеет данные 7&326659cd&0. Обратите внимание, что это не уникальный идентификатор экземпляра, а следовательно – не серийный номер, о котором мы говорили ранее.

После того как мы получили значение из параметра *ParentIdPrefix*, мы находим букву накопителя, назначенную этому устройству, посредством определения местонахождения в разделе MountedDevices записи *DOSDevices*, которая содержит в своих данных значение *ParentIdPrefix*. На работающем компьютере мы можем сделать это, выполнив щелчок правой кнопкой мыши по каждому параметру реестра и выбрав пункт «Изменить» (“Modify”); когда откроется диалоговое окно «Изменение двоичного параметра» (“Edit Binary Value”), мы можем просмотреть содержимое данных на предмет наличия значения параметра *ParentIdPrefix*. Параметр *ParentIdPrefix* хранится в реестре в виде строки, а параметры *DOSDevices* в разделе реестра MountedDevices – в виде двоичных данных, поэтому необходимо выполнить преобразование. На илл. 4.15 показано диалоговое окно «Изменение двоичного параметра» (“Edit Binary Value”) для параметра \DOSDevices\F:.



Илл. 4.15. Данные параметра \DosDevices\F: из раздела MountedDevices.

На илл. 4.15 мы четко видим значение *ParentIdPrefix* – 7&326659cd&0. Используя значение *ParentIdPrefix*, чтобы установить соответствие между разделами реестра USBSTOR и MountedDevices, мы можем найти букву накопителя, назначенную устройству. Продолжая судебный анализ образа данных, мы можем найти ссылки, указывающие на накопитель F:\, в других местах реестра или в файлах ярлыков. Затем мы можем сопоставить отметки времени *LastWrite* раздела уникального идентификатора экземпляра, раздела MountedDevices и отметки времени последнего изменения, доступа и создания в файлах, чтобы составить временную шкалу.

Предупреждение

Устанавливая взаимосвязь между разделами USBSTOR и MountedDevices с помощью значения *ParentIdPrefix* устройства, имейте в виду, что несколько устройств, возможно, подключались к системе одно за другим, и им назначалась одна и та же буква накопителя. У меня несколько USB флеш-накопителей разных производителей и разных емкостей, и каждый раз, когда я по одному подключал эти устройства к ноутбуку, они все сопоставлялись с накопителем F:\. Устанавливая такую взаимосвязь, не нужно забывать об этом, особенно во время анализа ОС Windows XP или 2003. Далее в этой главе вы увидите, что Windows Vista и Windows 7 также имеют свои особенности в вопросах, касающихся сопоставления устройства с буквой накопителя.

Изучая раздел MountedDevices, вы, возможно, заметили, что здесь есть записи \DosDevices (в частности, \DosDevices\C:), длина данных которых составляет только 12 байт (три двойных слова). На илл. 4.16 показан пример таких данных.

\DosDevices\С:	REG_...	16 23 ab 41 00 7e 00 00 00 00 00 00
\DosDevices\D:	REG_...	16 23 ab 41 00 5e c6 52 07 00 00 00

Илл. 4.16. Данные параметров \DosDevices\С: и \DosDevices\D: из раздела MountedDevices.

Двоичные параметры, показанные на илл. 4.16, состоят из сигнатуры накопителя (также известной как идентификатор тома) для НЖМД (первое двойное слово) и смещения раздела (второе и третье двойные слова). Сигнатура накопителя находится по смещению 0x1b8 (440 – в десятичной системе счисления) в главной загрузочной записи НЖМД.

Для параметра \DosDevices\С:, показанного на илл. 4.16, второе и третье двойные слова вместе преобразовываются в шестнадцатеричное значение 0x7e00, которое равно 32256 в десятичной системе счисления. Размер каждого сектора на НЖМД равен 512 байт; $32256/512 = 63$, поэтому раздел С:\ начинается в секторе 63.

Посмотрев на параметр \DosDevices\D: на илл. 4.16, мы увидим, что двоичные данные указывают, что накопитель D:\ имеет такую же сигнатуру, что накопитель С:\, но другое смещение раздела. Это объясняется тем, что накопитель D:\ представляет собой другой раздел на том же физическом диске, что и накопитель С:\. Используя информацию о смещениях разделов, можно вычислить, что размер раздела С:\ чуть больше 29 Гб.

Чтобы продемонстрировать, как найти эту информацию во время судебной экспертизы, я запустил ProDiscover и открыл пробное дело. В приложении Registry Viewer я нашел запись \DosDevices\С: в разделе MountedDevices и увидел, что сигнатура накопителя – 5D EC 5D EC в шестнадцатеричном формате. Затем я переключился в режим просмотра кластеров в деле (которое начинается с кластера 0). На илл. 4.17 вы видите фрагмент данных из ProDiscover, в которых показана часть главной загрузочной записи.

000000180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000190	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001A0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001B0	00 00 00 00 00 2C 44 63	5D EC 5D EC 00 00 80 01,0c]ij...e.
0000001C0	01 00 07 FE BF 4F 3F 00	00 00 11 1E 91 00 00 00	...b?0?....'
0000001D0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001E0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000001F0	00 00 00 00 00 00 00 00 00 00 00 00 55 AAua

Илл. 4.17. Режим просмотра кластеров в ProDiscover; показана сигнатурна накопителя (5D EC 5D EC).

Как показано на илл. 4.17, сигнатурна накопителя, которую мы нашли в разделе MountedDevices, видна в смещении 0x1B8.

Почему это так важно? Ведь если мы уже создали образ НЖМД, то должны получить и уметь проверить сигнатурну накопителя как из реестра, так и из главной загрузочной записи. Однако эта информация становится полезной, когда с делом связаны внешние НЖМД. В своей повседневной работе в качестве специалиста по расследованию инцидентов и судебного эксперта я использую внешние накопители для хранения данных клиента, образов накопителей и файлов журналов. Таким образом, после того, как дело будет закончено, а итоговый отчет принят, я могу легко очистить внешний накопитель. Если бы я сохранял все эти данные на своем ноутбуке, мне бы пришлось очищать его накопитель, а также переустанавливать операционную систему со всеми приложениями и восстанавливать различные данные, в том числе PGP-ключи. Кроме того, когда я пишу эту книгу, все файлы, связанные с ней, сохраняются на НЖМД Western Digital емкостью 120 Гб, который подключен через USB. Этот накопитель отображается в разделе USBSTOR со следующим идентификатором класса устройства:

Disk&Ven_WDC_WD12&Prod_00UE-00KVT0&Rev_0000

Как и следовало ожидать, у накопителя есть серийный номер (который я могу проверить с помощью UVCView), что указано в уникальном идентификаторе экземпляра,

отображаемом в реестре. Однако в разделе реестра для уникального идентификатора экземпляра *нет параметра с именем ParentIdPrefix*. Когда это устройство подключается к моему ноутбуку, оно отображается как накопитель G:\, а соответствующая информация о нем, в том числе сигнатура накопителя, хранится в разделе MountedDevices. Если бы этот внешний НЖМД имел отношение к делу, которое я расследую, я бы мог связать данное устройство с системой, даже если для него нет параметра *ParentIdPrefix*.

Параметры в разделе MountedDevices, которые начинаются с «\??\Volume», можно связать с отдельным пользователем, перейдя к упомянутому ниже разделу в пользовательском файле «NTUSER.DAT»:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
```

В этом разделе есть несколько подразделов, но на данном этапе нас интересуют те, имена которых похожи на глобально уникальные идентификаторы, например, {d37df1d3-265b-11db-a216-0015c51b9712}. В разделе MountedDevices, как правило, также находится соответствующий параметр «\??\Volume {d37df1d3-265b-11db-a216-0015c51b9712}». Модуль «mp2.pl» программы RegRipper анализирует подразделы в пользовательском разделе MountPoints2 и перечисляет записи по типу точки монтирования, как показано ниже (добавлено выделение):

```
C:\Perl\forensics\rr>rip.pl -p mp2 -r d:\cases\NTUSER.DAT
Launching mp2 v.20090115
MountPoints2
Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
LastWrite Time Mon Sep 26 23:32:34 2005 (UTC)
Drives:
  A Mon Sep 26 22:50:40 2005 (UTC)
  D Mon Sep 26 22:50:40 2005 (UTC)
  C Mon Sep 26 22:50:40 2005 (UTC)
Remote Drives:
  ##192.168.1.22#c$ Mon Sep 26 23:26:00 2005 (UTC)
  ##192.168.1.71#c$ Mon Sep 26 23:32:34 2005 (UTC)
Volumes:
  {2490fb15-f08b-11d8-958e-806d6172696f} Mon Sep 26 22:50:58 2005 (UTC)
  {2490fb13-f08b-11d8-958e-806d6172696f} Mon Sep 26 23:37:22 2005 (UTC)
  {082b2da3-6b36-11d9-95d0-000c29960ded} Mon Sep 26 23:14:02 2005 (UTC)
  {2490fb12-f08b-11d8-958e-806d6172696f} Mon Sep 26 22:50:58 2005 (UTC)
Analysis Tip: Correlate the Volume entries to those found in the
MountedDevices entries that begin with "\??\Volume".
```

Глобально уникальные идентификаторы, перечисленные под строкой «Volumes:», можно затем сопоставить с параметрами, указанными в разделе реестра MountedDevices. Например, модуль «mountdev2.pl» программы RegRipper был запущен для анализа файла куста System (извлеченного из того же образа системы, что и файл «NTUSER.DAT», использованный ранее) и показал следующую информацию (добавлено выделение):

```
Device: \?\?\STORAGE#RemovableMedia#7&2713a8a1&0&RM#\{53f5630d-b6bf-11d0
-94f2-00a0c91efb8b}
\?\?\Volume\{082b2da3-6b36-11d9-95d0-000c29960ded}
\DosDevices\E:
```

В данном случае съемное запоминающее устройство с указанным значением *ParentIdPrefix* было монтируено как накопитель E:\ и было доступно пользователю, из профиля которого был извлечен файл «NTUSER.DAT». На первый взгляд, это логично в системах, которые используются преимущественно одним пользователем, однако в системах с несколькими профилями пользователей эксперту важно определить, кто из пользователей имел доступ к съемному накопителю.

Инструменты и ловушки...

Несанкционированное копирование данных

Мне задают много вопросов, касающихся несанкционированного копирования данных. В большинстве случаев эксперты, исследующие систему или образ данных, хотят знать, какие данные были скопированы из системы, например, на съемный накопитель. Дело в том, что существует множество способов, с помощью которых пользователь может скопировать данные из системы, например, вложить файл в веб-почту, использовать FTP-клиент и т. д. При поиске признаков того, что файлы были скопированы из системы на съемный накопитель, необходимо учитывать тот факт, что на сегодняшний день невозможно найти очевидные артефакты этого процесса, и вам придется клонировать оба носителя данных (т. е. накопитель, который был источником, и съемный накопитель, на который выполнялось копирование). Насколько мне известно, артефакты операции копирования, такие как перетаскивание мышью или использование команды *copy*, не записываются в реестре или файловой системе.

Но если пользователь скопирует файл на флеш-накопитель, а затем дважды щелкнет по этому файлу (например, чтобы проверить, что он скопирован правильно), то ОС Windows создает файл ярлыка (.lnk) для этого файла в пользовательской папке «Недавние документы» (“Recent Documents”). Хотя этот ярлык предоставит множество полезной информации (отметки времени, путь к файлу и т. д.), единственными данными, которые указывают на то, что файл был копирован из системы на флеш-накопитель, могут быть отметки времени изменения, последнего обращения и создания, встроенные в сам файл ярлыка. Тем не менее, ни один из имеющихся артефактов явно не свидетельствует о том, что файл был копирован из системы на флеш-накопитель.

Переносные устройства

Весной 2008 года Роб Ли (Rob Lee) из компании SANS обнаружил, что ОС Windows Vista сохраняет сведения о переносных устройствах в файле куста Software, и сообщил мне об этом. Хотя веб-сайт Microsoft содержит много информации, касающейся стандартов переносных устройств, он практически не предоставляет никаких сведений о следующем разделе реестра:

```
HKLM\Software\Microsoft\Windows Portable Devices\Devices
```

В этом разделе могут находиться подразделы, имена которых отформатированы примерно так же, как в разделе USBSTOR, обсуждавшемся выше. Фактически, некоторые из имен подразделов могут содержать слово «USBSTOR». В этих подразделах можно найти параметр *FriendlyName*, значение которого похоже на имя, отображаемое в окне «Мой компьютер», когда вы подключаете съемный накопитель. Модуль «port_dev.pl» программы RegRipper извлекает информацию из раздела Devices и отображает ее так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -r d:\cases\vista\software -p port_dev
Launching port_dev v.20090118
RemovDev
Microsoft\Windows Portable Devices\Devices
LastWrite Time Mon Jan 12 12:44:46 2009 (UTC)
Device      :
LastWrite   : Fri Sep 19 01:49:23 2008 (UTC)
SN          :
Drive       : Canon EOS DIGITAL REBEL XTi
Device      :
LastWrite   : Fri Dec 12 03:22:09 2008 (UTC)
SN          :
Drive       Apple iPod
```

```
Device      : DISK&VEN_APPLE&PROD_IPOD&REV_1.62
LastWrite   : Fri Sep 21 01:42:42 2007 (UTC)
SN          : 000A270018A0E610&0
Drive       : IPOD (F:)
Device      : DISK&VEN_BEST_BUY&PROD_GEEK_SQUAD_U3&REV_6.15
LastWrite   : Thu Feb 7 13:26:19 2008 (UTC)
SN          : 0C90195032E36889&0
Drive       : GEEKSQUAD (F:)
Device      : DISK&VEN_CASIO&PROD_DIGITAL_CAMERA&REV_1.00
LastWrite   : Sat Jun 28 18:38:28 2008 (UTC)
SN          : 1100101211329640&0
Drive       : Removable Disk (F:)
Device      : DISK&VEN_CASIO&PROD_DIGITAL_CAMERA&REV_1.00
LastWrite   : Sat Dec 15 01:17:56 2007 (UTC)
SN          : 6&14BB4B7C&0
Drive       : Removable Disk (F:)
```

Как видите, к компьютеру с ОС Vista подключался ряд устройств (в выходных данных показана только часть имеющейся информации), в том числе устройство iPod, цифровые камеры и флеш-накопитель Geek Squad. Кроме того, отображается серийный номер (если таковой имеется), а также значение *FriendlyName*, которое указано в выходных данных в строке «*Drive:*». Как видите, все четыре перечисленных устройства были сопоставлены с накопителем F:. Исследовав несколько систем Windows XP, я обнаружил, что при поочередном подключении к компьютеру ряда съемных USB-устройств им всем назначается одна и та же буква накопителя. В ОС Windows XP и 2003 у вас могут возникнуть трудности при попытке сопоставить съемное запоминающее устройство с буквой, назначаемой накопителю, который подключается к компьютеру, что было показано в этой главе ранее во время обсуждения раздела MountedDevices. Однако, похоже, что в ОС Windows Vista (и, вероятно, в Windows 7) в кусте Software сохраняется как минимум часть данных о назначении букв накопителей.

Поиск сведений о пользователях

Информация о пользователях сохраняется в реестре в файле куста SAM. При обычных обстоятельствах это куст недоступен даже для администраторов; чтобы получить к нему доступ, необходимо вручную отредактировать соответствующие права. Этому есть объяснение: несмотря на то, что большинство параметров реестра можно редактировать, существуют разделы реестра, в которых даже незначительные изменения могут сделать систему непригодной для использования. Один из таких разделов – это файл куста SAM.

Большая часть полезных данных в кусте SAM закодирована в двоичном формате; к счастью, заголовочный файл «sam.h» языка C, созданный Питером Нордалом-Хагеном, помогает расшифровать структуры и преобразовать их в понятную информацию.

Скрипт ProScript «userdump.pl» можно использовать для извлечения информации о пользователе и групповом членстве из приложения Registry Viewer в программе ProDiscover, сразу после того, как Registry Viewer будет заполнено данными. (Скрипт версии 0.31, 20060522 доступен в каталоге ch4\code\ProScripts на носителе, который идет в комплекте с этой книгой.) Для того чтобы запустить скрипт, нажмите кнопку **«Запустить ProScript»** (“Run ProScript”) в строке меню и выберите местонахождение скрипта в диалоговом окне **«Запустить ProScript»** (“Run ProScript”). (При необходимости можно также ввести параметры для скрипта ProScript.) Выберите скрипт **«user-dump.pl»**, и, как только исполнение скрипта завершится, информация, полученная из куста SAM, будет показана в окне результатов, откуда ее можно будет скопировать, а затем вставить в файл или отчет.

Запустив скрипт «userdump.pl» в текущем пробном деле, мы можем просмотреть часть результатов и понять, какой объем информации возвращается скриптом. Например,

этот скрипт извлекает информацию об учетных записях пользователей системы, в том числе имя, комментарий, дату создания учетной записи, число входов в систему и пользовательские флаги (предоставляющие информацию об учетной записи). Скрипт также отображает время последнего входа в систему, если оно не равно нулю.

```
Username : Administrator
Comment  : Built-in account for administering the computer/domain
Acct Creation Date : Thu Aug 19 17:17:29 2004
RID      : 500
Logins   : 0
Flags    :
          Password does not expire
          Normal user account
Username : Mr. Evil
Acct Creation Date : Thu Aug 19 23:03:54 2004
RID      : 1003
Logins   : 15
Flags    :
          Password does not expire
          Normal user account
```

Эта информация о пользователе хранится в параметре *F*, который находится по следующему адресу:

SAM\SAM\Domains\Account\Users\{*RID*}

Относительный идентификатор (“Relative Identifier”, RID) является частью идентификатора безопасности (“Security Identifier”, SID), определяющей пользователя или группу относительно органа, выдавшего идентификатор SID. Помимо некоторой информации о способах создания идентификаторов SID, Microsoft предоставляет список идентификаторов RID (<http://support.microsoft.com/kb/157234>) для известных пользователей и групп, а также для известных псевдонимов (которые показаны в разделе SAM\SAM\Domains\Builtin\Aliases).

Параметр *F* в этом разделе содержит двоичные данные, и его нужно проанализировать соответствующим образом, чтобы извлечь всю информацию (см. файл «sam.h», часть исходного кода утилиты Питера). В содержимом параметра *F* есть несколько важных дат, а именно несколько отметок времени и дат, представленных 64-разрядными значениями структуры FILETIME. Ниже показаны эти значения и их местонахождение:

- § в байтах 8–15 содержится дата последнего входа в систему для учетной записи пользователя;
- § в байтах 24–31 содержится дата последнего сброса пароля (если пароль не был сброшен или изменен, эта дата соотносится с датой создания учетной записи);
- § в байтах 32–39 содержится дата истечения срока действия учетной записи;
- § в байтах 40–47 содержится дата последней неудачной попытки входа в систему (для того чтобы дата изменилась для отдельной учетной записи, имя этой учетной записи должно быть правильным, поэтому эту дату можно также назвать датой последнего использования неверного пароля).

Такие инструменты, как Registry Viewer от компании AccessData, автоматически расшифровывают эту информацию, см. илл. 4.18.

Key Properties	
Last Written Time	9/26/2005 23:37:51 UTC
SID unique identifier	1003
Logon Name	Harlan
Last Logon Time	9/26/2005 23:37:51 UTC
Last Password Change Time	8/18/2004 0:49:42 UTC
Last Failed Login Time	9/26/2005 23:37:47 UTC

SAM\SAM\Domains\Account\Users\0000003EB

Илл. 4.18. Фрагмент данных из Registry Viewer, в которых показан расшифрованный параметр F.

Приложение Registry Viewer от компании AccessData доступно по адресу www.accessdata.com/downloads.html; если у вас нет электронного ключа AccessData, оно будет работать в демонстрационном режиме. Чтобы использовать Registry Viewer для расшифровки этих параметров, сначала необходимо извлечь из образа необработанный файл реестра (в данном случае файл SAM) и скопировать его в другое место.

Perl-модуль Parse::Win32Registry (доступный по адресу <http://search.cpan.org/~jmacfarla/>) предоставляет межплатформенный способ для анализа содержимого необработанных файлов реестра, которые были подобным образом извлечены или к которым можно получить доступ с помощью других средств. Чтобы установить этот модуль версии 0.41 (на момент написания книги) в ОС Windows, просто используйте утилиту Perl Package Manager (ppm) от компании ActiveState, как показано ниже:

```
C:\Perl>ppm install parse-win32registry
```

Скрипт «smparse.pl» (автономный Perl-скрипты «sam_parse.pl» находится в каталоге ch4\code\old на носителе, который идет в комплекте с этой книгой) использует данный модуль для извлечения и отображения этой, а также дополнительной информации из Registry Viewer:

```
Username      : Harlan [1003]
Full Name    :
User Comment :
Last Login Date : Mon Sep 26 23:37:51 2005 Z
Pwd Reset Date : Wed Aug 18 00:49:42 2004 Z
Pwd Fail Date  : Mon Sep 26 23:37:47 2005 Z
Login Count   : 35
--> Password does not expire
--> Normal user account
```

Скрипт «smparse.pl» не только показывает отметки времени, имеющиеся в данных пользователя из файла SAM, но и анализирует пользовательские флаги и некоторые отметки времени входа пользователя в систему.

Так как модуль Parse::Win32Registry не зависит от интерфейса Windows API, скрипты Perl, которые используют этот модуль, можно запускать на различных платформах. Это значит, что эксперты и специалисты по расследованиям инцидентов могут использовать другие операционные системы, помимо Windows, когда нужно проанализировать содержимое необработанных файлов реестра. Если файлы реестра доступны (т. е. извлечены из образа и т. д.), то код, написанный с помощью этого модуля, можно запускать в ОС Linux, Windows или даже в системах Mac OS X, поддерживающих Perl.

Кроме того, в каждом из этих разделов есть параметр *V*, который также содержит двоичные данные и который можно проанализировать, чтобы получить такие параметры учетной записи пользователя, как полное имя, комментарий, путь к скрипту входа в систему (если таковой имеется) и хеш-значения зашифрованных паролей.

Скрипт «samparse.pl» также находит информацию о группах в системе, включая имя группы, комментарий к группе и данные о пользователях, назначенных этой группе:

```
Group Name      : Administrators [2]
LastWrite       : Wed Aug 18 00:46:24 2004 Z
Group Comment   : Administrators have complete and unrestricted access to the
                  computer/domain
Users :
  S-1-5-21-839522115-1801674531-2147200963-500
  S-1-5-21-839522115-1801674531-2147200963-1003
```

Информация о групповом членстве хранится в разделе SAM\SAM\Domains\BuiltIn\Aliases. В каждом подразделе RID из раздела Aliases есть параметр *C*, который содержит двоичные данные и который необходимо проанализировать, чтобы определить какие пользователи являются членами группы. Мне кажется, что лучший план для анализа этих двоичных данных описан в блоге Андреаса Шустера (Andreas Schuster) (http://computer.forensikblog.de/en/2006/02/list_members_of_a_windows_group.html). Эта информация была включена в состав скрипта ProScript «userdump.pl»:

Информация о пользователях и групповом членстве очень важна для понимания контекста экспертизы, особенно в отношении того, какие пользователи имели доступ к системе, и какой уровень доступа у них был (посредством членства в группах). Большую часть этой информации можно получить во время анализа работающего компьютера, используя доступные инструменты. Имея больше знаний о различных структурах, хранящихся в реестре, можно извлечь похожую информацию из созданного образа данных.

Отслеживание действий пользователя

Для того чтобы отслеживать действия пользователя в системе, можно использовать несколько разделов реестра. Этот тип разделов отличается от разделов, связанных с местами автозапуска. Эти разделы реестра можно найти в пользовательском файле «NTUSER.DAT»; они изменяются (т. е. в них добавляются записи), если пользователь выполняет отдельные действия. Когда это происходит, время *LastWrite* раздела обновляется, что возвращает нас к понятию реестра как файла журнала. Кроме того, есть разделы, отслеживающие действия пользователя и добавляющие связанную с параметрами реестра информацию об отметках времени (например, в подразделах *UserAssist*); эта информация об отметках времени хранится в данных параметра.

Куст HKEY_CURRENT_USER содержит большую часть информации о действиях пользователя в так называемых списках последних использованных объектов (англ. *most recently used lists* – MRU). В разделах этого куста хранится перечень файлов или команд, а также параметр *MRUlist*. Каждому параметру в разделе назначается идентификатор,

например, строчная буква, а параметр *MRUlist* показывает порядок, в котором происходит обращение к этим параметрам.

Разделы UserAssist

О разделах UserAssist написано довольно много, и большая часть написанного существует в виде вопросов. Отдельные разделы, представляющие для нас интерес, находятся по следующему пути в пользовательском файле «NTUSER.DAT»:

Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{GUID}\Count

GUID – глобально уникальный идентификатор; в данном случае два раздела с подобными именами находятся в разделе UserAssist: 5E6AB780-7743-11CF-A12B-00AA004AE837 и 75048700-EF1F-11D0-9888-006097DEACF9. В кусте HKEY_CLASSES_ROOT идентификатор GUID 5E6AB780-7743-11CF-A12B-00AA004AE837 указывает на панель инструментов Internet Toolbar (например, %SystemRoot%\system32\browseui.dll), а идентификатор GUID 75048700-EF1F-11D0-9888-006097DEACF9 – на элемент ActiveDesktop (например, %SystemRoot%\system32\SHELL32.dll). Важность этой информации станет понятной, после того как вы узнаете, что находится в этих разделах.

В каждом разделе Count может находиться неопределенное количество параметров. Когда я впервые начал искать информацию об этих разделах (чаще всего они назывались UserAssist, а не Count), то обнаружил интернет-сайты, на которых сообщалось о более чем 18 000 записях в одном разделе и приблизительно 400 записях в другом. Речь шла об одной операционной системе Windows 2000, которая на момент публикации этой информации в Интернете проработала около 5 лет. Что же такого особенного в этих разделах? Вкратце, они в некоторой степени регистрируют действия пользователей. Да-да, вы не ошиблись. Эти разделы отчасти регистрируют действия пользователя, как файл журнала.

Однако если вы перейдете к этим разделам в редакторе реестра, вы совсем этого не увидите. Вы увидите данные типа HRZR_HVGBBYONE, которые не имеют никакого смысла. Это объясняется тем, что параметры в этих двух разделах зашифрованы по алгоритму ROT-13. Алгоритм *ROT-13* представляет собой шрифт Цезаря, в котором каждая буква заменяется буквой, смещенной на 13 позиций в алфавите. Используя простую замену (на языке Perl: tr/N-ZA-Mn-za-m/A-Za-z/), мы увидим, что HRZR_HVGBBYONE на самом деле – UEUME_UITOOLBAR. Это имя уже более удобно для чтения, но пока мы не приблизились к ответу.

Имена параметров в обоих разделах зашифрованы по алгоритму ROT-13, и их можно легко расшифровать. Фактически, Perl-скрипт «uassist.pl» (входящий в состав носителя, который идет в комплекте с этой книгой) показывает, как можно просто выполнить такое преобразование в работающей системе. Настоящее сокровище этих разделов находится в данных, связанных с каждым параметром. Расшифрованное имя параметра часто указывает на приложение или исполняемый файл. В таких случаях данные, как правило, равны 16 байтам (четыре двойных слова) и содержат не только число запусков (количество запусков приложения или исполняемого файла), но и последнее время запуска (8-байтовое значение структуры *FILETIME*). Число запусков хранится во втором двойном слове и начинается с пяти; таким образом, число запусков 6 означает, что приложение было запущено один раз. Значение структуры *FILETIME* находится в третьем и четвертом двойных словах.

Расшифровывая имена параметров, вы увидите, что многие из них начинаются с символов *UEUME_*, за которыми следуют *RUNPATH*, *RUNPIDL*, *RUNCPL* и т. д. Эти теги можно относительно легко классифицировать:

- § **RUNPATH** относится к абсолютному пути в файловой системе; этот тег встречается, когда пользователь дважды щелкает по значку исполняемого файла в проводнике Windows или вводит имя приложения в поле «Выполнить» (“Run”) меню «Пуск» (“Start”).
- § **RUNCPL** относится к запуску элементов панели управления.
- § В теге **RUNPIDL** символы *PIDL* являются частью внутреннего пространства имен проводника и используются для ссылки на объекты. В отношении разделов UserAssist это чаще всего ярлыки (LNK-файлы), которые можно встретить, например, в меню «Пуск» (“Start”), когда пользователь переходит в раздел «Недавние документы» (“Recent documents”) и выбирает файл.

Например, компьютер, на котором я пишу эту книгу, – Dell Latitude D820, купленный в начале августа 2006 года. Каждый раз, когда я покупаю новый компьютер, я переформатирую НЖМД и заново устанавливаю операционную систему. В случае с компьютерами Dell это означает, что мне необходимо загрузить и установить несколько драйверов (на сайте Dell можно легко найти необходимые драйверы). Запуская скрипт «uassist.pl» на своем компьютере, я вижу следующую запись:

```
UEME_RUNPATH:F:\D820\A02_bios.exe
Mon Aug 7 16:35:39 2006 -- (1)
```

Скрипт возвращает значение структуры *FILETIME* в формате локального времени, поэтому мы видим, что приложение «D820_A02_bios.exe» запускалось один раз 7 августа. Другие записи содержат:

```
UEME_RUNCPL:"C:\WINDOWS\system32\desk.cpl",Display
Thu Aug 24 21:27:45 2006 -- (1)
UEME_RUNCPL:"C:\WINDOWS\system32\powercfg.cpl",Power Options
Thu Aug 24 21:27:07 2006 -- (1)
```

Здесь мы видим, что элементы панели управления «Экран» (“Display”) и «Электропитание» (“Power Options”) исполнялись 24 августа, и каждый из них был запущен только один раз. Ради интереса, 4 октября в 22:55 я открыл элемент «Экран» (“Display”) в панели управления, затем запустил этот скрипт Perl и обнаружил, что дата изменилась на «Wed Oct 4 22:55:59 2006» (среда, 4 октября, 22:55:59, 2006 год).

Итак, эти разделы в основном регистрируют количество запусков определенных приложений и время последнего выполнения этих действий. Эта информация может оказаться очень полезной во время работы над делом. Например, если вы видите запись типа «UEME_RUNCPL:timedate.cpl», это может означать, что пользователь обращался к элементу панели управления «Дата и время» (“Date and Time”), возможно, чтобы изменить системное время.

Скрипт «userassist.pl» поможет собрать информацию из файлов «NTUSER.DAT» во время экспертизы. Он использует модуль Parse::Win32Registry, чтобы получить доступ к необработанному файлу «NTUSER.DAT» (извлеченному из созданного образа) и найти раздел UserAssist, который содержит идентификатор GUID, указывающий на элемент ActiveDesktop. Скрипт получает время *LastWrite* для раздела, а затем анализирует этот раздел, извлекая и расшифровывая имена параметров. Выходные данные скрипта, показанные ниже, сортируются по отметкам времени, найденным в данных для каждого параметра:

```
LastWrite time = Mon Sep 26 23:33:06 2005 (UTC)
Mon Sep 26 23:33:06 2005 (UTC)
UEME_RUNPATH
UEME_RUNPATH:C:\WINDOWS\system32\notepad.exe
```

```
Mon Sep 26 23:26:43 2005 (UTC)
    UEME_RUNPATH:Z:\WINNT\system32\sol.exe
Mon Sep 26 23:22:30 2005 (UTC)
    UEME_UISCUT
    UEME_RUNPATH:Downloads.lnk
Mon Sep 26 23:16:26 2005 (UTC)
    UEME_RUNPATH:C:\Program Files\Morpheus\Morpheus.exe
Mon Sep 26 23:16:25 2005 (UTC)
    UEME_RUNPATH:Morpheus.lnk
Mon Sep 26 23:15:04 2005 (UTC)
    UEME_RUNPATH:C:\Program Files\Internet Explorer\iexplore.exe
Mon Sep 26 23:04:08 2005 (UTC)
    UEME_RUNPATH:d:\bintext.exe
```

Когда я демонстрирую, как эти инструменты анализируют файлы кустов реестра в ОС Windows XP and 2003, многие интересуются, будут ли они работать в Vista. Да, будут. Вот фрагмент выходных данных из пользовательского файла куста в ОС Windows Vista:

```
Sun Dec 28 02:19:10 2008 (UTC)
    UEME_RUNPATH:C:\Program Files\Zango\bin\10.3.75.0\ZangoUninstaller.exe (2)
Sun Dec 28 02:15:32 2008 (UTC)
    UEME_RUNPIDL:%csidl123%\LimeWire\LimeWire 4.18.3.lnk (1)
Tue Dec 23 01:43:50 2008 (UTC)
    UEME_RUNPIDL:%csidl123%\AIM\AIM 6.lnk (1)
Tue Dec 23 01:43:24 2008 (UTC)
    UEME_RUNPATH:C:\Program Files\AIM\aim.exe (21)
    UEME_RUNPIDL:C:\Users\Public\Desktop\AIM.lnk (2)
Thu Dec 18 23:07:27 2008 (UTC)
    UEME_RUNPATH:C:\Program Files\Microsoft Works\WksWP.exe (80)
    UEME_RUNPIDL:%csidl123%\Microsoft Works\Microsoft Works
Word Processor.lnk (47)
Wed Dec 17 23:14:58 2008 (UTC)
    UEME_RUNPATH:C:\Users\user\Music\LimeWire\LimeWire.exe (4)
Wed Dec 17 23:14:57 2008 (UTC)
    UEME_RUNPIDL:C:\Users\user\Desktop\LimeWire 4.18.3.lnk (4)
Wed Dec 17 01:54:05 2008 (UTC)
    UEME_RUNPIDL:%csidl123%\Adobe Photoshop Elements 6.0.lnk (1)
Tue Dec 16 02:21:28 2008 (UTC)
    UEME_RUNPIDL:%csidl123%\iTunes\iTunes.lnk (15)
```

Одно из преимуществ анализа содержимого раздела UserAssist состоит в том, что скрипт не только показывает, какие действия выполнил пользователь через оболочку (например, дважды щелкнул по значку, запустил приложение из меню «Пуск» (“Start”) или получил доступ к элементам панели управления), но также показывает, когда были совершены эти действия.

Perl-модуль Parse::Win32Registry позволяет получить легкий доступ к двоичному содержимому файлов куста реестра. Некоторые аспекты реестра изменились после выпуска различных версий операционных систем, однако двоичная структура реестра осталась без изменений. Структура и параметры раздела UserAssist не менялись с версии Windows 2000 по Vista, но бегло изучив бета-версию Windows 7 в январе 2009 года, я понял, что в ней появились изменения, на которые нужно обратить внимание. Хотя двоичный формат файлов кустов реестра остался прежним, содержимое разделов UserAssist изменилось, поэтому для его анализа потребуется добавить дополнительный код в модули программы RegRipper.

Скрипт «uassist.pl», используемый в программе ProDiscover, доступен на носителе, который идет в комплекте с этой книгой, в каталоге ch4\code\Proscripts. Этот скрипт (версия 0.11, 20060522) можно запустить для анализа реестра после того, как приложение Registry Viewer будет заполнено данными. Он анализирует записи раздела реестра

UserAssist для всех пользователей и извлекает информацию в доступном для чтения формате, расшифровывая имена разделов и извлекая из данных число запусков и время последнего запуска, где это возможно. После этого скрипт сортирует все значения по отметкам времени, чтобы эту информацию можно было использовать для анализа временной шкалы. Эта версия скрипта ProScript отправляет результаты в окно команд в программе ProDiscover, откуда эксперт может скопировать результаты, а затем вставить их в файл или отчет.

Ваш компьютер защищен?

Анализ конкретной ситуации: Была ли дефрагментация?

Однажды я столкнулся с ситуацией, когда сотрудник моего клиента, вероятно, зная, что находится под подозрением и, пока у него был доступ к своему компьютеру, возможно, удалил ряд файлов. Имена данных файлов были обозначены как удаленные, но их содержимое просто отсутствовало. Я начал подозревать, что он не только удалил файлы, но и отформатировал НЖМД. Информация в разделе UserAssist свидетельствовала о том, что этот сотрудник запускал элемент «Установка и удаление программ» (“Add/Remove Programs”) из панели управления. Судя по информации в каталоге «Prefetch» (подробнее см. в главе 5), была запущена программа дефрагментации, но в разделе UserAssist ничего не указывало на то, что сотрудник запускал такую программу. Зарегистрированное событие, как оказалось, было частью неполной дефрагментации, которую ОС Windows XP выполняет раз в три дня.

Совет

Если вы исследуете образ системы, в которой пользователь установил Internet Explorer седьмой версии, то в разделе реестра UserAssist найдете третий подраздел.

Раздел MUICache

Помимо параметров раздела UserAssist, есть еще один способ узнать, какие программы запускались пользователем на компьютере. Раздел MUICache находится по следующему адресу:

Software\Microsoft\Windows\ShellNoRoam\MUICache

В отличие от раздела UserAssist, в параметрах раздела MUICache нет отметок даты и времени. В действительности с параметрами раздела MUICache связаны описательные комментарии. Хотя некоторые из параметров, такие как те, что начинаются с символа «@», вероятно, являются записями по умолчанию для программ, предустановленных на компьютере, другие, похоже, создаются, когда программы запускаются в системе.

Модуль «muicache.pl» программы RegRipper извлекает данные из раздела MUICache в файле куста «NTUSER.DAT», пропускает параметры, начинающиеся с «@» и отображает оставшиеся параметры так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -r d:\cases\NTUSER.DAT -p muicache
Launching muicache v.20080324
MUICache
Software\Microsoft\Windows\ShellNoRoam\MUICache
LastWrite Time Mon Sep 26 23:34:11 2005 (UTC)
    C:\Program Files\VMware\VMware Tools\VMwareTray.exe (VMwareTray)
    C:\Program Files\VMware\VMware Tools\VMwareUser.exe (VMwareUser)
    C:\Program Files\WinZip\WZQKPICK.EXE (WinZip Executable)
    C:\PROGRA~1\MOZILL~1\FIREFOX.EXE (Firefox)
    C:\Program Files\Internet Explorer\iexplore.exe (Internet Explorer)
    C:\WINDOWS\system32\notepad.exe (Notepad)
    C:\WINDOWS\system32\cmd.exe (Windows Command Processor)
```

```
C:\WINDOWS\regedit.exe (Registry Editor)
C:\WINDOWS\notepad.exe (Notepad)
C:\WINDOWS\Explorer.EXE (Windows Explorer)
C:\PROGRA~1\WINZIP\winzip32.exe (WinZip)
C:\WINDOWS\system32\zipfldr.dll (Compressed (zipped) Folders)
C:\WINDOWS\System32\shimgvw.dll (Windows Picture and Fax Viewer)
C:\WINDOWS\system32\mspaint.exe (Paint)
C:\Program Files\Windows NT\Accessories\WORDPAD.EXE (WordPad)
C:\WINDOWS\system32\shell32.dll (Windows Shell Common Dll)
C:\Program Files\Windows Media Player\wmplayer.exe (Windows Media Player)
d:\bintext.exe (bintext)
C:\WINDOWS\system32\SNDVOL32.EXE (Volume Control)
C:\Program Files\Morpheus\Morpheus.exe (Morpheus)
C:\Program Files\Windows Media Player\setup_wm.exe (Microsoft Windows Media Configuration Utility)
C:\WINDOWS\inf\unregmp2.exe (Microsoft Windows Media Player Setup Utility)
Z:\WINNT\system32\sol.exe (Solitaire Game Applet)
D:\ShellExe.exe (ShellExe launches documents from autorun files!)
D:\PDServer.exe (PDServer, Win32 Application)
```

Как видите, показано множество имен параметров и данных. Как и в предыдущих случаях, практически отсутствует документация (особенно от производителя), в которой бы описывалось, как параметры добавляются в этот раздел и при каких обстоятельствах они могут быть удалены из раздела. Однако, судя по всему, для того, чтобы параметр добавился в раздел MUICache, пользователь должен запустить на компьютере программу или приложение. Следовательно, содержимое этого раздела даст эксперту представление о действиях пользователя, даже если последний удалил приложение или (как было показано в предшествующих выходных данных модуля) запустил приложение с внешнего носителя.

Диана Барретт (Diane Barrett) из Университета передовых технологий (University of Advancing Technology, UAT) работала в области артефактов виртуализации, т. е. искала признаки применения технологий виртуализации на компьютерах. В одной из своих презентаций Диана показала, что признаки использования таких приложений виртуализации, как MokaFive (дополнительные сведения см. на сайтах www.mokafive.com/ и <http://en.wikipedia.org/wiki/Moka5>) и MojoPac (www.mojopac.com/), можно увидеть в разделе MUICache. В некоторых случаях такие типы приложений могут применяться, чтобы сохранить действия пользователя в секрете, так как программы главного компьютера не используются, а такие приложения, как веб-браузеры, настроены так, чтобы не сохранять кэш или историю действий пользователя. Интересно, что, по имеющимся данным, MojoPac сохраняет свой собственный реестр, поэтому, если вы имеете дело с этой средой виртуализации, рекомендуется также выполнить клонирование и анализ устройства с программным обеспечением MojoPac. Кроме того, не забывайте, что существуют и могут использоваться другие среды виртуализации, такие как бесплатная программа Portable Virtual Machine от компании MetroPipe (www.metropipe.net/pvpm.php).

Списки MRU

Многие приложения сохраняют списки MRU – списки недавно использованных файлов. В работающем приложении эти имена файлов обычно отображаются в нижней части раскрывающегося списка при выборе пункта «Файл» (“File”) в строке меню.

Возможно самый известный (и самый полный) список MRU в реестре хранится в разделе RecentDocs:

```
\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs
```

В этом разделе довольно много параметров, все из которых содержат двоичные данные. Нас интересуют параметры с именами в виде цифр (эти параметры содержат имена файлов, к которым был получен доступ, в Unicode) и параметр *MRUListEx*, регистрирующий порядок (в виде двойных слов), в котором к файлам был получен доступ. Например, на моем компьютере первое двойное слово (т. е. 4-байтовое значение) в данных параметра *MRUListEx* – это 0x26, или 38 в десятичной системе счисления. Параметр с именем «38» указывает на каталог, который я открывал. Учитывая, что добавление параметра и связанных с ним данных в раздел, а также изменение параметра *MRUListEx* означает изменение раздела, время *LastWrite* раздела RecentDocs подскажет нам, когда к этому файлу был получен доступ.

Еще раз повторю, что параметры в разделе RecentDocs (и его подразделах) имеют имена в виде цифр, однако эти имена не означают порядок, в котором был получен доступ к файлам, указанным в данных параметров. Эта информация хранится в данных параметра *MRUListEx*.

Совет

В большинстве случаев в разделах со списками MRU есть данные, помогающие определить, когда было совершено последнее действие. В разделе RecentDocs параметр *MRUListEx* содержит упорядоченный список таких действий. В других разделах эта информация может сохраняться по-другому. Однако время *LastWrite* раздела соответствует времени совершения самого последнего действия. При проведении анализа это будет дополнительной информацией, которую можно использовать для составления временной шкалы действий в системе.

В разделе RecentDocs также есть ряд подразделов; имена каждого из этих подразделов представляют собой расширения файлов, которые были открыты (.doc, .txt, .html и т. д.). Параметры в этих подразделах сохраняются так же, как в разделе RecentDocs. Имена параметров пронумерованы, а сами параметры содержат имя файла, к которому был получен доступ, в виде данных двоичного типа (в Unicode). Еще в одном параметре *MRUListEx*, содержащем двоичный тип данных, сохраняются сведения о порядке, в котором к файлам был получен доступ (первыми перечислены последние открытые файлы), в виде двойных слов. Модуль «recentdocs.pl» анализирует содержимое раздела RecentDocs, а также содержимое каждого из подразделов, и перечисляет параметры в порядке *MRUListEx*, как показано во фрагменте выходных данных модуля ниже:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.MOV
LastWrite Time Mon Oct 27 00:23:40 2008 (UTC)
MRUListEx = 4,1,3,7,9,2,8,0,6,5
 4 = CIMG1801.MOV
 1 = CIMG1800.MOV
 3 = CIMG2597.MOV
 7 = CIMG2596.MOV
 9 = CIMG2595.MOV
 2 = CIMG2594.MOV
 8 = CIMG2593.MOV
 0 = CIMG2592.MOV
 6 = CIMG2591.MOV
 5 = CIMG2589.MOV
```

```
Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.mp3
LastWrite Time Fri Jul 11 19:45:05 2008 (UTC)
MRUListEx = 1,0,4,9,8,7,6,5,3,2
 1 = full_9c0a82e420f4e17f240aa142f209935e.mp3
 0 = full_7702b626e3e56525356e08ef5bde6d9e.mp3
 4 = djsona.mp3
```

```
9 = full_27fb554b26316c4b20cd8e6a013a6214.mp3
8 = 03-britney_spears-radar.mp3
7 = Britney_Spears_-_Radar.mp3
6 = full_bac9c7b07c84213659d8a8b9098d8379.mp3
5 = heizman.mp3
3 = full_025d9af06f9e17201f60f248207f905b.mp3
2 = Christina_Aguilera_-_Can't_Hold_Us_Down.mp3
```

Как видите, модуль отображает имя подраздела RecentDocs, время *LastWrite* для этого раздела, содержимое параметра *MRUListEx* (в котором удалено последнее значение 0xffffffff), а также имена параметров в порядке *MRUListEx*. В этом примере мы видим, что пользователь смотрел фильм формата MOV в понедельник, 27 октября, и слушал отдельный MP3-файл в пятницу, 11 июля.

Еще один известный список MRU находится в разделе RunMRU:

```
\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
```

Этот раздел содержит список всех значений, введенных в окне «Выполнить» («Run»), выбранном в меню «Пуск» («Start»). На илл. 4.19 показано, как может выглядеть содержимое этого раздела.

 a	notepad\1
 b	calc\1
 c	regedit\1
 d	sol\1
 MRUList	dbc\1

Илл. 4.19. Часть данных из раздела RunMRU.

Данные параметра *MRUList* из раздела RunMRU сохраняются в виде простого текста и более удобны для чтения, чем данные параметра *MRUListEx* в разделе RecentDocs. Однако, как и в разделе RecentDocs, самые последние введенные значения перечисляются первыми в параметре *MRUList*.

Записи добавляются в этот раздел, когда пользователь щелкает по кнопке «Пуск» («Start»), выбирает пункт «Выполнить» («Run») и вводит команду, имя файла или другие значения. Модуль «runmru.pl» отображает информацию из раздела RunMRU так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -p runmru -r d:\cases\lenovo\NTUSER.DAT
Launching runmru v.20080324
RunMru
Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU
LastWrite Time Thu Jan 31 21:43:17 2008 (UTC)
MRUList = edchbgfa
a  \\Server\1
b  \\Server\Share\1
c  calc\1
d  notepad\1
e  regedit\1
f  conf\1
g  services.msc\1
h  cmd\1
```

Параметр *MRUList* в разделе RunMRU показывает, что последний элемент, введенный в поле «Выполнить», – это «е», что в данном случае соответствует команде «regedit». Имена параметров из раздела RunMRU, как и параметры в разделе RecentDocs, не означают порядок, в котором различные элементы были введены в поле «Выполнить»

(“Run”); эта информация сохраняется в параметре *MRUList*. В предыдущем примере значение «h» (т. е. «cmd») не было последним элементом, введенным в поле «Выполнить» (“Run”); этим элементом была строка «regedit».

Во время расследований, связанных с действиями пользователей, в разделе RunMRU мне встречались записи, которые показывали доступ к удаленным компьютерам (см. предыдущий пример), а также к приложениям и файлам на съемных накопителях.

Еще один раздел, похожий на RunMRU, – это раздел TypedURLs.

\Software\Microsoft\Internet Explorer\TypedURLs

Подобно разделу RunMRU, раздел TypedURLs сохраняет список URL-адресов, которые пользователь вводит в адресную строку в браузере Internet Explorer. Однако имена параметров в разделе TypedURLs перечислены таким образом, что самый последний введенный URL-адрес имеет наименьший номер; следовательно, в этом разделе нет параметра *MRUList* или *MRUListEx*. Модуль «typedurls.pl» отображает информацию из этого раздела так, как показано ниже:

```
C:\Perl\forensics\rri>rip.pl -p typedurls -r d:\cases\lenovo\NTUSER.DAT
Launching typedurls v.20080324
TypedURLs
Software\Microsoft\Internet Explorer\TypedURLs
LastWrite Time Mon Feb 4 23:02:44 2008 (UTC)
url1 -> http://mail.yahoo.com/
url2 -> http://groups.yahoo.com/
url3 -> http://www.forensicfocus.com/
url4 -> http://www.orbitz.com/
url5 -> http://www.facebook.com/
url6 -> http://www.google.com/
url7 -> http://www.blogger.com/
url8 -> http://news.yahoo.com/
```

Эту информацию можно объединить с временными файлами Интернета, чтобы показать веб-сайты, на которые пользователь переходил по ссылке, и веб-сайты, адреса которых пользователь вводил самостоятельно.

Списки MRU можно также найти в следующем разделе:

\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU

Этот раздел (подробная информация о нем доступна по адресу <http://support.microsoft.com/kb/322948>) содержит списки файлов, к которым был получен доступ через диалоговые окна «Открыть» (“Open”) и «Сохранить как» (“Save As”) в оболочке Windows. Как и в разделе RecentDocs, в OpenSaveMRU есть подразделы отдельных расширений для файлов, которые были открыты или сохранены. Но, как и раздел RunMRU, этот раздел состоит из легких для чтения строковых данных. Содержимое этого раздела может оказаться полезным в различных случаях. Во-первых, некоторые расширения файлов не часто встречаются во время обычного использования системы, поэтому в разделе OpenSaveMRU подразделы для таких расширений файлов могут иметь только одну запись с именем *a*. В таком случае данные для параметра *MRUList* будут содержать только значение *a*, как показано на илл. 4.20. Время *LastWrite* этого раздела подскажет вам, когда этот файл был открыт или сохранен.

 a	REG_SZ	G:\book2\memory\pe_image.h.htm
 MRUList	REG_SZ	a

Илл. 4.20. Фрагмент данных из подраздела в разделе OpenSaveMRU.

Открыв в программе ProDiscover образ, полученный из ОС Windows XP, я перешел в раздел OpenSaveMRU. Время *LastWrite* для подраздела exe указано как 17 августа 2004 года, 11 часов 18 минут утра. На илл. 4.21 показано содержимое подраздела exe.

a	REG_SZ	C:\Documents and Settings\Mr. Evil\Desktop\valsetup250.exe
MRUList	REG_SZ	cdba
b	REG_SZ	C:\Documents and Settings\Mr. Evil\Desktop\netstumblerinstaller_0_4_0.exe
c	REG_SZ	C:\Documents and Settings\Mr. Evil\Desktop\ethereal-setup-0.10.6.exe
d	REG_SZ	C:\Documents and Settings\Mr. Evil\Desktop\WinPcap_3_01_a.exe

Илл. 4.21. Фрагмент данных из подраздела exe в разделе OpenSaveMRU (в ProDiscover).

На илл. 4.21 показано, что последний открывавшийся файл – это программа установки Ethereal (Ethereal – пакет инструментов для захвата и анализа сетевого трафика, сегодня называется Wireshark), которая использовалась для установки приложения. Эту информацию затем можно сопоставить с содержимым раздела UserAssist (используя скрипт «uassist.pl» версии 0.11, 20060522). После того как скрипт проанализирует тестовый образ, мы увидим следующие данные:

```
--> Fri Aug 27 15:34:54 2004
--> UEME_RUNPATH:C:\Program Files\Ethereal\ethereal.exe
--> Fri Aug 27 15:33:02 2004
--> UEME_RUNPATH:C:\Program Files\Cain\Cain.exe
--> Fri Aug 27 15:28:36 2004
--> UEME_RUNPATH:C:\Documents and Settings\Mr. Evil\Desktop\ethereal-setup-0.10.6.exe
--> Fri Aug 27 15:15:08 2004
--> UEME_RUNPATH:C:\Documents and Settings\Mr.Evil\Desktop\WinPcap_3_01_a.exe
--> Fri Aug 27 15:14:44 2004
--> UEME_RUNCPL
--> UEME_RUNCPL:"C:\WINDOWS\System32\appwiz.cpl",Add or Remove Programs
--> Fri Aug 27 15:12:35 2004
--> UEME_RUNPATH:C:\Program Files\Network Stumbler\NetStumbler.exe
--> Fri Aug 27 15:12:11 2004
--> UEME_RUNPATH:C:\Documents and Settings\Mr. Evil\Desktop\netstumblerinstaller_0_4_0.exe
```

Как видите, наш пользователь был очень занят 27 августа 2004 года. (Скрипт ProScript извлекает необработанные данные *FILETIME* и преобразовывает их в формат времени UTC, который приблизительно соответствует времени по Гринвичу. Программа ProDiscover показывает все отметки времени по отношению к компьютеру эксперта и параметрам раздела *TimeZoneInformation*.) Раздел реестра *TimeZoneInformation* показывает, что система во время работы была настроена на центральное поясное время с автоматическим переходом на летнее время и обратно. Значение *ActiveTimeBias* равно 300 минут (5 часов), а значение *ActiveTimeBias* на моем компьютере – 240 минут (4 часа). Исходя из этого, мы видим, что примерно через 10 минут после сохранения на своем компьютере программы установки Ethereal подозреваемый установил это приложение.

Эта информация также может быть полезна для того, чтобы показать использование внешних запоминающих устройств. Помимо того, что время *LastWrite* этого подраздела содержит дату и время подключения устройства к компьютеру, эту информацию можно сопоставить с содержимым раздела *MountedDevices*, чтобы получить дополнительные сведения об устройстве.

Еще один список MRU можно найти в следующем разделе:

Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts

Подразделы в этом разделе соответствуют расширениям для файлов, которые были открыты на компьютере. В этих подразделах с именами в виде расширений файлов находятся подразделы с именами OpenWithProgIDs и OpenWithList. Эти записи реестра указывают системе, какое приложение использовать для открытия файла с данным расширением, когда пользователь выполняет двойной щелчок по файлу. Данная информация может быть полезна во время экспертизы, так как пользователь мог удалить, а затем установить приложение, которое использовалось для открытия файлов с отдельным расширением. После того как это приложение удалено, информация о нем может все еще оставаться в разделе FileExts. Модуль «fileexts.pl» программы RegRipper анализирует и отображает эту информацию.

Помощник по поиску

Когда пользователь нажимает кнопку «Пуск» (“Start”) в ОС Windows XP, открывает окно «Поиск» (“Search”), а затем выбирает раздел «Файлы и папки» (“For Files and Folders”), то поисковые термины, введенные в диалоговом окне, сохраняются в следующем разделе реестра:

Software\Microsoft\Search Assistant\ACMru

Раздел ACMru обычно представляет собой сочетание четырех подразделов: 5001, 5603, 5604 и 5647. Подраздел 5001 содержит список MRU для помощника поиска в Интернете, подраздел 5603 содержит список MRU для поиска файлов и папок в ОС Windows XP, а подраздел 5604 содержит список MRU, который соответствует диалоговому окну «Слово или фраза в файле» (“Word or phrase in a file”). Подраздел 5647 содержит список MRU для имен компьютеров, введенных в разделе «Компьютеры или людей» (“For computers or people”) в диалоговом окне «Результаты поиска» (“Search Results”). Имена параметров в этих подразделах состоят из трехзначных чисел. Наименьшее число (т. е. 000) соответствует самому последнему поиску, а время *LastWrite*, связанное с этим разделом, содержит дату и время выполнения этого поиска. Модуль «acmru.pl» отображает информацию из этого раздела так, как показано ниже:

```
C:\Perl\forensics\rr>rip.pl -p acmru -r d:\cases\NTUSER.DAT
Launching acmru v.20080324
ACMru - Search Assistant
Software\Microsoft\Search Assistant\ACMru
LastWrite Time Mon Sep 26 23:02:08 2005 (UTC)
5603 [Mon Sep 26 23:32:56 2005 (UTC)]
    000 -> port*
    001 -> sol.exe
    002 -> hacker*
    003 -> hack*
    004 -> lad*
5604 [Mon Sep 26 23:33:30 2005 (UTC)]
    000 -> disk
    001 -> ha*
```

Знания о назначении различных подразделов и о том, как они заполняются данными, помогут вам получить представление о действиях пользователя в системе. Эта информация может быть полезна во время расследований, когда нужно определить, что и когда делал пользователь. В предыдущем примере пользователь выполнял поиск имен файлов, используя такие термины, как «*port**» и «*sol.exe*», а также искал файлы, содержащие ключевое слово «*disk*». Во время одного из расследований, связанного с получением несанкционированного доступа к компьютеру через службы терминалов, мы видели, что злоумышленник выполнял поиск слова «*bank**».

Информация о поиске для устаревших систем, таких как Windows 2000, хранится в другом разделе реестра, и ее можно найти, если система Windows была обновлена с версии 2000 до XP. Раздел, о котором идет речь, – это:

```
Software\Microsoft\Internet Explorer\Explorer Bars\{C4EE31F3-4768-11D2-BE5C-00A0C9A83DA1}
```

Согласно содержимому раздела HKEY_CLASSES_ROOT\CLSID, этот идентификатор GUID указывает на панель браузера для поиска файлов, которая содержится в файле «shell32.dll». Два подраздела в этом разделе, FilesNamedMRU и ContainingTextMRU, соответствуют подразделам 5603 и 5604 (в указанном порядке) в ОС Windows XP.

Подключение к другим компьютерам

Если пользователь использует мастер подключения сетевого диска (щелкнув правой кнопкой мыши по значку «**Мой компьютер**» (“My Computer”) и выбрав пункт «**Подключить сетевой диск**» (“Map Network Drive”)), чтобы подключиться к удаленному компьютеру, то в нижеуказанном разделе создается список MRU:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\Map Network Drive MRU
```

Каждой записи присваивается буква в качестве имени параметра, а параметр *MRUList* показывает порядок, в котором пользователь подключался к каждому накопителю или общему ресурсу.

Каждый раз, когда пользователь использует мастер подключения сетевого диска или команду *net use*, тома, добавляемые пользователем в систему, будут показаны в следующем разделе:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2
```

Как было упомянуто выше, подразделы из раздела MountPoints2, которые отображаются как идентификаторы GUID, можно сопоставить с записями \??\Volume в разделе MountedDevices. Эти идентификаторы GUID также можно сопоставить с подразделом CPC\Volume в разделе MountPoints2.

На своем компьютере я использовал команду *net use*, чтобы выполнить проверку, а когда я подключился к общему ресурсу C\$ на другом компьютере, то увидел такие подразделы, как ##192.168.1.22#c\$ и ##192.168.1.71#c\$.

Эти IP-адреса (у меня неструктурированная («плоская») контрольная сеть, не являющаяся доменом, поэтому имена компьютеров – это в основном IP-адреса) также отображаются в следующем разделе реестра:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\ComputerDescriptions
```

В этом разделе сохраняются описания компьютеров, которые можно увидеть в программе просмотра сети. Для компьютеров, являющихся частью домена, в этом разделе обычно показано несколько имен компьютеров. Однако для автономных систем, таких как домашние компьютеры и другие компьютеры, которые не являются частью домена, в этом разделе, вероятно, не будут перечисляться параметры. На моем домашнем компьютере в этом разделе отображаются только компьютеры, к которым я подключился с помощью команды *net use*. Я использовал свой рабочий компьютер, чтобы подключиться к интрасети своего клиента через виртуальную частную сеть, после чего в разделе ComputerDescriptions появилось несколько компьютеров, к которым я подключился. Один из них имел описание «Samba 2.2.7a», что свидетельствует о том, что это компьютер с ОС

Linux, на котором запущена программа Samba. Так как этот раздел находится в файле «NTUSER.DAT», в том же компьютере могут быть разные записи для разных пользователей.

Запись компакт-дисков

При расследовании инцидента, когда пользователь, возможно, удалил данные с компьютера, нужно рассмотреть несколько способов, посредством которых это могло быть сделано. Скопировал ли пользователь файлы на съемное USB-устройство (например, флеш-накопитель, iPod и т. д.), или он отправил эти данные в виде вложений, используя веб-почту (Yahoo!, Gmail)? Еще один способ вывода данных, который нужно рассмотреть – это встроенная в ОС Windows XP возможность записи компакт-дисков. Хотя многие компьютеры выпускаются с уже установленным программным обеспечением для записи CD- и DVD-дисков (например, на моих компьютерах Dell и ноутбуке Lenovo ThinkPad были предустановлены продукты Sonic/Roxio), системы Windows XP и Vista имеют встроенную возможность записи компакт-дисков. Эта возможность для Windows XP описана в статье № 279157 из базы знаний Microsoft (<http://support.microsoft.com/kb/279157>). Когда пользователь вставляет чистый записываемый или перезаписываемый компакт-диск, появляется диалоговое окно, в котором предлагается возможность «**Открыть компакт диск для записи; используется Проводник**» (“Open writable CD folder using Windows Explorer”). После того как папка компакт-диска будет открыта, пользователь может перетащить в нее файлы и каталоги, которые сохраняются в специальной промежуточной области, пока пользователь не выберет опцию «**Записать файлы на компакт-диск**» (“Write these files to CD”). Когда пользователь будет готов записать файлы на компакт-диск, в промежуточной области создается единый файл-образ диска с именем «Cd burning stash file.bin». На компьютерах с ОС Windows XP эта специальная промежуточная область по умолчанию находится в каталоге:

```
%USERPROFILE%\Local Settings\Application Data\Microsoft\CD Burning
```

На компьютерах с ОС Windows Vista промежуточная область по умолчанию находится в каталоге:

```
%USERPROFILE%\AppData\Local\Microsoft\Windows\Burn\Burn
```

Местонахождение промежуточной области указано в пользовательском файле куста «NTUSER.DAT» в следующем разделе:

```
Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
```

Модуль «shellfolders.pl» программы RegRipper извлекает из этого раздела имена параметров с их данными и выводит их в упорядоченном виде. Пользователи могут изменить местонахождение каталога записи компакт-дисков, отредактировав этот параметр реестра и указав в нем другое местонахождение. Если путь к каталогу был изменен, это может свидетельствовать о том, что на этом компьютере применялись правила компании, или что пользователь имеет определенный уровень технических навыков. Это может также означать, что эксперту нужно выполнить поиск артефактов файла с расширением .bin в другом каталоге.

Совет

Согласно статье № 326982 из базы знаний Microsoft (<http://support.microsoft.com/kb/326982>), пользователи не могут копировать файлы на диск

CD-R/CD-RW в ОС Windows 2003, потому что по умолчанию служба СОМ записи компакт-дисков IMAPI отключена. Это объясняется тем, что Windows 2003 считается серверной операционной системой, и возможность записи компакт-дисков не имеет большого значения. Тем не менее, эксперты должны проверить во всех ОС Windows не только параметры в разделе Shell Folders, но и статус службы СОМ записи компакт-дисков IMAPI в разделе Services, чтобы определить, использовал ли пользователь эту функциональную возможность для несанкционированного копирования данных из компьютера.

Обмен мгновенными сообщениями и одноранговые сети

Программы для обмена мгновенными сообщениями (“Instant messaging”, IM) и однорангового (“peer-to-peer”, P2P) доступа к файлам чрезвычайно популярны среди пользователей всех поколений. Если раньше требовалось немало времени, чтобы написать письмо, и чтобы это письмо дошло до адресата, сейчас можно мгновенно отправить электронное сообщение, которое пользователь получит, как только войдет в систему. Вы можете находиться на другом конце света, но, тем не менее, получите уведомление в тот момент, когда ваш друг войдет в IM-приложение. Или, используя одноранговый доступ к файлам, можно найти любое количество полезных (или не очень полезных) файлов – музыку, фильмы, изображения и т. д. Популярность этих технологий стала причиной появления множества разнообразных платформ и клиентских приложений. Такие компании, как Yahoo!, AOL и Microsoft, выпустили свои клиентские приложения для обмена мгновенными сообщениями; каждое из приложений имеет свои функциональные возможности и оставляет уникальные «следы» в системе. Кроме того, можно использовать приложения сторонних разработчиков, чтобы заменить эти клиенты или даже объединить их в один интерфейс. Например, программа Trillian (trillian.cc) позволяет объединять учетные записи других сервисов, поэтому пользователям нужно только войти в эту программу, чтобы получить доступ к различным IM-сетям. Сервис Meebo.com предоставляет похожий веб-интерфейс.

Такое же многообразие приложений существует для одноранговых сетей. Каждая технология представляет определенные трудности, когда дело касается судебной экспертизы. Например, как эксперту определить, с кем общался подозреваемый, используя IM-приложение, если это приложение не сохраняет переписку по умолчанию? Или как эксперту узнать, сохранил ли пользователь переписку самостоятельно, или это было выполнено посредством стороннего дополнения для сохранения истории сообщений? Что касается однорангового доступа к файлам, как эксперту определить, какие поисковые термины использовал подозреваемый, какие файлы были получены из файлообменной сети, и как установить источник этих файлов?

Разнообразие IM- и P2P-клиентов так велико, что потребуется отдельная книга, чтобы полностью рассмотреть вопросы судебной экспертизы для каждой из технологий. А с учетом того, что, как и другие приложения, IM- и P2P-клиенты изменяются с появлением новых версий, что включает в себя добавление функциональных возможностей и создание новых разделов реестра и файлов, задача составления каталога судебных артефактов таких приложений становится еще более трудной. Например, когда я использовал старую версию программы обмена мгновенными сообщениями, предлагаемой компанией AOL (“AOL Instant Messaging”, AIM), в профиле пользователя создавался отдельный набор разделов реестра, где сохранялся зашифрованный пароль. Это происходило, когда пользователь выбирал опцию автоматически входить в сеть AIM без повторного ввода пароля. Если бы во время экспертизы вам нужно было собрать информацию о действиях пользователя в программе AIM, вы могли бы использовать тот зашифрованный пароль, чтобы создать аналогичный профиль на другом компьютере, а затем войти в программу как тот пользователь. Не так давно я решил попробовать новую версию AIM-клиента – Triton; мне она понравилась, хотя требуется время, чтобы к ней

привыкнуть. Одно из главных изменений в интерфейсе заключается в том, что теперь для каждой переписки не открывается новое окно клиента, каждое окно сообщений представлено в виде вкладки в едином окне. Изменения также коснулись реестра: теперь в кусте HKEY_CURRENT_USER\Software вы не найдете записей для настроек AOL или AIM.

Еще хуже то, что не было приложено никаких усилий, чтобы совместно создать каталог таких артефактов. В течение многих лет эксперты и сотрудники правоохранительных органов сталкиваются с ситуациями, когда им нужно анализировать артефакты IM- и P2P-приложений, тем не менее, никто не пытался разработать базу данных или вики-сайт этих элементов. Это область исследования, которую еще нужно развивать.

Совет

Для отслеживания действий пользователя можно использовать другие разделы реестра. Например, в разделе ShellNoRoam\BagMRU (<http://support.microsoft.com/kb/813711>) сохраняется информация о пользовательских параметрах представления и настройках вида для папок. Это означает, что пользователь должен иметь доступ к оболочке (т. е. по умолчанию к проводнику Windows, используя клавиатуру или приложение удаленного рабочего стола), чтобы сделать эти настройки. В разделах Explorer\Streams (<http://support.microsoft.com/kb/235994>) и StreamMRU сохраняется информация о размере и местоположении окна при его закрытии. Эти данные свидетельствуют о том, что пользователь выполнил определенные действия, чтобы изменить размер или расположение окна на рабочем столе, что также служит признаком отдельного взаимодействия пользователя с проводником Windows.

Точки восстановления системы в Windows XP

В ОС Windows XP есть функция «Восстановление системы» (“System Restore”), которая сохраняет несколько точек восстановления. Если операционная система станет непригодной для использования или начнет работать не так, как следует, пользователь может вернуть конфигурацию компьютера в предыдущее состояние, когда система работала правильно. Готов признать, что я сам неоднократно пользовался этой функцией. Время от времени я делаю (под «делаю» подразумевается «устанавливаю») что-то, что в итоге приводит мою систему к сбоям в работе. Кроме того, сбои могут быть вызваны и другими причинами. Нет ничего лучше возможности вернуть систему в более раннее, нормальное рабочее состояние. Я уверен, что многие пользователи в этом со мной согласны.

Эта возможность чрезвычайно полезна не только для пользователей, но и для судебных экспертов. Ведь она представляет собой функцию, которая работает в фоновом режиме и незаметно для пользователя создает резервные копии важных данных о конфигурации системы. Точки восстановления создаются на основе определенных условий или событий, таких как установка приложений, неподписанных драйверов или автоматических обновлений. Точки восстановления можно создавать вручную; кроме того, служба восстановления системы по умолчанию создает такие точки один раз в день.

Чтобы лучше понять, насколько полезны точки восстановления системы для судебной экспертизы, необходимо иметь представление о том, как работает функция восстановления: какие данные резервируются, а какие – нет, и какие разделы реестра управляют поведением функции «Восстановление системы».

Функция «Восстановление системы» восстанавливает следующие элементы:

- § реестр;
- § локальные (а не перемещаемые) профили;
- § базу данных COM+;
- § DLL-кэш защиты файлов Windows;
- § базу данных WMI;

- § метабазу IIS (Internet Information Server);
- § файлы с расширениями, перечисленными в части *<include>* списка контролируемых расширений файлов в разделе «Восстановление системы» («System Restore») пакета Platform SDK.

Функция «Восстановление системы» *не* восстанавливает:

- § параметры управления цифровыми правами (DRM);
- § куст SAM;
- § параметры WPA (информация о проверке подлинности Windows не восстанавливается);
- § отдельные каталоги и файлы, перечисленные в списке контролируемых расширений файлов в разделе «Восстановление системы» («System Restore») пакета Platform SDK;
- § любой файл с расширением, не указанным как *<included>* в списке контролируемых расширений файлов в разделе «Восстановление системы» («System Restore») пакета Platform SDK;
- § данные, созданные пользователем и сохраненные в профиле пользователя;
- § содержимое перенаправленных папок.

Необходимо отметить, что, несмотря на то, что функция восстановления системы не восстанавливает куст SAM, она все равно включает его (по крайней мере его часть) в резервную копию. Проверяя точки восстановления, вы увидите копии куста SAM вместе с другими файлами реестра.

Так как речь в этой главе идет о системном реестре, точки восстановления системы интересуют нас больше всего потому, что в них содержатся определенные файлы реестра, такие как NTUSER.DAT, SYSTEM, SOFTWARE и SAM. На илл. 4.22 показано содержимое каталога снимков точки восстановления, отображаемое в программе ProDiscover.

Select	File Name	File Exten...	Size
<input type="checkbox"/>	Repository		
<input type="checkbox"/>	ComDb	Dat	22512 bytes
<input type="checkbox"/>	domain	txt	36 bytes
<input type="checkbox"/>	_REGISTRY_MACHINE_SAM		24576 bytes
<input type="checkbox"/>	_REGISTRY_MACHINE_SECURITY		45056 bytes
<input type="checkbox"/>	_REGISTRY_MACHINE_SOFTWARE		9134080 bytes
<input type="checkbox"/>	_REGISTRY_MACHINE_SYSTEM		4558848 bytes
<input type="checkbox"/>	_REGISTRY_USER_DEFAULT	DEFAULT	241664 bytes
<input type="checkbox"/>	_REGISTRY_USER_NTUSER_S-1-5-18		241664 bytes
<input type="checkbox"/>	_REGISTRY_USER_NTUSER_S-1-5-19		229376 bytes
<input type="checkbox"/>	_REGISTRY_USER_NTUSER_S-1-5-20		229376 bytes
<input type="checkbox"/>	_REGISTRY_USER_NTUSER_S-1-5-21...		679936 bytes
<input type="checkbox"/>	_REGISTRY_USER_USRCLASS_S-1-5...		8192 bytes
<input type="checkbox"/>	_REGISTRY_USER_USRCLASS_S-1-5...		8192 bytes
<input type="checkbox"/>	_REGISTRY_USER_USRCLASS_S-1-5...		16384 bytes

Илл. 4.22. Каталог снимков точки восстановления, показанный в ProDiscover.

Как видите, с точки зрения анализа реестра, функция восстановления системы сохраняет довольно много полезной информации. Файлы реестра, которые сохраняются в

точках восстановления, содержат только фрагменты файлов, расположенных в каталоге system32\config, но они все равно могут дать эксперту представление о конфигурации системы в определенные моменты времени в прошлом.

Мы рассмотрим файлы, которые функции восстановления системы включает в резервную копию, в главе 5. В этой главе мы сосредоточимся на файлах реестра.

Наши методы анализа, особенно с использованием таких инструментов, как Offline Registry Parser, так же эффективны для файлов реестра, находящихся в точках восстановления, как и для файлов, которые можно найти в каталоге system32\config. В действительности многие разделы и параметры, обсуждавшиеся в этой главе, также находятся в копиях файлов реестра в точках восстановления. Это позволяет эксперту взглянуть в прошлое и узнать, какие параметры конфигурации и установленные программы присутствовали на компьютере в то время.

Однако во время анализа точек восстановления нужно обратить внимание на некоторые моменты. По умолчанию функция восстановления системы требует 200 Мб свободного пространства на накопителе. Если это требование не выполняется, функция восстановления системы переходит в состояние бездействия, пока это дисковое пространство не станет доступным. Этот факт нужно учитывать во время расследования, если вы не видите точки восстановления, которые ожидали найти в системе. Некоторые эксперты могут предположить, что кто-то смог получить доступ к каталогу «System Volume Information» и намеренно удалил точки восстановления, когда на самом деле они не создавались.

Еще один факт, о котором нужно знать во время работы с точками восстановления, – это то, что раздел SYSTEM\ControlSet00x\Control\BackupRestore также играет важную роль при определении того, какие данные включаются в резервную копию функцией восстановления системы. Этот раздел содержит три подраздела: AsrKeysNotToRestore, FilesNotToBackup и KeysNotToRestore, которые содержат разделы, не восстанавливаемые в процессе автоматического восстановления системы, файлы, не включаемые в резервную копию, и разделы, не восстанавливаемые функцией «Восстановление системы», соответственно. В каждом из подразделов хранится перечень разделов реестра или файлов (в случае с файлами вы увидите расширения с подстановочными знаками, что означает «все файлы с таким расширением»). Эти параметры и их данные также могут повлиять на то, что эксперт увидит, или к чему он получит доступ во время анализа образа данных.

Сведения о конфигурации функции «Восстановление системы» (<http://support.microsoft.com/kb/295659>) хранятся в следующем разделе:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestore

Здесь находится несколько важных параметров. Параметр *RPGlobalInterval* определяет, как часто создаются точки восстановления. Его значение по умолчанию равно 86400, что указывает ОС Windows XP создавать одну точку восстановления каждый календарный день ($60 \text{ сек.} \times 60 \text{ сек./час} \times 24 \text{ час./сут.} = 86400 \text{ сек.}$, или один календарный день). Если значение параметра *DisableSR* равно 1, значит функция «Восстановление системы» отключена. По умолчанию это значение равно 0. Параметр *RPLifeInterval* указывает время хранения точек восстановления (7776000 секунд = 90 дней).

Простой способ получить информацию о восстановлении системы на работающем компьютере с ОС Windows XP – использовать WMI-классы *SystemRestore* (<http://msdn.microsoft.com/en-us/library/aa378951.aspx>) и *SystemRestoreConfig* (<http://msdn.microsoft.com/en-us/library/aa378955.aspx>). Perl-скрипт «sr.pl», доступный на DVD-носителе, который идет в комплекте с этой книгой, показывает пример того, как можно использовать эти классы. Этот Perl-скрипт находит параметры конфигурации функции «Восстановление системы» (в основном, параметры реестра), которые доступны посредством WMI-класса *SystemRestoreConfig*, и показывает информацию о каждой точке

восстановления (т. е. порядковый номер, дату создания и строку, в которой описывается причина создания точки восстановления). Информацию, доступную посредством класса *SystemRestore*, можно получить из файлов в точках восстановления (т. е. файла «gr.log»); эта тема будет рассмотрена в главе 5.

Как файлы реестра в точках восстановления могут быть полезны с точки зрения эксперта? Файлы кустов реестра, хранящиеся в точках восстановления, содержат большую часть той же информации, что можно найти в самой работающей системе. Если у вас нет образа накопителя, и вы хотите увидеть, как выглядят эти файлы, загрузите на компьютер с ОС Windows XP утилиту «psexec.exe» от Microsoft, а затем введите команду **psexec -s cmd**. В результате откроется командная строка, запущенная от имени учетной записи системы, что в связи с разрешениями NTFS требуется, чтобы получить доступ к каталогу «System Volume Information». Перейдите в каталог «System Volume Information», используя следующую команду:

```
cd \sys*
```

Затем перейдите в подкаталог, содержащий точки восстановления:

```
cd _restore*
```

Если на этом этапе вы выведете список каталогов, то увидите несколько точек восстановления, которые перечислены как имена каталогов, начинающиеся с символов RP. Если (с помощью команды cd) вы войдете в один из этих каталогов, а затем перейдете в подкаталог снимков, то увидите файлы реестра. Отсюда можно скопировать любой из этих файлов в другой каталог для анализа. Один из способов просмотра информации в файлах кустов реестра – открыть редактор реестра и выбрать куст **HKEY_USERS**. В меню «Файл» (“File”) выберите команду «Загрузить куст» (“Load Hive”), а затем перейдите к одному из файлов кустов, который вы скопировали из точки восстановления. Далее нужно указать имя для куста, например «**test hive**» или «**test system**» (если используете файл System). На илл. 4.23 показан файл куста System, загруженный таким способом.



Илл. 4.23. Куст System, загруженный в редактор реестра из точки восстановления.

Теперь можно просматривать содержимое разделов и даже запускать инструменты анализа реестра, чтобы извлечь параметры и данные так же, как вы бы это сделали на работающем компьютере. Кроме того, можно экспорттировать параметры из куста.

Еще один способ просмотреть файлы реестра – использовать один из инструментов программы RegRipper, упоминавшийся ранее в этом главе, который называется «*triphpl*».

Это специальная версия «rip.pl», предназначенная для работы с точками восстановления Windows XP, которые находятся в образах клонированных данных. Инструмент «rip.pl» исследует файл куста, запуская один подключаемый модуль, а затем получает доступ к точкам восстановления и запускает тот же модуль для анализа соответствующего файла куста, расположенного в точках восстановления. (На данный момент «ripxp.pl» запускает только один модуль за раз, так как запуск нескольких модулей, перечисленных в файле модулей, может привести к тому, что будет показано слишком много информации.) Инструмент выполняет все операции автоматически, чтобы уменьшить вероятность возникновения ошибок и увеличить производительность для эксперта.

Перед запуском «ripxp.pl» необходимо выполнить определенные действия. Сначала монтируйте созданный образ в режиме только для чтения, используя одну из имеющихся бесплатных (VDKWin или ImDisk) или коммерческих (Smart Mount или Mount Image Pro) утилит; дополнительные сведения об этих средствах см. в главе 5. В этом примере я монтировал образ на испытательном компьютере как накопитель H:\. Затем откройте командную строку и введите команду **psexec -s cmd**; как уже упоминалось выше, это необходимо, чтобы получить доступ к файлам точек восстановления в каталоге «System Volume Information».

На этом этапе рекомендуется открыть еще одну командную строку с помощью команды psexec.exe, которая уже использовалась, а затем перейти в каталог точек восстановления в монтированном образе, используя команды, указанные ранее в этой главе. После того как я выполнил эти действия, выходные данные команды *dir*, примененной к моему тестовому образу, выглядят так, как показано ниже (некоторые строки перенесены):

```
H:\System Volume Information\_restore{506D8C9A-F73D-497E-AAD1FD40F23F5B51}>dir
Volume in drive H has no label.
Volume Serial Number is B0A6-5D8E
Directory of H:\System Volume Information\_restore{506D8C9A-F73D-497E AAD1-FD40F23F5B51}
01/30/2008 09:51 AM           130 drivetable.txt
01/30/2008 11:33 PM    <DIR>          RP0
01/30/2008 11:43 PM    <DIR>          RP1
01/30/2008 09:09 AM    <DIR>          RP2
01/30/2008 09:47 AM    <DIR>          RP3
01/30/2008 09:51 AM           24 _driver.cfg
01/30/2008 11:33 PM           22,712 _filelst.cfg
                           3 File(s)           22,866 bytes
                           4 Dir(s)        380,102,656 bytes free
```

Итак, похоже, что в монтированном образе есть три точки восстановления. Теперь вернитесь в первую открытую командную строку и перейдите в каталог, где хранится утилита «ripxp.pl» и папка модулей. Введите **ripxp.pl** (или **ripxp**, если вы используете скомпилированную версию, работающую в Windows), чтобы увидеть синтаксис для этого инструмента командной строки. Мы начнем с того, что запустим простой запрос с целью узнать имя компьютера, используя модуль «compname.pl». Команда, которую нужно ввести, выглядит так:

```
C:\Perl\forensics\rr>ripxp.pl -d "H:\System Volume Information\_restore{506D8C9A-F73D-497E-AAD1-FD40F23F5B51}"
-r H:\Windows\system32\config\System -p compname
```

Как видите, лучше скопировать и вставить путь к каталогу точек восстановления из одной командной строки в другую или сохранить его в текстовом документе для последующего использования. Команда довольно длинная, хотя состоит только из трех

элементов. Так как путь к каталогу точек восстановления содержит пробелы, нужно взять его в кавычки.

Но на самом деле нас интересуют выходные данные команды, которые показаны ниже:

```
RipXP v.20081001
Launched Tue Jan 13 01:18:44 2009 z
H:\Windows\system32\config\System
ComputerName = ACME-N6A1H8ZLJ1
-----
Restore Point Info
Description : System Checkpoint
Type : System Checkpoint
Creation Time : Thu Jan 31 04:33:11 2008

H:\System Volume Information\_restore{506D8C9A-F73D-497E-AAD1-
FD40F23F5B51}\RP1\snapshot\_REGISTRY_MACHINE_SYSTEM
ComputerName = ACME-N6A1H8ZLJ1
-----
Restore Point Info
Description : Installed VMware Tools
Type : Application Install
Creation Time : Thu Jan 31 04:43:38 2008

H:\System Volume Information\_restore{506D8C9A-F73D-497E-AAD1-FD40F23F5B51}\
RP2\snapshot\_REGISTRY_MACHINE_SYSTEM
ComputerName = ACME-N6A1H8ZLJ1
-----
Restore Point Info
Description : Installed WinZip 11.1
Type : Application Install
Creation Time : Wed Jan 30 14:09:49 2008

H:\System Volume Information\_restore{506D8C9A-F73D-497E-AAD1-
FD40F23F5B51}\RP3\snapshot\_REGISTRY_MACHINE_SYSTEM
ComputerName = ACME-N6A1H8ZLJ1
```

Как упоминалось выше, утилита «ripxp.pl» начинает работу с запуска выбранного модуля («compname.pl») для анализа файла куста (т. е. куста System, который находится в каталоге по умолчанию). Затем она получает доступ к каждой точке восстановления, анализирует файл «gr.log» (дополнительные сведения о формате этого файла см. в главе 5), показывает информацию о точке восстановления (т. е. причину и дату ее создания), после чего запускает модуль для анализа соответствующего файла куста («ripxp.pl» содержит код, который позволяет определить отдельный тип файла куста).

Утилиту «ripxp.pl» можно использовать не только для анализа файла куста System. Например, с помощью нижеупомянутой команды можно запустить модуль «userassist.pl» для анализа файла «NTUSER.DAT», а также всех соответствующих файлов «NTUSER.DAT» в точках восстановления.

```
C:\Perl\forensics\rr>ripxp.pl -d "H:\System Volume Information\_restore
{506D8C9AF73D-497E-AAD1-FD40F23F5B51}"
-r "H:\Documents and Settings\username\NTUSER.DAT" -p userassist
```

Или можно использовать следующую команду, чтобы запустить модуль «auditpol.pl» для анализа всех файлов куста Security как в каталоге system32\config, так и в точках восстановления:

```
C:\Perl\forensics\rr>ripxp.pl -d "H:\System Volume Information\_restore
{506D8C9AF73D-497E-AAD1-FD40F23F5B51}"
```

```
-r H:\Windows\system32\config\Security -p auditpol
```

Теперь вам должно быть понятно, что из точек восстановления в ОС Windows XP можно извлечь большое количество данных за прошедшие периоды времени. Приведу пример, в котором объясняется, почему вам может понадобиться такой инструмент, как «*grip.pl*»: предположим, вы расследуете дело и подозреваете, что некая программа была удалена из системы. Настроив компьютер, используемый для анализа данных, как указано выше, и применив утилиту «*grip.pl*», вы сможете запустить модуль «*comprname.pl*», чтобы определить, когда было изменено имя компьютера, или модуль «*tmicache.pl*», чтобы установить, когда исполняемый файл впервые запускался на компьютере. «*Ripxp.pl*» – очень мощный инструмент, который работает с теми же подключаемыми модулями, что RegRipper и «*rip.pl*», и который можно использовать для просмотра данных системы за прошедшие периоды и даже, возможно, для наблюдения за изменениями, произошедшими со временем. Например, вы подозреваете, что пользователь отключил аудит или спящий режим в течение определенного периода времени, или удалил несколько важных записей в файле куста «*NTUSER.DAT*». Вы можете быстро и легко просмотреть эти данные и получить более полное представление о действиях пользователя.

Перенаправление реестра

Когда эксперт имеет дело с 64-разрядной версией ОС Windows, ему следует помнить, что в этой операционной системе применяется так называемое перенаправление реестра. Это означает, что в 64-разрядных версиях Windows данные реестра, относящиеся к 32-разрядным приложениям, хранятся в отдельном месте в файле куста *Software*. Согласно статье № 305097 из базы знаний Microsoft (<http://support.microsoft.com/kb/305097>), данные 32-разрядных приложений записываются в следующий раздел:

HKLM\Software\WOW6432Node

Похоже, что файлы кустов реестра из 64-разрядных версий Windows не отличаются от файлов кустов из 32-разрядных версий Windows на двоичном уровне, но различия в разделах реестра, такие как в случае с перенаправлением, могут осложнить поиск по реестру или использование таких инструментов, как RegRipper, если эксперт не будет учитывать этот раздел.

Виртуализация

С появлением ОС Windows Vista корпорация Microsoft начала использовать механизм виртуализации в реестре. В предыдущих версиях Windows многие приложения выполнялись от имени администратора, и если обычный пользователь пытался запустить это приложение, то возникали ошибки, связанные с недостатком прав у пользователя. В ОС Windows Vista корпорация Microsoft решила реализовать механизм виртуализации, в результате чего операции записи в реестре, которые могут повлиять на работу всей системы (т. е. записи в куст *Software*), будут выполняться в области, выделенной для пользователя. Это не позволяет потенциально опасным операциям в реестре (например, создание или изменение разделов и параметров) воздействовать на всю систему в целом. Например, если приложение пытается выполнить запись в раздел HKLM\Software\имя_приложения, оно будет автоматически перенаправлено в следующий раздел:

HKEY_USERS\{SID_пользователя}_Classes\VirtualStore\Machine\Software\

Этот раздел, в свою очередь, не существует в пользовательском файле куста «NTUSER.DAT», вместо этого с ним сопоставляется нижеуказанный файл, когда пользователь входит в систему:

```
%UserProfile%\AppData\Local\Microsoft\Windows\usrclass.dat
```

Согласно статье Марка Руссиновича о службе контроля учетных записей в ОС Windows Vista (<http://technet.microsoft.com/en-us/magazine/2007.06.uac.aspx>), «виртуализируются только разделы, которые обычно изменяются устаревшими приложениями». Изменения в виртуализированных разделах (такие разделы, как Software\Classes, Software\Microsoft\Windows и Software\Microsoft\Windows NT исключены из виртуализации) перенаправляются в пользовательский файл «usrclass.dat». Этот файл – действительный файл куста на двоичном уровне; его можно открыть многими программами для просмотра реестра, и к нему можно получить доступ с помощью Perl-модуля Parse::Win32Registry. Однако на момент написания этой книги ни один из подключаемых модулей, входящих в состав дистрибутива RegRipper, не был специально предназначен для извлечения информации из этого файла куста.

Удаленные разделы реестра

В начале этой главы мы говорили о том, что необходимо понимать структуру разделов реестра, так как эксперт может найти признаки разделов в свободном пространстве накопителя, файле подкачки, дампе памяти и т. д. Знание структуры разделов позволит ему преобразовать найденные данные в формат, доступный для чтения, что добавит контекст к ранее полученной информации. Еще одно место, где можно найти разделы реестра, которые не видны в редакторе реестра или других инструментах анализа, находится в свободном пространстве самих файлов кустов.

Как было показано ранее на илл. 4.3, разделы реестра можно определить по сигнатуре *nk*. Весной 2008 года Йоланта Томассен (Jolanta Thomassen) установила, что если двойное слово, предшествующее сигнатуре раздела, интерпретировать как длинное целое без знака, то для большинства действительных разделов реестра, это значение будет отрицательным числом. (Применяя Perl-функцию *unpack()*, значение нужно интерпретировать с помощью символа *L*, а не *V*, который во многих случаях используется для распаковки значений двойного слова). Это относится к разделам реестра, отображаемым в редакторе реестра и других подобных инструментах. Йоланта также обнаружила сигнатуры для предположительно действительных разделов, для которых предшествующее значение двойного слова было положительным; ее исследование показало, что на самом деле это были удаленные разделы реестра, постоянно находящиеся в самом файле куста.

Исследование Йоланты Томассен было частью ее дипломной работы в Ливерпульском университете. Я встречал Йоланту на конференции DFRWS 2008, которая проходила в Балтиморе в августе 2008 года. В конференции также принимал участие Тим Морган (Tim Morgan), который проводил свое, не связанное с работой Йоланты, исследование удаленных разделов в файлах кустов реестра. Исследование Тима, которое называется «Recovering Deleted Data from the Windows Registry», и его презентационные слайды можно найти на сайте конференции DFRWS 2008 (www.dfrws.org/2008/program.shtml).

Йоланта выпустила работающий Perl-код, который обрабатывает файл куста, извлекая и отображая информацию об удаленных разделах и параметрах реестра, а также о резервном пространстве в файле куста. (Файл куста должен быть извлечен из образа клонированных данных или доступен в образе, который монтирован в режиме только для чтения, или доступен посредством программы F-Response, созданной Мэттом Шенноном (Matt Shannon).) Perl-скрипт Йоланты Томассен «regslack.pl» (вместе со

скомпилированной версией этого инструмента для Windows) включен в каталог ch4\code\jt на носителе, который идет в комплекте с этой книгой.

Для того чтобы запустить скрипт «regslack.pl», просто введите в командной строке следующее:

```
C:\perl\jt>regslack.pl d:\cases\SAM > sam.txt
```

Скрипт «regslack.pl» не требует дополнительных модулей Perl, поэтому его можно запустить, используя только основной интерпретатор языка Perl (доступен на веб-сайте ActiveState.com). Вышеуказанная команда перенаправляет выходные данные скрипта из консоли в файл с именем «sam.txt». Рекомендуется выполнять эту команду, так как скрипт отправляет на консоль большое количество информации, поэтому копирование этих данных в файл делает их просмотр удобнее, а также сохраняет их для последующего анализа. Часть файла «sam.txt» с данными восстановленного раздела реестра из файла SAM показана ниже:

```
SAM\SAM\Domains\Account\Users\000003F7
Offset: 0x4ed8 [Thu Sep 13 12:31:00 2007]
Number of values: 2
Offset: 0x5a30 -->REG_BINARY; F;
02 00 01 00 00 00 00 84 56 8d 99 7a 69 c8 01 .....V..zi...
00 00 00 00 00 00 00 04 d4 d7 ad 73 51 c8 01 .....sQ..
00 00 00 00 00 00 00 34 e9 55 e2 81 53 c8 01 .....4.U..S..
f8 03 00 00 01 02 00 00 10 02 00 00 00 00 00 00 .....
00 00 c7 01 01 00 00 00 00 ff ff eb 06 91 7c .....|
Offset: 0x4330 -->REG_BINARY; V;
```

В этом примере скрипт «regslack.pl» нашел раздел реестра, относящийся к пользователю, а также содержимое параметров *F* и *V* (показано имя параметра *V*, но, для краткости, все двоичное содержимое данных этого параметра не отображается). В выходных данных выше можно увидеть имя раздела и восстановленный путь к нему, смещение, в котором находился раздел, а также время *LastWrite* этого раздела. Помимо имен и данных для этих параметров, показаны смещения самих параметров, что позволяет выполнить визуальный контроль. Скрипт «regslack.pl» также показывает резервное пространство в файле куста.

Извлекая данные этого типа во время экспертизы, можно найти очень много информации. Как вы поняли из этой главы, ряд разделов и параметров в реестре предоставляет сведения о присутствии вредоносных программ, действиях пользователя и т. д. Злоумышленники могут удалить разделы реестра, пытаясь замести свои следы, а пользователи могут попытаться скрыть информацию, свидетельствующую о несанкционированных действиях. Однако в некоторых случаях, даже если сами разделы реестра удалены, вы можете восстановить эти разделы из файла куста.

Краткое изложение

Знания о том, как осуществлять поиск и просмотр определенной информации в реестре, могут оказаться чрезвычайно ценными навыками для администраторов, консультантов и судебных экспертов. Реестр охотно поделится своими секретами с теми, кто знает, где искать и как интерпретировать найденные данные. Исследование внутренней структуры реестра чем-то похоже на то, как Индиана Джонс охотился за древними тайнами в зыбучих песках времени.

В таблице «regref.xls», включенной в состав носителя, который идет в комплекте с этой книгой, вы найдете подробный (по вполне очевидным причинам я не решаюсь использовать слово «полный») список мест автозапуска и пользовательских списков MRU в реестре. Несмотря на то, что список составлен очень подробно, его следует рассматривать как отправную точку для исследования. Перечисленные разделы реестра были получены из списков в Интернете, из таких приложений, как Autoruns, из отчетов о вредоносном ПО на веб-сайтах антивирусных программ и из личного опыта.

Эту главу не следует рассматривать как полную справочную информацию обо всех разделах реестра, которые могут быть важны для конкретного дела. Разделы реестра со схожей функциональностью могут иметь разные имена и разное местонахождение в зависимости от приложений, а в некоторых случаях в зависимости от версий одного и того же приложения. Ни одну книгу нельзя считать полным авторитетным справочным источником о разделах реестра, представляющих интерес для эксперта; она была бы слишком большой и дорогой, а информация в ней почти сразу устарела бы в день публикации. Цель этой главы – показать, какая информация доступна и как находить дополнительные данные в реестре.

Содержимое DVD-диска

Носитель, который идет в комплекте с этой книгой, содержит каталог, предназначенный специально для этой главы. Каталог включает в себя несколько подкаталогов, один из которых содержит код, рассматриваемый в этой главе, а также дистрибутив RegRipper (в том числе утилиту «rip.pl» и все подключаемые модули; утилита «grip.pl», как было указано выше, не входит в состав дистрибутива). Кроме того, носитель содержит все Perl-скрипты из первого издания книги (в подкаталоге «old»), а также скрипты ProScript (Perl-скрипты, написанные специально для использования с программой судебной экспертизы ProDiscover), обсуждавшиеся в этой главе. В подкаталоге «spreadsheet» находится копия справочной таблицы по реестру, которую я составил некоторое время тому назад. В таблице перечисляются несколько разделов и параметров реестра, представляющих интерес для судебного эксперта, а также дается краткое описание их назначения и, при необходимости, справочная информация. Таблица разделена на несколько листов, каждый из которых охватывает отдельную область функциональных возможностей.

Кроме того, на диске есть каталог «samples», содержащий файлы реестра из настоящих систем – не только файлы, которые находятся в каталоге system32\config, но и файлы из точек восстановления. Я рекомендую вам изучить эти файлы, открыть их в шестнадцатеричном редакторе и использовать любые инструменты, включенные в подкаталог «code» для этой главы, чтобы вы смогли познакомиться как с этими инструментами, так и с самими необработанными файлами реестра.

Быстрое повторение

Внутри системного реестра

- § Реестр ОС Windows – это двоичная, иерархическая база данных, содержащая сведения о конфигурации. Реестр управляет разнообразными параметрами конфигурации операционной системы и приложений и хранит информацию о различных аспектах взаимодействия пользователя с системой.
- § Понимая формат различных структур реестра (т. е. разделов и параметров), вы сможете анализировать и просматривать части реестра, которые находятся в оперативной памяти и свободном пространстве накопителя.
- § Некоторые части реестра энергозависимы; они создаются при запуске системы, и их нельзя найти в файле-образе данных.
- § Разделы реестра (и некоторые параметры) имеют связанные с ними отметки времени, которые можно использовать во время анализа временной шкалы. Поэтому реестр можно рассматривать как своеобразный файл журнала.
- § Не существует единого стандарта для способа хранения информации в реестре. Одни разделы списков MRU, например, сохраняют свои параметры в виде двоичных данных, другие – в виде строк ASCII (что значительно облегчает поиск ASCII строк). Кроме того есть разделы, в которых имена параметров зашифрованы по алгоритму ROT-13, или разделы, в которых поиск строк затруднен из-за особого способа сохранения данных в параметрах. Необходимо понимать структуру отдельных разделов и параметров, чтобы уметь анализировать хранящиеся там данные.
- § Существуют инструменты, предназначенные для отслеживания доступа к реестру и изменений в реестре на работающем компьютере; эту информацию можно использовать для поиска важных разделов реестра, а также для определения артефактов, оставляемых в результате работы приложений и действий пользователя.

Анализ системного реестра

- § Несколько мест в реестре содержат информацию, которая относится к большинству расследований. Другие области реестра содержат информацию, связанную с отдельными типами расследований, такими как вторжение, мошенничество или нарушение правил использования сети.
- § Отдельные разделы и параметры реестра могут играть важную роль в расследовании, но часто именно взаимосвязь нескольких разделов и параметров предоставляет наиболее полную картину действий пользователя в системе.
- § Точки восстановления системы в ОС Windows XP содержат части реестра, которые могут быть полезны во время расследования. Например, исследуя содержимое сохраненного файла SAM, эксперт может определить, когда изменилось членство пользователя в группах (в частности, с группы «Пользователи» на группу «Администраторы»), если это имеет отношение к расследованию. Кроме того, можно узнать, какие приложения устанавливались на компьютере в последнее время.

Часто задаваемые вопросы

Вопрос: Как определить, какие объекты модуля поддержки браузера (ВНО) установлены?

Ответ: Объекты ВНО содержатся в кусте HKEY_LOCAL_MACHINE, то есть они влияют на работу всех пользователей в системе. Объекты ВНО перечислены в разделе Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects. В этом разделе каждый объект ВНО перечислен как подраздел с именем в виде глобально уникального идентификатора (GUID). Отсюда можно перейти в раздел Software\Classes\CLSID в кусте HKEY_LOCAL_MACHINE и найти каждый идентификатор GUID. После того как вы найдете раздел с таким же идентификатором GUID, что и у объекта ВНО, проверьте параметр «По умолчанию» (“Default”) этого раздела, где будет указано имя объекта ВНО. Чтобы узнать, какой DLL файл связан с объектом ВНО, проверьте параметр «По умолчанию» (“Default”) подраздела InProcServer. Perl-скрипт «bho.pl», находящийся на DVD-диске, который идет в комплекте с этой книгой, можно использовать для поиска информации об объектах ВНО в локальной системе.

Вопрос: Выполняя поиск в системе, я нашел в пользовательском кусте раздел реестра (Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains), который имеет несколько подразделов с доменными именами веб-сайтов. Что это за раздел и что представляют эти подразделы?

Ответ: Эти записи можно добавить в браузере Internet Explorer, перейдя в меню «Сервис» (“Tools”) | «Свойства обозревателя» (“Internet Options”) | «Безопасность» (“Security”) | «Ограниченные узлы» (“Restricted Sites”) и нажав кнопку «Узлы» (“Sites”). Внимательно изучите перечисленные записи, так как некоторые вредоносные программы добавляют в этот раздел веб-сайты, чтобы пользователь не мог получить к ним доступ. Хотя записи в этом разделе могут свидетельствовать о действии администратора или применении групповых политик, они также могут означать заражение вредоносной программой.

Вопрос: Во время расследования выяснилось (на основе информации из ОС Windows XP, установленных программ и т. д.), что пользователь компьютера имеет права локального администратора. Обсудив этот вопрос с начальником отдела информационных технологий, я обнаружил, что всем сотрудникам предоставляется доступ к компьютерам только на уровне пользователя. Как узнать, когда пользователь был добавлен в группу «Администраторы» на компьютере?

Ответ: Такая информация о пользователях хранится в файле SAM. Время *LastWrite* раздела реестра, в котором хранится информация о членстве в группах, может предоставить вам ключи к решению этой задачи. Кроме того, в точках восстановления системы в ОС Windows XP есть достаточно информации за прошедшие периоды времени, в которой можно найти, когда идентификатор RID пользователя был последний раз связан с группой «Пользователи», а не с группой «Администраторы».

Содержание

Введение	2
Внутри системного реестра	3
Структура реестра в файле куста	6
Системный реестр как файл журнала	10
Наблюдение за изменениями в реестре	11
Анализ системного реестра	13
RegRipper	14
Rip	17
RipXP	21
Информация о системе	21
Имя компьютера (ComputerName)	22
Сведения о часовом поясе (TimeZoneInformation)	24
Сетевые интерфейсы	24
MAC-адрес	25
Общие ресурсы	26
Политика аудита и журналы регистрации событий	27
Идентификаторы SSID беспроводной сети	30
Места автоматического запуска	32
Начальная загрузка системы	34
Вход пользователя в систему	36
Действия пользователей	36
Перечисление мест автозапуска в реестре	40
Функция автоматического запуска носителей	41
Параметр NtfsDisableLastAccessUpdate	43
Параметр NukeonDelete	43
Съемные USB-носители	44
Проблемы, связанные с USB-устройствами	48
Монтируемые устройства	50
Переносные устройства	54
Поиск сведений о пользователях	55
Отслеживание действий пользователя	58
Разделы UserAssist	59
Раздел MUICache	62
Списки MRU	63
Помощник по поиску	68
Подключение к другим компьютерам	69
Запись компакт-дисков	70
Обмен мгновенными сообщениями и одноранговые сети	71
Точки восстановления системы в Windows XP	72
Перенаправление реестра	78
Виртуализация	78
Удаленные разделы реестра	79
Краткое изложение	81
Содержимое DVD-диска	81
Быстрое повторение	82
Часто задаваемые вопросы	83



<http://computer-forensics-lab.org>

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