Практическое создание криминалистических образов

Русский перевод практического руководства Брюса Никкеля по безопасному созданию, проверке и управлению криминалистическими образами носителей с помощью инструментов Linux.

Больше крутых материалов в моём телеграм-канале @grogrigs

Предисловие

Книга «Практическое создание криминалистических образов» очень нужна и выходит как нельзя более своевременно. В последние годы сохранение цифровых доказательств приобрело важнейшее значение для корпоративного управления, соблюдения нормативных требований, уголовного и гражданского судопроизводства, а также военных операций. Эта тенденция не ограничена географически: она наблюдается на большинстве континентов, в том числе в развивающихся странах.

Компетентные организации сохраняют имеющие отношение к делу компьютерные системы при разборе жалоб сотрудников, нарушений правил и случаев прекращения трудовых отношений. Некоторые организации даже сохраняют данные заблаговременно, особенно ради соблюдения нормативных требований. В этой книге представлены масштабируемые решения, которые можно внедрить в масштабах предприятия с разумными затратами.

Цифровые доказательства фигурируют в большинстве уголовных дел, и ответственность за сохранение данных всё чаще ложится на небольшие правоохранительные органы, располагающие ограниченными ресурсами или недостаточной подготовкой. Для таких ведомств книга «Практическое создание криминалистических образов» станет бесценным источником практических решений повседневных задач.

Гражданские дела могут затрагивать огромные объёмы данных, распределённых между множеством источников, включая компьютеры, серверы, съёмные носители и ленты резервного копирования. В подобных обстоятельствах необходимы производительные и действенные методы, и эта книга отвечает также и этим требованиям.

Поскольку сохранение цифровых доказательств приобретает всё большее значение во множестве областей, критически важно применять надлежащие процедуры сохранения. Недостатки такой процедуры способны породить проблемы на всех последующих этапах цифрового расследования, тогда как доказательства, сохранённые с помощью криминалистически корректных методов и средств, создают основу для построения убедительного дела.

Кроме того, растущая потребность в сохранении цифровых доказательств повышает спрос на инструменты, которые надёжны, доступны по цене и пригодны к адаптации для различных сред и сценариев применения.

«Практическое создание криминалистических образов» отвечает этим требованиям благодаря сосредоточенности на технологиях с открытым исходным кодом. У таких инструментов есть следующие преимущества: высокая прозрачность, низкая стоимость и возможность адаптации. Прозрачность позволяет другим специалистам тщательнее оценивать надёжность инструментов с открытым исходным кодом. Помимо тестирования по принципу чёрного ящика с применением известных наборов данных, можно проанализировать и сам исходный код.

Снижение стоимости криминалистического сохранения важно как для ведомств с ограниченными ресурсами, так и для организаций, которым приходится обрабатывать большие объёмы данных.

Возможность приспособить инструменты с открытым исходным кодом к потребностям конкретной среды даёт существенное преимущество. Одни организации встраивают такие инструменты и средства сохранения в автоматизированные процессы предприятия или криминалистической лаборатории, а другие развёртывают те же инструменты на переносных системах для работы на месте происшествия.

Освоение любых процессов и инструментов цифровой криминалистики требует значительных усилий, особенно когда речь идёт об инструментах с открытым исходным кодом. Обширный опыт и глубокие знания Брюса Никкеля проявляются в поразительной ясности технического материала этой книги: благодаря ей текст доступен начинающим и одновременно интересен специалистам.

Начав с теории и основных требований к созданию криминалистических образов, книга затем углубляется в технические аспекты получения таких образов с помощью инструментов с открытым исходным кодом. Применение SquashFS - простое, но весьма остроумное и оригинальное решение, которое предлагает практический открытый подход к одной из центральных задач криминалистического копирования. Завершается книга обсуждением важных этапов управления криминалистическими образами и их подготовки к исследованию.

«Практическое создание криминалистических образов» - незаменимый справочник для всех, кто отвечает за сохранение цифровых доказательств, включая коммерческие организации, правоохранительные органы и структуры по борьбе с терроризмом.

Эоган Кейси, PhD
профессор киберпреступности и цифровых расследований
Школа криминальных наук
Факультет права, криминальных наук и государственного управления
Лозаннский университет, Швейцария
август 2016 года

Введение

Добро пожаловать в книгу «Практическое создание криминалистических образов: сохранение цифровых доказательств с помощью инструментов Linux». В ней рассматриваются различные приёмы работы в командной строке, предназначенные для получения дисковых образов, используемых в качестве цифровых доказательств, и управления ими. Получение дисковых образов - первый этап сохранения цифровых криминалистических доказательств при подготовке к посмертному исследованию и анализу.

Почему я написал эту книгу

Сегодня на рынке представлено множество книг по цифровой криминалистике. Однако важности криминалистического получения данных и сохранения доказательств обычно уделяется слишком мало внимания. Нередко эта тема лишь бегло рассматривается в небольших главах или подразделах более объёмной книги. Мне показалось, что получение данных и сохранение доказательств - достаточно обширная тема для отдельной книги, и настоящая работа восполняет этот пробел в литературе.

Ещё одним побудительным мотивом стало моё желание каким-либо образом отплатить профессиональному сообществу. Более десяти лет проработав в лаборатории цифровой криминалистики и регулярно применяя инструменты с открытым исходным кодом для различных задач наряду с коммерческими средствами, я захотел предоставить коллегам и другим специалистам дополнительный источник информации.

Третьей причиной стала возрастающая роль сохранения криминалистических доказательств в частном секторе. Расследования должностных нарушений, мошенничества, деятельности вредоносных программ, кибератак и других злоупотреблений становятся всё более обычным делом в частных компаниях. Тем не менее этапам, необходимым для получения и сохранения доказательств, зачастую не придают должного значения. Чтобы привлечь преступников к ответственности, правоохранительным органам нужны доказательства, полученные и сохранённые надлежащим образом. В гражданских делах, связанных с раскрытием электронной информации, может потребоваться корректное получение и сохранение дисковых образов. Следование признанным процедурам сбора и сохранения цифровых доказательств полезно и крупным организациям, внутренние подразделения которых разбирают трудовые споры, нарушения правил и сообщения информаторов.

Чем отличается эта книга

Эта книга представляет собой техническое руководство по выполнению процедур. В ней объясняется применение Linux как платформы компьютерной криминалистики, в особенности для получения криминалистических образов носителей и сохранения доказательств. Я включил примеры, демонстрирующие известные криминалистические методы с использованием свободных или открытых средств компьютерной криминалистики для получения данных с самых разных исследуемых носителей.

В отличие от книг по криминалистике в Linux, охватывающих широкий круг тем, связанных с анализом приложений и операционных систем, эта книга сосредоточена на одной конкретной области компьютерной криминалистики: криминалистическом получении данных, также

называемом созданием криминалистических образов носителей. Сюда входят подготовка, получение, сохранение цифровых доказательств с различных типов носителей и управление ими. Именно корректность получения данных с носителя делает этот процесс «криминалистическим».

Наряду с инструментами с открытым исходным кодом книга содержит примеры использования нескольких проприетарных инструментов командной строки, которыми можно пользоваться бесплатно, хотя их исходный код закрыт.

Я рассматриваю некоторые новые аппаратные технологии, которые пока не успели войти в другие книги по криминалистике. Среди них NVME и SATA Express, диски с собственными секторами по 4 Кбайт, гибридные SSD, SAS, UASP/USB3x, Thunderbolt и другие. С одними из них сравнительно легко работать в контексте цифровой криминалистики, другие создают более сложные задачи.

Я также представляю новый криминалистический метод, в котором сжатая файловая система SquashFS используется как простой и практичный контейнер криминалистических доказательств. Вместе с книгой предоставляется сценарий оболочки `sfsimage`, позволяющий сохранять доказательства в криминалистических контейнерах SquashFS.

Почему командная строка

Почему книга, основанная на командной строке, вообще полезна или актуальна сегодня? Компьютерная командная строка существует со времён телетайпов 1960-х годов, то есть ей уже более полувека. Хотя в вычислительной технике возраст порой считают признаком устаревания, он может свидетельствовать и о зрелости с надёжностью - именно так обстоит дело с командной строкой Linux/Unix. Даже Microsoft признала ценность и мощь командной строки, представив и продвигая PowerShell как альтернативу устаревающей командной строке DOS.

Есть немало причин, по которым командная строка за прошедшие годы не утратила популярности и по-прежнему актуальна для тем этой книги. Вот несколько примеров:

• **Более широкие возможности создания сценариев и автоматизации.** Графический интерфейс предназначен для человека, тогда как командной строкой может пользоваться и человек, и машина. Благодаря этому она особенно удобна для написания сценариев и автоматизации работы.
• **Более глубокое понимание внутренних механизмов.** Графические инструменты зачастую представляют собой лишь оболочки для средств командной строки. Изучение последних помогает понять, что происходит внутри при работе с графическими оболочками.
• **Гибкость и эффективность.** Выполняя определённые задачи в командной строке, вы получаете больше гибкости, возможностей и контроля. Например, конвейеры и перенаправление позволяют объединить несколько этапов в одной командной строке.
• **Философия Unix.** Традиционная философия Unix состоит в создании простых инструментов, каждый из которых хорошо выполняет одну задачу, тогда как крупные графические программы объединяют богатую и сложную функциональность в одном монолитном приложении.

• **Удалённый доступ.** Работать в командной строке удалённо по `ssh` легко и безопасно. Иногда удалённый доступ к оболочке остаётся единственным вариантом, особенно при работе с виртуальными или облачными серверами либо системами, находящимися в других городах или странах.
• **Серверы без монитора.** На серверах Unix и Linux, где произошёл инцидент, командная строка может оказаться единственным вариантом, поскольку графический интерфейс мог быть вообще не установлен.
• **Встраиваемые системы.** Во всё более популярных встраиваемых системах Unix и Linux, таких как Raspberry Pi, Beagleboard и другие устройства Интернета вещей, может быть доступен лишь интерфейс командной строки.
• **Вложения в знания.** В отличие от графических средств, инструменты командной строки со временем меняются незначительно. Если вы потратите время на освоение такого инструмента, вам не придётся заново изучать всё после его обновления или добавления новых возможностей.
• **Личные предпочтения.** Некоторые технические специалисты просто предпочитают командную строку графическому интерфейсу и при наличии выбора пользуются именно ею.

Эта книга служит руководством по применению командной строки для криминалистического получения цифровых данных в ходе расследований и реагирования на инциденты. Эквивалентные графические средства и оболочки в ней не рассматриваются.

Целевая аудитория и предварительные требования

При работе над книгой я ориентировался на определённую аудиторию. В ходе написания многих разделов я исходил из ряда ожиданий и предположений.

Кому следует прочесть эту книгу

Книга прежде всего принесёт пользу двум группам читателей. Во-первых, она поможет опытным криминалистам усовершенствовать навыки работы в командной строке Linux при получении цифровых данных. Во-вторых, она пригодится опытным администраторам Unix и Linux, желающим освоить методы криминалистического получения данных.

Книга адресована растущему числу практикующих специалистов из самых разных областей: участникам групп реагирования на инциденты; специалистам по компьютерной криминалистике в крупных организациях; техническим специалистам по криминалистике и раскрытию электронной информации из юридических, аудиторских и консалтинговых компаний; а также экспертам-криминалистам из правоохранительных органов.

К концу книги вы должны получить всестороннее и целостное представление о наборе доступных инструментов командной строки для криминалистического получения данных с носителей и управления криминалистическими образами.

Необходимые знания

Предполагается, что вы обладаете практическими знаниями об операционных системах, прежде всего о среде оболочки Unix и Linux. В примерах книги широко применяется оболочка Bash. Вам также необходимо понимать, как запускать программы командной строки и выполнять базовые операции объединения программ в конвейер и перенаправления данных между ними.

Кроме того, вам понадобятся базовые знания принципов цифровой криминалистики, включая технологию блокировки записи, посекторное получение данных и сохранение целостности доказательств с помощью криптографического хеширования. При выполнении представленных примеров наличие этих фундаментальных знаний подразумевается.

Предварительно установленная платформа и программы

У вас должна быть рабочая платформа Linux с уже установленными необходимыми инструментами. В книге не объясняется, как искать, загружать, компилировать или устанавливать различные средства. Если у вас сравнительно новый компьютер - выпущенный не более чем за год до даты публикации книги - и современный дистрибутив Linux, примеры должны работать без затруднений. Некоторые инструменты не входят в стандартные дистрибутивы Linux, однако их легко найти на GitHub или с помощью поисковой системы.

Структура книги

Эта книга задумана не как хронологическая последовательность действий, а скорее как сборник рецептов для отдельных задач. Тем не менее изложение подчиняется логическому порядку: от настройки платформы, планирования и подготовки до получения данных и последующих операций.

В целом книга построена как справочник, поэтому читать её от начала до конца необязательно. Некоторые разделы предполагают знание и понимание предшествующего материала; в таких случаях приведены соответствующие перекрёстные ссылки.

• **Глава 0** представляет собой общее введение в цифровую криминалистику. Я также рассматриваю историю и развитие этой области и упоминаю важные события, определившие направление её эволюции. Особое внимание уделяется значению стандартов, необходимых для получения цифровых доказательств, которые можно использовать в суде. Книга в целом стремится сохранять международный характер и не зависеть от региональных правовых юрисдикций. Сегодня это важно, поскольку всё больше уголовных расследований пересекают государственные границы и затрагивают несколько юрисдикций. Кроме того, на фоне роста криминалистических возможностей частного сектора книга будет полезна частным криминалистическим лабораториям, особенно в международных компаниях.

• **Глава 1** содержит технический обзор устройств массового хранения, разъёмов и интерфейсов, а также команд и протоколов доступа к носителям. В ней рассматриваются технологии, с которыми типичный эксперт-криминалист сталкивается в профессиональной лаборатории. Я постарался помочь вам ясно понять различия между интерфейсами носителей, туннелированием и преобразованием протоколов, а также способы подключения и взаимодействия носителей с основной системой.
• **Глава 2** содержит обзор Linux как платформы криминалистического получения данных. В ней кратко обсуждаются преимущества и недостатки применения Linux и программ с открытым исходным кодом. Объясняется, как ядро Linux распознаёт и обрабатывает вновь подключаемые к системе устройства и каким образом можно получить доступ к этим устройствам. Глава знакомит читателя с дистрибутивами Linux и выполнением команд в оболочке. В ней также объясняется применение конвейеров и перенаправления - важного понятия, используемого на протяжении всей книги.
• **Глава 3** посвящена различным необработанным и криминалистическим форматам, широко применяемым на практике. Эти форматы служат цифровыми «пакетами для вещественных доказательств», в которых хранятся данные, полученные с носителей. В главе объясняется устройство необработанных образов, описываются коммерческие криминалистические форматы, такие как EnCase и FTK, а также форматы исследовательского сообщества, например AFF. Кроме того, представлен простой контейнер криминалистических доказательств на основе SquashFS и инструмент для управления им.
• **Глава 4** служит переходной точкой: здесь книга оставляет теорию и вступает в более практическую и процедурную область. Глава начинается с примеров ведения журналов и аудиторских следов, а также сохранения данных команд для использования в официальных криминалистических отчётах. Затем рассматриваются различные вопросы планирования и материально-технического обеспечения, с которыми часто сталкиваются эксперты. В заключительном разделе создаётся криминалистически корректная рабочая среда с заблокированной записью, подготавливающая читателя к непосредственному процессу получения данных.
• **Глава 5** продолжает процесс: исследуемый диск подключается к основной системе, выполняющей получение данных, после чего с него собирается информация ATA, SMART и другие сведения. На этом этапе снимаются ограничения доступа к носителю, такие как HPA и DCO, а заблокированные и самошифрующиеся диски становятся доступными. В главе также рассматривается несколько специальных тем, в частности режим внешнего диска Apple. К этому моменту диск подготовлен, и можно выполнять команды получения данных.
• **Глава 6** посвящена непосредственному получению данных и демонстрирует несколько видов криминалистического получения с помощью как открытых, так и проприетарных инструментов. Особое внимание уделяется сохранению доказательств в процессе получения посредством хешей, подписей и служб присвоения временных меток. В главе также разбираются различные сценарии работы с повреждёнными блоками и ошибками, а также удалённое получение данных по сети. К числу специальных тем относятся получение данных с магнитных лент и из систем RAID.

• **Глава 7** сосредоточена на управлении полученными дисковыми образами. Предполагается, что криминалистический образ уже успешно создан, после чего описываются типичные задачи, выполняемые по завершении получения данных. К ним относятся сжатие, разделение и шифрование образов; преобразование между криминалистическими форматами; клонирование или дублирование образов; передача образов другим сторонам и подготовка к долгосрочному хранению. Завершается глава разделом о безопасном уничтожении данных.
• **Глава 8** рассматривает ряд специальных задач, которые можно выполнить после получения данных при подготовке к исследованию. В их число входят доступ к образам через loop-устройства, работа с образами виртуальных машин и доступ к образам, зашифрованным средствами операционной системы, включая BitLocker, FileVault, TrueCrypt/VeraCrypt и другие. В главе также рассматривается доступ к другим контейнерам виртуальных дисков. Эти методы позволяют проводить криминалистический анализ образов и безопасно просматривать файловую систему с помощью обычных файловых диспетчеров и других программ.
• **Глава 9** частично переходит в область криминалистического анализа и демонстрирует извлечение подмножеств данных из образов. Сюда входят выявление и извлечение разделов, в том числе удалённых; извлечение промежутков между разделами; извлечение резервного пространства файловой системы; а также извлечение ранее скрытых областей диска DCO и HPA. В главе приведено несколько примеров фрагментарного извлечения данных, включая извлечение отдельных секторов и блоков.

В каждой главе может быть описано несколько разных инструментов для выполнения одной и той же задачи. Нередко одну задачу можно решить несколькими доступными средствами, причём в зависимости от обстоятельств один инструмент может оказаться полезнее другого. В таких случаях я обсуждаю преимущества и недостатки каждого из них.

Каждый раздел главы имеет приблизительно одинаковую структуру. Заголовок даёт общее описание темы. Во вводном абзаце раскрывается мотивация раздела и объясняется, почему конкретная задача полезна для расследований, цифровой криминалистики или реагирования на инциденты. Во многих случаях эта мотивация обусловлена юридическими или признанными в отрасли стандартами. Важно знать и понимать эти стандарты, поскольку они подтверждают криминалистическую корректность выполняемой работы. При необходимости я привожу ссылки на исходный код инструментов, дополнительные сведения или другие представляющие интерес статьи.

Перед знакомством с новым инструментом или демонстрацией его работы я привожу абзац, описывающий назначение или функции средства и его значение для цифровой криминалистики. Иногда читателю может быть интересна и история инструмента, поэтому в таких случаях я рассказываю также о ней.

После описания задачи и инструмента или инструментов следуют один или несколько примеров командной строки, а также вывод команд, показанный моноширинным шрифтом с фиксированной шириной символов. Команда может повторяться для демонстрации разных вариантов или расширенных способов применения. За каждым примером команды следует абзац с описанием выполняемой команды и объяснением полученного результата.

Заключительный абзац может содержать сведения о возможных неожиданных сложностях, оговорках, рисках, типичных проблемах и ошибках, с которыми вы способны столкнуться и которые имеют значение для цифровых криминалистических расследований.

Область охвата книги

Книга сосредоточена на криминалистическом получении данных с распространённых носителей и действиях, необходимых для сохранения доказательств. Хотя в ней показаны некоторые операции сортировки и анализа, криминалистический анализ данных приложений и операционных систем в целом находится за рамками этой книги.

За её рамками остаётся и ряд других областей, включая получение данных не с традиционных носителей: например, криминалистический сбор сетевого трафика, получение содержимого памяти работающих систем, получение данных из облачных сред и тому подобное.

В разных местах я упоминаю носители корпоративного класса и устаревшие носители, однако не привожу практических примеров. В криминалистических лабораториях они встречаются реже. Тем не менее многие представленные методы в целом применимы и к корпоративному либо устаревшему оборудованию хранения данных.

Получение данных с проприетарных устройств также выходит за рамки книги. С помощью показанных инструментов и методов может оказаться возможным получить данные с мобильных телефонов, планшетов или устройств Интернета вещей новейшего поколения, если ядро Linux видит их как блочные устройства, однако такие устройства я специально не рассматриваю.

Условные обозначения и формат

Примеры кода, команд и их вывода набраны моноширинным шрифтом с фиксированной шириной символов - примерно так, как они выглядят на экране компьютерного терминала. В некоторых местах несущественные части вывода команд удалены или сокращены и заменены многоточием (`...`), а строки, не помещающиеся в поля книги, перенесены и снабжены отступом.

Команды, которые можно выполнять без привилегий `root`, показаны с приглашением `$`. Перед привилегированными командами, которые обычно требуется запускать от имени `root`, стоит приглашение #. Ради краткости применение `sudo` и других средств повышения привилегий показано не всегда. В некоторых разделах приведены дополнительные сведения о выполнении командных процедур пользователем без привилегий `root`.

В издательстве компьютерной литературы принято переносить временные метки в блоках кода и вывода команд на будущую дату после выхода книги, чтобы содержание выглядело новее. Мне показалось неуместным писать книгу о сохранении целостности доказательств и одновременно манипулировать доказательствами, представленными в самой книге, искусственно сдвигая временные метки вперёд. Весь вывод команд, который вы видите в этой книге, отражает фактические результаты тестирования и исследований, включая исходные даты и временные метки. За исключением удаления менее существенных фрагментов с помощью `...` и удаления пустых строк в конце, я оставил вывод команд неизменным.

В конце книги нет библиографии. Все ссылки оформлены как сноски в нижней части той страницы, на которой упоминается источник.

Рабочая станция следователя или эксперта называется основной системой получения данных или основной системой исследования. Диск и образ, с которых получают данные, называются исследуемым диском, подозреваемым диском или диском с доказательствами.

На протяжении книги несколько терминов употребляются как взаимозаменяемые. В обобщённом смысле слова «диск», «накопитель», «носитель» и «хранилище» часто заменяют друг друга. Слова «эксперт-криминалист», «эксперт» и «аналитик» обозначают человека - то есть вас, - который

использует основную систему исследования для различных криминалистических задач. Термины «создание образа», «получение данных» и «получение» также используются как взаимозаменяемые, однако слово «копирование» намеренно исключено, чтобы избежать смешения с обычным копированием вне криминалистического контекста.

Глава 0. Обзор цифровой криминалистики

Небольшой исторический экскурс в область цифровой криминалистики вплоть до наших дней помогает понять, как развивалась эта дисциплина, и даёт дополнительный контекст для некоторых проблем и трудностей, с которыми сталкиваются специалисты криминалистической отрасли.

История цифровой криминалистики

Здесь я расскажу о становлении современной цифровой криминалистики как научной дисциплины.

До 2000 года

По сравнению с другими научными дисциплинами история цифровой криминалистики коротка. Первые криминалистические исследования, связанные с компьютерами, начались в 1980-е годы, когда практикующие специалисты почти исключительно представляли правоохранительные или военные организации. Распространение домашних компьютеров и коммутируемых служб BBS в 1980-е годы пробудило первый интерес к компьютерной криминалистике в правоохранительном сообществе. В 1984 году ФБР разработало новаторскую программу анализа компьютерных доказательств. Кроме того, рост числа злоупотреблений и атак через Интернет привёл в 1988 году к созданию Группы реагирования на компьютерные чрезвычайные ситуации (Computer Emergency Response Team, CERT). CERT была учреждена Управлением перспективных исследовательских проектов Министерства обороны США (Defense Advanced Research Projects Agency, DARPA) и размещается в Университете Карнеги - Меллона в Питтсбурге.

В 1990-е годы доступ к Интернету переживал бурный рост, а персональные компьютеры в домах стали обычным явлением. В то время компьютерная криминалистика превратилась в одну из главных тем для правоохранительных органов. В 1993 году ФБР провело первую из нескольких международных конференций по компьютерным доказательствам для правоохранительных органов, а в 1995 году была создана Международная организация по компьютерным доказательствам (International Organization of Computer Evidence, IOCE), которая начала разрабатывать рекомендации по стандартам. «Компьютерная преступность» стала реальностью не только в Соединённых Штатах, но и во всём мире. В 1999 году Ассоциация руководителей полиции (Association of Chief Police Officers, ACPO) подготовила руководство по надлежащей практике для сотрудников правоохранительных органов Великобритании, работавших с компьютерными доказательствами. Кроме того, в конце 1990-х годов Дэн Фармер и Витсе Венема создали первое криминалистическое программное обеспечение с открытым исходным кодом - The Coroner's Toolkit.

2000-2010 годы

После наступления нового тысячелетия целый ряд факторов повысил спрос на цифровую криминалистику. Трагедия 11 сентября 2001 года оказала огромное влияние на мировое восприятие безопасности и реагирования на инциденты. Бухгалтерские скандалы вокруг Enron и Anderson привели к принятию в Соединённых Штатах закона Сарбейнса - Оксли, призванного защищать инвесторов посредством повышения точности и достоверности раскрываемых компаниями сведений. Закон потребовал от организаций формализовать процессы реагирования на инциденты и расследования, как правило включающие те или иные возможности цифровой криминалистики или сбора доказательств. Растущее внимание к интеллектуальной собственности также повлияло на гражданские организации. Мошенничество в Интернете, фишинг и другие инциденты, связанные с интеллектуальной собственностью и брендами, создали дополнительный спрос на расследования и сбор доказательств. Одноранговый обмен файлами, начавшийся с Napster, и появление законодательства об авторских правах в цифровой среде в виде Закона об авторском праве в цифровую эпоху (Digital Millennium Copyright Act, DMCA) повысили спрос на расследование нарушений цифровых авторских прав.

С 2000 года сообщество цифровой криминалистики добилось значительных успехов в превращении своей области в научную дисциплину. Конференция DFRWS 2001 года сформулировала важные определения и задачи криминалистического сообщества и дала цифровой криминалистике следующее определение:

> Применение научно полученных и проверенных методов для сохранения, сбора, проверки, идентификации, анализа, интерпретации, документирования и представления цифровых доказательств из цифровых источников с целью содействовать восстановлению картины событий, признанных преступными, развивать такое восстановление либо помогать предвидеть несанкционированные действия, способные нарушить запланированную работу.[1]

Пока криминалистическое сообщество определяло границы своей области и стремилось стать признанным направлением научных исследований, одновременно формализовывались стандарты, руководства и процедуры передовой практики для специалистов. Научная рабочая группа по цифровым доказательствам (Scientific Working Group on Digital Evidence, SWGDE) сформулировала определения и стандарты, включая требование применять стандартные операционные процедуры (Standard Operating Procedures, SOP) в правоохранительных органах. Конференция IOCE 2000 года во Франции занималась формализацией процедур для сотрудников правоохранительных органов посредством руководств и контрольных перечней. На 13-м симпозиуме INTERPOL по криминалистике, также проходившем во Франции, были изложены требования к группам, работающим в сфере цифровой криминалистики, и сформулирован всеобъемлющий набор стандартов и принципов для государственных учреждений и правоохранительных органов. Министерство юстиции США опубликовало подробное руководство для сотрудников правоохранительных органов, первыми прибывающих на место происшествия, - *US DOJ Electronic Crime Scene Investigation: A Guide for First Responders*, - а проект NIST по тестированию средств компьютерной криминалистики (Computer Forensics Tool Testing, CFTT) подготовил первую спецификацию средств создания дисковых образов (*Disk Imaging Tool Specification*).

В течение этого десятилетия для публикации растущего массива знаний появилось несколько рецензируемых академических журналов. В 2002 году был основан *The International Journal of Digital Evidence* (IJDE), прекративший существование в 2007 году, а в 2004 году - *Digital Investigation: The International Journal of Digital Forensics & Incident Response*.

С 2010 года по настоящее время

После 2010 года целый ряд событий сместил внимание к расследованию кибератак и утечек данных и к сбору связанных с ними доказательств. WikiLeaks (http://www.wikileaks.org/) начал публиковать утекшие материалы вооружённых сил США, включая видеозаписи и дипломатические телеграммы. Группа Anonymous приобрела известность благодаря распределённым атакам типа «отказ в обслуживании» (Distributed Denial of Service, DDoS) и другой хактивистской деятельности. LulzSec взломала HBGary Federal и другие компании и опубликовала похищенные у них данные.

Исследование вредоносных программ класса Advanced Persistent Threat (APT) стало одной из главных тем отрасли. Общественности стал известен масштаб государственного шпионажа, который одни правительства вели с помощью вредоносных программ против других государств и частных компаний. Был обнаружен червь Stuxnet, нацеленный на системы диспетчерского управления и сбора данных SCADA, в частности на системы управления иранской ядерной программой. Компания Mandiant опубликовала результаты своего расследования деятельности APT1 - подразделения кибервойны китайской армии. Эдвард Сноуден передал огромный архив документов, раскрывших масштабы взломов, проводившихся АНБ. Публикация данных итальянской компании HackingTeam показала существование профессионального рынка эксплойтов, продаваемых правительствам, правоохранительным органам и компаниям частного сектора.

Крупные утечки стали серьёзной проблемой для частных компаний после кражи сведений о кредитных картах и других данных у Sony, Target, JPMorgan Chase, Anthem и иных организаций.

Мировая банковская отрасль столкнулась с резким ростом числа банковских вредоносных программ - Zeus, Sinowal/Torpig, SpyEye, Gozi, Dyre, Dridex и других, - которые успешно атаковали клиентов банков ради финансового мошенничества. Позднее получили распространение атаки с требованием выкупа: программы-вымогатели, DDoS за биткойны и тому подобное.

Такое разнообразие взломов, атак и злоупотреблений расширило область интересов цифровой криминалистики: в неё вошли захват и анализ сетевого трафика, а также получение содержимого оперативной памяти заражённых работающих систем.

Тенденции и сложности криминалистического получения данных

Область цифровой криминалистики постоянно преобразуется вслед за изменениями и достижениями в технологиях и преступной деятельности. В этом разделе я рассматриваю современные проблемы, тенденции и изменения, влияющие на традиционное криминалистическое получение данных с носителей.

Изменение объёма, местонахождения и сложности доказательств

Самое очевидное изменение, влияющее на получение криминалистических образов, связано с ёмкостью дисков. На момент написания книги потребительские жёсткие диски способны хранить 10 Тбайт данных. Появление простых в использовании устройств RAID увеличило логическую ёмкость дисков ещё сильнее. Столь большие объёмы создают трудности для традиционных процессов получения данных в криминалистических лабораториях.

Другая проблема - множество устройств хранения, обнаруживаемых на месте преступления или связанных с инцидентом. Единственный домашний компьютер сменился пёстрым набором компьютеров, ноутбуков, планшетов, мобильных телефонов, внешних дисков, USB-накопителей, карт памяти, компакт-дисков, DVD и других устройств, хранящих значительные объёмы данных. Задача состоит в том, чтобы найти и изъять все относящиеся к делу носители, а затем получить их образы таким образом, чтобы всё содержимое одновременно оставалось доступным инструментам криминалистического анализа.

Перемещение доказательств в облако также порождает множество трудностей. Иногда на устройствах конечных пользователей остаются лишь кешированные копии, а основная часть данных хранится у поставщиков облачных услуг. Правоохранительным органам бывает сложно собрать эти данные, если они находятся за пределами соответствующей правовой юрисдикции; частным организациям трудно сделать это, когда договор с внешним облачным поставщиком не предусматривает криминалистической поддержки.

Быстрое распространение Интернета вещей также вскоре создаст трудности для криминалистического сообщества. Множество небольших электронных устройств, подключённых к Интернету, - медицинские мониторы, часы, дисплеи параметров окружающей среды, камеры наблюдения и прочие приборы - обычно не обладают большим объёмом памяти. Однако в них могут храниться полезные телеметрические данные: временные метки, сведения о местонахождении и перемещениях, параметры окружающей среды и тому подобное. Со временем выявление этих данных и получение доступа к ним станет стандартной частью сбора криминалистических доказательств.

Вероятно, самая трудная проблема, с которой сегодня сталкиваются эксперты-криминалисты, - тенденция к проприетарным, полностью закрытым устройствам. Архитектуры персональных компьютеров и дисковых устройств исторически были открыты и хорошо документированы, что

позволяло создавать стандартные криминалистические инструменты доступа к данным. Однако всё более широкое применение проприетарных программ и оборудования затрудняет такие разработки. Особенно остро эта проблема проявляется в сфере мобильных устройств, где для получения низкоуровневого доступа к блокам файловой системы может потребоваться сначала выполнить джейлбрейк устройства, то есть фактически взломать его.

Аспекты нескольких юрисдикций

Международный характер преступности в Интернете создаёт ещё одну проблему для экспертов-криминалистов. Представьте компанию в стране A, атакованную злоумышленником из страны B, который использует промежуточные прокси-серверы в стране C, чтобы через внешнего подрядчика в стране D взломать инфраструктуру и вывести похищенные данные в зону сбора в стране E. В таком сценарии участвуют пять разных государств, а значит, потенциально требуется координация пяти правоохранительных органов, взаимодействующих как минимум с пятью компаниями в пяти правовых юрисдикциях. Сегодня подобный сценарий с участием нескольких стран не является необычным - напротив, он встречается довольно часто.

Сотрудничество отрасли, научного сообщества и правоохранительных органов

Всё более сложный и технически развитый характер преступной деятельности в Интернете способствует углублению сотрудничества и совместной работы по сбору разведывательных сведений и доказательств, а также по координации расследований.

Такое взаимодействие между конкурирующими компаниями одной отрасли можно рассматривать как борьбу с общим врагом: банковской отрасли - с банковскими вредоносными программами, поставщиков интернет-услуг - с DDoS и спамом и так далее. Сотрудничество пересекло и границу между частным и государственным секторами: правоохранительные органы вместе с отраслевыми партнёрами противодействуют преступности в рамках государственно-частных партнёрств (public-private partnerships, PPP). Это многостороннее взаимодействие создаёт возможности выявлять, собирать и передавать цифровые доказательства. Трудность состоит в том, чтобы частные партнёры понимали природу цифровых доказательств и могли соблюдать стандарты, выполнения которых ожидают от правоохранительных органов государственного сектора. Это повысит вероятность успешного уголовного преследования на основании доказательств, собранных частным сектором.

Третья группа, сотрудничающая с промышленностью и правоохранительными органами, - академическое исследовательское сообщество. Обычно в него входят университетские криминалистические лаборатории и исследовательские подразделения в области безопасности, которые углублённо изучают теоретические и высокотехнические аспекты компьютерной преступности и криминалистики. Эти исследователи могут посвятить время анализу проблем и получить представление о новых преступных методах и криминалистических приёмах. Иногда они способны оказать правоохранительным органам помощь там, где стандартные криминалистические инструменты не позволяют извлечь или проанализировать необходимые доказательства. Академические группы также должны понимать требования и ожидания, связанные с управлением цифровыми доказательствами и их сохранением.

Принципы посмертной компьютерной криминалистики

На принципы цифровой криминалистики как научной дисциплины влияет ряд факторов, включая официально определённые стандарты, рецензируемые исследования, отраслевое регулирование и передовую практику.

Стандарты цифровой криминалистики

Стандарты сбора и сохранения традиционных вещественных доказательств в значительной степени зависели от местной правовой юрисдикции. В отличие от них, сбор цифровых доказательств сформировался в международной взаимосвязанной среде, где в исследованиях и разработке стандартов участвует множество юрисдикций. Оборудование, программы, форматы файлов, сетевые протоколы и другие технологии, как правило, одинаковы во всём мире. Поэтому стандарты и процедуры сбора цифровых доказательств в разных юрисдикциях согласованы лучше. Хороший пример - использование блокираторов записи при подключении дисков к системам создания образов: эта практика признана почти повсеместно.

Существует несколько официальных органов стандартизации, которые определяют стандарты криминалистического получения данных. Национальный институт стандартов и технологий США (National Institute of Standards and Technology, NIST) ведёт программу тестирования средств компьютерной криминалистики (Computer Forensic Tool Testing, CFTT). Её цель сформулирована следующим образом:

> Цель проекта Computer Forensic Tool Testing (CFTT) Национального института стандартов и технологий (NIST) состоит в разработке методологии тестирования программных средств компьютерной криминалистики посредством создания общих спецификаций инструментов, процедур и критериев тестирования, тестовых наборов и испытательного оборудования.

Хотя NIST ориентирован на Соединённые Штаты, многие его стандарты принимаются на международном уровне или как минимум влияют на органы стандартизации других стран.

Международная организация по стандартизации (International Organization for Standardization, ISO) также предлагает ряд стандартов, относящихся к цифровым доказательствам. Для криминалистического получения данных имеют значение руководящие указания ISO по идентификации, сбору, получению и сохранению цифровых доказательств:

> ISO/IEC 27037:2012 содержит руководящие указания по конкретным действиям при обращении с цифровыми доказательствами: идентификации, сбору, получению и сохранению потенциальных цифровых доказательств, которые могут обладать доказательной ценностью.
>
> Стандарт содержит рекомендации для отдельных лиц применительно к распространённым ситуациям, возникающим на протяжении всего процесса обращения с цифровыми доказательствами, помогает организациям в проведении внутренних разбирательств и облегчает обмен потенциальными цифровыми доказательствами между юрисдикциями.

Отдельные полицейские ведомства могут иметь собственные стандарты, описывающие процесс сбора доказательств. Например, в Великобритании Ассоциация руководителей полиции (ACPO) выпускает *Руководство ACPO по надлежащей практике работы с цифровыми доказательствами* (*ACPO Good Practice Guide for Digital Evidence*). В руководстве сказано:

> Настоящее руководство по передовой практике подготовлено направлением ACPO по борьбе с преступностью и первоначально утверждено кабинетом ACPO в декабре 2007 года. Цель документа - предоставить рекомендации не только правоохранительным органам, но и всем, кто содействует расследованию происшествий и преступлений в сфере кибербезопасности. По мере изменения законодательства и политики документ будет обновляться и при необходимости переиздаваться.

В этом документе упоминается ряд других стандартов и документов, выпущенных ACPO и иными организациями.

Министерство юстиции США поддерживает руководство *Расследование электронных преступлений на месте происшествия: руководство для сотрудников, прибывающих первыми* (*Electronic Crime Scene Investigation: A Guide for First Responders*). Во введении к нему сказано:

> Настоящее руководство призвано помочь правоохранительным органам штатов и местным правоохранительным органам, а также другим сотрудникам, прибывающим первыми, которые могут отвечать за сохранение места совершения электронного преступления и за распознавание, сбор и защиту цифровых доказательств.

Многие другие международные организации содействуют разработке стандартов, создавая криминалистические рабочие группы, комитеты и сообщества.

Рецензируемые исследования

Ещё одним источником стандартов и методов цифровой криминалистики служат рецензируемые исследования и научные конференции. На них представляют новейшие достижения и методы исследовательского сообщества цифровой криминалистики. Основывать криминалистическую работу на рецензируемых научных исследованиях особенно важно при использовании новых методов и технологий, поскольку они могли ещё не пройти проверку в судах.

Существует несколько международных академических исследовательских сообществ, которые пополняют совокупность знаний. Самый известный научный журнал в области криминалистики - *Digital Investigation: The International Journal of Digital Forensics & Incident Response*, более десяти лет публикующий академические исследования в этой сфере. Заявленные цели и область журнала описаны следующим образом:

> Журнал *Digital Investigation* освещает передовые разработки в области цифровой криминалистики и реагирования на инциденты со всего мира. Это широко цитируемое издание помогает специалистам по цифровым расследованиям следить за новыми технологиями, полезными инструментами, актуальными исследованиями, методами расследования и способами реагирования на нарушения безопасности. Практикующие специалисты из корпоративной, уголовной и военной сфер делятся на страницах журнала знаниями и опытом, в том числе современными проблемами и усвоенными уроками в следующих областях:
>
> **Рецензируемые исследования.** Новые подходы к решению проблем цифровых расследований, включая прикладные исследования по анализу конкретных технологий и применение информатики для решения задач, возникающих в цифровой криминалистике и при реагировании на инциденты.
>
> **Отчёты практикующих специалистов.** Разборы расследований и отчёты, описывающие, как практикующие специалисты справляются с новыми проблемами в этой области, включая усовершенствованные методы проведения эффективных цифровых расследований. . . .

Ведущей научной конференцией по исследованиям в области цифровой криминалистики является Digital Forensics Research WorkShop (DFRWS). Она проводится с 2001 года и всё это время проходила в США, однако в 2014 году появилось отдельное европейское мероприятие. Заявленные цели DFRWS таковы:[2]

• привлекать новые точки зрения и содействовать обмену идеями ради развития цифровой криминалистики;
• способствовать научному обсуждению исследований в области цифровой криминалистики и их применения;
• привлекать опытных аналитиков и экспертов из правоохранительного, военного и гражданского секторов, чтобы сосредоточить исследования на потребностях практикующих специалистов, различных условиях расследований и практической применимости;
• определять основные технологии, на которых следует сосредоточить полезные исследования и разработки;
• содействовать обнаружению, объяснению и представлению убедительных, неоспоримых доказательств, способных выдержать повышенное внимание судов и других лиц, принимающих решения в гражданской и военной среде;
• создавать и расширять общий словарь, чтобы сообщество говорило на одном языке;
• регулярно участвовать в обсуждениях и совместной деятельности, обеспечивая чёткую направленность, высокий интерес и результативность;
• поддерживать динамичное сообщество экспертов из академической и практической среды;
• повышать научную строгость цифровой криминалистики;
• вдохновлять следующее поколение на создание новых решений.

Для полной прозрачности сообщу: я являюсь редактором журнала *Digital Investigation* и участвую в организационном комитете DFRWS Europe.

Отраслевые нормы и передовая практика

Отраслевые нормативные требования могут накладывать дополнительные условия или ограничения на сбор цифровых доказательств.

В частном секторе отраслевые стандарты и передовую практику разрабатывают различные организации и отраслевые группы. Например, Консультативный совет по обеспечению информации (Information Assurance Advisory Council, IAAC) выпускает *Руководство по цифровым расследованиям и доказательствам для директоров и корпоративных консультантов* (*Directors and Corporate Advisors' Guide to Digital Investigations and Evidence*).

Другим источником служат стандарты и процессы, предписанные законодательными и регулирующими органами, например требования американского закона Сарбейнса - Оксли к возможностям сбора доказательств.

Некоторые требования к цифровым доказательствам могут зависеть от отрасли. Например, региональные нормы здравоохранения могут устанавливать требования к защите данных и предусматривать различные процедуры криминалистического реагирования и сбора доказательств при нарушении безопасности. Для операторов связи могут действовать нормы хранения журналов и предоставления правоохранительным органам доступа к передаваемым через инфраструктуру сообщениям.

Банковские регуляторы также устанавливают требования и стандарты для цифровых доказательств. Хороший пример - Денежно-кредитное управление Сингапура (Monetary Authority of Singapore, MAS), которое предоставляет банковскому сообществу подробные стандарты в таких областях, как безопасность и реагирование на инциденты (http://www.mas.gov.sg/regulations-and-financial-stability/regulatory-and-supervisory-framework/risk-management/technology-risk.aspx).

В связи с недавним ростом числа кибератак на разные отрасли - финансовую, медицинскую и другие - регулирующие органы в будущем могут играть более значительную роль во влиянии на стандарты сбора доказательств и в их определении.

Принципы, применяемые в этой книге

Книга посвящена криминалистическим задачам, общим для частного и государственного секторов. Примеры начинаются с упрощённого криминалистического получения данных, после чего демонстрируются дополнительные свойства и возможности этого процесса. В их число входят сохранение доказательств посредством криптографического хеширования и подписания, ведение журналов, производительность, обработка ошибок и защита полученного образа. Я также объясняю несколько методов создания образов по сети и рассматриваю специальные темы, такие как магнитные ленты и системы RAID.

Для криминалистического получения данных необходимо выполнить несколько предварительных условий:

• исследуемый накопитель подключён и распознан ядром Linux;
• установлена блокировка записи;
• исследуемый накопитель однозначно идентифицирован и документирован;
• возможен полный доступ к устройству, то есть HPA, DCO и защита ATA отключены;
• для получения данных имеется достаточно времени и ёмкости хранилища.

Процесс криминалистического получения данных и тестирование инструментов хорошо документированы в сообществе цифровой криминалистики, и к ним предъявляются определённые требования. Полезным источником служит учреждённая NIST программа CFTT.

К требованиям NIST верхнего уровня для создания криминалистических образов относятся следующие:

• инструмент должен создать потоковую побитовую копию либо образ исходного диска или раздела;
• инструмент не должен изменять исходный диск;
• инструмент должен регистрировать ошибки ввода-вывода;
• документация инструмента должна быть правильной.

Эти принципы, изложенные в опубликованной NIST статье,[3] создают основу для остальной части книги. Они существуют для того, чтобы сохранялась целостность доказательств, а вмешательство либо предотвращалось, либо обнаруживалось.

Некоторые исследования оспаривают представление о том, что полное получение данных возможно с учётом ограничений интерфейса ATA, применяемого для доступа к физическому диску.[4] Теоретически полное получение охватывает все секторы магнитных дисков и память под уровнем трансляции флеш-накопителей в SSD и флеш-устройствах, а теперь распространяется и на закрытые мобильные устройства, образы которых невозможно создать традиционными методами работы с блочными устройствами. Добиться «полного» получения всего физического хранилища устройства становится всё труднее. В отношении мобильных устройств криминалистическое сообщество уже разграничило физическое и логическое получение данных: последнее означает копирование файлов и данных, а не создание образа секторов накопителя.

В примерах этой книги криминалистически полным считается получение данных из областей диска, к которым можно надёжно и воспроизводимо обращаться с помощью общедоступных программных инструментов, использующих опубликованные спецификации интерфейсов. Области диска, доступные только посредством закрытых проприетарных инструментов изготовителя - внутренних средств диагностики, инструментов разработки и тому подобных - либо посредством разборки оборудования - выпаивания микросхем, замены блока головок, извлечения пластин диска и так далее, - находятся за рамками книги.

Это было краткое введение в область цифровой криминалистики. Глава 1 продолжит знакомство с технологиями носителей данных и интерфейсами, применяемыми для их подключения к основной системе получения данных.

Сноски

[1] [назад] Gary Palmer, “A Roadmap for Digital Forensic Research.” Digital Forensics Research Workshop (DFRWS), 2001. Technical report DTR-T0010-01, Utica, New York.
[2] [назад] http://www.dfrws.org/about-us/
[3] [назад] https://utica.edu/academic/institutes/ecii/publications/articles/A04BC142-F4C3-EB2B-462CCC0C887B3CBE.pdf
[4] [назад] “Forensic Imaging of Hard Disk Drives-What We Thought We Knew,” *Forensic Focus*, January 27, 2012, http://articles.forensicfocus.com/2012/01/27/forensic-imaging-of-hard-disk-drives-what-we-thought-we-knew-2/.

Глава 1. Обзор носителей информации

Эта глава служит обзором шин персональных компьютеров, распространённых носителей массовой памяти, физических разъёмов и интерфейсов, а также низкоуровневых команд протоколов, применяемых для обмена данными с подключёнными устройствами хранения. Кроме того, она даёт основы, необходимые для понимания криминалистического получения данных с носителей, описанного в остальной части книги.

В целом технологии массового хранения делятся на три широкие категории: магнитные носители, энергонезависимую, или флеш-память, и оптические носители. Носитель может быть встроенным в устройство или съёмным. Устройство также содержит электронные компоненты накопителя, необходимые для взаимодействия с носителем. Система обращается к устройствам хранения через внутреннюю либо внешнюю шину или интерфейс.

Глава начинается с обзора этих трёх технологий хранения и затрагивает ключевые вопросы цифровой криминалистики. В двух заключительных разделах описано подключение таких устройств к системе Linux и обмен данными с ней; там же я рассматриваю вопросы, представляющие особый интерес для эксперта-криминалиста.

Основное внимание в этой главе уделено современным архитектурам и компонентам персональных компьютеров. Некогда популярные устаревшие технологии могут упоминаться, но не рассматриваются подробно. Кроме того, я ограничил обзор компьютерным оборудованием, которым пользуются отдельные люди - сотрудники, домашние пользователи и другие - и небольшие серверные среды, не охватывая технологии крупных предприятий. Технологии хранения в больших корпоративных средах не всегда подходят для традиционного создания криминалистических образов дисковых носителей: иногда из-за одного лишь объёма хранилища

традиционное получение данных становится практически невозможным, а критически важные корпоративные системы обычно нельзя отключить, как небольшие системы на основе персональных компьютеров.

Магнитные носители

Магнитные носители - старейшая из трёх основных технологий хранения, если не считать предшествовавшие им перфоленты и перфокарты, и в настоящее время лидируют по ёмкости. Сегодня преимущественно используются два вида магнитных носителей: жёсткие диски и ленты. Оба обеспечивают высокую ёмкость и надёжность как для оперативного хранения, так и для автономного архивирования.

> **Примечание.** Соперничество по ёмкости между магнитными дисками и твердотельными накопителями (SSD) обостряется. Во время работы над этой книгой был анонсирован SSD ёмкостью 16 Тбайт, который после выпуска мог стать крупнейшим диском в мире.

Жёсткие диски

Жёсткие диски неизменно обеспечивали большую ёмкость, чем другие носители, например SSD или оптические диски. На момент написания книги на потребительском рынке доступны жёсткие диски ёмкостью 10 Тбайт, и ожидается появление ещё более ёмких моделей.

Жёсткие диски состоят из вращающихся пластин, покрытых магнитным материалом, как показано на рис. 1-1. Несколько пластин закреплены друг над другом на шпинделе, а головки чтения и записи на подвижном рычаге, то есть приводе, могут считывать закодированные данные с магнитной поверхности и записывать их на неё. Сейчас к распространённым форм-факторам жёстких дисков относятся 3,5, 2,5 и 1,8 дюйма. Поскольку жёсткие диски являются механическими устройствами, они чувствительны к ударам, падениям, пыли, влаге и другим факторам окружающей среды. К типичным неисправностям относятся царапины на поверхности пластин, застрявшие или повреждённые головки, отказ двигателя и электронных схем.

Подлинная физическая геометрия диска - головки, пластины, дорожки и число секторов на дорожке - скрыта от компьютера абстракцией и доступна как последовательность секторов посредством адресации логических блоков (Logical Block Addressing, LBA). Сектор - наименьшая адресуемая единица диска для чтения и записи данных. Исторически стандартный физический сектор жёсткого диска имел размер 512 байт, однако современные диски перешли к секторам по 4 Кбайт. Большинство нынешних накопителей по-прежнему эмулируют 512-байтовые секторы, но на рынке уже представлены диски с собственными секторами по 4 Кбайт, известные как накопители 4Kn. Диски 4Kn обладают преимуществами в производительности и, вероятно, когда-нибудь вытеснят традиционные накопители с эмуляцией 512-байтовых секторов. Более подробно диски 4Kn рассматриваются в разделе «Расширенный формат 4Kn» на странице 41.

Рис. 1-1. Магнитный жёсткий диск

Традиционная компьютерная криминалистика возникла из необходимости анализировать жёсткие диски, которые и сегодня остаются важными источниками доказательств. В частности, когда операционная система «удаляет» файл, она лишь разрывает все ссылки на блоки его данных на диске, в отличие от SSD, где для очистки нераспределённых блоков применяется команда TRIM. Эти блоки не стираются с магнитных пластин и остаются на диске, откуда криминалистические инструменты могут их восстановить, пока они не будут перезаписаны.

Магнитные ленты

На рынке домашних пользователей магнитные ленты почти исчезли, однако малые предприятия и корпоративные среды продолжают применять их для резервного копирования и архивирования. Ленты, показанные на рис. 1-2, относятся к ранним формам цифрового хранения и считаются зрелой технологией, надёжной для долговременного автономного хранения. В отличие от дисков, SSD, флеш-накопителей и оптических дисков, данные на ленте можно читать и записывать только последовательно. Для произвольного доступа к разным блокам пользователь должен перемотать ленту назад или вперёд до нужного места, прежде чем его можно будет прочитать или записать. Отсутствие произвольного блочного доступа не позволяет использовать ленты как обычные файловые системы. Данные хранятся на лентах в виде последовательности ленточных файлов; каждый файл обычно представляет собой архив, содержащий файловую систему либо группу файлов и каталогов и использующий такой формат, как TAR, DUMP и тому подобное. Ленточными накопителями управляют с помощью ленточных команд SCSI, предназначенных для чтения и записи данных, позиционирования или перемотки ленты и её извлечения.

> **Примечание.** Новые накопители Linear Tape-Open, или LTO, способны имитировать обычную файловую систему с помощью Linear Tape File System (LTFS), однако произвольного доступа это не обеспечивает: файлы всё равно читаются и записываются последовательно.

Рис. 1-2. Магнитные ленты

Ниже приведены примеры накопителя LTO5 с интерфейсом Fibre Channel и накопителя DAT160 с интерфейсом USB; оба подключены к системе Linux. Вывод `dmesg` для этих ленточных накопителей выглядит следующим образом:

```
[ 11.290176] scsi 1:0:0:0: Sequential-Access TANDBERG LTO-5 HH
Y629 PQ: 0 ANSI: 6
[ 11.293554] scsi 1:0:0:0: Attached scsi generic sg5 type 1
[ 11.345030] st: Version 20101219, fixed bufsize 32768, s/g segs 256
[ 11.361189] st 1:0:0:0: Attached scsi tape st0
...
[ 3263.575014] usb 1-8: new high-speed USB device number 14 using xhci_hcd
[ 3263.703245] usb 1-8: New USB device found, idVendor=03f0, idProduct=0225
[ 3263.703250] usb 1-8: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 3263.703253] usb 1-8: Product: DAT160 USB Tape
[ 3263.703255] usb 1-8: Manufacturer: Hewlett Packard
[ 3263.703257] usb 1-8: SerialNumber: 48553101234E4648
[ 3263.704156] usb-storage 1-8:1.0: USB Mass Storage device detected
[ 3263.704295] scsi host12: usb-storage 1-8:1.0
[ 3264.713397] scsi 12:0:0:0: Sequential-Access HP DAT160
WU8A PQ: 0 ANSI: 3
[ 3264.722279] st 12:0:0:0: Attached scsi tape st1
```

После записи файлов ленточного архива на ленту также записывается маркер конца данных (End Of Data, EOD). Он сообщает накопителю, что достигнут конец данных на ленте, и не позволяет читать дальше. Однако с криминалистической точки зрения любые данные за маркером EOD

представляют интерес, поскольку там могут находиться сведения от предыдущих операций записи на ленту. Универсальных команд SCSI для получения данных за маркером EOD не существует. Для этой задачи необходимы специализированные ленточные накопители и оборудование.

Устаревшие магнитные носители

Существует множество устаревших типов магнитных носителей, особенно съёмных. Дискеты прошли через несколько поколений, прежде чем устарели. В 1980-е и 1990-е годы на рынке пользовался популярностью целый ряд проприетарных устройств хранения, таких как Jaz, Zip, Syquest и другие. Больше не применяется и множество разновидностей магнитных лент, например 4-миллиметровые DAT, 8-миллиметровые Exabyte и QIC. Криминалистическое получение данных с таких носителей находится за рамками книги. Однако при наличии исправного оборудования и интерфейсов данные с большинства подобных старых устройств можно получить теми же методами, которые описаны здесь. Если ядро Linux распознаёт секторный носитель и предоставляет его как блочное устройство, данные с него можно получить. Если ядро Linux распознаёт ленточный накопитель как ленточное устройство SCSI, к нему можно обращаться стандартными ленточными командами SCSI. Для проприетарных хранилищ могут существовать драйверы ядра или инструменты пользовательского пространства, обеспечивающие доступ к устаревшим устройствам.

Энергонезависимая память

Энергонезависимая память, обычно основанная на технологии флеш-памяти NAND, становится всё популярнее и начинает заменять магнитные жёсткие диски там, где не требуется очень большая ёмкость. Термин NAND означает транзисторы, работающие как логический элемент И-НЕ. Этот вид памяти ставит перед экспертами-криминалистами новый круг задач, поскольку на низком уровне обладает не теми свойствами, что магнитные диски.[1]

SSD и флеш-носители обычно представляют собой накопители на основе NAND и не имеют движущихся деталей. Данные хранятся в массиве ячеек памяти, а уровень абстракции, называемый уровнем трансляции флеш-памяти (Flash Translation Layer, FTL), заставляет накопитель вести себя подобно жёсткому диску - как линейная последовательность секторов. Поскольку диски с энергонезависимой памятью реализованы электронными схемами, а не механикой, они бесшумны, потребляют меньше энергии и не подвержены такому же риску физических повреждений, как жёсткие диски. Что касается производительности, они быстрее выполняют произвольный доступ и запись, поскольку им не приходится физически перемещать головки в нужные места диска. По этой же причине дефрагментация файловых систем не даёт преимущества в производительности. Память накопителей SSD и флеш-устройств менее долговечна, чем магнитные пластины жёстких дисков. Для продления срока службы SSD применяются такие методы, как выравнивание износа и избыточное резервирование. Выравниванием износа называется механизм распределения операций чтения и записи по накопителю, обеспечивающий равномерное использование блоков в течение срока службы устройства. По мере деградации блоков или утраты возможности записи FTL выводит их из эксплуатации и заменяет блоками из пула зарезервированных, то есть избыточно выделенных, блоков. Данные из таких «выведенных из эксплуатации» блоков можно прочитать, если снять, то есть выпаять, физические микросхемы и считать их память. Некоторые профессиональные криминалистические лаборатории выполняют эту процедуру, которую иногда называют chip-off, для различных накопителей на основе флеш-памяти. При изготовлении под избыточное

резервирование может отводиться от 10 до 25 процентов объёма флеш-диска, недоступных пользователю. В настоящее время существует проект микропрограммы SSD с открытым исходным кодом, полезный для изучения и исследования базовой технологии SSD. Дополнительные сведения о нём доступны по адресу http://www.openssd-project.org/wiki/The_OpenSSD_Project.

Твердотельные накопители

SSD, показанный на рис. 1-3, был разработан как непосредственная замена обычным дискам SATA. SATA, или Serial AT Attachment, - стандартный интерфейс дисковых накопителей. SSD оснащены стандартными интерфейсами SATA, поддерживают технологию самоконтроля, анализа и отчётности Self-Monitoring, Analysis and Reporting Technology (SMART) и используют обычные команды ATA с некоторыми дополнениями. Даже физический форм-фактор распространённых потребительских SSD совпадает с форм-фактором магнитных жёстких дисков. Новые SSD ставят перед экспертами цифровой криминалистики несколько задач: с одной стороны, речь идёт о возможности восстановить данные из нераспределённых секторов накопителя, с другой - о невозможности доступа к областям избыточного резервирования. В устройствах SSD и операционных системах, поддерживающих команду ATA TRIM, нераспределённые блоки диска могут стираться при подготовке к следующему использованию, поскольку перед записью или изменением блок SSD необходимо стереть. Это снижает возможность восстановления данных из нераспределённых блоков, которые обычно служат ценным источником доказательств на магнитных дисках.

Рис. 1-3. SSD

Поддерживаемые SSD функции TRIM можно определить командой `hdparm`. Например:

```
# hdparm -I /dev/sda
...
Commands/features:
Enabled Supported:
...
* Data Set Management TRIM supported (limit 1 block)
* Deterministic read data after TRIM
...
```

Современное поколение SSD, основанное на новых стандартах SATA Express и NVM Express, взаимодействует непосредственно с шиной PCI Express.

Флеш-накопители USB

Небольшие переносные флеш-накопители USB, показанные на рис. 1-4, называют по-разному: флешками, USB-брелоками, флеш-ключами или просто флеш-накопителями USB. Сначала флеш-накопители пришли на смену дискетам, а теперь благодаря низкой цене и высокой ёмкости вытесняют компакт-диски и DVD.

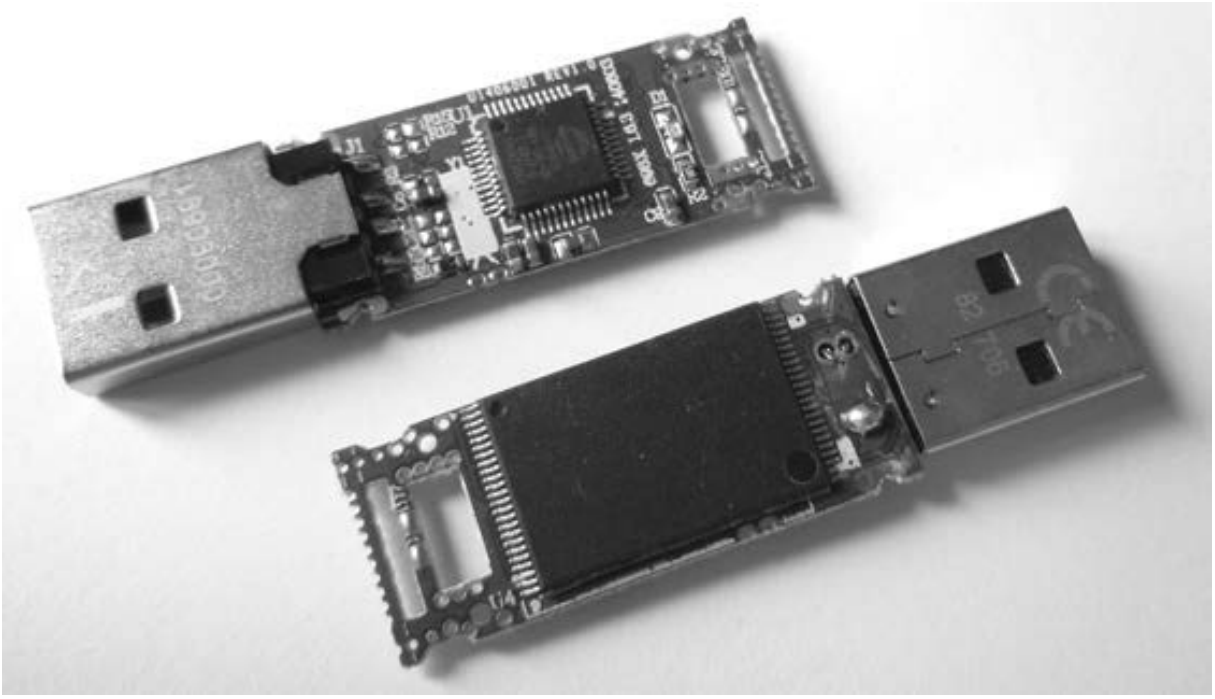

Рис. 1-4. Флеш-накопитель USB

Однако малый размер и большая ёмкость флеш-накопителей USB создают угрозу информационной безопасности. Поэтому большинство производителей предлагают решения с шифрованием. Чаще всего программное шифрование является необязательным. Владелец накопителя должен явно установить предоставленную изготовителем программу шифрования либо применить альтернативное средство, например BitLocker или TrueCrypt. Некоторые USB-накопители оснащены обязательными аппаратными системами шифрования. Однако дополнительные усилия и сложность программного шифрования, а также значительно более высокая стоимость аппаратного шифрования мешают их широкому распространению. Поэтому чаще всего по-прежнему применяются обычные незашифрованные USB-устройства.

Съёмные карты памяти

Популярность переносных устройств - мобильных телефонов, планшетов, камер и других - создала рынок съёмных карт памяти, которые можно заменить, когда они заполнятся или когда потребуется скопировать данные на персональный компьютер либо другое устройство. Различные карты памяти показаны на рис. 1-5. Как правило, они основаны на флеш-памяти и после установки в устройство чтения представляются как линейная последовательность секторов, подобная жёсткому диску.

Рис. 1-5. Карты флеш-памяти

Наиболее распространены карты стандарта Secure Digital (SD), выпускаемые в нескольких форм-факторах и скоростных классах. Карты CompactFlash (CF) популярны в профессиональной фототехнике и по существу представляют собой интерфейс PATA/IDE в меньшем форм-факторе. Обращаться к ним можно через переходник интерфейса PATA/IDE. Устройство чтения карт, подключённое по USB, обеспечивает доступ к картам обоих типов (см. рис. 1-6).

Рис. 1-6. Устройство чтения карт USB

> **Примечание.** Parallel ATA (PATA) и Integrated Drive Electronics (IDE) - устаревшие стандарты, определяющие параллельный интерфейс между накопителем и компьютерной системой.

Устаревшая энергонезависимая память

Многие устаревшие карты памяти исчезли, когда рынок перестала устраивать их предельная ёмкость или проприетарные интерфейсы. В качестве примеров можно назвать Sony Memory Stick, карты Panasonic P2, SmartMedia и другие носители PCMCIA/PC Card. Обычно такие карты памяти подключаются к системе Linux как блочные устройства с линейной последовательностью секторов, и при наличии физического устройства чтения их образы можно создавать теми же методами, что и для других карт памяти.

Оптические носители

К распространённым сегодня оптическим носителям относятся CD-ROM, DVD-ROM и диски Blu-ray. Различные виды оптических носителей отличаются физическими и химическими свойствами. Их видимые различия показаны на рис. 1-7.

Рис. 1-7. Сверху вниз: DVD-ROM, Blu-ray, CD-ROM

Оптические диски обычно бывают предназначенными только для чтения, однократной записи либо многократной записи и чтения. Профессионально изготовленные диски штампуют, а не записывают. Хотя записываемые оптические диски постепенно устаревают, они по-прежнему часто служат источниками цифровых доказательств. Во многих криминалистических лабораториях их всё ещё применяют для передачи и хранения сжатых криминалистических образов.

Оптический диск содержит одну спиральную дорожку с последовательностью углублений и площадок на отражающей поверхности; лазер считывает их и интерпретирует как биты данных. Для записи лазер выжигает точки на поверхности, изменяя её отражательную способность, вследствие чего записанные области интерпретируются как биты данных. Эта последовательность битов разделена на секторы, каждый из которых после кодирования и коррекции ошибок содержит 2048 байт данных, доступных пользователю.

У оптических дисков есть одна общая черта с магнитными лентами: данные записываются как единая линейная строка байтов, а файлы не фрагментируются. В отличие от ленты, на диске легко переходить к произвольным областям, поэтому его можно подключить как файловую систему только для чтения. Тем не менее запись на оптический диск остаётся неудобной и, как на ленте, должна выполняться в последовательном порядке байтов.

Компакт-диски

Старейшие из трёх видов оптических дисков, CD-ROM/CD-R, некогда были самыми популярными оптическими носителями для домашних резервных копий, личных архивов и обмена информацией. Однако с распространением удобных флеш-накопителей USB большой ёмкости компакт-диски стали реже использоваться для хранения.

Различные спецификации компакт-дисков описывает совокупность стандартов под названием Rainbow Books; существует несколько распространённых стандартов CD. Дополнительные сведения см. на странице интеллектуальной собственности Philips: http://www.ip.philips.com/licensing/program/16/.

• Музыкальные компакт-диски, или Compact Disc-Digital Audio (CD-DA), определены Красной книгой и стандартом IEC 60908. Данные в этом формате разделены на несколько звуковых дорожек.
• Компакт-диски с данными, или Compact Disc-Read Only Memory (CD-ROM), описаны Жёлтой книгой и стандартами ISO/IEC 10149 и ECMA 130 (http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-130.pdf).
• Записываемые компакт-диски, или Compact Disc-Recordable/ReWritable (CD-R/CD-RW), относятся к стандарту Оранжевой книги и позволяют записывать данные на компакт-диск (CD-R) либо многократно их перезаписывать (CD-RW).
• К менее распространённым стандартам относятся Photo-CD, Video CD (VCD), а также иные редкие варианты, расширения и усовершенствования общепринятых стандартов.

Каждый компакт-диск содержит линейный поток битов - углублений и площадок, - представленный абстракцией в виде линейной последовательности секторов. На следующем уровне абстракции над секторами находятся сессии, каждая из которых содержит вводную область и таблицу содержания (Table Of Contents, TOC). Компакт-диск может быть многосессионным, и у каждой сессии имеется собственная TOC.

В сессии компакт-диска с данными может находиться файловая система, содержащая файлы и структуру каталогов. Ниже приведены примеры файловых систем CD-ROM:

• **High Sierra** - исходный стандарт для персональных компьютеров: имена в формате 8.3 и верхнем регистре;
• **ISO9660** - обновлённая версия High Sierra для разных платформ;

• **Joliet** - расширения ISO9660 от Microsoft для Windows 95 и более новых систем;
• **HFS** - файловая система Macintosh;
• **Rock Ridge** - расширения ISO9660 для POSIX;
• **El Torito** - стандарт загрузочных дисков.

С криминалистической точки зрения можно прочитать пользовательские данные со всего компакт-диска. На нём нет аналога маркера EOD с магнитных лент или областей DCO/HPA на жёстком диске, недоступных пользователю. Однако в файловой системе могут присутствовать специфические артефакты, для интерпретации которых потребуется особый анализ.

Криминалистические блокираторы записи - оборудование, предназначенное для предотвращения вмешательства в данные накопителя и загрязнения обнаруженных на нём доказательств, - для CD-ROM не нужны, поскольку по умолчанию такие диски предназначены только для чтения. Операционная система не обновит временные метки и не изменит данные на компакт-диске лишь из-за того, что его вставили в накопитель и обратились к нему.

У компакт-дисков есть уникальные идентификаторы, полезные в криминалистическом контексте. Уникальный идентификатор источника (Source Unique Identifier, SID) штампуется на диске и содержит сведения о предприятии, изготовившем конкретный оптический диск. Этот код начинается с IFPI и физически отштампован во внутренней области диска, где его легко прочитать невооружённым глазом. Идентификационный код устройства записи (Recorder Identification Code, RID) записывается в системные секторы диска и связывает записанный компакт-диск с создавшим его накопителем. Получить RID без специализированного оборудования непросто.

Другие физические признаки, например свидетельствующие о пиратском происхождении или подделке, находятся за рамками этой книги, однако руководство Международной федерации производителей фонограмм (International Federation of the Phonographic Industry, IFPI) доступно по адресу http://www.ifpi.org/content/library/manual-of-guidance-chap3-english.pdf.

Цифровые универсальные диски

Физические характеристики DVD отличаются от характеристик компакт-дисков, но логически эти носители похожи. DVD содержит одну спиральную дорожку битов, разделённую на секторы по 2048 байт.

DVD бывают одно- и двусторонними, а также одно- и двухслойными. Удвоение числа сторон или слоёв фактически удваивает ёмкость. Их стандарты похожи на стандарты компакт-дисков, но содержат некоторые дополнения, описанные по адресу http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-382.pdf. Стандарты DVD-Video, DVD-ROM, DVD-R и DVD-RW соответствуют аналогам для компакт-дисков, однако существует и дополнительный стандарт DVD-RAM. Был создан альтернативный набор стандартов DVD+R и DVD+RW с той же ёмкостью, что у DVD-R и DVD-RW, но форматы «+» и «-» несовместимы, хотя большинство современных накопителей читают оба.

Самая распространённая файловая система DVD - Universal Disk Format (UDF), разработанная для пакетной записи как замена ISO9660.

Диски Blu-ray

В дисках Blu-ray (BD) применяются новые процессы физического производства, позволяющие ещё больше увеличить ёмкость. При этом BD остаются похожими на CD и DVD: в них используется спиральная дорожка данных, разделённая на секторы по 2048 байт.

Их стандарты имеют близкие аналоги среди CD и DVD и включают BD-ROM (только для чтения), BD-R (однократная запись), BD-RE (многократная запись) и BD-XL (многократная запись с удвоенной ёмкостью).

DVD и BD поддерживают защиту содержимого с помощью шифрования, что способно затруднить получение данных с защищённого диска во время криминалистического расследования. Инструменты и методы расшифровки содержимого, защищённого средствами управления цифровыми правами DRM, существуют, однако находятся за рамками книги.

Устаревшие оптические накопители

Существует ряд устаревших оптических накопителей, в частности картриджные устройства однократной записи и многократного чтения (Write Once, Read Many, WORM). Такие устройства чаще применялись в корпоративной среде, обычно использовали интерфейс SCSI и предоставляли доступ к данным как к линейной последовательности секторов. Если в распоряжении всё ещё имеются накопитель и совместимая плата контроллера, а ядро Linux распознаёт носитель, его можно читать и анализировать так же, как другие оптические диски.

Интерфейсы и физические разъёмы

В этом разделе я дам обзор распространённых интерфейсов накопителей с точки зрения эксперта-криминалиста. Работая в хорошо оснащённой криминалистической лаборатории, эксперт может получать и анализировать данные с носителей через самые разные интерфейсы устройств.

Общая тенденция вычислительной техники, особенно интерфейсов хранения, состоит в переходе от параллельных и совместно используемых шин к последовательным соединениям «точка - точка». Некогда популярный параллельный интерфейс PATA/IDE, в котором два диска использовали один кабель, сменился SATA с одним диском на последовательном кабеле. Параллельная общая шина SCSI поддерживала несколько дисков, а теперь её заменил SAS с отдельным последовательным разъёмом для каждого диска. Исходная шина PCI была параллельной и совместно использовалась несколькими интерфейсными платами, но её сменила PCI Express - последовательная шина с выделенными линиями для каждого интерфейса, которая также применяется для подключения накопителей SATA Express и NVM Express. Параллельный интерфейс принтера сменился USB - последовательным протоколом, который, однако, использует общую шину. С ростом скорости передачи согласовывать временные характеристики параллельных электрических сигналов стало сложно. Сериализация и десериализация данных на выделенных последовательных линиях позволили достичь более высоких скоростей, чем координация множества параллельных линий данных.

Serial ATA

Самым популярным сегодня внутренним интерфейсом носителей является SATA. Внутри персональных компьютеров большинство жёстких дисков, SSD и оптических накопителей подключаются к интерфейсам SATA на системной плате или дополнительных адаптерах шины. Последовательная архитектура SATA заменила Parallel ATA (PATA, или IDE), который прежде был преобладающим интерфейсом потребительских дисков.

Стандартом SATA управляет международная организация Serial ATA International Organization (http://www.sata-io.org/). В текущей третьей редакции SATA обеспечивает скорость до 6 Гбит/с; первая и вторая редакции поддерживали соответственно 1,5 и 3,0 Гбит/с. Помимо внутреннего интерфейса существует внешний eSATA, позволяющий непосредственно подключать внешние диски SATA. Интерфейс SATA показан на рис. 1-8.

Рис. 1-8. Интерфейс диска SATA

Для небольших переносных устройств был разработан уменьшенный форм-фактор mini-SATA (mSATA). Показанный на рис. 1-9 интерфейс mSATA позволяет подключать небольшие твердотельные накопители SATA прямо к системной плате без отдельного кабеля SATA.

Рис. 1-9. Интерфейс диска mSATA

Системные платы обычно позволяют устанавливать диски SATA в режиме Advanced Host Controller Interface (AHCI) или IDE. Стандартный интерфейс адаптера SATA AHCI определён по адресу http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/serial-ata-ahci-spec-rev1-3-1.pdf. Режим IDE предоставляет устаревший дисковый интерфейс для старых операционных систем, не поддерживающих стандарт AHCI, например ранних версий Windows XP. Выбранный режим не влияет на криптографический хеш криминалистического образа. Диск можно извлечь из исследуемого компьютера, работавшего в режиме IDE, подключить к машине эксперта в режиме AHCI и получить данные без потерь или изменений.

Другой интерфейс, micro SATA, который не следует путать с mSATA, показан на рис. 1-10. Он был разработан для 1,8-дюймовых дисков и тонких проигрывателей CD/DVD, но сегодня применяется реже. Для получения данных с накопителей mSATA и micro SATA можно использовать обычные блокираторы записи SATA вместе с различными переходниками.

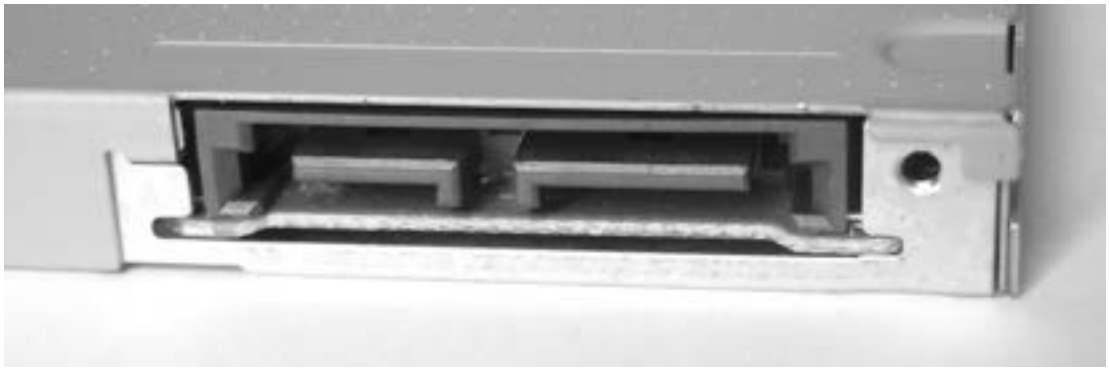

Рис. 1-10. Интерфейс диска Micro SATA

Более совершенный и набирающий популярность форм-фактор - интерфейс M.2, показанный на рис. 4-4. Представленный в спецификации SATA 3.2 интерфейс M.2 объединяет два стандарта в одном разъёме. В зависимости от требований совместимости плата M.2 может использовать интерфейс AHCI/SATA либо NVMHCI/NVME. При работе с платами M.2 обязательно выясняйте, какой именно интерфейс использует плата; на момент написания книги большинство плат M.2 на рынке работает в режиме ACHI/SATA.

> **Примечание переводчика.** В исходнике дана ссылка на рис. 4-4, тогда как расположенная ниже иллюстрация имеет номер 1-11. Обозначение ACHI/SATA также сохранено в том виде, в котором оно приведено автором.

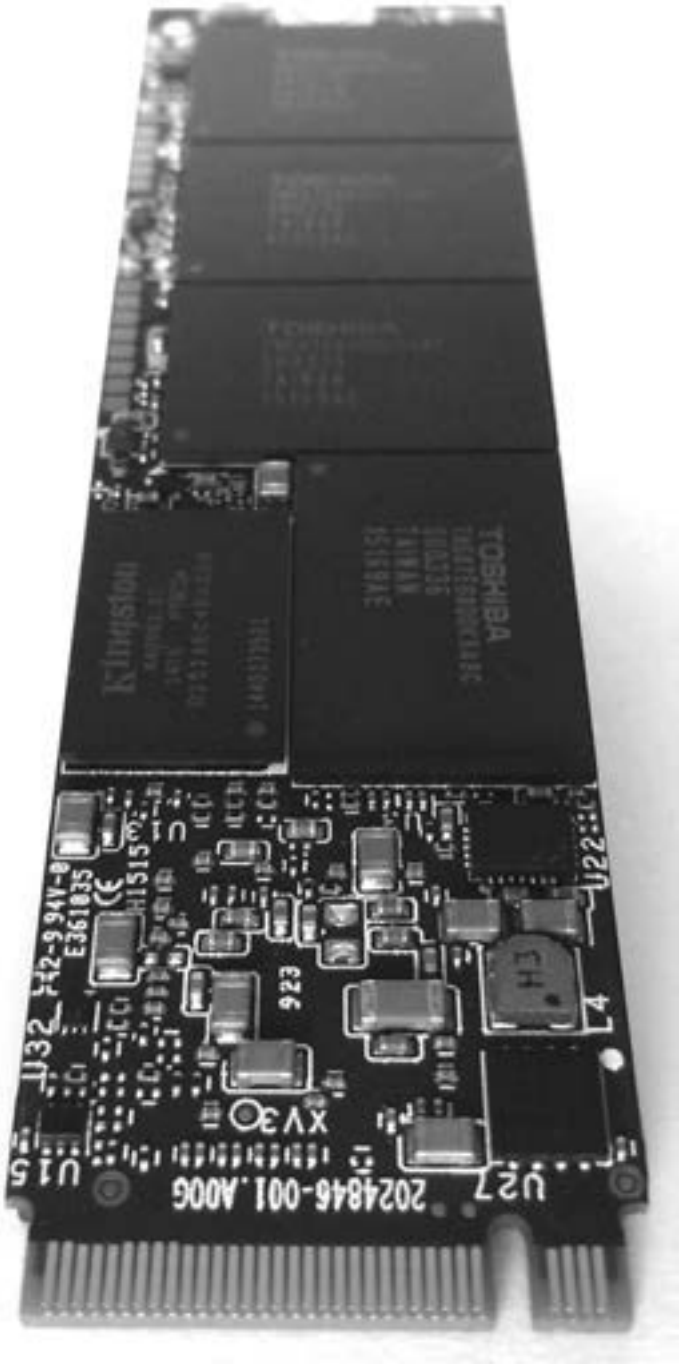

Рис. 1-11. Интерфейс диска M.2

Интерфейс дисков SATA Express, показанный на рис. 1-12, устраняет несколько уровней стека протоколов SATA, позволяя устройствам хранения, преимущественно SSD, подключаться непосредственно к шине PCI Express. Такие накопители продолжают использовать стандарт AHCI и отличаются от NVME.

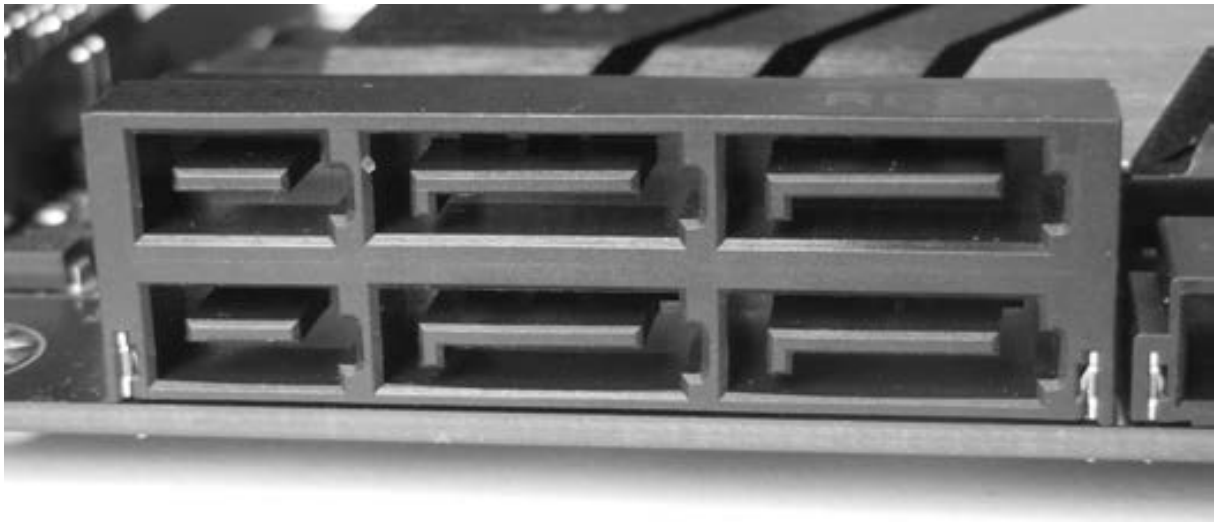

Рис. 1-12. Интерфейс диска SATA Express

Спецификация SATA 3.2 поддерживает две линии PCI Express, обеспечивающие SATA Express скорость 16 Гбит/с. Для накопителей SATA Express на основе PCI и M.2 существуют блокираторы записи.

Serial Attached SCSI и Fibre Channel

Параллельный интерфейс SCSI почти исчез как с потребительского, так и с корпоративного рынка. На потребительском рынке его заменил SATA, а на корпоративном - Serial Attached SCSI (SAS) и Fibre Channel (FC). Физический разъём интерфейса SAS, показанный на рис. 1-13, позволяет подключать как диски SAS, так и SATA и обращаться к ним через объединительную плату SAS. Физический разъём дисков SAS несколько отличается от разъёма дисков SATA; для подключения нескольких дисков к адаптеру шины выпускаются различные разветвительные кабели. Современные диски SAS-3 работают со скоростью 12 Гбит/с, вдвое быстрее SATA-3. SAS-4 обеспечит скорость 22,5 Гбит/с.

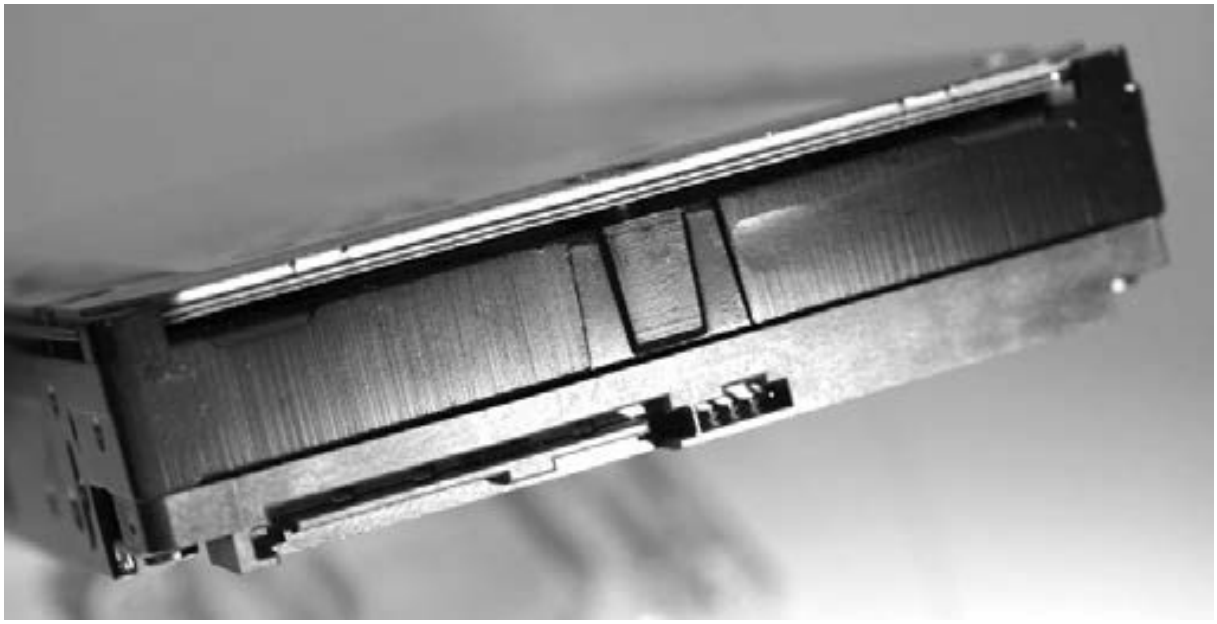

Рис. 1-13. Интерфейс диска SAS

Гнездовой разъём Mini-SAS HD 4i показан на рис. 1-14.

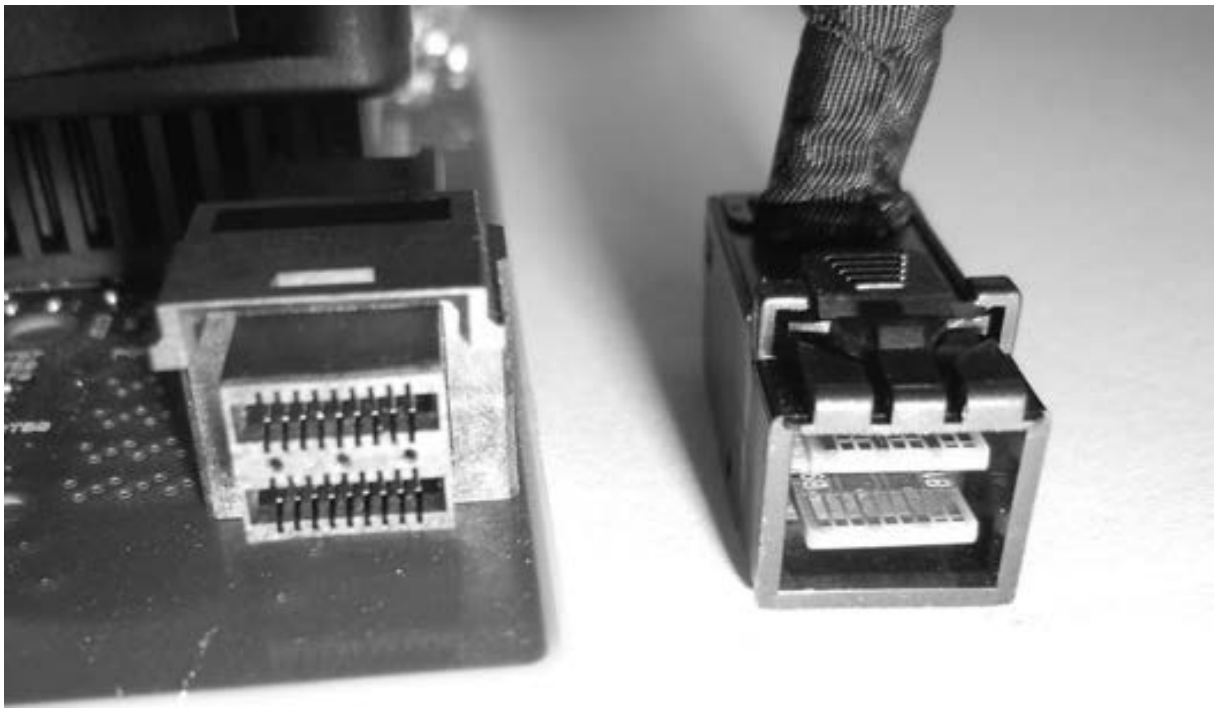

Рис. 1-14. Интерфейс SFF-8632/Mini-SAS HD

Стандарт SAS сопровождает технический комитет T10 Международного комитета по стандартам информационных технологий (International Committee for Information Technology Standards, INCITS). Действующий документ стандарта называется *Serial Attached SCSI-3 (SAS-3) INCITS 519-2014*. Дополнительные сведения, включая проекты будущих стандартов, доступны по адресу http://t10.org/.

Накопители SAS нельзя подключать к адаптерам шины SATA, а для создания образов дисков SAS требуются отдельные блокираторы записи SAS.

Интерфейс Fibre Channel часто применяется для подключения корпоративных массивов хранения. Медные и оптические разъёмы Fibre Channel показаны на рис. 1-15.

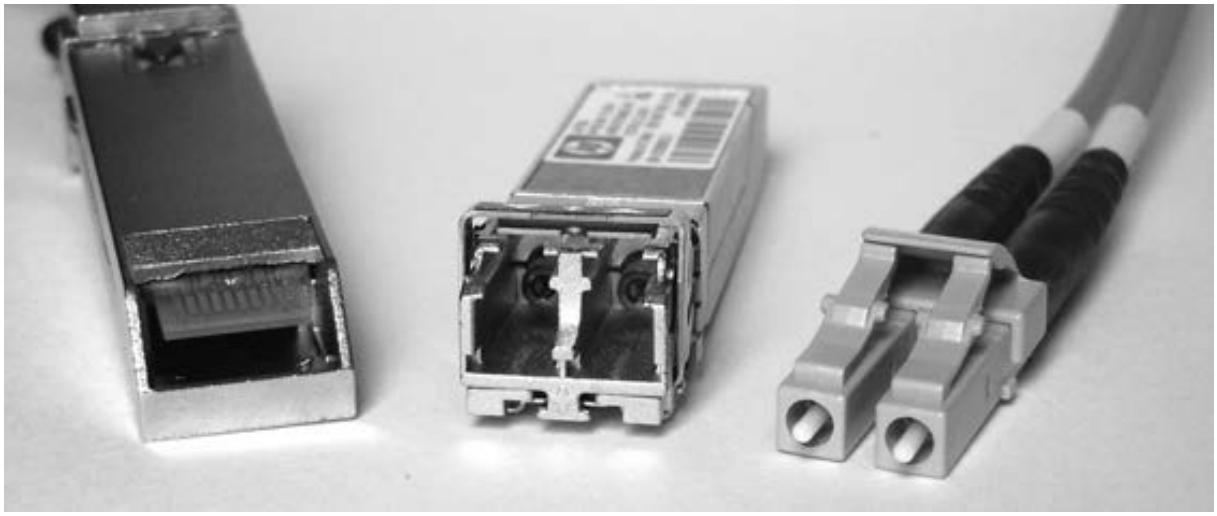

Рис. 1-15. Интерфейсы Fibre Channel

Жёсткие диски со встроенными интерфейсами Fibre Channel устаревают и в основном уже заменены накопителями SAS. На момент написания книги аппаратных блокираторов записи для дисковых интерфейсов Fibre Channel на рынке нет.

Non-Volatile Memory Express

Интерфейс Non-Volatile Memory Express (NVME) был с нуля спроектирован как альтернатива дисковому интерфейсу SATA на основе AHCI. Он предназначен для непосредственного подключения к шине PCI Express, поэтому не нуждается в адаптере шины AHCI/SATA и связанных с ним физических интерфейсах и уровнях протокола SATA. Архитектура NVME ориентирована именно на твердотельные накопители, для которых создан более простой и эффективный набор команд. Устройства NVME могут подключаться прямо к системной плате через разъём PCIE, использовать режим PCIE NVMe интерфейса M.2 либо интерфейс U.2 (SFF-8639). Физический интерфейс непосредственно подключён к шине PCIE, как показано на рис. 1-16.

Рис. 1-16. SSD NVME с интерфейсом PCIE

Существует также вариант M.2, или Next Generation Form Factor (NGFF). Такие устройства вставляются непосредственно в разъёмы M.2 системной платы либо в платы-переходники для разъёмов PCIE; в обоих случаях применяется режим NVME, а не AHCI/SATA. В настоящее время большинство дисков SSD M.2 используют AHCI/SATA, а не NVME, однако это может измениться, поскольку преимущества NVME в производительности весьма убедительны. Диск NVME M.2 показан на рис. 1-17.

Рис. 1-17. SSD NVME с интерфейсом M.2

Интерфейс NVME U.2 (см. рис. 1-18) позволяет подключать по кабелю или через объединительную плату накопители в традиционном 2,5-дюймовом форм-факторе. Интерфейс и кабель U.2 (SFF-8639), механически похожие на SAS, но имеющие дополнительные контакты для линий PCIE, соединяют корпус накопителя со штекером mini-SAS HD на переходнике M.2, установленном на системной плате.

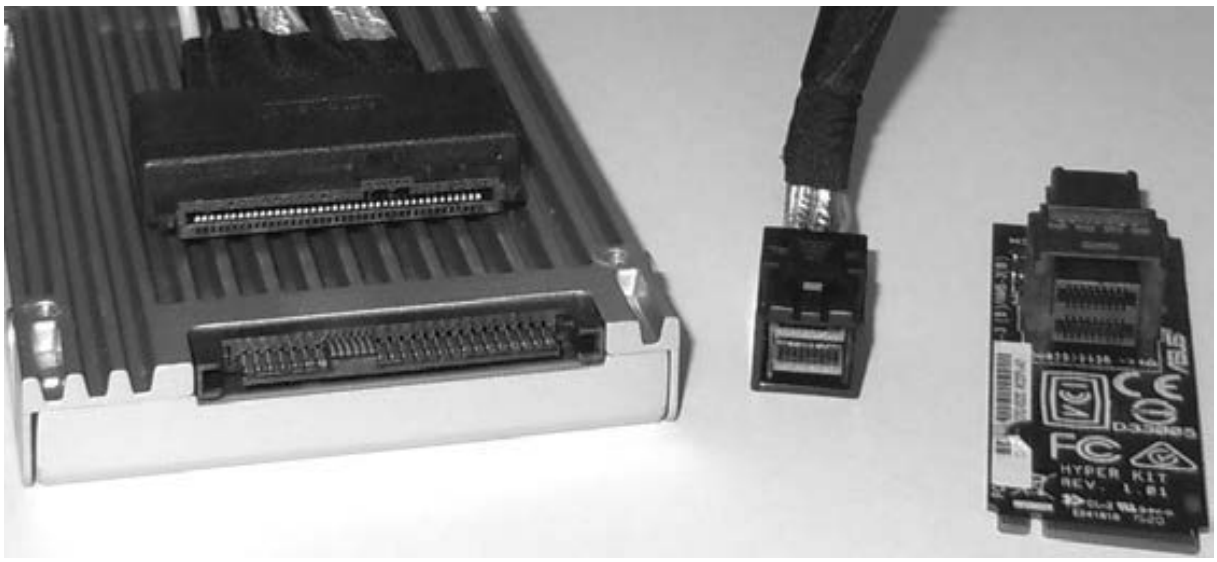

Рис. 1-18. SSD форм-фактора 2,5 дюйма с интерфейсом U.2, кабель-переходник U.2 - mini-SAS HD и переходник mini-SAS HD - M.2 для системной платы

Диски NVME не видны обычным инструментам команд SCSI или SATA, поскольку не используют команды SCSI или SATA. Инструменты Linux, предназначенные для взаимодействия с дисковыми интерфейсами ATA или SCSI, как правило, не работают с дисками NVME без дополнительной поддержки совместимости. У дисков NVME не привычные имена файлов `/dev/sd*`, а `/dev/nvme*n*`. Устройство NVME создаётся с дополнительным номером пространства имён. Пространство имён, обозначенное буквой `n`, представляет собой распределение пространства на накопителе NVME на уровне ниже операционной системы. Дисковые устройства `/dev/nvme*n*` являются блочными и должны нормально работать при обращении к файлу устройства Linux. Например, ниже показана команда `mmls` из Sleuth Kit, работающая с диском NVME:

```
# mmls /dev/nvme0n1
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
00:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----     0000000000   0000002047   0000002048   Unallocated
02:  00:00     0000002048   0781422767   0781420720   Linux (0x83)
#
```

На момент написания книги NVME оставался сравнительно новой технологией, а компания Tableau лишь недавно создала первый криминалистический аппаратный блокиратор записи PCI Express с поддержкой NVME. Из-за сложности перехвата команд NVME на шине PCIE блокираторы записи дороги и сложны в реализации. Дополнительные сведения см. в моей статье “NVM Express Drives and Digital Forensics”, где эта проблема рассмотрена подробнее.[2]

Universal Serial Bus

Интерфейс USB был создан для объединения и замены устаревающих внешних интерфейсов периферийных устройств, таких как RS-232, параллельный интерфейс принтера, интерфейс клавиатуры и мыши PS/2 и другие проприетарные интерфейсы персональных компьютеров. Он проектировался как универсальный интерфейс для дисков, клавиатур, мышей, звуковых устройств, сетевых подключений, принтеров, сканеров и небольших подключаемых устройств, например мобильных телефонов. Всё больше устройств Интернета вещей (Internet of Things, IoT) можно подключить к персональному компьютеру по USB, и они могут содержать данные, полезные в качестве криминалистических доказательств. Поскольку эта книга посвящена криминалистическому получению данных с устройств массового хранения, я ограничу обсуждение устройствами хранения USB.

Флеш-накопители, оптические приводы, некоторые ленточные накопители и даже магнитные жёсткие диски (см. рис. 1-19) могут иметь интерфейс USB, непосредственно встроенный в электронику устройства. Чаще всего класс массовой памяти USB представлен потребительскими USB-накопителями, или флешками, и внешними корпусами для жёстких дисков.

Рис. 1-19. Магнитный жёсткий диск форм-фактора 1,8 дюйма со встроенным интерфейсом USB

Исходный протокол USB для устройств класса массовой памяти известен как Bulk-Only Transport (BOT). Сейчас BOT является наиболее распространённым транспортным протоколом USB, однако с ростом скорости дисков и появлением USB3 он превращается в узкое место и может быть заменён протоколом USB Attached SCSI Protocol (UASP). Подобно AHCI для SATA, USB также имеет определённый стандарт интерфейса контроллера основной системы. Extensible Host Controller Interface (xHCI) заменяет несколько более старых стандартов USB, в частности OHCI, UHCI и EHCI. Его спецификация доступна по адресу http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/extensible-host-controler-interface-usb-xhci.pdf.

Новейший интерфейс USB 3.1 - Type C, показанный на рис. 1-20. Type C является многофункциональным интерфейсом: он подходит для устройств USB 3.1 и Thunderbolt 3, а также для подачи питания. Физический штекер можно вставлять любой стороной, поэтому при подключении к системе не нужно совмещать верх и низ.

Рис. 1-20. Интерфейс USB Type C

Во внешних корпусах дисков USB обычно находится один или несколько накопителей SATA. Если это осуществимо, диски SATA следует извлекать, чтобы получить прямой доступ к интерфейсу ATA. Это позволяет непосредственно опрашивать интерфейс накопителя и в некоторых случаях даёт преимущество в производительности, например для корпусов USB 2.0 с диском SATA.

Для устройств USB существуют специально предназначенные криминалистические блокираторы записи. Их можно использовать с накопителями, флеш-памятью и другими устройствами, имеющими встроенный интерфейс USB.

Thunderbolt

Thunderbolt был совместно разработан Apple и Intel как высокоскоростной внешний интерфейс для подключения дисков, видеомониторов и устройств PCI Express через единый разъём (см. рис. 1-21). Первоначально он носил кодовое название Light Peak и задумывался как оптоволоконное соединение. В Thunderbolt 1 и Thunderbolt 2 физическим интерфейсом служит Mini DisplayPort, а Thunderbolt 3 перешёл на кабель и разъём USB Type C. Популярность Thunderbolt, прежде всего среди пользователей Apple, в значительной мере объясняется компанией Apple, продвигавшей его вместе со своим оборудованием. Интерфейс Thunderbolt 3 объединяет PCI Express, DisplayPort и USB3. Thunderbolt 1, 2 и 3 обеспечивают скорость соответственно 10, 20 и 40 Гбит/с.

Рис. 1-21. Интерфейс Thunderbolt

На компьютерах Apple режим внешнего диска Target Disk Mode (TDM) можно использовать с Thunderbolt, как и с FireWire, чтобы система представлялась другому подключённому компьютеру в виде внешнего корпуса диска. TDM предписывает микропрограмме Apple предоставить внутренние диски как блочные устройства, к которым можно обращаться как к внешним накопителям SCSI, что удобно при работе с криминалистическими инструментами. Криминалистическое получение данных с компьютера Apple посредством TDM я продемонстрирую в разделе «Режим внешнего диска Apple» на странице 137.

Во внешних дисках Thunderbolt обычно находится один или несколько накопителей SATA. В крупных корпусах RAID с Thunderbolt интерфейс может использовать контроллер SAS вместе с несколькими дисками SATA или SAS.

Вывод Linux `dmesg` для внешнего диска, подключённого через Thunderbolt, выглядит следующим образом:

```
[ 53.408619] thunderbolt 0000:05:00.0: 0:1: hotplug: scanning
[ 53.408792] thunderbolt 0000:05:00.0: 0:1: is connected, link is up (state: 2)
[ 53.408969] thunderbolt 0000:05:00.0: initializing Switch at 0x1 (depth: 1,
up port: 1)
...
[ 53.601118] thunderbolt 0000:05:00.0: 1: hotplug: activating pcie devices
[ 53.601646] thunderbolt 0000:05:00.0: 0:6 <-> 1:2 (PCI): activating
...
[ 53.602444] thunderbolt 0000:05:00.0: path activation complete
[ 53.602679] pciehp 0000:04:03.0:pcie24: Card present on Slot(3-1)
[ 53.609205] pciehp 0000:04:03.0:pcie24: slot(3-1): Link Up event
...
[ 56.375626] ata7: SATA link up 6.0 Gbps (SStatus 133 SControl 300)
[ 56.382070] ata7.00: ATA-8: ST1000LM024 HN-M101MBB, 2BA30003, max UDMA/133
[ 56.382074] ata7.00: 1953525168 sectors, multi 0: LBA48 NCQ (depth 31/32), AA
[ 56.388597] ata7.00: configured for UDMA/133
[ 56.388820] scsi 7:0:0:0: Direct-Access ATA ST1000LM024 HN-M 0003 PQ: 0
ANSI: 5
[ 56.389341] sd 7:0:0:0: Attached scsi generic sg2 type 0
[ 56.389342] sd 7:0:0:0: [sdc] 1953525168 512-byte logical blocks:
(1.00 TB/931 GiB)
[ 56.389345] sd 7:0:0:0: [sdc] 4096-byte physical blocks
```

```
[ 56.389408] sd 7:0:0:0: [sdc] Write Protect is off
[ 56.389413] sd 7:0:0:0: [sdc] Mode Sense: 00 3a 00 00
[ 56.389449] sd 7:0:0:0: [sdc] Write cache: enabled, read cache: enabled, doesn't
support DPO or FUA
[ 56.403702] sdc: [mac] sdc1 sdc2
[ 56.404166] sd 7:0:0:0: [sdc] Attached SCSI disk
```

На момент написания книги ядро Linux поддерживает интерфейсы Thunderbolt на компьютерах Apple. Дополнительные платы для системных плат персональных компьютеров, например ASUS ThunderboltEX II, не поддерживают горячее подключение устройств. Однако если загрузить компьютер с уже подключённым диском Thunderbolt, BIOS или микропрограмма компьютера инициализирует адаптер Thunderbolt до загрузки операционной системы, благодаря чему внешние диски станут видны ядру.

На момент написания книги блокираторов записи для интерфейсов Thunderbolt на рынке нет. Накопителей с непосредственно встроенным интерфейсом Thunderbolt тоже пока не существует - доступны только корпуса.

Устаревшие интерфейсы

Этот раздел не излагает долгую историю компьютерных дисковых интерфейсов, а лишь кратко рассматривает недавно устаревшие технологии IDE, параллельного SCSI и FireWire. Они по-прежнему могут иметь значение для криминалистического расследования, когда обнаруженное или изъятое в качестве доказательства оборудование оказывается старым.

Интерфейс IDE (см. рис. 1-22) и его усовершенствованная версия EIDE, обычно применявшиеся для 3,5-дюймовых дисков, были популярны до того, как их заменил SATA.

Рис. 1-22. Интерфейс диска IDE

Интерфейс mini IDE (см. рис. 1-23) был разработан для 2,5-дюймовых дисков ноутбуков и применялся до замены на SATA.

Рис. 1-23. Интерфейс диска Mini IDE

Интерфейс micro IDE ZIF (см. рис. 1-24) был разработан для 1,8-дюймовых жёстких дисков субноутбуков и других небольших электронных устройств и применялся до того, как его заменили интерфейсы mSATA и M.2.

Рис. 1-24. Интерфейс диска Micro IDE ZIF

Интерфейс FireWire, или IEEE1394 (см. рис. 1-25), был разработан компанией Apple как высокоскоростная внешняя шина для подключения видеооборудования и дисковых накопителей. В основном его заменили Thunderbolt и USB3.

Рис. 1-25. Интерфейсы FireWire

Параллельный интерфейс SCSI (см. рис. 1-26) почти исчез с потребительского рынка; преимущественно его заменил SATA, а в корпоративной среде - SAS.

Рис. 1-26. Интерфейсы SCSI

Криминалистические блокираторы записи для IDE, SCSI и FireWire широко распространены и работают также с переходниками для mini IDE, micro IDE ZIF и различных интерфейсов SCSI.

Команды, протоколы и мосты

Обмен данными между устройствами хранения и компьютерными системами концептуально в чём-то похож на многоуровневую организацию сетей LAN/WAN.[3] Его также можно разделить на несколько уровней абстракции. Физический уровень состоит из кабелей, проводников и электрических сигналов. Над ним расположен канальный уровень, где цифровые биты и байты передаются упорядоченным образом в кадрах или пакетах канального уровня. Поверх канального уровня отправитель и получатель обмениваются протоколами и командами для запроса и получения данных. В предыдущих разделах я описал физические соединения и дисковые интерфейсы. Здесь будут рассмотрены наборы команд более высоких уровней для ATA, SCSI и NVME. Рисунок 1-27 помогает представить разные уровни абстракции в общей системе.

Команды ATA

Современные команды Advanced Technology Attachment (ATA) первоначально развились из стандарта Американского национального института стандартов (American National Standards Institute, ANSI), определявшего интерфейс AT Attachment для дисковых накопителей.[4] Исходный стандарт описывал физический интерфейс - кабели, контакты и прочее, - электрические сигналы и логические команды, которые можно было подавать.

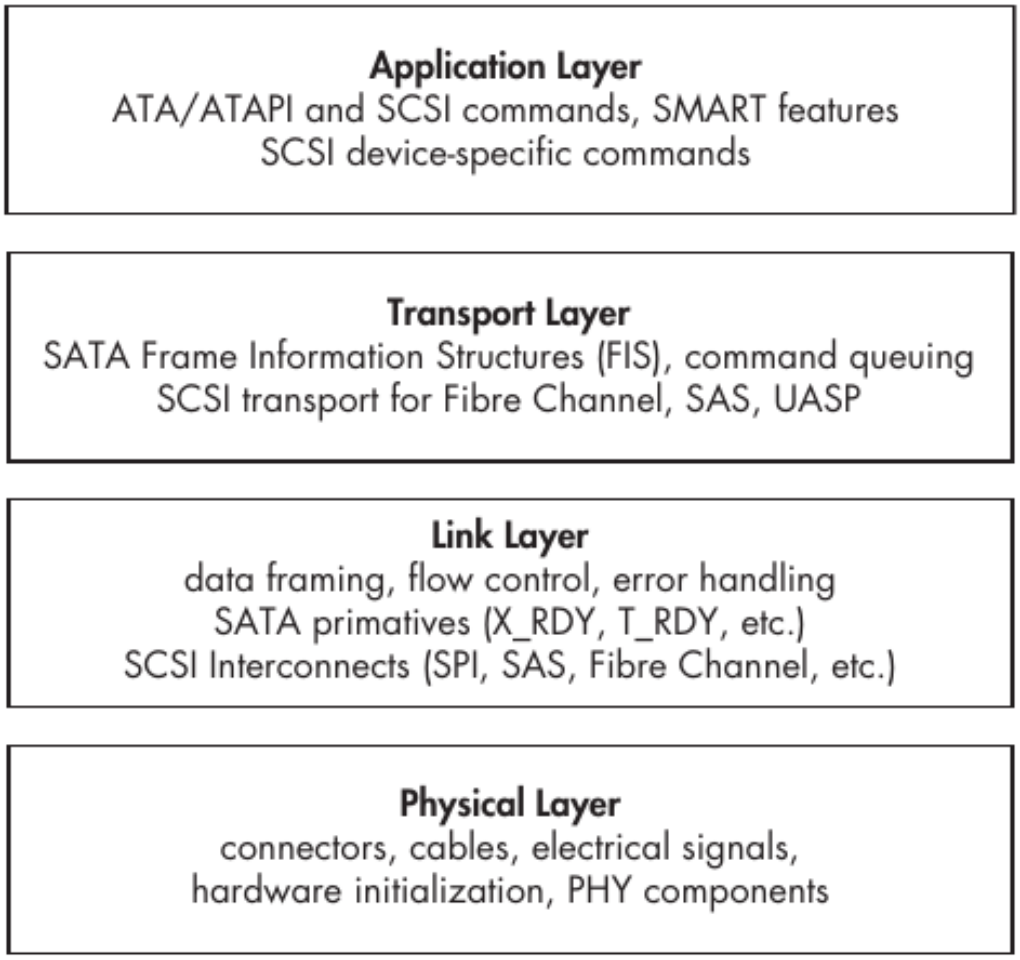

Рис. 1-27. Уровни абстракции применительно к дисковым интерфейсам

Перевод текста на схеме.

- **Прикладной уровень:** команды ATA/ATAPI и SCSI, функции SMART; команды для конкретных устройств SCSI.
- **Транспортный уровень:** информационные структуры кадров SATA (FIS), очереди команд; транспорт SCSI для Fibre Channel, SAS и UASP.
- **Канальный уровень:** формирование кадров данных, управление потоком, обработка ошибок; примитивы SATA (`X_RDY`, `T_RDY` и другие); соединения SCSI (SPI, SAS, Fibre Channel и другие).
- **Физический уровень:** разъёмы, кабели, электрические сигналы, инициализация оборудования и компоненты PHY.

Современный стандарт[5] описывает наборы функций, которыми могут обладать накопители, и команды ATA, доступные для управления каждым устройством. Интерфейс пакетных команд ATA (ATA Packet Interface, ATAPI) добавляет в стандарт ATA набор пакетных команд, позволяя применять дополнительные команды, не относящиеся непосредственно к функциям дискового накопителя, например извлекать носитель, инкапсулировать команды SCSI и выполнять другие действия.

Набор команд ATA (ATA Command Set, ACS) определяет команды и параметры, которые можно загрузить в регистры ATA и отправить накопителю для выполнения. Некоторые распространённые команды ATA приведены в табл. 1-1.

Таблица 1-1. Распространённые команды ATA

Команда	Код команды
`DEVICE RESET`	`08h`
`READ SECTOR(S)`	`20h`
`WRITE SECTOR(S)`	`30h`
`SEEK`	`70h`
`DOWNLOAD MICROCODE`	`92h`
`MEDIA EJECT`	`EDh`
`SECURITY UNLOCK`	`F2h`

Многие команды ATA и ATAPI можно подавать с помощью утилит `hdparm` и `smartctl`. Комитет T13 (http://t13.org/) публикует стандарт, определяющий полный набор команд. Дополнительные материалы доступны по адресу http://sata-io.org/.

Понимание основных принципов работы команд ATA/ATAPI даёт знания, необходимые для понимания действия криминалистических блокираторов записи SATA и IDE, которые не позволяют командам ATA/ATAPI изменять диски. Знание принципов выполнения команд также помогает понять, каким образом эксперт-криминалист может запрашивать атрибуты диска и в конечном счёте успешно получать с него данные.

Команды SCSI

Обмен командами SCSI между адаптером шины и диском соответствует модели «клиент - сервер». Адаптер шины основной системы (Host Bus Adapter, HBA) выступает клиентом, или инициатором, а диск - сервером, или целью. Инициатор отправляет цели команды, возможно вместе с данными, а цель возвращает ответ, также, возможно, сопровождаемый данными. Изначально SCSI проектировался для самых разных устройств, включая сканеры, ленты, принтеры и не только жёсткие диски, поэтому обладает богатым набором команд. Команды передаются от инициатора (HBA) к цели (накопителю) в блоке дескриптора команды (Command Descriptor Block, CDB), содержащем саму команду и её параметры. Цель получает CDB, выполняет запрос и возвращает ответ. Это взаимодействие отчасти напоминает обмен запросом и ответом на основе UDP в клиент-серверной архитектуре TCP/IP.

Команды SCSI позволяют читать блоки данных с диска и записывать их на него, управлять оборудованием - извлекать компакт-диски или менять ленты, - сообщать состояние и диагностическую информацию и выполнять другие действия. Обычно эти команды скрыты от конечного пользователя, но существуют различные утилиты, позволяющие подавать команды SCSI из пользовательского пространства. С помощью инструментов пакета `sg3_utils` из командной строки Linux можно отправлять даже произвольные команды. Например, следующая команда подаёт низкоуровневую команду SCSI для чтения первого сектора диска SCSI:

```
# sg_raw -r 512 /dev/sda 08 00 00 00 01 00
SCSI Status: Good

Received 512 bytes of data:
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
...
1a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1b0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1c0 02 00 ee ff ff ff 01 00 00 00 ff ff ff ff 00 00 ................
1d0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1e0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
1f0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa ..............U.
```

В этом примере `sg_raw` получает команду CDB, которую следует отправить диску SAS `/dev/sda`. Первый байт, `0x08`, задаёт шестибайтовую команду SCSI `READ`. Последующие нули указывают LUN и начальный сектор. Значение `0x01` задаёт число читаемых секторов, а параметр `-r 512` сообщает `sg_raw`, сколько байтов вывести. Этот пример должен работать и с накопителями, подключёнными через SAS, SATA и USB.

Технические стандарты команд SCSI и SAS поддерживает технический комитет T10 организации INCITS (InterNational Committee for Information Technology Standards). Эти стандарты доступны по адресу http://t10.org/. Лучше понять протокол команд SCSI помогут книги по программированию SCSI.

Понимание основных принципов работы команд SAS/SCSI имеет значение для этой книги. Оно даёт знания, необходимые для понимания действия криминалистических блокираторов записи SAS/SCSI, которые не позволяют командам SAS/SCSI изменять диски. Знание принципов работы SAS/SCSI помогает понять, как эксперт-криминалист может запрашивать атрибуты диска и в конечном счёте выполнять криминалистическое получение данных.

Команды SCSI также применяются для управления лентами и оптическими устройствами.

Команды NVME

Набор команд NVME был создан с нуля, без обратной совместимости с существующими наборами команд SCSI или ATA. Он предназначен для SSD, непосредственно подключённых к шине PCI Express, и использует преимущества параллельной обработки несколькими процессорами и мгновенного позиционирования SSD, которому не присуща задержка на перемещение дисковых головок. Разработчики стандарта также учли, что непосредственное подключение накопителя к шине PCIE позволяет устранить значительную часть накладных расходов протоколов ATA или SCSI. В результате появился новый минимальный набор без устаревших команд и требований обратной совместимости. Производительность и эффективность накопителей NVME значительно выше, чем SATA или SAS. Развитая система очередей команд предоставляет до 64 тысяч очередей, каждая из которых способна содержать 64 тысячи команд; для сравнения, SATA имел 32 очереди по одной команде.

Таблица соответствий между командами SCSI и NVME доступна по адресу http://nvmexpress.org/. Наиболее распространённые примеры приведены в табл. 1-2.

Таблица 1-2. Сопоставление команд SCSI и NVME

SCSI	NVME
COMPARE AND WRITE	Compare and Write
READ	Read
WRITE	Write
WRITE BUFFER	Firmware Image Download, Firmware Image Activate
INQUIRY	Identify
READ CAPACITY	Identify
REPORT LUNS	Identify
MODE SENSE	Identify, Get Features
LOG SENSE	Get Features, Get Log Page
SYNCHRONIZE CACHE	Flush
FORMAT UNIT	Format NVM

Дисковые устройства можно подключать к компьютерной системе через различные шины и мосты. Производительность повышается, если накопитель подключён как можно ближе к процессору и памяти. Сегодня минимальное расстояние до процессора и памяти обеспечивает интерфейс NVME на выделенных линиях шины PCI Express 3.0. Диски в оперативной памяти быстрее и эффективнее, однако создаются операционной системой и не являются устройствами энергонезависимого хранения. Устройство NVME можно подключить непосредственно к процессору без южного моста набора микросхем и накладных расходов традиционного дискового протокола, такого как ATA или SCSI.

Мосты, туннелирование и сквозная передача

ATA и SCSI - два самых распространённых протокола взаимодействия с носителями. Команды можно передавать через различные шины физического уровня или транспортные уровни и даже туннелировать либо инкапсулировать внутри других протоколов. Эта сложность скрыта от пользователя, а к подключённым устройствам можно обращаться через стандартные блочные устройства, предоставляемые ядром Linux.

В качестве примера рассмотрим жёсткий диск SATA, вставленный в автономную док-станцию USB, которая подключена к внешнему порту платы USB3 PCI Express, установленной в персональном компьютере. Обмен между интерфейсом диска и док-станцией происходит на нижнем уровне протокола SATA с помощью пакетов Frame Information Structures (FIS). Док-станция и плата USB3 взаимодействуют на нижнем уровне протокола USB посредством BOT или USAP. Обмен между платой USB3 и компьютером происходит на нижнем уровне протокола PCI Express посредством пакетов уровня транзакций PCIE (Transaction Layer Packets, TLP) и пакетов канального уровня (Data Link Layer Packets, DLLP). Наконец, обращение к диску через все эти мосты выполняется по протоколу команд SCSI. Таким образом, для подключения диска используется несколько физических и канальных уровней. Пользователю кажется, что диск доступен непосредственно, а нижние уровни протоколов скрыты от него абстракцией.

> **Примечание.** FIS входят в протокол SATA. У PCI Express есть собственный набор протоколов, включающий TLP и DLLP.

Для каждой физической шины обмен между подключёнными к ней устройствами обеспечивает компонент PHY, название которого происходит от слова physical. Работа компонентов PHY на шине отчасти напоминает взаимодействие двух модемов по кабелю WAN. PHY преобразует цифровые единицы и нули в соответствующие требованиям электрические сигналы, ожидаемые на данной шине. После того как PHY обслужит физический уровень, протокол канального уровня управляет потоком битов и байтов, передаваемым по шине в обоих направлениях. Канальный уровень организует поток данных в кадры или отдельные пакеты информации, которые могут обрабатывать верхние уровни.

Для USB, PCIE, Thunderbolt, SAS, SATA, Fibre Channel и других технологий существуют описания физического и канального уровней. Обычно стандарты позволяют универсальным драйверам операционной системы работать с физическим оборудованием независимо от конкретного устройства. Например, адаптеры USB определены стандартом xHCI, адаптеры SATA - стандартом AHCI, а устройства NVME - стандартом NVMHCI, который теперь называется просто стандартом NVME. Соответствие этим стандартам позволяет поддерживать оборудование без дополнительных проприетарных драйверов.

Хотя ATA/ATAPI и SCSI представляют собой разные наборы команд, оба в некоторой степени поддерживают туннелирование и сквозную передачу команд друг друга. ATA использует интерфейс ATAPI для инкапсуляции команд SCSI при взаимодействии с оптическими и ленточными накопителями и другими устройствами, которые понимают команды SCSI для извлечения носителей и иные команды, отсутствующие в протоколе ATA. SCSI поддерживает сквозную передачу ATA, позволяющую отправлять команды ATA по протоколу SCSI. Трансляция SAT (SCSI-ATA Translation) обеспечивает двустороннее преобразование между командами SCSI и ATA при их взаимодействии с носителями. В Linux это преобразование реализовано как программный интерфейс ядра в библиотеке `libata`. Библиотека `libata`, название которой иногда записывают как `libATA`, предоставляет ряд функций для взаимодействия с контроллерами и устройствами ATA.

Мосты играют важную роль в криминалистическом получении данных, поскольку лежат в основе аппаратных блокираторов записи. Обычно аппаратный блокиратор записи представляет собой мост, способный перехватывать отправляемые диску команды ATA или SCSI и не позволять командам изменять его секторы.

Специальные темы

В этом разделе рассматривается ряд специальных тем, связанных с массовым хранением и цифровой криминалистикой. Значение некоторых областей, таких как UASP и SSHD, для цифровой криминалистики комментируется лишь кратко. Другие области, включая DCO, HPA и накопители NVME, здесь представлены впервые и подробнее рассматриваются далее в книге.

Области диска DCO и HPA

По мере развития стандартов ATA/ATAPI создавались функции, призванные помочь изготовителям систем. Защищённая область основной системы (Host Protected Area, HPA) появилась в стандарте ATA/ATAPI-4 и позволила изготовителям резервировать части диска для использования за пределами обычной операционной системы. Например, в HPA можно хранить постоянные данные, сведения для восстановления системы, данные гибернации и тому подобное. Доступ к ним контролирует микропрограмма системы, а не установленная операционная система. Функция наложения конфигурации устройства (Device Configuration Overlay, DCO) появилась в стандарте ATA/ATAPI-6 и позволила управлять сообщаемыми характеристиками и ёмкостью диска. Благодаря этому изготовители могли поставлять диски разных производителей, сохраняя одинаковое число доступных пользователю секторов и одинаковые функции. Это упрощало поддержку и замену накопителей. HPA и DCO могут сосуществовать, однако сначала необходимо создать DCO, а затем HPA.

Преступники и злоумышленники использовали HPA и DCO не по назначению, скрывая незаконные файлы и код вредоносных программ. Обнаружение и удаление HPA и DCO, открывающее секторы, скрытые от обычного пользователя, описано в разделе «Предоставление доступа к скрытым секторам» на странице 118. Более подробное криминалистическое описание HPA и DCO приведено в статье “Hidden Disk Areas: HPA and DCO”.[6] Сведения об использовании HPA государственными структурами также доступны по адресам https://www.schneier.com/blog/archives/2014/02/swap_nsa_exploi.html и https://leaksource.files.wordpress.com/2013/12/nsa-ant-swap.jpg.

Служебные области диска

К некоторым служебным областям жёсткого диска обычно нельзя обращаться стандартными инструментами Linux. Эти области содержат списки повреждённых секторов и служебные секторы изготовителя, которые иногда называют отрицательными секторами. Для доступа к ним требуется специализированное программное и аппаратное обеспечение диагностики дисков. Пример доступа к служебной области диска и дополнительные материалы приведены в разделе «Доступ к служебной области накопителя» на странице 122. Не путайте служебную область диска с HPA или DCO. К областям HPA и DCO легко обратиться с помощью стандартных команд и методов ATA, чего нельзя сделать со служебными областями диска.

USB Attached SCSI Protocol

USB предоставляет два режима доступа к устройствам класса массовой памяти: более распространённый BOT и новый UASP. В связи с возросшими скоростями USB3 и USB3.1 был разработан новый, более эффективный транспортный протокол класса массовой памяти USB под названием USB Attached SCSI Protocol (UASP). Этот новый протокол также называют UAS, а в маркетинговых названиях продуктов иногда употребляются обозначения USB3 Boost, USB3 Turbo или USB3 Extreme. Дополнительные сведения доступны на сайтах http://usb.org/ и http://t10.org/ организаций, совместно разработавших стандарт. UASP повышает производительность благодаря очередям команд, асинхронной обработке и усовершенствованным возможностям управления задачами и командами.

В выводе `dmesg` подключённого диска USB с поддержкой UASP видно, что для работы применяется протокол `uas`:

```
[15655.838498] usb 2-6.2: new SuperSpeed USB device number 6 using xhci_hcd
...
[15655.952172] scsi host14: uas
...
[15666.978291] sd 14:0:0:0: [sdk] 3907029168 512-byte logical blocks:
(2.00 TB/1.81 TiB)
...
[15667.033750] sd 14:0:0:0: [sdk] Attached SCSI disk
```

Напротив, когда тот же диск подключён через традиционный интерфейс USB BOT, для работы загружается протокол `usb-storage`:

```
[15767.853288] usb 2-6.2: new SuperSpeed USB device number 7 using xhci_hcd
...
[15767.918079] usb-storage 2-6.2:1.0: USB Mass Storage device detected
[15767.918195] usb-storage 2-6.2:1.0: Quirks match for vid 174c pid 55aa: 400000
[15767.918222] scsi host15: usb-storage 2-6.2:1.0
...
[15777.728944] sd 15:0:0:0: [sdk] 3907029168 512-byte logical blocks:
(2.00 TB/1.81 TiB)
...
[15777.820171] sd 15:0:0:0: [sdk] Attached SCSI disk
```

С криминалистической точки зрения важно отметить, что применяемый транспортный протокол не влияет на содержимое диска USB и не изменяет криптографический хеш криминалистического образа. Более того, блокираторы записи на основе UAS выгодно применять ради преимуществ в производительности; например, блокираторы записи Tableau USB3 используют UAS.

> **Примечание.** Небольшой совет: при работе на повышенных скоростях USB3 важным становится качество кабелей USB. Длинные кабели USB3 низкого качества способны вызывать ошибки чтения во время получения данных. Профессиональной криминалистической лаборатории стоит приобрести короткие высококачественные кабели USB3.

Расширенный формат 4Kn

С ростом ёмкости дисков отрасль обнаружила, что эффективность можно повысить, перейдя от 512-байтовых секторов к секторам размером 4096 байт. Международная ассоциация оборудования и материалов для дисковых накопителей (International Disk Drive Equipment and Materials Association, IDEMA) разработала стандарт Advanced Format для физических секторов по 4096 байт (см. http://www.idema.org/?page_id=2369). С 2009 года изготовители жёстких дисков обязались применять Advanced Format IDEMA для выпуска дисков с секторами по 4 Кбайт. Даже при физических секторах по 4 Кбайт большинство современных дисков эмулирует 512-байтовые секторы; такие устройства называются Advanced Format 512e. Диски, предоставляющие основной системе и операционной системе собственные секторы по 4 Кбайт, называются Advanced Format 4Kn. В нижнем ценовом сегменте накопители Advanced Format 4Kn пока редки, но применяются в корпоративной среде. Для корпоративных дисков большой ёмкости большинство изготовителей предлагает две модели: 512e и 4Kn. Официальный логотип дисков 4Kn показан на рис. 1-28.

Хороший обзор Advanced Format и секторов по 4 Кбайт можно найти на YouTube: https://www.youtube.com/watch?v=TmH3iRLhZ-A/.

Рис. 1-28. Логотип Advanced Format 4Kn

Когда ядро Linux обнаруживает подключённый диск, оно выводит число секторов и размер логического сектора, а иногда явно сообщает и физический размер. В приведённом ниже фрагменте вывода `dmesg` показаны два диска одинаковой ёмкости: один Advanced Format 512e, другой 4Kn. Если разделить число 512-байтовых секторов на 8 или умножить число секторов по 4 Кбайт на 8, станет видно, что ёмкость дисков одинакова, но количество секторов различается.

```
...
[ 13.605000] scsi 1:0:1:0: Direct-Access TOSHIBA MG03SCA300 0108 PQ: 0
ANSI: 5
...
[ 16.621880] sd 1:0:1:0: [sdb] 5860533168 512-byte logical blocks: (3.00 TB/2.73
TiB)
...
[ 14.355068] scsi 1:0:2:0: Direct-Access ATA TOSHIBA MG04ACA3 FP2A PQ: 0
ANSI: 6
...
[ 16.608179] sd 1:0:2:0: [sdc] 732566646 4096-byte logical blocks: (3.00 TB/2.73
TiB)
```

В системе Linux размеры логического и физического секторов диска можно узнать через псевдофайловую систему `/sys`. Например, физический и логический размеры секторов подключённого диска `/dev/sda` определяются следующим образом:

```
# dmesg
...
[ 16.606585] sd 1:0:0:0: [sda] 7814037168 512-byte logical blocks: (4.00 TB/3.64
TiB)
...
# cat /sys/block/sda/queue/logical_block_size
512
# cat /sys/block/sda/queue/physical_block_size
4096
# blockdev --getpbsz /dev/sda
4096
# blockdev --getss /dev/sda
512
```

Здесь показаны два метода: чтение псевдофайловой системы `/sys`, доступное также непривилегированному пользователю, и применение команды `blockdev`.

Некоторые SSD позволяют выбирать размер физического сектора с помощью средства для работы с микропрограммой. Например, в некоторых новых SSD Intel размер сектора можно переключать между 512 и 4096 с помощью предоставленного Intel инструмента командной строки (https://downloadcenter.intel.com/download/23931/).

Несколько особенностей дисков с секторами по 4 Кбайт представляют интерес для сообщества цифровой криминалистики и рассматриваются далее в этом разделе. У некоторых ранних дисков Western Digital Advanced Format 512e имелась перемычка - контакты 7 и 8, - которая внутренне сдвигала секторы, совмещая начало создаваемых по умолчанию разделов Windows XP с началом сектора размером 4 Кбайт. Такая настройка перемычки для повторного выравнивания диска значительно повышала производительность. Изменение положения перемычек выравнивания секторов повлияет на хеш криминалистического образа и потенциально на анализ диска. При криминалистическом создании или проверке образа критически важно сохранять то положение перемычек, в котором накопитель был первоначально изъят.

Применение дисков 4Kn повлияет на значение резервного пространства. Резервное пространство оперативной памяти, или memory slack, - неиспользуемая часть последнего сектора файла; не следует путать его с резервным пространством файла, то есть неиспользуемой частью последнего блока файловой системы, принадлежащего файлу. При использовании дисков 4Kn с файловыми системами, в которых размер блока равен 4 Кбайт, резервное пространство оперативной памяти совпадает с резервным пространством файла. Операционные системы, которые перед записью заполняют неиспользуемую часть 4-килобайтного сектора нулями, устраняют возможность

обнаружить полезные данные в резервном пространстве файлов в файловых системах с 4-килобайтными блоками.

Криминалистические программы, предполагающие размер сектора 512 байт, могут завершиться с ошибкой или, что ещё хуже, выдать неправильный результат. При работе с дисками 4Kn важно убедиться, что криминалистическая программа распознаёт и использует секторы 4Kn. Sleuth Kit по умолчанию выбирает 512-байтовые секторы, поэтому для дисков 4Kn ему необходимо явно указать размер 4 Кбайт. В следующем примере `mmls` сначала выдаёт неправильный результат с настройкой по умолчанию, а затем правильный результат после указания верного размера сектора.

```
# mmls /dev/sde
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

     Slot      Start        End          Length       Description
00:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----     0000000000   0000000255   0000000256   Unallocated
02:  00:00     0000000256   0732566645   0732566390   Linux (0x83)
03:  -----     0732566646   5860533167   5127966522   Unallocated
...
# mmls -b 4096 /dev/sde
DOS Partition Table
Offset Sector: 0
Units are in 4096-byte sectors

     Slot      Start        End          Length       Description
00:  Meta      0000000000   0000000000   0000000001   Primary Table (#0)
01:  -----     0000000000   0000000255   0000000256   Unallocated
02:  00:00     0000000256   0732566645   0732566390   Linux (0x83)
#
```

После указания 4096-байтового сектора параметром `-b` секторы раздела Linux представлены единицами по 4 Кбайт, а нераспределённая область в конце накопителя отсутствует. Пример успешного получения данных с диска с собственными секторами по 4 Кбайт приведён в разделе «Инструменты dcfldd и dc3dd» на странице 144.

Диски Advanced Format 4Kn пока применяются редко. Неясно, как секторы 4Kn повлияют на существующие криминалистические программы получения и анализа данных, представленные сегодня на рынке, особенно если такие инструменты изначально предполагают размер сектора 512 байт. Эта область требует дальнейших исследований со стороны сообщества цифровой криминалистики.

Пространства имён NVME

Спецификация NVME вводит понятие пространств имён, позволяющих разделять накопитель на более низком уровне, скрытом абстракцией от обычной операционной системы. Криминалистический образ накопителя с несколькими пространствами имён необходимо создавать отдельно для каждого пространства. Определить их число можно несколькими способами.

Отправив административную команду идентификации контроллера с помощью инструмента `nvme-cli`, можно проверить число поддерживаемых и используемых пространств имён. В следующем примере показаны различные сведения об их поддержке:

```
# nvme id-ctrl /dev/nvme1 -H
NVME Identify Controller:
vid : 0x144d
ssvid : 0x144d
sn : S2GLNCAGA04891H
mn : Samsung SSD 950 PRO 256GB
fr : 1B0QBXX7
...
oacs : 0x7
[3:3] : 0 NS Management and Attachment Not Supported
...
[0:0] : 0x1 SMART/Health Log Page per NS Supported
...
nn : 1
...
```

Здесь поле поддержки необязательных административных команд Optional Admin Command Support (OACS) показывает, что этот накопитель не поддерживает управление пространствами имён. Поле Number of Namespaces (`nn`) сообщает число пространств имён контроллера - на данном устройстве оно одно.

Размер пространства имён также можно проверить с помощью `nvme-cli` и сравнить со спецификациями изготовителя:

```
# nvme id-ns /dev/nvme0n1
NVME Identify Namespace 1:
nsze : 0x2e9390b0
ncap : 0x2e9390b0
nuse : 0x2e9390b0
...
```

Здесь `nsze` обозначает размер пространства имён, `ncap` - его ёмкость, а `nuse` - используемый объём. Если эти значения совпадают с документированным изготовителем размером накопителя, они подтверждают, что используется одно пространство имён.

Третья проверка наличия нескольких устройств пространств имён состоит в простом просмотре устройств `/dev/nvme0n2*`, `/dev/nvme0n3*` и других, обнаруженных операционной системой.

На момент написания книги потребительских накопителей с поддержкой нескольких пространств имён, доступных для тестирования, не существовало. Сведения в этом разделе получены из спецификации NVME и документации инструмента.

Твердотельные гибридные диски

Гибрид твердотельного и традиционного магнитного диска был разработан для того, чтобы по доступной цене предоставить большую ёмкость и производительность, сравнимую с SSD.

Такие гибридные накопители, известные как твердотельные гибридные диски (Solid State Hybrid Disks, SSHD), дополнительно кешируют часто используемые секторы в твердотельной памяти. SSHD могут работать полностью независимо от операционной системы или принимать от неё «подсказки», помогающие решить, какие блоки кешировать.

Начиная с SATA 3.2 была добавлена функция передачи сведений гибридному накопителю, позволяющая основной системе сообщать ему информацию для кеширования с помощью команд ATA.

До настоящего времени криминалистике SSHD посвящено мало исследований, поэтому последствия для получения данных неясны. SSHD содержат небольшой SSD, который должен выполнять выравнивание износа. Гибридный накопитель без поддержки команд TRIM, вероятно, не будет стирать данные в нераспределённых блоках файловой системы.

Описанные здесь гибридные системы встроены в электронику одного накопителя. Гибридная система может состоять и из двух отдельных устройств: небольшого SSD и более крупного магнитного жёсткого диска. Эквивалентное гибридное кеширование достигается посредством драйверов операционной системы или проприетарных систем, таких как Intel Smart Response Technology.

Заключительные мысли

В этой главе я рассмотрел различные виды носителей - магнитные, энергонезависимые и оптические - и исследовал разные типы накопителей. Я описал внутренние и внешние интерфейсы подключения устройств хранения к основной системе эксперта. Кроме того, я объяснил протоколы доступа к накопителям и рассмотрел несколько менее распространённых специальных тем. Материал главы изложен с точки зрения эксперта-криминалиста. Теперь у вас должна быть прочная основа для понимания следующей главы, посвящённой применению Linux как платформы криминалистического получения данных.

Сноски

[1] [назад] Jeff Hedlesky, “Advancements in SSD Forensics” (presentation, CEIC2014, Las Vegas, NV, May 19-22, 2014).
[2] [назад] Bruce Nikkel, *Digital Investigation* 16 (2016): 38-45, doi:10.1016/j.diin.2016.01.001.
[3] [назад] Многоуровневая организация сетей LAN/WAN описывается семью уровнями абстракции сетевого обмена в модели взаимодействия открытых систем (Open Systems Interconnection, OSI).
[4] [назад] *AT Attachment Interface for Disk Drives*, ANSI X3.221-199x, Revision 4c, X3T10, 1994.
[5] [назад] *ATA/ATAPI Command Set-2 (ACS-2)*, Revision 2.
[6] [назад] Mayank R. Gupta, Michael D. Hoeschele, and Marcus K. Rogers, “Hidden Disk Areas: HPA and DCO,” *International Journal of Digital Evidence* 5, no. 1 (2006), https://www.utica.edu/academic/institutes/ecii/publications/articles/EFE36584-D13F-2962-67BEB146864A2671.pdf.

Глава 2. Linux как платформа криминалистического получения данных

В этой главе Linux описывается как платформа для криминалистического получения цифровых данных и обсуждаются её преимущества и недостатки. Я также рассматриваю признание Linux и программного обеспечения с открытым исходным кодом в сообществе цифровой криминалистики, а в заключительном разделе даю обзор соответствующих основ Linux, которые понадобятся для понимания последующих разделов книги.

В примерах книги преимущественно используется Ubuntu Linux Server 16.04 LTS, поддерживаемая до апреля 2021 года, с оболочкой Bourne Again Shell (Bash) версии 4.3.x. Примеры должны работать и в других дистрибутивах Linux и операционных системах, например OS X или Windows, если применять те же либо более новые версии инструментов и соответствующим образом изменить имена устройств. На протяжении книги выражения «командная строка», «оболочка» и Bash употребляются как взаимозаменяемые.

Linux и открытое программное обеспечение в криминалистическом контексте

Растущая популярность программного обеспечения с открытым исходным кодом (open source software, OSS), такого как Linux, сделала его важной платформой цифровой криминалистики. Многие исследователи обсуждали преимущества OSS для выполнения критериев Дауберта, касающихся надёжности доказательств.[1] Брайан Карриер, автор Sleuth Kit, исследовал юридические доводы в пользу открытых криминалистических инструментов и предположил, что некоторые части криминалистических программ, хотя и необязательно все, следует сделать открытыми.[2]

Главное преимущество OSS в криминалистическом контексте - прозрачность. В отличие от проприетарных коммерческих программ, исходный код можно анализировать и открыто проверять. Кроме того, академические исследователи могут изучать его и развивать результаты работы других участников сообщества. Открытые криминалистические приложения стали инструментами и строительными блоками научных исследований в криминалистике. У OSS есть и недостатки, а в некоторых ситуациях его применение лишено смысла. В частности, открытость сообщества открытого исходного кода порой может противоречить конфиденциальному характеру продолжающихся криминалистических расследований. Преимущества и недостатки Linux и OSS обсуждаются в следующих разделах.

Преимущества Linux и OSS в криминалистических лабораториях

Общедоступность OSS означает, что оно доступно каждому, а не только тем, кто приобрёл лицензии или подписал соглашения о неразглашении. Любой заинтересованный человек может свободно загружать, применять, исследовать и изменять OSS, не уплачивая лицензионных сборов или платы за использование.

Доступ к исходному коду позволяет адаптировать программу и облегчает её интеграцию с другим программным и аппаратным обеспечением и процессами криминалистической лаборатории. Доступ на уровне исходного кода расширяет возможности автоматизации рабочих нагрузок и создания сценариев. Автоматизация уменьшает необходимое участие человека, тем самым ограничивая риск человеческой ошибки и высвобождая специалистов для другой работы.

В лабораториях с большим объёмом дел автоматизация необходима для оптимизации и упрощения процессов. Поскольку исходный код можно свободно изменять, OSS удаётся приспособить к требованиям конкретной криминалистической лаборатории. В особенности программы командной строки позволяют связывать множество задач и заданий в конвейеры посредством сценариев оболочки, выстраивая сквозной процесс.

Поддержка OSS обладает несколькими преимуществами. Неформальная помощь сообщества может быть превосходной, а ответы на просьбы о помощи в списках рассылки и чатах порой приходят за считанные минуты. В некоторых случаях быстро реализуются исправления, устраняются ошибки и выполняются запросы новых функций.

Linux и OSS идеально подходят для академической криминалистической лаборатории, поскольку используют открытые опубликованные стандарты, а не закрытые или проприетарные. Сообщества разработчиков OSS сотрудничают с конкурирующими группами, а не противодействуют им. Обучение у других, заимствование кода и идей у других с должным указанием авторства и развитие чужих результатов поощряются и служат основой обучения и накопления знаний.

Независимость от поставщика, которую даёт OSS, предотвращает привязку к продуктам одного изготовителя и способствует взаимодействию и совместимости технологий и организаций. Благодаря этому со временем проще менять программное обеспечение: отдельные компоненты можно заменять новыми или альтернативными технологиями, не затрагивая системы и процессы в целом.

Недостатки Linux и OSS в криминалистических лабораториях

Недостатки Linux и OSS одновременно служат доводами в пользу закрытых проприетарных программ. Коммерческие реализации инструментов часто обладают в этой области своими выгодами и преимуществами.

Модель поддержки сообществом открытого исходного кода не гарантирует надёжность, точность или достоверность. Качество предоставляемых сообществом ответов сильно различается: одни бывают превосходными, другие - ошибочными и даже опасными. Зачастую официальной службы поддержки вообще нет. В ситуациях, где требуется гарантированная круглосуточная поддержка, преимущество имеют коммерческие поставщики.

Поддержка в мире открытого исходного кода столь же прозрачна, как сами программы: её видят все. Однако в криминалистической лаборатории материалы дел и расследования могут быть деликатными или конфиденциальными. Обращение за помощью к общественности способно раскрыть либо скомпрометировать подробности продолжающегося расследования. Поэтому в открытой модели поддержки возникают вопросы информационной безопасности и неприкосновенности частной жизни.

Взаимодействие с проприетарными технологиями создаёт сложности для открытых интерфейсов и API. Закрытые технологии, сведения о которых не публикуются, часто не лицензируют, а исследуют методами обратной разработки. Такие попытки нередко остаются неполными, рискуют неправильно реализовать конкретную технологию и могут потребовать долгого времени.

Свободное OSS часто разрабатывается добровольцами, а программа может находиться в бесконечном состоянии разработки. Некоторые проекты забрасывают или они погибают из-за отсутствия внимания. В других возникают ответвления кода: часть разработчиков решает скопировать существующую кодовую базу и развивать её в направлении, отличном от выбранного первоначальными авторами.

Свободное OSS порой недостаточно отшлифовано. В нём могут встречаться ошибки; его бывает трудно освоить или использовать. Документация может быть плохой, а иногда единственной документацией служит исходный код. В отличие от коммерческих программ, обучение работе с продуктом обычно не предоставляется. Освоение Unix/Linux требует времени и усилий; в частности, командная строка не столь интуитивна, как полностью графическая среда. Многие, впервые попадая в мир свободного и открытого программного обеспечения, проходят период обучения, причём осваивать приходится не только программу, но и общие установки и образ мышления окружающего сообщества.

Поставщики коммерческих программ в криминалистическом сообществе обеспечивают определённую защищённость результатов и дают гарантии правильной работы своих программ. Некоторые криминалистические компании даже предлагали выступить в суде, чтобы защитить результаты, полученные их продуктами. В сообществе свободного и открытого исходного кода никто не отвечает за созданные программы и не принимает за них ответственность. Они предоставляются «как есть» и «на страх и риск пользователя».

Очевидно, что OSS подходит не для каждой ситуации, и книга не утверждает обратного. Во многих представленных далее примерах OSS полезнее для обучения и демонстрации внутренних механизмов, чем в качестве жизнеспособной альтернативы профессиональным коммерческим криминалистическим программам.

Ядро Linux и устройства хранения

Традиционные системы Unix, философию которых наследует Linux, были спроектированы так, что всё в них является файлом. Каждый файл относится к определённому типу: обычному файлу, каталогу, блочному устройству, символьному устройству, именованному каналу, жёсткой ссылке или программной, то есть символической, ссылке, подобной файлам LNK в Windows. На рабочей станции эксперта интерес для криминалиста представляют файлы блочных устройств подключённых исследуемых дисков, потенциально содержащих доказательства. В этом разделе описываются устройства Linux, прежде всего блочные устройства носителей.

Обнаружение устройств ядром

В системах Unix и Linux есть специальный каталог `/dev`, где хранятся особые файлы, соответствующие известным ядру устройствам. В ранних системах Unix и Linux файлы устройств в `/dev` требовалось создавать вручную командой `mknod` либо посредством сценариев `MAKEDEV`, создававших устройства при загрузке или по мере необходимости. С появлением оборудования plug-and-play понадобился более динамический подход, и для автоматического обнаружения нового оборудования и создания файлов устройств была разработана `devfs`. Необходимость лучше взаимодействовать со сценариями и программами пользовательского пространства привела к созданию `udev`, которая заменила `devfs`. Сегодня `udev` объединена с `systemd` и запускает демон `systemd-udevd`.

При подключении нового устройства к основной системе или его отключении прерывание уведомляет ядро об изменении оборудования. Ядро сообщает об этом системе `udev`, которая создаёт соответствующие устройства с надлежащими разрешениями, выполняет сценарии и программы настройки или удаления и отправляет сообщения другим демонам, например через `dbus`.

Наблюдать за работой `udev` можно с помощью инструмента `udevadm` в режиме мониторинга:

```
# udevadm monitor
monitor will print the received events for:
UDEV - the event that udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[7661.685727] add /devices/pci0000:00/0000:00:14.0/usb1/1-14 (usb)
KERNEL[7661.686030] add /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0
(usb)
KERNEL[7661.686236] add /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/
host9 (scsi)
KERNEL[7661.686286] add /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/
host9/scsi_host/host9 (scsi_host)
...
KERNEL[7671.797640] add /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/
host9/target9:0:0/9:0:0:0/block/sdf (block)
KERNEL[7671.797721] add /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/
host9/target9:0:0/9:0:0:0/block/sdf/sdf1 (block)
...
```

Здесь диск был подключён к порту USB, а udev выполнила настройку всех соответствующих файлов устройств и ссылок.

Команда udevadm позволяет также определить список связанных файлов и путей подключённых устройств. Например:

```
# udevadm info /dev/sdf
P: /devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/host9/target9:0:0/9:0:0:0/
block/sdf
N: sdf
S: disk/by-id/ata-ST2000DL003-9VT166_5YD83QVW
S: disk/by-id/wwn-0x5000c50048d79a82
S: disk/by-path/pci-0000:00:14.0-usb-0:14:1.0-scsi-0:0:0:0
E: DEVLINKS=/dev/disk/by-path/pci-0000:00:14.0-usb-0:14:1.0-scsi-0:0:0:0 /dev/disk/
by-id/wwn-0x5000c50048d79a82 /dev/disk/by-id/ata-ST2000DL003-9VT166_5YD83QVW
E: DEVNAME=/dev/sdf
E: DEVPATH=/devices/pci0000:00/0000:00:14.0/usb1/1-14/1-14:1.0/host9/target9:0:0/
9:0:0:0/block/sdf
E: DEVTYPE=disk
E: ID_ATA=1
...
```

Понимание дерева устройств Linux важно при криминалистическом получении и анализе данных. При выполнении криминалистических команд и сборе сведений об устройстве критически важно знать, какие устройства относятся к локальной машине следователя, какие являются подозреваемыми накопителями, какое устройство служит блокиратором записи и так далее.

Устройства хранения в /dev

После обнаружения ядром подключённые накопители появляются в каталоге /dev как блочные устройства. Файлы необработанных дисковых устройств именуются по определённым правилам: sd* для SCSI и SATA, hd* для IDE, md* для массивов RAID, nvme*n* для накопителей NVME и другие имена для менее распространённых или проприетарных драйверов дисковых устройств.

Отдельные разделы, обнаруженные ядром, представлены нумерованными необработанными устройствами, например hda1, hda2, sda1, sda2 и так далее. Блочные устройства разделов представляют целые разделы как непрерывную последовательность секторов диска. Обычно раздел содержит файловую систему, которую ядро может смонтировать и предоставить пользователям как обычную часть дерева каталогов. Большинство криминалистических инструментов способно и должно исследовать необработанные устройства и устройства разделов без монтирования файловой системы.

Другие специальные устройства

Для примеров этой книги полезно знать ещё несколько устройств. «Битовая корзина» /dev/null отбрасывает все записываемые в неё данные. При обращении к /dev/zero предоставляется непрерывный поток нулей. Генератор случайных чисел /dev/random выдаёт поток случайных данных. Имена ленточных накопителей обычно начинаются с /dev/st, а к другим внешним носителям можно обращаться через /dev/cdrom или /dev/dvd; зачастую это символические ссылки на /dev/sr*. В некоторых случаях доступ к устройствам осуществляется через интерфейс универсального драйвера SCSI /dev/sg*.

К другим специальным псевдоустройствам относятся /dev/loop* и /dev/mapper/*. Они подробнее рассматриваются на протяжении книги.

Ядро Linux и файловые системы

Файловые системы организуют хранилище в иерархическую структуру каталогов, или папок, и файлов. Они создают уровень абстракции над блочными устройствами.

Поддержка файловых систем ядром

Ядро Linux поддерживает множество файловых систем; их список приведён по адресу https://en.wikipedia.org/wiki/Category:Linux_kernel-supported_file_systems. Это может быть полезно при выполнении некоторых криминалистических задач. Однако для криминалистического получения данных поддержка файловой системы не требуется, поскольку создание образа работает с блочным устройством ниже файловой системы и схемы разделов.

Чтобы предоставить единообразный интерфейс для файловых систем разных типов, ядро Linux реализует уровень абстракции виртуальной файловой системы (Virtual File System, VFS). Он позволяет монтировать обычные файловые системы носителей - EXT*, NTFS, FAT и другие, - сетевые файловые системы - `nfs`, `sambafs`/`smbfs` и другие, - файловые системы пользовательского пространства на основе FUSE,[3] составные файловые системы - `encryptfs`, `unionfs` и другие, - а также специальные псевдофайловые системы, например `sysfs` и `proc`.

Диаграмма стека хранения Linux, показанная на рис. 2-1, помогает понять взаимосвязь файловых систем, устройств, драйверов и аппаратных устройств в ядре Linux.

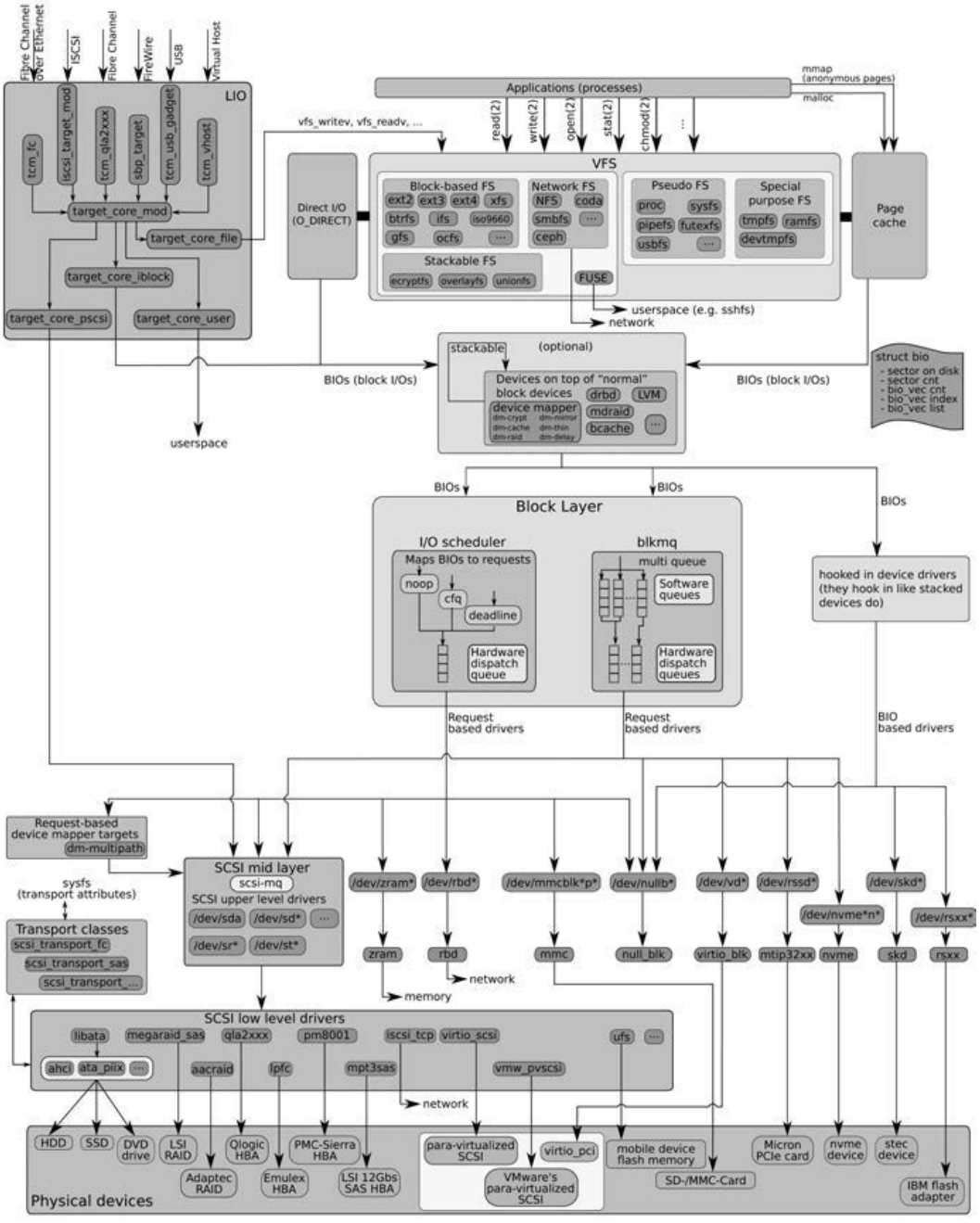

Рис. 2-1. Диаграмма стека хранения Linux (https://www.thomas-krenn.com/en/wiki/Linux_Storage_Stack_Diagram, используется на условиях лицензии CC Attribution-ShareAlike 3.0 Unported)

Перевод служебных подписей на схеме.

Исходная подпись	Перевод
Applications (processes)	Приложения (процессы)
anonymous pages	анонимные страницы памяти
Direct I/O (O_DIRECT)	прямой ввод-вывод (`O_DIRECT`)
Block-based FS	блочные файловые системы
Network FS	сетевые файловые системы
Pseudo FS	псевдофайловые системы
Special purpose FS	файловые системы специального назначения
Stackable FS	составные файловые системы
Page cache	кеш страниц
userspace	пользовательское пространство
network	сеть
BIOs (block I/Os)	BIO, операции блочного ввода-вывода
stackable (optional)	составной уровень (необязательно)
Devices on top of “normal” block devices	устройства поверх «обычных» блочных устройств
device mapper	диспетчер устройств
Block Layer	блочный уровень
I/O scheduler	планировщик ввода-вывода
Maps BIOs to requests	преобразует BIO в запросы
multi queue	несколько очередей
Software queues	программные очереди
Hardware dispatch queues	аппаратные очереди диспетчеризации
hooked in device drivers	встроено в драйверы устройств
Request based drivers	драйверы на основе запросов
BIO based drivers	драйверы на основе BIO
Request-based device mapper targets	цели диспетчера устройств на основе запросов
SCSI mid layer	средний уровень SCSI
Transport classes	классы транспорта
SCSI low level drivers	низкоуровневые драйверы SCSI
Physical devices	физические устройства

Монтирование файловых систем в Linux

Часто неправильно понимают различие между подключённым и смонтированным дисковым устройством. Для получения данных с устройства и даже для обращения к нему средствами криминалистического анализа монтирование не требуется. Криминалистические инструменты, работающие непосредственно с блочными устройствами, получают доступ к подключённым дискам без их монтирования операционной системой.

Чтобы файловая система на дисковом устройстве стала доступна в Unix и Linux как обычная структура каталогов, её необходимо явно смонтировать. Монтирование файловой системы означает лишь предоставление доступа к ней стандартным средствам работы с файлами - файловым диспетчерам, приложениям и другим программам, - подобно буквам дисков в DOS/Windows. Linux не использует буквы дисков: смонтированные диски становятся частью локальной файловой системы и присоединяются к любому выбранному месту её дерева. Это место называется точкой монтирования файловой системы. Например, следующая команда монтирует USB-накопитель в системе следователя, используя `/mnt` как точку монтирования:

```
# mount /dev/sdb1 /mnt
```

Перед физическим отключением смонтированного диска в Linux сначала размонтируйте файловую систему, чтобы предотвратить её повреждение. Команде `umount` - именно `umount`, а не `unmount` - можно передать имя устройства или точку монтирования. Две следующие команды выполняют одно и то же действие по размонтированию дисковой файловой системы:

```
# umount /dev/sdb1
# umount /mnt
```

После размонтирования файловой системы необработанный диск по-прежнему виден ядру и доступен инструментам работы с блочными устройствами, хотя сама файловая система не смонтирована. Размонтированный диск можно безопасно физически отключить от системы получения данных следователя.

Не подключайте и не монтируйте подозреваемые накопители без блокиратора записи. При этом высок риск изменить, повредить или уничтожить цифровые доказательства. Современные операционные системы обновляют временные метки последнего доступа при обращении к файлам и каталогам. Любые демоны пользовательского пространства - средства индексирования поиска, генераторы миниатюр и другие - могут записать данные на диск и перезаписать доказательства; файловые системы могут попытаться выполнить восстановление; журнальные файловые системы способны записать данные журнала; возможны и другие ошибки человека. Файловую систему можно смонтировать с блокиратором записи: она будет доступна как обычная файловая система, но только для чтения, что обеспечит защиту цифровых доказательств.

Доступ к файловым системам посредством криминалистических инструментов

При работе с такими криминалистическими средствами, как Sleuth Kit, `dcfldd`, `foremost` и другими, к файловой системе можно обратиться без монтирования, используя правильное блочное устройство, представляющее содержащий её раздел. В большинстве случаев это будет нумерованное устройство, например `/dev/sda1`, `/dev/sda2`, `/dev/sdb1` и так далее, обнаруженное ядром Linux.

Если ядро Linux не обнаруживает файловую систему, может потребоваться явно указать её. Файловая система не будет правильно обнаружена по любой из следующих причин:

• файловая система не поддерживается основной системой - отсутствует модуль ядра или сама файловая система не поддерживается;
• таблица разделов повреждена или отсутствует;
• раздел удалён;
• смещение файловой системы на диске неизвестно;
• файловую систему необходимо сделать доступной, например разблокировать устройство или расшифровать раздел.

В последующих разделах книги я объясню методы применения loop-устройств для доступа к разделам и файловым системам, которые ядро Linux или различные криминалистические инструменты не обнаруживают автоматически.

Дистрибутивы и оболочки Linux

При создании рабочей станции следователя для криминалистического получения или анализа цифровых данных полезно понимать основные принципы построения и состав системы Linux.

Дистрибутивы Linux

Строго говоря, термин Linux обозначает только ядро, которое и является собственно операционной системой.[4] Графический интерфейс, инструменты и утилиты и даже оболочка командной строки относятся не к Linux, а к дистрибутиву Linux. Дистрибутив представляет собой функциональный пакет, который обычно содержит ядро Linux, программы установки и диспетчеры пакетов, как правило уникальные для дистрибутива, а также различные дополнительные программы и утилиты, включая стандартные приложения: офисные пакеты, веб-браузеры, почтовые клиенты и средства чата. Официальное ядро Linux существует только одно, но имеется множество дистрибутивов Linux, в том числе Red Hat, SUSE, Arch и Debian. Есть также немало производных дистрибутивов. Например, Ubuntu основан на Debian, CentOS - на Red Hat, а Manjaro - на Arch. Полный перечень дистрибутивов и других открытых операционных систем, не относящихся к Linux, см. на сайте http://distrowatch.com/.

Графический интерфейс разных дистрибутивов Linux состоит из нескольких компонентов, назначение которых полезно понимать. Оконная система X11 представляет собой сервер отображения, который взаимодействует с графическим оборудованием и предоставляет интерфейс к графическим примитивам X11; более новой альтернативой X11 служит Wayland. Оконный диспетчер управляет перемещением, изменением размера, размещением окон и другими операциями с ними. В качестве примеров можно назвать Compiz, Mutter и OpenBox; ими можно пользоваться без среды рабочего стола. Среда рабочего стола определяет внешний вид и поведение дистрибутива и работает поверх оконного диспетчера. К популярным рабочим столам относятся Gnome, KDE, Xfce и Mate. Графическую среду для рабочей станции эксперта-криминалиста можно выбирать по личным предпочтениям: она никак не влияет на собираемые или анализируемые доказательства. Примеры книги выполнялись в системе без графического интерфейса - Ubuntu Server.

Оболочка

Оболочка представляет собой командную строку, через которую люди или машины подают команды для управления операционной системой. Она запускает и останавливает программы, устанавливает программное обеспечение, выключает систему и выполняет другую работу. Можно утверждать, что командная оболочка предлагает более мощные функции и возможности, чем графические среды.

В примерах книги используется среда командной строки. Для инструментов командной строки могут существовать графические аналоги или оболочки, но в этой книге они не рассматриваются.

Самая распространённая сегодня оболочка, используемая по умолчанию в большинстве дистрибутивов Linux, - Bash. Примеры книги используют Bash, но могут работать и в других оболочках, например `zsh` или `csh`.

Выполнение команд

Оболочка - всего лишь ещё одна программа, выполняемая в системе. Люди взаимодействуют с ней, вводя команды, а машины - выполняя сценарии оболочки.

Пользователь обычно вводит команду в приглашении, а затем нажимает клавишу **ENTER** или **RETURN**. В зависимости от запущенной программы и настройки оболочки вывод может появиться или отсутствовать.

Конвейеры и перенаправление

Полезная возможность командной строки Unix/Linux - передача потоков данных программам и файлам посредством конвейеров и перенаправления. Это отчасти похоже на перетаскивание и копирование со вставкой в графических средах, но обладает гораздо большей гибкостью.

Программа может получать данные из вывода других программ или из файлов в файловой системе. Она также может передавать вывод на вход другой программе либо отправлять его в файл файловой системы.

Следующие примеры показывают, как `tool.sh` перенаправляет вывод в `file.txt`, получает входные данные из `file.txt` и передаёт свой вывод по конвейеру на вход `othertool.sh`:

```
$ tool.sh > file.txt
$ tool.sh < file.txt
$ tool.sh | othertool.sh
```

Механизм конвейеров и перенаправления не ограничен отдельными командами или файлами: их можно объединять в последовательность из нескольких программ:

```
$ tool.sh < file.txt | othertool.sh | lasttool.sh > lastfile.txt
```

Конвейеры и перенаправление широко применяются на протяжении книги. Они позволяют выполнять несколько задач одной строкой команд и облегчают создание сценариев и автоматизацию, устраняя необходимость участия человека. В примерах книги конвейеры и перенаправление используются для получения образов носителей, передачи данных между криминалистическими программами и сохранения представляющих интерес доказательных сведений в файлах.

Заключительные мысли

В этой главе я рассмотрел применение Linux как жизнеспособной платформы для криминалистического получения данных, включая её преимущества и недостатки. Я сделал обзор дистрибутивов Linux и принципов работы ядра. С точки зрения эксперта-криминалиста были показаны понятия устройств и файловых систем, а также применение оболочек, конвейеров и перенаправления. Теперь вы обладаете знаниями Linux, необходимыми для понимания примеров в остальной части книги.

Сноски

[1] [назад] Erin Kenneally, “Gatekeeping Out of the Box: Open Source Software as a Mechanism to Assess Reliability for Digital Evidence,” *Virginia Journal of Law and Technology* 6, no. 13 (2001).
[2] [назад] Brian Carrier, “Open Source Digital Forensic Tools: The Legal Argument” [technical report] (Atstake Inc., October 2002).
[3] [назад] FUSE - реализация файловой системы в пользовательском пространстве (см. https://en.wikipedia.org/wiki/Filesystem_in_Userspace).
[4] [назад] Вопрос о включении GNU в название Linux вызывает разногласия; см. https://en.wikipedia.org/wiki/GNU/Linux_naming_controversy.

Глава 3. Форматы криминалистических образов

В этой главе представлен обзор различных инструментов получения данных, контейнеров доказательств и широко применяемых сегодня форматов криминалистических образов. Форматы криминалистических образов и контейнеры доказательств - это структуры, в которых криминалистически полученный образ хранится вместе с дополнительными сведениями по делу: временем и продолжительностью получения, способом создания образа, его размером, ошибками, хешами и так далее. К дополнительным возможностям криминалистических форматов обычно относятся сжатие файлов и шифрование. В этой главе показано выполнение криминалистических задач в командной строке с применением нескольких форматов.

Содержательную вводную статью о различных криминалистических форматах можно найти на сайте Digital Forensic Research Workshop (DFRWS) по адресу http://www.dfrws.org/CDESF/survey-dfrws-cdesf-diskimg-01.pdf.

Распространённые криминалистические форматы, рассматриваемые в этой главе, можно определить командой Sleuth Kit `img_stat`:

```
# img_stat -i list
Supported image format types:
raw (Single or split raw file (dd))
aff (Advanced Forensic Format)
afd (AFF Multiple File)
afm (AFF with external metadata)
afflib (All AFFLIB image formats (including beta ones))
ewf (Expert Witness format (encase))
```

Помимо этих форматов, в главе представлен специальный метод, использующий SquashFS как практичный криминалистический контейнер, пригодный для работы со стандартными криминалистическими инструментами.

> **Примечание.** В отношении криминалистических образов важно понимать, что они копируют не файлы, а секторы диска - от сектора 0 до последнего доступного сектора. Размер необработанного образа всегда равен полному размеру диска независимо от числа файлов в его файловой системе.

Необработанные образы

Необработанные образы не являются форматом как таковым: это фрагмент необработанных данных, образ которых получен из источника доказательств. В них нет дополнительных метаданных, кроме сведений о самом файле образа - имени, размера, временных меток и другой информации в собственном индексном дескрипторе образа.

Извлечение необработанного образа технически несложно: это простая передача последовательности байтов с исходного устройства в целевой файл. Обычно она выполняется без каких-либо преобразований или трансляции.

Для извлечения необработанных образов чаще всего применяют инструменты копирования дисковых блоков, например `dd` и её варианты. Они рассматриваются в следующих разделах.

Традиционная dd

Самый простой и одновременно старейший инструмент создания необработанных образов - исходная утилита Unix `dd`. Она не была предназначена для сбора доказательств, однако простая побайтовая передача удобна для создания образов дисковых устройств, поскольку формирует полную низкоуровневую копию отдельных секторов диска, сохраняя структуру файловой системы, файлы, каталоги и метаданные. В то же время такие возможности, как ведение журнала, обработка ошибок и хеширование, в ней недостаточны или вообще отсутствуют; `dd` можно применять, когда лучшей альтернативы нет. Проект Computer Forensic Tool Testing (CFTT) протестировал несколько стандартных версий `dd`. Результаты опубликованы на сайте CFTT: http://www.cftt.nist.gov/disk_imaging.htm.

Утилита `dd` была создана в 1970-е годы в ранних системах UNIX для преобразования порядка байтов и копирования блоков. Изначально она предназначалась для преобразования данных в кодировке EBCDIC, применявшейся на мейнфреймах, в кодировку ASCII, предпочтительную в среде UNIX. Программа просто получает блоки данных из источника, при необходимости выполняет преобразование или трансляцию, а затем помещает блоки в заданное место назначения - на другое устройство или в файл. Современные версии `dd` получили усовершенствования, благодаря которым подходят для криминалистического получения данных с таких устройств, как диски и ленты.

Криминалистические варианты dd

Поскольку исходный инструмент dd не проектировался для криминалистического применения, в нём отсутствуют некоторые необходимые возможности. Поэтому позднее были разработаны основанные на dd средства со следующими желательными криминалистическими функциями:

• криптографическое хеширование;
• улучшенная обработка ошибок;
• ведение журнала;
• повышение производительности;
• контрольная проверка;
• наблюдение за ходом работы, поскольку создание криминалистического образа может занимать много часов.

Два наиболее распространённых варианта утилиты dd - dcfldd, созданная Николасом Харбором в Лаборатории компьютерной криминалистики Министерства обороны США (Defense Computer Forensics Lab, DCFL) в 2002 году, и dc3dd, созданная Джесси Корнблюмом в 2007 году во время его работы в Центре киберпреступности Министерства обороны США (Defense Cyber Crime Center, DC3).

Инструмент dcfldd основан на GNU dd и, среди прочего, получил дополнительные функции хеширования, улучшенного ведения журнала и разделения выходных файлов. Хотя он не обновлялся с 2006 года, им пользуются и сегодня. Александр Дюлануа создал исправленную версию dcfldd, включавшую некоторые исправления ошибок Debian; она доступна по адресу https://github.com/adulau/.

Более новый инструмент dc3dd реализован в виде исправления и поэтому легче следует за изменениями кода GNU dd. Он продолжает сопровождаться и недавно получал обновления. В нём имеются криминалистические функции, похожие на возможности dcfldd, а также улучшенное ведение журнала и обработка ошибок.

Оба инструмента, dcfldd и dc3dd, происходят от традиционной dd и обладают сходными возможностями. Ни один не имеет встроенной поддержки записи в криминалистические форматы FTK, EnCase или AFF, сжатия либо шифрования образа, однако эти задачи можно решить посредством конвейеров команд и перенаправления. Примеры обоих инструментов встречаются на протяжении книги. Для dcfldd и dc3dd существуют отчёты CFTT о тестировании.

Инструменты восстановления данных

Стоит упомянуть несколько инструментов восстановления данных благодаря их надёжной обработке ошибок и настойчивым методам восстановления. Эти средства создавались без расчёта на криминалистику, но могут пригодиться в ситуациях, когда все остальные криминалистические инструменты не смогли получить данные с сильно повреждённого носителя.

GNU ddrescue и dd_rescue имеют похожие названия, но являются разными, независимо разработанными инструментами. На момент написания книги оба активно развивались и обладали разными полезными возможностями. Хотя в их названиях упоминается dd, ни один не использует синтаксис команд dd.

GNU ddrescue была создана в 2004 году Антонио Диасом Диасом и включена в Debian под именем пакета gddrescue. Она применяет настойчивые и агрессивные методы, пытаясь восстановить повреждённые области диска.

Инструмент `dd_rescue` был создан Куртом Гарлоффом в 1999 году и обладает развитой системой модулей, поддерживающей сжатие, шифрование, хеширование и другие функции.

К похожим инструментам восстановления носителей относятся `myrescue` и `safecopy`. Некоторые из этих средств будут продемонстрированы в главах 6 и 7.

Криминалистические форматы

Несколько недостатков необработанных образов привели к появлению криминалистических форматов файлов. При создании образа носителя как доказательства существуют метаданные о расследовании, следователе, характеристиках накопителя, журналах и временных метках, криптографических хешах и тому подобном. Помимо метаданных, полученный образ нередко требуется сжать или зашифровать. Специализированные криминалистические форматы облегчают реализацию этих возможностей; наиболее распространённые из них описаны ниже.

Криминалистические форматы файлов иногда называют контейнерами доказательств. В некоторых исследованиях также изложено понятие цифровых пакетов доказательств.[1] Инструменты получения данных в криминалистических форматах демонстрируются в главе 6.

EnCase EWF

Компания Guidance Software, одна из старейших фирм, выпускающих криминалистические программы, создаёт свой основной комплект EnCase, использующий Expert Witness Format (EWF). Формат EWF поддерживает метаданные, сжатие, шифрование, хеширование, разделённые файлы и другие возможности. В 2006 году Йоахим Мец создал `libewf` - открытую библиотеку и инструменты, разработанные методами обратной разработки; их поддержку можно включить при компиляции Sleuth Kit.

FTK SMART

Формат FTK SMART компании AccessData непосредственно конкурирует с EnCase EWF. Это проприетарный формат, также включающий метаданные, сжатие, шифрование, хеширование, разделённые файлы и другие возможности. Инструмент командной строки `ftkimager`, бесплатный, но не открытый, доступен у AccessData и демонстрируется в главах 6 и 7.

AFF

Advanced Forensic Format (AFF) был создан Симсоном Гарфинкелем как открытый, рецензируемый и опубликованный формат. Он включает все ожидаемые возможности криминалистического формата, а также дополнительные функции шифрования и подписания посредством стандартных сертификатов X.509. Пакет AFFlib содержит ряд инструментов для преобразования формата AFF и управления им.

AFF версии 3 сопровождается отдельно по адресу http://github.com/sshock/AFFLIBv3/. В 2009 году была опубликована статья об AFF версии 4.[2] Современный сайт AFF версии 4 находится по адресу http://www.aff4.org/. О создании Рабочей группы Advanced Forensic Format 4 (AFF4 Working Group, AFF4 WG) объявили летом 2016 года, а её первое совещание состоялось на конференции DFRWS в Сиэтле.

SquashFS как контейнер криминалистических доказательств

На протяжении книги я буду демонстрировать метод создания гибридного криминалистического контейнера, который сочетает простое получение необработанного образа с возможностью хранить сопроводительные сведения по делу, подобно более совершенным криминалистическим форматам. Метод использует SquashFS как контейнер криминалистических доказательств вместе с небольшим сценарием оболочки `sfsimage`, управляющим различными аспектами контейнера. В результате сжатый образ объединяется в одном пакете с журналами создания образа, сведениями о дисковом устройстве и любой другой информацией - фотографиями, формами цепочки хранения доказательств и так далее. Файлы находятся в доступной только для чтения файловой системе SquashFS, к которой можно обращаться без специальных криминалистических инструментов.

Общие сведения о SquashFS

SquashFS - сильно сжатая файловая система только для чтения, созданная для Linux. Её разработал Филип Лоуэр в 2002 году, а в 2009 году, начиная с версии ядра 2.6.29, она вошла в дерево ядра Linux.

SquashFS проектировалась преимущественно для загрузочных компакт-дисков и встраиваемых систем, однако обладает рядом свойств, привлекательных для контейнера криминалистических доказательств:

• SquashFS является файловой системой с высокой степенью сжатия;
• она доступна только для чтения: элементы можно добавлять, но нельзя удалять или изменять;
• она сохраняет `uid`/`gid` следователя и временные метки создания;
• она поддерживает очень большие файлы, теоретически до 16 ЭиБ;
• она включена в ядро Linux и легко монтируется как файловая система только для чтения;
• файловая система представляет собой открытый стандарт, инструменты для которого существуют в Windows и OS X;
• инструмент `mksquashfs` использует для создания контейнера все доступные процессоры.

Применение SquashFS как контейнера криминалистических доказательств - практичная альтернатива другим криминалистическим форматам, поскольку упрощает управление сжатыми необработанными образами, полученными посредством `dd`. Инструмент `sfsimage`, описанный далее, предоставляет функции, необходимые для управления контейнерами криминалистических доказательств SquashFS.

Контейнеры криминалистических доказательств SquashFS

Современные ядра Linux по умолчанию поддерживают файловые системы SquashFS. Для монтирования и доступа к SquashFS не нужны дополнительные модули или повторная компиляция ядра. Однако для создания файла SquashFS, добавления в него файла или просмотра содержимого требуется пакет `squashfs-tools`.[3] В зависимости от выбранного инструмента создания образов могут понадобиться дополнительные пакеты криминалистических программ, например `dcfldd`, `dc3dd` или `ewfacquire`.

Мой сценарий оболочки `sfsimage` доступен по адресу http://digitalforensics.ch/sfsimage/. Запуск `sfsimage` без параметров выводит справку с описанием использования:

```
$ sfsimage
Sfsimage: a script to manage forensic evidence containers with squashfs
Version: Sfsimage Version 0.8
Usage:
sfsimage -i diskimage container.sfs
sfsimage -a file ... container.sfs
sfsimage -l container.sfs ...
sfsimage -m container.sfs ...
sfsimage -m
sfsimage -u container.sfs ...
Where:
diskimage is a disk device, regular file, or "-" for stdin
container.sfs is a squashfs forensic evidence container
file is a regular file to be added to a container
and the arguments are as follows:
-i images a disk into a newly created*.sfs container
-a adds a file to an existing*.sfs container
-l lists the contents of an existing*.sfs container
-m mounts an*.sfs container in the current directory
-m without options shows all mounted sfs containers
-u umounts an*.sfs container
```

Для настройки `sfsimage` можно изменить сценарий либо создать отдельные файлы `sfsimage.conf`, которые он будет использовать. Файл конфигурации документирован комментариями и примерами и позволяет определить следующие параметры:

• предпочтительную команду создания или получения образа: `dd`, `dcfldd`, `dc3dd` и так далее;
• предпочтительную команду опроса устройства: `hdparm`, `tableau-parm` и так далее;
• каталог по умолчанию для монтирования контейнера доказательств; по умолчанию используется текущий рабочий каталог;
• способ управления привилегированными командами: `sudo`, `su` и так далее;
• разрешения и `uid`/`gid` создаваемых файлов.

Сценарий `sfsimage` использует шаблон имён `*.sfs` для контейнеров криминалистических доказательств SquashFS. Вместе со сценарием поставляется страница руководства `sfsimage(1)` с более подробными сведениями.

Чтобы получить образ диска в контейнер SquashFS, запустите `sfsimage` с параметром `-i`, указав дисковое устройство и имя контейнера доказательств. Будет создан контейнер с образом и исходными метаданными о только что обработанном устройстве. В этом примере `sfsimage` настроен на применение `dc3dd` как инструмента создания образа:

```
$ sfsimage -i /dev/sde kingston.sfs
Started: 2016-05-14T20:44:12
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i /dev/sde
Current working directory: /home/holmes
Forensic evidence source: if=/dev/sde
Destination squashfs container: kingston.sfs
Image filename inside container: image.raw
Acquisition command: sudo dc3dd if=/dev/sde log=errorlog.txt hlog=hashlog.txt
hash=md5 2>/dev/null | pv -s 7918845952
7.38GiB 0:01:19 [95.4MiB/s] [=========================================>] 100%
Completed: 2016-05-14T20:45:31
```

Здесь создаётся контейнер SquashFS, внутри которого формируется обычный необработанный образ. Также создаются дополнительные журналы и сведения; их можно добавить и отдельно.

Дополнительные доказательства добавляются в контейнер командой `sfsimage` с параметром `-a`. Например, если в ранее созданный контейнер криминалистических доказательств требуется добавить фотографию физического диска, это выполнит следующая команда:

```
$ sfsimage -a photo.jpg kingston.sfs
Appending to existing 4.0 filesystem on kingston.sfs, block size 131072
```

Чтобы просмотреть содержимое контейнера криминалистических доказательств SquashFS, запустите сценарий `sfsimage` с параметром `-l`:

```
$ sfsimage -l kingston.sfs
Contents of kingston.sfs:
drwxrwxrwx holmes/holmes 135 2016-05-14 20:46 squashfs-root
-r--r--r-- holmes/holmes 548 2016-05-14 20:45 squashfs-root/errorlog.txt
-r--r--r-- holmes/holmes 307 2016-05-14 20:45 squashfs-root/hashlog.txt
-r--r--r-- holmes/holmes 7918845952 2016-05-14 20:44 squashfs-root/image.raw
-rw-r----- holmes/holmes 366592 2016-05-14 20:45 squashfs-root/photo.jpg
-r--r--r-- holmes/holmes 431 2016-05-14 20:45 squashfs-root/sfsimagelog.txt
```

Этот вывод показывает содержимое контейнера `*.sfs` без его монтирования, а также правильное время создания или добавления файлов. Журнал ошибок, журнал хешей и журнал `sfsimage` документируют действия и ошибки. Файл `photo.jpg` - фотография, впоследствии добавленная в контейнер.

После монтирования файла `*.sfs` можно обращаться к полученному образу и добавленным файлам метаданных в контейнере SquashFS. Содержимое становится доступно как обычная часть файловой системы. Поскольку SquashFS доступна только для чтения, опасности изменения содержимого нет.

В следующем примере файл `*.sfs` монтируется параметром `-m`, после чего к полученному образу применяются обычные криминалистические инструменты - в данном случае `mmls` из Sleuth Kit:

```
$ sfsimage -m kingston.sfs
kingston.sfs.d mount created
$ mmls kingston.sfs.d/image.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot       Start        End          Length       Description
000:  Meta       0000000000   0000000000   0000000001   Primary Table (#0)
001:  -------    0000000000   0000002047   0000002048   Unallocated
002:  000:000    0000002048   0015466495   0015464448   Linux (0x83)
```

Обратите внимание, что смонтированный контейнер `*.sfs` по умолчанию появляется как каталог `*.sfs.d`. После монтирования обращаться к файлам внутри каталога можно обычными средствами операционной системы, криминалистическими инструментами и даже путём экспорта файлов как общего сетевого диска.

Когда точка монтирования `*.sfs.d` больше не нужна, размонтируйте её параметром `-u`:

```
$ sfsimage -u kingston.sfs.d
kingston.sfs.d unmounted
```

Запуск `sfsimage -m` без точки монтирования выводит список всех смонтированных контейнеров SquashFS. В одной системе можно одновременно смонтировать несколько контейнеров.

С файлами дисковых образов всегда было трудно работать в криминалистической среде из-за их размеров. Большие диски создают проблемы с пространством и материально-технические сложности. Практичные методы сжатия, такие как SquashFS, помогают справиться с этой проблемой. Чтобы показать практическую пользу сжатой файловой системы, `sfsimage` применили для получения образа исследуемого диска ёмкостью 8 Тбайт (`bonkers`) в системе следователя, где имелось лишь 2 Тбайт дискового пространства. Полное получение заняло более 16 часов, а полученный сжатый файл SquashFS имел размер всего 1 Тбайт. Смонтированный файл SquashFS предоставляет доступ ко всем 8 Тбайт в виде файла необработанного образа. Образ сжимается на лету без каких-либо временных файлов. Размеры файла `*.sfs` и файла образа показаны ниже.

```
$ ls -l bonkers.sfs bonkers.sfs.d/bonkers.raw
-rw-r----- 1 holmes root 1042820382720 Jun 28 13:06 bonkers.sfs
-r--r--r-- 1 root root 8001562156544 Jun 27 20:19 bonkers.sfs.d/bonkers.raw
```

Использование SquashFS - практичное и эффективное решение для работы с необработанными файлами в сжатом виде и альтернатива другим контейнерам криминалистических доказательств.

Заключительные мысли

В этой главе были представлены различные форматы криминалистических образов. Я дал краткий обзор и изложил историю разных инструментов, которые можно применять для криминалистического получения данных с накопителя. Вы также познакомились с файловой системой SquashFS и сценарием `sfsimage`, предназначенным для создания контейнеров криминалистических доказательств SquashFS и управления ими. Представленные в главе инструменты и форматы будут использоваться в примерах на протяжении остальной книги.

Сноски

[1] [назад] Philip Turner, “Unification of Digital Evidence from Disparate Sources (Digital Evidence Bags)” (paper presented at Digital Forensic Research Workshop [DFRWS], New Orleans, Louisiana, August 18, 2005), http://dfrws.org/2005/proceedings/turner_evidencebags.pdf.
[2] [назад] M.I. Cohen, Simson Garfinkel, and Bradley Schatz, “Extending the Advanced Forensic File Format to Accommodate Multiple Data Sources, Logical Evidence, Arbitrary Information and Forensic Workflow,” *Digital Investigation* 6 (2009): S57-S68.
[3] [назад] В системах на основе Debian пакет устанавливается командой `apt-get install squashfs-tools`.

Глава 4. Планирование и подготовка

В этой главе описаны подготовительные действия, выполняемые перед созданием образа диска или носителя. К ним относятся организация аудиторского следа действий следователя, сохранение вывода для отчётов и выбор правил именования. Кроме того, я описываю различные материально-технические трудности криминалистического получения данных с носителей и создание защищённой среды с блокировкой записи.

Тема криминалистической готовности отчасти пересекается с разделами этой главы. Однако криминалистическая готовность - более широкое понятие, включающее общее планирование, бюджетирование, инфраструктуру лаборатории, подготовку сотрудников, закупку оборудования и программ и другие вопросы. Если перечисленные требования считать криминалистической готовностью «макроуровня», сведения этой главы можно назвать готовностью «микроуровня». Здесь внимание сосредоточено на более узкой области: настройке рабочей среды эксперта-криминалиста, а также инструментах и отдельных задачах, необходимых для анализа диска или носителя.

Стоит отметить, что криминалистическая готовность организации частного сектора, например корпоративной криминалистической лаборатории, отличается от готовности некоторых государственных организаций, таких как правоохранительные органы. Частные организации, особенно крупные корпоративные ИТ-среды, могут сами определять устройство и эксплуатацию своей ИТ-инфраструктуры. В такой контролируемой среде криминалистическую готовность можно встроить в ИТ-инфраструктуру, предоставив эксперту преимущества на случай расследования или инцидента. В этой главе рассматриваются подготовительные криминалистические задачи, общие для частного и государственного секторов.

Ведение аудиторского следа

Аудиторский след или журнал хранит хронологическую запись действий, событий и задач. Он может иметь разную степень подробности и вестись вручную либо автоматически. В этом разделе рассматриваются несколько методов ручного отслеживания задач в командной строке, а также автоматизированная регистрация действий в ней.

Управление задачами

Во время криминалистического исследования полезно вести общий журнал запланированных и завершённых действий. Запланированные задачи превращаются в выполненные, а выполненные составляют историческую запись исследования. Во время работы вам часто будет приходить в голову задача, которую нужно решить позднее, или уже выполненная задача, которую следует отметить. Быстрые заметки и более полные списки задач становятся всё ценнее по мере увеличения продолжительности исследования - до многих часов, дней или ещё большего срока - либо при участии нескольких экспертов.

Ведение списка запланированных и выполненных задач во время исследования важно по множеству причин:

• помогает ничего не забыть;
• позволяет не повторять уже выполненную работу;
• улучшает взаимодействие и координацию при работе в группах;
• демонстрирует соблюдение политик и процедур;
• облегчает учёт, включая выставление счетов;
• помогает готовить документацию и отчёты, например официальные отчёты об инцидентах или криминалистические заключения;
• позволяет провести разбор после инцидента, определить усвоенные уроки и поддержать оптимизацию процессов;
• помогает сохранять долговременную историческую запись выполненных действий;
• поддерживает обучение и подготовку новых участников группы;
• служит руководством, помогающим вспомнить сложные процедуры;
• предоставляет сведения для устранения неполадок и получения поддержки;
• сохраняет запись работы, выполненной внешними и сторонними экспертами.

Доступно множество коммерческих диспетчеров задач и средств управления расследованиями, однако этот раздел посвящён простому управлению задачами из командной строки. Командная строка позволяет быстро отслеживать задачи и действия, не покидая терминал ради доступа к другому графическому или веб-приложению.

Существует множество открытых диспетчеров задач командной строки, пригодных для управления действиями эксперта-криминалиста. К важнейшим критериям относятся надёжная запись задач и подробная временная метка, а не только дата.

Taskwarrior

Taskwarrior - популярный диспетчер задач с множеством функций для быстрого и эффективного управления большими списками. Дополнительные сведения доступны по адресу http://taskwarrior.org/. Следующие примеры показывают практическое применение команд Taskwarrior в криминалистической лаборатории.

Чтобы добавить несколько запланированных задач:

```
$ task add acquire PC disk and transfer to evidence safe due:friday
Created task 1.
$ task add have a meeting with investigation team to plan analysis
Created task 2.
```

Чтобы вывести текущий список задач; команда `task info` покажет время и более подробные сведения:

```
$ task list
ID Due Age Description
1 2015-06-05 1m acquire PC disk and transfer to evidence safe
2 3s have a meeting with investigation team to plan analysis
2 tasks
```

Чтобы завершить задачу из списка:

```
$ task 2 done
Completed task 2 'have a meeting with investigation team to plan analysis'.
Completed 1 task.
```

Чтобы зарегистрировать выполненную задачу, не помещая её в список:

```
$ task log requested history of PC use at the firm
Logged task.
```

Taskwarrior полезен для управления большим числом задач. Он формирует отчёты, выполняет поиск и сортировку и предоставляет различные уровни настраиваемой детализации. Taskwarrior сохраняет временные метки и уникальные идентификаторы UUID каждой задачи, управляет приоритетами запланированных задач и хранит историю выполненных действий. Возможность создавать определяемые пользователем атрибуты позволяет приспособить его к конкретной среде, например криминалистической лаборатории или процессу исследования.

Todo.txt

Список выполненных и запланированных задач можно вести и посредством редактирования простого текстового файла. Примером служит формат файла `todo.txt`, созданный Джиной Трапани; дополнительные сведения см. на http://todotxt.com/. Система `todo.txt` определяет формат файла для дат создания и выполнения задач, приоритетов, проектов и контекстов. Она также предоставляет сценарий оболочки для управления файлом `todo.txt`. Хотя сценарий `todo.sh` выполняет все необходимые операции со списком задач `todo.txt`, самим форматом можно управлять в обычном текстовом редакторе. В этой записи приоритет обозначается скобками - `(A)`, `(B)` и так далее, - ключевые слова контекста предваряются знаком `@`, а ключевые слова проекта - знаком `+`. Выполненные задачи начинаются с `x`.

Ниже приведён пример файла `todo.txt`:

```
(A) Sign chain of custody forms @reception
(B) Start forensic acquisition +Disk_A @imagingstation
Discuss analysis approach with investigation team @meetingroom
x 2015-05-30 08:45 upgrade ram in imaging PC @imagingstation
```

Приложения `todo.txt` используют только даты, но не временные метки. При применении этой системы время выполненной задачи необходимо добавлять вручную.

Псевдоним оболочки

Журнал выполненных действий эксперта можно вести и без программ управления задачами. Например, следующий простой псевдоним оболочки перенаправляет краткое описание в файл вместе с временной меткой:

```
$ alias log="echo $2 \`date +%FT%R\` >> ~/examiner.log"
```

Имя файла журнала и формат даты можно настроить по своему усмотрению. Чтобы быстро записать действие или просмотреть прошлые действия, достаточно однострочной команды, которую можно ввести в любой момент исследования. Когда происходит что-то значительное или заслуживающее внимания, введите `log`, а за ним краткое описание выполненного действия. Например:

```
$ log removed hdd from PC and attached to examiner machine
...
$ log started forensic acquisition of the disk
...
$ log acquisition completed, disk sealed in evidence bag
...
$ cat ~/examiner.log
2015-05-30T09:14 informed that seized PC was enroute to forensic lab
2015-05-30T10:25 PC arrived, chain of custody forms signed
2015-05-30T10:47 removed hdd from PC and attached to examiner machine
2015-05-30T10:55 started forensic acquisition of the disk
2015-05-30T15:17 acquisition completed, disk sealed in evidence bag
2015-05-30T16:09 disk transferred to evidence safe for storage
```

Простые системы управления задачами полезны сотрудникам, которые проводят много времени в командной строке. Они также удобны при удалённой работе в системах через защищённую оболочку `ssh`.

История оболочки

В этом разделе объясняется настройка автоматизированного журнала команд, введённых экспертом в командной строке. В идеале такая регистрация не должна повышать сложность работы или мешать текущему криминалистическому исследованию. С помощью различных инструментов действия эксперта в командной строке можно записывать автоматическими фоновыми процессами. В ходе криминалистического расследования такой подход совершенно прозрачен для самого эксперта.

Оболочка Unix/Linux изначально не проектировалась с учётом ведения журнала или аудиторского следа. Ранее создавались исправления, расширявшие механизм истории; предпринимались попытки перехватывать команды во время работы оболочки; коммерческие продукты выполняли различные виды корпоративного журналирования. Разработка надёжной системы аудита и контроля соблюдения требований, которая записывала бы все команды с временными метками,

включая встроенные команды оболочки, запускаемые программы и конвейеры, находится за рамками книги.

Историю оболочки Bash можно настроить так, чтобы она удовлетворяла следующим основным требованиям:

• записывала команду, введённую экспертом;
• записывала временную метку каждой введённой команды;
• записывала все команды, включая повторы, комментарии и команды с начальным пробелом;
• не допускала усечения или перезаписи файлов истории;
• предотвращала конфликты при использовании нескольких окон терминала в одной системе;
• включала историю команд привилегированного и непривилегированного пользователя.

Применение обычной истории Bash в качестве аудиторского следа - примитивное решение. В неё не записываются важные сведения: время завершения команды, рабочий каталог, в котором она выполнялась, и код возврата. История Bash также не защищена от вмешательства: эксперт легко может изменить или удалить её. Создание защищённой, устойчивой к вмешательству среды аудита с ограниченным доступом находится за рамками книги.

Некоторые оболочки, например `zsh`, обладают дополнительными функциями истории, позволяющими регистрировать затраченное время. Среди других предложенных решений по улучшению журналирования оболочки - применение `PS1`, `PROMPT_COMMAND`, `trap`, `DEBUG` и привязок клавиш для изменения команды перед выполнением. Расширить регистрацию действий в командной строке можно также посредством журналирования `sudo`, `auditd` или специальных сценариев, например `preexec.sh`. Полезное руководство по адресу http://www.pointsoftware.ch/en/howto-bash-audit-command-logger/ подробно рассматривает эту проблему и предлагает решение. Аудиторский след командной строки следует адаптировать к политикам и ожиданиям конкретной лаборатории.

Для базового журналирования команд можно настроить встроенную функцию истории оболочки. Bash обладает рядом полезных возможностей, в том числе позволяет включить временные метки вводимых команд. Чтобы выполнить основные требования из предыдущего списка, добавьте следующие команды в сценарии запуска Linux, например `.bashrc`:

```
set -o history
shopt -s histappend
export HISTCONTROL=
export HISTIGNORE=
export HISTFILE=~/.bash_history
export HISTFILESIZE=-1
export HISTSIZE=-1
export HISTTIMEFORMAT="%F-%R "
```

Эти команды обеспечивают включение истории и режим добавления, а не перезапись при каждом новом входе. Две переменные, `HISTCONTROL` и `HISTIGNORE`, управляют тем, какие команды сохраняются в файле истории. Распространённая настройка по умолчанию игнорирует повторы и команды, начинающиеся с пробела. Чтобы гарантировать полную регистрацию всех команд, переменным `HISTCONTROL` и `HISTIGNORE` явно присваиваются пустые значения. Переменная `HISTFILE` явно задаётся, чтобы история команд, хранящаяся в памяти, сохранялась при завершении оболочки. Для `HISTFILESIZE` и `HISTSIZE` устанавливается значение `-1`, чтобы история не усекалась и не перезаписывалась. Переменная `HISTTIMEFORMAT` включает запись временных меток в файл истории и позволяет задать формат времени. Формат может учитывать региональные настройки и должен включать временную метку, а не только дату.

По завершении исследования историю можно сохранить в текстовый файл и включить в сопроводительные файлы данных исследования. Затем историю можно очистить и подготовить к следующему исследованию следующими командами:

```
$ history > examiner_bash_history.txt
$ history -c; history -w
```

Синхронизация истории между несколькими экземплярами оболочки может быть непростой, поскольку каждая оболочка хранит историю в памяти и записывает её в файл только при завершении. Настройка переменной `PROMPT_COMMAND='history -a; history -r'` записывает, добавляя, и считывает новые команды из файла истории Bash при каждом отображении приглашения.

Активно развиваемый регистратор команд Snoopy предоставляет ряд функций, включая запись команд в `syslog`. Snoopy представляет собой предварительно загружаемую библиотеку-обёртку для системных вызовов `execv()` и `execve()`. Она прозрачна для пользователей; включить и настроить её можно, добавив библиотеку Snoopy в `/etc/ld.so.preload` и отредактировав файл `/etc/snoopy.ini`.

Например, предположим, что в приглашении Bash вводится следующая последовательность команд:

```
# fls -p -r /dev/sda1 | grep -i "\.doc$" |wc -l
10
```

Каждая из этих команд отдельно записывается в `syslog` вместе с различными подробностями:

```
Jun 5 10:47:05 lab-pc snoopy[1521]: [uid:0 sid:1256 tty:(none) cwd:/ filename:
/bin/grep]: grep -i \.doc$
Jun 5 10:47:05 lab-pc snoopy[1522]: [uid:0 sid:1256 tty:(none) cwd:/ filename:
/usr/bin/wc]: wc -l
Jun 5 10:47:05 lab-pc snoopy[1520]: [uid:0 sid:1256 tty:/dev/pts/0 cwd:/ filename:
/usr/bin/fls]: fls -p -r /dev/sda1
```

Дополнительные сведения и последняя версия Snoopy доступны по адресу https://github.com/a2o/snoopy/.

Регистраторы терминала

Иногда полезно показать работу, выполненную в терминале, вместе с выводом команд (`stdout`), сообщениями об ошибках (`stderr`) и другими сообщениями или действиями, видимыми во время сеанса. Существует несколько инструментов для записи действий сеанса и даже их последующего воспроизведения.

Самый известный инструмент - `script`. В следующем примере `script` запускается так, чтобы вывод добавлялся в файл вместе с данными времени для воспроизведения. После запуска `script` можно выполнять любые обычные команды оболочки, и они будут сохранены для последующего просмотра.

```
$ script -a -tscript.timing script.output
Script started, file is script.output
```

По завершении записываемого сеанса введите `exit` или нажмите **CTRL-D**. Просмотреть запись можно командой `scriptreplay`:

```
$ scriptreplay -m1 -tscript.timing script.output
...[session plays back here]...
```

К распространённым проблемам, осложняющим применение этого метода, относятся обработка управляющих символов и таких событий, как изменение размера терминала. Доступны и другие регистраторы и перехватчики TTY с похожими возможностями, например `ttyrec` и `termrec`.

Мультиплексоры терминала, такие как `tmux` и GNU `screen`, также обеспечивают некоторую степень журналирования, полезную в определённых ситуациях. В `screen` можно настроить регистрацию отсоединённого сеанса из самого сеанса, нажав **CTRL-A**, а затем **H**.

Мультиплексор терминала `tmux` теперь поддерживает журналирование посредством параметра `pipe-pane`, как показано ниже:

```
$ tmux pipe-pane -o -t session_index:window_index.pane_index 'cat >> ~/output
.window_index-pane_index.txt'
```

Аудит Linux

Профессиональным лабораториям может потребоваться более надёжное журналирование или аудиторский след для выполнения строгих политик организации либо нормативных требований. Один из способов добиться этого - пакет аудита Linux `auditd`. Обычно он предполагает работу демона `auditd` с `pam_tty_audit.so`, настроенным как модуль PAM. Действия аудиторского следа можно просматривать командой `aureport`.

Применение `auditd` даёт несколько преимуществ безопасности, особенно в сочетании с детализированным управлением доступом, например посредством `sudo`. Аудиторские следы, особенно записываемые на центральный узел журналирования, можно сделать сравнительно устойчивыми к вмешательству, повысив тем самым целостность записи исследовательской работы.

Всеобъемлющий аудиторский след способен записывать всю активность TTY, включая нажатия клавиш, а также отслеживать доступ к файлам и множество других системных событий. Настройка аудита и формирования аудиторских отчётов может быть сложным процессом и находится за рамками книги.

Обсуждение других решений и приёмов можно найти по адресам http://www.pointsoftware.ch/en/howto-bash-audit-command-logger/ и http://whmcr.com/2011/10/14/auditd-logging-all-commands/.

Начиная с Bash 4.1 добавлена новая возможность записывать историю команд в `syslog`; для её включения может потребоваться повторная компиляция.

Организация собранных доказательств и вывода команд

При проведении криминалистического исследования в командной строке вывод различных инструментов и утилит часто сохраняют в файлы для последующего обращения и подготовки отчётов. Это можно сделать, перенаправив вывод команд в текстовые файлы. Затем файлы сохраняются вместе с остальными собранными данными исследования. При сборе и сохранении больших объёмов доказательных данных важно поддерживать понятную и упорядоченную структуру файлов и каталогов. В этом разделе рассматриваются разные стратегии достижения этой цели.

Правила именования файлов и каталогов

Чтобы уменьшить путаницу среди всех файлов, каталогов, точек монтирования, образов и других сохранённых данных, собранных при исследовании, лучше придерживаться правил именования. Имена должны быть достаточно описательными и интуитивными, но следует избегать избыточности в словах и расширениях. Самое важное - последовательно применять правила именования на протяжении всего расследования или инцидента и в разных инцидентах.

С системами, устройствами хранения и съёмными носителями связаны определённые уникальные идентификаторы. Они могут пригодиться при выборе правил именования:

• номер актива компании или инвентарный номер персонального компьютера;
• серийный номер дискового накопителя, присвоенный изготовителем;
• 64-разрядное всемирное имя (World Wide Name, WWN) дискового накопителя;
• UUID блочного устройства для файловых систем и RAID;
• значение криминалистического хеша образа дискового накопителя;
• 48-разрядный MAC-адрес сетевой интерфейсной платы (Network Interface Card, NIC);
• номер доказательства криминалистической лаборатории, возможно указанный на наклейке или бирке накопителя;
• номер пакета доказательств криминалистической лаборатории, содержащего диск.

Там, где это разумно, начинайте любую нумерацию с 1, а не с 0. Программисты и инженеры склонны начинать с 0, но люди, читающие и проверяющие отчёты об исследовании, - юристы, судьи, руководители и другие - могут не иметь технической подготовки и ожидают, что нумерация начнётся с 1.

На протяжении книги для файлов необработанных образов используется расширение `*.raw`. Распространённое расширение `*.dd` подразумевает применение инструмента `dd`, хотя это может быть не так. Расширение `*.raw` точно описывает файл, не связывая его с конкретным средством получения образа.

В идеальном случае имя файла необработанного образа должно связывать криминалистический образ с уникальным атрибутом физического объекта. При использовании криминалистического формата эти уникальные сведения можно встроить в файл образа как метаданные. Это позволяет связать отдельный физический диск с образом, а отдельный образ - с физическим диском. После этого диск и образ остаются связанными независимо от окружающего контекста: имён каталогов, полок с доказательствами и прочего. Так устанавливается звено цепочки хранения доказательств между физическим и цифровым мирами.

Если анализируется большое число дисков, в имя файла образа можно включить серийный номер. Имя может содержать разную степень подробности. Хотя `server12-slot3-seagate-3.5in-disk-500gb-SN12345ACBDEE.raw` очень описательно, оно может оказаться чрезмерно подробным и неудобным. Практичные правила именования для многих

простых инцидентов могут ограничиваться видом носителя и номером, например `disk1`, `tape1`, `ssd1`, `stick1`, `card1`, `cdrom1`, `dvd1`, `bluray1`, `floppy1` и так далее. Иногда лучше всего подойдёт краткое описание диска с серийным номером, например `crucial-ssd-15030E69A241.raw`. Часто полезно выбирать имена образов, которые эксперты легко употребляют в разговоре, например: «Мы нашли файл на `disk1`». Термины в разговорах, необработанном выводе исследования и итоговых отчётах должны следовать единой номенклатуре.

Когда файлы извлекаются из дисковых образов, архивов или других составных образов, добавляйте к имени знак подчёркивания, чтобы обозначить извлечение. Это не позволит вам и другим людям случайно открыть вредоносную программу, HTML-страницу с отслеживающими элементами, макросы в документах Office и другие исполняемые файлы и сценарии, способные запуститься при открытии.

Ниже показано несколько примеров:

```
$ icat image.raw 68 > photo.jpg_
$ icat image.raw 34 > customerlist.xls_
$ icat image.raw 267 > super-updater57.exe_
```

Если имя извлечённого файла уже заканчивается знаком подчёркивания, добавьте ещё один. Знак подчёркивания в конце ясно показывает, что файл извлечён как доказательство с подозреваемого накопителя.

При анализе извлечённого файла, сохранении вывода инструмента или создании ручных заметок создайте текстовый файл с исходным именем, добавив к нему `_.txt`. Например:

```
$ exif photo.jpg_ > photo.jpg_.txt
$ vi customerlist.xls_.txt
$ objdump -x super-updater57.exe_ > super-updater57.exe_.txt
```

Расширение `_.txt` обозначает, что текстовый файл содержит заметки, вывод инструментов и результаты криминалистического анализа извлечённого файла. Имя связано с файлом, первоначально извлечённым из образа. Текстовый файл может содержать закладки и аннотации эксперта, доступные для поиска. Если иным образом не очевидно, откуда получен извлечённый файл - с какого диска, раздела и так далее, - хорошей практикой будет создавать такие сопутствующие текстовые файлы; они также могут объяснять, почему файл был выбран для извлечения.

Расширение файла всегда должно обозначать формат содержимого. Например:

- `*.txt` можно открыть и прочитать в текстовом редакторе;
- `*.raw` - необработанный дамп данных диска, памяти и тому подобного;
- `*.pcap` - захваченный сетевой трафик;
- `*.db` - база данных, возможно список файлов Sleuth Kit;
- `*.sfs` - контейнер доказательств SquashFS;
- `*.e01` и `*.aff` - криминалистические форматы.

С каждым делом, инцидентом или расследованием связан физический носитель. Ему соответствует криминалистический образ и связанный вывод разных программ, например `hdparm` и `smartctl`. С каждым криминалистическим образом связан вывод различных программ, таких как `mmls` и `fls`, а с каждым извлечённым файлом может быть связан вывод `exif`, `objdump` и других средств. Правила именования помогают сохранить всё в порядке и позволяют системе организации масштабироваться по мере роста данных расследования.

Какой объём сведений следует включать в имена файлов и каталогов? Когда разумнее создать соответствующий текстовый файл с дополнительным описанием? Как связать такой файл с образом? Рассмотрим два варианта представления одного инцидента.

Пример сведений, встроенных в имена файлов:

```
case42.txt
image6.case42.raw
image6.case42.raw.txt
mmls.image6.case42.txt
fls.part1.image6.case42.txt
```

Пример тех же сведений, встроенных в структуру каталогов:

```
./case42/case.txt
./case42/image6/image.raw
./case42/image6/image.raw.txt
./case42/image6/mmls.txt
./case42/image6/part1/fls.txt
```

Для вручную написанных заметок, дополнительных описаний, оговорок, проблем и других разрозненных комментариев в определённом контексте полезно хранить сведения в простых файлах `notes.txt` или `readme.txt` в рабочих каталогах. Они могут служить напоминаниями, подсказками или предупреждениями, которые вы или другие эксперты прочтёте позднее.

При записи веб-адресов, открытие которых может быть рискованным, заменяйте `http` на `hxxp`, чтобы другие люди случайно не щёлкнули ссылку. Такие ссылки могут вести к вредоносным программам, личным сайтам, за посещением которых следит подозреваемый, страницам с отслеживающими элементами или другому содержимому, к которому нельзя обращаться без понимания последствий.

Масштабируемая структура каталогов исследования

У каждого инцидента, дела или расследования должен быть один уникальный каталог, например `case42`. Все собранные доказательства, образы и аналитическая работа должны находиться в иерархии под этим единственным корневым каталогом. Хорошо спланированная структура каталогов способна расти вместе с расследованием. Единый каталог удобен и тогда, когда над одним инцидентом работают несколько экспертов, совместно использующих структуру. Будьте готовы реорганизовать её, если сложность инцидента возрастёт. При извлечении большого числа файлов для индивидуального анализа рассмотрите возможность создать каталог экспорта, подобный используемому в EnCase.

Масштаб исследования нередко неожиданно увеличивается: криминалистическое исследование одного подозреваемого диска может превратиться в крупную работу с несколькими персональными компьютерами и множеством дисков. Предположим, кто-то сообщил о странном или подозрительном поведении компьютера либо сотрудника. Для исследования изымают один диск. Предварительные результаты показывают участие флеш-накопителя USB. Его находят и исследуют, после чего с инцидентом связывают второй компьютер. В нём два внутренних жёстких диска и устройство записи DVD. Дальнейший поиск обнаруживает спрятанную в шкафу коробку DVD, заполненных данными. Затем выясняется, что к инциденту также относятся внешний жёсткий диск USB и запасной ноутбук в другом здании. Объём собранных доказательств вырос от одного жёсткого диска до 16 носителей. Такой гипотетический инцидент не является редкостью в крупных организациях. При подготовке к исследованию следует предусматривать расширение охвата. Правила именования должны масштабироваться по мере роста расследования.

Некоторыми компьютерами пользуются несколько людей, а некоторые люди работают на нескольких компьютерах. Ноутбук необязательно привязан к физическому месту. Съёмные носители могут совместно использоваться и подключаться к разным компьютерам и ноутбукам. Со временем оборудование компьютеров меняется, офисы могут переезжать, сотрудники

подразделений сменяются, а организация реструктурируется. Проектируйте имена файлов и каталогов с учётом этих изменений.

По мере исследования число выходных файлов растёт, поскольку анализируется больше собранных данных и создаётся новый вывод. Хорошая практика - создать структуру каталогов, разделяющую файлы и организующую вывод исследования. Как и имя файла, имя каталога должно обозначать содержимое, не раскрывая конфиденциальных сведений. Отдельный каталог для каждого анализируемого диска или образа разделяет файлы и позволяет масштабировать расследование.

Самое небольшое исследование обычно включает один диск. Немного более крупное может охватывать персональный компьютер с несколькими дисками; рассмотрим следующую структуру каталогов:

```
HDD1
HDD2
HDD3
```

Другой пример - исследование целого рабочего места, состоящего из настольного компьютера, возможно с несколькими дисками, ноутбука, нескольких USB-накопителей, нескольких CD-ROM и DVD и внешнего дискового блока. Удобная структура каталогов организует каждый носитель вместе с файлами вывода команд и позволяет без труда расширять исследование дальше. Представьте более крупное расследование, охватывающее несколько рабочих мест в нескольких офисных зданиях разных стран. В больших международных организациях такие расследования возможны, поэтому продуманные правила именования сохранят порядок в процессе исследования.

Удобно полагаться на структуру каталогов для разделения вывода команд по разным дискам, компьютерам, пользователям и местам. Тогда эти сведения не придётся включать в имена выходных файлов. Например:

```
OFFICE-US123
    USER-123456
        PC1-HDA
        CD1
        CD2
    USER-98765
        PC1-HDA
        PC1-HDB
        NB1-HDA
        USB1
        USB2
        DVD1
OFFICE-UK567
    PC1-HDA
```

В этом примере два офиса, US123 и UK567, находятся соответственно в Соединённых Штатах и Великобритании. Американский офис разделён по рабочим местам пользователей, а для каждого исследуемого носителя выделен каталог. Компьютер британского офиса не связан с определённым пользователем, возможно потому, что находится в переговорной, и это отражено в структуре каталогов.

Вместо идентификатора сотрудника для носителя в структуре каталогов можно использовать инвентарный номер, присвоенный ИТ-подразделением организации. С этим уникальным идентификатором, вероятно, связаны дополнительные сведения: дата покупки, подразделение, местонахождение офиса, данные пользователя, установленное программное обеспечение, история эксплуатации и так далее. Требования конфиденциальности могут потребовать исключить сведения из имён файлов и структуры каталогов. Например, в имена не следует включать имена подозреваемых или лиц, являющихся объектом расследования. Вместо них используйте идентификатор, инициалы или номер сотрудника. Можно также применять кодовые названия

расследований. Они обеспечивают минимальную защиту, если сведения будут потеряны, похищены или иным образом окажутся доступны позднее.

Сохранение вывода команд посредством перенаправления

После создания структуры каталогов для хранения результатов анализа разных исследуемых объектов обычный вывод команд оболочки из `stdout` перенаправляется в файлы следующим образом:

```
# fls /dev/sda1 > fls-part1.txt
# fls /dev/sda2 > fls-part2.txt
```

Чтобы включить обычный вывод и сообщения об ошибках, необходимо перенаправить в файл дескрипторы `stdout` и `stderr`. Новые версии Bash предлагают легко запоминающийся способ: добавить к знаку перенаправления амперсанд; это применимо и при передаче по конвейеру другой программе:

```
# fls /dev/sda &> fls-part1.txt
```

В других оболочках и ранних версиях Bash для объединения `stderr` и `stdin` может потребоваться запись `2>&1`. Например:

```
# fls /dev/sda > fls-part1.txt 2>&1
```

Если текстовый файл уже существует и в него нужно добавить сведения, операция добавления обозначается как `>>`. Например:

```
# grep clientnames.xls fls-part1.txt >> notes.txt
```

Здесь все вхождения известного имени файла добавляются в конец `notes.txt`.[1] Если `notes.txt` не существует, он будет создан.

Многие криминалистические задачи командной строки требуют значительного времени и могут выполняться по нескольку часов, например создание образа диска или операции с очень большими файлами. Полезно иметь временную метку, показывающую продолжительность команды. Эту функцию предоставляет команда `time`. Существуют две распространённые реализации: встроенная команда оболочки с базовыми возможностями и утилита GNU с дополнительными функциями. Главное преимущество встроенной версии `time` заключается в измерении времени всего конвейера команд, тогда как GNU `time` измеряет только первую команду конвейера.

Ниже показано применение команды `time` для запуска программы создания дискового образа:

```
# time dcfldd if=/dev/sdc of=./ssd-image.raw
3907328 blocks (122104Mb) written.
3907338+1 records in
3907338+1 records out

real 28m5.176s
user 0m11.652s
sys 2m23.652s
```

Оболочка `zsh` умеет записывать затраченное командой время как часть файла истории. В Bash такой возможности пока нет.

Ещё одна полезная в некоторых ситуациях команда - средство добавления временных меток `ts`. К каждой строке любого переданного в `ts` вывода будет добавлена временная метка.

```
# (ls -l image.raw; cp -v image.raw /exam/image.raw; md5sum /exam/image.raw) |ts
May 15 07:45:28 -rw-r----- 1 root root 7918845952 May 15 07:40 image.raw
May 15 07:45:40 'image.raw' -> '/exam/image.raw'
May 15 07:45:53 4f12fa07601d02e7ae78c2d687403c7c /exam/image.raw
```

В этом примере были выполнены три команды, объединённые круглыми скобками, а их вывод передан `ts`, в результате чего сформирована временная шкала.

Оценка материально-технических требований инфраструктуры получения данных

При криминалистическом получении данных с носителей важны различные материально-технические вопросы. Управление крупными полученными криминалистическими образами - нетривиальная задача, требующая планирования и предусмотрительности. Необходимо учитывать такие факторы, как ёмкость диска, продолжительность, производительность и условия окружающей среды.

Размеры образов и требования к дисковому пространству

Криминалистические образы носителей на несколько порядков крупнее небольших файлов, с которыми обычно работает персональный компьютер. Управление дисковыми образами такого размера требует дополнительных размышлений и планирования. При подготовке системы исследования необходимо учитывать и определённые материально-технические факторы. Тщательная подготовка и планирование исследования сэкономят время и силы, а также помогут избежать проблем, способных нарушить процесс.

При создании криминалистического образа диска объёмом в сотни гигабайт или терабайты копируются не файлы, а отдельные секторы. Даже если на диске ёмкостью 1 Тбайт находится единственный документ Microsoft Word размером 20 Кбайт, несжатый криминалистический образ всё равно займёт 1 Тбайт. На момент написания книги на рынке уже появились диски объёмом 10 Тбайт, что ещё больше усложняет криминалистическое получение данных.

Главные материально-технические факторы управления дисковыми образами - время эксперта и ёмкость дисков основной системы. Перед началом криминалистического получения данных с исследуемого диска или носителя необходимо задать ряд вопросов:

• можно ли анализировать подключённое хранилище на месте без создания криминалистического образа;

• каков размер исследуемого диска;
• сколько свободного пространства имеется на машине эксперта;
• насколько хорошо образ потенциально поддаётся сжатию;
• сколько пространства требуется криминалистическим инструментам для обработки и временных файлов;
• какое предполагаемое число файлов предстоит извлечь для дальнейшего анализа;
• сколько оперативной памяти и пространства подкачки доступно на машине эксперта;
• возможно ли добавление к тому же делу или инциденту других исследуемых дисков;
• предполагается ли отдельно извлекать всё резервное или нераспределённое пространство диска;
• планируется ли извлекать отдельные разделы, возможно включая раздел подкачки;
• может ли потребоваться преобразование из одного криминалистического формата в другой;
• нужно ли готовить дисковые образы к перевозке в другое место;
• содержат ли исследуемые диски образы виртуальных машин, которые нужно извлекать и анализировать отдельно;
• содержат ли исследуемые диски большое число сжатых файлов и архивов;
• применяется ли на исследуемых дисках полнодисковое шифрование;
• потребуется ли записывать образы на другой диск или DVD для хранения либо перевозки;
• понадобится ли восстанавливать файлы из повреждённой или частично перезаписанной файловой системы;
• каким образом выполняется резервное копирование основной системы эксперта.

В некоторых ситуациях создавать образ диска необязательно. При определённых видах сортировки или беглого поиска может быть достаточно подключить диск к основной системе эксперта и работать непосредственно с активным исследуемым диском. По результатам сортировки или поиска можно решить, нужен ли криминалистический образ. В корпоративной среде такой подход способен означать простой сотрудника, которому придётся ждать проверки или анализа изъятого диска. Корпоративные среды обычно используют стандартную конфигурацию компьютера конечного пользователя, рассчитанную на отсутствие локальных пользовательских данных: вся информация сохраняется на серверах или в облаках. Экономичнее может оказаться простая замена исходного диска новым. Диски компьютеров конечных пользователей недороги, и замена исследуемого диска новым может сократить расходы, если учесть стоимость простоя сотрудника и время, необходимое для создания образа диска на месте.

Сжатие файлов

Сжатие решает ряд проблем с ёмкостью, с которыми сталкивается эксперт-криминалист. Полученный образ можно хранить в сжатом криминалистическом формате, однако результативность зависит от нескольких факторов.

Выбранный алгоритм сжатия в некоторой степени повлияет на итоговый размер и время, необходимое для сжатия исследуемого диска. Более высокий коэффициент сжатия потребует больше времени на сжатие, а затем и на распаковку.

Сравнительно новый диск персонального компьютера с большим числом нетронутых секторов, содержимое которых изготовитель заполнил нулями, будет сжиматься лучше, чем старый диск со значительным объёмом остаточных данных в нераспределённых секторах.

Диски с большим объёмом уже сжатых файлов - `*.mp3`, `*.avi` и других - почти не поддаются дальнейшему сжатию, поэтому дополнительное сжатие даст криминалистическим инструментам создания образов меньше преимуществ.

Зашифрованные исследуемые диски или диски с большим числом зашифрованных файлов сжимаются хуже незашифрованного содержимого из-за более высокой энтропии данных.

Разреженные файлы

Разреженные файлы заслуживают упоминания, поскольку имеют определённые преимущества, однако могут создавать проблемы при расчёте ёмкости диска. Некоторые файловые системы вместо фактической записи всей последовательности нулей на диск представляют её в файле посредством метаданных. В разреженных файлах имеются «дыры», где заведомо находится последовательность нулей. Для демонстрации образ нового накопителя, состоящего преимущественно из нулевых секторов, дважды получают посредством GNU `dd`:[2] сначала как обычный необработанный файл, а затем как разреженный.

```
# dd if=/dev/sde of=image.raw
15466496+0 records in
15466496+0 records out
7918845952 bytes (7.9 GB, 7.4 GiB) copied, 112.315 s, 70.5 MB/s
# dd if=/dev/sde of=sparse-image.raw conv=sparse
15466496+0 records in
15466496+0 records out
7918845952 bytes (7.9 GB, 7.4 GiB) copied, 106.622 s, 74.3 MB/s
```

Команда GNU `dd` предоставляет параметр `conv=sparse`, создающий разреженный целевой файл. Из этих примеров `dd` видно, что число переданных блоков одинаково для обычного и разреженного файла. В следующем выводе размер файла и хеш MD5 также совпадают. Однако обратите внимание, насколько различается число блоков, занятых в файловой системе: 7 733 252 против 2600.

```
# ls -ls image.raw sparse-image.raw
7733252 -rw-r----- 1 root root 7918845952 May 15 08:28 image.raw
2600 -rw-r----- 1 root root 7918845952 May 15 08:30 sparse-image.raw
# md5sum image.raw sparse-image.raw
325383b1b51754def26c2c29bcd049ae image.raw
325383b1b51754def26c2c29bcd049ae sparse-image.raw
```

Хотя разреженному файлу требуется гораздо меньше пространства, в качестве его размера по-прежнему указывается полное число байтов. Это может привести к путанице при расчёте фактически доступной ёмкости диска. Разреженные файлы часто применяются в образах виртуальных машин и способны стать проблемой при извлечении для анализа.

Разреженные файлы можно использовать и как способ уплотнения файлов образов, однако предпочтительнее и рекомендуется применять сжатые криминалистические форматы или контейнеры SquashFS. Не все программы и утилиты правильно обрабатывают разреженные файлы; они могут создавать трудности при перемещении между файловыми системами и платформами. Некоторые программы даже способны разворачивать разреженные файлы при чтении.

Сообщаемые размеры файлов и образов

Важно понимать, как сообщаются размеры данных. При работе с криминалистическими инструментами размер может обозначать байты, секторы диска, блоки файловой системы или другие единицы измерения. К обозначению байтов можно добавлять множитель - килобайты, мегабайты, гигабайты, терабайты и так далее, - причём он может означать кратность 1000 или 1024. Сектор диска может иметь размер 512 или 4096 байт. Размер блока файловой системы зависит от её типа и параметров создания. При документировании размеров в криминалистическом контексте важно всегда указывать поясняющие единицы измерения.

Многие инструменты Linux поддерживают параметр `-h`, сообщающий размеры файлов в понятной человеку форме. Например, команды `ls -lh`, `df -h` и `du -h` упрощают просмотр размеров файлов и разделов. Ниже приведён пример вывода `ls` для нескольких файлов:

```
# ls -l
total 4
-rw-r----- 1 root root 2621440000 Jan 29 14:44 big.file
-rw-r----- 1 root root 104857600 Jan 29 14:41 medium.file
-rw-r----- 1 root root 51200 Jan 29 14:42 small.file
-rw-r----- 1 root root 56 Jan 29 14:44 tiny.file
# ls -lh
total 4.0K
-rw-r----- 1 root root 2.5G Jan 29 14:44 big.file
-rw-r----- 1 root root 100M Jan 29 14:41 medium.file
-rw-r----- 1 root root 50K Jan 29 14:42 small.file
-rw-r----- 1 root root 56 Jan 29 14:44 tiny.file
```

Размеры во втором выводе команды гораздо легче читать и понимать.

Перемещение и копирование криминалистических образов

Перемещение и копирование криминалистических дисковых образов из одного места в другое требует планирования и предусмотрительности. Не относитесь к файлам образов так же, как к обычным пользовательским файлам, хотя технически они ничем не отличаются.

Получение, копирование и перемещение больших дисковых образов может занимать много часов и даже дней в зависимости от размера и скорости исходного диска и других факторов производительности. Рассмотрим перечень типичных размеров файлов и дисковых образов и среднего времени, необходимого для копирования файла с одного диска на другой:[3]

• простое текстовое электронное письмо ASCII размером 5 Кбайт - менее 1 секунды;
• типичный музыкальный файл MP3 размером 5 Мбайт - менее 1 секунды;
• ISO-образ компакт-диска размером 650 Мбайт - около 5 секунд;
• типичный DVD или загруженный из iTunes фильм размером 5-6 Гбайт - около 1 минуты;
• обычный образ мобильного телефона размером 64 Гбайт - около 10 минут;
• обычный образ диска ноутбука размером 250 Гбайт - 30-40 минут;
• типичный образ настольного компьютера размером 1 Тбайт - более 2 часов;
• типичный образ внешнего диска USB размером 2 Тбайт - более 4 часов;
• образ внутреннего диска размером 8 Тбайт - более 16 часов.

Если процесс копирования или перемещения уже начат, его прерывание может оставить данные в неполном состоянии либо потребовать дополнительного времени для возврата к исходному состоянию. Операция копирования или перемещения может создать временные файлы или привести к временному существованию двух копий образов.

В целом заранее тщательно обдумывайте копирование и перемещение крупных наборов данных и не прерывайте процесс после его начала.

Оценка времени завершения задач

Криминалистическое получение данных требует времени. Пока оно выполняется, люди и другие процессы могут ждать. Поэтому важно рассчитывать и оценивать время завершения различных процессов. Определите также, нужно ли сообщать предполагаемое время завершения другим сторонам: руководству, юридическим группам, правоохранительным органам или другим следователям. Важно управлять ожиданиями относительно необходимого времени.

Следует рассмотреть, в частности, следующие важные вопросы:

• можно ли безопасно оставить получение данных выполняться на ночь без присмотра;
• будет ли машина эксперта непригодна для другой работы во время получения данных из-за производительности или иных причин;
• можно ли выполнять другую исследовательскую работу одновременно с получением криминалистического образа;
• в каких случаях несколько задач можно выполнять параллельно;
• существуют ли задачи или процессы, которые можно выполнять только последовательно;
• есть ли задачи, блокирующие другие до своего завершения;
• можно ли совместно использовать, делегировать или распределить рабочую нагрузку между несколькими экспертами.

Предполагаемое время завершения получения данных можно рассчитать. По предыдущей работе и процессам вам должно быть известно приблизительное время первоначальной подготовки. Оно включает заполнение документов, создание необходимой структуры каталогов, документирование оборудования, подключение подозреваемых накопителей к основной системе эксперта, выбор подхода к получению данных и другие действия. Это даст оценку продолжительности этапа, предшествующего получению.

Ожидаемое время получения данных с носителя можно рассчитать по известному объёму данных, проходящих через самый медленный компонент системы, то есть узкое место.

Производительность и узкие места

Чтобы повысить эффективность криминалистического получения данных, можно оптимально настроить основную систему эксперта и оценить узкие места.

Узкое место производительности существует всегда: это просто самый медленный компонент системы, которого вынуждены ждать все остальные. В криминалистической среде в идеале узким местом должен быть исследуемый диск. Он является источником доказательств и единственной переменной производительности, которую нельзя или не следует изменять.

Оценить производительность разных компонентов системы можно по спецификациям изготовителей, посредством опроса системы различными инструментами или с помощью тестов производительности и измерений.

К полезным инструментам проверки скорости разных компонентов относятся `dmidecode`, `lshw`, `hdparm` и `lsusb`. Ниже показано несколько примеров командной строки.

Чтобы проверить семейство и модель процессора, текущую и максимальную частоту, число ядер и потоков, а также другие флаги и характеристики, используйте следующую команду:

```
# dmidecode -t processor
```

Команда для просмотра кеша процессора L1, L2 и L3:

```
# dmidecode -t cache
```

Чтобы просмотреть память, включая занятые разъёмы, объём, разрядность данных, скорость и другие подробности, используйте следующую команду:

```
# dmidecode -t memory
```

Команда для просмотра числа разъёмов PCI, их использования, обозначений и типов:

```
# dmidecode -t slot
```

Команда для просмотра интерфейсов хранения, их типов - SATA, NVME, SCSI и других - и скорости:

```
# lshw -class storage
```

Чтобы просмотреть скорость, интерфейс, кеш, частоту вращения и другие сведения о подключённых дисках, в данном примере об устройстве `/dev/sda`, используйте:

```
# hdparm -I /dev/sda
```

Чтобы просмотреть скорость внешних интерфейсов USB и, возможно, подключённого блокиратора записи, используйте:

```
# lsusb -v
```

> **Примечание.** Эти сведения можно получить множеством разных методов и команд. Каждая показанная здесь команда представляет один пример получения необходимых данных о производительности. Полный перечень всех возможных инструментов и методов находится за рамками книги.

Чтение документации изготовителей и опрос системы позволяют определить скорости разных компонентов. Для точного измерения лучше применять инструменты тестирования оборудования и профилирования программ. Среди средств проверки производительности можно назвать `mbw` для памяти и `bonnie++` для дискового ввода-вывода.

На производительность также влияют состояние и настройка операционной системы. Наблюдение за журналами оборудования основной системы - `syslog`, `dmesg` - может выявить сообщения об ошибках, неправильную конфигурацию и другие признаки неэффективности. Для наблюдения за производительностью и нагрузкой работающей машины эксперта используются `htop`, `iostat`, `vmstat`, `free` и `nmon`.

Операционную систему можно оптимизировать, обеспечив минимальное число фоновых процессов, включая запланированные посредством `cron`, настроив ядро с помощью `sysctl -a`, файловые системы основной системы эксперта посредством `tunefs`, а также управляя дисковой подкачкой и кешированием. Кроме того, убедитесь, что операционная система эксперта работает на физическом оборудовании, а не в виртуальной машине.

При поиске узких мест или оптимизации полезно представить поток данных от исследуемого диска к диску основной системы эксперта. Во время получения данные проходят через следующие аппаратные интерфейсы и компоненты:

• пластины или носитель исследуемого диска - какова скорость вращения и задержка;
• интерфейс исследуемого диска - какая версия SATA;
• логика блокиратора записи - добавляет ли она задержку;
• интерфейс блокиратора записи со стороны основной системы эксперта - используется ли USB3 с UASP;
• интерфейс основной системы эксперта - не разделяет ли USB3 шину с другими устройствами, не применяется ли мост;
• шина PCI - используется ли PCI Express и какова скорость;
• процессор, память и ядро операционной системы - каковы скорость, DMA и разрядность данных.

Эти компоненты будут пройдены дважды: сначала между исследуемым диском и основной системой эксперта, затем между основной системой и диском эксперта, где сохраняется получаемый образ.

Убедитесь, что поток данных между исследуемым диском и процессором или памятью проходит не по тому же пути, что поток между процессором или памятью и целевым диском основной системы эксперта. Например, если полевая система создания образов оснащена блокиратором записи и внешним диском для получаемого образа и оба устройства подключены к локальным портам USB, они могут совместно использовать одну шину. Тогда доступная пропускная способность будет разделена между двумя дисками, что приведёт к неоптимальной производительности.

При настройке сетевой производительности главным фактором становится скорость базовой сети, а к способам повышения производительности относятся крупные кадры Ethernet и аппаратное вычисление контрольных сумм TCP высокопроизводительной сетевой платой. Полезно также выяснить, когда и зачем разные программы обращаются к сети, например для автоматических обновлений или сетевого резервного копирования.

Подводя итог: подготовьте общий план или стратегию предполагаемых действий по получению данных. Заранее создайте хорошо проверенные процессы и инфраструктуру. Убедитесь, что правильно спланировали ёмкость и выполнили оптимизацию. Обеспечьте возможность наблюдать за выполняющимися действиями.

Для сравнения в табл. 4-1 приведены наиболее распространённые скорости шин, имеющие значение для основной системы криминалистического исследования, в байтах в секунду. Хороший справочник по скоростям передачи битов различных интерфейсов и шин доступен по адресу https://en.wikipedia.org/wiki/List_of_device_bit_rates.

Таблица 4-1. Распространённые скорости шин и интерфейсов

Шина или интерфейс	**Скорость**
Внутренние шины	
PCI Express 3.0 x16	15 750 Мбайт/с
PCI Express 3.0 x8	7880 Мбайт/с
PCI Express 3.0 x4	3934 Мбайт/с
PCI, 64 бита/133 МГц	1067 Мбайт/с
Накопители	
SAS4	2400 Мбайт/с
SAS3	1200 Мбайт/с
SATA3	600 Мбайт/с
SATA2	300 Мбайт/с
SATA1	150 Мбайт/с

Шина или интерфейс	Скорость
Внешние интерфейсы	
Thunderbolt3	5000 Мбайт/с
Thunderbolt2	2500 Мбайт/с
USB3.1	1250 Мбайт/с
USB3.0	625 Мбайт/с
Gigabit Ethernet	125 Мбайт/с
FW800	98 Мбайт/с
USB2	60 Мбайт/с

Температура и факторы окружающей среды

Во время криминалистического получения данных с диска каждый доступный сектор читается непрерывно, часто по многу часов. В результате рабочая температура диска может возрасти и вызвать проблемы. Перегрев увеличивает риск отказа, особенно у старых дисков. Исследователи Google подготовили содержательную статью об отказах жёстких дисков: http://research.google.com/archive/disk_failures.pdf.

Чтобы снизить риск ошибок чтения, повреждённых блоков или полного отказа диска, во время получения стоит следить за его температурой. Большинство изготовителей публикует нормальный диапазон рабочих температур накопителей, включая максимально допустимую.

Температуру накопителя можно вручную опрашивать несколькими инструментами. Простой инструмент `hddtemp` обращается к интерфейсу SMART за температурой диска:

```
# hddtemp /dev/sdb
/dev/sdb: SAMSUNG HD160JJ: 46C
```

`hddtemp` можно запустить как демон, периодически записывающий сведения в `syslog`, где удастся отслеживать достижение определённых порогов.

Более подробный вывод температуры диска, а иногда и её историю предоставляет `smartctl`. Например:

```
# smartctl -x /dev/sdb
...
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME FLAGS VALUE WORST THRESH FAIL RAW_VALUE
...
190 Airflow_Temperature_Cel -O---K 100 055 000 - 46
194 Temperature_Celsius -O---K 100 055 000 - 46
...
Current Temperature: 46 Celsius
Power Cycle Max Temperature: 46 Celsius
Lifetime Max Temperature: 55 Celsius
SCT Temperature History Version: 2
Temperature Sampling Period: 1 minute
Temperature Logging Interval: 1 minute
Min/Max recommended Temperature: 10/55 Celsius
Min/Max Temperature Limit: 5/60 Celsius
Temperature History Size (Index): 128 (55)

Index Estimated Time Temperature Celsius
56 2015-06-07 19:56 50 ******************************
```

```
...
62 2015-06-07 20:02 55 **********************************
63 2015-06-07 20:03 55 **********************************
64 2015-06-07 20:04 51 ******************************
...
55 2015-06-07 22:03 46 ************************
```

Если диск начинает перегреваться во время получения данных, примите меры для снижения температуры. В качестве немедленного действия временно приостановите процесс и продолжите, когда диск остынет. В зависимости от применяемого метода это может быть простая отправка сигнала процессу Linux посредством **CTRL-Z** или команды `kill -SIGTSTP` с последующим идентификатором процесса. Когда температура снизится до приемлемого уровня, получение можно возобновить с того же места, где оно было приостановлено.

Такое приостановление и возобновление процесса не должно влиять на криминалистическую корректность получения данных. Процесс приостанавливается с сохранением рабочего состояния: текущего сектора, целевого файла, переменных среды и так далее. Пример приостановления и возобновления процесса создания образа из оболочки нажатием **CTRL-Z** выглядит следующим образом:

```
# dcfldd if=/dev/sdb of=./image.raw
39424 blocks (1232Mb) written.^Z
[1]+ Stopped dcfldd if=/dev/sdb of=./image.raw
# fg
dcfldd if=/dev/sdb of=./image.raw
53760 blocks (1680Mb) written.
...
```

Здесь выполняющаяся команда `dcfldd` приостанавливается нажатием **CTRL-Z**. Возобновите процесс командой `fg`, то есть переведите его на передний план. Процесс можно продолжить и командой `kill -SIGCONT`. Дополнительные сведения об управлении заданиями и сигналах см. в документации Bash и на странице руководства `SIGNAL(7)`.

Наблюдение за температурой и предупреждения можно автоматизировать средствами Nagios, Icinga и других систем мониторинга инфраструктуры. Такие системы следят за различными параметрами среды и выдают предупреждения при приближении к критическим порогам или их превышении.

Во многих криминалистических лабораториях во время создания образов применяют радиаторы или охладители дисков, чтобы уменьшить перегрев исследуемых накопителей. Это рекомендуется при длительных сеансах получения данных, особенно со старыми дисками.

Попытка использовать для снижения температуры некоторые методы управления питанием почти ничего не даст. Они останавливают вращение диска после периода бездействия, однако при непрерывном создании образа времени простоя практически нет.

Организация криминалистической защиты от записи

Основополагающая составляющая сбора цифровых доказательств - криминалистически корректное получение данных с носителя. Частично достичь этой цели[4] можно, обеспечив механизм блокировки записи до подключения диска к основной системе криминалистического получения данных.

Когда диск подключают к персональному компьютеру с современной операционной системой, автоматические процессы значительно повышают риск изменения данных, а следовательно, уничтожения доказательств. Автоматические попытки смонтировать разделы, сформировать миниатюры для графических файловых диспетчеров, проиндексировать содержимое для

локальных поисковых баз, выполнить антивирусное сканирование и другие действия создают риск изменения подключённого накопителя. Временные метки могут обновиться, уничтожив потенциальные доказательства. Удалённые файлы в нераспределённых частях диска могут быть перезаписаны, что также уничтожит доказательства. Обнаруженные вредоносные программы или вирусы - именно те доказательства, которые может искать следователь, - способны быть удалены. Журнальная файловая система может записать на диск изменения, ожидающие в журнале. Возможны попытки восстановить повреждённую файловую систему или собрать и синхронизировать компоненты RAID.

Помимо потенциального автоматического уничтожения доказательств, значительный риск создаёт человеческая ошибка. Люди могут случайно скопировать или удалить файлы, просматривать файловую систему с обновлением временных меток последнего доступа либо ошибочно выбрать не то устройство и выполнить разрушительное действие.

Блокираторы записи разработаны для защиты носителей от нежелательного изменения данных. Обязательное применение блокираторов в стандартных процессах и процедурах криминалистической лаборатории демонстрирует должную осмотрительность. Оно соответствует передовой отраслевой практике обращения с носителями как с доказательствами в цифровой криминалистике. Блокираторы записи гарантируют подключение носителя к рабочей станции эксперта только для чтения.

Программа NIST Computer Forensic Tool Testing (CFTT) устанавливает формальные требования к блокираторам записи. Спецификация устройства аппаратной блокировки записи *Hardware Write Block (HWB) Device Specification, Version 2.0* доступна по адресу http://www.cftt.nist.gov/hardware_write_block.htm. В ней определены следующие требования верхнего уровня:

• устройство HWB не должно передавать защищённому устройству хранения команду, изменяющую находящиеся на нём данные;
• устройство HWB должно возвращать данные, запрошенные операцией чтения;
• устройство HWB должно без изменений возвращать любые запрошенные у накопителя сведения, имеющие значение для доступа;
• любое состояние ошибки, о котором устройство хранения сообщило HWB, должно передаваться основной системе.

Доступны аппаратные и программные блокираторы записи: автономное оборудование, устанавливаемые программные пакеты и загрузочные криминалистические компакт-диски. Иногда сам носитель может иметь встроенный режим только для чтения.

Аппаратные блокираторы записи

Предпочтительный метод блокировки записи использует аппаратные устройства, расположенные между исследуемым диском и рабочей станцией эксперта. Аппаратный блокиратор перехватывает отправленные диску команды, которые могли бы изменить данные. Переносное устройство блокировки записи, защищающее накопитель SATA, - Tableau компании Guidance Software - показано на рис. 4-1.

Рис. 4-1. Переносной блокиратор записи SATA

У аппаратного блокиратора обычно есть переключатель или светодиод, показывающий, действует ли функция блокировки записи. На рис. 4-2 показано многофункциональное устройство Tableau компании Guidance Software, предназначенное для установки непосредственно в рабочую станцию эксперта. Оно способно защищать накопители SATA, SAS, IDE, FireWire и USB.

Рис. 4-2. Многофункциональный блокиратор записи в отсеке накопителя

Блокираторы записи могут предоставлять основной системе получения данных сведения о состоянии. Примером служит инструмент `tableau-parm` (https://github.com/ecbftw/tableau-parm/), который способен запрашивать сведения у аппаратного блокиратора Tableau. Это открытое средство позволяет проверить состояние блокировки записи диска, подключённого через Tableau. Например:

```
$ sudo tableau-parm /dev/sdg
WARN: Requested 255 bytes but got 152 bytes)
## Bridge Information ##
chan_index: 0x00
chan_type: SATA
writes_permitted: FALSE
declare_write_blocked: TRUE
declare_write_errors: TRUE
bridge_serial: 000ECC550035F055
bridge_vendor: Tableau
bridge_model: T35u-R2
firmware_date: May 23 2014
firmware_time: 09:43:37
## Drive Information ##
drive_vendor: %00%00%00%00%00%00%00%00
drive_model: INTEL SSDSA2CW300G3
drive_serial: CVPR124600ET300EGN
drive_revision: 4PC10302
## Drive HPA/DCO/Security Information ##
security_in_use: FALSE
security_support: TRUE
hpa_in_use: FALSE
```

```
hpa_support: TRUE
dco_in_use: FALSE
dco_support: TRUE
drive_capacity: 586072368
hpa_capacity: 586072368
dco_capacity: 586072368
```

Согласно документации Tableau, поле `drive_vendor` для некоторых накопителей может не содержать никаких сведений.[5]

На заключительном этапе редактирования книги на рынке появились первые блокираторы записи PCI Express. Ниже показан пример от Tableau. При подключении накопителя NVME через блокиратор записи PCI Express вывод `dmesg` выглядит следующим образом:

```
[194238.882053] usb 2-6: new SuperSpeed USB device number 5 using xhci_hcd
[194238.898642] usb 2-6: New USB device found, idVendor=13d7, idProduct=001e
[194238.898650] usb 2-6: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[194238.898654] usb 2-6: Product: T356789u
[194238.898658] usb 2-6: Manufacturer: Tableau
[194238.898662] usb 2-6: SerialNumber: 0xecc3500671076
[194238.899830] usb-storage 2-6:1.0: USB Mass Storage device detected
[194238.901608] scsi host7: usb-storage 2-6:1.0
[194239.902816] scsi 7:0:0:0: Direct-Access NVMe INTEL SSDPEDMW40 0174
PQ: 0 ANSI: 6
[194239.903611] sd 7:0:0:0: Attached scsi generic sg2 type 0
[194240.013810] sd 7:0:0:0: [sdc] 781422768 512-byte logical blocks: (400 GB/
373 GiB)
[194240.123456] sd 7:0:0:0: [sdc] Write Protect is on
[194240.123466] sd 7:0:0:0: [sdc] Mode Sense: 17 00 80 00
[194240.233497] sd 7:0:0:0: [sdc] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
[194240.454298] sdc: sdc1
[194240.673411] sd 7:0:0:0: [sdc] Attached SCSI disk
```

Блокиратор работает как мост USB3 и предоставляет накопитель NVME как устройство SCSI. Этот конкретный блокиратор поддерживает накопители PCI Express со стандартами AHCI и NVME. Поддерживаются обычные разъёмы PCI Express (рис. 4-3) и M.2 (рис. 4-4). Для подключения накопителей NVME U.2 (SFF-8639) можно применять стандартные переходники mini-SAS - PCI Express или M.2. Блокираторы записи PCI с поддержкой NVME выпускает и Wiebetech.

Главное преимущество аппаратных блокираторов - независимость от операционной системы. Они работают прозрачно и отдельно от основной системы получения данных, устраняя необходимость сопровождать драйверы или совместимость с операционной системой. Благодаря этому они идеально подходят для среды получения данных на основе Linux.

Рис. 4-3. Док-станция с блокиратором записи для накопителей в разъёмах PCI Express

Рис. 4-4. Многофункциональный блокиратор записи и док-станция для накопителей PCI Express M.2

Особая благодарность швейцарской компании Arina AG за предоставленное оборудование блокировки записи, использованное для испытаний в этой книге.

Программные блокираторы записи

У программных блокираторов записи несколько противоречивая история. С современными операционными системами их становится всё труднее разрабатывать и сопровождать. Обновления поставщика операционной системы, изменения конфигурации экспертом и дополнительно установленное программное обеспечение создают риск отключения, перезаписи, обхода или отказа реализованной программными средствами блокировки записи.

Программные блокираторы трудно реализовать. Простое монтирование диска только для чтения командой `mount -o ro` не гарантирует, что диск не будет изменён. Свойство «только для чтения» в данном контексте относится к файловой системе, а не к дисковому устройству. Ядро всё ещё может выполнять запись на диск по разным причинам. Программную блокировку необходимо реализовать в ядре ниже уровня виртуальной файловой системы и даже ниже других драйверов устройств, реализующих конкретный интерфейс накопителя, например AHCI. В Linux применялось несколько низкоуровневых программных методов блокировки записи, но их успех был ограниченным.

Такие инструменты, как `hdparm` и `blockdev`, могут перевести диск в режим только для чтения, установив флаг ядра. Например:

```
# hdparm -r1 /dev/sdk

/dev/sdk:
 setting readonly to 1 (on)
 readonly      = 1 (on)
```

Тот же флаг можно установить посредством `blockdev`:

```
# blockdev --setro /dev/sdk
```

Метод установки флагов ядра зависит от правильной настройки `udev`, которая должна переводить вновь подключённые накопители в режим только для чтения до того, как любой другой процесс успеет их изменить.

Для специальной реализации функции криминалистической блокировки записи было создано и исправление ядра. Дополнительные сведения доступны по адресу https://github.com/msuhanov/Linux-write-blocker/. Несколько криминалистических загрузочных компакт-дисков используют исправление ядра для блокировки записи, созданное Максимом Сухановым. Следующий вспомогательный сценарий управляет программной блокировкой записи на криминалистическом загрузочном диске DEFT Linux:

```
% cat /usr/sbin/wrtblk
#!/bin/sh

# Пометить указанное блочное устройство как доступное только для чтения
[ $# -eq 1 ] || exit
[ ! -z "$1" ] || exit
bdev="$1"
[ -b "/dev/$bdev" ] || exit
[ ! -z $bdev##loop*$ ] || exit
blockdev --setro "/dev/$bdev" || logger "wrtblk: blockdev --setro /dev/$bdev
failed!"

# Пометить родительское блочное устройство как доступное только для чтения
syspath=$(echo /sys/block/*/"$bdev")
```

```
[ "$syspath" = "/sys/block/*/$bdev" ] && exit
dir=$syspath%/*$
parent=$dir##*/$
[ -b "/dev/$parent" ] || exit
blockdev --setro "/dev/$parent" || logger "wrtblk: blockdev --setro /dev/$parent
failed!"
```

Исправление реализовано в ядре и включается и отключается вспомогательными сценариями. Эти сценарии просто используют команду `blockdev`, чтобы пометить устройство как доступное только для чтения.

NIST CFTT выполнил испытания программных средств блокировки записи; результаты доступны по адресу http://www.cftt.nist.gov/software_write_block.htm.

Аппаратные блокираторы записи остаются самым безопасным и рекомендуемым способом защиты носителей во время криминалистического получения данных.

Криминалистические загрузочные диски Linux

Необходимость реагировать на инциденты и выполнять сортировку на месте привела к разработке загрузочных дисков Linux с программами, необходимыми для таких задач. С этих дисков можно загрузить исследуемый компьютер и обращаться к локально подключённому хранилищу посредством различных криминалистических инструментов. Криминалистические загрузочные диски проектируются так, чтобы защищать обнаруженные носители от записи, если потребуется получить их криминалистический образ. Подключённый диск можно сделать записываемым командой наподобие `wrtblk` из предыдущего примера; это полезно при создании образа после подключения внешнего целевого диска. Криминалистические загрузочные диски обладают и сетевыми функциями, позволяя выполнять удалённый анализ и получение данных.

Криминалистические загрузочные диски полезны в следующих случаях:

• персональный компьютер исследуется без вскрытия для извлечения диска;
• блокиратор записи недоступен;
• при сортировке нужно быстро проверить компьютеры на наличие определённого доказательства, прежде чем решать, создавать ли образ;
• необходимы инструменты на основе Linux, например Sleuth Kit или Foremost, которые иначе недоступны;
• техническому специалисту-криминалисту необходимо удалённо работать через `ssh`.

К числу популярных криминалистических загрузочных дисков, которые сейчас продолжают сопровождаться, относятся:

• Kali Linux, ранее BackTrack, на основе Debian: https://www.kali.org/;
• Digital Evidence & Forensics Toolkit (DEFT) на основе Ubuntu Linux: http://www.deftlinux.net/;

• Pentoo, криминалистический диск на основе Gentoo Linux: http://pentoo.ch/;
• C.A.I.N.E, Computer Forensics Linux Live Distro, на основе Ubuntu Linux: http://www.caine-live.net/.

Сопровождение и тестирование криминалистических загрузочных дисков требует большой работы. Ранее существовало множество других таких дисков. Поскольку их состав постоянно меняется, обязательно исследуйте вопрос и используйте последние работоспособные и сопровождаемые версии.

Носители с физическим режимом только для чтения

Некоторые носители имеют механизм защиты от записи, полезный в криминалистическом контексте. Например, у большинства лент есть ползунок или язычок, который предписывает ленточному накопителю считать их доступными только для чтения, как показано слева на рис. 4-5. У ленты LTO-5 внизу слева закрытый язычок означает защиту от записи, а у ленты DAT160 вверху слева защиту означает открытый язычок.

У карт памяти SD имеется переключатель блокировки записи, показанный справа на рис. 4-5.

Рис. 4-5. Язычки защиты от записи на лентах и картах SD

У старых флеш-накопителей USB может иметься переключатель защиты от записи. У некоторых очень старых жёстких дисков IDE есть перемычка, которую можно установить так, чтобы электроника накопителя считала его доступным только для чтения.

Для CD-ROM, DVD и дисков Blu-ray блокиратор записи не нужен, поскольку по умолчанию они предназначены только для чтения. Простое обращение к перезаписываемому диску не изменит временные метки или другие данные на нём: изменения на такие оптические носители необходимо явно записывать.

Заключительные мысли

В этой главе вы узнали, как организовать базовый аудит, регистрацию действий и управление задачами. Я рассмотрел такие темы, как правила именования и масштабируемые структуры каталогов, а также различные трудности, связанные с размерами образов, планированием ёмкости накопителей, производительностью и условиями окружающей среды. Наконец, обсуждалась критически важная составляющая криминалистической блокировки записи. Теперь вы готовы подключить исследуемый накопитель к основной системе получения данных для подготовки к выполнению криминалистического процесса.

Сноски

[1] [назад] Точка в этом примере может интерпретироваться как регулярное выражение. Здесь ради простоты это не учитывается.
[2] [назад] Команда GNU `cp` также позволяет создавать разреженные файлы при копировании.
[3] [назад] Проверено на типичном компьютере с процессором i7 и двумя дисками SATA3 посредством `dd`.
[4] [назад] Криминалистически корректное получение данных также охватывает полноту данных и сохранение целостности.
[5] [назад] “Tableau Bridge Query-Technical Documentation,” accessed 8 December 2005, previously available for download. Contact Guidance Software for more information.

Глава 5. Подключение исследуемого носителя к основной системе получения данных

В этой главе рассматриваются физическое подключение исследуемого носителя к основной системе исследования, идентификация исследуемого устройства в системе и запрос сведений у его микропрограммы. Вы также узнаете о методах удаления HPA и DCO, снятия паролей ATA и расшифровки самошифрующихся накопителей. Завершается глава несколькими специальными темами хранения данных. Начнём с осмотра оборудования исследуемого компьютера.

Осмотр оборудования исследуемого компьютера

Когда персональный компьютер или ноутбук изымают на месте либо доставляют в криминалистическую лабораторию для исследования, осмотр не должен ограничиваться внутренними дисками. Необходимо полностью проверить конфигурацию оборудования компьютера, настройки BIOS, аппаратные часы и другие параметры.

> **Примечание.** Эта книга рассматривает получение данных с «мёртвых» дисков, то есть уже выключенных накопителей и компьютеров. В зависимости от организации при прибытии на место преступления или инцидента, где работают включённые машины, применяется процедура сортировки. Она может включать фотографирование экранов, использование устройств, имитирующих движение мыши, чтобы предотвратить включение защищённых паролем экранных заставок, или запуск инструментов создания дампа памяти. Первичная сортировка работающих компьютеров находится за рамками книги.

Физический осмотр компьютера и извлечение диска

Прежде чем отсоединять кабели накопителей или выкручивать диски из отсеков, сфотографируйте исследуемый компьютер, чтобы документировать конфигурацию оборудования, число дисков и способ их подключения к системной плате.

Извлекайте диски осторожно, особенно из старых компьютеров, которые могли не вскрываться много лет. Сфотографируйте верхнюю сторону каждого накопителя, чтобы запечатлеть серийный номер и другие сведения на этикетке. Для каждого диска отметьте место подключения кабеля на системной плате. Если на плате имеется несколько портов SATA, запишите, каким портом пользовался каждый диск.

Откройте лотки оптических приводов и убедитесь, что в них нет дисков. У большинства оптических приводов есть небольшое отверстие, позволяющее вручную освободить дверцу без включения питания.

Проверьте разъёмы PCI на наличие накопителей PCI SATA Express или PCI NVME. Если на системной плате есть разъём M.2 либо mSATA, проверьте наличие плат SSD.

Проверка оборудования исследуемого компьютера

После извлечения всех накопителей из корпуса исследуемого компьютера включите системную плату и запишите конфигурацию BIOS, показания часов, порядок загрузки, возможные журналы BIOS, версию и другие сведения.

Если требуются дополнительные сведения об исследуемом компьютере, изучите его посредством криминалистического загрузочного диска с инструментами анализа оборудования: `lshw`, `dmidecode`, `biosdecode`, `lspci` и другими.

Некоторые сведения, специфичные для изготовителя, удастся получить специальными инструментами поставщика, например `vpddecode` для оборудования IBM и Lenovo или `ownership` для тегов владения оборудованием Compaq.

Осмотрите и документируйте также любые дополнительные компоненты оборудования, например модули памяти и платы PCI.

Подключение исследуемого диска к основной системе получения данных

После физического подключения исследуемого накопителя к рабочей станции эксперта с применением механизма блокировки записи необходимо определить правильное блочное устройство, связанное с этим накопителем. Чтобы надёжно идентифицировать исследуемый диск в основной системе получения данных, перечислите устройства хранения, подтвердите уникальные идентификаторы физического накопителя и определите соответствующий файл устройства в `/dev`. В этом разделе действия рассматриваются подробнее.

Просмотр оборудования основной системы получения данных

Понимание конфигурации оборудования основной системы исследования полезно для настройки производительности, планирования ёмкости, поддержания стабильной платформы, устранения неполадок, локализации отказов и снижения риска человеческой ошибки. В этом разделе приведены примеры инструментов для перечисления и просмотра оборудования персонального компьютера.

Инструмент `lshw` позволяет быстро получить обзор оборудования рабочей станции эксперта:

```
# lshw -businfo
```

Сведения о шинах описывают адреса конкретных устройств, например `pci@domain:bus:slot.function`, `scsi@host.channel.target.lun` и `usb@bus:device`.

С помощью `lshw` можно искать и подключённые устройства определённого типа. Например:

```
# lshw -businfo -class storage
Bus info Device Class Description
=====================================================
...
usb@2:5.2 scsi22 storage Forensic SATA/IDE Bridge
...
# lshw -businfo -class disk
Bus info Device Class Description
=====================================================
...
scsi@22:0.0.0 /dev/sdp disk 120GB SSD 850
...
```

Обратите внимание, что `scsi22` связан с `scsi@22:.0.0.0`, а тот - с `/dev/sdp`. Определение файла устройства Linux для подключённого физического накопителя подробнее рассматривается в следующих разделах.

Если исследуемый накопитель подключён снаружи, скорее всего, применяется USB, Thunderbolt, FireWire или eSATA, а в редких случаях - Fibre Channel.

При внутреннем подключении, вероятнее всего, используется кабель SATA, разъём PCI Express, интерфейс M.2 или кабель SAS либо, возможно, устаревший интерфейс вроде параллельного SCSI или IDE.

Устройства, подключённые к шине PCI, включая параллельную PCI и PCI Express, можно перечислить инструментом `lspci`:

```
# lspci
```

Шина PCI распределяет устройства по классам; дополнительные сведения об идентификаторах и классах PCI доступны по адресу http://pci-ids.ucw.cz/. Интерес представляют устройства класса контроллеров массовой памяти с идентификатором 01, поскольку они управляют подключёнными носителями.

Новые версии `lspci`, начиная с `pciutils` 3.30, умеют перечислять шину PCI по классам устройств, что помогает выделить интересующее оборудование. Следующая команда перечисляет все контроллеры массовой памяти SATA с идентификатором класса 01 и подкласса 06:

```
# lspci -d ::0106
```

Эта команда перечисляет все контроллеры массовой памяти SCSI, IDE, RAID, ATA, SATA, SAS и NVME в системе:

```
# for i in 00 01 04 05 06 07 08; do lspci -d ::01$i; done
```

Другой класс PCI, способный управлять подключёнными носителями, - контроллеры последовательной шины с идентификатором 0C. Следующая команда перечисляет все контроллеры последовательной шины USB с идентификатором класса 0C и подкласса 03:

```
# lspci -d ::0C03
```

Следующая команда перечисляет все контроллеры последовательных шин FireWire, USB и Fibre Channel в основной системе эксперта:

```
# for i in 00 03 04; do lspci -d ::0C$i; done
```

Если исследуемый накопитель подключён по USB, он не появится на шине PCI. Устройства USB можно отдельно перечислить командой `lsusb`. Без параметров она выводит список всех подключённых USB-устройств:

```
# lsusb
...
Bus 001 Device 005: ID 0951:1665 Kingston Technology
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Здесь флеш-накопитель USB подключён к шине USB 1 и получил идентификатор устройства USB 5. Команда `lsusb -v` выведет о нём более подробные сведения.[1]

Приведённые инструменты и примеры дают обзор контроллеров носителей и оборудования, подключённого к рабочей станции эксперта. Дополнительные параметры и возможности описаны на страницах руководства `lshw(1)`, `lspci(8)` и `lsusb(8)`; они позволяют просматривать оборудование подробнее.

Идентификация исследуемого накопителя

Понимание оборудования рабочей станции эксперта, особенно доступных шин и контроллеров, помогает установить место подключения исследуемого диска. Следующий шаг - однозначно подтвердить его личность по отличительным сведениям: серийному номеру, уникальному номеру модели или другому уникальному свойству.

Идентифицировать исследуемое устройство можно несколькими способами. Если диск подключён через шину USB и показан инструментом `lsusb`, дополнительные сведения можно получить, указав идентификатор `vendor:productID` исследуемого диска:

```
# lsusb -vd 0781:5583
Bus 004 Device 002: ID 0781:5583 SanDisk Corp.
...
idVendor 0x0781 SanDisk Corp.
idProduct 0x5583
bcdDevice 1.00
iManufacturer 1 SanDisk
iProduct 2 Ultra Fit
iSerial 3 4C530001200627113025
...
wSpeedsSupported 0x000e
Device can operate at Full Speed (12Mbps)
Device can operate at High Speed (480Mbps)
Device can operate at SuperSpeed (5Gbps)
...
```

Уникальные сведения из этого вывода - серийный номер и другие данные - позволяют подтвердить, что подключённое устройство является исследуемым накопителем. Если серийный номер или иные уникальные свойства совпадают с физически подключённым диском, правильное устройство идентифицировано.

Почти ко всем накопителям можно обращаться командами SCSI; заметное исключение - непосредственно подключённые диски NVME. Для опроса подключённого устройства хранения можно использовать инструмент `lsscsi`. Он поддерживает множество протоколов транспортного уровня, включая SATA, USB, SAS, FireWire, ATA, SCSI, Fibre Channel и другие. `lsscsi` также полезен для сопоставления путей устройств ядра с файлами в `/dev`:

```
# lsscsi -v
...
[6:0:0:0] disk ATA INTEL SSDSA2CW30 0302 /dev/sda
dir: /sys/bus/scsi/devices/6:0:0:0 [/sys/devices/pci0000:00/0000:00:1f.2/ata7/
host6/target6:0:0/6:0:0:0]
...
```

Ядро выводит информационное сообщение при подключении устройств к основной системе или их отключении от нее. Эти сообщения находятся в кольцевом буфере ядра, который просматривают инструментом `dmesg`. При запуске `dmesg` с флагом `-T` выводятся удобочитаемые временные метки; они полезны, когда нужно определить, какое устройство было добавлено в известный момент времени:

```
# dmesg -T
...
[Sun May 15 13:44:45 2016] usb 2-1: new SuperSpeed USB device number 9 using
xhci_hcd
[Sun May 15 13:44:45 2016] usb 2-1: New USB device found, idVendor=0781,
idProduct=5583
[Sun May 15 13:44:45 2016] usb 2-1: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[Sun May 15 13:44:45 2016] usb 2-1: Product: Ultra Fit
[Sun May 15 13:44:45 2016] usb 2-1: Manufacturer: SanDisk
[Sun May 15 13:44:45 2016] usb 2-1: SerialNumber: 4C530001141203113173
[Sun May 15 13:44:45 2016] usb-storage 2-1:1.0: USB Mass Storage device detected
[Sun May 15 13:44:45 2016] scsi host24: usb-storage 2-1:1.0
[Sun May 15 13:44:46 2016] scsi 24:0:0:0: Direct-Access SanDisk Ultra Fit
1.00 PQ: 0 ANSI: 6
[Sun May 15 13:44:46 2016] sd 24:0:0:0: Attached scsi generic sg5 type 0
[Sun May 15 13:44:46 2016] sd 24:0:0:0: [sdf] 30375936 512-byte logical blocks:
(15.6 GB/14.5 GiB)
```

```
[Sun May 15 13:44:46 2016] sd 24:0:0:0: [sdf] Write Protect is off
[Sun May 15 13:44:46 2016] sd 24:0:0:0: [sdf] Mode Sense: 43 00 00 00
[Sun May 15 13:44:46 2016] sd 24:0:0:0: [sdf] Write cache: disabled, read cache:
enabled, doesn't support DPO or FUA
[Sun May 15 13:44:46 2016] sdf: sdf1
[Sun May 15 13:44:46 2016] sd 24:0:0:0: [sdf] Attached SCSI removable disk
```

Этот вывод позволяет идентифицировать подключенное физическое устройство, связав USB-устройство с идентификатором узла SCSI и именем блочного устройства. В данном примере `usb 2-1:` обозначает шину 2 и физический порт 1 (разъем). USB-накопителю присвоен номер устройства 9, и он использует драйвер `xhci_hcd` (поддерживающий USB3). Выводятся строки идентификаторов поставщика и продукта `idVendor=0781`, `idProduct=5583`, а за ними - информационные строки с производителем, продуктом и серийным номером (они могут отличаться от `idVendor` и `idProduct`). Драйвер `usb-storage` для транспорта Bulk-Only обнаруживает устройство (для устройств UASP он не нужен), а `scsi host24:` показывает, что устройству присвоен номер узла SCSI, соответствующий адресу SCSI `24:0:0:0:`. Создаются два устройства: `sg5` (универсальное SCSI) и `sdf` (блочное устройство), соответствующие `/dev/sg5` и `/dev/sdf`. Запрашиваются некоторые сведения об уже установленном SCSI-устройстве и обнаруживаются таблицы разделов (`sdf1`).

Более простая команда для перечисления всех подключенных устройств хранения, включая описательные сведения и пути устройств, - `lsblk`. Новые версии `lsblk` предоставляют параметры вывода для поставщика, модели, редакции, серийного номера и номера WWN (World Wide Name; https://en.wikipedia.org/wiki/World_Wide_Name). Кроме того, `lsblk` предоставляет полезные технические сведения: имя и размер устройства, физический и логический размеры сектора, транспорт (USB, SATA, SAS и так далее), адрес SCSI и многое другое:

```
# lsblk -pd -o TRAN,NAME,SERIAL,VENDOR,MODEL,REV,WWN,SIZE,HCTL,SUBSYSTEMS,HCTL
```

Большинство показанных здесь инструментов просто читает различные файлы и каталоги из каталога Linux `/proc`. Дополнительные сведения о подключенных накопителях и других структурах ядра находятся в дереве `/proc`. Более подробное описание файловой системы `proc` приведено на странице руководства `proc(5)`.

Опрос исследуемого диска для получения сведений

После подключения исследуемого диска к рабочей станции эксперта и однозначного определения правильного устройства Linux, с которым предстоит работать, можно собрать дополнительные метаданные об устройстве. У самого устройства можно запросить сведения о диске, микропрограмме, данные SMART и другие параметры конфигурации.

Для запроса сведений, хранящихся на жестком диске, доступен ряд инструментов. Как правило, доступ к этим сведениям микропрограммы получают с помощью низкоуровневых команд интерфейса ATA или SCSI, непосредственно взаимодействующих с электроникой накопителя.

Документирование идентификационных данных устройства

К этому моменту у вас должен быть ряд подробностей и технических идентификаторов подключенного к основной системе эксперта диска, включая следующие:

• Поставщик, марка и модель
• Серийный номер или WWN
• Имя устройства Linux
• Домен PCI:шина:слот.функция
• PCI `vendorID:deviceID`
• USB шина:устройство
• USB `vendorID:productID`
• SCSI узел:канал:цель:lun

Эти сведения можно сохранить для отчетности, перенаправив вывод различных команд инструментов в текстовые файлы.

Задокументируйте доказательство применения блокиратора записи. Если используется аппаратный блокиратор записи, например Tableau, опросите его и сохраните результаты:

```
# tableau-parm /dev/sdc > write-blocked.txt
```

Здесь `/dev/sdc` следует заменить соответствующим устройством исследуемого диска.

Если используется программный блокиратор записи, например `wrtblk`, запросите у `blockdev` отчет о текущем состоянии устройства (включая флаг "только чтение"):

```
# blockdev --report /dev/sda > wrtblk.txt
```

Здесь `/dev/sda` следует заменить соответствующим устройством исследуемого диска.

Если исследуемый диск подключен по USB, его можно указать либо как `шина:устройство` (с помощью `-s`), либо как `поставщик:продукт` (с помощью `-d`). Следующие две команды создадут и сохранят один и тот же подробный вывод:

```
# lsusb -v -s 2:2 > lsusb.txt
# lsusb -v -d 13fe:5200 > lsusb.txt
```

Здесь `2:2` и `13fe:5200` следует заменить соответствующими значениями исследуемого диска в основной системе получения данных.

Команде `lsblk` можно передать устройство Linux, а флаг `-O` выведет все доступные столбцы:

```
# lsblk -O /dev/sda > lsblk.txt
```

Здесь `/dev/sda` следует заменить соответствующим устройством исследуемого диска в основной системе получения данных.

Команда `lsscsi` также позволяет сохранить определенное представление подключенного диска, если указать используемый адрес SCSI:

```
# lsscsi -vtg -L 16:0:0:0 > lsscsi.txt
```

Здесь `16:0:0:0` следует заменить соответствующим адресом SCSI исследуемого диска в основной системе получения данных.

При желании соответствующий вывод `dmesg` также можно скопировать в текстовый файл.

Показанные в этом разделе примеры иллюстрируют сохранение вывода команд для конкретного исследуемого диска. Ради краткости в последующих главах примеры сохранения данных в файлы иногда не приводятся, а основное внимание уделяется построению команд.

Опрос возможностей и функций диска с помощью hdparm

Многие из рассмотренных ранее инструментов (`lsusb`, `lspci`, `lsblk` и другие) запрашивали сведения у системы Linux и структур ядра. Однако дополнительные сведения можно запросить непосредственно у накопителя. Инструмент `hdparm` полезен для отправки команд большинству дисков, подключенных к системе Linux.

Инструмент `hdparm` работает, отправляя запросы драйверам дисков операционной системы (с помощью `ioctl`) для получения сведений о диске. С точки зрения криминалистики интересными или полезными для документирования могут быть следующие элементы:

• Подробные сведения о геометрии диска (физической и логической)
• Поддерживаемые диском стандарты, функции и возможности
• Состояния и флаги, относящиеся к конфигурации диска
• Сведения DCO и HPA
• Сведения о безопасности
• Сведения поставщика: марка, модель и серийный номер
• Идентификатор устройства WWN (если он существует)
• Время, необходимое для безопасного стирания (для большинства дисков оно приблизительно равно времени получения образа)

Более подробные сведения о возможностях `hdparm` приведены на странице руководства `hdparm(8)`.

Следующий пример показывает, как получить с помощью `hdparm` обзор диска, применив флаг `-I` вместе с необработанным дисковым устройством. Листинг снабжен комментариями, существенными для экспертов-криминалистов.

Вывод начинается с документирования сведений о накопителе, включая производителя, модель, серийный номер и стандарты, которым он соответствует. В выводе также присутствуют различные параметры накопителя: физический и логический размеры сектора, число секторов, форм-фактор и другие физические свойства.

```
# hdparm -I /dev/sda

/dev/sda:

ATA device, with non-removable media
Model Number: WDC WD20EZRX-00D8PB0
Serial Number: WD-WCC4NDA2N98P
Firmware Revision: 80.00A80
Transport: Serial, SATA 1.0a, SATA II Extensions, SATA Rev 2.5,
SATA Rev 2.6, SATA Rev 3.0
Standards:
Supported: 9 8 7 6 5
Likely used: 9
Configuration:
Logical max current
cylinders 16383 16383
heads 16 16
sectors/track 63 63
--
CHS current addressable sectors: 16514064
LBA user addressable sectors: 268435455
```

```
LBA48 user addressable sectors: 3907029168
Logical Sector size: 512 bytes
Physical Sector size: 4096 bytes
device size with M = 1024*1024: 1907729 MBytes
device size with M = 1000*1000: 2000398 MBytes (2000 GB)
cache/buffer size = unknown
Nominal Media Rotation Rate: 5400
Capabilities:
LBA, IORDY(can be disabled)
Queue depth: 32
Standby timer values: spec'd by Standard, with device specific minimum
R/W multiple sector transfer: Max = 16 Current = 16
DMA: mdma0 mdma1 mdma2 udma0 udma1 udma2 udma3 udma4 udma5*udma6
Cycle time: min=120ns recommended=120ns
PIO: pio0 pio1 pio2 pio3 pio4
Cycle time: no flow control=120ns IORDY flow control=120ns
...
```

Следующий раздел вывода описывает доступные на накопителе функции, а звездочка (*) указывает, включена ли функция в данный момент. (Для понимания функций, специфичных для поставщика, может потребоваться дополнительная закрытая документация.) Это полезно при подготовке к получению криминалистического образа, поскольку указывает состояние наборов функций безопасности и других возможностей, таких как DCO (набор функций Device Configuration Overlay).

```
...
Commands/features:
Enabled Supported:
* SMART feature set
Security Mode feature set
* Power Management feature set
* Write cache
* Look-ahead
* Host Protected Area feature set
* WRITE_BUFFER command
* READ_BUFFER command
* NOP cmd
* DOWNLOAD_MICROCODE
Power-Up In Standby feature set
* SET_FEATURES required to spinup after power up
SET_MAX security extension
* 48-bit Address feature set
* Device Configuration Overlay feature set
* Mandatory FLUSH_CACHE
* FLUSH_CACHE_EXT
* SMART error logging
* SMART self-test
* General Purpose Logging feature set

* 64-bit World wide name
* WRITE_UNCORRECTABLE_EXT command
* {READ,WRITE}_DMA_EXT_GPL commands
* Segmented DOWNLOAD_MICROCODE
* Gen1 signaling speed (1.5Gb/s)
* Gen2 signaling speed (3.0Gb/s)
* Gen3 signaling speed (6.0Gb/s)
* Native Command Queueing (NCQ)
* Host-initiated interface power management
* Phy event counters
* NCQ priority information
* READ_LOG_DMA_EXT equivalent to READ_LOG_EXT
* DMA Setup Auto-Activate optimization
Device-initiated interface power management
* Software settings preservation
* SMART Command Transport (SCT) feature set
* SCT Write Same (AC2)
```

```
* SCT Features Control (AC4)
* SCT Data Tables (AC5)
unknown 206[12] (vendor specific)
unknown 206[13] (vendor specific)
unknown 206[14] (vendor specific)
...
```

Следующий раздел вывода `hdparm` содержит дополнительные сведения о функциях безопасности, активных в данный момент. Они важны при определении того, заблокирован или зашифрован накопитель. Время, необходимое для безопасного стирания, также дает приблизительную оценку продолжительности получения образа (если узким местом по производительности является исследуемый накопитель).

```
...
Security:
Master password revision code = 65534
supported
not enabled
not locked
not frozen
not expired: security count
supported: enhanced erase
324min for SECURITY ERASE UNIT. 324min for ENHANCED SECURITY ERASE UNIT.
...
```

В заключительном разделе вывода `hdparm` снова отображается WWN, но на этот раз он разбит на NAA (который описывает остальную часть WWN), назначенный IEEE идентификатор поставщика OUI и оставшуюся часть WWN (уникальную для накопителя).

```
...
Logical Unit WWN Device Identifier: 50014ee25fcfe40c
NAA : 5
IEEE OUI : 0014ee
Unique ID : 25fcfe40c
Checksum: correct
```

Вывод `hdparm` содержит ряд элементов, представляющих интерес для экспертов-криминалистов: как для документирования, так и в качестве сведений для дальнейшего анализа. Чтобы включить полный вывод `hdparm -I` в криминалистический отчет, его можно перенаправить в текстовый файл.

Сходный инструмент для опроса дисков SCSI - `sdparm`, позволяющий обращаться к страницам режимов SCSI. Запуск `sdparm` с флагами `-a -l` получает подробный список параметров диска. Более краткий запрос `sdparm -i` позволяет извлечь Vital Product Data (VPD), которые содержат уникальные идентификационные сведения о марке, модели и серийном номере дисков SCSI и SAS.

Извлечение данных SMART с помощью smartctl

SMART был разработан в начале 1990-х годов для наблюдения за жесткими дисками и прогнозирования отказов. В 1995 году его включили в стандарт SCSI-3 (стандарт SCSI-3: X3T10/94-190 Rev 4), а в 1997 году - в стандарт ATA-3 (стандарт ATA-3: X3.298-1997). Поскольку некоторые подробности об аппаратной части диска могут представлять ценность для криминалистических исследований, в этом разделе вы познакомитесь с несколькими способами извлечения сведений SMART об аппаратной части диска.

Команда `smartctl` входит в пакет `smartmontools` и предоставляет доступ к интерфейсу SMART, встроенному почти во все современные жесткие диски. Команда `smartctl` опрашивает подключенное оборудование ATA, SATA, SAS и SCSI.

SMART предоставляет ряд переменных и статистических показателей диска, некоторые из которых могут представлять интерес для эксперта-криминалиста. Например:

• Статистика ошибок на диске и общего состояния диска
• Число включений питания диска
• Число часов работы диска
• Число прочитанных и записанных байтов (часто выражаемое в гигабайтах)
• Различные журналы SMART (история температуры и так далее)[2]

В следующем примере показаны данные SMART, запрошенные у накопителя. Листинг снабжен комментариями, существенными для экспертов-криминалистов.

Флаг `-x` предписывает `smartctl` вывести все доступные сведения. Первый блок вывода - раздел сведений, содержащий уникальные идентификационные сведения о накопителе. Большинство этих сведений можно получить и другими инструментами, например `hdparm`, как показано в предыдущих примерах.

```
# smartctl -x /dev/sda
smartctl 6.4 2014-10-07 r4002 [x86_64-linux-4.2.0-22-generic] (local build)
Copyright (C) 2002-14, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Model Family: Western Digital Green
Device Model: WDC WD20EZRX-00D8PB0
Serial Number: WD-WCC4NDA2N98P
LU WWN Device Id: 5 0014ee 25fcfe40c
Firmware Version: 80.00A80
User Capacity: 2,000,398,934,016 bytes [2.00 TB]
Sector Sizes: 512 bytes logical, 4096 bytes physical
Rotation Rate: 5400 rpm
Device is: In smartctl database [for details use: -P show]
ATA Version is: ACS-2 (minor revision not indicated)
SATA Version is: SATA 3.0, 6.0 Gb/s (current: 6.0 Gb/s)
Local Time is: Thu Jan 7 12:33:43 2016 CET
SMART support is: Available - device has SMART capability.
SMART support is: Enabled
AAM feature is: Unavailable
APM feature is: Unavailable
Rd look-ahead is: Enabled
Write cache is: Enabled
ATA Security is: Disabled, NOT FROZEN [SEC1]
Wt Cache Reorder: Enabled
...
```

Следующий раздел данных SMART показывает состояние накопителя и результаты самопроверок. Нездоровое состояние накопителя заранее предупреждает о возможных проблемах при получении образа. Затем перечисляются дополнительные возможности SMART.

```
...
=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED

General SMART Values:
Offline data collection status: (0x82) Offline data collection activity
was completed without error.
Auto Offline Data Collection: Enabled.
Self-test execution status: ( 0) The previous self-test routine completed
without error or no self-test has ever
been run.
Total time to complete Offline
data collection: (30480) seconds.
```

```
Offline data collection
capabilities: (0x7b) SMART execute Offline immediate.
Auto Offline data collection on/off support.
Suspend Offline collection upon new
command.
Offline surface scan supported.
Self-test supported.
Conveyance Self-test supported.
Selective Self-test supported.
SMART capabilities: (0x0003) Saves SMART data before entering
power-saving mode.
Supports SMART auto save timer.
Error logging capability: (0x01) Error logging supported.
General Purpose Logging supported.
Short self-test routine
recommended polling time: ( 2) minutes.
Extended self-test routine
recommended polling time: ( 307) minutes.
Conveyance self-test routine
recommended polling time: ( 5) minutes.
SCT capabilities: (0x7035) SCT Status supported.
SCT Feature Control supported.
SCT Data Table supported.
...
```

В следующем разделе приводятся дополнительные статистические сведения о накопителе. Возможный криминалистический интерес здесь представляет статистика истории использования накопителя: например, суммарное число часов, в течение которых на накопитель подавалось питание (`Power_On_Hours`), и число включений питания накопителя (`Power_Cycle_Count`). Оба атрибута могут соотноситься с ПК, из которого был извлечен накопитель. Общее число прочитанных и записанных адресов логических блоков (LBA) показывает использование объема накопителя в прошлом.

```
...
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME FLAGS VALUE WORST THRESH FAIL RAW_VALUE
1 Raw_Read_Error_Rate POSR-K 200 200 051 - 0
3 Spin_Up_Time POS--K 181 180 021 - 5908
4 Start_Stop_Count -O--CK 100 100 000 - 61
5 Reallocated_Sector_Ct PO--CK 200 200 140 - 0
7 Seek_Error_Rate -OSR-K 200 200 000 - 0
9 Power_On_Hours -O--CK 099 099 000 - 989
10 Spin_Retry_Count -O--CK 100 253 000 - 0
11 Calibration_Retry_Count -O--CK 100 253 000 - 0
12 Power_Cycle_Count -O--CK 100 100 000 - 59

192 Power-Off_Retract_Count -O--CK 200 200 000 - 33
193 Load_Cycle_Count -O--CK 199 199 000 - 3721
194 Temperature_Celsius -O---K 119 110 000 - 31
196 Reallocated_Event_Count -O--CK 200 200 000 - 0
197 Current_Pending_Sector -O--CK 200 200 000 - 4
198 Offline_Uncorrectable ----CK 200 200 000 - 4
199 UDMA_CRC_Error_Count -O--CK 200 200 000 - 0
200 Multi_Zone_Error_Rate ---R-- 200 200 000 - 4
||||||_ K auto-keep
|||||__ C event count
||||___ R error rate
|||____ S speed/performance
||_____ O updated online
|______ P prefailure warning
...
```

Следующий раздел - каталог журналов, описывающий журналы SMART, доступные на накопителе. Журналы включаются в вывод `smartctl -x`, а повторяющиеся записи удаляются ("пропускаются"). Некоторые из этих журналов могут представлять интерес при криминалистическом исследовании.

```
...
General Purpose Log Directory Version 1
SMART Log Directory Version 1 [multi-sector log support]
Address Access R/W Size Description
0x00 GPL,SL R/O 1 Log Directory
0x01 SL R/O 1 Summary SMART error log
0x02 SL R/O 5 Comprehensive SMART error log
0x03 GPL R/O 6 Ext. Comprehensive SMART error log
0x06 SL R/O 1 SMART self-test log
0x07 GPL R/O 1 Extended self-test log
0x09 SL R/W 1 Selective self-test log
0x10 GPL R/O 1 SATA NCQ Queued Error log
0x11 GPL R/O 1 SATA Phy Event Counters log
0x80-0x9f GPL,SL R/W 16 Host vendor specific log
0xa0-0xa7 GPL,SL VS 16 Device vendor specific log
0xa8-0xb7 GPL,SL VS 1 Device vendor specific log
0xbd GPL,SL VS 1 Device vendor specific log
0xc0 GPL,SL VS 1 Device vendor specific log
0xc1 GPL VS 93 Device vendor specific log
0xe0 GPL,SL R/W 1 SCT Command/Status
0xe1 GPL,SL R/W 1 SCT Data Transfer
...
```

Следующий раздел сведений журнала отображает результаты самопроверок. Неудачные самопроверки заранее предупреждают о возможных проблемах при получении образа.

```
...
SMART Extended Comprehensive Error Log Version: 1 (6 sectors)
No Errors Logged

SMART Extended Self-test Log Version: 1 (1 sectors)
Num Test_Description Status Remaining LifeTime(hours) LBA_of...
# 1 Short offline Completed without error 00% 0 -

SMART Selective self-test log data structure revision number 1
SPAN MIN_LBA MAX_LBA CURRENT_TEST_STATUS
1 0 0 Not_testing
2 0 0 Not_testing
3 0 0 Not_testing
4 0 0 Not_testing
5 0 0 Not_testing
Selective self-test flags (0x0):
After scanning selected spans, do NOT read-scan remainder of disk.
If Selective self-test is pending on power-up, resume after 0 minute delay.

SCT Status Version: 3
SCT Version (vendor specific): 258 (0x0102)
SCT Support Level: 1
Device State: Active (0)
...
```

Следующий блок вывода описывает статистику температуры накопителя. Эти сведения могут быть полезны для наблюдения в процессе получения образа. Для целей исследования интерес могут представлять минимальная и максимальная температуры, достигнутые за весь срок службы накопителя, если их можно соотнести с факторами окружающей среды, связанными с ПК подозреваемого. Данные SMART, специфичные для поставщика, не входят в общий стандарт SMART, и для их понимания может потребоваться дополнительная закрытая документация.

```
...
Current Temperature: 31 Celsius
Power Cycle Min/Max Temperature: 22/31 Celsius
Lifetime Min/Max Temperature: 20/41 Celsius
Under/Over Temperature Limit Count: 0/0
Vendor specific:
01 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
...
```

Некоторые накопители с поддержкой SMART ведут журнал истории температуры. Продолжительность истории можно вычислить, умножив интервал на размер истории. В этом примере 478 минут составляют примерно 8 часов данных о температуре. У некоторых дисков интервал регистрации температуры установлен значительно больше (один час или более). Интервал регистрации температуры потенциально полезен для исследований: если диск был изъят сразу после преступления, известные изменения температуры можно было бы соотнести с записью температуры диска.

```
...
SCT Temperature History Version: 2
Temperature Sampling Period: 1 minute
Temperature Logging Interval: 1 minute
Min/Max recommended Temperature: 0/60 Celsius
Min/Max Temperature Limit: -41/85 Celsius
Temperature History Size (Index): 478 (175)

Index Estimated Time Temperature Celsius
176 2016-01-07 05:00 ? -
... ..(300 skipped). .. -
477 2016-01-07 10:01 ? -
0 2016-01-07 10:02 29 **********
1 2016-01-07 10:03 30 ***********
... ..( 68 skipped). .. ***********
70 2016-01-07 11:12 30 ***********
71 2016-01-07 11:13 31 ************
... ..(103 skipped). .. ************
175 2016-01-07 12:57 31 ************
...
```

Заключительный раздел вывода в этом примере показывает статистику физических ошибок. Полезно сравнить эту статистику со значениями во время или в конце получения образа, чтобы убедиться, что в процессе не возникли физические ошибки.

```
...
SCT Error Recovery Control command not supported
Device Statistics (GP/SMART Log 0x04) not supported
SATA Phy Event Counters (GP Log 0x11)
ID Size Value Description
0x0001 2 0 Command failed due to ICRC error
0x0002 2 0 R_ERR response for data FIS
0x0003 2 0 R_ERR response for device-to-host data FIS
0x0004 2 0 R_ERR response for host-to-device data FIS
0x0005 2 0 R_ERR response for non-data FIS
0x0006 2 0 R_ERR response for device-to-host non-data FIS
0x0007 2 0 R_ERR response for host-to-device non-data FIS
0x0008 2 0 Device-to-host non-data FIS retries
0x0009 2 6 Transition from drive PhyRdy to drive PhyNRdy
0x000a 2 6 Device-to-host register FISes sent due to a COMRESET
0x000b 2 0 CRC errors within host-to-device FIS
```

```
0x000f 2 0 R_ERR response for host-to-device data FIS, CRC
0x0012 2 0 R_ERR response for host-to-device non-data FIS, CRC
0x8000 4 14532 Vendor specific
```

В зависимости от поставщика накопителя могут существовать и другие журналы SMART. Дополнительные сведения о прочих флагах и запросах, которые можно отправлять подключенным исследуемым дискам, приведены на странице руководства `smartctl(8)`.

Включение доступа к скрытым секторам

В криминалистической литературе обработка HPA и DCO часто включается в процесс создания образа. Более того, некоторые программы создания образов способны обнаруживать и удалять эти скрытые области во время получения образа. В этой книге обнаружение и удаление HPA/DCO относится к процессу подготовки, а не к непосредственному созданию образа. После того как эти скрытые области стали доступны, никакой специальной методики для создания их образа не требуется. Это просто сектора диска, защищенные параметрами конфигурации накопителя. Предоставить к ним доступ для последующего процесса создания образа - простой подготовительный шаг. Удаление HPA или DCO изменяет конфигурацию накопителя, но не изменяет его содержимое.[3]

В этом разделе также рассматриваются служебные сектора накопителя и служебные области диска, однако эта тема упоминается лишь кратко, поскольку получить доступ к таким областям с помощью распространенных инструментов с открытым исходным кодом непросто.

Удаление DCO

DCO был разработан, чтобы позволить производителям компьютерных систем придавать разным моделям накопителей одинаковый набор функций. С помощью DCO можно отключить определенные функции и уменьшить емкость накопителя (число доступных для использования секторов) в соответствии с требованиями поставщика. Выявление и удаление DCO - стандартная криминалистическая практика при анализе диска подозреваемого.

DCO представляет собой общую конфигурационную накладку и позволяет переопределить множество функций. Это понятие относится не только к числу секторов накопителя.

Определить наличие DCO и получить фактическое число доступных секторов позволяют две команды `hdparm`. Первая команда определяет, включен ли на накопителе набор функций DCO. В этом примере текущий размер диска указан как 474 Гбайт, или 926773168 секторов (при размере сектора 512 байт), а звездочка (*) рядом с `Device Configuration Overlay feature set` показывает, что он активен:

```
# hdparm -I /dev/sdl

/dev/sdl:

ATA device, with non-removable media
Model Number: WDC WD5003AZEX-00MK2A0
...
LBA48 user addressable sectors: 926773168
Logical Sector size: 512 bytes
Physical Sector size: 4096 bytes
device size with M = 1024*1024: 452525 MBytes
device size with M = 1000*1000: 474507 MBytes (474 GB)
...
* Device Configuration Overlay feature set
...
```

Вторая команда специально запрашивает функции, измененные с помощью DCO:

```
# hdparm --dco-identify /dev/sdl

/dev/sdl:
DCO Revision: 0x0002
The following features can be selectively disabled via DCO:
Transfer modes:
udma0 udma1 udma2 udma3 udma4 udma5 udma6
Real max sectors: 976773168
ATA command/feature sets:
security HPA
SATA command/feature sets:
NCQ interface_power_management SSP
```

В этом примере значение `Real max sectors` равно 976773168, а заявленный размер на 25 Гбайт меньше этого значения, что указывает на наличие DCO. Заявленный размер 474 Гбайт также не соответствует маркировке 500 Гбайт на физическом накопителе. Ожидаемое число секторов можно подтвердить, сверив номер модели накопителя с документацией поставщика на продукт.

Подтвердив наличие DCO с помощью `hdparm`, его можно удалить той же командой. Сначала запустите `hdparm`, чтобы убедиться, что конфигурация накопителя не заблокирована и не заморожена:

```
# hdparm -I /dev/sdl

/dev/sdl:

ATA device, with non-removable media
Model Number: WDC WD5003AZEX-00MK2A0
...
Security:
...
not locked

not frozen
...
```

Некоторые BIOS или операционные системы при загрузке отправляют команду ATA, замораживающую конфигурацию DCO для предотвращения злоумышленных изменений. В этом случае горячее подключение кабеля питания накопителя после загрузки должно привести к его раскручиванию в незамороженном состоянии.[4] Многие мосты USB автоматически раскручивают подключенный диск в незамороженном состоянии. Если накопитель заблокирован, обратитесь к разделу "Идентификация и разблокирование дисков, защищенных паролем ATA" на странице 126.

Когда накопитель будет готов, можно отправить соответствующую команду ATA для сброса DCO, сделав доступными дополнительные скрытые сектора.

Простой запуск команды `hdparm` с параметром `--dco-restore` ничего не сделает, а лишь выведет предупреждение:

```
# hdparm --dco-restore /dev/sdl

/dev/sdl:
Use of --dco-restore is VERY DANGEROUS.
You are trying to deliberately reset your drive configuration back to the factory
defaults.
This may change the apparent capacity and feature set of the drive, making all data
on it inaccessible.
You could lose*everything*.
Please supply the --yes-i-know-what-i-am-doing flag if you really want this.
Program aborted.
```

Следуя указаниям и добавив флаг `--yes-i-know-what-i-am-doing`, DCO можно удалить следующим образом:

```
# hdparm --yes-i-know-what-i-am-doing --dco-restore /dev/sdl

/dev/sdl:
issuing DCO restore command
```

Теперь при повторном запуске команды `hdparm -I` будут показаны все сектора.

```
# hdparm -I /dev/sdl

/dev/sdl:

ATA device, with non-removable media
Model Number: WDC WD5003AZEX-00MK2A0
...
LBA48 user addressable sectors: 976773168

Logical Sector size: 512 bytes
Physical Sector size: 4096 bytes
device size with M = 1024*1024: 476940 MBytes
device size with M = 1000*1000: 500107 MBytes (500 GB)
...
```

Теперь можно получить образ накопителя или проанализировать его криминалистическими инструментами. Важно зафиксировать точное смещение секторов скрытой области DCO: оно пригодится, если потребуется извлечь только сектора DCO для отдельного анализа.

Удаление DCO с помощью `hdparm` может оказаться непростым. Если команды удаления вызывают проблемы с конкретным накопителем, прочитайте страницу руководства `hdparm(8)`.

У инструмента `tableau-parm` есть флаг `-r`, который должен удалять DCO (и, возможно, HPA) с накопителя.

Удаление HPA

HPA был разработан, чтобы позволить производителям компьютерных систем хранить данные способом, который обычно недоступен пользователю. Примеры применения HPA включают диагностические инструменты, разделы восстановления и так далее. Эти специальные области часто активируются горячими клавишами BIOS во время запуска.

Обнаружить наличие HPA можно одной командой `hdparm`:

```
# hdparm -N /dev/sdl

/dev/sdl:
max sectors = 879095852/976773168, HPA is enabled
```

Здесь `HPA is enabled` указывает на наличие HPA. Поле `max sectors` содержит число видимых секторов, за которым следует фактическое число секторов. В этом примере разность двух значений дает 50 Гбайт - это и есть защищенная область основной системы.

Временно удалить HPA можно той же командой (как и при удалении DCO, появляется предупреждение и требуется флаг `--yes-i-know-what-i-am-doing`):

```
# hdparm --yes-i-know-what-i-am-doing -N 976773168 /dev/sdl

/dev/sdl:
setting max visible sectors to 976773168 (temporary)
max sectors = 976773168/976773168, HPA is disabled
```

Результат этой команды лишь временный: после следующего отключения и включения питания накопителя исходная HPA вернется на место. Чтобы сделать изменение постоянным, добавьте p к числу секторов следующим образом:

```
# hdparm --yes-i-know-what-i-am-doing -N p976773168 /dev/sdl

/dev/sdl:

setting max visible sectors to 976773168 (permanent)
max sectors = 976773168/976773168, HPA is disabled
```

Теперь HPA удалена, и можно получить образ накопителя или проанализировать его криминалистическими инструментами. Важно зафиксировать точное смещение секторов скрытой области HPA: оно пригодится, если потребуется извлечь только сектора HPA для отдельного анализа.

Удаление HPA с помощью `hdparm` может оказаться непростым. Если команды удаления вызывают проблемы с конкретным накопителем, прочитайте страницу руководства `hdparm(8)`.

Ранее криминалистический комплект Sleuth Kit содержал две утилиты для обнаружения и временного удаления HPA: `disk_stat` и `disk_sreset`. В 2009 году их удалили, поскольку те же функции появились в других инструментах, например `hdparm`.

Доступ к служебной области накопителя

Жестким дискам необходимо хранить такие сведения, как журналы SMART, пароли ATA, списки поврежденных секторов, микропрограммы и другие постоянные данные. Как правило, эти сведения хранятся на пластинах диска в зарезервированных, недоступных пользователю секторах, называемых системной областью (также служебной областью, отрицательными секторами или секторами обслуживания). Доступ к этой области осуществляется посредством закрытых команд поставщика, которые обычно не публикуются.

Общего систематического подхода к доступу в системные области диска не существует. Каждый производитель дисков реализует системные области по-своему, отраслевых стандартов нет, а общедоступных инструментов мало. Существуют специализированные коммерческие инструменты, например PC-3000 компании Ace Laboratory или Atola Insight Forensic, способные обращаться к служебным областям многих дисков.[5] [6]

В некоторых случаях стандартные интерфейсы SATA, USB или SAS можно обойти и обратиться к носителю информации через порты отладки или диагностики, встроенные в электронику накопителя. Эти интерфейсы могут использовать последовательный RS-232/TTL, JTAG для доступа к микросхемам[7] либо недокументированные закрытые команды поставщика, передаваемые по обычному интерфейсу накопителя. Такой доступ к носителю не стандартизован ни между производителями, ни даже между накопителями одного производителя.

Для иллюстрации в следующем примере показано чтение сведений через последовательный интерфейс накопителя Seagate Barracuda ST500DM002. У накопителя рядом с разъемом данных SATA имеется последовательный порт, к которому можно обратиться кабелем USB TTL 3 В. В этом примере используется стандартная программа эмуляции последовательного терминала - команда Linux `cu` (connect UNIX).

На рис. 5-1 показана фотография USB-кабеля, подключенного к штыревой колодке на задней стороне накопителя.

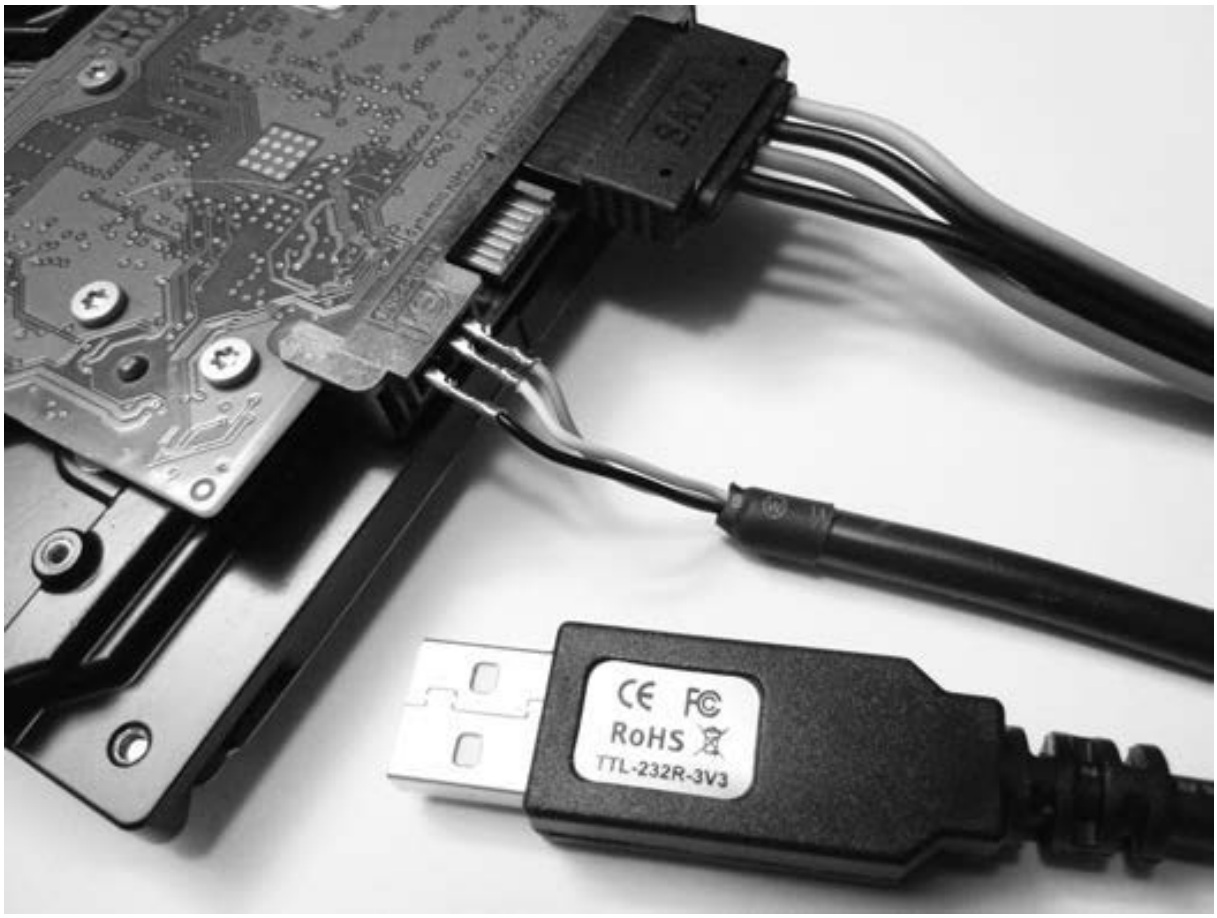

Рис. 5-1. Доступ к микропрограмме диска через последовательный порт

> **Предупреждение.** Этот метод не следует применять без специальной подготовки или инструментов. Существует риск физически повредить диск без возможности восстановления.

После подключения терминала и подачи питания на накопитель выводится сообщение загрузки. Нажатие `CTRL-Z` переводит накопитель в диагностический режим с командной строкой микропрограммы накопителя (подобно терминалам UNIX или аналоговым модемам).

```
$ cu -s 38400 -l /dev/ttyUSB0
Connected.
Boot 0x10M
Spin Up[0x00000000][0x0000B67C][0x0000BA10]
Trans.

Rst 0x10M
MC Internal LPC Process
Spin Up
(P) SATA Reset

ASCII Diag mode

F3 T>
```

Через этот диагностический интерфейс можно получить подробные низкоуровневые сведения о диске. В следующем примере команда x уровня 2 раскрывает внутреннюю физическую геометрию накопителя и разбиение пользовательской и системной областей:

```
F3 2>x

User Partition
LBAs 000000000000-0000075D672E
PBAs 000000000000-0000076F8EDD
HdSkew 006E, CylSkew 002D
ZonesPerHd 11

Head 0, PhyCyls 000000-040001, LogCyls 000000-03F19C

Physical Logical Sec Sym Sym Data
Zn Cylinders Cylinders Track Wedge Track Rate
00 000000-0003FB 000000-0003FB 010F 0D77 000F4D40 1263.750
01 0003FC-005A41 0003FC-005A41 0130 0F1A 00112A40 1417.500
...
Head 1, PhyCyls 000000-039877, LogCyls 000000-038B61

Physical Logical Sec Sym Sym Data
Zn Cylinders Cylinders Track Wedge Track Rate
00 000000-00035B 000000-00035B 0130 0F16 001124A0 1415.625
01 00035C-004E72 00035C-004E72 0145 1025 00125E80 1516.875
...
System Partition
LBAs 000000000000-0000000972CF
PBAs 000000000000-00000009811F
HdSkew 006E, CylSkew 0018
ZonesPerHd 02

Head 0, PhyCyls 040002-040155, LogCyls 000000-000152

Physical Logical Sec Sym Sym Data
Zn Cylinders Cylinders Track Wedge Track Rate
00 040002-0400AB 000000-0000A9 0394 063D 00072AE0 592.500
01 0400AC-040155 0000AA-000152 0394 063D 00072AE0 592.500

Head 1, PhyCyls 039878-0399CB, LogCyls 000000-000152

Physical Logical Sec Sym Sym Data
Zn Cylinders Cylinders Track Wedge Track Rate

00 039878-039921 000000-0000A9 0394 063D 00072AE0 592.500
01 039922-0399CB 0000AA-000152 0394 063D 00072AE0 592.500
```

Подобные диагностические интерфейсы могут предоставлять доступ к секторам диска в системных областях и к другим сведениям, недоступным иными способами. Существуют интернет-форумы, где обсуждаются низкоуровневый доступ к дискам и восстановление данных, например HDDGURU и The HDD Oracle.

К способам доступа к нижележащим областям SSD или флеш-носителей относится физическое снятие (выпаивание) микросхем памяти, иногда называемое `chip-off`. Затем содержимое памяти этих микросхем можно извлечь и реконструировать в читаемые блоки данных.

Некоторые устройства (Интернет вещей, мобильные устройства и так далее) могут иметь интерфейс JTAG, предоставляющий доступ к содержимому памяти. JTAG - хорошо документированный стандарт, который можно применять в криминалистическом контексте для извлечения данных (см. Evidence Magazine).

Более глубокое рассмотрение этих методов выходит за рамки книги. Интерфейсы JTAG и последовательный доступ к дискам упомянуты здесь в иллюстративных целях, чтобы вы знали о существовании подобных методов в криминалистической отрасли.

Защита паролем ATA и самошифрующиеся накопители

В этом разделе рассматриваются стандартные функции безопасности, реализованные поставщиками дисков. К ним относятся блокирование накопителя, защита паролем, самошифрующиеся накопители и другие механизмы безопасности. Хотя некоторые из обсуждаемых здесь функций не получили широкого распространения, понимать их все же важно в условиях профессиональной криминалистической лаборатории.

Методы восстановления паролей подробно здесь не описываются. Примеры показывают, как подключить защищенный паролем носитель к основной системе получения данных при подготовке к созданию образа. Предполагается, что пароли уже известны.

Способы получения паролей выходят за рамки книги, но методы восстановления могут включать следующее:

• Полный перебор с исчерпывающими попытками ввода разных паролей до нахождения правильного.
• Поиск паролей, скрытых или сохраненных в доступном месте.
• Знание о повторном использовании пароля в разных учетных записях или устройствах. Восстановление в одном месте предоставляет доступ ко всем.
• В зависимости от юрисдикции человека могут юридически обязать предоставить пароли.
• Пароль может добровольно сообщить дружелюбный или готовый к сотрудничеству владелец (возможно, потерпевший) либо сотрудничающий со следствием соучастник.
• В корпоративных ИТ-средах могут существовать депонирование ключей или резервные копии.

Идентификация и разблокирование дисков, защищенных паролем ATA

Команды ATA/ATAPI (http://www.t13.org/) определяют набор функций безопасности, ограничивающий доступ к диску с помощью паролей. Когда эта функция включена, микропрограмма запрещает выполнение определенных команд ATA, включая доступ к содержимому, пока не будет предоставлен требуемый пароль. Это лишь функция управления доступом; для защиты данных на диске она не использует шифрование.

Инструмент hdparm позволяет определить, включен ли на диске набор функций безопасности. Например:

```
# hdparm -I /dev/sda
...
Commands/features:
Enabled Supported:
...
* Security Mode feature set
...
Security:
Master password revision code = 1
supported
enabled
locked
not frozen
not expired: security count
supported: enhanced erase
Security level high
60min for SECURITY ERASE UNIT. 60min for ENHANCED SECURITY ERASE UNIT.
...
```

Сведения Commands/features: указывают, что набор функций Security Mode существует и включен, а сведения Security: также подтверждают, что функция поддерживается и включена.

Если в разделе Security: указано enabled, задан пользовательский пароль и накопитель будет заблокирован при загрузке. Если накопитель заблокирован, как в предыдущем примере, доступ к нему запрещен до предоставления правильного пароля. При попытке обратиться к диску операционная система может выдать ошибку устройства или ошибку неудачного выполнения команды. Стандарт T13 определяет, какие команды разрешены, когда диск заблокирован. При заблокированном диске по-прежнему возможен доступ к ряду команд, в том числе к запросу сведений SMART.

Можно задать два пароля: пользовательский и главный. Если задан пользовательский пароль, защита включена (как показано в предыдущем примере). Задание только главного пароля защиту не включает.

Если главный пароль ни разу не задавался (при этом заводской пароль по умолчанию все же может существовать), значение Master password revision code будет равно 65534. При первом задании главного пароля это значение устанавливается в 1 и увеличивается при каждом последующем задании главного пароля.

Два уровня безопасности управляют действием правильных паролей. Security level соответствует биту MASTER PASSWORD CAPABILITY в стандарте T13 и может иметь значение high или maximum. Если установлен высокий уровень безопасности, накопитель можно разблокировать как пользовательским, так и главным паролем. Если установлен максимальный уровень, главный пароль разрешает команды безопасного стирания, но разблокировать накопитель позволяет только пользовательский пароль.

Некоторые ПК после загрузки могут отправлять команду замораживания защиты, чтобы предотвратить отправку последующих команд безопасности даже с правильными паролями (для предотвращения злоумышленных атак с установкой пароля). Вывод `Security` команды `hdparm` показывает, заморожен ли накопитель. Многие мосты USB автоматически раскручивают подключенный диск в незамороженном состоянии, однако при сохраняющихся затруднениях можно попробовать несколько вариантов:

• Проверить в BIOS параметры включения или отключения команды замораживания
• Использовать криминалистический загрузочный компакт-диск, предотвращающий отправку команд замораживания
• Подключить диск к отдельной плате контроллера (не встроенной в системную плату)
• Выполнить горячее подключение диска к системе (если оно поддерживается)
• Использовать системную плату, не отправляющую команды замораживания

Если пользовательский пароль известен, а защита накопителя не заморожена, разблокировать накопитель можно следующим образом:

```
# hdparm --security-unlock "mysecret99" /dev/sdb
security_password="mysecret99"

/dev/sdb:
Issuing SECURITY_UNLOCK command, password="mysecret99", user=user
```

По умолчанию `hdparm` передается пользовательский пароль, а главный пароль необходимо явно указать дополнительным параметром командной строки. Если главный пароль известен, а установлен высокий уровень безопасности, разблокировать накопитель с помощью главного пароля можно следующим образом:

```
# hdparm --user-master m --security-unlock "companysecret22" /dev/sdb
security_password="companysecret22"

/dev/sdb:
Issuing SECURITY_UNLOCK command, password="companysecret22", user=master
```

Если ни один пароль не известен, доступ к диску обычными инструментами невозможен. Сведения о пароле хранятся в служебных или системных областях диска и, как правило, недоступны без специального оборудования или инструментов. Тем не менее существует несколько дополнительных вариантов, которые рассматриваются далее.

Главный пароль может быть установлен в заводское значение по умолчанию, и его можно использовать для получения доступа к накопителю (если установлен высокий, а не максимальный уровень безопасности). Списки заводских главных паролей по умолчанию легко найти в Интернете.

Полный перебор для определения главного или пользовательского пароля неэффективен, поскольку после пяти неудачных попыток накопитель необходимо сбрасывать. Однако при наличии небольшого набора вероятных паролей многократные попытки становятся осуществимыми и благодаря удаче могут привести к успеху.

Специализированные компании по восстановлению данных предоставляют услуги и аппаратные инструменты, способные восстановить или сбросить пароли ATA Security Feature Set из служебных областей диска. Успех гарантирован не для всех дисков, но компании по восстановлению данных часто публикуют перечни поддерживаемых дисков. В некоторых случаях диск придется отправить в лабораторию компании, что может иметь последствия для цепочки хранения доказательств. Дополнительные сведения см. в разделе "Доступ к служебной области накопителя" на странице 122.

Поставщик жесткого диска, возможно, сможет помочь отключить или сбросить пароль ATA. Это будет зависеть от готовности поставщика накопителя к сотрудничеству, возможности доказать право собственности на диск и его содержимое, полномочий запрашивающей стороны и мотивов восстановления данных.

Могут существовать аппаратные и микропрограммные модификации, а также опубликованные исследователями методы, предоставляющие доступ к определенным моделям жестких дисков. Исследовательское сообщество в области безопасности регулярно находит новаторские способы доступа к данным в труднодоступных местах и их изменения.

Идентификация и разблокирование самошифрующихся накопителей Opal

Самошифрующиеся накопители (self-encrypting drives, SED) представляют собой разновидность полнодискового шифрования (full-disk encryption, FDE). В отличие от программного FDE (TrueCrypt, FileVault, LUKS и так далее), где шифрованием управляет операционная система, возможности шифрования SED встроены непосредственно в электронику и микропрограмму накопителя. SED не зависят от операционной системы и основаны на независимых от поставщиков стандартах. Международный орган, отвечающий за определение стандарта, - Trusted Computing Group (TCG; http://www.trustedcomputinggroup.org/). Этот стандарт называется TCG Storage Security Subsystem Class: Opal, Specification Version 2.00.

В этом разделе показано, как идентифицировать накопители с шифрованием Opal и использовать соответствующие ключи для разблокирования накопителя. Восстановление ключей шифрования выходит за рамки книги. В приведенных здесь примерах предполагается, что ключ известен.

Физический осмотр накопителя уже может показать, является ли он Opal SED. На рис. 5-2 показана напечатанная на этикетке накопителя строка Physical Secure ID (PSID). Эта строка используется функцией Opal RevertSP, которая безопасно создает новый ключ, уничтожая все данные и возвращая накопитель в первоначальное заводское состояние. PSID нельзя запросить у накопителя: его необходимо прочитать физически либо отсканировать, если имеется QR-код. Наличие строки PSID не означает, что накопитель заблокирован и пароли заданы; оно лишь указывает, что накопитель поддерживает полнодисковое шифрование Opal.

Рис. 5-2. PSID самошифрующегося накопителя Opal

У полнодискового шифрования есть проблема курицы и яйца. Если зашифрован весь накопитель, включая загрузочный сектор, как система может выполнить главную загрузочную запись (MBR) и запросить пароль или другие учетные данные безопасности?

Решением стало создание теневой MBR и ее хранение в системной области диска (там же, где хранятся данные SMART, списки поврежденных блоков и так далее). Когда диск Opal заблокирован, основной системе видна только теневая MBR. Это группа незашифрованных секторов (она может быть большой - например, размером 150 Мбайт), которая выполняется как обычная MBR (основная система совершенно не знает, что использует теневую MBR). Эта альтернативная загрузочная область может выполнять код, запрашивающий пароль, обращаться к микросхеме Trusted Platform Module (TPM) или смарт-карте либо получать другие учетные данные. После разблокирования диска становится видна настоящая MBR и может начаться обычный процесс загрузки.

Для управления шифрованием Opal SED в Linux был создан инструмент командной строки с открытым исходным кодом. Первоначально он назывался `msed` и находился по адресу https://github.com/r0m30/msed/, но недавно был переименован в `sedutil-cli` и перемещен по адресу https://github.com/Drive-Trust-Alliance/sedutil/. Этот инструмент все еще разрабатывается и может работать не со всеми накопителями. Тщательно следуйте инструкциям и убедитесь, что в ядре включен параметр `libata.allow_tpm`.

Следующая команда сканирует локальную систему в поисках всех SED-накопителей, совместимых с Opal. Из четырех подключенных накопителей один диск определяется как Opal версии 2:

```
# sedutil-cli --scan
Scanning for Opal compliant disks
/dev/sda 2 Crucial_CT250MX200SSD1 MU01
/dev/sdb No WDC WD20EZRX-00D8PB0 80.00A80
/dev/sdc No INTEL SSDSA2CW300G3 4PC10302
/dev/sdd No Kingston SHPM2280P2H/240G OC34L5TA
No more disks present ending scan
```

У накопителя можно запросить сведения о состоянии Opal, включая то, зашифрован ли диск, заблокирован ли он и имеется ли теневая MBR (в этом примере присутствуют все три признака):

```
# sedutil-cli --query /dev/sda
/dev/sda ATA Crucial_CT250MX200SSD1 MU01 15030E69A241
...
Locking function (0x0002)

Locked = Y, LockingEnabled = Y, LockingSupported = Y, MBRDone = N,
MBREnabled = Y, MediaEncrypt = Y
...
```

Можно отправить две команды: первую для отключения блокирования, вторую - чтобы сообщить диску, что теневая MBR больше не нужна (MBR "готова"). В этом примере пароль - `xxmonkey`:

```
# sedutil-cli --disableLockingRange 0 xxmonkey /dev/sda
- 16:33:34.480 INFO: LockingRange0 disabled

# sedutil-cli --setMBRDone on xxmonkey /dev/sda
- 16:33:54.341 INFO: MBRDone set on
```

На этом этапе сообщение ядра (`dmesg`) может показать изменение доступных устройств. Теперь состояние в этом примере выглядит следующим образом:

```
# sedutil-cli --query /dev/sda
/dev/sda ATA Crucial_CT250MX200SSD1 MU01 15030E69A241
...
Locking function (0x0002)
Locked = N, LockingEnabled = Y, LockingSupported = Y, MBRDone = Y,
MBREnabled = Y, MediaEncrypt = Y
...
```

Накопитель больше не заблокирован, и теневая MBR больше не видна. Настоящая MBR и остальная часть расшифрованного диска доступны, и к ним можно обращаться обычными криминалистическими инструментами. Теперь становится видна таблица разделов установки Linux, как показано в следующем примере:

```
# mmls /dev/sda
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0471887871 0471885824 Linux (0x83)
03: ----- 0471887872 0471889919 0000002048 Unallocated
04: Meta 0471889918 0488396799 0016506882 DOS Extended (0x05)
05: Meta 0471889918 0471889918 0000000001 Extended Table (#1)
06: 01:00 0471889920 0488396799 0016506880 Linux Swap / Solaris x86 (0x82)
07: ----- 0488396800 0488397167 0000000368 Unallocated
```

Заблокированный накопитель, у которого не включена теневая MBR, создаст несколько сообщений об ошибках в выводе ядра `dmesg`.

Простой пример, описанный в этом разделе, приведен исключительно для иллюстрации. Некоторые диски Opal могут вести себя с этим инструментом иначе. В реальных ситуациях ключ может быть не простым паролем, а привязанным к TPM или другому корпоративному механизму безопасности. Если в такой ситуации подать неправильные команды, данные на диске могут быть безвозвратно уничтожены (мгновенно, если уничтожен ключ).

С точки зрения криминалистики может быть полезно также создать образ теневой MBR для анализа. В ней могут содержаться интересные артефакты, относящиеся ко времени настройки шифрования диска. Также вполне возможно, что в области теневой MBR накопителей с поддержкой Opal могут быть скрыты данные.

Зашифрованные флеш-накопители

USB-флеш-накопители, продаваемые как "защищенные" устройства, часто поставляются с закрытым программным решением шифрования от поставщика. Некоторые накопители предлагают независимое от операционной системы шифрование с аутентификацией посредством клавиатуры, считывателя отпечатков пальцев или смарт-карты (см. рис. 5-3).

Рис. 5-3. Зашифрованные USB-флеш-накопители

Для закрытых решений может не существовать совместимого инструмента управления доступом, что затрудняет получение расшифрованных данных в Linux. После аутентификации устройства со встроенным механизмом аутентификации должны выглядеть как обычные USB-устройства хранения.

Заблокированные защищенные флеш-накопители могут вести себя по-разному при подключении к основной системе. Некоторые вообще не показывают, что были подключены к основной системе. Некоторые выглядят как устройство съемных носителей без носителя (подобно устройству чтения карт памяти). Некоторые отображаются как CD-ROM с доступной для запуска или установки программой, управляющей накопителем.

Существуют также более крупные внешние накопители с аппаратным шифрованием, для разблокирования которых может требоваться PIN-код. Пример такого накопителя описан в главе 7 (см. рис. 7-1 на странице 216).

Подключение съемных носителей

В этом разделе рассматривается подключение устройств, использующих съемные носители информации. Наиболее распространенные примеры съемных носителей - оптические диски, карты памяти и магнитные ленты. В некотором смысле съемный носитель подключается к основной системе получения данных дважды. Сначала подключается электроника устройства, а затем отдельным шагом вставляется съемный носитель. Начнем с обсуждения накопителей оптических носителей.

Накопители оптических носителей

Оптические накопители обычно подключаются внутри системы по SATA или снаружи по USB. Накопители появляются в дереве устройств Linux, но без носителя. Выполнение криминалистических команд для пустого накопителя дает очевидный результат, как показано здесь:

```
# mmls /dev/cdrom
Error opening image file (raw_open: file "/dev/cdrom" - No medium found)
```

Две полезные команды предоставляют сведения о подключенном накопителе и вставленных дисках. Команда `cd-drive` сообщает подробности о подключенном оптическом накопителе (внутреннем или внешнем), включая различные функции, поддерживаемые носители и так далее:

```
# cd-drive
cd-drive version 0.83 x86_64-pc-linux-gnu
...
CD-ROM drive supports MMC 3
Drive: /dev/cdrom
Vendor : ASUS
Model : BW-16D1HT
Revision : 1.01
Profile List Feature
Blu Ray BD-RE
Blu Ray BD-R random recording
Blu Ray BD-R sequential recording
Blu Ray BD-ROM
DVD+R Double Layer - DVD Recordable Double Layer
DVD+R - DVD Recordable
DVD+RW - DVD Rewritable
DVD-R - Double-layer Jump Recording
DVD-R - Double-Layer Sequential Recording
Re-recordable DVD using Sequential Recording
Re-recordable DVD using Restricted Overwrite
Re-writable DVD
Re-recordable DVD using Sequential recording
Read only DVD

CD-RW Re-writable Compact Disc capable
Write once Compact Disc capable
Read only Compact Disc capable
...
Removable Medium Feature
Tray type loading mechanism
can eject the medium or magazine via the normal START/STOP command
can be locked into the Logical Unit
...
```

После вставки диска в накопитель сведения о носителе можно получить командой `cd-info`. Результат включает режим, формат и сведения об издателе:

```
# cd-info
cd-info version 0.83 x86_64-pc-linux-gnu
Disc mode is listed as: CD-DA
CD-ROM Track List (1 - 1)
#: MSF LSN Type Green? Copy? Channels Premphasis?
1: 00:02:00 000000 data false no
170: 39:42:20 178520 leadout (400 MB raw, 400 MB formatted)
Media Catalog Number (MCN): 0000000000000
TRACK 1 ISRC: 000000000000
Last CD Session LSN: 0
audio status: invalid
_____________________________________

CD Analysis Report
CD-ROM with ISO 9660 filesystem
ISO 9660: 154301 blocks, label `SOLARIS_2_5_1_SPARC '
```

```
Application: NOT SPECIFIED
Preparer : SOLARIS_PRODUCT_ENGINEERING
Publisher : SUNSOFT_INC
System : SUNSOFT_INC
Volume : SOLARIS_2_5_1_SPARC
Volume Set : SOLARIS_2_5_1_SERIES
```

Извлечь оптический носитель можно командой оболочки `eject`.

Использовать блокираторы записи с оптическими накопителями нет необходимости. Простой доступ к файлам на диске не обновляет никакие временные метки. Для изменения оптического диска требуются явные команды записи, что снижает риск случайного изменения.

Накопители на магнитной ленте

Список подключенных ленточных накопителей можно определить инструментом `lshw` с классом `tape`. Вывод содержит сведения о поставщике накопителя, серийном номере и устройстве.

В этом примере найдены два ленточных накопителя (LTO и DAT):

```
# lshw -class tape
*-tape
description: SCSI Tape
product: LTO-5 HH
vendor: TANDBERG
physical id: 0.0.0
bus info: scsi@13:0.0.0
logical name: /dev/nst0
version: Y629
serial: HU1246T99F
capabilities: removable
configuration: ansiversion=6
*-tape
description: SCSI Tape
product: DAT160
vendor: HP
physical id: 0.0.0
bus info: scsi@15:0.0.0
logical name: /dev/nst1
version: WU8A
serial: HU10123NFH
capabilities: removable
configuration: ansiversion=3
```

Накопители на магнитной ленте обычно являются устройствами SCSI, которые можно опрашивать стандартными командами SCSI. Стандартный инструмент управления лентами - `mt`: он предоставляет сведения о состоянии накопителя, управляет положением ленты и извлекает носитель. Инструмент `mt` может предоставить основные сведения о ленте, но `tapeinfo` более всеобъемлющ. В этом примере инструменты `mt` и `tapeinfo` запрашивают состояние ленточного накопителя LTO с загруженной лентой:

```
# mt -f /dev/nst0 status
SCSI 2 tape drive:
File number=1, block number=0, partition=0.
Tape block size 0 bytes. Density code 0x58 (no translation).
Soft error count since last status=0
General status bits on (81010000):
EOF ONLINE IM_REP_EN

# tapeinfo -f /dev/nst0
Product Type: Tape Drive
Vendor ID: 'TANDBERG'
Product ID: 'LTO-5 HH '
Revision: 'Y629'
Attached Changer API: No
```

```
SerialNumber: 'HU1246T99F'
MinBlock: 1
MaxBlock: 16777215
SCSI ID: 0
SCSI LUN: 0
Ready: yes
BufferedMode: yes
Medium Type: Not Loaded
Density Code: 0x58
BlockSize: 0
DataCompEnabled: yes
DataCompCapable: yes
DataDeCompEnabled: yes
CompType: 0x1
DeCompType: 0x1
Block Position: 166723430
Partition 0 Remaining Kbytes: 1459056
Partition 0 Size in Kbytes: 1459056
ActivePartition: 0
EarlyWarningSize: 0
NumPartitions: 0
MaxPartitions: 1
```

Головка ленты установлена на втором файле на ленте (файл 1 находится после файла 0). Смещение блока и смещение файла полезны при криминалистическом получении отдельных файлов с ленты.

С помощью `mt` можно перемотать ленты и перевести их в автономный режим (извлечь):

```
# mt -f /dev/nst0 status
```

При подключении ленточного устройства к системе Linux создается ряд соответствующих устройств.

```
# ls -1 /dev/*st0*
/dev/nst0
/dev/nst0a
/dev/nst0l
/dev/nst0m
/dev/st0
/dev/st0a
/dev/st0l
/dev/st0m
```

Устройства `st*` автоматически перематывают ленту после каждой команды (что требуется не всегда), а `nst*` являются устройствами без перемотки. Символы `a`, `l` и `m` обозначают то же устройство, но с другими характеристиками (размером блока, сжатием). При криминалистическом получении следует использовать устройства `nst*` (без дополнительного символа `a`, `l` или `m`).

Карты памяти

Карты памяти обычно подключаются к основной системе через USB-адаптер с несколькими разъемами для разных типов карт памяти. При подключении адаптер создает съемное устройство SCSI для каждого разъема (даже когда разъемы пусты). Это поведение можно наблюдать в следующем выводе `dmesg`.

```
[ 2175.331711] usb 1-7: new high-speed USB device number 10 using xhci_hcd
[ 2175.461244] usb 1-7: New USB device found, idVendor=058f, idProduct=6362
[ 2175.461249] usb 1-7: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 2175.461252] usb 1-7: Manufacturer: Generic
[ 2175.461938] usb-storage 1-7:1.0: USB Mass Storage device detected
[ 2175.462143] scsi host15: usb-storage 1-7:1.0
[ 2176.458662] scsi 15:0:0:0: Direct-Access Generic USB SD Reader 1.00
PQ: 0 ANSI: 0
[ 2176.459179] scsi 15:0:0:1: Direct-Access Generic USB CF Reader 1.01
```

```
PQ: 0 ANSI: 0
[ 2176.459646] scsi 15:0:0:2: Direct-Access Generic USB SM Reader 1.02
PQ: 0 ANSI: 0
[ 2176.460089] scsi 15:0:0:3: Direct-Access Generic USB MS Reader 1.03
PQ: 0 ANSI: 0
[ 2176.460431] sd 15:0:0:0: Attached scsi generic sg11 type 0
[ 2176.460641] sd 15:0:0:1: Attached scsi generic sg12 type 0
[ 2176.460863] sd 15:0:0:2: Attached scsi generic sg13 type 0
[ 2176.461150] sd 15:0:0:3: Attached scsi generic sg14 type 0
[ 2176.463711] sd 15:0:0:0: [sdj] Attached SCSI removable disk
[ 2176.464510] sd 15:0:0:1: [sdk] Attached SCSI removable disk
[ 2176.464944] sd 15:0:0:2: [sdl] Attached SCSI removable disk
[ 2176.465339] sd 15:0:0:3: [sdm] Attached SCSI removable disk
```

По мере установки носителей в разъемы они становятся доступны как USB-устройства массовой памяти с линейной последовательностью "секторов", которые можно получить криминалистическим способом. В продолжение предыдущего примера теперь в разъем устройства чтения карт памяти вставлена карта, и она отображается как блочное устройство:

```
[ 2310.750147] sd 15:0:0:0: [sdj] 7959552 512-byte logical blocks: (4.07 GB/3.79 GiB)
[ 2310.753162] sdj: sdj1
```

Инструменты опроса оборудования, такие как `hdparm` и `smartctl`, могут давать ненадежные результаты, поскольку карты памяти не имеют функций ATA, присущих более сложным накопителям со специализированной электронной схемой.

Подключение других устройств хранения

Иногда носитель информации подключается к основной системе криминалистического получения данных и ведет себя особым образом. В частности, полезно знать об особенностях поведения портативных устройств, компьютерных систем Apple и накопителей NVME.

Режим целевого диска Apple

Режим целевого диска (Target Disk Mode, TDM) позволяет компьютерам Apple с микропрограммой OpenBoot или более новой микропрограммой загрузиться в состояние, при котором система Mac выглядит как внешний дисковый корпус, а внутренние диски доступны как целевые устройства SCSI. Ранние реализации TDM использовали шину FireWire, но позднее перешли на Thunderbolt. Этот режим активируется удержанием клавиши `T` при включении компьютера Apple.

Компьютер Linux без адаптера Thunderbolt может с помощью переходника использовать FireWire и добиться того же результата. На рис. 5-4 показана фотография адаптера Thunderbolt - FireWire.

Рис. 5-4. Адаптер Thunderbolt - FireWire

Обязательно загружайте устройство Apple (удерживая клавишу T) с уже подключенным адаптером Thunderbolt - FireWire; в противном случае микропрограмма Apple не будет использовать адаптер FireWire для целевого устройства.

В следующем примере показан вывод `dmesg` ноутбука Apple в режиме TDM, подключенного к компьютеру Linux с помощью адаптера Thunderbolt - FireWire (Thunderbolt на устройстве Apple; FireWire на компьютере Linux):

```
[ 542.964313] scsi host10: SBP-2 IEEE-1394
[ 542.964404] firewire_core 0000:0e:00.0: created device fw1: GUID
000a27020064d0ef, S800
[ 543.163093] firewire_sbp2 fw1.0: logged in to LUN 0000 (0 retries)
[ 543.163779] scsi 10:0:0:0: Direct-Access-RBC AAPL FireWire Target 0000
PQ: 0 ANSI: 3
[ 543.164226] sd 10:0:0:0: Attached scsi generic sg10 type 14
[ 543.165006] sd 10:0:0:0: [sdj] 236978176 512-byte logical blocks:
(121 GB/113 GiB)
[ 543.165267] sd 10:0:0:0: [sdj] Write Protect is off
[ 543.165271] sd 10:0:0:0: [sdj] Mode Sense: 10 00 00 00
[ 543.165759] sd 10:0:0:0: [sdj] Write cache: enabled, read cache: enabled,
doesn't support DPO or FUA

[ 543.171533] sdj: sdj1 sdj2 sdj3
[ 543.173479] sd 10:0:0:0: [sdj] Attached SCSI disk
```

ПК-системы с Linux и портами Thunderbolt встречаются нечасто, а поддержка в ядре Linux все еще разрабатывается. В качестве альтернативы современные компьютеры Apple можно загрузить с криминалистического загрузочного CD/USB-устройства и получить их образ на локально подключенный диск доказательств.

SSD-накопители NVME

Накопители NVME конкурируют с SATA Express способом непосредственного подключения к шине PCI Express. На момент написания книги аппаратные блокираторы записи для накопителей NVME появились совсем недавно. Компания Tableau (Guidance Software) выпускает USB-мосты с горячим подключением для накопителей NVME и SATA Express. В иллюстративных целях в приведенных здесь примерах используется устройство NVME, непосредственно подключенное к системе Linux.

Для перечисления подключенных устройств NVME можно использовать инструмент `nvme` из пакета `nvme-cli`:

```
# nvme list
Node Model Version Namepace Usage ...
---------------- -------------------- -------- -------- -------------------------
/dev/nvme0n1 INTEL SSDPE2MW400G4 1.0 1 400.09 GB / 400.09 GB ...
/dev/nvme1n1 Samsung SSD 950 PRO 1.1 1 3.01 GB / 256.06 GB ...
...
```

Кроме того, с помощью инструмента `nvme` следует проверить каждый накопитель NVME на наличие нескольких пространств имен. В этом примере существует только одно пространство имен:

```
# nvme list-ns /dev/nvme1
[ 0]:0x1
```

При наличии нескольких пространств имен, возможно, потребуется получать каждое из них отдельно. В этом состоит принципиальное отличие от других накопителей, где один накопитель рассматривается как линейный набор секторов, который можно получить за один проход. Накопители NVME с несколькими пространствами имен, вероятно, потребуют особого рассмотрения.[8]

Важно отметить, что стандарт NVME был создан с нуля без обратной совместимости со стандартами SCSI или ATA (AHCI и так далее). У него собственный набор команд, и он работает независимо от других дисковых систем. Поэтому некоторые инструменты могут работать с оборудованием NVME не так, как ожидается. Любой криминалистический инструмент, работающий непосредственно с низкоуровневыми драйверами устройств, например SATA или SAS, не будет работать с NVME. Однако криминалистические инструменты, работающие с виртуальным блочным уровнем, должны продолжать работать нормально. Кроме того, криминалистические блокираторы записи PCI могут действовать как мост и представлять устройство как устройство SCSI. Например, здесь инструмент `mmls` из Sleuth Kit используется для накопителя NVME, подключенного к основной системе исследования:

```
# mmls /dev/nvme1n1
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0167774207 0167772160 Linux (0x83)
03: 00:01 0167774208 0335546367 0167772160 Linux (0x83)
04: 00:02 0335546368 0500118191 0164571824 Linux (0x83)
```

Обратите внимание, что устройство называется `nvme1n1`, а не просто `nvme1`. При использовании команд с накопителями NVME необходимо указывать пространство имен накопителя.

Как и у других накопителей, у накопителей NVME есть журнал SMART, но получить к нему доступ современными на момент написания версиями `smartctl` нельзя. Однако журнал SMART можно извлечь инструментом `nvme` следующим образом:

```
# nvme smart-log /dev/nvme1
Smart Log for NVME device:/dev/nvme1 namespace-id:ffffffff
critical_warning : 0
temperature : 46 C
available_spare : 100%
available_spare_threshold : 10%
percentage_used : 0%
data_units_read : 2,616
data_units_written : 5,874
host_read_commands : 19,206
host_write_commands : 56,145
controller_busy_time : 0
power_cycles : 34
power_on_hours : 52
unsafe_shutdowns : 17
media_errors : 0
num_err_log_entries : 7
```

Инструмент `nvme` обладает рядом возможностей для опроса подключенных накопителей NVME. Дополнительные сведения см. на странице руководства `nvme(1)` или по адресу https://github.com/linux-nvme/.

На момент написания книги накопители NVME были развивающейся технологией. Благодаря многочисленным преимуществам в производительности и эффективности в будущем они могут получить большее распространение.

Другие устройства с блочным или символьным доступом

Можно создать образ любого устройства, которое ядро Linux определяет как блочное. Некоторые устройства появляются как блочные сразу после подключения к основной системе. Например, так ведут себя многие обычные MP3- и музыкальные проигрыватели, камеры и другие мобильные устройства.

Некоторые устройства, прежде чем стать доступными как блочные, необходимо перевести в другой, "дисковый", режим. Часто этот режим можно выбрать в пользовательском интерфейсе устройства.

Некоторые USB-устройства многофункциональны и наряду с хранением могут предоставлять другие режимы USB. Перед получением образа такие устройства может потребоваться переключить в режим `usb-storage`. Инструмент Linux `usb_modeswitch` способен опрашивать некоторые многофункциональные USB-устройства и переключать режимы.

Заключительные мысли

В этой главе вы научились подключать исследуемый диск к машине получения данных и однозначно идентифицировать устройство для создания образа. Вы познакомились с различными аспектами аппаратного обеспечения ПК (USB, PCI, блочными устройствами и так далее), узнали, как опрашивать систему получения данных и сам накопитель для получения сведений о микропрограмме и SMART. Я продемонстрировал удаление HPA и DCO, а также различные виды защиты, встроенной в аппаратную часть некоторых накопителей. Теперь у вас есть знания, необходимые для криминалистического получения данных, которому посвящена глава 6.

Сноски

[1] [назад] В выводе `lsusb -v` дескриптор устройства `iSerial` для устройств `Linux Foundation...root hub` указывает на адрес устройства PCI, соответствующего контроллеру USB.
[2] [назад] Доступные статистические показатели и журналы SMART различаются у разных поставщиков жестких дисков.
[3] [назад] Исследование криминалистического анализа областей HPA и DCO см. в работе Mayank R. Gupta, Michael D. Hoeschele и Marcus K. Rogers, "Hidden Disk Areas: HPA and DCO", *International Journal of Digital Evidence* 5, no. 1 (2006).
[4] [назад] На некоторых системных платах порты SATA необходимо настроить в BIOS для горячего подключения.
[5] [назад] Исследование возможности сокрытия данных в служебных секторах см. в работе Ariel Berkman, "Hiding Data in Hard-Drive's Service Areas", Recover Information Technologies LTD, 14 февраля 2013 года.
[6] [назад] Todd G. Shipley и Bryan Door, "Forensic Imaging of Hard Disk Drives: What We Thought We Knew", Forensic Focus, 27 января 2012 года.
[7] [назад] Joint Test Action Group (JTAG) определяет стандартизированный отладочный интерфейс для доступа к электронным компонентам.
[8] [назад] На момент написания книги у автора не было доступа для тестирования накопителей NVME с поддержкой нескольких пространств имен. Эти выводы основаны на изучении стандартов и документации.

Глава 6. Получение криминалистического образа

В этой главе объясняется криминалистическое создание образов носителей информации с упором на получение образа криминалистически корректным способом. Это означает извлечение максимального объема данных с конкретного носителя информации, сведение к минимуму воздействия на устройство хранения и носитель, сохранение собранных доказательств и документирование процесса (включая ошибки).

Здесь вы прочитаете о нескольких инструментах и подходах, а также о сильных и слабых сторонах каждого из них. В результате вы сможете принять обоснованное решение о том, какой инструмент наиболее подходит в конкретной ситуации. Вы узнаете, как использовать различные бесплатные инструменты или инструменты с открытым исходным кодом для создания криминалистических образов, такие как `dd`, `dcfldd`, `dc3dd`, `ewfacquire` и `ftkimager-cli`. Кроме того, я описываю инструмент `sfsimage` - сценарий, который использует существующие инструменты получения данных для создания контейнера криминалистических доказательств SquashFS.

Как выбрать инструмент для создания образа диска? В некоторой степени это вопрос личных предпочтений. Возможно, один инструмент вы знаете лучше другого или доверяете определенному инструменту на основании прошлого опыта (либо не доверяете ему на основании прошлого опыта). У каждого инструмента есть свои сильные стороны и уникальные функции. Криминалистические лаборатории, широко использующие EnCase или FTK, могут выбрать `ewfacquire` или `ftkimager-cli` по причинам совместимости и соблюдения правил.

`Dcfldd` и `dc3dd` основаны на зрелом и хорошо испытанном программном обеспечении; они разработаны для криминалистического получения необработанных образов с расширенным хешированием и журналированием. Для дисков с большим числом поврежденных блоков хорошим выбором может стать GNU `ddrescue`. Для встроенного хеширования, шифрования и сжатия во время получения образа интересной альтернативой могут быть современные версии `dd_rescue`.

В конечном счете используемый инструмент будет зависеть от организационных правил криминалистической лаборатории, вида исследования, ваших личных предпочтений и других обстоятельств. В этой книге не рекомендуется какой-либо конкретный инструмент.

Во всех примерах этой главы предполагается следующее:

• Исследуемое устройство хранения физически подключено к рабочей станции получения данных эксперта-криминалиста.
• Исследуемое устройство хранения однозначно идентифицировано.
• Для предотвращения изменения исследуемого диска приняты соответствующие меры блокирования записи.
• Выполнено планирование емкости дисков, гарантирующее отсутствие проблем с дисковым пространством.

Получение образа инструментами dd

Файл образа, создаваемый инструментами на основе `dd`, не является "форматом" в том же смысле, что и другие криминалистические форматы, например EnCase EWF или FTK SMART. Образы, созданные инструментами на основе `dd`, не имеют заголовка, окончания, внутренних маркеров или описательных метаданных о деле либо инциденте. Они представляют собой просто необработанную зеркальную копию участка данных - в данном случае исследуемого диска или другого устройства массовой памяти.

> **Предупреждение.** Инструменты `dd` не прощают ошибок и по соответствующей команде безвозвратно перезапишут любой незащищенный диск.

Чтобы снизить риск повреждения доказательств или рабочей станции эксперта, всегда дважды проверяйте следующее:

• Блокиратор записи защищает диск доказательств или исследуемый диск.
• Серийный номер устройства ввода (`if=`) совпадает с серийным номером на физической этикетке исследуемого диска.
• Подтверждено, что выходной файл (`of=`) является обычным файлом, расположенным в системе эксперта, или программой, способной обработать ожидаемые входные данные из `stdin`.

Стандартный Unix dd и GNU dd

Синтаксис команды `dd` просто задает входной и выходной файлы и может включать другие параметры, изменяющие поведение команды. В следующем примере показано применение `dd` для копирования блочного устройства диска в файл:

```
# dd if=/dev/sde of=image.raw
15466496+0 records in

15466496+0 records out
7918845952 bytes (7.9 GB) copied, 130.952 s, 60.5 MB/s
```

Здесь `if=` задает входной файл, которым в данном случае является необработанное дисковое устройство, подключенное к системе получения данных. Параметр `of=` задает выходной файл - обычный файл, содержащий необработанные данные, скопированные с дискового устройства.

После завершения `dd` сообщает число переданных байтов. Число переданных байтов можно разделить на размер сектора; результат должен точно совпасть с числом секторов, определенным при подключении устройства.

При использовании `dd` для криминалистического создания образа диска могут возникнуть трудности. Если в середине получения образа происходят ошибки чтения, `dd` прерывает работу с сообщением `Input/output error`. Эту проблему можно устранить, добавив `conv=noerror`, что заставит `dd` пропустить нечитаемый блок и продолжить работу. Проблема пропуска нечитаемых блоков состоит в том, что смещения секторов для блоков файловой системы в оставшейся части диска меняются в файле назначения, из-за чего остальная файловая система диска выглядит поврежденной. Для иллюстрации представьте страницы книги. Допустим, страница 99 вырвана. Если оглавление указывает на главу, начинающуюся со страницы 200, найти ее все равно можно. Номера страниц книги сохранились даже при отсутствии одной страницы. Но это не так, когда сектор 99 вырван из образа диска (из-за ошибки чтения). Все последующие сектора перенумеровываются, и "оглавление" файловой системы после сектора 99 будет указывать на неправильные блоки.

Параметр `sync` исправляет это, заполняя нечитаемый выходной блок нулями и фактически создавая "фиктивный" сектор или блок (полный нулей), представляющий отсутствующий блок. Тогда остальная часть образа диска будет иметь правильные номера секторов (смещения), ожидаемые содержащейся в нем файловой системой.

Использование предыдущего примера, но на этот раз с защитой от нечитаемых блоков (их пропуском и заполнением нулями), дает следующий результат:

```
# dd if=/dev/sde of=image.raw conv=noerror,sync
15466496+0 records in
15466496+0 records out
7918845952 bytes (7.9 GB) copied, 136.702 s, 57.9 MB/s
```

Заполнение выходных данных влияет на криминалистическое получение тем, что образ изменяется и в него добавляются новые данные (нули). Криптографические контрольные суммы диска не будут совпадать с исходными данными на диске (особенно если на диске появляются новые или изменяющиеся нечитаемые области). Этой проблемой можно управлять с помощью журналирования окон хеширования. Это обсуждается в разделе "Окна хеширования" на странице 152.

Еще одна проблема `dd` состоит в том, что размер блока передачи может превышать физический размер сектора носителя. При возникновении ошибки чтения это создает проблему, поскольку нулями заполняются оставшиеся сектора более крупного блока, а не только один нечитаемый сектор. Это означает, что некоторые нормальные читаемые сектора могут не попасть в криминалистический образ. Размер блока, превышающий размер сектора, также может привести к добавлению дополнительных заполняющих секторов в конец криминалистического образа (если размер образа не делится на размер блока). Потенциальный выигрыш в производительности от увеличения размера блока необходимо сопоставить с риском потери доказательств из-за большого заполненного блока.

Традиционный `dd` не имеет возможностей хеширования, журналирования в файл или других функций, ожидаемых от инструмента криминалистического получения данных. Поскольку необработанный образ не содержит метаданных об исходном исследуемом диске, любые сведения, описывающие диск, необходимо документировать отдельно (либо частично включать некоторые сведения в имя файла).

Инструменты dcfldd и dc3dd

Два популярных производных инструмента dd, dcfldd и dc3dd, были независимо разработаны специально для применения в криминалистическом контексте.

Поскольку dcfldd и dc3dd происходят от GNU dd, они используют сходный синтаксис команд. Ни один из этих инструментов не имеет встроенной поддержки записи в криминалистические форматы (FTK, EnCase, AFF), сжатия или шифрования образа. Но эти функции можно реализовать с помощью конвейеров команд, что я продемонстрирую в последующих разделах.

В следующем примере dcfldd используется для создания образа диска с обеспечением того, что блоки, содержащие нечитаемые сектора, заполняются и не вызывают прерывания:

```
# dcfldd if=/dev/sde of=image.raw conv=noerror,sync errlog=error.log
241664 blocks (7552Mb) written.
241664+0 records in
241664+0 records out
```

Ошибки записываются в отдельный файл журнала ошибок. Инструмент dcfldd не использует conv=noerror,sync по умолчанию; этот параметр необходимо добавить вручную.

Сходная команда создания образа с помощью dc3dd показана в следующем примере. По умолчанию dc3dd хорошо справляется с ошибками во время получения данных. Флаг conv=noerror,sync не требуется, поскольку его действие встроено. Вывод подробно документируется как в stdout, так и в файле журнала. Вот простой пример получения образа:

```
# dc3dd if=/dev/sde of=image.raw log=error.log
dc3dd 7.2.641 started at 2016-05-07 14:37:10 +0200
compiled options:
command line: dc3dd if=/dev/sde of=image.raw log=error.log
device size: 15466496 sectors (probed), 7,918,845,952 bytes
sector size: 512 bytes (probed)
7918845952 bytes ( 7.4 G ) copied ( 100% ), 80 s, 95 M/s
input results for device `/dev/sde':
15466496 sectors in
0 bad sectors replaced by zeros

output results for file `image.raw':
15466496 sectors out
dc3dd completed at 2016-05-07 14:38:30 +0200
```

Сценарий sfsimage также можно настроить на использование dcfldd или dc3dd для создания образа в криминалистическом контейнере SquashFS. В следующем примере образ накопителя с собственными секторами 4 Кбайт (собственный размер сектора 4096 байт) создается с помощью sfsimage:

```
# sfsimage -i /dev/sdd 4Knative.sfs
Started: 2016-05-07T17:16:54
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i /dev/sdd
Current working directory: /exam
Forensic evidence source: if=/dev/sdd
Destination squashfs container: 4Knative.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd if=/dev/sdd log=errorlog.txt hlog=hashlog.txt
hash=md5 2>/dev/null | pv -s 3000592982016
2.73TiB 5:29:31 [ 144MiB/s] [==========================================>] 100%
Completed: 2016-05-07T22:47:42

# cat /sys/block/sdd/queue/logical_block_size
4096

# cat /sys/block/sdd/queue/physical_block_size
4096
```

Этот пример также показывает, что физический и логический размеры сектора накопителя не влияют на получение образа при использовании инструментов создания образов в стиле dd.

У dcfldd и dc3dd имеются дополнительные функции криптографического хеширования, разделения образа и передачи данных внешним программам через конвейер. Я продемонстрирую эти функции в различных ситуациях на протяжении оставшейся части книги.

Получение образа в криминалистических форматах

Несколько форматов образов были специально разработаны с учетом потребностей криминалистики. Некоторые из них, например FTK и EnCase, являются коммерческими закрытыми форматами, которые были исследованы методом обратной разработки, что позволило создать совместимые инструменты с открытым исходным кодом. В следующих двух разделах описаны инструменты получения данных с использованием этих закрытых форматов.

Инструмент ewfacquire

Инструмент получения данных, специализирующийся на форматах Guidance EnCase Expert Witness, - ewfacquire из libewf (https://github.com/libyal/libewf/). Этот инструмент принимает информационные параметры в командной строке или запрашивает их в интерактивном режиме. Можно выбрать один из нескольких коммерческих форматов, включая различные форматы EnCase, а также FTK. Инструмент ewfacquire создает файлы получения данных, обеспечивающие совместимость с EnCase, FTK и Sleuth Kit. Инструмент также может преобразовывать необработанные образы в другие форматы.

В этом примере ewfacquire получает данные подключенного дискового устройства (MacBook Air, подключенного к рабочей станции эксперта в режиме целевого диска через адаптер Thunderbolt - FireWire):

```
# ewfacquire -c best -t /exam/macbookair /dev/sdf
ewfacquire 20160424

Device information:
Bus type: FireWire (IEEE1394)
Vendor:
Model:
Serial:

Storage media information:
Type: Device
Media type: Fixed
Media size: 121 GB (121332826112 bytes)
Bytes per sector: 512

Acquiry parameters required, please provide the necessary input
Case number: 42
Description: The case of the missing vase
Evidence number: 1
Examiner name: holmes
Notes: The vase was blue.
Media type (fixed, removable, optical, memory) [fixed]:
Media characteristics (logical, physical) [physical]:
Use EWF file format (ewf, smart, ftk, encase1, encase2, encase3, encase4, encase5,
encase6, encase7, encase7-v2, linen5, linen6, linen7, ewfx) [encase6]:
Start to acquire at offset (0 <= value <= 121332826112) [0]:
The number of bytes to acquire (0 <= value <= 121332826112) [121332826112]:
Evidence segment file size in bytes (1.0 MiB <= value <= 7.9 EiB) [1.4 GiB]:
```

```
The number of bytes per sector (1 <= value <= 4294967295) [512]:
The number of sectors to read at once (16, 32, 64, 128, 256, 512, 1024, 2048, 4096,
8192, 16384, 32768) [64]:
The number of sectors to be used as error granularity (1 <= value <= 64) [64]:
The number of retries when a read error occurs (0 <= value <= 255) [2]:
Wipe sectors on read error (mimic EnCase like behavior) (yes, no) [no]:

The following acquiry parameters were provided:
Image path and filename: /exam/macbookair.E01
Case number: 42
Description: The case of the missing vase
Evidence number: 1
Examiner name: holmes
Notes: The vase was blue.

Media type: fixed disk
Is physical: yes
EWF file format: EnCase 6 (.E01)
Compression method: deflate
Compression level: best
Acquiry start offset: 0
Number of bytes to acquire: 113 GiB (121332826112 bytes)
Evidence segment file size: 1.4 GiB (1572864000 bytes)
Bytes per sector: 512
Block size: 64 sectors
Error granularity: 64 sectors
Retries on read error: 2
Zero sectors on read error: no
Continue acquiry with these values (yes, no) [yes]:

Acquiry started at: May 07, 2016 14:54:52
This could take a while.

Status: at 0.0%
acquired 60 MiB (62914560 bytes) of total 113 GiB (121332826112 bytes)
completion in 2 hour(s), 8 minute(s) and 38 second(s) with 14 MiB/s
(15712616 bytes/second)
...
Status: at 99.9%
acquired 112 GiB (121329188864 bytes) of total 113 GiB (121332826112 bytes)
completion in 0 second(s) with 51 MiB/s (54069886 bytes/second)

Acquiry completed at: May 07, 2016 15:32:16

Written: 113 GiB (121332826300 bytes) in 37 minute(s) and 24 second(s) with
51 MiB/s (54069886 bytes/second)
MD5 hash calculated over data: 083e2131d0a59a9e3b59d48dbc451591
ewfacquire: SUCCESS
```

Получение данных с помощью `ewfacquire` успешно завершилось за 37 минут, а файл размером 120 Гбайт был разделен на 54 сжатых файла `*.E0` общим размером 79 Гбайт.

AccessData ftkimager

AccessData предоставляет бесплатные предварительно скомпилированные версии FTK Imager для командной строки. Инструмент называется `ftkimager`; двоичные файлы (без исходного кода) доступны для Debian Linux, Fedora Linux, OS X и Windows. Их можно загрузить с сайта AccessData по адресу http://accessdata.com/product-download/digital-forensics/.

Инструмент `ftkimager` может получать входные данные с необработанного устройства, из файла или из `stdin`. Выходные данные он записывает в формат FTK SMART, формат EnCase EWF или в `stdout`. Потоки `stdin` и `stdout` особенно полезны для конвейерной передачи данных другим программам и от них. Поддерживается и ряд других функций, включая добавление метаданных

дела в сохраняемые форматы, сжатие, разделение выходного файла ("фрагменты образа"), хеширование и зашифрованные образы.

В следующем базовом примере показано применение `ftkimager` для получения образа подключенного диска:

```
# ftkimager /dev/sdf --s01 --description "SN4C53000120 Ultra Fit" sandisk
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

В этом примере исходным устройством был флеш-накопитель SanDisk, доступный как `/dev/sdf`, а именем файла назначения было `sandisk`. Поскольку по умолчанию используется необработанный формат, добавление флага `--s01` сохраняет данные в формате FTK SMART. С помощью флага `--description` в метаданные были добавлены серийный номер и строка модели.

`ftkimager` создает файл журнала с основными метаданными и дополнительными сведениями, добавленными с помощью флагов командной строки, как показано здесь:

```
# cat sandisk.s01.txt
Case Information:
Acquired using: ADI3
Case Number:
Evidence Number:
Unique description: SN4C53000120 Ultra Fit
Examiner:
Notes:
---------------------------------------------------------------

Information for sandisk:

Physical Evidentiary Item (Source) Information:
[Device Info]
Source Type: Physical
[Drive Geometry]
Cylinders: 14832
Heads: 64
Sectors per Track: 32
Bytes per Sector: 512
Sector Count: 30375936
Source data size: 14832 MB
Sector count: 30375936

[Computed Hashes]

MD5 checksum: a2a9a891eed92edbf47ffba9f4fad402
SHA1 checksum: 2e73cc2a2c21c9d4198e93db04303f9b38e0aefe

Image Information:
Acquisition started: Sat May 7 15:49:07 2016
Acquisition finished: Sat May 7 15:53:07 2016
Segment list:
sandisk.s01
sandisk.s02
```

Те же сведения можно извлечь с помощью флага `--print-info`, передав вместе с ним имя файла.

Контейнер криминалистических доказательств SquashFS

Инструмент `sfsimage` представляет собой просто сценарий-обертку оболочки, который можно настроить на использование любого инструмента создания образов, поддерживающего корректную запись образа в `stdout`. Сценарий принимает этот поток байтов образа и помещает их внутрь сжатой файловой системы SquashFS.

В этом примере `sfsimage` был настроен на использование `dc3dd` в качестве инструмента создания образов путем изменения переменной `DD` в начале сценария оболочки:

```
DD="dc3dd if=$DDIN log=errorlog.txt hlog=hashlog.txt hash=md5"
```

Затем образ блочного устройства создается с помощью флага `-i`:

```
$ sfsimage -i /dev/sde philips-usb-drive.sfs
Started: 2016-05-07T15:40:03
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i /dev/sde
Current working directory: /exam
Forensic evidence source: if=/dev/sde
Destination squashfs container: philips-usb-drive.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd if=/dev/sde log=errorlog.txt hlog=hashlog.txt
hash=md5 2>/dev/null | pv -s 7918845952
7.38GiB 0:01:18 [95.7MiB/s] [=====================================>] 100%
Completed: 2016-05-07T15:41:22
```

В следующем выводе показан размер сжатого файла `*.sfs`:

```
$ ls -lh *.sfs
-rw-r----- 1 holmes holmes 4.5G May 7 15:41 philips-usb-drive.sfs
```

Содержимое файла-контейнера SquashFS можно перечислить командой `sfsimage -l` или смонтировать его (только для чтения) с помощью `sfsimage -m`. В процессе получения данных `sfsimage` сохраняет журнал ошибок, журнал хешей и собственный журнал вместе с файлом необработанного образа. Дополнительные файлы можно добавить в контейнер `sfsimage` с помощью `sfsimage -a`.

Получение образа в несколько мест назначения

Гибкость механизма конвейеров Unix позволяет выполнить несколько сложных задач за один автоматический этап. И `dc3dd`, и `dcfldd` позволяют указать несколько имен файлов назначения, благодаря чему можно одновременно создавать несколько копий образа. В следующем примере показано создание образа диска с одновременной записью в несколько мест назначения: локальной копии в основной системе получения данных и второй копии на смонтированном внешнем диске третьей стороны. Эти два выходных файла задаются несколькими флагами `of=` следующим образом:

```
# dc3dd if=/dev/sde of=/exam/local-lab.raw of=/ext/third-party.raw
dc3dd 7.2.641 started at 2016-05-07 15:56:10 +0200
compiled options:
command line: dc3dd if=/dev/sde of=/exam/local-lab.raw of=/ext/third-party.raw
device size: 15466496 sectors (probed), 7,918,845,952 bytes
sector size: 512 bytes (probed)
7918845952 bytes ( 7.4 G ) copied ( 100% ), 79 s, 95 M/s
input results for device `/dev/sde':
15466496 sectors in
0 bad sectors replaced by zeros
output results for file `/exam/local-lab.raw':
15466496 sectors out
output results for file `/ext/third-party.raw':
15466496 sectors out
dc3dd completed at 2016-05-07 15:57:30 +0200
```

Этот метод полезен, если один образ создается для анализа, а другой - для резервного копирования, если дополнительный образ создается для третьей стороны или в любой другой ситуации, когда требуется несколько копий образа. Эти два образа должны быть идентичными, что можно проверить сравнением криптографических контрольных сумм.

Сохранение цифровых доказательств средствами криптографии

Сохранение целостности доказательств имеет основополагающее значение для процесса цифровой криминалистики. Целостность можно поддерживать с помощью криптографических хешей и дополнительно усилить криптографическими подписями технических специалистов, выполнивших получение данных. Цель хеширования или подписания образов состоит в том, чтобы проверить, что образ не изменился с момента его получения. Поскольку судебное разбирательство и представление доказательств могут занять месяцы или даже годы, полезно подтвердить, что за это время доказательства не были изменены. Это можно рассматривать как своего рода цифровую цепочку хранения.

В следующих нескольких разделах демонстрируется применение окон хеширования, подписания с помощью PGP и S/MIME и временных меток RFC-3161 для сохранения цифровых доказательств. Начнем с примеров базового криптографического хеширования.

Базовое криптографическое хеширование

Криптографическое хеширование криминалистических образов обычно включается в процесс создания образа. Весь образ носителя (каждый сектор по порядку) пропускается через одностороннюю хеш-функцию. На момент написания книги четыре основных инструмента создания криминалистических образов, рассматриваемые в книге, поддерживали криптографические алгоритмы хеширования, показанные в табл. 6-1.

Таблица 6-1. Поддерживаемые криптографические алгоритмы хеширования

Инструмент	Поддерживаемые алгоритмы хеширования
`dcfldd`	MD5, SHA1, SHA256, SHA384, SHA512
`dc3dd`	MD5, SHA1, SHA256, SHA512
`ewfacquire`	MD5, SHA1, SHA256
`ftkimager`	MD5, SHA1

Инструменты, использующие криминалистические форматы, обычно создают хеш по умолчанию. И `ftkimager`, и `ewfacquire` автоматически создают хеши в процессе получения данных, как вы видели в предыдущих примерах.

Чтобы создать один или несколько хешей с помощью `dcfldd`, укажите нужные алгоритмы хеширования в командной строке следующим образом:

```
# dcfldd if=/dev/sde of=image.raw conv=noerror,sync hash=md5,sha256
241664 blocks (7552Mb) written.Total (md5): ebda11ffb776f183325cf1d8941109f8
Total (sha256): 792996cb7f54cbfd91b5ea9d817546f001f5f8ac05f2d9140fc0778fa60980a2
241664+0 records in
241664+0 records out
```

В `dc3dd` алгоритмы хеширования указываются многократным применением `hash=`, как показано здесь:

```
# dc3dd if=/dev/sde of=image.raw hash=md5 hash=sha1 hash=sha512
dc3dd 7.2.641 started at 2016-05-07 16:02:56 +0200
compiled options:
command line: dc3dd if=/dev/sde of=image.raw hash=md5 hash=sha1 hash=sha512
device size: 15466496 sectors (probed), 7,918,845,952 bytes
sector size: 512 bytes (probed)
7918845952 bytes ( 7.4 G ) copied ( 100% ), 80 s, 94 M/s

input results for device `/dev/sde':
15466496 sectors in
0 bad sectors replaced by zeros
ebda11ffb776f183325cf1d8941109f8 (md5)
62e5045fbf6a07fa77c48f82eddb59dfaf7d4d81 (sha1)
f0d1132bf569b68d900433aa52bfc08da10a4c45f6b89847f244834ef20bb04f8c35dd625a31c2e3
a29724e18d9abbf924b16d8f608f0ff0944dcb35e7387b8d (sha512)
output results for file `image.raw':
15466496 sectors out
dc3dd completed at 2016-05-07 16:04:17 +0200
```

Традиционная команда `dd` не поддерживает хеширование. Вместо этого в процессе получения данных образ необходимо передать по конвейеру отдельной программе, что можно сделать командой Unix `tee`:

```
# dd if=/dev/sde | tee image.raw | md5sum
15466496+0 records in
15466496+0 records out
7918845952 bytes (7.9 GB, 7.4 GiB) copied, 108.822 s, 72.8 MB/s
ebda11ffb776f183325cf1d8941109f8 -
```

Если для `dd` не указан `of=`, данные отправляются в `stdout`, откуда их можно перенаправить или передать по конвейеру другой программе. В этом примере они передаются по конвейеру команде Unix `tee`, которая одновременно сохраняет данные в файл и отправляет их в `stdout`. Затем данные передаются независимому инструменту хеширования `md5sum`, который создает хеш. Помимо `md5sum`, пакет программ Linux `coreutils` включает другие программы хеширования: `sha1sum`, `sha224sum`, `sha256sum`, `sha384sum` и `sha512sum`.

Процесс проверки созданных хешей объясняется в разделе "Проверка целостности криминалистического образа" на странице 197.

Окна хеширования

При создании образа старого или поврежденного диска могут возникать ошибки чтения блоков. Эти ошибки могут появляться в случайных местах в процессе получения, а их частота со временем может возрастать. Это создает трудность для сохранения целостности доказательств, поскольку при каждом чтении диска (повторном получении, дублировании, проверке и так далее) криптографический хеш может отличаться.

Решение этой проблемы - применение окон хеширования, или фрагментарного хеширования. Окно хеширования представляет собой отдельный криптографический хеш меньшей последовательности секторов диска. Например, размер окна хеширования 10 Мбайт во время получения образа создает отдельный хеш для каждой последовательности секторов размером 10 Мбайт и список хешей диска. Если один сектор становится нечитаемым (или по какой-либо причине изменяется), хеш этого окна будет недействительным. Но все остальные окна хеширования на диске сохранят свою целостность. Поэтому даже если хеш всего диска недействителен, совпадение окна хеширования сохранит подтверждение целостности данных внутри него.

Среди коммерческих криминалистических форматов ранние версии Expert Witness Format (EWF) использовали для отдельных блоков данных только контрольные суммы циклического избыточного кода (CRC). Более новые версии не являются открытыми форматами, а `ftkimager` не имеет параметров для создания или просмотра окон хеширования.

Чтобы создать окна хеширования с помощью `dcfldd`, необходимо добавить параметр `hashwindow=`, задающий размер окна. Список окон хеширования можно сохранить в файл во время получения данных с помощью параметра `hashlog=` с именем файла. В следующем примере задан размер окна хеширования 1 Мбайт, а хеши каждого диапазона секторов записываются в `stdout`:

```
# dcfldd if=/dev/sde of=image.raw conv=noerror,sync hashwindow=1M
0 - 1048576: e0796359399e85ecc03b9ca2fae7b9cf
1048576 - 2097152: 5f44a2407d244c24e261b00de65949d7
2097152 - 3145728: d6d8c4ae64b464dc77658730aec34a01
3145728 - 4194304: 0eae942f041ea38d560e26dc3cbfac48
4194304 - 5242880: 382897281f396b70e76b79dd042cfa7f
5242880 - 6291456: 17664a919d533a91df8d26dfb3d84fb9
6291456 - 7340032: ce29d3ca2c459c311eb8c9d08391a446
7340032 - 8388608: cd0ac7cbbd58f768cd949b082de18d55
256 blocks (8Mb) written.8388608 - 9437184: 31ca089fce536aea91d957e070b189d8
9437184 - 10485760: 48586d6dde4c630ebb168b0276bec0e3
10485760 - 11534336: 0969f7533736e7a2ee480d0ca8d9fad1
...
```

Группы одинаковых секторов диска будут иметь одинаковое значение хеша. Это часто происходит, когда большие участки диска заполнены нулями или повторяющимся шаблоном.

В dc3dd окна хеширования называются фрагментарным хешированием, а хеши можно создавать не по диапазонам секторов, а для каждого разделенного файла. В следующем примере хеши диапазонов секторов в каждом разделенном файле записываются в журнал:

```
# dc3dd if=/dev/sda hof=image.raw ofs=image.000 ofsz=1G hlog=hash.log hash=md5
dc3dd 7.2.641 started at 2016-05-07 17:10:31 +0200
compiled options:
command line: dc3dd if=/dev/sda hof=image.raw ofs=image.000 ofsz=1G hlog=hash.log
hash=md5
device size: 15466496 sectors (probed), 7,918,845,952 bytes
sector size: 512 bytes (probed)
7918845952 bytes ( 7.4 G ) copied ( 100% ), 114 s, 66 M/s
7918845952 bytes ( 7.4 G ) hashed ( 100% ), 24 s, 314 M/s
input results for device `/dev/sda':
15466496 sectors in
0 bad sectors replaced by zeros
5dfe68597f8ad9f20600a453101f2c57 (md5)

c250163554581d94958018d8cca61db6, sectors 0 - 2097151
cd573cfaace07e7949bc0c46028904ff, sectors 2097152 - 4194303
83d63636749194bcc7152d9d1f4b9df1, sectors 4194304 - 6291455
da961f072998b8897c4fbed4c0f74e0e, sectors 6291456 - 8388607
4cd5560038faee09da94a0c829f07f7a, sectors 8388608 - 10485759
516ba0bdf8d969fd7e86cd005c992600, sectors 10485760 - 12582911
c19f8c710088b785c3f2ad2fb636cfcd, sectors 12582912 - 14680063
fb2eb5b178839878c1778453805b8bf6, sectors 14680064 - 15466495
output results for file `image.raw':
15466496 sectors out
[ok] 5dfe68597f8ad9f20600a453101f2c57 (md5)
output results for files `image.000':
15466496 sectors out
dc3dd completed at 2016-05-07 17:12:25 +0200
```

Если имеется только один файл образа (то есть он не разделен), отдельных окон хеширования нет - есть лишь один хеш всего образа. В предыдущем примере были созданы восемь файлов образа, а хеши MD5 каждого файла совпадают с указанными во время получения данных. Это легко подтвердить с помощью md5sum следующим образом:

```
# md5sum image.*
c250163554581d94958018d8cca61db6 image.000
cd573cfaace07e7949bc0c46028904ff image.001
83d63636749194bcc7152d9d1f4b9df1 image.002
da961f072998b8897c4fbed4c0f74e0e image.003
4cd5560038faee09da94a0c829f07f7a image.004
516ba0bdf8d969fd7e86cd005c992600 image.005
c19f8c710088b785c3f2ad2fb636cfcd image.006
fb2eb5b178839878c1778453805b8bf6 image.007
```

Подписание образа с помощью PGP или S/MIME

Значение хеша полезно для сохранения целостности образа с течением времени, но любой человек может в любой момент вычислить криптографический хеш образа. Представьте диск, измененный неуполномоченным лицом, которое создало новый хеш образа диска. Если исходный хеш не был надлежащим образом защищен во время первоначального получения данных, трудно доказать, какой хеш (старый или новый) является правильным. Криптографическое подписание криминалистических образов связывает человека (или его ключ) с целостностью образа. Эксперт-криминалист, руководитель или внешняя нейтральная сторона могут подписать образ во время получения данных.

Это не означает, что многотерабайтные образы необходимо передавать людям для подписания. Достаточно подписать хеш накопителя или список окон хеширования. Лучший вариант - подписать весь журнал вывода, содержащий временные метки, число полученных байтов и все результирующие криптографические хеши.

Подобно тому как уполномоченное лицо подписывает бумажные формы ручкой, оно может подписывать цифровые формы цифровой подписью. В отличие от подписей ручкой на бумаге, цифровые подписи трудно подделать (если только закрытый ключ не был похищен). Два популярных стандарта подписания цифровых данных - Pretty Good Privacy (PGP) и Secure/Multipurpose Internet Mail Extensions (S/MIME).

Наиболее распространенная реализация стандарта OpenPGP в Linux - GnuPG (GPG).[1] Существует три разных способа подписания: обычная двоичная подпись, подпись открытого текста и отсоединенная подпись. Наиболее полезно применять подпись открытого текста, поскольку она показывает текст вместе с подписью и может быть легко встроена в другие документы и отчеты.

В следующем примере Ш. Холмс выполнил криминалистическое получение образа диска и подписывает журнал вывода, содержащий хеш MD5 и другие сведения:

```
$ gpg --clearsign hash.log
You need a passphrase to unlock the secret key for
user: "Sherlock Holmes <holmes@digitalforensics.ch>"
2048-bit RSA key, ID CF87856B, created 2016-01-11

Enter passphrase:
```

Предыдущая команда создала файл `hash.log.asc`, содержащий содержимое файла вместе с подписью:

```
$ cat hash.log.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

dc3dd 7.2.641 started at 2016-05-07 17:23:49 +0200
compiled options:
command line: dc3dd if=/dev/sda hof=image.raw ofs=image.000 ofsz=1G hlog=hash.log
hash=md5
input results for device `/dev/sda':
5dfe68597f8ad9f20600a453101f2c57 (md5)
c250163554581d94958018d8cca61db6, sectors 0 - 2097151
cd573cfaace07e7949bc0c46028904ff, sectors 2097152 - 4194303
83d63636749194bcc7152d9d1f4b9df1, sectors 4194304 - 6291455

da961f072998b8897c4fbed4c0f74e0e, sectors 6291456 - 8388607
4cd5560038faee09da94a0c829f07f7a, sectors 8388608 - 10485759
516ba0bdf8d969fd7e86cd005c992600, sectors 10485760 - 12582911
c19f8c710088b785c3f2ad2fb636cfcd, sectors 12582912 - 14680063
fb2eb5b178839878c1778453805b8bf6, sectors 14680064 - 15466495
output results for file `image.raw':
[ok] 5dfe68597f8ad9f20600a453101f2c57 (md5)
output results for files `image.000':
dc3dd completed at 2016-05-07 17:25:40 +0200
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1

iQEcBAEBAgAGBQJXLgnoAAoJEEg0vvzPh4VrdeAH/0EhCLFSWwTZDNUrIn++1rI3
XI6KuwES19EKR18PrK/Nhf5MsF3xyy3c/j7tjopkfnDGLYRA615ycWEvIJlevNh7
k7QHJoPTDnyJcF29uuTINPWk2MsBlkNdTTiyA6ab3U4Qm+DMC4wVKpOp/io52qq3
KP7Kh558aw8m+0Froc0/4sF7rer9xvBThA2cw+ZiyF5a8wTCBmavDchCfWm+NREr
RIncJV45nuHrQW8MObPOK6G34mruT9nSQFH1LR1FL830m/W69WHS2JX+shfk5g5X
I6I7jNEn6FgiRyhm+BizoSl5F6mv3ff6mRlVysGDJ+FXE3CiE6ZzK+jNB7Pw+Zg=
=6GrG
-----END PGP SIGNATURE-----
```

Этот подписанный текст позднее может проверить любая третья сторона, располагающая копией открытого ключа GPG Холмса.

Другой стандарт шифрования, который можно применять для подписания файлов, - S/MIME. Применение S/MIME опирается на сертификаты X.509 из инфраструктуры открытых ключей (PKI), частной внутри организации или предоставляемой общедоступным центром сертификации (CA). Если у уполномоченного лица есть персональный сертификат (обычно тот же, который оно использует для подписания и шифрования электронной почты S/MIME), его можно применить для подписания файлов, содержащих сведения о получении данных.

Инструмент `gpgsm` входит в GnuPG2 и поддерживает управление ключами X.509, шифрование и подписи по стандарту S/MIME. После создания необходимых ключей и установки сертификатов `gpgsm` можно использовать для подписания файлов сходным с GPG образом. Следующая команда создает подпись указанного файла:

```
$ gpgsm -a -r holmes@digitalforensics.ch -o hash.log.pem --sign hash.log
```

Флаг `-a` указывает использовать ASCII armor - способ кодирования двоичных данных в текстовом формате - вместо двоичного представления (поскольку такой текст легче копировать в отчеты или электронные письма). Флаг `-r` указывает, какой ключ получателя использовать для подписания. В этом примере команды используется адрес электронной почты, но ключ также можно задать идентификатором, отпечатком или совпадающими компонентами строк X.509. Параметр `-o` задает выходной файл подписи, а `--sign` предписывает `gpgsm` создать подпись указанного файла `hash.log`.

При использовании для подписания `gpgsm` создаст файл подписи PEM[2], похожий на следующий:

```
-----BEGIN SIGNED MESSAGE-----
MIAGCSqGSIb3DQEHAqCAMIACAQExDzANBglghkgBZQMEAgEFADCABgkqhkiG9w0B
BwGggCSABIICIApkYzNkZCA3LjIuNjQxIHN0YXJ0ZWQgYXQgMjAxNi0wMS0xMSAy
...
GR2YC4Mx5xQ63Kbxg/5BxT7rlC7DBjHOVMCMJzVPy4OVUOXPnL2IdP2dhvkOtojk
UKIjSw40xIIAAAAAAAA=
-----END SIGNED MESSAGE-----
```

После создания подписи уполномоченной стороной значения хешей и сведения об исходном криминалистическом получении данных изменить нельзя. Внести изменения и снова подписать их может только лицо, создавшее подпись.[3] Эти подписи позволяют проверять целостность сведений о получении данных без участия подписавшего их лица. Процесс проверки подписи описан в главе 7.

Персональные сертификаты S/MIME можно приобрести подобно сертификатам SSL для сайтов. Обзор центров сертификации, предлагающих персональные сертификаты S/MIME, находится по адресу https://www.sslshopper.com/email-certificates-smime-certificates.html. С помощью персонального сертификата S/MIME сведения о получении данных можно также подписать, просто отправив подписанное электронное сообщение с содержимым журнала вывода.

Примеры в этом разделе просты и используют инструменты GNU Privacy Guard. Существуют и другие инструменты командной строки для криптографического подписания. Инструмент командной строки OpenSSL предоставляет богатый криптографический набор, включающий возможность подписывать файлы с помощью сертификатов X.509 и S/MIME. В следующем разделе OpenSSL используется для демонстрации криптографического присвоения временных меток.

Присвоение временных меток RFC-3161

Подписи PGP или S/MIME прочно связывают уполномоченное лицо (или нескольких лиц) с целостностью файла, содержащего результаты криминалистического получения данных. В некоторых случаях полезно также прочно связать результаты криминалистического получения данных с определенным моментом времени. Это можно сделать с помощью независимой службы временных меток.

Присвоение временных меток - формальный стандарт, определенный в RFC-3161, который описывает формат запроса и ответа временной метки. OpenSSL может создавать и отправлять запросы временных меток и проверять ответы. В следующем примере создается совместимый с RFC-3161 запрос временной метки для журнала получения данных, создающий файл запроса с расширением `*.tsq`:

```
$ openssl ts -query -data hash.log -out hash.log.tsq -cert
```

Этот запрос временной метки содержит хеш файла `hash.log`, а не сам файл. Файл не отправляется на сервер временных меток. Это важно с точки зрения информационной безопасности. Поставщику службы временных меток доверяются только сведения о времени, а не содержимое файлов, которым присваиваются временные метки.

Созданный запрос затем можно отправить службе временных меток с помощью команды `tsget`, входящей в OpenSSL.[4] В следующем примере используется служба FreeTSA:

```
$ tsget -h https://freetsa.org/tsr hash.log.tsq
```

В некоторых дистрибутивах Linux этот сценарий может отсутствовать или быть неисправным. Обойти проблему можно, вручную отправив запрос временной метки командой `curl` следующим образом:

```
$ curl -s -H "Content-Type: application/timestamp-query" --data-binary
"@hash.log.tsq" https://freetsa.org/tsr > hash.log.tsr
```

Если сервер временных меток принимает запрос, он возвращает временную метку, совместимую с RFC-3161. В этом примере временная метка сохраняется в файле `hash.log.tsr` с расширением `*.tsr`. Содержимое временной метки можно просмотреть командой OpenSSL `ts`:

```
$ openssl ts -reply -in hash.log.tsr -text
Status info:
Status: Granted.
Status description: unspecified
Failure info: unspecified

TST info:
Version: 1
Policy OID: 1.2.3.4.1
Hash Algorithm: sha1
Message data:
0000 - 63 5a 86 52 01 24 72 43-8e 10 24 bc 24 97 d0 50 cZ.R.$rC..$.$..P
0010 - 4a 69 ad a9 Ji..
Serial number: 0x0AF4
Time stamp: May 7 22:03:49 2016 GMT
Accuracy: 0x01 seconds, 0x01F4 millis, 0x64 micros
Ordering: yes

Nonce: 0xBC6F68553A3E5EF5
TSA: DirName:/O=Free TSA/OU=TSA/description=This certificate digitally signs
documents and time stamp requests made using the freetsa.org online
services/CN=www.freetsa.org/emailAddress=busilezas@gmail.com/L=Wuerzburg/
C=DE/ST=Bayern
Extensions:
```

Копия файла `hash.log.tsr` служит доказательством того, что результаты получения данных существовали в определенный момент времени. Независимая третья сторона также может проверить действительность временной метки. Проверка временных меток будет продемонстрирована в главе 7.

В Интернете доступен ряд бесплатных и коммерческих служб временных меток. Вот несколько примеров:

- Comodo RFC-3161 Timestamping Service: http://timestamp.comodoca.com/?td=sha256
- FreeTSA: http://freetsa.org/index_en.php
- Polish CERTUM PCC - General Certification Authority: http://time.certum.pl/
- Safe Creative Timestamping Authority (TSA) server: http://tsa.safecreative.org/
- StartCom Free RFC-3161 Timestamping Service: http://tsa.startssl.com/rfc3161
- Zeitstempeldienst der DFN-PKI: http://www.pki.dfn.de/zeitstempeldienst/

Примеры в двух последних разделах прочно связывают человека и момент времени с целостностью образа. Для защиты закрытых ключей и гарантии физического владения токеном при подписании образа можно применять криптографические токены, например смарт-карты или аппаратные модули безопасности (HSM). Криптографические ключи на аппаратных токенах нельзя скопировать или похитить. Примеры аппаратных токенов, которые можно использовать для создания криптографических подписей, включают Nitrokey, Yubikey и смарт-карты GnuPG OpenPGP.

Работа с отказами и ошибками накопителей

Иногда криминалистическая лаборатория получает для анализа проблемный жесткий диск. Диск может быть старым, поврежденным или отказывающим. У него могут возникать ошибки интерфейса, ошибки чтения пластин, ошибки головок, сбросы двигателя и другие ошибки. В некоторых случаях частичный криминалистический образ накопителя все же можно получить. В зависимости от размера диска, размера блока и числа нечитаемых секторов создание образа неисправного диска может занять несколько дней.

Важно понимать, что описанные здесь ошибки относятся к аппаратной части накопителя. Они не относятся к программным ошибкам, таким как поврежденные файловые системы, уничтоженные таблицы разделов и так далее.

В этом разделе на примерах показано, как разные инструменты обрабатывают состояния ошибок. Инструмент `dmsetup` полезен для имитации ошибок диска и проверки поведения криминалистических инструментов при различных видах отказов; он использовался в нескольких последующих примерах (дисковое устройство - `/dev/mapper/errdisk`). В следующем разделе приведен обзор того, как `dc3dd`, `dcfldd`, `ewfacquire` и `ftkimager` обрабатывают ошибки и сообщают о них.

Обработка ошибок криминалистическими инструментами

В следующем примере инструмент `dcfldd` встречает диск с двумя ошибками. Места ошибок на диске (смещения блоков) выводятся в `stdout` и записываются в указанный файл следующим образом:

```
# dcfldd if=/dev/mapper/errdisk of=errdisk.raw conv=noerror,sync errlog=error.log
...

# cat error.log
dcfldd:/dev/mapper/errdisk: Input/output error
(null)+15 records in
(null)+16 records out
dcfldd:/dev/mapper/errdisk: Input/output error
(null)+29 records in
(null)+32 records out
(null)+62496 records in
(null)+62501 records out
```

При испытании `dcfldd` в Debian Linux обнаружилось несколько ошибок программы. Размер блока для заполнения оставался равным 4 Кбайт, даже когда был указан размер блока 512 байт (`dd` вел себя так же). При некоторых ошибках `dcfldd` входил в бесконечный цикл, и его приходилось завершать вручную.

Инструмент `dc3dd` предоставляет очень подробный обзор встреченных ошибок. Ошибки отправляются в `stdout` и сохраняются в указанном файле журнала следующим образом:

```
# dc3dd if=/dev/mapper/errdisk of=errdisk.raw log=error.log
...

# cat error.log
dc3dd 7.2.641 started at 2016-01-12 19:42:26 +0100
compiled options:
command line: dc3dd if=/dev/mapper/errdisk of=errdisk.raw log=error.log
device size: 4000000 sectors (probed), 2,048,000,000 bytes
sector size: 512 bytes (probed)
[!!] reading `/dev/mapper/errdisk' at sector 1000 : Input/output error
[!!] 4 occurences while reading `/dev/mapper/errdisk' from sector 2001 to sector 2004
: Input/output error
2048000000 bytes ( 1.9 G ) copied ( 100% ), 5.74919 s, 340 M/s
input results for device `/dev/mapper/errdisk':
4000000 sectors in
5 bad sectors replaced by zeros

output results for file `errdisk.raw':
4000000 sectors out
dc3dd completed at 2016-01-12 19:42:31 +0100
```

Инструмент `ewfacquire` по умолчанию предлагает гранулярность ошибок 64 сектора; ее можно изменить на 1, чтобы уменьшить число секторов, заполняемых нулями. В этом примере `ewfacquire` обнаружил только две ошибки чтения (подобно `dcfldd`, он пропустил и заполнил блок 4 Кбайт, не проверяя остальные сектора):

```
# ewfacquire -t errdisk /dev/mapper/errdisk
ewfacquire 20150126
...
The number of bytes per sector (1 <= value <= 4294967295) [512]:
The number of sectors to read at once (16, 32, 64, 128, 256, 512, 1024, 2048, 4096,
8192, 16384, 32768) [64]:
The number of sectors to be used as error granularity (1 <= value <= 64) [64]: 1
The number of retries when a read error occurs (0 <= value <= 255) [2]: 1
Wipe sectors on read error (mimic EnCase like behavior) (yes, no) [no]: yes
...
Acquiry completed at: Jan 12, 2016 19:57:58
Written: 1.9 GiB (2048000804 bytes) in 14 second(s) with 139 MiB/s (146285771
```

```
bytes/second).
Errors reading device:
total number: 2
at sector(s): 1000 - 1008 number: 8 (offset: 0x0007d000 of size: 4096)
at sector(s): 2000 - 2008 number: 8 (offset: 0x000fa000 of size: 4096)
MD5 hash calculated over data: 4d319b12088b3990bded7834211308eb
ewfacquire: SUCCESS
```

`ftkimager` сообщает об ошибках и записывает их в журнал. В следующем примере используется реальный физически неисправный диск (оригинальный iPod первого поколения), поскольку `ftkimager` не работал с имитированными ошибками, созданными с помощью `dmsetup`:

```
# ftkimager /dev/sdg ipod
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
234.25 / 4775.76 MB (11.71 MB/sec) - 0:06:27 left
Image creation complete.

# cat ipod.001.txt
Case Information:
Acquired using: FTK
...

ATTENTION:
The following sector(s) on the source drive could not be read:
491584 through 491591
491928 through 491935
The contents of these sectors were replaced with zeros in the image.
...
```

Каждый из инструментов криминалистического получения обладал некоторыми возможностями обнаружения, обработки и журналирования ошибок. Однако для дисков со значительным числом ошибок или аппаратным повреждением более уместными могут быть специализированные инструменты. В следующем разделе описывается применение для этой цели инструментов восстановления данных.

Инструменты восстановления данных

Следует упомянуть несколько инструментов восстановления дисковых блоков благодаря надежной обработке ошибок и агрессивным методам восстановления. Хотя эти инструменты создавались не для криминалистики, они полезны в ситуациях, когда другие криминалистические инструменты потерпели неудачу.

Инструмент `ddrescue` (автор Antonio Diaz Diaz) разработан для восстановления блоков с поврежденных дисков. В отличие от семейства инструментов `dd`, он использует многоэтапный алгоритм восстановления и может запускаться для диска несколько раз, чтобы заполнить пробелы в образе. Алгоритм включает чтение проблемных участков диска в обратном направлении для увеличения числа восстановленных секторов и различные повторные операции за несколько проходов.

Завершенная операция ddrescue создает статистику, описывающую долю успешно восстановленных данных:

```
# ddrescue /dev/sda image.raw image.log
rescued: 40968 MB, errsize: 2895 kB, current rate: 0 B/s
ipos: 39026 MB, errors: 38, average rate: 563 kB/s
opos: 39026 MB, run time: 20.18 h, successful read: 8.04 h ago
Finished
```

Файл журнала, создаваемый ddrescue, показывает время начала и окончания и подробный обзор проблемных областей диска:

```
# Rescue Logfile. Created by GNU ddrescue version 1.19
# Command line: ddrescue /dev/sda image.raw image.log
# Start time: 2015-06-13 22:57:39
# Current time: 2015-06-14 19:09:03
# Finished
# current_pos current_status
0x9162CAC00 +
# pos size status
0x00000000 0x4F29D000 +
0x4F29D000 0x00002000 -

0x4F29F000 0x00253000 +
...
```

Инструмент dd_rescue (обратите внимание на символ подчеркивания) был разработан Куртом Гарлоффом в конце 1990-х годов. Хотя его имя содержит dd, синтаксис команды совершенно иной, и преобразования данных он не выполняет (как и ddrescue). Но он передает блоки данных подобно dd. Несколько функций делают этот инструмент возможным вариантом для цифровой криминалистической лаборатории. При возникновении ошибок диска размер блока динамически изменяется, автоматически уменьшаясь до размера физического блока. После периода без ошибок размер блока снова изменяется для повышения производительности. Можно также создавать образ диска в обратном направлении, от конца диска к началу. Этот метод полезен, если накопителю трудно читать данные после определенной точки на диске.

Инструмент myrescue разработан так, чтобы сначала обходить нечитаемые области (без повторных попыток) и сосредоточиться на восстановлении как можно большей части читаемых областей. После копирования читаемых секторов он обрабатывает неудачные диапазоны. Документация инструмента рекомендует давать проблемным накопителям отдохнуть пару часов между повторными попытками.

Другой инструмент, recoverdm, также выполняет восстановление данных. Его особенность состоит в возможности восстанавливать данные с поврежденного диска на уровне секторов или отдельных файлов. Инструмент имеет дополнительные функции для дискет и оптических носителей.

Ошибки SMART и ядра

Сведения SMART на диске могут предоставить дополнительные показатели состояния накопителя и вероятности успешного восстановления. Например:

```
# smartctl -x /dev/sda
smartctl 6.4 2014-10-07 r4002 [x86_64-linux-3.19.0-18-generic] (local build)
Copyright (C) 2002-14, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Model Family: Maxtor DiamondMax D540X-4K
Device Model: MAXTOR 4K040H2
Serial Number: 672136472275
Firmware Version: A08.1500
User Capacity: 40,971,571,200 bytes [40.9 GB]
Sector Size: 512 bytes logical/physical
...
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME FLAGS VALUE WORST THRESH FAIL RAW_VALUE
1 Raw_Read_Error_Rate P--R-K 100 253 020 - 0
3 Spin_Up_Time POS--K 087 086 020 - 1678
4 Start_Stop_Count -O--CK 078 078 008 - 14628
5 Reallocated_Sector_Ct PO--CK 003 001 020 NOW 486

7 Seek_Error_Rate PO-R-- 100 100 023 - 0
9 Power_On_Hours -O--C- 073 073 001 - 17814
10 Spin_Retry_Count -OS--K 100 100 000 - 0
11 Calibration_Retry_Count PO--C- 100 080 020 - 0
12 Power_Cycle_Count -O--CK 100 100 008 - 294
13 Read_Soft_Error_Rate PO-R-- 100 100 023 - 0
194 Temperature_Celsius -O---K 094 083 042 - 17
195 Hardware_ECC_Recovered -O-RC- 100 031 000 - 7137262
196 Reallocated_Event_Count ----C- 100 253 020 - 0
197 Current_Pending_Sector -O--CK 003 001 020 NOW 486
198 Offline_Uncorrectable ----C- 100 253 000 - 0
199 UDMA_CRC_Error_Count -O-RC- 199 199 000 - 1
||||||_ K auto-keep
|||||__ C event count
||||___ R error rate
|||____ S speed/performance
||_____ O updated online
|______ P prefailure warning

Read SMART Log Directory failed: scsi error badly formed scsi parameters
ATA_READ_LOG_EXT (addr=0x00:0x00, page=0, n=1) failed: scsi error aborted command
Read GP Log Directory failed
...
ATA Error Count: 9883 (device log contains only the most recent five errors)
...
Error 9883 occurred at disk power-on lifetime: 17810 hours (742 days + 2 hours)
...
Error 9882 occurred at disk power-on lifetime: 17810 hours (742 days + 2 hours)
...
Error 9881 occurred at disk power-on lifetime: 17810 hours (742 days + 2 hours)
...
```

При криминалистическом получении необходимо отмечать все сообщения об ошибках и отказах, появляющиеся в `dmesg` или выводе инструментов. Если секторы не удалось прочитать и было добавлено нулевое заполнение, это необходимо зафиксировать (в зависимости от используемого инструмента криминалистического получения оно будет записано в журнал).

Другие варианты для отказавших накопителей

В этом разделе приведено еще несколько советов и замечаний, помогающих получать данные с проблемных дисков.

В некоторых случаях диск может правильно работать лишь несколько минут в холодном состоянии, после чего становится недоступным или нестабильным. Если диск работает правильно несколько минут до отказа, возможно, со временем удастся создать образ, многократно возобновляя восстановление. Некоторые из инструментов, упомянутых в разделе "Инструменты восстановления данных" на странице 162, ведут файл, содержащий состояние восстановления после последней попытки. Операцию восстановления можно прервать и позднее возобновить с места остановки.

После некоторого времени попыток создать образ накопителя дайте ему остыть и повторите попытку. Иногда по мере перегрева накопителя проблемы доступа усугубляются. В этой ситуации опять же полезны функции возобновления работы инструментов восстановления дисков.

Если вы подозреваете неисправность электроники накопителя и доступен второй исправный идентичный накопитель (то есть той же марки, модели и редакции микропрограммы),[5] возможно, удастся временно переставить электронику накопителя для восстановления данных. Для этого не требуется вскрывать диск, поэтому риск повреждения (из-за пыли и так далее) минимален.

Профессиональные компании по восстановлению данных располагают чистыми помещениями, где обученный персонал может вскрывать накопители, освобождать застрявшие головки, заменять приводы и выполнять другие тонкие операции с накопителем. Не пытайтесь выполнять эти процедуры без надлежащей среды, оборудования и подготовки. Одно лишь вскрытие накопителя вне чистого помещения подвергнет его воздействию частиц пыли и повредит диск.

Поврежденные оптические диски

Большинство упомянутых ранее инструментов должно работать и с оптическими носителями. Некоторые инструменты имеют дополнительные функции или особое поведение для оптических носителей.

Для оптических носителей `ddrescue` рекомендует указывать размер сектора 2048 байт. Вот пример работы `ddrescue` по восстановлению поврежденного диска CD-ROM:

```
# ddrescue -b 2048 /dev/cdrom cdrom.raw
GNU ddrescue 1.19
Press Ctrl-C to interrupt
rescued: 15671 kB, errsize: 3878 kB, current rate: 0 B/s
ipos: 408485 kB, errors: 126, average rate: 12557 B/s
opos: 408485 kB, run time: 20.80 m, successful read: 31 s ago
Copying non-tried blocks... Pass 2 (backwards)
```

Обратите внимание, что `ddrescue` читает CD-ROM в обратном направлении, пытаясь восстановить блоки.

Для частично восстанавливаемых оптических дисков с поврежденной файловой системой можно использовать инструменты карвинга для извлечения файлов. Карвер данных, разработанный для оптических дисков, - `dares` (ftp://ftp.heise.de/pub/ct/ctsi/dares.tgz); он поддерживает различные форматы файловых систем оптических дисков.

В этом разделе рассматривалась работа с отказами и ошибками накопителей. Отказы и ошибки накопителей действительно случаются и могут привести к частичной или полной потере данных. При возникновении проблем с накопителем обязательно документируйте характер ошибки и, где возможно, затронутый сектор.

Получение образа по сети

Создание образа диска по сети может быть полезно по нескольким причинам:

• Диск может находиться удаленно, и физически изъять и отправить его в центральную криминалистическую лабораторию может быть неосуществимо (возможно, из-за нарушения работы предприятия, отсутствия ресурсов или других логистических проблем).
• Критичный по времени инцидент может потребовать как можно скорее получить образ удаленного накопителя без задержек на доставку (в зависимости от пропускной способности сети, размера диска и сроков доставки отправка диска все же может оказаться быстрее).[6]
• На машине в локальной криминалистической лаборатории может находиться диск в ПК, который невозможно физически извлечь. Причиной могут быть конструкция ПК, отсутствие необходимых инструментов или риск причинить повреждение либо уничтожить доказательства.

В целом изъятие дисков плохо масштабируется в крупных организациях, поэтому широко развернутое корпоративное решение для удаленной сортировки и получения дисков является обычным явлением. Классический пример - EnCase Enterprise; многие новые компании выводят на рынок сходные продукты.

Как и при создании образов дисков, для криминалистического получения по сети существует множество возможностей. Большинство решений предполагает загрузку удаленной машины с криминалистического компакт-диска, установление сетевого соединения и передачу вывода `dd` по сети в локальный файл. Это можно сделать просто сочетанием `dd` и `netcat`. Защищенные соединения также можно устанавливать с помощью `ssh` или защищенных альтернатив `netcat`, например `socat` и `cryptcat`.

В этом разделе приведено несколько примеров с использованием `ssh` для защищенного сетевого соединения. Но сначала рассмотрим `rdd`, специально разработанный с учетом криминалистического получения данных.

Удаленное создание криминалистического образа с помощью rdd

Инструмент `rdd`, предназначенный для получения образов дисков по сети, был разработан Netherlands Forensic Institute (NFI). `rdd` имеет ряд полезных функций, включая хеширование, журналирование, сжатие, обработку ошибок, разделение файлов, индикаторы хода работы и статистику. Поддержку вывода EWF можно включить во время компиляции. `rdd` использует модель клиент-сервер, где исследуемый ПК (загруженный с криминалистического загрузочного компакт-диска) является клиентом, а ПК эксперта - сервером. Получение выполняется запуском процесса прослушивания на сервере (ПК эксперта) и команды получения на клиенте.

Встроенной защиты у `rdd` нет; ее необходимо добавить с помощью VPN, защищенной оболочки или эквивалентного средства. При использовании `rdd` в недоверенных или враждебных сетях сетевой трафик необходимо шифровать, а прослушивающие порты TCP не должны быть открыты. Это можно сделать в два этапа: установить защищенный сетевой канал и использовать его для получения данных.

Без защиты инструмент `rdd` все же полезен в доверенном сегменте сети в защищенной лабораторной среде, при использовании перекрестных кабелей Ethernet или при соединении двух ПК кабелем FireWire. (Интерфейсы FireWire можно использовать как сетевые интерфейсы.)

На рабочей станции эксперта запустите серверный режим `rdd-copy`, указав `-S`, как показано в следующем примере. Его необходимо запустить до запуска клиента. Убедитесь, что межсетевые экраны или фильтрация пакетов `iptables` не блокируют порт TCP 4832 (порт по умолчанию).

```
# rdd-copy -S --md5 -l server.log

# cat server.log
2016-01-13 01:34:21 +0100:
2016-01-13 01:34:21 +0100: 2016-01-13 01:34:21 CET
2016-01-13 01:34:21 +0100: rdd version 3.0.4
...
2016-01-13 01:34:21 +0100: rdd-copy -S --md5 -l server.log
2016-01-13 01:34:21 +0100: ========== Parameter settings ==========
2016-01-13 01:34:21 +0100: mode: server
2016-01-13 01:34:21 +0100: verbose: no
2016-01-13 01:34:21 +0100: quiet: no
2016-01-13 01:34:21 +0100: server port: 4832
2016-01-13 01:34:21 +0100: input file: <none>
2016-01-13 01:34:21 +0100: log file: server.log
...
2016-01-13 01:37:05 +0100: === done***
2016-01-13 01:37:05 +0100: seconds: 147.787
2016-01-13 01:37:05 +0100: bytes written: 7918845952
2016-01-13 01:37:05 +0100: bytes lost: 0
2016-01-13 01:37:05 +0100: read errors: 0
2016-01-13 01:37:05 +0100: zero-block substitutions: 0
2016-01-13 01:37:05 +0100: MD5: a3fa962816227e35f954bb0b5be893ea
...
```

На удаленном исследуемом ПК запустите клиентский режим `rdd-copy` с помощью `-C`. Укажите устройство ввода с помощью `-I`. Устройством ввода может быть любое локально подключенное устройство хранения (в этом примере это был удаленный USB-накопитель). Для выходного файла `-O` предусмотрен дополнительный параметр, указывающий сетевое назначение. Клиент сообщает серверу, какой файл использовать для полученного образа, применяя традиционное соглашение Unix `имя_узла:/путь/к/имени_файла`:

```
# rdd-copy -C --md5 -l client.log -I /dev/sde -O -N lab-pc:/evi/image.raw

# cat client.log
2016-01-13 01:34:37 +0100:
2016-01-13 01:34:37 +0100: 2016-01-13 01:34:37 CET
2016-01-13 01:34:37 +0100: rdd version 3.0.4

...
2016-01-13 01:34:37 +0100: rdd-copy -C --md5 -l client.log -I /dev/sde -O -N
lab-pc:/evi/image.raw
2016-01-13 01:34:37 +0100: ========== Parameter settings ==========
2016-01-13 01:34:37 +0100: mode: client
2016-01-13 01:34:37 +0100: verbose: no
2016-01-13 01:34:37 +0100: quiet: no
2016-01-13 01:34:37 +0100: server port: 4832
2016-01-13 01:34:37 +0100: input file: /dev/sde
2016-01-13 01:34:37 +0100: log file: client.log
2016-01-13 01:34:37 +0100: output #0
2016-01-13 01:34:37 +0100: output file: /evi/image.raw
2016-01-13 01:34:37 +0100: segment size: 0
2016-01-13 01:34:37 +0100: output as ewf compression: no ewf
2016-01-13 01:34:37 +0100: output host: lab-pc
2016-01-13 01:34:37 +0100: output port: 4832
...
2016-01-13 01:37:05 +0100: === done***
2016-01-13 01:37:05 +0100: seconds: 147.787
2016-01-13 01:37:05 +0100: bytes written: 7918845952
2016-01-13 01:37:05 +0100: bytes lost: 0
2016-01-13 01:37:05 +0100: read errors: 0
2016-01-13 01:37:05 +0100: zero-block substitutions: 0
2016-01-13 01:37:05 +0100: MD5: a3fa962816227e35f954bb0b5be893ea
...
```

И клиент, и сервер задают файлы журналов с помощью `-l` и алгоритм хеширования, который можно проверить в конце передачи. Ход работы клиента и сервера можно наблюдать, добавив `-P 1` на одной или обеих сторонах.

Защищенное удаленное создание образа с помощью ssh

Когда `rdd` недоступен, базовое получение можно выполнить одной командой `ssh` либо на удаленном ПК с исследуемым диском, либо на ПК эксперта. В следующем примере показано создание образа диска (в данном случае USB-накопителя, подключенного к удаленному ПК) по сети через сеанс защищенной оболочки, инициированный с удаленного ПК:

```
# dd if=/dev/sdb | ssh lab-pc "cat > sandisk-02028302BCA1D848.raw"
7856127+0 records in
7856127+0 records out
4022337024 bytes (4.0 GB) copied, 347.411 s, 11.6 MB/s
```

Команда `dd` выполняется локально, а ее вывод передается по конвейеру команде `ssh`. Защищенная оболочка передает этот поток данных программе `cat` на ПК эксперта. Вывод программы `cat` перенаправляется в файл на ПК эксперта. После завершения будет доступен необработанный образ, который можно исследовать другими криминалистическими инструментами.

Получить образ можно и через защищенную оболочку, инициированную с рабочей станции эксперта и подключающуюся к удаленному ПК с исследуемым диском. Следующий пример демонстрирует это на ПК эксперта, где снова создается образ того же USB-накопителя:

```
# ssh remote-pc "dd if=/dev/sdb" > sandisk-02028302BCA1D848.raw
7856127+0 records in
7856127+0 records out
4022337024 bytes (4.0 GB) copied, 343.991 s, 11.7 MB/s
```

Здесь защищенной оболочке предписывается выполнить команду `dd` на удаленной (исследуемой) машине. Вывод удаленной команды `dd` становится выводом локальной команды `ssh` и перенаправляется в локальный файл. После завершения на ПК эксперта доступен файл необработанного образа для анализа.

Базовые команды `dd`, показанные в этом разделе, можно заменить на `dcfldd`, `dc3dd` или любой другой инструмент получения данных, создающий образ в `stdout`.

Этим способом можно собирать и другие сведения об удаленной (исследуемой) машине. Для иллюстрации приведены примеры сбора данных об удаленном ПК, запущенном с загрузочного компакт-диска DEFT Linux. В этом примере данные `hdparm`, `smartctl` и `lshw` собираются и сохраняются на рабочей станции эксперта:

```
# ssh remote-pc "hdparm --dco-identify /dev/sda" > dco.lenovo-W38237SJ.txt
# ssh remote-pc "hdparm -I /dev/sda" > hdparm.lenovo-W38237SJ.txt
# ssh remote-pc "smartctl -x /dev/sda" > smartctl.lenovo-W38237SJ.txt
# ssh remote-pc "lshw" > lshw.lenovo-W38237SJ.txt
```

Как и в предыдущем примере, `ssh` выполняет различные команды на удаленной машине, а вывод перенаправляется в файлы на локальной рабочей станции эксперта. Серийный номер диска включается в имя файла, чтобы обеспечить очевидную связь между физическим диском и собранными файлами данных.

Удаленное получение в контейнер доказательств SquashFS

Как было показано ранее, SquashFS можно использовать как контейнер криминалистических доказательств, а `sfsimage` - для создания образов локальных дисков. Сценарий `sfsimage` также может создавать образ диска на удаленной машине непосредственно в контейнер криминалистических доказательств SquashFS. Здесь показаны два примера.

Удаленный вывод `dd` можно передать через `ssh` локальной команде `sfsimage`, создав контейнер криминалистических доказательств SquashFS с необработанным образом:

```
$ ssh root@remote-pc "dd if=/dev/mmcblk0" | sfsimage -i - remote-pc.sfs
Started: 2016-05-08T10:30:34
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i -
Current working directory: /home/holmes
Forensic evidence source:

Destination squashfs container: remote-pc.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd log=errorlog.txt hlog=hashlog.txt hash=md5
2>/dev/null | pv -s 0
31116288+0 records inMiB/s] [ <=> ]
31116288+0 records out
15931539456 bytes (16 GB, 15 GiB) copied, 597.913 s, 26.6 MB/s
14.8GiB 0:09:58 [25.4MiB/s] [ <=> ]
Completed: 2016-05-08T10:40:32
```

В этом примере доступ к удаленному ПК выполняется пользователем `root` (`root@remote-pc`), а образ удаленной карты памяти (`/dev/mmcblk0`) создается командой `dd` в `stdout`. Поток `stdout` передается по соединению `ssh` локальной команде `sfsimage`, где `-` (`stdin`) является входным файлом.

Второй способ использует тот же принцип, но с переменными сценария оболочки `sfsimage`. В блоке `config()` сценария `sfsimage` или в отдельном файле `sfsimage.conf` можно задать переменные и параметры конфигурации, управляющие поведением `sfsimage`. Если присвоить переменной `DD` команду `ssh`, `mksquashfs` будет получать входные данные с удаленной машины через `ssh`. Здесь показан файл конфигурации в текущем рабочем каталоге:

```
$ cat sfsimage.conf
DD="ssh root@remote-pc \"dd if=/dev/mmcblk0\""
SQSUDO=""
```

Двойные кавычки в переменной `DD` необходимо экранировать. Переменной `SQSUDO` присваивается пустая строка, поскольку локальные привилегии `root` не нужны. При запуске `sfsimage` с этим файлом конфигурации в локальном рабочем каталоге ваши параметры конфигурации переопределят стандартные параметры `sfsimage`.

Важно отметить, что входной файл по-прежнему следует указывать дефисом (`-`), поскольку команда `ssh` в переменной `DD` внутренне передает входные данные в `stdin`. Удаленное получение с помощью `sfsimage` в таком виде выглядит следующим образом:

```
$ sfsimage -i - remote-pc.sfs
Started: 2016-05-08T10:56:30
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i -
Current working directory: /home/holmes
Forensic evidence source:
Destination squashfs container: remote-pc.sfs
Image filename inside container: image.raw
Aquisition command: ssh root@remote-pc "dd if=/dev/mmcblk0" 2>/dev/null | pv -s 0
14.8GiB 0:09:03 [ 28MiB/s] [ <=> ]
Completed: 2016-05-08T11:05:33
```

Этот пример конфигурации DD был приведен прежде всего для иллюстрации возможности встраивать команды удаленного создания образов по сети в `sfsimage`. Встраивание сложных команд получения данных в файлы конфигурации в целом можно использовать для изменения работы сценария `sfsimage`.

Получение удаленного диска в формате EnCase или FTK

Удаленные команды `ssh` можно также передавать по конвейеру другим программам для выполнения задач или преобразования в другие форматы. Полезный пример - удаленное получение необработанного образа и его преобразование в EnCase/EWF в процессе записи на диск. В этом примере образ удаленного ПК дистанционно создается на рабочей станции эксперта и сохраняется в файлы `*.ewf`:

```
# ssh remote-pc "dd if=/dev/sda" | ewfacquirestream -D 16048539022588504422 -t
eepc-16048539022588504422
ewfacquirestream 20140608

Using the following acquiry parameters:
Image path and filename: eepc-16048539022588504422.E01
Case number: case_number
Description: 16048539022588504422
Evidence number: evidence_number
Examiner name: examiner_name
Notes: notes
Media type: fixed disk
Is physical: yes
EWF file format: EnCase 6 (.E01)
Compression method: deflate
Compression level: none
Acquiry start offset: 0
Number of bytes to acquire: 0 (until end of input)
Evidence segment file size: 1.4 GiB (1572864000 bytes)
Bytes per sector: 512
Block size: 64 sectors
Error granularity: 64 sectors
Retries on read error: 2
Zero sectors on read error: no

Acquiry started at: Jun 22, 2015 21:22:47
This could take a while.
...
Status: acquired 3.7 GiB (3999301632 bytes)
in 7 minute(s) and 38 second(s) with 8.3 MiB/s (8732099 bytes/second).
7815024+0 records in
7815024+0 records out
4001292288 bytes (4.0 GB) copied, 451.948 s, 8.9 MB/s

Acquiry completed at: Jun 22, 2015 21:30:25

Written: 3.7 GiB (4001526432 bytes) in 7 minute(s) and 38 second(s) with 8.3 MiB/s
(8736957 bytes/second).
MD5 hash calculated over data: e86d952a68546fbdab55d0b205cd1c6e
ewfacquirestream: SUCCESS
```

В этом примере описание ПК (`eepc`) и серийный номер (`16048539022588504422`) встроены в имя файла образа. Заключительный вывод команды `dd` показывается после завершения, а сразу за ним следует сообщение о завершении `ewfacquirestream`.

Полученный образ можно криминалистически анализировать с помощью EnCase, Sleuth Kit или любого другого инструмента с поддержкой EWF.

```
# ls -l eepc-16048539022588504422.*
-rw-r----- 1 root root 1572852270 Jun 22 21:30 eepc-16048539022588504422.E01
-rw-r----- 1 root root 1572851461 Jun 22 21:30 eepc-16048539022588504422.E02
-rw-r----- 1 root root 857059301 Jun 22 21:30 eepc-16048539022588504422.E03
```

Дополнительные флаги `ewfacquirestream` позволяют предоставить больше подробностей метаданных дела, увеличить сжатие и задействовать другие функции. Дополнительные сведения см. на странице руководства `ewfacquirestream(1)`.

Создание образа работающей системы со снимками Copy-On-Write

В общем случае создавать криминалистический образ работающей системы, когда получаемые диски содержат запущенную операционную систему, бессмысленно. Блоки работающей системы непрерывно изменяются. За время, необходимое для получения посекторного образа, файловая система существенно изменится, из-за чего ее копия в образе окажется поврежденной и несогласованной.

Иногда загрузить систему с криминалистического загрузочного компакт-диска для удаленного получения образа может быть невозможно. На работающих серверах, которые нельзя выключить, в некоторых ситуациях можно воспользоваться тем же способом замораживания файловой системы, который применяется для резервного копирования. В системах с файловыми системами Copy-on-Write (CoW) определенный объем криминалистического создания образов может быть возможен, если со снимками файловой системы связаны блочные устройства (например, Logical Volume Manager [LVM]). Это предоставит согласованный снимок блоков файловой системы в определенный момент времени. Если у снимка файловой системы CoW нет связанного блочного устройства, по крайней мере файлы будут заморожены для получения на уровне файлов.

Если исследуемая система является облачной виртуальной машиной, создание образа работающей системы по сети может быть единственным вариантом, если поставщик облака не может предоставить образы снимков.

Получение съемных носителей

Съемные носители уникальны тем, что устройство накопителя может быть подключено к системе и работать без какого-либо носителя. Блочные устройства, которые можно получить криминалистическим способом, становятся доступны только после установки носителя. USB-флеш-накопители можно описывать как съемные устройства, но не как съемные носители. Носитель не извлекается из USB-флеш-накопителя, если только это не адаптер карты памяти или устройство чтения карт.

В этом разделе рассматриваются основные виды съемных носителей: карты памяти, оптические диски и магнитные ленты.

Карты памяти

Большинство карт памяти ведет себя сходно с обычными накопителями. Их хранилище представляется как линейная последовательность блоков, создавая вид обычного накопителя с секторами, к которым можно обращаться любым инструментом, работающим с блочными устройствами.

На рис. 6-1 карта Micro SD установлена в адаптер SD, адаптер SD - в устройство чтения карт SD, а устройство чтения - в ПК. Здесь несколько съемных элементов наложены друг на друга и все равно выглядят как блочное устройство, образ которого можно создать обычным способом.

Рис. 6-1. Адаптеры съемной карты памяти

В этом примере все три элемента были вставлены и подключены к основной системе получения данных. Ядро обнаружило их и создало блочное устройство `/dev/sdg`:

```
# dmesg
...
[65396.394080] usb-storage 3-2:1.0: USB Mass Storage device detected
[65396.394211] scsi host21: usb-storage 3-2:1.0
[65397.392652] scsi 21:0:0:0: Direct-Access SanDisk SDDR-113 1.00 PQ:
0 ANSI: 0

[65397.393098] sd 21:0:0:0: Attached scsi generic sg5 type 0
[65398.073649] sd 21:0:0:0: [sdf] 3911680 512-byte logical blocks: (2.00 GB/1.87
GiB)
[65398.074060] sd 21:0:0:0: [sdf] Write Protect is on
...
```

На адаптере SD включен язычок защиты от записи, что видно в выводе `dmesg`.

В этом примере образ карты Micro SD создается в контейнере доказательств SquashFS с помощью сценария `sfsimage`:

```
$ sfsimage -i /dev/sdf MicroSD.sfs
Started: 2016-05-08T11:19:35
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i /dev/sdf
Current working directory: /home/holmes
Forensic evidence source: if=/dev/sdf
Destination squashfs container: MicroSD.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd if=/dev/sdf log=errorlog.txt hlog=hashlog.txt hash=md5
2>/dev/null | pv -s 2002780160
1.87GiB 0:02:34 [12.3MiB/s] [================================================>] 100%
Completed: 2016-05-08T11:22:10
```

После создания образа карту памяти можно безопасно извлечь из устройства чтения карт (при условии, что она не была смонтирована).

Оптические диски

Разные виды оптических носителей различаются физическими и химическими свойствами; однако после установки в подключенный оптический накопитель у них больше сходств, чем различий. Три наиболее распространенных диска (DVD, CD-ROM и Blu-ray) имеют размер сектора 2048 байт и выглядят как линейная последовательность секторов (подобно ленте, но расположенной по спирали). Основные различия состоят в плотности битов данных (абстрагированной аппаратной частью устройства) и емкости диска.

Создавать образы дисков данных несложно; процесс сходен с созданием образов жестких дисков или флеш-носителей. Здесь показан пример создания образа оптического диска с помощью `dc3dd`:

```
# dc3dd if=/dev/cdrom of=datacd.raw
dc3dd 7.2.641 started at 2016-01-13 23:04:31 +0100
compiled options:
command line: dc3dd if=/dev/cdrom of=datacd.raw
device size: 331414 sectors (probed), 678,735,872 bytes
sector size: 2048 bytes (probed)
678735872 bytes ( 647 M ) copied ( 100% ), 142 s, 4.5 M/s

input results for device `/dev/cdrom':
331414 sectors in
0 bad sectors replaced by zeros
output results for file `datacd.raw':
331414 sectors out
dc3dd completed at 2016-01-13 23:06:53 +0100
```

После этого файл образа `datacd.raw` можно анализировать обычными криминалистическими инструментами.

Восстановление Compact Disc Digital Audio (CDDA), или музыкальных компакт-дисков, отличается от восстановления дисков данных. Они содержат набор музыкальных дорожек, представляющих собой линейные потоки битов, закодированных импульсно-кодовой модуляцией (PCM). В отличие от компакт-дисков с данными, здесь допускаются некоторые ошибки. Поэтому были созданы инструменты, пытающиеся восстанавливать CDDA и устранять проблемы накопителей, такие как смещение и дрожание кадров.[7] Большинство инструментов CDDA - простые программы копирования музыкальных компакт-дисков, преобразующие дорожки CD в аудиофайлы (с повторным кодированием в другой аудиоформат). В этом примере `cdparanoia` выполняет необработанное извлечение данных PCM:

```
# cdparanoia --output-raw --log-summary 1- cdda.raw
cdparanoia III release 10.2 (September 11, 2008)

Ripping from sector 0 (track 1 [0:00.00])
to sector 251487 (track 15 [4:58.72])
outputting to cdda.raw

(== PROGRESS == [ | 251487 00 ] == :^D* ==)

Done.
```

Эта команда копирует весь музыкальный компакт-диск в единый файл необработанного звукового образа PCM, содержащий все звуковые дорожки. Затем этот файл можно импортировать в программу анализа звука. Поскольку звуковые данные не изменялись и не перекодировались, потери или ухудшения качества звука нет.

Диски DVD и Blu-ray с управлением цифровыми правами (DRM) и региональной защитой трудно восстанавливать. Инструменты Linux и инструкции по восстановлению зашифрованного содержимого существуют, но намеренно оставлены за рамками книги.

Магнитные ленты

В домашних условиях ленты практически исчезли. Но они по-прежнему используются в малых, средних и корпоративных средах для резервного копирования и архивирования. В редких случаях может поступить запрос на восстановление данных с лент. Например, в корпоративных криминалистических лабораториях старые ленты иногда находят при реорганизации отделов компании или переезде.

Исторически популярными были ленты 4 мм DAT, 8 мм Exabyte и DLT. Сегодня наиболее распространены LTO и 8 мм DAT. Максимальная собственная и сжатая емкость этих лент составляет 160 Гбайт/320 Гбайт для DAT-320 и 6 Тбайт/15 Тбайт для LTO-7. Современные накопители LTO также поддерживают зашифрованные ленты.

Современные ленточные накопители подключаются к основным системам через интерфейс SAS или Fibre Channel. Исторически почти все ленточные накопители следовали стандартам SCSI Stream Command (SSC; последним является SSC-5).

Ленточные технологии используют собственное понятие "файлов", размещаемых на ленте последовательно. Как правило, ленточный файл состоит из архива резервной копии, созданного программой резервного копирования или архивирования. К ленточным файлам нельзя обращаться произвольно, как к дисковым накопителям и оптическим дискам. Вместо этого к ним обращаются, перемещаясь вперед или назад к началу файла с определенным номером, а затем читая логические блоки до конца файла.

На лентах имеются различные маркеры, сообщающие ленточному накопителю положение головки на ленте (см. рис. 6-2). Здесь важно понимать следующие маркеры:

BOT или BOM (Beginning of Tape или Media). Сообщает накопителю, где можно начинать чтение или запись данных.

EOF (End of File). Сообщает накопителю, что достигнут конец ленточного файла.

EOD (End of Data). Сообщает накопителю, что достигнут конец записанных данных (находится сразу после последнего ленточного файла). Это логический конец ленты.

PEOT, EOT или EOM ([Physical] End of Tape или Media). Сообщает накопителю, что достигнут конец физической длины ленты.

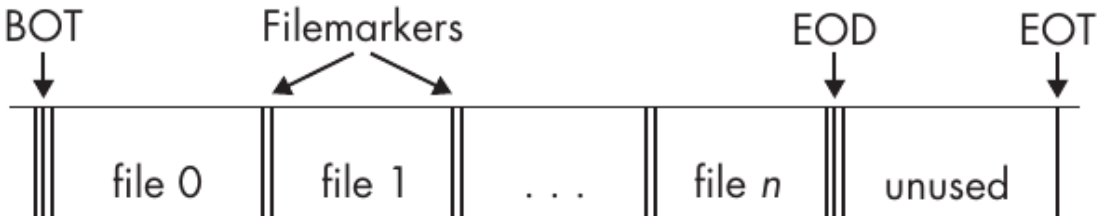

Рис. 6-2. Файлы и маркеры на ленте

При криминалистическом получении лент необходимо скопировать каждый файл на ленте вплоть до маркера EOD (последнего читаемого файла на ленте). Прочитать данные после EOD на ленте стандартными командами SCSI невозможно. Некоторые криминалистические компании предлагают специализированное оборудование и услуги, позволяющие восстанавливать данные после EOD.

Файлы с ленты можно извлекать с помощью вариантов `dd`. В следующем примере с ленты восстановлены три ленточных файла. Для доступа к ленте выбрано устройство без перемотки, обычно `/dev/nst0` в Linux, чтобы накопитель не перемотал ленту до копирования всех файлов. Команда выполняется многократно, всегда с одним и тем же устройством ввода (она берет следующий файл на ленте), пока не появится `0+0 records in`, указывающее, что все файлы извлечены:

```
# dcfldd if=/dev/nst0 of=file0.tape hashlog=hash0.log
0+46 records in
14+1 records out

# dcfldd if=/dev/nst0 of=file1.tape hashlog=hash1.log
22016 blocks (688Mb) written.
0+70736 records in
22105+0 records out

# dcfldd if=/dev/nst0 of=file2.tape hashlog=hash2.log
256 blocks (8Mb) written.
0+1442 records in
450+1 records out

# dcfldd if=/dev/nst0 of=file3.tape hashlog=hash3.log
0+0 records in
0+0 records out
```

После восстановления ленточных файлов можно проанализировать их тип. Часто достаточно обычной программы определения типа файла, чтобы установить использованный формат архива или резервной копии. В этом примере извлечены два файла `.tar` и один файл `.dump`:

```
# ls -l
total 722260
-rw-r----- 1 root root 471040 Jan 14 01:46 file0.tape
-rw-r----- 1 root root 724336640 Jan 14 01:46 file1.tape
-rw-r----- 1 root root 14766080 Jan 14 01:47 file2.tape
-rw-r----- 1 root root 0 Jan 14 01:47 file3.tape
-rw-r----- 1 root root 46 Jan 14 01:46 hash0.log
-rw-r----- 1 root root 46 Jan 14 01:46 hash1.log
-rw-r----- 1 root root 46 Jan 14 01:47 hash2.log
-rw-r----- 1 root root 46 Jan 14 01:47 hash3.log

# file *.tape
file0.tape: POSIX tar archive (GNU)
file1.tape: POSIX tar archive (GNU)
file2.tape: new-fs dump file (little endian), Previous dump Thu Jan 14 01:39:29
2016, This dump Thu Jan 1 01:00:00 1970, Volume 1, Level zero, type: tape
header, Label none, Filesystem / (dir etc), Device /dev/sdf1, Host lab-pc,
Flags 3
file3.tape: empty
```

Каждый из файлов `hash*.log` содержит отдельный хеш MD5 для каждого извлеченного ленточного файла. В этом примере `file3.tape` пуст и его можно игнорировать.

RAID и многодисковые системы

Криминалистическое получение систем Redundant Array of Independent Disks (RAID) связано с рядом трудностей и может требовать дополнительных действий. Важно планировать емкость, поскольку может потребоваться создать образы большого числа дисков.

В этом разделе предполагается, что образы отдельных дисков RAID созданы раздельно и находятся на рабочей станции получения данных. Цель состоит в сборке дисков из образов и предоставлении доступа к уровню метаустройства в виде файла или блочного устройства, что позволяет использовать инструменты криминалистического анализа.

Как правило, системы RAID создают собственные сведения заголовка в начале диска (а иногда в конце диска). Заголовок содержит уникальные идентификаторы (UUID), имена массивов, временные метки, сведения о конфигурации RAID и другую служебную информацию.

Получение закрытых RAID

Если использовался аппаратный контроллер RAID и программ для автономной сборки RAID не существует, может потребоваться клонировать диски RAID и загрузить систему исследования с физически установленным контроллером.

Примеры этого раздела посвящены программному RAID Linux, но доступен ряд инструментов с открытым исходным кодом, способных поддерживать получение и анализ закрытых систем RAID.

Например, следующие пакеты содержат такие инструменты и доступны из репозитория программ Debian:

• `dpt-i2o-raidutils` - утилиты управления аппаратным RAID Adaptec I2O
• `array-info` - инструмент командной строки для сообщения состояния RAID нескольких типов
• `cciss-vol-status` - средство проверки состояния томов HP SmartArray RAID
• `cpqarrayd` - инструмент наблюдения за контроллерами HP (Compaq) SmartArray
• `dpt-i2o-raidutils` - утилиты управления аппаратным RAID Adaptec I2O
• `mpt-status` - инструмент получения состояния RAID из контроллеров `mpt` (и других аппаратных контроллеров RAID)
• `varmon` - монитор VA RAID

Помимо этих программных пакетов, инструмент `dmraid` способен идентифицировать метаданные RAID ряда закрытых форматов. Список поддерживаемых форматов можно получить с помощью флага `-l` следующим образом:

```
# dmraid -l
asr : Adaptec HostRAID ASR (0,1,10)
ddf1 : SNIA DDF1 (0,1,4,5,linear)

hpt37x : Highpoint HPT37X (S,0,1,10,01)
hpt45x : Highpoint HPT45X (S,0,1,10)
isw : Intel Software RAID (0,1,5,01)
jmicron : JMicron ATARAID (S,0,1)
lsi : LSI Logic MegaRAID (0,1,10)
nvidia : NVidia RAID (S,0,1,10,5)
pdc : Promise FastTrack (S,0,1,10)
sil : Silicon Image(tm) Medley(tm) (0,1,10)
via : VIA Software RAID (S,0,1,10)
dos : DOS partitions on SW RAIDs
```

Инструмент `dmraid` использует тот же механизм отображения устройств, который показан в разделе "Работа с отказами и ошибками накопителей" на странице 159 (где инструмент `dmsetup` использовался для имитации ошибок). На странице руководства `dmraid(8)` приведен ряд примеров повторной сборки различных закрытых конфигураций RAID.[8]

JBOD и чередующиеся диски RAID-0

Just a Bunch Of Disks (JBOD) - термин, обозначающий объединение нескольких дисков в один логический накопитель (без конфигурации RAID для производительности или избыточности). Для сборки группы дисков в единое устройство JBOD можно использовать команду `dmsetup`.

При построении устройств из нескольких дисков полезно иметь отдельный файл таблицы, определяющий устройство, смещения и отображения. В этот простой текстовый файл можно также включить комментарии со сведениями о дисках.

В следующем примере имеется JBOD с тремя дисками разного размера (особенность систем JBOD состоит в возможности использовать любое сочетание размеров накопителей). Файл таблицы отображения устройства JBOD (`jbod-table.txt` в этом примере) определяет способ их объединения. Выполните команду `dmsetup`, передав таблицу как входные данные, чтобы создать устройство в `/dev/mapper`:

```
# cat jbod-table.txt
0 15589376 linear /dev/sdm 0
15589376 15466496 linear /dev/sdn 0
31055872 15728640 linear /dev/sdo 0

# dmsetup create jbod < jbod-table.txt
```

Эта таблица определяет три отображения, создающие файл устройства, который появится в `/dev/mapper`. Каждая строка задает смещение в устройстве отображения, число отображаемых секторов, тип цели (`linear`) и целевое устройство со смещением (здесь сектор ноль, поскольку требуется все устройство). Правильно вычислить смещения может быть непросто. При возникновении проблем сначала перепроверьте смещения.

Таблица передается по конвейеру команде `dmsetup create` с указанием имени устройства отображения. После создания устройства с ним можно работать обычными криминалистическими инструментами. В следующем примере команда `fsstat` из Sleuth Kit применяется к вновь созданному устройству:

```
# fsstat /dev/mapper/jbod
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: cfd74d32abd105b18043840bfd2743b3
...
```

Когда устройство отображения больше не нужно, удалите его командой `dmsetup` следующим образом:

```
# dmsetup remove /dev/mapper/jbod
```

Дополнительные сведения о различных типах отображения устройств, используемых инструментом `dmsetup`, см. на странице руководства `dmsetup(8)`. Отображения устройств можно применять для шифрования, снимков, систем RAID и даже имитации ошибок и отказывающих устройств (что полезно для испытания поведения криминалистических инструментов).

Чередующиеся диски RAID-0 создаются ради производительности, а не избыточности. Группа дисков в конфигурации RAID-0 имеет суммарную емкость всех накопителей, а доступ к дискам распределяется по массиву (производительность возрастает с добавлением дисков).

Если известны смещения и размер фрагмента чередующегося массива RAID-0, инструмент `dmsetup` может создать устройство отображения, представляющее собранный массив.

В следующем примере к основной системе получения данных подключен RAID-0, состоящий из двух чередующихся дисков. Известно, что исследуемая система RAID имеет 2048 начальных секторов с метаданными, а размер фрагмента равен 128 секторам. После этого RAID можно собрать следующим образом:

```
# dmsetup create striped --table '0 117243904 striped 2 128 /dev/sda 2048 /dev/sdb
2048'
```

Это устройство `/dev/mapper` можно анализировать обычными криминалистическими инструментами для файловых систем. Здесь показан пример применения команды `fls` из Sleuth Kit к вновь созданному устройству:

```
# fls /dev/mapper/striped
r/r 4-128-4: $AttrDef
r/r 8-128-2: $BadClus
r/r 8-128-1: $BadClus:$Bad
```

Не забудьте удалить устройство после выполнения задач.

Динамические диски Microsoft

Microsoft создала Logical Disk Manager (LDM) для управления логическими томами; для анализа динамических дисков Microsoft можно использовать инструмент Linux `ldmtool`. Цель состоит в предоставлении тома для доступа на уровне блоков криминалистическим инструментам.

В этом примере к основной системе получения данных подключены два исследуемых диска с томом, созданным Microsoft LDM. Группа дисков LDM идентифицируется глобально уникальным идентификатором (GUID). Диски можно просканировать на наличие GUID группы дисков, который позволит получить дополнительные сведения о группе дисков с помощью команды `ldmtool show`:

```
# ldmtool scan /dev/sda /dev/sdb
[
"04729fd9-bac0-11e5-ae3c-c03fd5eafb47"
]

# ldmtool show diskgroup 04729fd9-bac0-11e5-ae3c-c03fd5eafb47
{
"name" : "LENNY-Dg0",
"guid" : "04729fd9-bac0-11e5-ae3c-c03fd5eafb47",
"volumes" : [
"Volume1"
],
"disks" : [
"Disk1",
"Disk2"
]
}
```

Команда `show` предоставляет имя и GUID группы дисков, имена томов и имена дисков. Этих сведений достаточно для создания устройства отображения.

Зная GUID и имя тома, можно создать устройство тома:

```
# ldmtool create volume 04729fd9-bac0-11e5-ae3c-c03fd5eafb47 Volume1
[
"ldm_vol_LENNY-Dg0_Volume1"
]
```

Это создает в `/dev/mapper` устройство, соответствующее файловой системе динамического диска (эквивалент устройству раздела наподобие `/dev/sda1`). После этого с устройством можно работать обычными инструментами криминалистического анализа. Ниже показан пример с командой `fsstat` из Sleuth Kit:

```
# fsstat /dev/mapper/ldm_vol_LENNY-Dg0_Volume1
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS

Volume Serial Number: 0CD28FC0D28FAD10
OEM Name: NTFS
Version: Windows XP

METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 786432
First Cluster of MFT Mirror: 2
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
...
```

Когда устройство больше не нужно, удалите его командой `dmsetup`, как показано в разделе "JBOD и чередующиеся диски RAID-0" на странице 179.

Зеркальные диски RAID-1

Зеркальные диски просты и состоят из двух идентичных дисков (или должны быть идентичными, если они были синхронизированы). Создайте образ каждого диска в отдельный файл образа. В зависимости от программного или аппаратного обеспечения зеркалирования в начальных секторах диска может находиться заголовок, который необходимо пропускать при анализе.

В следующем примере показаны зеркальные диски с разделом EXT4. Программа зеркалирования (Linux Software RAID) использовала первые 32 768 секторов, и зеркальная файловая система начинается с этого смещения на физических дисках и без смещения для составного устройства[9] `md0`:

```
# fsstat /dev/md0
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: f45d47511e6a2db2db4a5e9778c60685
...

# fsstat -o 32768 /dev/sde
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: f45d47511e6a2db2db4a5e9778c60685
...

# fsstat -o 32768 /dev/sdg
FILE SYSTEM INFORMATION
--------------------------------------------
```

```
File System Type: Ext4
Volume Name:
Volume ID: f45d47511e6a2db2db4a5e9778c60685
```

В этом примере та же файловая система на `md0` также обнаруживается со смещением 32 Кбайт на двух физических устройствах (`sde` и `sdg`). Криптографические контрольные суммы зеркальных дисков, вероятно, не совпадут друг с другом, поскольку сведения заголовка RAID могут различаться (уникальные UUID дисков и так далее), а диски могут быть синхронизированы не идеально.

Linux RAID-5

Если несколько дисков входят в массив Linux RAID, их можно получить по отдельности, а затем собрать несколькими способами. Инструмент `dmsetup` предоставляет интерфейс к `mdadm` посредством таблиц. Инструмент `mdadm` может работать с отображенными или зацикленными устройствами. В следующем примере используются три полученных образа накопителей из конфигурации Linux MD RAID-5.

Анализ отдельных таблиц разделов с помощью `mmls` обнаруживает раздел Linux RAID в секторе 2048 каждого образа диска (`sda.raw`, `sdb.raw` и `sdc.raw`). Это смещение в секторах преобразуется (с помощью арифметического расширения Bash) в смещение в байтах для команды `losetup`:

```
# mmls sda.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0312580095 0312578048 Linux RAID (0xfd)
03: ----- 0312580096 0312581807 0000001712 Unallocated

# echo $((2048*512))
1048576
```

Для каждого диска массива создается устройство loop только для чтения с вычисленным смещением в байтах (2048 секторов, то есть 1048576 байт):

```
# losetup --read-only --find --show -o 1048576 sda.raw
/dev/loop0

# losetup --read-only --find --show -o 1048576 sdb.raw
/dev/loop1

# losetup --read-only --find --show -o 1048576 sdc.raw
/dev/loop2
```

Предыдущие команды создают устройства loop, соответствующие трем полученным файлам образов. Массив можно собрать с помощью `mdadm` следующим образом:

```
# mdadm -A --readonly /dev/md0 /dev/loop0 /dev/loop1 /dev/loop2
mdadm: /dev/md0 has been started with 3 drives.
```

Теперь метадисковое устройство RAID `/dev/md0` можно открывать и анализировать обычными криминалистическими инструментами. Здесь показан пример с командой `fsstat` из Sleuth Kit:

```
# fsstat /dev/md0
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: 37b9d96d8ba240b446894383764412
...
```

Вновь созданное устройство также можно смонтировать и обращаться к нему обычными файловыми инструментами. Стандартную команду Linux `mount` можно использовать следующим образом:

```
# mkdir mnt
# mount /dev/md0 mnt
mount: /dev/md0 is write-protected, mounting read-only
```

После завершения анализа выполните шаги в обратном порядке для очистки, включая применение команды `stop` системы `mdadm`:

```
# umount mnt
# rmdir mnt
# mdadm --stop /dev/md0
# losetup -d /dev/loop2
# losetup -d /dev/loop1
# losetup -d /dev/loop0
```

В зависимости от конфигурации системы ядро Linux может автоматически пытаться повторно собрать подключенные устройства RAID при их обнаружении, возможно запуская операцию восстановления, способную уничтожить доказательства. Важно применять блокирование записи с работающими устройствами и обеспечивать создание устройств loop и массивов только для чтения.

Многие методы, описанные в этой главе, применимы к устройствам loop, отображающимся на файлы образов. Дополнительные примеры создания и использования устройств loop приведены в главе 8.

Заключительные мысли

В этой главе я рассмотрел основную тему книги - криминалистическое получение данных. Вы научились использовать разные инструменты на основе `dd`, создавать образы в криминалистических форматах и применять SquashFS как контейнер криминалистических доказательств. Были показаны различные аспекты сохранения доказательств средствами криптографии, включая хеширование, окна хеширования, подписание и присвоение временных меток. Теперь вы глубже понимаете управление ошибками и восстановление при создании образов проблемных носителей. Вы умеете создавать образы по сети, а также образы съемных носителей и многодисковых систем (RAID). Это ядро процесса криминалистического получения данных.

Сноски

[1] [назад] Предполагается, что уполномоченное лицо установило GnuPG и безопасно создало пару ключей.
[2] [назад] PEM первоначально был определен в стандарте Privacy Enhanced Mail, а сегодня обычно обозначает формат файлов, используемый для хранения сертификатов X.509.
[3] [назад] Для снижения риска похищения ключей или злоумышленных изменений одним лицом можно также использовать несколько подписей разных людей.
[4] [назад] В некоторых системах это сценарий Perl, расположенный в `/usr/lib/ssl/misc`.
[5] [назад] В отрасли восстановления данных это называется донорским накопителем.
[6] [назад] Здесь уместна цитата Эндрю С. Таненбаума: "Никогда не недооценивайте пропускную способность универсала, полного лент и мчащегося по шоссе".
[7] [назад] `cdparanoia` был разработан, когда у накопителей CD было больше проблем с качеством, чем у современных накопителей.
[8] [назад] Heinz Mauelshagen, "dmraid - Device-Mapper RAID Tool: Supporting ATARAID Devices via the Generic Linux Device-Mapper". Доклад на Linux Symposium, Оттава, Онтарио, 20-23 июля 2005 года.
[9] [назад] Изначально драйвер Linux `md` означал `mirror device`, а в некоторых операционных системах такие устройства называются метаустройствами.

Глава 7. Управление криминалистическими образами

В этой главе рассматриваются различные аспекты управления файлами криминалистических образов после получения. Образы дисков огромны по сравнению с обычными файлами на диске, поэтому перемещение, копирование и преобразование больших файлов образов может быть неудобным и занимать много времени. Вы узнаете ряд методов управления большими файлами образов, помогающих преодолеть различные трудности. К ним относятся сжатие и разделение образов для упрощения работы, защита образов шифрованием и преобразование образов между форматами. Кроме того, я описываю процедуры монтирования образа только для чтения с целью безопасного локального просмотра и демонстрирую криминалистическое клонирование (или дублирование диска). Я также рассматриваю безопасное и надежное хранение и передачу больших файлов образов по сети. Глава завершается описанием безопасного стирания и удаления образов и файлов. Начнем с управления сжатием образов.

Управление сжатием образов

Необработанные образы дисков всегда имеют размер, равный общему объему содержащихся в них секторов. Число файлов или объем данных на накопителе не имеют значения и не влияют на размер несжатого необработанного образа. При нынешнем широком распространении многотерабайтных дисков работа с образами в условиях ограничений времени и емкости дисков может быть затруднительна. Даже простое копирование образа может занять много часов. Отчасти уменьшить эту проблему можно, храня образы в сжатом виде.

Сжатие образов в криминалистическом контексте означает посекторное сжатие всего накопителя (в отличие от сжатия каждого файла на диске). Диски с многими гигабайтами или терабайтами пространства, на которое за весь срок службы накопителя ни разу ничего не записывалось, будут сжиматься лучше, поскольку значительная часть накопителя все еще состоит из нетронутых секторов, заполненных нулями. Активно использовавшиеся диски будут сжиматься хуже, если большинство секторов накопителя выделялось на протяжении его срока службы и все еще содержит остаточные данные. Образы дисков с большим числом звуковых и видеофайлов также будут сжиматься плохо, поскольку эти файлы уже сжаты собственными алгоритмами.

Важно выбрать наиболее подходящий и эффективный инструмент и метод сжатия. У некоторых инструментов могут быть ограничения размера файла - исходного файла или сжатого файла назначения. Другие инструменты могут быть неэффективными или использовать временные файлы во время сжатия, вызывая исчерпание памяти или проблемы с дисковым пространством. Некоторые из этих проблем при выполнении сжатия можно решить с помощью конвейеров и перенаправления.

Одна из самых полезных возможностей работы со сжатым криминалистическим образом - возможность применять к нему криминалистические инструменты без распаковки всего образа. Но некоторые инструменты сжатия создают проблему, поскольку не умеют выполнять поиск внутри сжатого файла. Поиск позволяет программе произвольно обращаться к любой точке файла. Криминалистические форматы разработаны так, чтобы аналитические программы могли на лету произвольно обращаться к сжатым образам. Все популярные криминалистические форматы поддерживают сжатие образов, которое обычно выполняется во время получения, хотя не все инструменты сжимают данные по умолчанию.

Стандартные инструменты сжатия Linux

Среди широко используемых сегодня в мире открытого исходного кода инструментов сжатия - `zip`, `gzip` и `bzip` (версии 1 или 2). В примерах этого раздела используется `gzip`, но можно применять и другие инструменты сжатия. Для попытки добиться лучшего сжатия ценой времени и циклов процессора можно регулировать уровень сжатия.

При наличии достаточного дискового пространства файл образа диска можно просто сжать на месте:

```
$ gzip image.raw
```

Эта команда создает файл `image.raw.gz` и после завершения удаляет исходный файл. Во время сжатия должно быть достаточно места для одновременного существования сжатого и несжатого файлов. То же относится к распаковке файлов с помощью `gunzip`.

Образы можно также сжимать на лету во время получения с помощью конвейеров и перенаправления. Например:

```
# dcfldd if=/dev/sde | gzip > image.raw.gz
```

Здесь входным файлом является необработанное дисковое устройство. Если для `dcfldd` не указать выходной файл, поток данных образа отправляется в `stdout`, откуда передается по конвейеру в `gzip`, а затем перенаправляется в сжатый файл.

Сжатый файл можно распаковать в файл необработанного образа, с которым непосредственно работают криминалистические инструменты. В качестве альтернативы распакованный поток можно передать по конвейеру некоторым программам через `stdout` и `stdin`. Например:

```
$ zcat image.raw.gz | sha256sum
1b52ab6c1ff8f292ca88404acfc9f576ff9db3c1bbeb73e50697a4f3bbf42dd0 -
```

Здесь `zcat` распаковывает образ и передает его по конвейеру программе для создания криптографического хеша SHA256. Следует отметить, что формат файла `gzip` содержит дополнительные метаданные, например временную метку создания, исходное имя файла и другие сведения. Хеш контейнера `gzip` (`image.raw.gz`) при каждом создании будет разным, хотя хеш находящегося внутри сжатого файла останется тем же.

Сжатый формат EnCase EWF

Инструмент `ewfacquire` предоставляет флаги для управления сжатием в процессе получения данных. Например:

```
# ewfacquire -c bzip2:best -f encase7-v2 /dev/sdj
ewfacquire 20150126
...
EWF file format: EnCase 7 (.Ex01)
Compression method: bzip2
Compression level: best
...
MD5 hash calculated over data: 9749f1561dacd9ae85ac0e08f4e4272e
ewfacquire: SUCCESS
```

В этом примере флаг `-c` может задавать алгоритм сжатия вместе с уровнем сжатия. Здесь использован алгоритм `bzip2`, настроенный на наилучший возможный уровень сжатия. Поскольку `bzip2` поддерживают только форматы EWFv2, параметром была указана версия формата `encase7-v2`. Обратите внимание, что `ewftools` необходимо компилировать с поддержкой `bzip2`.[1]

Сжатый формат FTK SMART

Инструмент командной строки `ftkimager` поддерживает сжатые образы во время получения данных, как показано в следующем примере:

```
# ftkimager --compress 9 --s01 /dev/sdj image
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

Сжатые контейнеры доказательств SquashFS

Напомним, что использование SquashFS как контейнера криминалистических доказательств описано в главе 6. При создании файла SquashFS можно настраивать несколько параметров сжатия. Доступны три алгоритма сжатия (`gzip`, `lzo`, `xz`), можно сжимать различные метаданные SquashFS (таблицу inode, расширенные атрибуты) и выполнять другие настройки. Дополнительные сведения см. на странице руководства `squashfs(1)`.

В этом примере файл необработанного образа был преобразован в сжатый файл SquashFS:

```
# mksquashfs image.raw image.sfs -comp lzo -noI
Parallel mksquashfs: Using 8 processors
Creating 4.0 filesystem on image.sfs, block size 131072.
...
Exportable Squashfs 4.0 filesystem, lzo compressed, data block size 131072
compressed data, uncompressed metadata, compressed fragments, compressed
xattrs
duplicates are removed
Filesystem size 615435.77 Kbytes (601.01 Mbytes)
31.45% of uncompressed filesystem size (1956923.96 Kbytes)
Inode table size 61232 bytes (59.80 Kbytes)
100.00% of uncompressed inode table size (61232 bytes)
Directory table size 31 bytes (0.03 Kbytes)
100.00% of uncompressed directory table size (31 bytes)
...
```

Здесь флаг `-comp` задает алгоритм сжатия `lzo` (`gzip` используется по умолчанию), а флаг `-noI` предотвращает сжатие inode (контейнера SquashFS, а не образа доказательств).

Сценарий оболочки `sfsimage` управляет созданием контейнеров криминалистических доказательств SquashFS с несколькими дополнительными криминалистическими функциями.

Применение сжатия имеет основополагающее значение при работе с большими криминалистическими образами. Однако даже сжатые образы все равно могут быть очень большими и неудобными в управлении. Есть и другой способ облегчить процесс: криминалистические образы можно разделить на несколько меньших частей.

Управление разделенными образами

Управление полученными образами дисков может быть затруднено из-за большого размера файлов. Разбиение образа на меньшие, более удобные части может помочь решить эту проблему. Рассмотрим следующие примеры, когда разделенный образ может быть полезен:

• Передачу по нестабильным сетевым соединениям можно выполнять несколькими небольшими загрузками разделенного образа.

• Большой образ может превышать максимальный размер файла, поддерживаемый программным инструментом. Разделение образа позволяет обойти ограничение.
• Носители информации, например ленты, CD или DVD, имеют фиксированную максимальную емкость. Разделенные образы позволяют использовать набор таких носителей.
• У некоторых файловых систем (особенно FAT) относительно небольшой максимальный размер файла.

Наиболее распространенное применение разделенных образов в цифровой криминалистике - передача и хранение доказательств. Исторически для этого образ записывался на набор CD или DVD.

Команда GNU split

В стандартных системах Unix и Linux имеется инструмент `split` для разбиения большого файла на несколько меньших. В следующем примере команда `split` разбивает существующий образ на фрагменты размером с DVD:

```
$ split -d -b 4G image.raw image.raw.

$ ls
image.raw.00 image.raw.01 image.raw.02 image.raw.03 image.raw.04
...
```

Флаг `-d` указывает добавить числовое расширение к `image.raw.` (обратите внимание на точку в конце), а флаг `-b` задает размер фрагментов, создаваемых из файла `image.raw`.

Используя сочетание конвейерной передачи между несколькими инструментами, можно одновременно выполнять сжатие и разделение во время получения данных, экономя время и место. Вот пример одной команды, получающей образ с помощью `dd`, сжимающей его с помощью `gzip` и разделяющей на фрагменты размером с CD:

```
# dd if=/dev/sdb | gzip | split -d -b 640m - image.raw.gz.
```

Входным файлом команды `split` является `-`, обозначающий `stdin`; команда разбивает сжатый поток байтов на части. Важно отметить, что части не сжаты `gzip` по отдельности и не могут распаковываться по отдельности. Перед распаковкой части необходимо собрать заново.

Разделение образов во время получения

Создаваемый образ жесткого диска можно разделять на части непосредственно в процессе получения, а не отдельным последующим этапом. Перед получением большого диска подумайте, может ли в будущем потребоваться разделенный образ и какой размер фрагмента будет наиболее разумным. Правильный изначально разделенный образ может сэкономить время и дисковое пространство в ходе исследования.

Разделенные образы распространены в цифровой криминалистике и поэтому хорошо поддерживаются инструментами криминалистического получения и анализа. Как правило, флаги позволяют задавать размер фрагмента и настраивать расширение разделенного образа.

Инструмент `dcfldd` имеет встроенную функцию разделения. Например, если позднее образ будет передан третьей стороне на наборе USB-накопителей по 16 Гбайт, его можно получить с помощью `dcfldd`, указав флаг `split=16G` перед выходным файлом:

```
# dcfldd if=/dev/sdc split=16G of=image.raw

# ls
image.raw.000 image.raw.001 image.raw.002 image.raw.003 image.raw.004
...
```

Стандартное расширение представляет собой трехзначное число, добавляемое к имени выходного файла.

С помощью инструмента `dc3dd` образы можно разделять во время получения, задавая размер выхода параметром `ofsz=`. Расширения файлов являются числовыми, как показано здесь:

```
# dc3dd if=/dev/sdh ofsz=640M ofs=image.raw.000

# ls -l
total 7733284
-rw-r----- 1 root root 671088640 Jan 14 10:59 image.raw.000
-rw-r----- 1 root root 671088640 Jan 14 10:59 image.raw.001
-rw-r----- 1 root root 671088640 Jan 14 10:59 image.raw.002
...
-rw-r----- 1 root root 671088640 Jan 14 11:00 image.raw.009
-rw-r----- 1 root root 671088640 Jan 14 11:00 image.raw.010
-rw-r----- 1 root root 536870912 Jan 14 11:00 image.raw.011
```

Убедитесь, что в расширении файла достаточно нулей, иначе `dc3dd` не сможет завершить работу и создаст сообщение об ошибке, например `[!!] file extensions exhausted for image.raw.0`. Последний файл набора обычно меньше остальных (если только размер образа не делится без остатка на размер разделенного файла).

Инструменты EnCase обычно по умолчанию разделяют образы во время получения. Получить диск в разделенный образ EnCase с помощью `ewfacquire` можно, задав максимальный размер файла сегмента флагом `-S`:

```
# ewfacquire -S 2G /dev/sdc
...

# ls
image.E01 image.E02 image.E03 image.E04 image.E05 image.E06
...
```

После этого коммерческий криминалистический комплект EnCase может непосредственно использовать эти образы.

Инструмент `ftkimager` предоставляет флаг `--frag` для сохранения образа по частям во время получения, как показано в этом примере:

```
# ftkimager /dev/sdk image --frag 20GB --s01
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.
...

# ls -l
total 53771524
-rw-r----- 1 holmes root 2147442006 Jul 2 08:01 image.s01
-rw-r----- 1 holmes root 1038 Jul 2 08:43 image.s01.txt
-rw-r----- 1 holmes root 2147412323 Jul 2 08:01 image.s02
-rw-r----- 1 holmes root 2147423595 Jul 2 08:02 image.s03
-rw-r----- 1 holmes root 2147420805 Jul 2 08:02 image.s04
...
```

Здесь диск получен с максимальным размером фрагмента 20 Гбайт, а форматом является сжатый образ SMART. Обратите внимание на добавленный файл `*.txt`, содержащий метаданные. В отличие от некоторых криминалистических форматов, эти сведения не встроены в разделенные файлы FTK, создаваемые `ftkimager`.

Доступ к набору файлов разделенного образа

Криминалистические инструменты, например Sleuth Kit, поддерживают непосредственную работу с набором разделенных образов без необходимости сначала собирать их. Чтобы перечислить поддерживаемые образы в Sleuth Kit, примените флаг `-i list` с любым инструментом Sleuth Kit для обработки образов:

```
$ mmls -i list
Supported image format types:
raw (Single raw file (dd))
aff (Advanced Forensic Format)
afd (AFF Multiple File)
afm (AFF with external metadata)
afflib (All AFFLIB image formats (including beta ones))
ewf (Expert Witness format (encase))
split (Split raw files)
```

В этом примере поддерживаются разделенные необработанные образы (включая файлы Unix `split`), разделенные образы AFF и разделенные файлы EnCase (хотя явно это не сказано, разделенные файлы EnCase поддерживаются). Некоторые из этих типов форматов образов, возможно, потребуется явно включить при компиляции Sleuth Kit.

В следующем примере образ EWF разделен на 54 части. Выполнение команды `img_stat` для первого файла предоставляет сведения о полном наборе файлов:

```
$ img_stat image.E01
IMAGE FILE INFORMATION
--------------------------------------------
Image Type: ewf
Size of data in bytes: 121332826112
MD5 hash of data: ce85c1dffc2807a205f49355f4f5a029
```

С помощью различных инструментов можно непосредственно работать с разделенными образами. Большинство команд Sleuth Kit работает с набором разделенных необработанных файлов, если указан первый файл типа разделенного образа.

Современные версии Sleuth Kit автоматически проверяют наличие наборов разделенных файлов:

```
$ mmls image.raw.000
```

В более ранних версиях Sleuth Kit может потребоваться указать тип разделенного образа:

```
$ fls -o 63 -i split image.000 image.001 image.002
```

Чтобы проверить, распознан ли набор разделенных файлов, команда `img_stat` покажет общее число распознанных байтов, а для необработанных типов - диапазоны смещений байтов каждой части:

```
$ img_stat image.raw.000
IMAGE FILE INFORMATION
--------------------------------------------
Image Type: raw
Size in bytes: 2003828736
--------------------------------------------
Split Information:
image.raw.000 (0 to 16777215)
image.raw.001 (16777216 to 33554431)
image.raw.002 (33554432 to 50331647)
image.raw.003 (50331648 to 67108863)
image.raw.004 (67108864 to 83886079)
...
```

Альтернативный способ определить поддержку разделенных файлов - запустить команду или инструмент с `strace -e open` и проверить, открывает ли он каждую часть разделенного файла.

Разделение файлов и работа с набором разделенных файлов полезны, но иногда их необходимо собрать в единый образ. Это показано в следующем разделе.

Сборка разделенного образа

Собирать разделенные криминалистические форматы обычно не требуется, поскольку инструменты, совместимые с определенным криминалистическим форматом (EWF, SMART или AFF), должны поддерживать разделенные файлы.

Поскольку необработанный образ не содержит заголовка или метаданных, сборка сводится к объединению набора фрагментов образа в единый образ. Аккуратно выполнять это следует в два этапа, как показано здесь:

```
$ ls -1 image.raw.*
image.raw.000
image.raw.001
image.raw.002
image.raw.003
...

$ cat image.raw.* > image.raw
```

Флаг `ls -1` перечисляет файлы, распознанные шаблоном подстановки оболочки. Обязательно подтвердите полноту и правильный порядок списка, прежде чем объединять файлы в единый образ. Если части разделенного образа отсутствуют или порядок файлов неверен, собранные части не образуют правильный криминалистический образ.

Если вы получили стопку DVD, каждый из которых содержит фрагмент сжатого необработанного образа, собрать их можно следующим образом:

```
$ cat /dvd/image.raw.gz.00 > image.raw.gz
$ cat /dvd/image.raw.gz.01 >> image.raw.gz
$ cat /dvd/image.raw.gz.02 >> image.raw.gz
$ cat /dvd/image.raw.gz.03 >> image.raw.gz
...
```

Здесь DVD последовательно вставляются и монтируются в `/dvd`, а разделенные части добавляются до восстановления файла образа. Обратите внимание: `>` в первой команде `cat` создает файл образа, а `>>` в последующих командах добавляет данные (не перезаписывая его). После добавления всех частей в файл назначения криптографический хеш распакованного образа должен совпасть с полученным во время приобретения.

Набор разделенных файлов сжатого образа можно также распаковать и собрать, передав все разделенные файлы по конвейеру в `zcat` и перенаправив вывод в файл:

```
# cat image.raw.gz.* | zcat > image.raw
```

Полезный метод AFFlib позволяет виртуально собрать набор фрагментов с помощью файловой системы FUSE. Инструмент `affuse` может представить набор разделенных файлов как полностью собранный файл необработанного образа следующим образом:

```
# ls
image.raw.000 image.raw.011 image.raw.022 image.raw.033 image.raw.044
image.raw.001 image.raw.012 image.raw.023 image.raw.034 image.raw.045
...
#

# affuse image.raw.000 /mnt

# ls -l /mnt
total 0
-r--r--r-- 1 root root 8011120640 1970-01-01 01:00 image.raw.000.raw
```

Здесь каталог с необработанными файлами представлен как единый файл образа диска, находящийся в виртуальной файловой системе `/mnt`. С этим необработанным файлом можно непосредственно работать криминалистическими инструментами.

Проверка целостности криминалистического образа

Проверка криптографического хеша образа имеет основополагающее значение для цифровой криминалистики и составляет основу сохранения цифровых доказательств. В этом разделе приведены примеры проверки криптографических хешей и подписей образа.

Проверка сохранности доказательств предполагает подтверждение того, что текущий криптографический хеш образа идентичен хешу, полученному ранее. Хеширование можно использовать для проверки успешности операции с диском или образом (получения, преобразования, передачи, резервного копирования и так далее). Его также можно использовать для проверки того, что диск или файл образа не подвергался вмешательству в течение более продолжительного времени (месяцев или даже лет).

Требования к хешированию (процедуры и алгоритмы) зависят от правовой юрисдикции применения и организационных правил, регулирующих криминалистическую лабораторию. Поэтому рекомендации по хешированию здесь не приводятся.

Проверка хеша, полученного во время создания образа

Если после получения диска необходимо проверить хеш получения, это простая (но, возможно, длительная) задача передачи содержимого диска по конвейеру программе криптографического хеширования. Использование другой программы для проверки хеша диска обеспечивает независимую проверку на уровне инструментов. Например:

```
# img_stat image.E01
IMAGE FILE INFORMATION
-----------------------------------------
Image Type: ewf
Size of data in bytes: 2003828736
MD5 hash of data: 9749f1561dacd9ae85ac0e08f4e4272e

# dd if=/dev/sdj | md5sum
3913728+0 records in
3913728+0 records out
9749f1561dacd9ae85ac0e08f4e4272e -
2003828736 bytes (2.0 GB) copied, 126.639 s, 15.8 MB/s
```

Здесь вывод `img_stat` показывает хеш MD5 получения данных, записанный инструментом создания образов EnCase. Затем для повторного вычисления хеша необработанного дискового устройства используется второй инструмент, обычный `dd`. В этом примере два хеша MD5 совпадают, подтверждая сохранение целостности доказательств.

Повторное вычисление хеша криминалистического образа

Каждый из криминалистических форматов и криминалистических инструментов на основе `dd` может записывать или регистрировать значение хеша образа диска. Для проверки записанного хеша можно повторно вычислить хеш образа диска. В следующем примере хеш был записан во время получения с помощью `dc3dd` и сохранен в `hashlog.txt`. Хеш можно проверить следующим образом:

```
# grep "(md5)" hashlog.txt
5dfe68597f8ad9f20600a453101f2c57 (md5)

# md5sum image.raw
5dfe68597f8ad9f20600a453101f2c57 image.raw
```

Хеши совпадают, подтверждая согласованность файла доказательств и журнала хешей и тем самым указывая, что целостность доказательств сохранена.

Следующий пример проверяет образ по хешу, хранящемуся в метаданных формата EnCase. В этом примере для проверки хеша используется специализированный инструмент `ewfverify`:

```
# ewfverify image.Ex01
ewfverify 20160424

Verify started at: May 14, 2016 14:47:32
This could take a while.
...
MD5 hash stored in file: 5dfe68597f8ad9f20600a453101f2c57
MD5 hash calculated over data: 5dfe68597f8ad9f20600a453101f2c57
ewfverify: SUCCESS
```

Здесь повторно вычисленный хеш совпадает, подтверждая согласованность файла образа EWF. Этот инструмент автоматически проверяет хеш набора разделенных файлов криминалистического формата EnCase.

Инструмент `affinfo` выполняет сходную проверку действительности файлов AFF. В этом примере проверяется хеш SHA1:

```
$ affinfo -S image.aff
image.aff is a AFF file
...
Validating SHA1 hash codes.
computed sha1: 9871 0FB5 531E F390 2ED0 47A7 5BE4 747E 6BC1 BDB0
stored sha1: 9871 0FB5 531E F390 2ED0 47A7 5BE4 747E 6BC1 BDB0 MATCH
```

Этот вывод подтверждает, что хеш образа внутри файла AFF совпадает с хешем, записанным в метаданных AFF.

Криптографические хеши разделенных необработанных образов

Вычислить криптографический хеш набора необработанных разделенных файлов несложно: объединенные части можно передать по конвейеру программе хеширования. В этом примере вычисляется хеш SHA256 набора разделенных необработанных файлов:

```
$ cat image.raw.* | sha256sum
12ef4b26e01eb306d732a314753fd86de099b02105ba534d1b365a232c2fd36a -
```

В этом примере предполагается, что имена файлов частей можно отсортировать в правильном порядке (здесь это можно проверить командой `ls -1 image.raw.*`). Команда `cat` необходима, поскольку она объединяет (собирает) все части перед передачей по конвейеру в `sha256sum`.

Криптографический хеш образа, который был сжат и разделен на части, можно проверить, создав конвейер команд из нескольких программ. В следующем примере `cat` собирает образ и передает его по конвейеру в `zcat` для распаковки. Вывод `zcat` отправляется программе хеширования, которая после завершения создает значение хеша:

```
$ cat image.raw.gz.* | zcat | md5sum
9749f1561dacd9ae85ac0e08f4e4272e -
```

Здесь команда `cat` необходима, поскольку она объединяет все разделенные части перед передачей в `zcat`. Команда `zcat image.raw.gz.*` завершится неудачей, поскольку попытается распаковать каждую часть, а не собранный образ.

В сообществе Unix выражение useless use of cat (UUOC) обозначает использование команды `cat` для отправки файла команде в случае, когда вместо нее можно было применить `<`. Традиционные сообщества Unix присуждали награды UUOC, поощряя более эффективное применение перенаправления команд оболочки. Однако в примерах этого раздела `cat` действительно необходима, поскольку выполняет функцию объединения.

Выявление несовпадающих окон хеширования

По мере старения дисков, а также их перевозки и обращения с ними существует риск повреждения, возможно приводящего к появлению поврежденных секторов. Если на исходном диске доказательств после первого создания образа возникают ошибки нечитаемых секторов, криптографическая контрольная сумма диска не совпадет. В этом случае ценны окна хеширования, поскольку они позволяют точнее определить, какая часть диска не совпала. Что еще важнее, окна хеширования могут показать, какие области диска все еще сохранены, даже если хеш всего диска не совпал.

Указанный размер окна хеширования определяет, как часто записывается новый хеш при получении диска или проверке окон хеширования диска. При сравнении двух списков хешей для проверки оба списка должны использовать одинаковый размер окна хеширования. Чтобы найти несовпадающие области, два журнала хешей можно сравнить инструментом Unix `diff`.

В следующем примере образ диска был создан с помощью `dcfldd` и сохранен журнал хешей с размером окна 10 Мбайт. Последующая проверка не смогла подтвердить MD5 всего диска и предоставила новый журнал хешей, также с размером окна 10 Мбайт:

```
$ diff hash1.log hash2.log
3c3
< 20971520 - 31457280: b587779d76eac5711e92334922f5649e
---
> 20971520 - 31457280: cf6453e4453210a3fd8383ff8ad1511d
193c193
< Total (md5): 9749f1561dacd9ae85ac0e08f4e4272e
---
> Total (md5): fde1aa944dd8027c7b874a400a56dde1
```

Этот вывод показывает несовпадающие хеши всего образа и диапазона байтов между 20971520 и 31457280. Деление на размер сектора 512 байт определяет диапазон секторов от 40960 до 61440, где возникло несовпадение хешей. Хеши остальной части диска по-прежнему действительны; криминалистически не сохранены только сектора с несовпадающими хешами. Содержимое (блоки, файлы, части файлов и так далее), находящееся в диапазоне секторов с несовпадающими хешами, на более позднем этапе можно исключить из представляемых доказательств. Если два криптографических хеша полного образа совпадают, можно считать, что все окна хеширования также совпадают.

Криптографические хеши криминалистических образов сохраняют целостность собранных доказательств. Однако сами значения хешей не защищены от злоумышленного или случайного изменения. Целостность вычисленных хешей можно сохранить с помощью криптографического подписания и временных меток. Проверка действительности подписей и временных меток показана в следующем разделе.

Проверка подписи и временной метки

В предыдущей главе было продемонстрировано применение GnuPG для подписания хешей диска. Подпись можно проверить, не располагая закрытым ключом подписавшего лица. Сам человек, подписавший доказательства, не нужен; требуется только его открытый ключ. В этом примере проверяется подпись GPG лица, подписавшего полученный образ диска:

```
$ gpg < hash.log.asc
dc3dd 7.2.641 started at 2016-05-07 17:23:49 +0200
compiled options:
command line: dc3dd if=/dev/sda hof=image.raw ofs=image.000 ofsz=1G hlog=hash.log
hash=md5

input results for device `/dev/sda':
5dfe68597f8ad9f20600a453101f2c57 (md5)
...
dc3dd completed at 2016-05-07 17:25:40 +0200
gpg: Signature made Sat 07 May 2016 17:29:44 CEST using RSA key ID CF87856B
gpg: Good signature from "Sherlock Holmes <holmes@digitalforensics.ch>"
```

Здесь содержимое подписанного сообщения (вывод получения данных и хеш) отображается вместе с сообщением GPG, указывающим, что подпись действительна.

Для сообщений, подписанных S/MIME, сходная команда проверит (или признает недействительной) подпись из файла PEM и выглядит следующим образом:

```
$ gpgsm --verify image.log.pem
gpgsm: Signature made 2016-01-25 19:49:42 using certificate ID 0xFFFFFFFFABCD1234
...
gpgsm: Good signature from "/CN=holmes@digitalforensics.ch/EMail=holmes@
digitalforensics.ch"
gpgsm: aka "holmes@digitalforensics.ch"
```

В главе 6 рассматривалось применение служб временных меток для создания временных меток RFC-3161 центром временных меток. Проверка временной метки сходна с проверкой подписи S/MIME и требует установки правильной цепочки сертификатов центра сертификации (CA), чтобы проверка прошла успешно. В этом примере проверяется созданная ранее временная метка FreeTSA (http://freetsa.org/).

Если сертификат CA службы временных меток не установлен в системе, его можно получить вручную. Сертификат TSA должен был вернуться в составе временной метки при выполнении запроса (из-за флага `-cert`). Для этого примера сертификат CA загружается из FreeTSA следующим образом:

```
$ curl http://freetsa.org/files/cacert.pem > cacert.pem
```

Если сертификаты CA и TSA доступны OpenSSL и действительны, временную метку можно проверить следующим образом:

```
$ openssl ts -verify -in hash.log.tsr -queryfile hash.log.tsq -CAfile cacert.pem
Verification: OK
```

Команда OpenSSL `ts` используется для проверки временной метки. Передаются запрос временной метки (`tsq`) и ответ временной метки (`tsr`), а в этом примере указывается файл с сертификатом CA сервера временных меток. Временная метка третьей стороны действительна (`Verification: OK`), что указывает, что файл (и содержащиеся в нем хеши криминалистического получения) не был изменен с указанного времени. Если определенный центр временных меток предполагается использовать постоянно, сертификаты CA можно добавить в доверенное хранилище CA операционной системы.

AFFlib также предусматривает подписание и проверку подписей полученных образов с помощью сертификатов X.509.

В этом разделе не рассматривались сеть доверия или инфраструктура открытых ключей (PKI), необходимые для доверия ключам, применяемым для подписания образов и проверки временных меток. В примерах предполагается, что такое доверие уже установлено.

Преобразование между форматами образов

Преобразование между форматами криминалистических образов может быть выгодно по разным причинам. Если в лаборатории появилось новое программное обеспечение или инфраструктура, а текущий формат не поддерживается либо менее эффективен, можно рассмотреть преобразование в другой формат. Если образ будет передан третьей стороне, у нее может быть предпочтительный формат образа. Если вы получаете образ от третьей стороны, возможно, захотите преобразовать его в предпочитаемый вами формат. В этом разделе приведены примеры преобразования между форматами в командной строке. Показано преобразование из нескольких исходных форматов, включая EnCase, FTK, AFF и необработанные образы. Кроме того, примеры демонстрируют преобразование различных форматов в контейнеры доказательств SquashFS.

При преобразовании между форматами образов предпочтительно использовать конвейеры и перенаправление. Избегайте инструментов, использующих временные файлы. Во время преобразования могут одновременно существовать две копии образа (одна или обе могут быть сжаты). При подготовке к преобразованию выполните планирование емкости.

После преобразования проверьте совпадение значений хешей исходного образа и назначения.

Преобразование из необработанных образов

Преобразовать необработанный образ в другой формат обычно несложно, поскольку можно использовать обычную функцию создания образа диска. Вместо имени необработанного устройства применяется имя файла необработанного образа.

В следующих примерах файл необработанного образа преобразуется в форматы EnCase и FTK. В первом примере `ewfacquire` преобразует `image.raw` в формат EnCase Expert Witness:

```
$ ewfacquire image.raw -t image -f encase7
ewfacquire 20160424

Storage media information:
Type: RAW image
Media size: 7.9 GB (7918845952 bytes)
Bytes per sector: 512

Acquiry parameters required, please provide the necessary input
Case number: 42

Description: The case of the missing red stapler
Evidence number: 1
Examiner name: S. Holmes
Notes: This red USB stick was found at the scene
...
Acquiry completed at: May 14, 2016 15:03:40
Written: 7.3 GiB (7918846140 bytes) in 54 second(s) with 139 MiB/s
(146645298 bytes/second)
MD5 hash calculated over data: 5dfe68597f8ad9f20600a453101f2c57
ewfacquire: SUCCESS
```

Здесь указанным исходным файлом является необработанный образ, а `-t` задает базовое имя целевых файлов EnCase `*.e01`. Была указана версия EnCase 7, а при выполнении команда задает ряд вопросов. Поскольку необработанный файл не содержит метаданных дела, их необходимо ввести вручную.

Преобразование необработанного образа в FTK SMART сходно: необработанный образ задается как источник, а метаданные дела добавляются вручную. При использовании `ftkimager` метаданные дела указываются в командной строке, как показано в этом примере:

```
$ ftkimager image.raw image --s01 --case-number 1 --evidence-number 1 --description
"The case of the missing red stapler" --examiner "S. Holmes" --notes "This USB
stick was found at the scene"
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

Флаг `--s01` задает создание сжатого образа SMART. Базовое имя файла задается просто как `image`, а соответствующие расширения файлов добавляются автоматически.

Преобразование образа в контейнер криминалистических доказательств SquashFS также выполняется одной простой командой при использовании сценария `sfsimage`:

```
$ sfsimage -i image.raw image.sfs
Started: 2016-05-14T15:14:13
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i image.raw
Current working directory: /exam
Forensic evidence source: if=/exam/image.raw
Destination squashfs container: image.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd if=/exam/image.raw log=errorlog.txt hlog=hashlog.txt
hash=md5 2>/dev/null | pv -s 7918845952

7.38GiB 0:00:22 [ 339MiB/s] [================================>] 100%
Completed: 2016-05-14T15:14:37
```

Здесь файл необработанного образа был указан вместе с именем файла контейнера SquashFS назначения. Сценарий `sfsimage` создает требуемое псевдоустройство SquashFS и добавляет сведения журнала и хешей как обычные текстовые файлы. Дополнительные метаданные дела можно вручную добавить в контейнер доказательств (с помощью `sfsimage -a`).

К обычному необработанному образу, сжатому `gzip`, нельзя обращаться напрямую типичными криминалистическими инструментами из-за невозможности выполнять поиск (произвольно обращаться к любому блоку файла) внутри файла `gzip`. Такие файлы лучше преобразовывать в сжатые форматы с поддержкой поиска. Тогда с ними можно работать непосредственно криминалистическими аналитическими инструментами. В этом примере необработанный файл образа, сжатый `gzip`, преобразуется с помощью `sfsimage` в сжатый файл SquashFS:

```
$ zcat image.raw.gz | sfsimage -i - image.sfs
Started: 2016-05-14T15:20:39
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i -
Current working directory: /exam
Forensic evidence source:
Destination squashfs container: image.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd log=errorlog.txt hlog=hashlog.txt hash=md5
2>/dev/null | pv -s 0
7.38GiB 0:00:38 [ 195MiB/s] [ <=> ]
Completed: 2016-05-14T15:21:18
```

Исходный файл остается в необработанном виде, но теперь находится внутри сжатой файловой системы. Полученный файл `*.sfs` можно смонтировать для доступа к необработанному образу, как показано здесь:

```
$ sfsimage -m image.sfs
image.sfs.d mount created

$ ls image.sfs.d/
errorlog.txt hashlog.txt image.raw sfsimagelog.txt
```

Файл необработанного образа можно преобразовать в файл AFF простой командой `affconvert`:

```
$ affconvert image.raw
convert image.raw --> image.aff
Converting page 119 of 119
md5: 9749f1561dacd9ae85ac0e08f4e4272e
sha1: 98710fb5531ef3902ed047a75be4747e6bc1bdb0
bytes converted: 2003828736
Total pages: 120 (117 compressed)
Conversion finished.
```

После этого метаданные дела можно добавить отдельным инструментом, например `affsegment`. Инструмент `affconvert` предоставляет разумные стандартные параметры сжатия, а полученный файл имеет расширение `*.aff` и базовое имя необработанного файла.

Следующий, заключительный пример показывает преобразование необработанного образа внутри контейнера криминалистических доказательств SquashFS в файл AFF с помощью команды `affconvert`:

```
# affconvert -Oaff image.sfs.d/image.raw
convert image.sfs.d/image.raw --> aff/image.aff
Converting page 953 of 953
md5: d469842a3233cc4e7d4e77fd81e21035
sha1: 9ad205b1c7889d0e4ccc9185efce2c4b9a1a8ec6
bytes converted: 16001269760
Total pages: 954 (954 compressed)
Conversion finished.
```

Поскольку SquashFS доступна только для чтения, необходимо указать `affconvert` записывать выходной файл в другой каталог, доступный для записи.

Преобразование из формата EnCase/E01

Пакет `libewf` содержит инструмент `ewfexport` для преобразования файлов EnCase EWF (`*.E0*`) в другие форматы. Это включает возможность читать один или несколько файлов и передавать их по конвейеру другим программам.

> **Примечание.** В некоторых старых версиях `ewfexport` есть ошибка, из-за которой строка `ewfexport: SUCCESS` добавляется в конец образа после экспорта в `stdout`. Эта добавленная строка вызывает несовпадение хешей MD5 образа. Строка имеет фиксированную длину 19 байт, поэтому ее можно подавить, передав поток через `tail -c 19`.

Создание контейнера SquashFS вручную

На протяжении книги вы видели примеры сценария оболочки `sfsimage`. Но полезно рассмотреть один пример создания файла SquashFS без сценария. Следующий пример позволит лучше понять внутреннюю работу `sfsimage`.

Следующее получение EnCase содержит 54 файла `*.E0`, которые будут собраны в единый необработанный образ и помещены в контейнер доказательств SquashFS:

```
# ls
image.E01 image.E10 image.E19 image.E28 image.E37 image.E46
image.E02 image.E11 image.E20 image.E29 image.E38 image.E47
image.E03 image.E12 image.E21 image.E30 image.E39 image.E48
image.E04 image.E13 image.E22 image.E31 image.E40 image.E49
image.E05 image.E14 image.E23 image.E32 image.E41 image.E50
image.E06 image.E15 image.E24 image.E33 image.E42 image.E51

image.E07 image.E16 image.E25 image.E34 image.E43 image.E52
image.E08 image.E17 image.E26 image.E35 image.E44 image.E53
image.E09 image.E18 image.E27 image.E36 image.E45 image.E54
```

Для начала нужен файл псевдоопределений `mksquashfs`, задающий команды, которые создадут файлы внутри контейнера SquashFS. Файл псевдоопределений содержит имя целевого файла, тип файла, разрешения, владельца и выполняемую команду. Вывод этой команды станет содержимым определенного имени файла внутри файловой системы SquashFS.

В следующем примере создан файл `pseudo_files.txt`, содержащий два определения. Первое извлекает метаданные EnCase с помощью `ewfinfo` и помещает их в `image.txt` (иначе эти метаданные были бы потеряны). Второе определение экспортирует необработанный образ из файлов `*.E0` в `image.raw`:

```
# cat pseudo_files.txt
image.txt f 444 root root ewfinfo image.E01
image.raw f 444 root root ewfexport -u -t - image.E01
```

Флаг `ewfexport -u` позволяет выполнять преобразование без участия пользователя (иначе инструмент задает пользователю вопросы). Флаг `-t` задает назначение, которым в этом примере является `stdout`, или дефис `-`.

С помощью этого файла определений можно создать сжатую файловую систему с созданными файлами следующим образом:

```
# mksquashfs pseudo_files.txt image.sfs -pf pseudo_files.txt
Parallel mksquashfs: Using 12 processors
Creating 4.0 filesystem on image.sfs, block size 131072.
ewfexport 20160424

Export started at: May 12, 2016 19:09:42
This could take a while.
...
Export completed at: May 12, 2016 19:28:56
Written: 113 GiB (121332826112 bytes) in 19 minute(s) and 14 second(s) with
100 MiB/s (105141097 bytes/second)
MD5 hash calculated over data: 083e2131d0a59a9e3b59d48dbc451591
ewfexport: SUCCESS
...
Filesystem size 62068754.40 Kbytes (60614.02 Mbytes)
52.38% of uncompressed filesystem size (118492706.13 Kbytes)
...
```

Полученная файловая система SquashFS `image.sfs` будет содержать три файла: файл необработанного образа `image.raw`, файл `image.txt` с метаданными и файл `pseudo_files.txt` с определениями и выполненными командами.

Дополнительные сведения о флагах и параметрах создания файловых систем SquashFS приведены на странице руководства `mksquashfs(1)`.

Содержимое файла SquashFS можно просмотреть командой `unsquashfs` следующим образом:

```
# unsquashfs -lls image.sfs
...
-r--r--r-- root/root 121332826112 2016-05-12 19:09 squashfs-root/image.raw
-r--r--r-- root/root 770 2016-05-12 19:09 squashfs-root/image.txt
-rw-r----- root/root 98 2016-05-12 16:58 squashfs-root/
pseudo_files.txt
```

Заключительный шаг - проверка сохранения доказательств путем сравнения значений хеша MD5. Команда `ewfinfo` предоставляет хеш MD5, вычисленный во время исходного получения EnCase. Вторую контрольную сумму MD5 можно вычислить с помощью `md5sum` для вновь преобразованного необработанного образа внутри контейнера SquashFS. Для этого сначала необходимо смонтировать файловую систему SquashFS. В следующем примере показан каждый из этих шагов:

```
# ewfinfo image.E01
ewfinfo 20160424
...
Digest hash information
MD5: 083e2131d0a59a9e3b59d48dbc451591

# mkdir image.sfs.d; mount image.sfs image.sfs.d

# md5sum image.sfs.d/image.raw
083e2131d0a59a9e3b59d48dbc451591 image.sfs.d/image.raw
```

Результат показывает совпадение двух хешей MD5, указывающее на успешное сохранение доказательств при преобразовании из EnCase в необработанный образ внутри контейнера SquashFS. Третий совпадающий хеш MD5 виден в выводе `ewfexport`, где он был вычислен в процессе преобразования.

Инструмент `ewfexport` также может преобразовывать или экспортировать в другие форматы EnCase. Когда смонтированная файловая система SquashFS `image.sfs.d` больше не нужна, ее можно размонтировать командой `umount image.sfs.d`. Сценарий `sfsimage` управляет этими шагами за вас.

Преобразование файлов из EnCase в FTK

Инструмент `ftkimager` может преобразовывать EnCase в FTK. В этом примере набор файлов EnCase `*.e01` преобразуется в файлы SMART со сжатием `ew` с тем же именем, но расширением `*.s01`:

```
# ftkimager image.E01 image --s01
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

Хеши проверяются и добавляются в новый файл FTK. Исходные метаданные дела не добавляются во вновь преобразованные файлы. Вместо этого они извлекаются из исходного формата и сохраняются как отдельный файл с тем же именем, но расширением `*.txt` (`image.s01.txt` в этом примере).

Преобразование из формата FTK

Инструмент командной строки `ftkimager` преобразует форматы EnCase и FTK друг в друга и позволяет использовать `stdin` и `stdout` для преобразования с файлами необработанных образов.

В следующем примере набор сжатых файлов FTK SMART `*.s01` преобразуется в формат EnCase EWF `*.E01`:

```
# ftkimager image.s01 image --e01
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

Метаданные дела не переносятся в новый формат, но автоматически сохраняются в отдельный файл (`image.E01.txt`).

`ftkimager` может преобразовывать файлы SMART `*.s01` в `stdout`, откуда их можно перенаправить в файлы необработанных образов или передать по конвейеру другим программам. В следующем примере набор файлов FTK SMART преобразуется в контейнер криминалистических доказательств SquashFS посредством передачи вывода `ftkimager` по конвейеру в `sfsimage`:

```
# ftkimager sandisk.s01 - | sfsimage -i - sandisk.sfs
Started: 2016-05-12T19:59:13
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i -
Current working directory: /exam
Forensic evidence source:
Destination squashfs container: sandisk.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd log=errorlog.txt hlog=hashlog.txt hash=md5
2>/dev/null | pv -s 0
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.
14.5GiB 0:01:37 [ 151MiB/s] [ <=> ]
Completed: 2016-05-12T20:00:51

# sfsimage -a sandisk.s01.txt sandisk.sfs
Appending to existing 4.0 filesystem on sandisk.sfs, block size 131072
```

При преобразовании из формата FTK в необработанный образ диска метаданные дела не переносятся. Метаданные дела, обычно находящиеся в отдельном текстовом файле, необходимо сохранить вручную. Их можно добавить в контейнер SquashFS, как показано в предыдущем примере с командой `sfsimage -a`.

После выполнения преобразования формата любого вида следует отдельно проверить значение хеша формата назначения, чтобы убедиться в сохранении целостности доказательств.

Преобразование из формата AFF

Инструмент `affconvert` может преобразовывать образы AFF в необработанный образ (и необработанный образ в формат AFF). `affconvert` не использует `stdin` или `stdout`, а читает или создает самостоятельные файлы. Следующий простой пример показывает преобразование файла AFF в необработанный образ:

```
$ affconvert -r image.aff
convert image.aff --> image.raw
Converting page 96 of 96
bytes converted: 1625702400
Conversion finished.
```

Чтобы преобразовать необработанный образ в формат AFF, просто выполните `affconvert image.raw`, и будет создан соответствующий файл `image.aff`.

Для конвейеров и перенаправления с файлами AFF можно использовать инструмент `affcat`. Предыдущий пример также можно выполнить с `affcat`, перенаправив вывод в файл (без каких-либо сведений о состоянии или завершении, что полезно для сценариев):

```
$ affcat image.aff > image.raw
```

Чтобы преобразовать образ AFF в EnCase или FTK, инструмент `affcat` может передать образ через `stdout` или `stdin` соответствующему инструменту, создающему новый образ в нужном формате. Например, преобразовать AFF в сжатый образ FTK SMART можно так:

```
$ affcat image.aff | ftkimager - image --s01
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

Здесь `-` обозначает файловый дескриптор `stdin`, получающий данные необработанного образа, `image` является базовым именем файла, а заключительный флаг `--s01` задает сжатый формат.

Аналогичным образом преобразование в различные форматы EnCase можно выполнять с помощью `ewfacquirestream`. Например:

```
$ affcat image.aff | ewfacquirestream -C 42 -E 1 -e "S. Holmes" -D "Data theft
case" image
ewfacquirestream 20160424

Using the following acquiry parameters:
Image path and filename: image.E01
Case number: 42
Description: Data theft case
Evidence number: 1
Examiner name: S. Holmes
...
Acquiry completed at: May 14, 2016 15:41:42
Written: 1.8 GiB (2003934492 bytes) in 10 second(s) with 191 MiB/s (200393449
bytes/second)
MD5 hash calculated over data: 9749f1561dacd9ae85ac0e08f4e4272e
ewfacquirestream: SUCCESS
```

В предыдущих примерах преобразования AFF метаданные дела (название дела, имя эксперта, время получения, хеши и так далее) не сохраняются при преобразовании AFF в другие форматы. Но эти сведения можно экспортировать с помощью `affinfo`, а затем вручную добавить или сохранить в формате назначения. В зависимости от инструмента метаданные также можно включать флагами командной строки, как в предыдущем примере с `-C 42 -E 1 -e "S. Holmes" -D "Data theft case"`.

Этот заключительный пример демонстрирует преобразование файла AFF в сжатый контейнер криминалистических доказательств SquashFS с помощью `sfsimage`:

```
$ affcat image.aff | sfsimage -i - image.sfs
Started: 2016-05-14T15:47:19
Sfsimage version: Sfsimage Version 0.8
Sfsimage command: /usr/bin/sfsimage -i -
Current working directory: /exam
Forensic evidence source:
Destination squashfs container: image.sfs
Image filename inside container: image.raw
Aquisition command: sudo dc3dd log=errorlog.txt hlog=hashlog.txt hash=md5
2>/dev/null | pv -s 0
1.87GiB 0:00:06 [ 276MiB/s] [ <=> ]
Completed: 2016-05-14T15:47:26
```

Метаданные из файлов AFF можно извлечь с помощью `affinfo`, а затем добавить в контейнер криминалистических доказательств SquashFS следующим образом:

```
$ affinfo image.aff > affinfo.txt

$ sfsimage -a affinfo.txt image.sfs
Appending to existing 4.0 filesystem on image.sfs, block size 131072
```

После преобразования образа сравните значения хешей исходного образа и назначения, чтобы убедиться в совпадении.

Защита образа с помощью шифрования

Важная, но часто игнорируемая составляющая цифровой криминалистики - информационная безопасность. Сведения, получаемые и извлекаемые в ходе исследования, следует считать конфиденциальными и надлежащим образом защищать.

Потеря конфиденциальности данных может иметь нежелательные последствия. Например, она может нарушить требования организационных правил, поставить под угрозу соблюдение юридических и нормативных требований, вызвать проблемы с конфиденциальностью потерпевших и повредить репутации расследующей организации. Недостаточная защита полученных доказательств может причинить ущерб любой из участвующих сторон, включая следователей и их работодателя, потерпевшего, обвиняемого и другие участвующие стороны. Утечка сведений также может помешать продолжающемуся расследованию или поставить его под угрозу.

В этом разделе основное внимание уделяется методам обеспечения защиты сведений, в частности поддержанию безопасности во время передачи и хранения данных (как долгосрочного, так и краткосрочного). Добавление защиты образов увеличивает сложность и время, необходимые для шифрования и последующего расшифрования образов, но приведенные здесь примеры стремятся сделать этот процесс как можно более простым и эффективным. Вместо более сложных систем PKI или сетей доверия используется базовое симметричное шифрование.

Помимо методов, показанных в этом разделе, для шифрования можно использовать формат архива ZIP. Современные версии с расширениями ZIP64 поддерживают размеры файлов свыше 4 Гбайт. Преимущество ZIP - высокая совместимость с другими платформами, например Windows.

Шифрование GPG

С помощью симметричного шифрования можно легко шифровать образы дисков для защиты во время передачи по сети или хранения. Шифрование GNU Privacy Guard (GPG) предоставляет свободную реализацию стандарта OpenPGP, определенного RFC-4880. Это альтернатива традиционному шифрованию PGP, созданному Филом Циммерманом в начале 1990-х годов.

При использовании GPG полезно запускать агент. (При использовании `gpg2` агент запускается автоматически.) Обычно это делается при входе в систему следующей командой:

```
$ eval $(gpg-agent --daemon)
```

Во всех последующих примерах применяется флаг `-v` для повышения подробности. Благодаря этому вывод полезнее для документирования (как в этой книге, так и при создании официальных криминалистических отчетов с описанием выполненных действий).

Использовать GPG для шифрования существующего образа очень просто:

```
$ gpg -cv image.raw
gpg: using cipher AES
gpg: writing to `image.raw.gpg'

Enter passphrase:
```

Запрашивается парольная фраза, и образ шифруется стандартным алгоритмом симметричного шифрования, создавая новый файл с расширением `.gpg`. Размер образа уменьшается, поскольку GPG при шифровании выполняет сжатие. Это видно здесь:

```
$ ls -lh
total 1.2G
-r--r----- 1 holmes holmes 1.9G May 14 15:56 image.raw
-rw-r----- 1 holmes holmes 603M May 14 15:57 image.raw.gpg
```

В предыдущем примере показано шифрование файла на месте. Но шифрование можно также выполнять на лету во время получения:

```
$ sudo dcfldd if=/dev/sde | gpg -cv > image.raw.gpg
Enter passphrase:
gpg: using cipher AES
gpg: writing to stdout
241664 blocks (7552Mb) written.
241664+0 records in
241664+0 records out
```

Здесь `dcfldd` получает подключенный диск через `/dev/sde` и передает его непосредственно программе GPG. Затем зашифрованный вывод GPG перенаправляется в файл. Команда `sudo` повышает привилегии до `root`, чтобы прочитать необработанное устройство.

Расшифровать зашифрованный GPG образ так же просто, как зашифровать. Отличия состоят только в применении флага расшифрования и необходимости указать выходной файл (по умолчанию вывод отправляется в `stdout`). В следующем примере файл образа, зашифрованный GPG, расшифровывается в обычный (незащищенный) файл:

```
$ gpg -dv -o image.raw image.raw.gpg
gpg: AES encrypted data
Enter passphrase:
gpg: encrypted with 1 passphrase
gpg: original file name='image.raw'
```

Этот пример демонстрирует симметричное шифрование без подписания. Для шифрования, расшифрования и подписания образов можно также использовать открытые и закрытые ключи GPG. Целостность проверяется сравнением хеша зашифрованного GPG образа с хешем файла необработанного образа следующим образом:

```
$ gpg -dv image.raw.gpg | md5sum
gpg: AES encrypted data
Enter passphrase:
gpg: encrypted with 1 passphrase
gpg: original file name='image.raw'
5dfe68597f8ad9f20600a453101f2c57 -

$ md5sum image.raw
5dfe68597f8ad9f20600a453101f2c57 image.raw
```

При расшифровании образа необходимо выполнить планирование емкости. После расшифрования будут существовать две копии образа (одна или обе могут быть сжаты).

Зашифрованный GPG файл не поддерживает поиск, поэтому непосредственно работать с его содержимым криминалистическими аналитическими инструментами нельзя.

Шифрование OpenSSL

Другие криптографические системы также могут обеспечивать защиту образов дисков. Набор инструментов OpenSSL (http://www.openssl.org/) предоставляет ряд алгоритмов для шифрования файлов. Например, чтобы зашифровать образ паролем с помощью 256-битного AES в режиме сцепления блоков шифротекста, используйте эту команду:

```
# openssl enc -aes-256-cbc -in image.raw -out image.raw.aes
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
```

OpenSSL обладает гибкостью в отношении типов и режимов шифров, предоставляя десятки вариантов. Поддерживаются также конвейеры и перенаправление; можно легко выполнять шифрование во время получения, например:

```
# dcfldd if=/dev/sdg | openssl enc -aes-256-cbc > image.raw.aes
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
241664 blocks (7552Mb) written.
241664+0 records in
241664+0 records out
```

Расшифрование файла, зашифрованного OpenSSL, также относительно несложно при условии, что известен алгоритм шифрования:

```
# openssl enc -d -aes-256-cbc -in image.raw.aes -out image.raw
enter aes-256-cbc decryption password:
```

Добавление флага `-d` означает, что это операция расшифрования (`enc` указывает на использование симметричных шифров). Поскольку OpenSSL не предоставляет автоматического способа определить использованное симметричное шифрование, важно документировать способ шифрования файла.

Если OpenSSL не был специально скомпилирован с `zlib`, он не сжимает файлы. Чтобы добавить сжатие на лету во время получения, добавьте в конвейер `gzip`:

```
# dcfldd if=/dev/sdg | gzip | openssl enc -aes-256-cbc > image.raw.gz.aes
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
241664 blocks (7552Mb) written.
241664+0 records in
241664+0 records out
```

Чтобы проверить криптографический хеш образа, можно выполнить сходный конвейер команд:

```
$ openssl enc -d -aes-256-cbc < image.raw.gz.aes | gunzip | md5sum
enter aes-256-cbc decryption password:
4f9f576113d981ad420bbc9c251bea0c -
```

Здесь команда расшифрования принимает как входные данные сжатый и зашифрованный файл и передает расшифрованный вывод по конвейеру в `gunzip`, который выводит необработанный образ программе хеширования.

Некоторые реализации ZIP также поддерживают встроенное шифрование и могут использоваться для защиты образов и других файлов доказательств.

Встроенное шифрование криминалистических форматов

GPG и OpenSSL - хорошо известные инструменты выполнения различных задач шифрования, обеспечивающие совместимость и взаимодействие с другими инструментами. Однако они не разработаны для цифровой криминалистики, и зашифрованные файлы образов нельзя непосредственно использовать стандартными криминалистическими инструментами (их сначала необходимо расшифровать).

Некоторые версии популярных криминалистических форматов, обсуждаемых в книге, поддерживают зашифрованные образы с произвольным доступом.

Программа `ftkimager` может защищать файлы образов паролем или сертификатом. Здесь показан пример шифрования паролем (`monkey99`) во время получения:

```
# ftkimager --outpass monkey99 --e01 /dev/sdg image
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.

Creating image...
Image creation complete.
```

> **Примечание.** Включать пароль в параметры команды, как правило, не следует. Пароль виден в истории оболочки, и любой может увидеть его в таблице процессов.

Попытка обратиться к зашифрованному образу без пароля или с неправильным паролем создаст следующие сообщения об ошибках:

```
** Source is encrypted; please provide credentials for decryption.
** AD Decryption setup failed.
```

Для работы с зашифрованным образом необходимо включить пароль в командную строку следующим образом:

```
# ftkimager --inpass monkey99 image.E01 - > image.raw
AccessData FTK Imager v3.1.1 CLI (Aug 24 2012)
Copyright 2006-2012 AccessData Corp., 384 South 400 West, Lindon, UT 84042
All rights reserved.
```

Некоторые версии формата EWF поддерживают шифрование; на момент написания книги поддержка в `libewf` находилась на разных этапах разработки. Актуальное состояние поддержки зашифрованных форматов см. в последней версии исходного кода.

Комплект AFFlib позволяет непосредственно обращаться к зашифрованным образам через библиотеку Advanced Forensics Format (AFF). С самого начала AFFlib разрабатывалась с учетом информационной безопасности. Она имеет ряд возможностей шифрования для защиты криминалистических образов, включая шифрование на основе пароля (симметричное) и сертификата (X.509). Защиту можно добавить к существующему полученному образу с помощью инструмента `affcrypto`. Вот пример:

```
# affcrypto -e -N monkey99 image.aff
image.aff: 967 segments; 0 signed; 967 encrypted; 0 pages;
0 encrypted pages
```

Современные версии `dd_rescue` реализуют интерфейс подключаемых модулей и на момент написания книги имели модули для сжатия LZO, криптографического хеширования и симметричного шифрования (AES). В следующем примере показано создание образа диска (`/dev/sdc`) и сохранение вывода в зашифрованном виде с помощью модуля AES:

```
# dd_rescue -L crypt=enc:passfd=0:pbkdf2 /dev/sdc samsung.raw.aes
dd_rescue: (info): Using softbs=128.0kiB, hardbs=4.0kiB
dd_rescue: (input): crypt(0): Enter passphrase:
dd_rescue: (warning): some plugins don't handle sparse, enabled -A/--nosparse!
dd_rescue: (info): expect to copy 156290904.0kiB from /dev/sdc
dd_rescue: (info): crypt(0): Derived salt from samsung.raw.aes=00000025433d6000
dd_rescue: (info): crypt(0): Generate KEY and IV from same passwd/salt
dd_rescue: (info): ipos: 156286976.0k, opos: 156286976.0k, xferd: 156286976.0k
errs: 0, errxfer: 0.0k, succxfer: 156286976.0k
+curr.rate: 38650kB/s, avg.rate: 56830kB/s, avg.load: 14.9%

>----------------------------------------.< 99% ETA: 0:00:00
dd_rescue: (info): read /dev/sdc (156290904.0kiB): EOF
dd_rescue: (info): Summary for /dev/sdc -> samsung.raw.aes
dd_rescue: (info): ipos: 156290904.0k, opos: 156290904.0k, xferd: 156290904.0k
errs: 0, errxfer: 0.0k, succxfer: 156290904.0k
+curr.rate: 29345kB/s, avg.rate: 56775kB/s, avg.load: 14.9%
>----------------------------------------< 100% TOT: 0:45:53
```

Если эксперты криминалистической лаборатории предполагают большой объем шифрования, подписания и присвоения временных меток образам и доказательствам, стоит инвестировать в использование PKI. Это может быть внутренняя система PKI или внешний коммерческий поставщик PKI.

Шифрование диска общего назначения

Примеры предыдущих разделов были сосредоточены на защите отдельных файлов или файловых контейнеров. Альтернативный вариант - защитить весь накопитель, на котором находятся файлы образов. Это можно сделать шифрованием файловой системы, аппаратно, в пользовательском пространстве или в ядре. В этом разделе приведено несколько примеров.

На рынке существуют защищенные внешние накопители большой емкости, которые можно использовать для безопасной перевозки файлов образов, например ThinkPad Secure Hard Drive компании Lenovo, один из которых показан на рис. 7-1. Эти накопители не зависят от операционной системы и шифруют содержимое с помощью PIN-кода, вводимого на физической клавиатуре устройства.

Рис. 7-1. Защищенный жесткий диск ThinkPad

TrueCrypt когда-то был самой популярной бесплатной кроссплатформенной программой для файловых систем. Но в мае 2014 года неожиданное и необъяснимое объявление разработчиков рекомендовало искать альтернативы TrueCrypt, поскольку разработка была прекращена. В результате появилось несколько ответвлений и совместимых проектов; некоторые из них перечислены здесь:

• VeraCrypt: https://veracrypt.codeplex.com/
• `tc-play`: https://github.com/bwalex/tc-play
• CipherShed: https://ciphershed.org/
• `zuluCrypt`: http://mhogomchungu.github.io/zuluCrypt/ (не реализация TrueCrypt, но заслуживающий упоминания диспетчер TrueCrypt)

В остальных примерах этого раздела используется VeraCrypt. На момент написания книги VeraCrypt активно разрабатывался и набирал популярность как альтернатива TrueCrypt.

В следующем примере полностью шифруется пустой внешний накопитель. Затем зашифрованный контейнер можно использовать для безопасной передачи или хранения данных доказательств. Инструмент `veracrypt` задает ряд вопросов о настройке зашифрованного контейнера. Обратите внимание, что в этом примере `/dev/sda` - накопитель эксперта, а не исследуемый накопитель.

```
# veracrypt -c /dev/sda
Volume type:
1) Normal
2) Hidden
Select [1]:

Encryption Algorithm:
1) AES
2) Serpent
3) Twofish
4) AES(Twofish)
5) AES(Twofish(Serpent))
6) Serpent(AES)
7) Serpent(Twofish(AES))
8) Twofish(Serpent)
Select [1]:

Hash algorithm:
1) SHA-512
2) Whirlpool
3) SHA-256
Select [1]:

Filesystem:
1) None
2) FAT
3) Linux Ext2
4) Linux Ext3
5) Linux Ext4
6) NTFS

Select [2]: 5

Enter password:
Re-enter password:
Enter PIM:
Enter keyfile path [none]:
Please type at least 320 randomly chosen characters and then press Enter:

The VeraCrypt volume has been successfully created.
```

Теперь накопитель инициализирован как контейнер VeraCrypt (это может занять много времени в зависимости от скорости ПК и размера накопителя). Чтобы смонтировать том VeraCrypt, используется простая команда с исходным устройством и точкой монтирования:

```
# veracrypt /dev/sda /mnt
Enter password for /dev/sda:
Enter PIM for /dev/sda:
Enter keyfile [none]:
Protect hidden volume (if any)? (y=Yes/n=No) [No]:

# veracrypt -l
1: /dev/sda /dev/mapper/veracrypt1 /mnt
```

Для безопасного удаления устройства необходимо "размонтировать" том VeraCrypt; это также выполняется простой командой, указывающей точку монтирования:

```
# veracrypt --dismount /mnt
```

После этого накопитель можно физически отключить от системы. Зашифрованный накопитель в этом примере является целым необработанным устройством, но можно также использовать файл-контейнер VeraCrypt. Точка монтирования в этом примере - `/mnt`, но она может находиться в любом месте файловой системы.

Существуют и другие системы полнодискового шифрования, которые можно использовать для защиты файлов криминалистических образов и других данных. Для создания зашифрованного накопителя для хранения и перевозки можно использовать самошифрующиеся накопители (SED), подробно рассмотренные в разделе "Идентификация и разблокирование самошифрующихся накопителей Opal" на странице 128, с командой `sedutil-cli`. Шифрование файловой системы, например Linux LUKS и `dm-crypt`, предоставляет сходные уровни защиты. Хотя эти системы шифрования защитят данные доказательств на накопителе, они могут быть несовместимы с другими операционными системами (например, Windows или OS X).

Клонирование и дублирование дисков

В некоторых ситуациях клон или дубликат диска предпочтительнее файла образа. Каждый дубликат представляет собой точную посекторную копию исходного диска. Криптографическая контрольная сумма нового клонированного диска совпадает с исходной. Клон диска может быть полезен по нескольким причинам:

• Для применения аналитических инструментов и методов, которым требуется запись на диск
• Для загрузки ПК с клоном диска
• Для восстановления массивов RAID с помощью закрытых контроллеров

Клонирование дисков - несложный процесс; по сути, это получение данных в обратном направлении. Обязательно соблюдайте осторожность при дублировании, поскольку можно уничтожить данные, если по ошибке использовать неправильное устройство назначения.

Подготовка диска-клона

Размер (число секторов) диска назначения, то есть клона, должен быть равен размеру исходного диска или превышать его. Поскольку клонирование предполагает посекторное копирование, диск назначения должен иметь емкость, достаточную для всех секторов исходного диска. В некоторых случаях больший диск назначения не создает проблем, поскольку ПК и операционная система ограничатся тем, что определено в таблице разделов, и проигнорируют остальную часть диска. В других случаях важно дублировать точное число секторов диска, поскольку программы и инструменты могут ожидать определенные номера секторов. Примеры включают анализ разделов GPT (где резервная копия хранится в конце диска), систем RAID и некоторых разновидностей вредоносных программ, частично находящихся в заключительных секторах диска.

Безопасное стирание (с заполнением секторов нулями) диска назначения перед клонированием критически важно для удаления следов предыдущих данных и снижения риска загрязнения клона.

Применение HPA для воспроизведения числа секторов

HPA можно использовать для имитации того же числа секторов на клонированном диске, что и на исходном.[2] Установка HPA на клонированном диске полезна, если ожидается точное совпадение числа секторов с исходным диском. Это особенно важно при восстановлении системы RAID с закрытым контроллером или дублировании диска с данными, ожидаемыми в заключительных секторах.

> **Примечание.** До установки HPA с помощью инструмента `hdparm` необходимо знать точное число секторов исходного накопителя (оно было определено при подключении накопителя к основной системе исследования).

В этом примере HPA на диске размером 500 Гбайт настраивается для дублирования накопителя размером 120 Гбайт. Исходный диск сообщает о 234441648 секторах по 512 байт, и это значение можно использовать для установки максимального числа видимых секторов на диске-клоне. Используйте следующие команды:

```
# hdparm -N /dev/sdk

/dev/sdk:
max sectors = 976773168/976773168, HPA is disabled

# hdparm --yes-i-know-what-i-am-doing -N p234441648 /dev/sdk

/dev/sdk:
setting max visible sectors to 234441648 (permanent)
max sectors = 234441648/976773168, HPA is enabled

# hdparm -I /dev/sdk
...
LBA user addressable sectors: 234441648
...
device size with M = 1000*1000: 120034 MBytes (120 GB)
...
```

Первая команда `hdparm -N` показывает исходное состояние с 500 Гбайт доступных секторов и отключенной HPA. Вторая команда `hdparm` требует флага `--yes-i-know-what-i-am-doing` для настройки опасных параметров, например изменения числа секторов. Параметр `-N p234441648` задает число секторов. Перед ним стоит буква `p`, чтобы изменение сохранялось после перезапуска накопителя. Заключительная команда `hdparm` проверяет, сообщает ли накопитель теперь новое число секторов, которое совпадает с числом секторов клона (120 Гбайт).

Запись файла образа на диск-клон

Чтобы записать образ на новый диск, используйте те же инструменты, что и при получении диска, но в обратном направлении.

Клон диска можно создать непосредственно с исходного диска подозреваемого или из ранее полученного файла образа с помощью стандартных утилит `dd`. В этом примере показана запись файла необработанного образа на диск-клон с помощью `dc3dd`:

```
# dc3dd if=image.raw of=/dev/sdk log=clone.log
dc3dd 7.2.641 started at 2016-01-16 01:41:44 +0100
compiled options:
command line: dc3dd if=image.raw of=/dev/sdk log=clone.log
sector size: 512 bytes (assumed)
120034123776 bytes ( 112 G ) copied ( 100% ), 663 s, 173 M/s
input results for file `image.raw':
234441648 sectors in

output results for device `/dev/sdk':
234441648 sectors out
dc3dd completed at 2016-01-16 01:52:48 +0100
```

Теперь криптографический хеш можно сверить с исходным. Если число секторов исходного диска и клона не совпадает, либо возникнет ошибка (если на клоне недостаточно секторов для завершения дублирования), либо значения хешей не совпадут.

Набор разделенных образов, сжатых образов или зашифрованных образов можно записать обратно на диск-клон, не создавая сначала обычный файл образа.

Для создания дисков-клонов можно также использовать необработанные форматы, например AFF, EnCase EWF или FTK SMART. Если определенный криминалистический инструмент не может записать образ обратно на устройство, он, возможно, способен передать необработанный образ по конвейеру программе `dd`, которая это умеет.

Передача и хранение образов

Безопасное и успешное управление передачей и долгосрочным хранением криминалистических образов требует обдумывания и планирования. Часто возникают ситуации, когда образ необходимо передать другой стороне, например другому отделу крупной организации, независимой сторонней криминалистической компании или правоохранительному органу.

На способ передачи влияют несколько факторов, прежде всего объем и безопасность данных. Кроме того, в зависимости от организации может потребоваться учитывать юридические и нормативные требования, а также требования организационных правил. Например, глобальный банк может не иметь возможности передавать некоторые образы дисков через государственные границы из-за банковских норм, запрещающих вывоз данных клиентов за пределы страны.

Долгосрочное хранение образов также требует обдумывания и планирования. Если образ будет вновь открыт через несколько лет, персонал, инструменты и инфраструктура могут быть уже другими. Важно документировать сохраненные данные и поддерживать обратную совместимость с программами, применявшимися в прошлом.

Запись на съемные носители

В прошлом для передачи полученных образов накопителей использовалась стопка CD или DVD. Благодаря сжатию и разделению применение этих носителей было дешевым и осуществимым способом передачи. Сегодня распространены диски на 4 и 6 Тбайт, а диски на 10 Тбайт уже представлены на потребительском рынке. Оптические диски больше не являются практичным носителем передачи для современных больших образов даже со сжатием. Однако для полноты здесь приведено несколько примеров.

В следующем простом примере показана запись файла SquashFS на CD-ROM. Команда `mkisofs` является символической ссылкой на `genisoimage` и используется для создания файловой системы, записываемой на диск инструментом `wodim`.

```
# mkisofs -r -J maxtor-2gb-L905T60S.sfs | wodim dev=/dev/cdrom -
...
Starting to write CD/DVD at speed 48.0 in real TAO mode for single session.
...
97.45% done, estimate finish Sat Jan 16 02:36:16 2016
98.88% done, estimate finish Sat Jan 16 02:36:15 2016
...
348929 extents written (681 MB)
Track 01: Total bytes read/written: 714606592/714606592 (348929 sectors).
```

Вот простой пример записи образа на DVD. Инструмент `growisofs` начинался как интерфейс к `genisoimage`, а затем превратился в инструмент общего назначения для записи DVD и Blu-ray.

```
# growisofs -Z /dev/dvd -R -J ibm-4gb-J30J30K5215.sfs
Executing 'genisoimage -R -J ibm-4gb-J30J30K5215.sfs | builtin_dd of=/dev/dvd
obs=32k seek=0'
...
99.58% done, estimate finish Sat Jan 16 02:30:07 2016
99.98% done, estimate finish Sat Jan 16 02:30:07 2016
1240225 extents written (2422 MB)
...
```

В следующем примере показана запись образа на диск Blu-ray с помощью команды `growisofs`:

```
# growisofs -allow-limited-size -Z /dev/dvd -R -J atlas-18gb.sfs
Executing 'genisoimage -allow-limited-size -R -J atlas-18gb.sfs | builtin_dd
of=/dev/dvd obs=32k seek=0'
...
This size can only be represented in the UDF filesystem.
...
/dev/dvd: pre-formatting blank BD-R for 24.8GB...
...
99.79% done, estimate finish Sat Jan 16 02:20:10 2016
99.98% done, estimate finish Sat Jan 16 02:20:10 2016
2525420 extents written (4932 MB)
...
```

Запись больших образов на оптические диски в Linux может иметь особенности. В зависимости от используемого накопителя и носителя может наблюдаться неожиданное или непоследовательное поведение. Обязательно испытайте совместимость накопителей и носителей перед применением в рабочей среде.

Недорогие диски для хранения и передачи

Создание стопки оптических дисков из набора разделенных и сжатых образов требует систематического процесса, который может занимать много времени и быть подвержен ошибкам. Максимальная емкость дисков Blu-ray в настоящее время составляет 100 Гбайт (BD-R XL). Стоимость одного гигабайта на дисках Blu-ray выше стоимости одного гигабайта на дешевых жестких дисках.

С учетом человеческих усилий, риска ошибки, времени, необходимого для записи данных на оптические диски, и стоимости гигабайта простая покупка и применение дешевых жестких дисков становится привлекательным вариантом автономного хранения и передачи криминалистических образов.

Передача больших объемов по сети

Некоторые вопросы получения образов по сети уже рассматривались в главе 6.

Передача больших полученных образов по сети может занимать много времени и насыщать внутреннюю корпоративную сеть или интернет-канал. Во время таких длительных сетевых передач могут также происходить разрывы соединения и истечения времени ожидания.

Передача больших криминалистических образов между узлами сети далеко не так быстра, как их передача между дисками локальной машины. Чтобы представить скорости пропускной способности сети в перспективе, полезно сравнить их с распространенными скоростями дисков. В табл. 7-1 сравниваются два быстрых интерфейса накопителей и два быстрых сетевых интерфейса.

Таблица 7-1. Скорости передачи распространенных интерфейсов

Интерфейс	Скорость
NVME	4000 Мбайт/с
SATA III	600 Мбайт/с
Gigabit Ethernet	125 Мбайт/с
Fast Ethernet	12,5 Мбайт/с

Более подробное сравнение различных скоростей передачи см. на странице Wikipedia https://en.wikipedia.org/wiki/List_of_device_bit_rates.

В зависимости от пропускной способности сети и размера образа физическая доставка контейнера хранения с полученными исследуемыми образами может оказаться быстрее сетевой передачи.

Но в некоторых ситуациях безопасная передача данных по сети необходима. Обеспечение безопасности во время передачи может иметь определенные побочные эффекты, например повышение сложности или снижение производительности. Для сетевой передачи данных по недоверенным или неизвестным сетям можно применять несколько стандартных защищенных протоколов, включая SSL/TLS, `ssh`/`sftp` или IPSEC.

В следующем простом примере показана передача файла криминалистического образа с помощью `scp` (secure copy) из пакета OpenSSH:

```
$ scp image.raw server1.company.com:/data/image.raw
image.raw 11% 1955MB 37.8MB/s 06:51 ETA
...
```

Здесь файл образа (`image.raw`) копируется по незащищенной сети в определенный каталог данных на удаленном сервере. Применение `scp` имеет несколько преимуществ, включая надежные алгоритмы шифрования, встроенное сжатие, состояние хода работы в реальном времени, расчетное время завершения и надежные возможности аутентификации. Что особенно важно для экспертов-криминалистов, `scp` поддерживает очень большие размеры файлов (при условии, что двоичный файл программы скомпилирован с 64-битной поддержкой больших файлов) и легко способен передавать большие образы дисков.

Если файл образа уже зашифрован, защита нижележащего канала может быть менее важна, и можно применять традиционные протоколы передачи файлов, например File Transfer Protocol (FTP) или Windows Server Message Block (SMB). Однако при использовании незащищенных протоколов со слабой аутентификацией для передачи зашифрованных файлов следует подтвердить целостность файла, проверив криптографический хеш после завершения передачи.

Безопасное стирание и удаление данных

Всякий раз, когда диск выбрасывается или используется повторно либо временные файлы больше не нужны, предпринимайте тщательные меры для надлежащего стирания содержимого. Для этой цели доступно несколько способов очистки и безопасного удаления из командной строки.

Удаление отдельных файлов

В некоторых ситуациях потребуется безопасно стереть отдельные файлы, но не весь диск. Например, может потребоваться удалить временные полученные образы в основной системе получения данных. В таком сценарии разумно использовать уничтожитель или очиститель файлов, поскольку это снижает риск уничтожения других данных на рабочей станции эксперта.

Стандартный пакет Linux `coreutils` включает инструмент `shred`, который пытается безопасно удалить файлы, как показано здесь:

```
$ shred -v confidential-case-notes.txt
shred: confidential-case-notes.txt: pass 1/3 (random)...
shred: confidential-case-notes.txt: pass 2/3 (random)...
shred: confidential-case-notes.txt: pass 3/3 (random)...
```

Пакет программ `secure_deletion toolkit` предоставляет набор инструментов, которые пытаются стирать пространство подкачки, кеш, память, inode и файлы. В частности, `srm` стирает отдельный файл. Другой инструмент командной строки, `wipe`, также стирает файлы.

Стирание отдельных файлов - сложный процесс, зависящий от множества переменных, связанных с операционной системой и используемой файловой системой. Нет гарантий, что все фрагменты стертого или уничтоженного файла полностью уничтожены.

Безопасное стирание устройства хранения

Стирание целых физических накопителей предполагает запись нулей или случайных байтов в каждый доступный пользователю сектор накопителя. Это не гарантирует стирания всех скрытых или недоступных пользователю областей физического накопителя. Сектора, защищенные HPA или DCO (которые можно удалить), переназначенные поврежденные сектора, избыточные области флеш-накопителей и недоступные системные области накопителя не доступны пользователю и поэтому не могут быть стерты обычными инструментами Linux. Несмотря на это, стирание всех доступных пользователю секторов все же предоставляет разумный уровень уверенности, поэтому это тщательный способ удаления данных при повторном использовании накопителей внутри лаборатории.

В зависимости от отношения конкретной организации к риску и ее правил удаление данных может требовать одного или нескольких из следующих действий:

• Не выполнять стирание, только обычное переформатирование
• Стереть все видимые сектора одним проходом нулей
• Стереть все видимые сектора несколькими проходами случайных данных
• Физически размагнитить накопители
• Физически измельчить накопители

Требуемый способ удаления представляет собой основанное на риске решение, зависящее от конфиденциальности данных на накопителе, возможной заинтересованности в восстановлении данных, стоимости и усилий восстановления и других факторов.

В первом примере `dc3dd` используется для записи нулей в каждый видимый сектор диска. Инструмент `dc3dd` имеет встроенную функцию стирания, которую можно использовать следующим образом:

```
# dc3dd wipe=/dev/sdi
dc3dd 7.2.641 started at 2016-01-16 00:03:16 +0100
compiled options:
command line: dc3dd wipe=/dev/sdi
device size: 29305206 sectors (probed), 120,034,123,776 bytes
sector size: 4096 bytes (probed)
120034123776 bytes ( 112 G ) copied ( 100% ), 3619 s, 32 M/s
input results for pattern `00':
29305206 sectors in
output results for device `/dev/sdi':
29305206 sectors out
dc3dd completed at 2016-01-16 01:03:35 +0100
```

Эту задачу можно также выполнить с помощью `dd`, используя `/dev/zero` как входной файл, но `dc3dd` работает быстрее.

Чтобы подтвердить стирание диска нулями, можно использовать `dd` для чтения диска в программу шестнадцатеричного дампа:

```
# dd if=/dev/sda | hd
```

Если весь диск заполнен нулями, инструмент шестнадцатеричного дампа (hd) выведет одну строку нулей, за которой следует звездочка (*), указывающая на повторение шаблона по всему диску.

```
0000000 000000 000000 000000 000000 000000 000000 000000 000000
*
```

Если результат показывает только нули, доступные пользователю сектора накопителя успешно стерты.

В следующем примере используется инструмент nwipe, ответвление Darik's Boot and Nuke (dban). nwipe позволяет задавать различные стандарты стирания, случайность и число раундов и предоставляет файл журнала действий. Здесь показан вариант стирания канадской RCMP TSSIT OPS-II:[3]

```
# nwipe --autonuke --nogui --method=ops2 /dev/sdj
[2016/01/15 23:14:56] nwipe: notice: Opened entropy source '/dev/urandom'.
[2016/01/15 23:14:56] nwipe: info: Device '/dev/sdj' has sector size 512.
[2016/01/15 23:14:56] nwipe: warning: Changing '/dev/sdj' block size from 4096 to
512.
[2016/01/15 23:14:56] nwipe: info: Device '/dev/sdj' is size 160041885696.
[2016/01/15 23:14:56] nwipe: notice: Invoking method 'RCMP TSSIT OPS-II' on device
'/dev/sdj'.
[2016/01/15 23:14:56] nwipe: notice: Starting round 1 of 1 on device '/dev/sdj'.
[2016/01/15 23:14:56] nwipe: notice: Starting pass 1 of 8, round 1 of 1, on device
'/dev/sdj'.
[2016/01/15 23:57:00] nwipe: notice: 160041885696 bytes written to device
'/dev/sdj'.
[2016/01/15 23:57:00] nwipe: notice: Finished pass 1 of 8, round 1 of 1, on device
'/dev/sdj'.
...
```

При стирании накопителей убедитесь, что DCO и HPA удалены. Для накопителей NVME убедитесь, что стерто каждое отдельное пространство имен (у большинства потребительских накопителей NVME имеется только одно пространство имен).

Отправка команд ATA Security Erase Unit

Стандарт ATA определяет команду безопасного стирания, которую можно отправить непосредственно накопителю для стирания диска. Команда ATA SECURITY ERASE UNIT записывает нули во все доступные пользователю сектора диска. Команда EXTENDED SECURITY ERASE записывает вместо нулей заранее определенный шаблон (заданный производителем накопителя).

Запуск hdparm отображает возможности и состояние защиты накопителя. Также предоставляется расчетное время, необходимое для безопасного стирания накопителя:

```
# hdparm -I /dev/sdh
...
device size with M = 1000*1000: 500107 MBytes (500 GB)
...
Security:
Master password revision code = 7
supported
enabled
not locked
not frozen
not expired: security count
supported: enhanced erase
Security level high
60min for SECURITY ERASE UNIT. 60min for ENHANCED SECURITY ERASE UNIT.
```

Некоторые накопители отклоняют команду стирания, если сначала явно не задать пароль. В следующем примере использовался накопитель Western Digital, и перед принятием команды `--security-erase` сначала был задан пароль `dummy`:

```
# hdparm --security-erase dummy /dev/sdh
security_password="dummy"

/dev/sdh:
Issuing SECURITY_ERASE command, password="dummy", user=user
```

Теперь накопитель безопасно стерт и может быть использован повторно. Если накопитель требует задания пароля, не забудьте отключить пароль после завершения безопасного стирания.

Уничтожение ключей зашифрованного диска

Зашифрованные диски и файловые системы можно безопасно уничтожить, уничтожив все известные копии ключа шифрования. Если ключ был создан на защищенном устройстве, например смарт-карте, TPM или накопителе Opal, существует только одна копия ключа. Если накопитель или файловая система были подготовлены в корпоративной среде, могут существовать резервные или депонированные копии ключа для восстановления.

Процедуры стирания ключей для зашифрованных накопителей на основе операционной системы, например Microsoft BitLocker, Apple FileVault, Linux LUKS/`dm-crypt` или вариантов TrueCrypt, требуют подробного знания мест хранения ключей. Ключи могут быть защищены паролем или парольной фразой и храниться в файле либо определенном блоке накопителя. Они также могут находиться в файле ключа в другом месте. Если найти и безопасно уничтожить все копии закрытого ключа невозможно, альтернативой является стирание диска полным способом, описанным в предыдущем разделе.

Как правило, защищенные внешние USB-флеш-накопители имеют функцию восстановления заводских параметров при потере пароля. Ее можно использовать для уничтожения ключа и, следовательно, содержимого накопителя. Например, флеш-накопитель Corsair Padlock2 можно сбросить, удерживая одновременно кнопки `KEY` и `0/1` в течение трех секунд, а затем введя `911`, чтобы сбросить ключ и уничтожить содержимое накопителя. На накопителях iStorage datashur удерживайте одновременно кнопки `KEY` и `2` в течение трех секунд, затем введите `999` для сброса ключа.

Уничтожение содержимого накопителей Opal SED также происходит мгновенно и сводится к уничтожению ключа шифрования на накопителе путем ввода Physical Security ID (PSID). Обычно PSID имеет QR-код на физическом корпусе накопителя, который можно отсканировать вместо ручного ввода. Получить PSID, опрашивая накопитель командами ATA, нельзя; он виден только на корпусе физического накопителя.

Команда `sedutil-cli` имеет специальный параметр для безвозвратного сброса ключа накопителя с помощью PSID:

```
# time sedutil-cli --yesIreallywanttoERASEALLmydatausingthePSID
3HTEWZB0TVOLH2MZU8F7LCFD28U7GJPG /dev/sdi
- 22:21:13.738 INFO: revertTper completed successfully

real 0m0.541s
user 0m0.000s
sys 0m0.000s
```

Ключ шифрования накопителя теперь сброшен, а данные фактически уничтожены. Диск возвращен к заводским параметрам, разблокирован и может использоваться повторно. Уничтожение данных на этом накопителе размером 120 Гбайт заняло половину секунды.

Заключительные мысли

В этой главе вы изучили различные методы управления криминалистическими образами, включая применение сжатия обычными инструментами Linux и встроенного сжатия криминалистических форматов. Вы увидели дополнительные примеры сжатой файловой системы SquashFS и сценария `sfsimage` для управления контейнерами криминалистических доказательств. Я продемонстрировал разделение и сборку образов, дублирование накопителей и преобразование между форматами образов. Вы также узнали, как проверять хеши, подписи и временные метки и защищать образы шифрованием во время передачи по сети и хранения. Наконец, я показал безопасное удаление файлов криминалистических образов и накопителей.

Здесь флаг `--s01` задает создание образа SMART со сжатием `ew`, а флаг `--compress` устанавливает наивысший уровень сжатия. Дополнительные сведения о параметрах сжатия `ftkimager` можно получить с помощью флага `--help`.

Встроенное сжатие AFFlib

Хотя AFFv3 признан устаревшим (http://forensicswiki.org/wiki/AFF) и использование `aimage` не рекомендуется (http://forensicswiki.org/wiki/Aimage), применение сжатия AFFv3 в `aimage` упоминается здесь в иллюстративных целях.

Следующий пример демонстрирует создание образа диска с помощью `aimage` и указанием алгоритма сжатия LZMA (вместо стандартного `zlib`):

```
# aimage --lzma_compress --compression=9 /dev/sdj image.aff
im->outfile=image.aff
image.aff****************************IMAGING REPORT*****************************
Input: /dev/sdj
Model: Nano S/N: 07A40C03C895171A
Output file: image.aff
Bytes read: 2,003,828,736
Bytes written: 628,991,770
raw image md5: 9749 F156 1DAC D9AE 85AC 0E08 F4E4 272E
raw image sha1: 9871 0FB5 531E F390 2ED0 47A7 5BE4 747E 6BC1 BDB0
raw image sha256: 85B7 6D38 D60A 91F6 A0B6 9F65 B2C5 3BD9 F7E7 D944 639C 6F40 B3C4
0B06 83D8 A7E5
Free space remaining on capture drive: 527,524 MB
```

Криминалистическая программа Sleuth Kit предоставляет встроенную поддержку сжатых образов AFFlib. AFFv4 вводит инструмент `aff4imager` с дополнительными функциями. Его можно найти по адресу https://github.com/google/aff4/.

Сноски

[1] [назад] На момент написания книги в самой новой версии `ewfacquire` поддержка `bzip2` была временно отключена (см. раздел 20160404 в файле `ChangeLog` пакета `libewf`).
[2] [назад] Размер в секторах можно также воспроизвести с помощью DCO.
[3] [назад] Автор - канадец, отсюда предпочтение методу RCMP. :-)

Глава 8. Специальные вопросы доступа к образам

В этой главе демонстрируются методы получения сведений о файлах образов дисков и предоставления к ним доступа как к блочным устройствам и смонтированным каталогам. Вы научитесь настраивать устройства loop и создавать логические устройства инструментами отображения устройств. Вы также изучите способы отображения или преобразования образов дисков с программным шифрованием, предоставляющие доступ к ним криминалистическим инструментам. Эти методы полезны в ситуациях, когда к содержимому образа нельзя обратиться непосредственно и требуется уровень активного преобразования или расшифрования. Примеры таких образов включают зашифрованные файловые системы, образы виртуальных машин (VM) и другие форматы файлов образов, которые криминалистические инструменты не поддерживают напрямую.

Каждый раздел также содержит примеры безопасного монтирования (только для чтения) файлов образов как обычных файловых систем в основной системе криминалистического получения данных. После этого файловую систему можно легко просматривать и открывать с помощью обычных программ, например файловых диспетчеров, офисных пакетов, средств просмотра файлов, медиапроигрывателей и так далее.

Криминалистически полученные файлы образов

Основой многих методов и примеров этого раздела является устройство Linux `loop` (не путать с loopback-устройством, являющимся сетевым интерфейсом). Устройство loop - псевдоустройство, которое можно связать с обычным файлом, сделав файл доступным как блочное устройство в `/dev`.

Системы Linux обычно создают по умолчанию восемь устройств loop, чего может быть недостаточно для основной системы криминалистического получения данных; их число можно увеличить вручную или автоматически при загрузке. Чтобы создать 32 устройства loop во время загрузки, добавьте `max_loop=32` в строку `GRUB_CMDLINE_LINUX_DEFAULT=` файла `/etc/default/grub`; после перезагрузки должны быть доступны 32 неиспользуемых устройства loop. Сценарий `sfsimage` использует устройства loop для монтирования контейнеров криминалистических доказательств SquashFS.

В этой главе будут рассмотрены разные образы VM распространенных систем виртуализации QEMU, VirtualBox, VMWare и Microsoft Virtual PC. Я также описываю доступ к зашифрованным операционной системой файловым системам, включая Microsoft BitLocker, Apple FileVault, Linux LUKS и VeraCrypt (ответвление TrueCrypt). Но начнем с самого простого вида образа: необработанного образа диска, полученного инструментом в стиле `dd`.

Файлы необработанных образов с устройствами loop

Простейшую демонстрацию устройства loop можно выполнить с помощью файла необработанного образа (возможно, полученного простой командой `dd`). Команда `losetup` подключает и отключает устройства loop в системе Linux. В этом примере создается блочное устройство для файла `image.raw`:

```
# losetup --read-only --find --show image.raw
/dev/loop0
```

Здесь флаги задают, что loop должен быть доступен только для чтения (`--read-only`), следует использовать следующее доступное устройство loop (`--find`) и показать его после завершения (`--show`). После этого указанный файл (`image.raw`) становится доступен как подключенное блочное устройство.

Запуск команды `losetup` без параметров отображает состояние всех настроенных устройств loop. Здесь видно только что созданное устройство:

```
# losetup
NAME SIZELIMIT OFFSET AUTOCLEAR RO BACK-FILE
/dev/loop0 0 0 0 1 /exam/image.raw
```

Устройство `/dev/loop0` теперь указывает на `/exam/image.raw`, и к нему можно обращаться любыми инструментами, работающими с блочными устройствами. Например, здесь команда `mmls` из Sleuth Kit видит таблицу разделов файла `image.raw` через устройство loop:

```
# mmls /dev/loop0
DOS Partition Table
```

```
Offset Sector: 0
Units are in 512-byte sectors

Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0058597375 0058595328 Linux (0x83)
03: 00:01 0058597376 0078129151 0019531776 Linux Swap / Solaris x86 (0x82)
04: 00:02 0078129152 0078231551 0000102400 NTFS (0x07)
05: 00:03 0078231552 0234441647 0156210096 Mac OS X HFS (0xaf)
```

Когда устройство loop больше не нужно, просто отключите его:

```
# losetup --detach /dev/loop0
```

Устройства loop гибки и настраиваемы. В предыдущем примере `mmls` файловая система начинается с сектора 2048. Можно указывать смещение при каждом запуске криминалистического инструмента, но проще иметь отдельное устройство для каждого раздела (подобно `/dev/sda1`). Командой `losetup` можно создать отдельное устройство loop только для этого раздела, указав правильные флаги смещения (`--offset`) и размера (`--sizelimit`). Однако более общепринятый способ - использовать отображение устройств.

Это можно сделать вручную с помощью `dmsetup` и таблиц отображения, как описано в разделе "RAID и многодисковые системы" на странице 178. Однако инструмент `kpartx` автоматизирует создание устройств разделов для определенного файла образа. В следующем примере криминалистически полученный образ с четырьмя разделами используется для демонстрации создания инструментом `kpartx` устройств отображения для каждого раздела:

```
# kpartx -r -a -v image.raw
add map loop0p1 (252:0): 0 58595328 linear /dev/loop0 2048
add map loop0p2 (252:1): 0 19531776 linear /dev/loop0 58597376
add map loop0p3 (252:2): 0 102400 linear /dev/loop0 78129152
add map loop0p4 (252:3): 0 156210096 linear /dev/loop0 78231552
```

Здесь инструмент `kpartx` читает таблицу разделов диска или файла образа, создает устройство loop для всего образа, а затем устройства отображения для каждого раздела. Флаг `-r` обеспечивает доступ только для чтения к loop накопителя и отображениям разделов, а флаг `-a` предписывает `kpartx` отобразить все найденное. Применяйте флаг подробного вывода `-v`, чтобы документировать вывод команды и показывать только что выполненное отображение.

В этом примере устройство loop (`/dev/loop0`) создано для всего файла образа и доступно как необработанное блочное устройство. Кроме того, устройства разделов теперь доступны в каталоге `/dev/mapper`, и к ним можно обращаться криминалистическими инструментами, работающими с разделами, без указания смещений. Вот несколько примеров команд Sleuth Kit для некоторых разделов:

```
# fsstat /dev/mapper/loop0p1
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name:
Volume ID: d4605b95ec13fcb43646de38f7f49680
...

# fls /dev/mapper/loop0p3
r/r 4-128-1: $AttrDef
r/r 8-128-2: $BadClus
r/r 8-128-1: $BadClus:$Bad
r/r 6-128-1: $Bitmap
r/r 7-128-1: $Boot
d/d 11-144-2: $Extend
r/r 2-128-1: $LogFile
r/r 0-128-1: $MFT
...
```

```
# fsstat /dev/mapper/loop0p4
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: HFS+
File System Version: HFS+
...
```

Файловую систему, отображенную на устройство из файла образа, можно безопасно смонтировать только для чтения. Это позволит обращаться к ней с помощью стандартного файлового диспетчера, приложений и других инструментов анализа файлов. Разделы loop можно монтировать и размонтировать, как показано в этом примере:

```
# mkdir p3
# mount --read-only /dev/mapper/loop0p3 p3
# mc ./p3
...
# umount p3
# rmdir p3
```

Здесь каталог `p3`, представляющий раздел, был создан в том же каталоге, что и необработанный образ. Затем `p3` использовался как точка монтирования (выбранная точка монтирования может находиться в любом месте файловой системы основной системы эксперта). Midnight Commander (`mc`) - текстовый файловый диспетчер (клон Norton Commander), используемый в этом примере для просмотра файлов смонтированного раздела. Когда точка монтирования больше не нужна, команда `umount` (это правильное написание только с одной буквой `n`) размонтирует файловую систему, а `rmdir` удаляет каталог точки монтирования. Это традиционный способ Unix монтировать и размонтировать файловую систему в основной системе.

Когда loop накопителя и отображения разделов больше не нужны, их можно удалить все сразу с помощью флага удаления `kpartx` (`-d`) и имени файла образа:

```
# kpartx -d image.raw
loop deleted : /dev/loop0
```

Обратите внимание, что это "удаление" никак не влияет на содержимое образа диска. Удаляются loop и отображения, а не образ накопителя; образ накопителя не изменяется.

Если у необработанного образа повреждена или перезаписана таблица разделов, образ можно просканировать на наличие файловых систем и вручную отобразить файловые системы как устройства с помощью `dmsetup` (используя таблицы `dmsetup`).

Для создания, монтирования, размонтирования или отключения устройства loop требуются привилегии `root`. Они также необходимы для работы криминалистических инструментов с устройством `/dev/loop`X. Примеры в этом разделе выполнялись от пользователя `root`, чтобы уменьшить сложность командных строк и упростить их понимание. Добавление `sudo` перед командами позволяет выполнять привилегированные команды от имени обычного пользователя.

Файлы образов криминалистических форматов

Пакет программ `ewflib` включает инструмент `ewfmount` для "монтирования" содержимого криминалистического образа, делающий его доступным как обычный файл необработанного образа. В следующем примере показана группа файлов `*.e01`. С помощью `mkdir` создается точка монтирования, в этом примере `raw`, которая будет содержать файл необработанного образа:

```
# ls
image.E01 image.E02 image.E03 image.E04 image.E05

# mkdir raw
```

Инструмент `ewfmount` создает файловую систему FUSE с виртуальным необработанным образом из одного или нескольких файлов EWF. Для доступа к файлу необработанного образа команду `ewfmount` можно запустить с первым файлом EnCase EWF и точкой монтирования:

```
# ewfmount image.E01 raw
ewfmount 20160424

# ls -l raw
total 0
-r--r--r-- 1 root root 16001269760 May 17 21:20 ewf1
```

После этого с виртуальным файлом необработанного образа можно работать инструментами, непосредственно не поддерживающими форматы EWF. В следующем примере шестнадцатеричный редактор (без поддержки EWF) используется в посекторном режиме для анализа необработанного образа:

```
# hexedit -s raw/ewf1
...
```

Инструмент `kpartx` снова полезен для идентификации разделов и создания соответствующих устройств loop, позволяющих применять инструменты, работающие с блочными устройствами, и монтировать файловые системы для обычного просмотра. Здесь показан вывод `kpartx` для файлов `*.e01`, смонтированных с помощью `ewfmount`:

```
# kpartx -r -a -v raw/ewf1
add map loop0p1 (252:0): 0 29848707 linear /dev/loop0 63
add map loop0p2 (252:1): 0 2 linear /dev/loop0 29848770
add map loop0p5 : 0 1397592 linear /dev/loop0 29848833
```

Продолжим этот пример, создав точку монтирования для раздела, смонтировав файловую систему и обратившись к ней:

```
# mkdir p1
# mount --read-only /dev/mapper/loop0p1 p1
# ls p1
cdrom home/ lib32/ media/ proc/ selinux/ tmp/ vmlinuz
bin/ dev/ initrd.img lib64 mnt/ root/ srv/ usr/
boot/ etc/ lib/ lost+found/ opt/ sbin/ sys/ var/
...
```

В этом примере точка монтирования, соответствующая разделу, создается в локальном каталоге, на ней монтируется устройство раздела, а доступ к файловой системе выполняется командой `ls`. По возможности избегайте `/mnt` и других общих каталогов монтирования при монтировании файлов и контейнеров доказательств. Криминалистическую работу проще выполнять, когда точки монтирования образа находятся в том же рабочем каталоге, что и остальные связанные файлы дела.

Как и прежде, после завершения работы необходимо очистить точки монтирования и виртуальные файлы. Это снова делается в обратном порядке:

```
# umount p1
# kpartx -d raw/ewf1
loop deleted : /dev/loop0
# fusermount -u raw
# rmdir p1 raw
```

В этом примере показана команда `fusermount`, но подойдет и стандартная команда Linux `umount`. Убедитесь, что текущий рабочий каталог не находится внутри точки монтирования и никакие программы не держат открытыми файлы внутри точек монтирования. Оба условия приведут к неудаче этих шагов очистки.

При использовании контейнеров криминалистических доказательств SquashFS доступ к необработанному образу можно получить, смонтировав файл `*.sfs` командой `sfsimage -m`, создав устройства разделов и затем смонтировав нужный раздел. После этого можно выполнять обычные команды с файловой системой исследуемого образа. Здесь приведен полный пример:

```
# sfsimage -m image.sfs
image.sfs.d mount created

# kpartx -r -a -v image.sfs.d/image.raw
add map loop1p1 (252:0): 0 29848707 linear /dev/loop1 63
add map loop1p2 (252:1): 0 2 linear /dev/loop1 29848770
add map loop1p5 : 0 1397592 linear /dev/loop1 29848833

# mkdir p1
# mount /dev/mapper/loop1p1 p1
mount: /dev/mapper/loop1p1 is write-protected, mounting read-only

# ls -l
...
```

После завершения доступа к необработанному образу и его файловым системам очистка контейнеров криминалистических доказательств SquashFS также выполняется в обратном порядке. Команда `sfsimage -u` размонтирует файловую систему SquashFS, как показано здесь:

```
# umount p1
# kpartx -d image.sfs.d/image.raw
loop deleted : /dev/loop1
# sfsimage -u image.sfs.d/
image.sfs.d unmounted
```

В этом разделе продемонстрировано несколько способов доступа к содержимому криминалистических форматов как к блочным устройствам и обычным файловым системам. Инструмент `ewfmount` также работает с файлами FTK SMART. У AFFlib имеется сходный инструмент `affuse` для монтирования файлов `*.aff`. И `ewfmount`, и `affuse` могут работать с одиночными или разделенными файлами соответствующих форматов.

Обратите внимание, что многие криминалистические инструменты (например, Sleuth Kit) способны работать непосредственно с криминалистическими форматами без необработанного блочного устройства или необработанного файла.

Подготовка загрузочных образов с помощью xmount

Эксперты-криминалисты часто хотят исследовать образ накопителя обычными инструментами, например файловыми диспетчерами, офисными пакетами, приложениями или другими средствами просмотра файлов. Это можно сделать, безопасно предоставив содержимое накопителя через точку монтирования только для чтения для доступа с локальной машины эксперта.

В некоторых случаях полезно загрузить исследуемый накопитель в VM, чтобы наблюдать за работающей исследуемой средой и непосредственно взаимодействовать с ней. Это позволяет видеть рабочий стол исследуемого компьютера и использовать установленные на нем программы. Для этого можно применить ряд инструментов, описанных в этом разделе.

Инструмент `xmount` (произносится как "crossmount") создает образ виртуального диска, который можно загрузить с помощью программного обеспечения VM, например VirtualBox или kvm-qemu. `xmount` позволяет имитировать накопитель с возможностью чтения и записи, заставляя VM считать диск доступным для записи, однако продолжает защищать образ, сохраняя его в состоянии только для чтения. Доступно несколько выходных форматов VM, включая raw, DMG, VDI, VHD, VMDK и VMDKS.

Входные форматы включают полученные криминалистическим способом файлы образов, например необработанные файлы `*.raw`, файлы EnCase `*.ewf` и файлы AFFlib `*.aff`.

Ниже приведен пример того, как необработанный образ (`image.raw`) настраивается с помощью `xmount` как файл VirtualBox `*.vdi`:

```
$ mkdir virtual
$ xmount --cache xmount.cache --in raw image.raw --out vdi virtual
$ ls virtual/
image.info image.vdi
$ cat virtual/image.info
------> The following values are supplied by the used input library(ies) <------
--> image.raw <--
RAW image assembled of 1 piece(s)
30016659456 bytes in total (27.955 GiB)
------> The following values are supplied by the used morphing library <------
None
$ virtualbox
```

В этом примере создается каталог `virtual` для хранения файла виртуального образа (он будет смонтирован с помощью FUSE). На основе существующего файла `image.raw` команда `xmount` создает в каталоге `./virtual` образ VDI VirtualBox с кэшированием записи. Это лишь виртуальное представление файла образа; он не копируется и не преобразуется (поэтому дисковое пространство на машине эксперта не расходуется). Флаги `--in` и `--out` задают используемый формат образа. Входными форматами должны быть raw, AFF или EWF. Возможны разные выходные форматы.

Загрузка образа ОС в VM может оказаться сложной, когда установленная ОС ожидает конфигурацию оборудования, отличную от предоставляемой VM. Как правило, для установок Linux это создает меньше проблем, но с Windows и OS X трудности вполне возможны. Для решения этой проблемы были созданы два инструмента, `opengates` и `openjobs`, которые подготавливают образы Windows и OS X к безопасной загрузке исследуемых дисков в виртуальной среде. Я не буду рассматривать использование `opengates` и `openjobs`, но дополнительную информацию о них можно найти по адресам pinguin.lu и pinguin.lu.

Когда образ VM больше не нужен, можно выполнить очистку, размонтировав виртуальный образ и удалив каталог точки монтирования:

```
$ fusermount -u virtual
$ ls virtual/
$ rmdir virtual
```

Возможно наличие файла `xmount.cache`, содержащего данные, записанные во время использования VM. Этот файл можно сохранить, если требуется продолжить предыдущий сеанс VM, или удалить.

Образы VM

С ростом производительности домашних компьютеров, появлением аппаратной виртуализации в большинстве современных CPU и доступностью недорогого или бесплатного программного обеспечения виртуализации возрастает потребность в анализе содержимого образов VM. В некоторых случаях на исследуемых PC можно обнаружить множество образов VM. Этот раздел посвящен доступу к распространенным типам файлов образов VM, таким как QCOW2, VDI, VMDK и VHD.

QEMU QCOW2

Формат QCOW2 является распространенным типом образов VM, встречающимся в Linux и используемым эмулятором QEMU. В этом разделе я сделаю образ QCOW2 доступным как блочное устройство и безопасно смонтирую его для просмотра.

Пакет `libqcow-utils` (написанный Йоахимом Метцем, автором ewflib) содержит инструменты `qcowinfo` и `qcowmount`. Оба инструмента можно использовать так же, как в предыдущих примерах применялись `ewfinfo` и `ewfmount`. Однако следующий пример показывает альтернативный метод с использованием команды `qemu-img`, модуля ядра `nbd` и инструмента `qemu-nbd`. Этот метод дает преимущества в производительности, поскольку работает в ядре, и сокращает число шагов, так как применять `kpartx` не требуется.

Для файла `*.qcow2` команда `qemu-img` может вывести сводную информацию:

```
# qemu-img info image.qcow2
image: image.qcow2
file format: qcow2
virtual size: 5.0G (5368709120 bytes)
disk size: 141M
cluster_size: 65536
Format specific information:
compat: 1.1
lazy refcounts: false
refcount bits: 16
corrupt: false
```

Чтобы получить доступ к образу QCOW в представлении необработанного образа с помощью nbd, необходимо загрузить модуль ядра `nbd`:

```
# modprobe nbd
# dmesg | grep nbd
[16771.003241] nbd: registered device at major 43
```

В отличие от команды `losetup`, устройство не выбирается автоматически. Устройство `/dev/nbd*` необходимо указать следующим образом:

```
# qemu-nbd --read-only --connect /dev/nbd0 image.qcow2
# dmesg | grep nbd0
[16997.777839] nbd0: p1
```

Здесь файл образа QCOW2 был подключен к модулю ядра в режиме только для чтения, а устройство раздела было обнаружено автоматически. Это необработанное устройство можно использовать с криминалистическими инструментами, как показано в следующем примере:

```
# mmls /dev/nbd0
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0010485759 0010483712 Linux (0x83)
```

Устройства разделов (в этом примере - имя необработанного устройства с добавлением `p1`) также готовы к непосредственному использованию с криминалистическими инструментами. Для иллюстрации ниже команда `fls` работает непосредственно с файловой системой на устройстве раздела:

```
# fls /dev/nbd0p1
d/d 11: lost+found
r/r 12: hosts
d/d 327681: $OrphanFiles
...
```

Локально смонтировать устройства для просмотра совсем несложно. Создается локальный каталог точки монтирования, после чего файловая система монтируется обычным образом:

```
# mkdir p1
# mount /dev/nbd0p1 p1
mount: /dev/nbd0p1 is write-protected, mounting read-only
# ls p1
hosts lost+found/
```

Очистка здесь сходна с примерами использования устройств loop, но требует меньше шагов. Все процессы должны закрыть файлы, а вам следует покинуть смонтированный каталог, чтобы его можно было размонтировать. Команда отключения `qemu-nbd` с указанием имени устройства отменит регистрацию устройства в ядре:

```
# umount p1
# qemu-nbd --read-only --disconnect /dev/nbd0

/dev/nbd0 disconnected
# rmdir p1
```

Необязательный шаг - удалить модуль ядра командой `rmmod nbd`. Однако оставлять его загруженным безвредно, если планируется монтировать другие образы QCOW. Можно также автоматически загружать модуль `nbd` при запуске, добавив его в файл `/etc/modules`.

VirtualBox VDI

VirtualBox - проект с открытым исходным кодом, поддерживаемый Oracle (ранее Sun Microsystems). Хотя он поддерживает несколько форматов образов VM, в последующих примерах используются образы VirtualBox VDI. Применяется та же команда `qemu-nbd`, что и ранее, но с образом OpenSolaris.

Пакет программного обеспечения VirtualBox включает ряд утилит; ниже показано, как инструмент `VBoxManage` выводит информацию об образе VDI:

```
# VBoxManage showhdinfo OpenSolaris.vdi
UUID: 0e2e2466-afd7-49ba-8fe8-35d73d187704
Parent UUID: base
State: created
Type: normal (base)
Location: /exam/OpenSolaris.vdi
Storage format: VDI
Format variant: dynamic default
Capacity: 16384 MBytes
Size on disk: 2803 MBytes
Encryption: disabled
```

Образы VirtualBox можно монтировать с помощью `qemu-nbd` и модуля ядра `nbd` (как было показано в предыдущем разделе на примере QCOW2). Представленный здесь пример OpenSolaris несколько отличается от схемы разбиения на разделы, используемой Windows и Linux. Также показаны несколько срезов диска[1]:

```
# qemu-nbd -c /dev/nbd0 OpenSolaris.vdi
# dmesg
...
[19646.708351] nbd0: p1
p1: <solaris: [s0] p5 [s1] p6 [s2] p7 [s8] p8 >
```

В этом примере один раздел Solaris (`p1`) содержит несколько срезов (`p5`, `p6`, `p7` и `p8`).

Можно использовать те же методы, что и в предыдущем примере QEMU, чтобы получить доступ к необработанному устройству и устройствам разделов, а затем смонтировать разделы как доступные только для чтения в локальной точке монтирования. И здесь применять `kpartx` для поиска разделов не требуется, поскольку ядро делает это автоматически. Завершив доступ к разделам (или, в данном случае, срезам), выполните шаги очистки, чтобы размонтировать файловые системы и отключить устройство nbd.

VMWare VMDK

Формат Virtual Machine DisK (VMDK) используется программными продуктами VMWare для VM. В следующем примере пакет программного обеспечения `libvmdk-utils` применяется к образу Apple Lion VMDK, разделенному на несколько частей:

```
# ls
lion-000001-s001.vmdk lion-000003-s007.vmdk lion-s009.vmdk
lion-000001-s002.vmdk lion-000003-s008.vmdk lion-s010.vmdk
lion-000001-s003.vmdk lion-000003-s009.vmdk lion-s011.vmdk
lion-000001-s004.vmdk lion-000003-s010.vmdk lion-s012.vmdk
lion-000001-s005.vmdk lion-000003-s011.vmdk lion-s013.vmdk
lion-000001-s006.vmdk lion-000003-s012.vmdk lion-s014.vmdk
lion-000001-s007.vmdk lion-000003-s013.vmdk lion-s015.vmdk
lion-000001-s008.vmdk lion-000003-s014.vmdk lion-s016.vmdk
...
```

Информацию о собранном образе и каждом из "экстентов" можно получить с помощью `vmdkinfo`:

```
# vmdkinfo lion.vmdk
vmdkinfo 20160119
VMware Virtual Disk (VMDK) information:
Disk type: 2GB extent sparse
Media size: 42949672960 bytes
Content identifier: 0xadba0513
Parent content identifier: 0xffffffff
Number of extents: 21
Extent: 1
Filename: lion-s001.vmdk
Type: Sparse
Start offset: 0
Size: 2146435072 bytes
...
```

Создание точки монтирования и монтирование образа делают его доступным как файл необработанного образа:

```
# mkdir lion
# vmdkmount lion.vmdk lion
vmdkmount 20160119

# ls -ls lion
total 0
0 -r--r--r-- 1 root root 42949672960 May 17 22:24 vmdk1
# mmls lion/vmdk1
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Safety Table
01: ----- 0000000000 0000000039 0000000040 Unallocated
02: Meta 0000000001 0000000001 0000000001 GPT Header
03: Meta 0000000002 0000000033 0000000032 Partition Table
04: 00 0000000040 0000409639 0000409600 EFI System Partition
05: 01 0000409640 0082616503 0082206864 Untitled
06: 02 0082616504 0083886039 0001269536 Recovery HD
07: ----- 0083886040 0083886079 0000000040 Unallocated
```

Использование `kpartx`, как было показано ранее в этой главе, создаст соответствующие блочные устройства диска и разделов. Затем можно либо непосредственно применять к ним инструменты криминалистического анализа, либо смонтировать их на локальной машине для просмотра файловой системы.

Microsoft VHD

Существует несколько способов сделать доступным формат виртуальных образов Microsoft VHD. Например, можно применить метод с `qemu-nbd` либо воспользоваться `libvhdi-utils` с инструментами `vhdiinfo` и `vhdimount`.

Третий метод предусматривает использование `blktap-utils` с интерфейсом Xen blktap xapi. Как и в случае метода nbd, для blktap требуется вставить модуль ядра и вручную выделить устройство. Порождается процесс `tapdisk`, который подключается к драйверу и получает указание открыть образ диска. Страницы руководства для `blktap-utils` не слишком полезны, но описание можно найти на веб-сайте Xen по адресам wiki.xen.org и lists.xen.org.

В завершение этого раздела я повторю процесс настройки устройств с помощью инструментов `libvhdi`. Для простоты в предыдущих примерах использовался привилегированный пользователь root. Однако следующие примеры демонстрируют непривилегированного пользователя, которому разрешено применять `sudo`.

Чтобы непривилегированный пользователь мог выполнять команды монтирования и размонтирования FUSE, необходимо задать `user_allow_other` в `/etc/fuse.conf`.

Информацию об образе можно получить с помощью `vhdiinfo`, и специальные привилегии для этого не требуются:

```
$ vhdiinfo windows.vhd
vhdiinfo 20160111

Virtual Hard Disk (VHD) image information:
Format: 1.0
Disk type: Dynamic
Media size: 136365211648 bytes
Identifier: c9f106a3-cf3f-6b42-a13f-60e349faccb5
```

Образ можно смонтировать с помощью FUSE без привилегий root, но необходимо явно указать команде `vhdimount` разрешить доступ пользователю root, добавив флаг `-X allow_root`. Этот флаг также нужен, чтобы пользователь root мог выполнять дальнейшие действия через `sudo` (например, создавать блочные устройства с помощью `kpartx`):

```
$ mkdir raw
$ vhdimount -X allow_root windows.vhd raw
vhdimount 20160111
$ ls -l raw/
total 0
-r--r--r-- 1 holmes holmes 136365211648 Jan 20 08:14 vhdi1
```

Теперь необработанный образ доступен в каталоге `./raw`, и к нему можно обращаться стандартными инструментами. Чтобы создать устройства loop и mapper, запустите `kpartx` с командой `sudo`. После создания устройств к ним можно обращаться инструментами через команду `sudo`. `sudo` требуется для любого доступа к блочному устройству. Ниже показаны примеры с `kpartx` и `fls`:

```
$ sudo kpartx -r -a -v ./raw/vhdi1
add map loop0p1 (252:0): 0 266334018 linear /dev/loop0 63
$ sudo fls /dev/mapper/loop0p1
r/r 4-128-4: $AttrDef
r/r 8-128-2: $BadClus
r/r 8-128-1: $BadClus:$Bad
r/r 6-128-1: $Bitmap
r/r 7-128-1: $Boot
d/d 11-144-4: $Extend
r/r 2-128-1: $LogFile
r/r 0-128-1: $MFT
```

Для монтирования файловой системы также требуется `sudo`, а явное указание `-o ro` монтирует ее только для чтения. Ниже приведен пример создания точки монтирования, монтирования файловой системы из предыдущего примера и доступа к ней с помощью `ls`:

```
$ mkdir p1
$ sudo mount -o ro /dev/mapper/loop0p1 p1
$ ls p1
AUTOEXEC.BAT IO.SYS $RECYCLE.BIN/
...
```

Для очистки этого сеанса требуется `sudo`, чтобы размонтировать необработанный образ и удалить устройства loop и mapper. Монтирование файла `*.vhd` с помощью FUSE можно удалить без привилегий root. Эти шаги показаны здесь:

```
$ sudo umount p1
$ sudo kpartx -d raw/vhdi1
loop deleted : /dev/loop0
$ fusermount -u raw
```

Команда `sudo` настраивается путем редактирования файла `/etc/sudoers`. Во многих примерах этой книги для простоты используется пользователь root, чтобы сократить число команд в и без того сложной командной строке. Хорошей практикой является работа от имени непривилегированного пользователя с использованием таких механизмов безопасности, как `sudo`.

Файловые системы, зашифрованные ОС

Теперь рассмотрим доступ к популярным зашифрованным файловым системам. Основное внимание уделяется не восстановлению ключа (хотя я и предлагаю пару вариантов), а доступу к файловым системам при наличии известного ключа. Предполагается, что ключи или пароли доступны из дампов памяти, систем депонирования или резервного копирования в организациях, от лиц, обязанных предоставить их по закону, от оказывающих содействие потерпевших, из коммерческих служб или программ восстановления либо из других источников.

Тип шифрования файловой системы можно определить с помощью различных инструментов анализа разделов, способных распознавать заголовки, магические числа и другие артефакты, уникальные для конкретного типа зашифрованной файловой системы. Обзор идентификации шифрования файловой системы в криминалистическом контексте можно найти по адресу encase-forensic-blog.guidancesoftware.com.

В этом разделе приведена информация, необходимая для создания из конкретного зашифрованного образа незашифрованного блочного устройства или файла, к которому можно обращаться криминалистическими инструментами или который можно безопасно смонтировать для локального просмотра.

Microsoft BitLocker

Текущим стандартным средством шифрования файловой системы Microsoft является BitLocker. Он выполняет шифрование на уровне блоков, защищая тома целиком. Вариант BitLocker, предназначенный для съемных носителей, называется BitLocker-To-Go и использует зашифрованные файлы-контейнеры в обычной незашифрованной файловой системе. В примерах этого раздела показаны два инструмента с открытым исходным кодом: `dislocker` и `libbde`.

Пакет `dislocker`, написанный Роменом Кольтелем, можно найти по адресу github.com. Он предоставляет различные инструменты для работы с томами BitLocker, включая просмотр метаданных, создание файлов расшифрованных образов и монтирование томов с помощью FUSE.

Инструмент `dislocker-find` сканирует все подключенные устройства разделов и указанные файлы, выявляя наличие томов BitLocker. Сканирование устройств BitLocker может быть необязательным, если исследуемое устройство уже было идентифицировано в процессе его подключения к узлу получения.

Команда `dislocker-metadata` предоставляет общую информацию о накопителе BitLocker. Следующий пример представляет собой образ, полученный с USB-накопителя. Накопитель зашифрован целиком и не имеет таблицы разделов. К файлу образа можно обратиться следующим образом:

```
# dislocker-metadata -V bitlocker-image.raw
...
Wed Jan 20 13:46:06 2016 [INFO] BitLocker metadata found and parsed.
Wed Jan 20 13:46:06 2016 [INFO] =====[ Volume header informations ]=====
Wed Jan 20 13:46:06 2016 [INFO] Signature: 'MSWIN4.1'
Wed Jan 20 13:46:06 2016 [INFO] Sector size: 0x0200 (512) bytes
...
Wed Jan 20 13:46:06 2016 [INFO] Number of sectors (64 bits): 0x0000000200000000
(8589934592) bytes
Wed Jan 20 13:46:06 2016 [INFO] MFT start cluster: 0x0000000000060001 (393217)
bytes
...
Wed Jan 20 13:46:06 2016 [INFO] ====================[ BitLocker information
```

```
structure ]====================
Wed Jan 20 13:46:06 2016 [INFO] Signature: '-FVE-FS-'
Wed Jan 20 13:46:06 2016 [INFO] Total Size: 0x02f0 (752) bytes (including
signature and data)
Wed Jan 20 13:46:06 2016 [INFO] Version: 2
Wed Jan 20 13:46:06 2016 [INFO] Current state: ENCRYPTED (4)
Wed Jan 20 13:46:06 2016 [INFO] Next state: ENCRYPTED (4)
Wed Jan 20 13:46:06 2016 [INFO] Encrypted volume size: 7918845952 bytes
(0x1d8000000), ~7552 MB
...
```

Вывод этой команды предоставляет большой объем подробной криптографической информации, которая здесь не показана. В целях документирования вывод `dislocker-metadata` можно сохранить в текстовый файл. Эта команда также способна работать непосредственно с подключенными устройствами.

Как и в предыдущих примерах с паролями и шифрованием, предполагается, что ключ у вас есть. Некоторые коммерческие инструменты позволяют попытаться восстановить ключ перебором паролей. Кроме того, для извлечения FVEK из образа памяти можно использовать подключаемый модуль volatility (github.com), в том числе совместно с инструментом получения дампов памяти inception. Использование этих инструментов здесь не рассматривается.

Можно создать виртуальный файл или блочное устройство, чтобы работать с расшифрованным представлением образа диска "на месте". Этот процесс сходен с примерами из раздела "Образы VM" на странице 237. Пакет программного обеспечения `dislocker` предоставляет инструмент для создания файловой системы FUSE с виртуальным представлением расшифрованного тома:

```
# mkdir clear
# dislocker-fuse -u -V bitlocker-image.raw clear
Enter the user password:
# ls -l clear/
total 0
-rw-rw-rw- 1 root root 7918845952 Jan 1 1970 dislocker-file
...
```

Файл, появившийся в каталоге `clear`, представляет расшифрованную файловую систему, и с ним можно работать обычными криминалистическими инструментами. Ниже приведен пример использования `fsstat` из Sleuth Kit:

```
# fsstat clear/dislocker-file
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT32
OEM Name: MSDOS5.0
Volume ID: 0x5a08a5ba
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): MY SECRETS
File System Type Label: FAT32
Next Free Sector (FS Info): 34304
Free Sector Count (FS Info): 15418664
...
```

Образ расшифрованной файловой системы можно безопасно смонтировать для обычного просмотра. Команда `mount` имеет параметр loop, позволяющий напрямую смонтировать файл образа раздела:

```
# mkdir files
# mount -o loop,ro clear/dislocker-file files
# ls files
Penguins.jpg private/ System Volume Information/
...
```

Очистка в этом примере сводится к размонтированию точки монтирования файлов, удалению монтирования FUSE и удалению каталогов монтирования:

```
# umount files
# rmdir files
# fusermount -u clear
# rmdir clear
```

Обратите внимание, что предыдущие примеры выполнялись с привилегиями root, чтобы уменьшить сложность и облегчить их понимание. Те же команды можно выполнить от имени непривилегированного пользователя:

```
$ dislocker-metadata -V bitlocker-image.raw
$ mkdir clear files
$ dislocker-fuse -u -V bitlocker-image.raw -- -o allow_root clear
$ sudo mount -o loop,ro,uid=holmes clear/dislocker-file files
...
$ sudo umount files
$ fusermount -u clear
$ rmdir clear files
```

Здесь `dislocker-fuse` передает драйверу FUSE параметр `-o allow_root`, позволяя применять `sudo` для монтирования и размонтирования. Параметр `uid=holmes` обеспечивает мистеру Холмсу доступ к смонтированным файлам без привилегий root. Предполагается, что мистер Холмс состоит в Unix-группе FUSE, а файл `/etc/fuse.conf` содержит строку `user_allow_other`.

При использовании `dislocker` для разблокирования контейнера BitLocker можно предоставить учетные данные трех видов. Флаг `-u` (примененный в предыдущем примере) указывает запросить пароль пользователя. Флаг `-p` предоставляет пароль восстановления (длиной 48 цифр). Флаг `-f` задает файл ключа (файл BEK).

При использовании пароля восстановления (`-p`) вместо пароля пользователя (`-u`) необходимо вручную ввести 48-значный пароль восстановления:

```
# dislocker-fuse -p -V bitlocker-image.raw clear
Enter the recovery password: XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX-XXXXXX
Valid password format, continuing.
```

Версия этой команды для пользователя без прав root передает FUSE флаги, разрешающие монтирование с помощью `sudo`:

```
$ dislocker-fuse -p -V bitlocker-image.raw -- -o allow_root clear
```

Можно также расшифровать образ BitLocker и отдельно сохранить его как обычный образ файловой системы (сохраняется только указанный том, а не таблица разделов или другие разделы). В зависимости от размера образа BitLocker это займет некоторое время, поскольку весь образ расшифровывается и записывается в новый файл образа на диске. Потребуется спланировать необходимую емкость, так как на узле получения будут занимать место два образа: зашифрованный и расшифрованный. Расшифрованную версию тома можно создать следующим образом:

```
# dislocker-file -u -V bitlocker-image.raw bitlocker-image.clear
Enter the user password:
# ls -hs
total 15G
7.4G bitlocker-image.clear 7.4G bitlocker-image.raw
```

Получившийся файл расшифрованного образа имеет тот же размер, что и исходный, поскольку каждый блок BitLocker был расшифрован, а блок открытого текста записан в новый образ. Для этой команды привилегии root не требуются.

Теперь файл расшифрованного образа BitLocker можно смонтировать и обратиться к нему как к разделу с помощью команды `mount` с параметром loop:

```
# mkdir files
# mount -o loop,ro bitlocker-image.clear files
# ls files/
Penguins.jpg private/ System Volume Information/
```

При использовании без прав root отличается только команда `mount`:

```
$ sudo mount -o loop,ro,uid=holmes bitlocker-image.clear files
```

Поскольку BitLocker является стандартным средством шифрования файловой системы на преобладающей платформе ОС, стоит привести второй пример с другим пакетом программного обеспечения. Пакет `libbde` (написанный Йоахимом Метцем, автором ewflib) также предоставляет библиотеки и инструменты для доступа к образам BitLocker.

Следующий пример несколько сложнее предыдущего, поскольку в нем используется диск ноутбука с обычной таблицей разделов (в отличие от USB-накопителя без таблицы разделов). После вычисления смещений по выводу `mmls` демонстрируется инструмент `bdeinfo`, предоставляющий компактный обзор контейнера BitLocker.

И `dislocker`, и `libbde` можно передать байтовое смещение начала зашифрованного тома BitLocker. Однако при работе с файлами образов томов или разделов либо с устройствами без разделов это не требуется. В данном примере полученный образ содержит таблицу разделов, поэтому необходимо вычислить смещение зашифрованного тома BitLocker в байтах.

> **Примечание.** Всегда точно определяйте единицы измерения, используемые командой. Одни инструменты применяют смещения в секторах, другие - в байтах. Важно различать эти единицы и правильно преобразовывать их друг в друга.

Следующий пример показывает, как определить байтовое смещение. Команда `mmls` из Sleuth Kit выводит таблицу разделов и смещения каждого раздела в секторах. Смещение в секторах необходимо преобразовать в байтовое смещение, которое можно использовать с инструментами расшифрования:

```
# mmls image0.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0004098047 0004096000 NTFS (0x07)
03: 00:01 0004098048 0625140399 0621042352 NTFS (0x07)

04: ----- 0625140400 0625142447 0000002048 Unallocated
# echo $((4098048*512))
2098200576
```

Показанное `mmls` смещение в секторах можно преобразовать в байтовое смещение, умножив его на размер сектора. В командной строке удобно использовать арифметическое расширение Bash. В этом примере смещение в секторах равно 4098048, а размер сектора - 512. Их произведение дает байтовое смещение 2098200576. Это значение можно использовать с командой `bdeinfo` следующим образом:

```
# bdeinfo -o 2098200576 image0.raw
bdeinfo 20160119
BitLocker Drive Encryption information:
Encryption method: AES-CBC 128-bit with Diffuser
Volume identifier: 5f61cbf2-75b5-32e5-caef-537fce3cf412
Creation time: Jan 10, 2014 17:43:50.838892200 UTC
Description :Notebook System 15.01.2014
Number of key protectors: 2
Key protector 0:
Identifier: 3cd1fd6c-2ecb-2dc7-c150-839ce9e710b6
Type: TPM
Key protector 1:
Identifier: 837ef544-e1ca-65c1-a910-83acd492bc1a
Type: Recovery password
...
```

Команда `bdemount` работает сходно с командой `dislocker` и создает виртуальный файл, представляющий расшифрованный образ (здесь полный ключ сокращен):

```
# mkdir raw
# bdemount -o 2098200576 -r 630641-...-154814 image.raw raw
```

Файл появится в каталоге `./raw`, где его можно анализировать непосредственно или смонтировать на устройство loop для обычного просмотра. Команды монтирования совпадают с командами в предыдущем примере BitLocker, поэтому здесь они не повторяются.

Apple FileVault

Встроенная в OS X система шифрования файловой системы Apple называется FileVault. Она также выполняет шифрование на уровне блоков, и для ее расшифрования доступно несколько инструментов с открытым исходным кодом. Здесь я опишу два инструмента: `libfvde` и `VFDecrypt`. (Пакет программного обеспечения `libfvde` написан Омаром Чаудари и Йоахимом Метцем; его можно найти по адресу github.com.)

Перед использованием инструментов `libfvde` необходимо вычислить правильное байтовое смещение зашифрованного тома FileVault. Команда `mmls` предоставляет смещение тома в секторах, которое необходимо преобразовать в байты:

```
# mmls image.raw
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Safety Table
01: ----- 0000000000 0000000039 0000000040 Unallocated
02: Meta 0000000001 0000000001 0000000001 GPT Header
03: Meta 0000000002 0000000033 0000000032 Partition Table
04: 00 0000000040 0000409639 0000409600 EFI System Partition
05: 01 0000409640 0235708599 0235298960 HDD
06: 02 0235708600 0236978135 0001269536 Recovery HD
07: ----- 0236978136 0236978175 0000000040 Unallocated
# echo $((409640*512))
209735680
```

Умножение смещения в секторах на размер сектора с помощью простого арифметического расширения Bash дает байтовое смещение 209735680, которое можно использовать с инструментами `fvdeinfo` и `fvdemount`.

Инструмент `fvdeinfo` предоставляет общую информацию о зашифрованном томе FileVault:

```
# fvdeinfo -o 209735680 image.raw
fvdeinfo 20160108
Core Storage information:
Physical volume:
Size: 120473067520 bytes
Encryption method: AES XTS
Logical volume:
Size: 120137519104 bytes
```

Чтобы расшифровать том FileVault, необходимо восстановить файл `EncryptedRoot.plist.wipekey` и предоставить либо пароль пользователя, либо ключ восстановления. Найти и извлечь файл `wipekey` можно с помощью инструментов Sleuth Kit:

```
# fls -r -o 235708600 image.raw | grep EncryptedRoot.plist.wipekey
+++++ r/r 1036: EncryptedRoot.plist.wipekey
# icat -o 235708600 image.raw 1036 > EncryptedRoot.plist.wipekey
```

Рекурсивный вывод `fls` для раздела Recovery HD использует смещение в секторах, найденное с помощью `mmls`. В выводе посредством `grep` выполняется поиск файла `EncryptedRoot.plist.wipekey`. После обнаружения для его извлечения применяется инструмент `icat` (с использованием inode, равного в этом примере 1036). Обратите внимание: с `fls` и `icat` использовалось смещение в секторах, а не в байтах.

24-символьный ключ восстановления применяется с флагом `-r` и теперь уже восстановленным файлом `EncryptedRoot.plist.wipekey`. Затем этот ключ можно использовать для создания монтирования с помощью FUSE расшифрованного представления тома, как показано ниже (ключ восстановления сокращен):

```
# mkdir clear
# fvdemount -o 209735680 -r FKZV-...-H4PD -e EncryptedRoot.plist.wipekey image.raw
clear
fvdemount 20160108
# ls -l clear
total 0
-r--r--r-- 1 root root 120137519104 Jan 20 22:23 fvde1
...
```

Вместо ключа восстановления (`-r`) можно предоставить пароль пользователя (`-p`) и, также используя файл `EncryptedRoot.plist.wipekey`, обращаться к получившемуся образу тома обычными криминалистическими инструментами. Ниже показан пример применения `fsstat` из Sleuth Kit к расшифрованному тому:

```
# fsstat clear/fvde1
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: HFS+
File System Version: HFS+
Volume Name: HDD
...
```

Этот расшифрованный том также можно смонтировать как обычную файловую систему для просмотра:

```
# mkdir files
# mount -o loop,ro clear/fvde1 files
# ls -l files
total 8212
drwxrwxr-x 1 root 80 50 Mar 2 2015 Applications/
drwxr-xr-x 1 root root 39 Jun 2 2015 bin/
drwxrwxr-t 1 root 80 2 Aug 25 2013 cores/
dr-xr-xr-x 1 root root 2 Aug 25 2013 dev/
...
```

По завершении аналитической работы потребуется выполнить очистку:

```
# umount files
# rmdir files
# fusermount -u clear
# rmdir clear
```

Обратите внимание, что предыдущие примеры выполнялись с привилегиями root, чтобы уменьшить сложность и облегчить их понимание. Большинство команд можно выполнить без прав root, за несколькими исключениями. Ниже приведены примеры команд, которые выглядят иначе при запуске непривилегированным пользователем:

```
$ fvdemount -o 209735680 -r FKZV-...-H4PD -e EncryptedRoot.plist.wipekey image.raw
-X allow_root clear
$ sudo mount -o loop,ro clear/fvde1 files
$ sudo ls files/Users/somebody/private/directory
$ sudo umount files
```

Строка `-X allow_root` в команде `fvdemount` разрешает пользователю root обращаться к каталогу, смонтированному с помощью FUSE. Команда `sudo` необходима для монтирования и размонтирования файловой системы hfsplus. При просмотре файловой системы также может потребоваться команда `sudo`, если разрешения файловой системы ограничивают доступ к файлам или каталогам.

Существует еще несколько примечательных инструментов с открытым исходным кодом для работы с образами FileVault. Инструмент `VFDecrypt` также позволяет расшифровывать образы FileVault. Изначально его написали Ральф-Филипп Вайнманн, Дэвид Халтон и Джейкоб Аппельбаум, а сейчас его сопровождает Дрейк Аллегрини. Инструмент можно найти по адресу github.com. Он способен расшифровать образ в незашифрованный образ тома.

Программное обеспечение FileVault Cracking было создано некоторыми из тех же авторов, что и VFDecrypt; его можно найти по адресу openciphers.sourceforge.net.

Linux LUKS

В мире открытого исходного кода доступен ряд систем шифрования файлов. Некоторые, например eCryptfs или encfs, работают с каталогами. Другие, например GPG и различные инструменты crypt, работают с отдельными файлами.

В этом разделе основное внимание уделяется системе шифрования LUKS, но также будут затронуты обычный dm-crypt и loop-AES. Все три можно настроить с помощью инструмента `cryptsetup`. (Инструмент `cryptsetup` также можно использовать для управления томами TrueCrypt, которые я опишу в следующем разделе.)

Следующие примеры работают с полученным криминалистическим способом образом, содержащим зашифрованную файловую систему LUKS. Мы создадим блочное устройство, представляющее расшифрованное содержимое зашифрованной файловой системы, и покажем

методы безопасного монтирования структуры файловой системы для просмотра обычными инструментами. У нас три цели: получить информацию о шифровании, создать устройство, доступное криминалистическим инструментам, и безопасно смонтировать файловую систему для обычного просмотра.

На первом шаге требуется байтовое смещение зашифрованного раздела LUKS. Смещение в секторах показывает команда `mmls` из Sleuth Kit, примененная к файлу образа. Байтовое смещение равно смещению в секторах, умноженному на размер сектора; простое арифметическое расширение Bash дает значение 1048576:

```
# mmls luks.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0058626287 0058624240 Linux (0x83)
# echo $((2048*512))
1048576
```

Байтовое смещение можно использовать для создания устройства loop, соответствующего зашифрованному разделу, с помощью `losetup`:

```
# losetup --read-only --find --show -o 1048576 luks.raw
/dev/loop0
```

Теперь зашифрованный раздел LUKS доступен как блочное устройство, которое может использовать инструмент `cryptsetup`. Информацию о зашифрованном разделе можно получить командой `luksDump` инструмента `cryptsetup`:

```
# cryptsetup luksDump /dev/loop0
LUKS header information for /dev/loop0
Version: 1
Cipher name: aes
Cipher mode: xts-plain64
Hash spec: sha1
Payload offset: 4096
MK bits: 256
MK digest: 8b 88 36 1e d1 a4 c9 04 0d 3f fd ba 0f be d8 4c 9b 96 fb 86
MK salt: 14 0f 0d fa 7b c3 a2 41 19 d4 6a e4 8a 16 fe 72
88 78 a2 18 7b 0f 74 8e 26 6d 94 23 3d 11 2e aa
MK iterations: 172000
UUID: 10dae7db-f992-4ce4-89cb-61d126223f05
Key Slot 0: ENABLED
Iterations: 680850
Salt: 8a 39 90 e1 f9 b6 59 e1 a6 73 30 ea 73 d6 98 5a
e1 d3 b6 94 a0 73 36 f7 00 68 a2 19 3f 09 62 b8

Key material offset: 8
AF stripes: 4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

С точки зрения криминалистики слоты ключей могут представлять интерес. Том LUKS может иметь до восьми ключей, а значит, потенциально существует восемь разных паролей, которые можно попытаться восстановить.

Имея пароль к зашифрованной файловой системе LUKS, можно применить команду open инструмента cryptsetup к устройству loop0, чтобы создать устройство mapper. Это устройство предоставляет расшифрованное представление зашифрованного образа. В данном примере устройство mapper называется clear:

```
# cryptsetup -v --readonly open /dev/loop0 clear
Enter passphrase for /hyb/luks/luks.raw:
Key slot 0 unlocked.
Command successful.
```

Зашифрованное устройство loop открывается с флагом --readonly. Также указан флаг подробного вывода (-v), предоставляющий дополнительные сведения об успешности ключа расшифрования. После ввода подходящего ключа в каталоге /dev/mapper появится новое (расшифрованное) устройство раздела, с которым можно работать стандартными криминалистическими инструментами. Например, можно запустить инструмент fsstat из Sleuth Kit:

```
# fsstat /dev/mapper/clear
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext4
Volume Name: My Secrets
Volume ID: ba673056efcc5785f046654c00943860
...
```

Это устройство раздела также можно смонтировать на локальной машине для обычного просмотра:

```
# mkdir clear
# mount --read-only /dev/mapper/clear clear
# ls clear
lost+found/ the plan.txt
```

После завершения исследования можно выполнить очистку. Все шаги выполняются в обратном порядке:

```
# umount clear
# rmdir clear
# cryptsetup close clear
# losetup --detach /dev/loop0
```

Обратите внимание, что это упрощенный пример с одним разделом на одном незагрузочном диске данных. На зашифрованном диске LUKS с загрузочной ОС может присутствовать дополнительный уровень Logical Volume Manager (LVM). У таких дисков в каталоге /dev/mapper могут появиться дополнительные устройства (root, swap и так далее). К каждому из этих устройств можно обращаться или монтировать его отдельно. В процессе очистки необходимо удалить устройства разделов с помощью dmsetup, прежде чем закрывать устройство LVM с помощью cryptsetup.

Для простоты показанные в этом разделе шаги выполнялись от имени пользователя root. Чтобы выполнить примеры от имени пользователя без прав root, для запуска losetup, cryptsetup, mount и umount потребуется sudo, как и для любых инструментов, обращающихся к устройству раздела /dev/mapper. В зависимости от смонтированной файловой системы могут быть полезны дополнительные пользовательские параметры (например, uid=holmes).

Образы, зашифрованные обычным dm-crypt и loop-AES, также можно расшифровать инструментом `cryptsetup`. Процесс сходен с предыдущим примером LUKS. Для команды `cryptsetup open` необходимо указать либо `plain`, либо `loopaes` с помощью флага `--type`. Например:

```
# cryptsetup -v --readonly open --type plain /dev/loop0 clear
Enter passphrase:
Command successful.
```

При использовании `--type loopaes` также потребуется файл ключа. Можно указать и `--type luks`, но в этом нет необходимости, поскольку он используется по умолчанию.

Дополнительную информацию о cryptsetup и LUKS можно найти по адресу gitlab.com. Совместимую реализацию для Windows можно найти по адресу github.com.

TrueCrypt и VeraCrypt

После прекращения разработки TrueCrypt появилось несколько ответвлений. В настоящее время преобладающим ответвлением является VeraCrypt. Оно обеспечивает обратную совместимость, а также предлагает новые расширения.

Я приведу два примера VeraCrypt: обычный зашифрованный контейнер и скрытый контейнер. Я использовал стандартную версию VeraCrypt для командной строки совместно со знакомыми инструментами, чтобы сделать контейнеры доступными для дальнейшего анализа.

В первом примере показан простой файл зашифрованного контейнера TrueCrypt или VeraCrypt. Флаг `--file-system=none` важен, поскольку не дает VeraCrypt монтировать какие-либо файловые системы:

```
$ veracrypt --mount-options=readonly --filesystem=none secrets.tc
Enter password for /exam/secrets.tc:
Enter PIM for /exam/secrets.tc:
Enter keyfile [none]:
```

С помощью флага `-l` можно вывести список всех расшифрованных контейнеров в системе узла, упорядоченный по номерам слотов. Номер слота - важный идентификатор для последующих команд. В данном примере номер слота равен 1 и используется знакомый каталог `/dev/mapper/*`:

```
$ veracrypt -l
1: /exam/secrets.tc /dev/mapper/veracrypt1 -
```

После предоставления правильных учетных данных можно запросить дополнительную информацию о контейнере, указав номер слота:

```
$ veracrypt --volume-properties --slot=1
Slot: 1
Volume: /exam/secrets.tc
Virtual Device: /dev/mapper/veracrypt1
Mount Directory:
Size: 2.0 GB
Type: Normal
Read-Only: Yes
Hidden Volume Protected: No
Encryption Algorithm: AES
Primary Key Size: 256 bits
Secondary Key Size (XTS Mode): 256 bits
Block Size: 128 bits
Mode of Operation: XTS
PKCS-5 PRF: HMAC-SHA-512
Volume Format Version: 2
Embedded Backup Header: Yes
```

Были созданы два устройства. Устройство `/dev/loop0` содержит зашифрованный необработанный образ (такой же, как файл в файловой системе). Указанное в свойствах тома устройство `/dev/mapper/veracrypt1` представляет расшифрованный том, с которым можно непосредственно работать криминалистическими инструментами. Ниже приведен пример исследования файловой системы с помощью Sleuth Kit:

```
$ sudo fls /dev/mapper/veracrypt1
r/r * 4: photo.jpg
r/r 6: spy-photo.jpg

v/v 66969091: $MBR
v/v 66969092: $FAT1
v/v 66969093: $FAT2
d/d 66969094: $OrphanFiles
```

Устройство mapper также можно смонтировать на локальной машине и просматривать файловую систему обычными инструментами:

```
$ mkdir clear
$ sudo mount -o ro,uid=holmes /dev/mapper/veracrypt1 clear
$ ls -l clear
total 360
-rwxr-x--- 1 holmes root 366592 Jan 21 23:41 spy-photo.jpg
```

Очевидно, что удаленные файлы не будут видны в области, смонтированной пользователем; они будут доступны только при использовании криминалистических инструментов через устройство `/dev/mapper/veracrypt1`.

И снова процесс очистки выполняется в порядке, обратном процессу настройки:

```
$ sudo umount clear
$ rmdir clear
$ veracrypt -d --slot=1
```

Второй пример VeraCrypt показывает, как получить доступ к скрытому тому. Одна из особенностей TrueCrypt и VeraCrypt заключается в возможности иметь два пароля, открывающих два отдельных тома. Использование обоих паролей сопоставляется в двух приведенных ниже выводах команд.

Здесь `hidden.raw` - накопитель VeraCrypt, содержащий скрытый том. Предоставление первого пароля создает работающий стандартный контейнер TrueCrypt с файлами, заявляющий полную емкость накопителя 1 ГБ и показывающий `Type: Normal`:

```
$ ls -l
total 3098104
-rw-r----- 1 holmes holmes 1024966656 Jan 22 00:07 hidden.raw
...
$ veracrypt --mount-options=readonly --filesystem=none hidden.raw
Enter password for /exam/hidden.raw: [XXXXXXXXXXX]
...
$ veracrypt --volume-properties --slot=1
Slot: 1
Volume: /exam/hidden.raw
Virtual Device: /dev/mapper/veracrypt1
Mount Directory:

Size: 977 MB
Type: Normal
Read-Only: Yes
...
$ sudo fls /dev/mapper/veracrypt1
...
r/r 20: fake secrets.pdf
...
```

Если размонтировать том, а затем снова смонтировать его с использованием пароля скрытого тома, появится совершенно другой набор файлов. Различается и время, необходимое для монтирования тома. Разблокирование контейнера в предыдущем примере заняло 3,5 секунды, тогда как разблокирование скрытого контейнера в том же файле потребовало 29 секунд. Причина в том, что сначала предпринимается попытка расшифровать стандартный том (с использованием всех поддерживаемых алгоритмов), а после ее неудачи наконец выполняется попытка расшифровать скрытый том. Теперь в свойствах тома показан реальный размер вместе с `Type: Hidden`:

```
$ veracrypt -d --slot=1
$ veracrypt --mount-options=readonly --filesystem=none hidden.raw
Enter password for /exam/hidden.raw: [YYYYYYYYYY]
...
$ veracrypt --volume-properties --slot=1
Slot: 1
Volume: /exam/hidden.raw
Virtual Device: /dev/mapper/veracrypt1
Mount Directory:
Size: 499 MB
Type: Hidden
Read-Only: Yes
...
$ sudo fls /dev/mapper/veracrypt1
...
r/r 19: the real hidden secrets.pdf
...
```

Сопоставленное устройство скрытого тома предоставляет файловую систему, которую можно непосредственно анализировать криминалистическими инструментами.

Томами TrueCrypt и VeraCrypt можно также управлять с помощью новых версий `cryptsetup` (1.6.7 и выше), получая сходные возможности монтирования.

Существуют коммерческие инструменты и инструменты с открытым исходным кодом для взлома контейнеров TrueCrypt/VeraCrypt, но их использование выходит за рамки этой книги.

Заключительные мысли

В этой главе вы научились делать полученные файлы образов доступными как блочные устройства, создавать устройства разделов и безопасно предоставлять их для использования обычными инструментами файловой системы. Вы также научились применять устройства loop и ближе познакомились с устройствами `/dev/mapper`. Я показал приемы загрузки исследуемых образов и продемонстрировал методы доступа к образам VM разных форматов. Наконец, вы узнали, как предоставлять для доступа различные зашифрованные файловые системы в расшифрованном виде.

Сноски

[1] [назад] Термин "срезы" происходит из BSD UNIX; это распространенная схема разбиения дисков в мире UNIX.

Глава 9. Извлечение частей криминалистических образов

В этой главе рассматривается выборочное извлечение областей данных с подключенного накопителя или из полученного криминалистическим способом файла образа. Вы научитесь извлекать целые разделы, удаленные или частично перезаписанные разделы, промежутки между разделами, а также различные свободные области томов и файлов. Кроме того, вы увидите, как извлекать специальные области, например разделы Unified Extensible Firmware Interface (UEFI), секторы, скрытые посредством DCO или HPA, и разделы гибернации, такие как Intel Rapid Start Technology.

В заключительных разделах демонстрируется извлечение данных из распределенных и нераспределенных (возможно, удаленных) областей диска для дальнейшего исследования, а также ручное извлечение секторов с использованием смещений. Начнем с определения схемы разделов накопителя.

Оценка схемы разделов и файловых систем

После подключения диска к системе или получения файла образа можно проанализировать схему разделов диска. В этом разделе объясняется, как идентифицировать файловые системы, таблицы разделов и широко используемые схемы разделов диска.

Компоновка диска, или схема разделов, означает метод организации разделов (или срезов) на жестком диске. Наиболее распространенные схемы разделов, встречающиеся в потребительских компьютерах, - DOS, GPT, BSD и APM (Apple Partition Map, иногда называемая mac). Начнем с идентификации схемы разделов, используемой на диске.

Схема разделов

Каждый раздел или срез диска содержит отдельную файловую систему либо используется для какой-либо другой специальной цели. Небольшая часть диска (часто только первый сектор) определяет его компоновку, задавая начальный сектор каждого раздела, размер и тип раздела, метки и так далее.

Чтобы определить схему разделов диска, можно исследовать начальные секторы в поисках характерных признаков. Официального обозначения "Assigned Number" для схем разделов не существует (всего их около полудюжины).

Не путайте это с типами или идентификаторами разделов DOS MBR, которые перечисляют до 255 возможных файловых систем и других форматов, способных находиться внутри раздела DOS. При подключении исследуемого диска к рабочей станции ядро Linux попытается обнаружить и интерпретировать используемую схему разделов и создаст устройства для каждого найденного раздела.

Для идентификации наиболее распространенных схем разделов можно использовать команду `mmstat` из Sleuth Kit. Список поддерживаемых схем разделов показан здесь:

```
# mmstat -t list
Supported partition types:
dos (DOS Partition Table)
mac (MAC Partition Map)
bsd (BSD Disk Label)
sun (Sun Volume Table of Contents (Solaris))
gpt (GUID Partition Table (EFI))
```

Запуск `mmstat` выводит название используемой схемы:

```
# mmstat image.raw
dos
```

В качестве альтернативы для идентификации схемы разделов можно использовать инструмент `disktype`. `disktype` предоставляет более подробную информацию и поддерживает разделы, файловые системы, а также контейнеры файлов и архивов. В следующем примере показан вывод `disktype`:

```
$ sudo disktype /dev/sda
--- /dev/sda
Block device, size 27.96 GiB (30016659456 bytes)
DOS/MBR partition map

Partition 1: 27.95 GiB (30015610880 bytes, 58624240 sectors from 2048)
Type 0x83 (Linux)
```

Исходный пакет программного обеспечения `disktype` можно найти по адресу disktype.sourceforge.net. Ответвление и несколько исправлений для `disktype` доступны по адресам github.com, github.com и github.com.

Носителю данных не обязательно иметь таблицу разделов или даже файловую систему. Двоичные данные можно записать непосредственно на необработанный диск, а обращаться к ним может любая программа, способная их интерпретировать (например, некоторые базы данных могут напрямую использовать необработанные диски). Диски могут не иметь схем разделов. В таких случаях файловая система начинается с нулевого сектора и продолжается до конца диска (то есть разделом служит весь диск). Это характерно для некоторых старых USB-накопителей и гибких дисков. В таких случаях инструменты анализа разделов будут неэффективны и обычно сообщат о ложной или несуществующей таблице разделов. Если инструменту не удается обнаружить тип раздела, стоит проверить, не была ли файловая система записана непосредственно на необработанное устройство. В этом примере `mmstat` ничего не находит, но `fsstat` идентифицирует файловую систему:

```
# mmls /dev/sdj
Cannot determine partition type
# disktype /dev/sdj
--- /dev/sdj
Block device, size 1.406 MiB (1474560 bytes)
FAT12 file system (hints score 5 of 5)
Volume size 1.390 MiB (1457664 bytes, 2847 clusters of 512 bytes)
# mmstat /dev/sdj
Cannot determine partition type
# fsstat /dev/sdj
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT12
...
```

Некоторые зашифрованные тома пытаются скрыть свое существование или информацию об используемой файловой системе и не применяют распознаваемую схему разделов.

Таблицы разделов

Схема разделов содержит дисковый блок или набор блоков, описывающих ее организацию. Они называются таблицами разделов (или метками диска в системах BSD), и сведения о них можно запрашивать с помощью различных инструментов.

Команда `mmls` из Sleuth Kit позволяет вывести таблицы разделов диска или полученного криминалистическим способом образа. В этом примере `mmls` обнаруживает обычную схему разделов DOS с разделом FAT32:

```
# mmls image.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000000062 0000000063 Unallocated
02: 00:00 0000000063 0005028344 0005028282 Win95 FAT32 (0x0b)
03: ----- 0005028345 0005033951 0000005607 Unallocated
```

Традиционная схема разделов DOS не способна работать с дисками размером более 2 ТБ. Схема разделов GPT была создана, чтобы организовывать диски большего размера с увеличенным числом разделов. GPT поддерживает 128 разделов по сравнению с 4 разделами DOS (не считая расширенных разделов). Я написал статью о криминалистическом анализе дисков GPT и таблиц разделов GUID; ее можно найти по адресу dx.doi.org.

В настоящее время большинство новых систем PC поставляются с разделами GPT. Ниже показан пример таблицы разделов системы Windows 8:

```
# mmls lenovo.raw
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Safety Table
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: Meta 0000000001 0000000001 0000000001 GPT Header
03: Meta 0000000002 0000000033 0000000032 Partition Table
04: 00 0000002048 0002050047 0002048000
05: 01 0002050048 0002582527 0000532480 EFI system partition
06: 02 0002582528 0003606527 0001024000
07: 03 0003606528 0003868671 0000262144 Microsoft reserved partition
08: 04 0003868672 1902323711 1898455040 Basic data partition
09: 05 1902323712 1953523711 0051200000
```

Гэри Кесслер предоставляет несколько инструментов разбора таблиц разделов, выдающих значительно более подробную информацию. Эти инструменты можно найти по адресу garykessler.net.

Чтобы продемонстрировать уровень детализации, предоставляемый инструментами разбора Кесслера, ниже приведена часть вывода для таблицы разделов из предыдущего примера, созданного с помощью инструмента `gptparser.pl`:

```
$ gptparser.pl -i lenovo.raw
GPT Parser V1.4 beta - Gary C. Kessler (14 March 2013)
Source file = /exam/lenovo.raw
Input file length = 17408 bytes.
***** LBA 0: Protective/Legacy MBR*****
000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
016: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
...
=== Partition Table #5 (LBA 3, bytes 0:127) ===
000-015 Partition type GUID: 0xA2-A0-D0-EB-E5-B9-33-44-87-C0-68-B6-B7-26-99-C7
GUID: EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
Type: Data partition (Linux*or* Windows)
016-031 Partition GUID: 0x64-12-FF-80-A7-F7-72-42-B6-46-25-33-6D-96-13-B5
GUID: 80FF1264-F7A7-4272-B646-25336D9613B5
032-039 First LBA: 0x00-08-3B-00-00-00-00-00 [3,868,672]
040-047 Last LBA: 0xFF-27-63-71-00-00-00-00 [1,902,323,711]
048-055 Partition attributes: 0x00-00-00-00-00-00-00-00
056-127 Partition name --
056: 42 00 61 00 73 00 69 00 63 00 20 00 64 00 61 00 B.a.s.i.c. .d.a.
072: 74 00 61 00 20 00 70 00 61 00 72 00 74 00 69 00 t.a. .p.a.r.t.i.
088: 74 00 69 00 6F 00 6E 00 00 00 00 00 00 00 00 00 t.i.o.n.........
104: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
120: 00 00 00 00 00 00 00 00 ........
Name: Basic data partition
...
```

Инструмент предоставляет подробную информацию о каждом из 128 разделов GPT.

Идентификация файловой системы

Инструмент `disktype`, уже представленный в разделе "Схема разделов" на странице 260, позволяет идентифицировать схемы разделов и файловые системы внутри разделов. Инструмент `fsstat` из Sleuth Kit предоставляет более полную информацию о файловой системе. `fsstat` может работать непосредственно с устройством раздела или с полученным криминалистическим способом образом, если указать смещение в секторах.

В предыдущих примерах смещение тома Windows в секторах в файле образа `lenovo.raw` равнялось 3868672. Это смещение можно передать инструменту `fsstat` с помощью флага `-o`, чтобы проанализировать метаданные файловой системы:

```
# fsstat -o 3868672 lenovo.raw
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: 4038B39F38B39300
OEM Name: NTFS
Volume Name: Windows8_OS
Version: Windows XP
METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 786432
...
```

Если накопитель подключен непосредственно к рабочей станции, ядро Linux попытается разобрать таблицу разделов и сделать устройства диска и разделов доступными в `/dev`, где к ним можно будет обращаться напрямую.

Однако если исследуется файл необработанного образа (`.raw`, `.ewf` и так далее), файлов устройств для образа не будет. Ядро не станет интерпретировать таблицу разделов и не создаст знакомые устройства разделов (`/dev/sda1`, `/dev/sda2` и так далее). При обращении к разделу внутри файла образа необходимо указывать смещение.

Лучше полагаться на криминалистические инструменты при определении сведений о разделах, чем доверять ядру. Если диск поврежден, ядро может отказаться создавать устройства разделов или создать их неправильно. В примерах этого раздела всегда указывалось смещение, а ядро не использовалось. В ситуациях, связанных с вредоносными программами, противодействием криминалистическому анализу или иным злонамеренным введением в заблуждение, предпочтение следует отдавать криминалистическим инструментам, а не ядру.

Извлечение разделов

В этом разделе описывается извлечение отдельных разделов, промежутков между разделами и других областей диска, таких как DCO и HPA. Начнем с нескольких основных примеров извлечения обычных разделов.

Извлечение отдельных разделов

Чтобы обращаться к отдельным разделам и извлекать их вместо всего жесткого диска, можно использовать несколько методов. Я продемонстрирую несколько примеров извлечения разделов с непосредственно подключенного накопителя с устройством раздела, устройства отображения раздела и файлов образов, обрабатываемых инструментами `mmcat` и инструментами в стиле `dd` из Sleuth Kit.

Если диск доступен как подключенное устройство, получение раздела сходно с полным получением данных с помощью необработанного устройства накопителя, но вместо него используется устройство раздела. В следующем примере первый раздел `/dev/sda` извлекается в файл:

```
# dcfldd if=/dev/sda1 of=partition.raw
```

Извлечение разделов требует некоторого планирования емкости, поскольку раздел займет дисковое пространство (возможно, наряду с полным образом накопителя). Если нужен лишь временный доступ к разделу из полученного файла образа, его можно подключить как устройство loop и обращаться к нему. Этот метод демонстрируют следующие шаги.

Сначала с помощью инструмента `mmls` определите раздел, который нужно подключить как loop:

```
# mmls lenovo.raw
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors
...
05: 01 0002050048 0002582527 0000532480 EFI system partition
...
```

Затем с помощью арифметического расширения Bash преобразуйте смещение и длину в секторах в смещение и длину в байтах:

```
# echo $((2050048*512))
1049624576
# echo $((532480*512))
272629760
```

После этого вычисленные байтовые смещение и длина передаются `losetup` для создания устройства loop:

```
# losetup --read-only --find --show --offset 1049624576 --sizelimit 272629760
lenovo.raw
/dev/loop2
```

К получившемуся устройству loop можно обращаться криминалистическими инструментами так же, как к устройству раздела подключенного диска. Ниже показан пример использования `fls` из Sleuth Kit:

```
# fls /dev/loop2
r/r 3: SYSTEM_DRV (Volume Label Entry)
d/d 4: EFI
d/d 5: BOOT
d/d * 7: MSIa11f8.tmp
d/d * 8: _SI2DBB4.TMP

d/d * 9: _190875_
...
```

Если необходимо извлечь раздел из существующего полученного образа в отдельный файл, можно использовать инструменты `dd` или команду `mmcat` из Sleuth Kit.

Первый шаг при извлечении раздела из полученного образа - определить раздел и сведения о секторах. В следующем примере таблица разделов полученного образа диска показывает раздел, который требуется извлечь:

```
# mmls image.raw
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
...
02: 00:00 0000000063 0078124094 0078124032 Linux (0x83)
...
```

Для извлечения раздела из уже полученного файла образа с помощью `dcfldd` или `dd` необходимо добавить параметры skip (в `dc3dd` применяется `iskip`) и count, которые заставляют команду перейти вперед к началу раздела и получить данные только в пределах его размера:

```
$ dcfldd if=image.raw of=partition.raw bs=512 skip=63 count=78124032
```

В этой команде размер блока равен 512 байтам, что соответствует размеру сектора, начало раздела находится в секторе 63, а извлечь следует 78124032 сектора. Выполнив небольшие дополнительные вычисления, можно повысить производительность этой команды, изменив размер блока с 512 байт на более крупный (но при этом не забудьте скорректировать параметры skip и count).

В Sleuth Kit версии 3.0 и выше для простого извлечения разделов можно использовать инструмент `mmcat`. Чтобы восстановить первый раздел из предыдущего примера с помощью `mmcat`, необходимо указать номер слота `mmls` (а не номер раздела DOS). В данном случае первый раздел находится в слоте `mmls` с номером два и может быть извлечен следующим образом:

```
$ mmcat image.raw 2 > partition.raw
```

Инструмент `mmcat` просто направляет вывод в stdout, поэтому его необходимо либо перенаправить в файл, либо передать программе по каналу.

Поиск и извлечение удаленных разделов

Для исчерпывающего поиска частично перезаписанных или удаленных разделов в полученном криминалистическим способом образе можно применить несколько методов. Sleuth Kit предоставляет простой инструмент `sigfind` для поиска двоичных строк-сигнатур. Два полезных инструмента для всестороннего поиска разделов - `gpart` и `testdisk`. Эти инструменты реализуют алгоритмы распознавания файловых систем с более интеллектуальным подбором вариантов для идентификации потерянных разделов.

Запуск `gpart` без параметров начинает сканирование разделов, пропуская области, идентифицированные как распределенные. Например:

```
# gpart lenovo.raw
Begin scan...
Possible partition(Windows NT/W2K FS), size(1000mb), offset(1mb)
Possible partition(Windows NT/W2K FS), size(3mb), offset(1030mb)
Possible partition(Windows NT/W2K FS), size(3mb), offset(1494mb)
Possible partition(Windows NT/W2K FS), size(926980mb), offset(1889mb)
Possible partition(Windows NT/W2K FS), size(25000mb), offset(928869mb)
End scan.
...
Guessed primary partition table:
Primary partition(1)
type: 007(0x07)(OS/2 HPFS, NTFS, QNX or Advanced UNIX)
size: 1000mb #s(2048000) s(2048-2050047)
chs: (0/32/33)-(406/60/28)d (0/32/33)-(406/60/28)r
...
```

Добавление флага `-f` предписывает `gpart` выполнить исчерпывающий поиск разделов в каждом секторе всего диска, даже в областях, где разделы обнаружиться не должны. Это займет значительно больше времени, чем стандартное сканирование `gpart` без флагов.

Инструмент `testdisk` (cgsecurity.org, написан Кристофом Гренье, который также создал инструмент карвинга `photorec`) помимо поиска разделов предоставляет еще несколько возможностей. Testdisk предлагает интерактивный интерфейс, поддерживает разные компоновки дисков (DOS, GPT, BSD и другие), обнаруживает несколько десятков типов разделов, создает журналы действий и может

извлекать обнаруженные разделы в файл. `testdisk` можно использовать с устройствами, файлами необработанных образов и даже файлами `*.e01`.

Используйте `testdisk` с осторожностью. Этот инструмент предназначен для исправления и восстановления разделов и легко может изменить доказательства. Перед его запуском на подключенных исследуемых дисках обязательно применяйте блокиратор записи.

Инструмент также включает всеобъемлющую интерактивную систему меню для определения параметров и действий. Ниже показан пример пакетного режима, работающего с подключенным диском:

```
# testdisk /list /dev/sdb
TestDisk 7.0, Data Recovery Utility, April 2015
Christophe GRENIER <grenier@cgsecurity.org>
http://www.cgsecurity.org
Please wait...
Disk /dev/sdb - 15 GB / 14 GiB - CHS 14663 64 32
Sector size:512

Model: SanDisk Ultra Fit, FW:1.00
Disk /dev/sdb - 15 GB / 14 GiB - CHS 14663 64 32
Partition Start End Size in sectors
1 P FAT32 LBA 0 1 1 14663 44 18 30031218 [NO NAME]
FAT32, blocksize=16384
```

Для поиска удаленных разделов можно выполнить некоторый объем ручного анализа. Если таблица разделов показывает на диске большую область нераспределенного пространства, проверьте ее на наличие раздела. В следующем примере `mmls` показывает почти 2,5 ГБ (4863378 секторов) пустого пространства в конце USB-накопителя:

```
# mmls /dev/sdb
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
000: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
001: ------- 0000000000 0000002047 0000002048 Unallocated
002: 000:000 0000002048 0025167871 0025165824 Win95 FAT32 (0x0c)
003: ------- 0025167872 0030031249 0004863378 Unallocated
```

Это нераспределенное пространство может оказаться удаленным разделом. В данном примере запуск `fsstat` со смещением пустого пространства обнаруживает действующую файловую систему:

```
# fsstat -o 25167872 /dev/sdb
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: Ext3
Volume Name:
Volume ID: 74a2f1b777ae52bc9748c3dbca837a80
Last Written at: 2016-05-21 15:42:54 (CEST)
Last Checked at: 2016-05-21 15:42:54 (CEST)
...
```

Обнаружив действующую файловую систему, можно использовать метаинформацию о ней, чтобы определить вероятный размер раздела. Зная размер и начальное смещение, можно извлечь обнаруженный раздел или продолжить его анализ. Для извлечения можно использовать инструменты в стиле dd или, что проще, mmcat:

```
# mmcat /dev/sdb 3 > deleted_partition.raw
```

Здесь вывод mmcat для удаленного раздела, обнаруженного в слоте mmls 003, направляется в файл deleted_partition.raw.

Идентификация и извлечение промежутков между разделами

В некоторых случаях между разделами могут существовать промежутки, созданные случайно либо из-за того, что соседние разделы сходятся на границах цилиндров или блоков. Возможны и намеренно созданные промежутки для сокрытия данных.

Идентифицировать и восстановить такие промежутки между разделами можно тем же способом, что и извлечь раздел. С помощью mmls определите размер и смещение промежутка в секторах, а затем извлеките его посредством dd или mmcat.

Ниже показан вывод mmls для таблицы разделов. Диск содержит два раздела, между которыми имеется промежуток:

```
# mmls /dev/sdb
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
000: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
001: ------- 0000000000 0000002047 0000002048 Unallocated
002: 000:000 0000002048 0015626236 0015624189 Linux (0x83)
003: ------- 0015626237 0015626239 0000000003 Unallocated
004: 000:001 0015626240 0030031249 0014405010 Linux (0x83)
```

В этом примере первый раздел заканчивается на секторе 15626236, а соседний раздел начинается с сектора 15626240, что указывает на промежуток из трех секторов. Хотя этот промежуток между разделами можно извлечь с помощью dd, проще использовать mmcat:

```
# mmcat /dev/sdb 3 > gap.raw
# ls -l gap.raw
-rw-r----- 1 root root 1536 May 21 16:11 gap.raw
```

Получившийся файл имеет размер три сектора и содержит данные из промежутка между двумя разделами. Более крупные промежутки между разделами, содержащие частично перезаписанные, поврежденные или идентифицируемые фрагменты файловой системы, можно анализировать инструментами карвинга, например foremost.

Интерес может представлять и промежуток между последним разделом и концом диска. Он может содержать такие артефакты, как содержимое ранее перезаписанных разделов, резервные копии раздела GPT или даже сегменты двоичного кода, которые пытается скрыть вредоносная программа.

Извлечение диапазонов секторов HPA и DCO

Вы уже научились идентифицировать и снимать ограничения HPA и DCO. После их снятия эти области диска можно извлечь для отдельного анализа.

В этом примере `hdparm` показывает наличие HPA, а вывод `mmls` - три слота, один из которых является разделом Linux:

```
# hdparm -N /dev/sdh
/dev/sdh:
max sectors = 234441648/976773168, HPA is enabled
# mmls /dev/sdh
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0234441647 0234439600 Linux (0x83)
```

После успешного снятия HPA (и указания ядру повторно сканировать шину SCSI) повторный запуск тех же команд дает другой вывод:

```
# hdparm -N p976773168 /dev/sdh
/dev/sdh:
setting max visible sectors to 976773168 (permanent)
max sectors = 976773168/976773168, HPA is disabled
# mmls /dev/sdh
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors
Slot Start End Length Description
00: Meta 0000000000 0000000000 0000000001 Primary Table (#0)
01: ----- 0000000000 0000002047 0000002048 Unallocated
02: 00:00 0000002048 0234441647 0234439600 Linux (0x83)
03: ----- 0234441648 0976773167 0742331520 Unallocated
```

Теперь `hdparm` указывает, что HPA отключена, а в выводе `mmls` появилась дополнительная строка (слот 03), представляющая секторы, ранее скрытые посредством HPA.

Команда `mmcat` с номером слота раздела 03 извлечет данные из HPA:

```
# mmcat /dev/sdh 3 > hpa.raw
```

В этом примере используется действующий диск, подключенный к узлу получения. Если файл образа получен с диска после снятия HPA, `mmls` увидит эту скрытую область.

Извлечение секторов, скрытых DCO, выполняется тем же методом, что показан здесь для HPA. Сначала с помощью `hdparm` откройте доступ к секторам, защищенным DCO, а затем извлеките их посредством `dd` или `mmcat`. Нет необходимости повторять эту процедуру в дополнительных примерах, специально демонстрирующих секторы из DCO.

Прочие виды поэтапного извлечения данных

В этом заключительном разделе я описываю различные дополнительные примеры поэтапного извлечения данных. Содержимое этого раздела (фактически, содержимое большей части этой главы) слегка пересекается с криминалистическим анализом файловых систем, который не входит в заявленные рамки книги. По этой причине примеры описаны несколько менее подробно.

Извлечение резервного пространства файловой системы

Резервное пространство - традиционное понятие цифровой криминалистики, обозначающее распределенные, но неиспользуемые данные в конце секторов диска, блоков файловой системы или файловых систем (соответственно, резервное пространство RAM, файла и раздела).

Чтобы наглядно представить резервное пространство, вообразите эту книгу жестким диском, где абзацы - секторы, главы - файлы, а основной текст - раздел. Обратите внимание: абзацы не заканчиваются точно в конце строки, главы - точно в конце страницы, а в конце книги может оказаться несколько дополнительных пустых страниц. Эти пустые места и есть "резервное пространство" книги. Если в случае носителя данных ОС или физический накопитель не записали в такие области нули явно, там все еще могут находиться данные ранее записанных файлов.

Исторически извлечение и анализ резервного пространства были полезны в криминалистических исследованиях. Однако ценность резервного пространства начинает снижаться из-за нескольких факторов:

• SSD применяют команды TRIM, чтобы обнулять нераспределенные блоки.

• Современные ОС записывают нули обратно в неиспользуемые части секторов и блоков.

• Диски с собственными секторами 4 КБ выровнены по размерам блоков файловой системы.

• ОС создают разделы и файловые системы, выровненные по границам блоков.

Получение и анализ потенциальных свободных областей по-прежнему являются добросовестными этапами криминалистического процесса.

Чтобы извлечь все резервное пространство заданного образа, можно использовать команду `blkls` из Sleuth Kit. Резервное пространство зависит от файловой системы, поэтому его необходимо извлекать отдельно для каждой файловой системы (нельзя просто использовать весь необработанный диск). В этом примере смещения файловых систем полученного образа находятся с помощью `mmls`, после чего извлекается резервное пространство каждой из них:

```
# mmls lenovo.raw
GUID Partition Table (EFI)
Offset Sector: 0

Units are in 512-byte sectors
Slot Start End Length Description
04: 00 0000002048 0002050047 0002048000
05: 01 0002050048 0002582527 0000532480 EFI system partition
06: 02 0002582528 0003606527 0001024000
...
08: 04 0003868672 1902323711 1898455040 Basic data partition
...
# blkls -o 2048 -s lenovo.raw > slack.04
# blkls -o 2050048 -s lenovo.raw > slack.05
# blkls -o 2582528 -s lenovo.raw > slack.06
# blkls -o 3868672 -s lenovo.raw > slack.08
```

Резервное пространство каждой распознанной файловой системы сохраняется в файл. Флаг `-s` команды `blkls` извлекает все резервное пространство (и только его).

Важно понимать, что резервное пространство не означает нераспределенные блоки или секторы. Резервное пространство - это неиспользуемая область внутри распределенных блоков и секторов файловой системы.

Извлечение нераспределенных блоков файловой системы

В следующем примере собираются все нераспределенные блоки файловых систем полученного образа. Нераспределенные блоки зависят от файловой системы, поэтому эту операцию необходимо выполнять отдельно для каждой распознанной файловой системы.

Здесь команда `mmls` снова используется для определения смещений каждой файловой системы, а команда `blkls` - для извлечения нераспределенных блоков:

```
# blkls -o 2048 lenovo.raw > unalloc.04
# blkls -o 2050048 lenovo.raw > unalloc.05
# blkls -o 2582528 lenovo.raw > unalloc.06
# blkls -o 3868672 lenovo.raw > unalloc.08
```

Правильный флаг `blkls` для извлечения нераспределенных блоков - `-A`, но поскольку такое поведение команды используется по умолчанию, его можно не указывать.

Можно выполнить и обратное действие - извлечь все (и только) распределенные блоки с помощью команды `blkls -a`.

Ручное извлечение с использованием смещений

В некоторых ситуациях для просмотра, поиска или ручного анализа содержимого диска либо полученного образа диска можно использовать шестнадцатеричный редактор. Он может предоставлять байтовое смещение, смещение в секторах или оба значения.

В этом примере для анализа диска применяется консольный инструмент `hexedit`:

```
# hexedit -s /dev/sda
```

Инструмент `hexedit` позволяет непосредственно редактировать файлы блочных устройств и очень большие файлы образов (без загрузки в память или временных файлов), а также предоставляет секторный режим (показываются секторы целиком и смещения в секторах).

В следующем примере смещение в секторах равно 2048 (начало раздела NTFS), байтовое смещение - `0x100181`, и показан весь сектор (обратите внимание: `hexedit` предполагает размер сектора 512 байт):

```
00100000 EB 52 90 4E 54 46 53 20 20 20 20 00 02 08 00 00 .R.NTFS .....
00100010 00 00 00 00 00 F8 00 00 3F 00 FF 00 00 08 00 00 ........?.......
00100020 00 00 00 00 80 00 80 00 01 48 00 00 00 00 00 00 .........H......
00100030 04 00 00 00 00 00 00 00 80 04 00 00 00 00 00 00 ................
00100040 F6 00 00 00 01 00 00 00 22 90 FD 7E 2E 42 12 09 ........"..~.B..
00100050 00 00 00 00 FA 33 C0 8E D0 BC 00 7C FB 68 C0 07 .....3.....|.h..
00100060 1F 1E 68 66 00 CB 88 16 0E 00 66 81 3E 03 00 4E ..hf......f.>..N
00100070 54 46 53 75 15 B4 41 BB AA 55 CD 13 72 0C 81 FB TFSu..A..U..r...
00100080 55 AA 75 06 F7 C1 01 00 75 03 E9 D2 00 1E 83 EC U.u.....u.......
00100090 18 68 1A 00 B4 48 8A 16 0E 00 8B F4 16 1F CD 13 .h...H..........
001000A0 9F 83 C4 18 9E 58 1F 72 E1 3B 06 0B 00 75 DB A3 .....X.r.;...u..
001000B0 0F 00 C1 2E 0F 00 04 1E 5A 33 DB B9 00 20 2B C8 ........Z3... +.
001000C0 66 FF 06 11 00 03 16 0F 00 8E C2 FF 06 16 00 E8 f...............
001000D0 40 00 2B C8 77 EF B8 00 BB CD 1A 66 23 C0 75 2D @.+.w......f#.u-
001000E0 66 81 FB 54 43 50 41 75 24 81 F9 02 01 72 1E 16 f..TCPAu$....r..
001000F0 68 07 BB 16 68 70 0E 16 68 09 00 66 53 66 53 66 h...hp..h..fSfSf
00100100 55 16 16 16 68 B8 01 66 61 0E 07 CD 1A E9 6A 01 U...h..fa.....j.
00100110 90 90 66 60 1E 06 66 A1 11 00 66 03 06 1C 00 1E ..f`..f...f.....
00100120 66 68 00 00 00 00 66 50 06 53 68 01 00 68 10 00 fh....fP.Sh..h..
00100130 B4 42 8A 16 0E 00 16 1F 8B F4 CD 13 66 59 5B 5A .B..........fY[Z
00100140 66 59 66 59 1F 0F 82 16 00 66 FF 06 11 00 03 16 fYfY.....f......
00100150 0F 00 8E C2 FF 0E 16 00 75 BC 07 1F 66 61 C3 A0 ........u...fa..
00100160 F8 01 E8 08 00 A0 FB 01 E8 02 00 EB FE B4 01 8B ................
00100170 F0 AC 3C 00 74 09 B4 0E BB 07 00 CD 10 EB F2 C3 ..<.t...........
00100180 0D 0A 41 20 64 69 73 6B 20 72 65 61 64 20 65 72 ..A disk read er
```

```
00100190 72 6F 72 20 6F 63 63 75 72 72 65 64 00 0D 0A 42 ror occurred...B
001001A0 4F 4F 54 4D 47 52 20 69 73 20 6D 69 73 73 69 6E OOTMGR is missin
001001B0 67 00 0D 0A 42 4F 4F 54 4D 47 52 20 69 73 20 63 g...BOOTMGR is c
001001C0 6F 6D 70 72 65 73 73 65 64 00 0D 0A 50 72 65 73 ompressed...Pres
001001D0 73 20 43 74 72 6C 2B 41 6C 74 2B 44 65 6C 20 74 s Ctrl+Alt+Del t
001001E0 6F 20 72 65 73 74 61 72 74 0D 0A 00 00 00 00 00 o restart.......
001001F0 00 00 00 00 00 00 00 00 80 9D B2 CA 00 00 55 AA ..............U.
--- sda --0x100181/0x6FD21E000--sector 2048--------------------------
```

На основе байтового смещения или смещения в секторах можно составить команды `dd`, чтобы извлечь найденное в шестнадцатеричном редакторе.

В следующем примере для извлечения диапазона данных (четырех секторов по 512 байт) используются размер сектора 512, смещение в секторах и число секторов:

```
# dd if=/dev/sda of=sectors.raw skip=2048 bs=512 count=4
```

Следующий пример извлекает тот же диапазон данных с использованием байтовых смещений. Параметр skip применяет арифметическое расширение Bash, чтобы преобразовать шестнадцатеричное значение в десятичное, необходимое для `dd`; размер блока равен 1 байту, а count - требуемому числу байтов.

```
# dd if=/dev/sda of=bytes.raw skip=$((0x100000)) bs=1 count=2048
```

Два предыдущих примера извлекают один и тот же блок данных (четыре сектора, или 2048 байт). Обратите внимание: при извлечении областей диска разумно убедиться, что смещения выровнены по секторам или блокам (то есть кратны размеру сектора или блока).

Если требуется извлечь диапазон блоков файловой системы, используйте команду `blkcat` из Sleuth Kit. В следующем примере извлекаются 25 блоков файловой системы, начиная с блока 100:

```
# blkcat /dev/sda1 100 25 > blocks.raw
```

Инструмент должен определить размер блока файловой системы.

Примеры в этом заключительном разделе показали, как обращаться к образам, использовать смещения и извлекать диапазон байтов, секторов или блоков. Для сопоставления секторов с блоками, а блоков - с inode и именами файлов можно также применять другие команды Sleuth Kit. Эти задачи зависят от файловой системы и относятся уже к области ее криминалистического анализа.

Заключительные мысли

В этой последней главе вы научились извлекать части накопителей и криминалистических образов. Основное внимание уделялось извлечению различных частей образа, таких как секторы, скрытые посредством HPA или DCO, удаленные разделы и промежутки между разделами. Вы также увидели ручное извлечение заданных секторов и блоков, включая нераспределенные блоки и резервное пространство. Эта глава граничила с криминалистическим анализом, поскольку в ней рассматривались идентификация схем разделов, понимание таблиц разделов и идентификация файловых систем. Поскольку книга посвящена криминалистическому получению данных, а не криминалистическому анализу, эта глава вполне уместна в качестве последней.

Заключительные замечания

Надеюсь, эта книга оказалась для вас полезным учебным пособием и в дальнейшем останется полезным справочником. Независимо от того, являетесь ли вы профессиональным экспертом-криминалистом или студентом, изучающим криминалистику, эта книга призвана продемонстрировать фундаментальные понятия, показать, как все работает, и предоставить набор практических примеров инструментов командной строки Linux.

Многие новые книги по криминалистике посвящены анализу на уровне приложений, облачной криминалистике, криминалистике мобильных устройств, аналитике больших данных и другим новым и увлекательным областям. Традиционное криминалистическое получение цифровых данных с носителей информации и сохранение доказательств могут на их фоне казаться менее увлекательными, но это по-прежнему фундаментальная функция, которую необходимо освоить начинающим экспертам-криминалистам.

Сообществу нельзя успокаиваться, когда речь идет о достижениях традиционной криминалистики носителей информации. В этой области продолжаются значительные изменения, и мы как сообщество должны следить за новейшими разработками. Эта книга задумана как ресурс, освещающий последние изменения в традиционной криминалистике носителей информации.

Очевидно, не все показанные здесь примеры, инструменты и методы подходят для любой профессиональной криминалистической лаборатории. Многие криминалистические инструменты с открытым исходным кодом представляют собой небольшие проекты разработки программного обеспечения, которые ведут добровольцы (иногда всего один разработчик), а некоторые из них и вовсе заброшены. Им непросто конкурировать с продуктами крупных компаний, разрабатывающих коммерческое программное обеспечение. Тем не менее даже инструменты на экспериментальных стадиях разработки помогут вам понять проблемы и представить, как могут выглядеть их решения. Кроме того, я призываю вас исследовать другие инструменты и методы, которые, возможно, не рассмотрены в этой книге: инструменты с открытым исходным кодом непрерывно и быстро меняются, а для каждого показанного здесь инструмента и метода существуют альтернативы, способные дать тот же результат.

И последнее напутствие читателям: учитесь!

Я пришел в цифровую криминалистику и расследования, потому что это область, в которой вы постоянно учитесь. Процесс расследования - это обучение: вы узнаете, как развивались события инцидента. Процесс цифровой криминалистики - это обучение: вы узнаете, как технологии взаимодействуют друг с другом, и восстанавливаете последовательность технологических действий. Исследования и разработки в цифровой криминалистике - это обучение: вы учитесь создавать новые инструменты и методы для преодоления трудностей и понимать сложные технологии, чтобы расширять совокупность знаний.

Цифровая криминалистика - увлекательная область. Наслаждайтесь ею!