Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Карвинг сжатых NTFS-разделов

Курсовая работа студента 445 группы
Щитинина Дмитрия Анатольевича

Научный руководитель,
ст. преподаватель

Ю. А. Губанов

Санкт-Петербург

2010

Оглавление

Введение

Файловый карвинг – это осуществление поиска файлов или записей, основанное на их содержимом, нежели на метаинформации о расположении на носителе.

Файловый карвинг успешно применяется как для восстановления информации, так и в цифровых расследованиях (digital forensic).

Цифровым расследованием называется процесс разработки и проверки гипотез, отвечающих на вопросы о цифровых событиях. Эксперт строит гипотезы на основе найденных улик, затем проверяет их поиском дополнительных улик, которые доказывали бы состоятельность данной гипотезы. Цифровой уликой называется цифровой объект, содержащий надежную информацию, которая поддерживает или опровергает гипотезу. [Кэрриэ, 2007]

К таким уликам, как правило, относятся:

- история посещения веб-страниц
- электронные письма
- истории бесед (Skype, ICQ и пр.)

В случае восстановления информации карвинг может использоваться для восстановления файлов с поврежденных носителей (например, в случае повреждения секторов, содержащих Disk Directory или Master File Table).

Благодаря применению в этих областях, файловый карвинг становится важной задачей.

В настоящее время разработано множество подходов и алгоритмов для осуществления карвинга. Применяемый подход зависит от многих факторов: структуры искомых файлов, содержимого файлов, файловой системы носителя (если она есть).

Современные файловые системы обладают богатыми возможностями: контроль доступа, контроль целостности, поддержка файлов большого объема, сжатие данных, шифрование и многое другое.

Примером такой файловой системы является NTFS – стандарт для семейства операционных систем семейства Microsoft Windows NT (как следствие - огромное число пользователей NTFS).

Сложность файловой системы и ее дополнительные возможности оказывают огромное влияние на применяемые подходы карвинга. Такое явление, как фрагментация файлов, которая присутствует во всех файловых системах, сильно затрудняет проведение карвинга. Использование дополнительных возможностей современных файловых систем,

таких как сжатие и шифрование данных, делают многие подходы проведения карвинга совсем безрезультатными.

В данной работе предлагается способ повысить эффективность проведения карвинга NTFS разделов (уменьшить влияние фрагментации и сжатия) путем восстановления части метаинформации о расположении данных и ее использование для обеспечения работоспособности «стандартных» подходов карвинга. Также в работе описывается пример внедрения такого подхода в существующий программный продукт.

1. Общие понятия карвинга

Файловый карвинг можно разделить на два типа:

- Базовый
- Продвинутый

Базовый карвинг имеет смысл применять при следующих предположениях:

- Начало файла не повреждено
- Файл не фрагментирован
- Файл не сжат (например, сжатие на уровне файловой системы в NTFS)

К базовым типам относятся:

- *Header/Footer Carving* – выделение данных, основанное на известных маркерах начала и конца файла
- *Header/Maximum (file) size Carving* – выделение данных, основанное на известном маркере заголовка и максимальной длине файла

Продвинутый карвинг применяется в случае фрагментированных файлов:

- Фрагменты могут идти не последовательно
- Порядок фрагментов может быть нарушен
- Некоторые фрагменты могут отсутствовать

К продвинутым типам относятся:

- *File Structure Based Carving* – основан на знании внутренней структуры формата разыскиваемых данных
- *Semantic Carving* – основан на анализе лингвистического либо смыслового содержания данных
- *Carving with Validation* – валидация искомого файла на основании его формата (хороший пример такого подхода описан в [Pereira, 2009]).
- *Fragment Recovery Carving* – данные собираются из разных фрагментов

Способ карвинга также сильно зависит от особенностей файловой системы исследуемого носителя.

Приложения, выполняющие карвинг, принято называть *карверами*.

2. Файловая система NTFS

NTFS (New Technology File System) – стандартная файловая система для семейства операционных систем Microsoft Windows NT (включая Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Windows Vista, Windows 7).

NTFS достаточно сложна, тем более что не существует официальной спецификации с описанием структуры NTFS раздела. Основным источником, который проливает свет на внутреннее устройство этой файловой системы, являются результаты исследований группы Linux-NTFS, описанные в [Richard Russon, Yuval Fledel], а также исходные коды драйвера NTFS для Linux. Эти результаты достигнуты путем реверс-инжиниринга файловой системы.

2.1 Возможности NTFS

NTFS обладает встроенными возможностями, такими как:

- Разграничение доступа к данным для различных пользователей и групп пользователей.
- Назначение квот (ограничений на максимальный объём дискового пространства, занимаемый теми или иными пользователями).
- Использование системы журналирования для повышения надёжности файловой системы.
- Сжатие данных для уменьшения используемого дискового пространства.
- Шифрование данных.

NTFS выделяется среди других файловых систем тем, что все служебные данные хранятся в файлах, причем эти файлы могут находиться в любом месте раздела. Таким образом, в отличие от остальных файловых систем, NTFS не обладает жестко заданной структурой (нельзя заранее сказать, в каких частях раздела находятся служебные данные).

2.2 Концепции NTFS

2.2.1 MFT

Основой NTFS является главная файловая таблица MFT (Master File Table), которая содержит информацию обо всех файлах и каталогах.

Каждому файлу или каталогу соответствует (как минимум) одна запись в этой таблице. Сами записи имеют достаточно простую структуру: сначала идет общий заголовок, затем атрибуты (небольшие структуры данных строго определенного назначения).

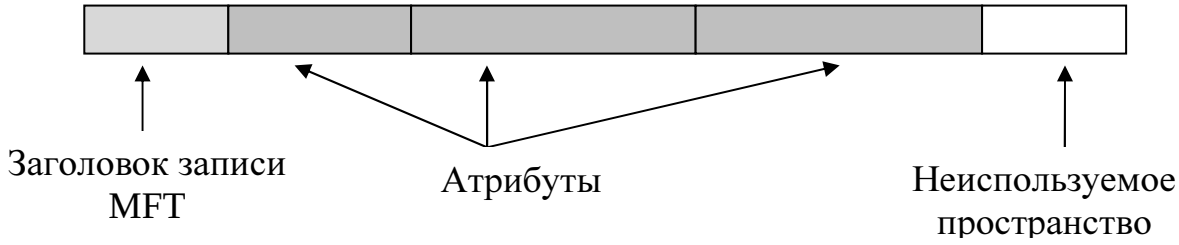

Рисунок 1: Структура MFT записи

Как правило, размер MFT-записи – 1024 Кб (хотя это число может варьироваться).

MFT также содержит запись для представления себя самой. Для этого используется первая запись в таблице и она описывает расположение MFT на диске. Первая MFT-запись - единственное место, в котором указывается местоположение MFT. При повреждении этой записи файловая система перестает работать.

Записи MFT не имеют фиксированной структуры и содержат атрибуты, в которых хранится конкретная информация.

В первом поле каждой записи MFT хранится сигнатура – ASCII строка «FILE».

Если атрибуты не помещаются в одной записи (такое происходит, если файл очень большой или сильно фрагментирован), то для файла создаются несколько записей MFT.

2.2.2. Атрибуты записей MFT

Большая часть записи MFT используется для хранения атрибутов – записей, содержащих данные определенного типа. Существует множество типов атрибутов, каждый из которых обладает своей внутренней структурой. Существуют атрибуты для имени файла, для временных данных, связанных с файлом, для содержимого файла и другие.

Атрибуты всех типов содержат две общие части: заголовок и содержимое. Заголовок стандартен для всех атрибутов, содержимое зависит от типа атрибута и не имеет фиксированного размера.

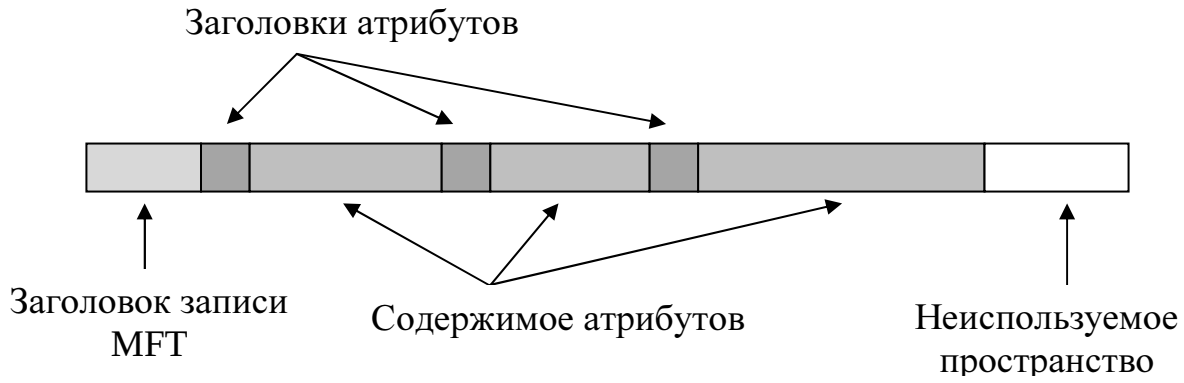

Рисунок 2: Атрибуты MFT записи

Заголовок атрибута определяет тип атрибута, его размер и имя, а также содержит флаги, указывающие на сжатие или шифрование содержимого.

Содержимое атрибутов имеет произвольный формат и произвольный размер. Например, один из атрибутов используется для хранения содержимого файла (которое может составлять гигабайты). Так как такое количество данных неприемлемо хранить в MFT записях размером 1024 байт, для решения этой проблемы предусмотрено два способа хранения содержимого атрибутов:

- Содержимое *резидентных* атрибутов хранится в самой MFT записи. Содержимое атрибута следует сразу за заголовком. Такой способ подходит только для небольших атрибутов.
- Содержимое *нерезидентных* атрибутов хранится во внешних кластерах файловой системы (т.е. в кластерах, не выделенных под MFT). При этом заголовок атрибута содержит адреса внешних кластеров, выделенных под содержимое атрибута. Является ли атрибут резидентным или нет, указывается в его заголовке.

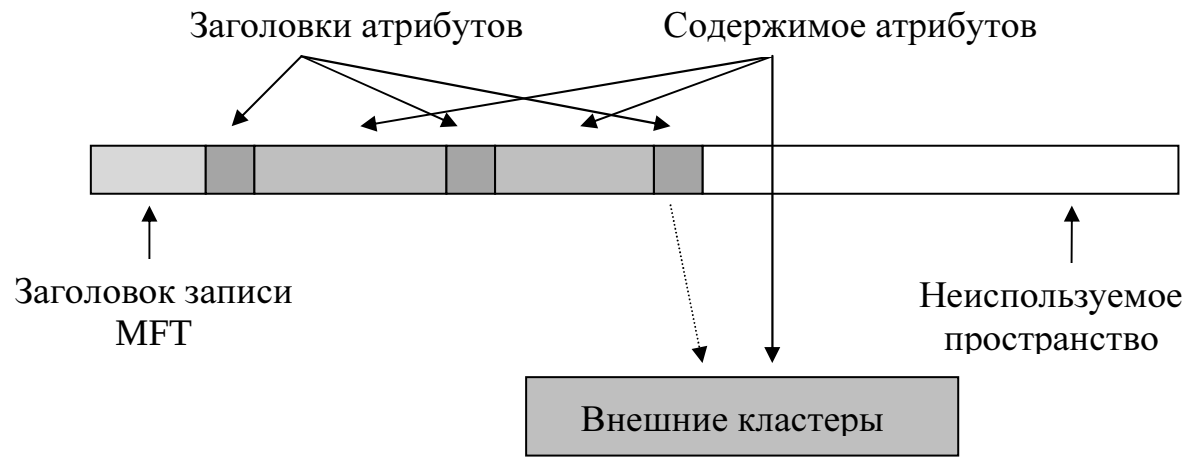

Рисунок 3: Запись MFT с нерезидентным атрибутом

Нерезидентные атрибуты хранятся в группах смежных кластеров (так называемых *сериях*). Серия определяется адресом начального кластера и длиной (количеством кластеров). В случае нерезидентного атрибута заголовок хранит список серий.

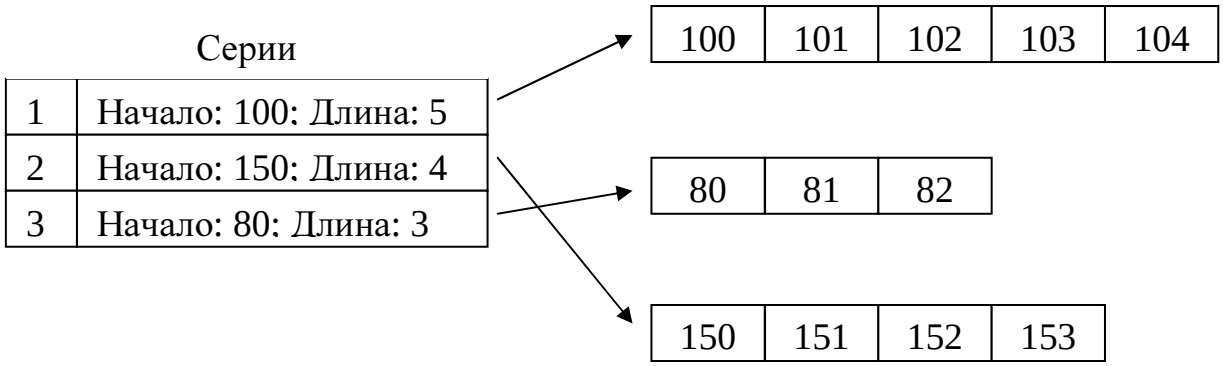

Рисунок 4: Список серий

2.2.3 Стандартные типы атрибутов

Каждый тип атрибута представлен некоторым идентификатором.

Таблица 1: Типы атрибутов MFT записей

ID типа	Имя	Описание
0x10	$STANDARD_INFORMATION	Общая информация (флаги, время создания, доступа, обновления, владелец)
0x20	$ATTRIBUTE_LIST	Список атрибутов (используется, когда атрибуты не помещаются в одной MFT записи)
0x30	$FILE_NAME	Имя файла (Unicode), время создания и последнего обращения
0x40	$VOLUME_VERSION	Версия тома

0x50	$SECURITY_DESCRIPTOR	Свойства безопасности файла
0x60	$VOLUME_NAME	Имя раздела
0x70	$VOLUME_INFORMATION	Версия файловой системы и флаги
0x80	$DATA	Содержимое файла
0x90	$INDEX_ROOT	Корневой узел индексного дерева
0xA0	$INDEX_ALLOCATION	Узлы индексного дерева
0xB0	$BITMAP	Битовая карта файла $MFT
0xC0	$SYMBOLIC_LINK	Информация о символьных ссылках (Windows NT)
0xC0	$REPARSE_POINT	Данные точек подключения (роль символьных ссылок в Windows 2000)
0xD0	$EA_INFORMATION	Для совместимости с приложениями OS/2 (HPFS)
0xE0	$EA	Для совместимости с приложениями OS/2 (HPFS)
0xF0	$LOGGED_UTILITY_STREAM	Ключи и информация о зашифрованных атрибутах

MFT запись не обязана обладать всеми перечисленными атрибутами. Как правило, выделенная запись имеет атрибуты $STANDARD_INFORMATION и $FILE_NAME. Каждый файл обладает атрибутом $DATA, в котором хранится содержимое файла. Если размер содержимого достигает примерно 700 байт, то атрибут становится нерезидентным и сохраняется во внешних кластерах. Таким образом, пользовательская информация хранится только в атрибутах $FILE_NAME и $DATA. Остальные типы атрибутов можно назвать служебными и предназначенными для внутреннего функционирования файловой системы (индексирование для быстрого поиска, поддержка целостности, восстановление, квотирование, разграничение доступа и пр.).

2.2.4 Разреженные атрибуты

Для уменьшения используемого места, NTFS может сохранять содержимое некоторых нерезидентных атрибутов в разреженном формате. Такой формат используется, если файл содержит кластеры, целиком заполненные нулями. В таком случае, эти кластеры не записываются на диск, но в списке серий атрибута делается специальная серия для таких кластеров. Она отличается от обычной серии тем, что содержит только поле «Длина».

Далее приведен пример сохранения файла размеров 12 кластеров с 3 нулевыми кластерами (отмечены серым цветом) в обычном и разреженном форматах.

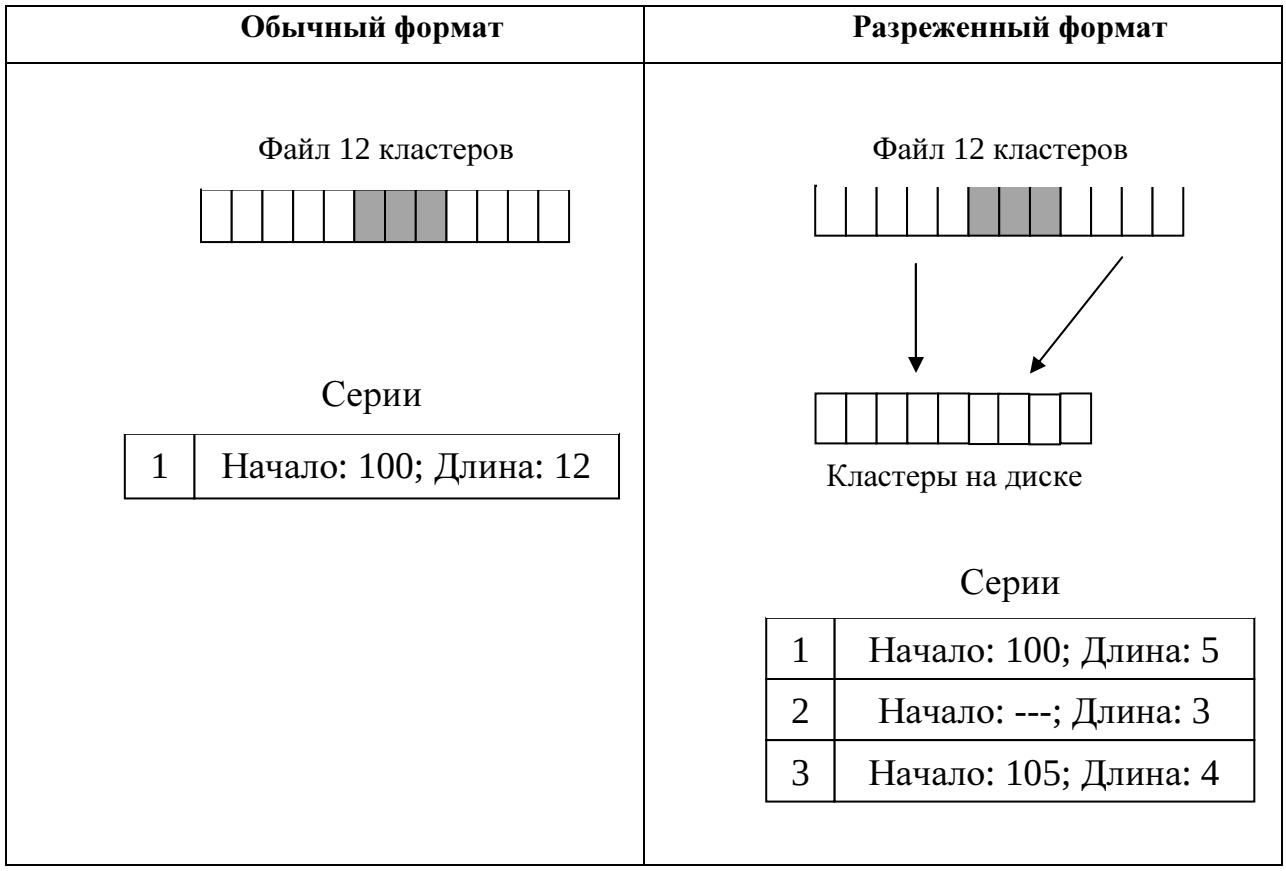

Рисунок 5: Хранение атрибутов в разреженном формате

2.2.5 Сжатые атрибуты

Для уменьшения используемого пространства NTFS позволяет хранить атрибуты в сжатом виде. Как утверждает Microsoft, сжиматься могут только нерезидентные атрибуты типа $DATA. Хотя сам алгоритм сжатия не опубликован Microsoft (как и все, что касается внутреннего устройства NTFS), он восстановлен группой Linux NTFS и описан в[Richard Russon, Yuval Fledel].

В общих чертах, сжатие содержимого атрибута происходит следующим образом.

Данные делятся на фрагменты одинакового размера, называемые *блоками сжатия*. В каждом блоке сжатия возможна одна из ситуаций:

- Все кластеры блока сжатия – нулевые. В таком случае создается разреженная серия, а на диске место не выделяется.
- После сжатия блока размер данных не уменьшился (сжатие оказалось неэффективным). В этом случае блок не сжимается, на диск записываются исходные данные и для них создается серия.

- После сжатия размер блока уменьшился. В этом случае на диск записываются сжатые данные и для них создается серия. После этой серии создается разреженная серия, чтобы общее число кластеров в сериях совпало с размеров блока сжатия.

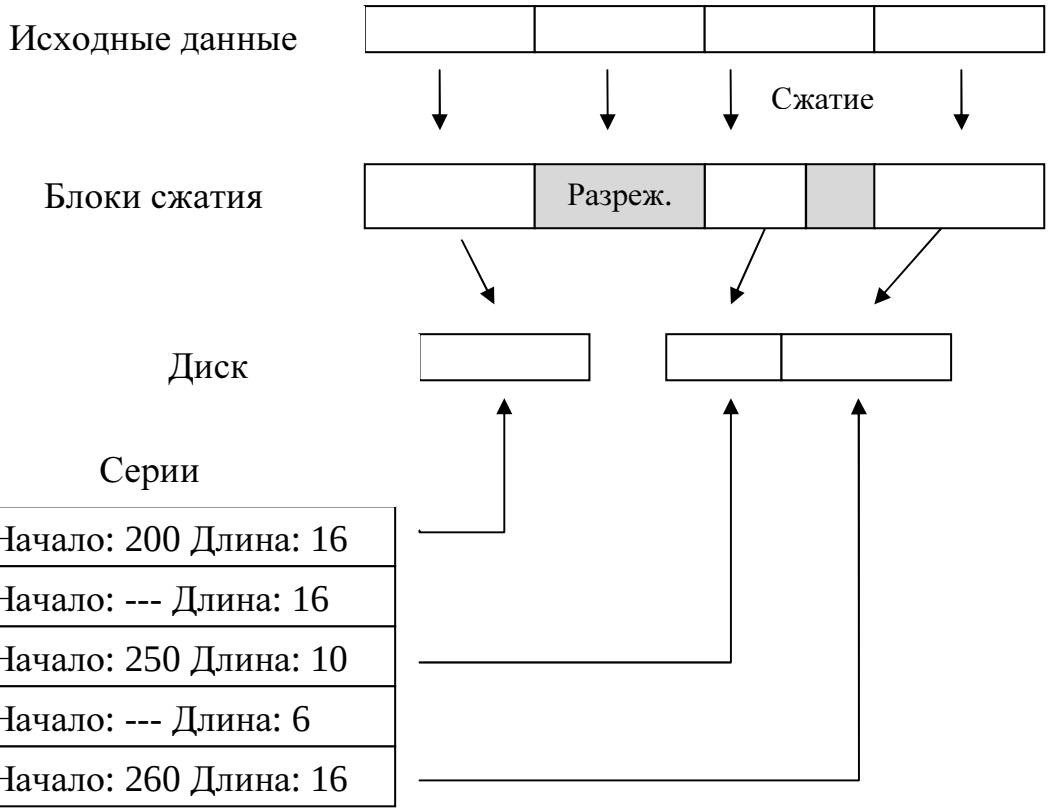

Рисунок 6: Сжатие атрибута

Для декомпрессии сжатого атрибута следует:

- Восстановить данные из всех серий
- Поделить данные на блоки сжатия
- Обработать каждый блок (он может быть не сжатым, может быть разреженным, может быть сжатым)

3. Постановка задачи

Такие возможности NTFS как хранение данных в разреженном виде и в сжатом виде сильно осложняют проведение карвинга.

Если для карвинга используется информация о заголовке файла и его длине, то в случае хранения данных в разреженном виде (если данные содержат много подряд идущих нулей) такой карвинг будет давать плохие результаты: в результат будут попадать много лишних данных.

В случае карвинга с валидацией дела обстоят еще хуже. Представим, что карвер работает следующим образом. Сначала ищет некую сигнатуру; найдя ее, проверяет, что она находится в некотором «контейнере» (объемлющей структуре данных). Для того чтобы убедиться, что найденная сигнатура принадлежит объекту, который интересен исследователю, нужно проверить принадлежность этому «контейнеру». Допустим, для этого нужно проверить, что за 2000 байт до найденной сигнатуры находится какой-нибудь признак этого контейнера (опять же, некая сигнатура). В случае разреженных данных смещение на 2000 байт назад может указывать совсем не на границу «контейнера». В таком случае валидация не пройдет, карвер пропустит найденную запись. Получается, что интересующие исследователя данные существуют, но они не нашлись. Ситуация с «контейнерами» встречается, как правило, при поиске содержимого таблиц баз данных, и в этом случае контейнером является сама таблица (которая имеет определенную структуру).

Сжатие данных еще больше снижает эффективность карвинга, т.к. в этом случае изменятся не только размер данных, но и содержимое. При этом могут не находится даже сигнатуры, т.к. при сжатии они могут изменяться. Сжатие также делает невозможным семантический карвинг (карвинг на основе лингвистического или смыслового содержания данных), т.к. обычные тексты очень хорошо сжимаются алгоритмом, реализованным в NTFS.

Таким образом, карвинг сжатых NTFS разделов становится очень сложной задачей, тем более, что для сжатия раздела не требуется специальных инструментов (обеспечивается операционной системой, достаточно выставить в настройках диска опцию «Сжимать данные»).

Задача, поставленная перед автором курсовой работы, состоит в разработке подхода для повышения эффективности карвинга сжатых NTFS разделов.

4. Обзор существующих подходов

Как же борются с NTFS-сжатием различные исследователи?

На удивление, существует крайне мало примеров решения этой проблемы; практически все утилиты карвинга обходят сжатие стороной (т.е. просто-напросто не обращают на него внимания, работают со сжатыми данными напрямую, вследствие чего показывают плохие результаты).

4.1 Поиск сжатых сигнатур

В своей статье “Shrinking the gap: Carving for NTFS-compressed files” исследователь Joachim Metz приводит пример поиска и восстановления файлов сообщений Microsoft Outlook. Идея подхода состоит в том, что зная сигнатуру заголовка искомых файлов и алгоритм сжатия, можно понять, как будет выглядеть сигнатура в сжатом виде. Далее запускается поиск такой сигнатуры на носителе и при нахождении происходит попытка декомпрессии данных. Автор отмечает, что даже столь простой подход обеспечивает достаточно хорошие результаты в случаях, если размер искомого файла менее 64 Кб. На самом деле, это число зависит от размера кластера и равно количеству байт в кластере помноженному на размер блока сжатия (который, как правило, равен 16 кластерам). В случае, описываемым автором, размер кластера составлял 4096 байт. Но будь у файловой системы размер кластера, например, 512 байт, успешно бы восстанавливались файлы размером не более 16 * 512 байт. Автор также говорит, что для восстановления файлов размером больше 64 Кб приходится применять алгоритм «грубой силы», т.е. перебор. В итоге, получался следующий алгоритм.

Пока не достигнуто некое установленное смещение:

- Попытаться провести декомпрессию блока сжатия (16 кластеров)
- Проверить, что декомпрессия прошла успешно (т.е. было извлечено 64 Кб либо достигнута граница).
- Если это так, то вернуть полученные (распакованные) данные в результат, иначе вернуть «сырые» данные.
- Если декомпрессия прошла успешно, то продолжит работу с кластера после обработанных кластеров блока сжатия. Иначе, продолжить работу со следующего кластера.

Следует отметить достоинства и недостатки такого подхода.

4.1.1 Достоинства

К достоинствам можно отнести относительную простоту (понимая алгоритм сжатия, легко понять, как выглядит сжатая сигнатура), а также эффективность работы (но на строго определенных файлах). Достоинством является также независимость от метаданных файловой системы. Есть только ”сырые” данные и только они нужны для алгоритма.

4.1.2 Недостатки

К недостаткам можно отнести то, что метод является слишком специальным: нужно анализировать поведение каждого типа искомых файлов при сжатии; а также то, что при восстановлении больших файлов требуется перебор, занимающий много времени. Это неприемлемо для больших объемов данных.

4.2 “Ручной” подход

Некоторые исследователи, зная точно, какой файл они хотят найти, ”вручную” разбирают NTFS-раздел. Имея утилиту для просмотра диска (например, NTFS Disk Explorer), исследователь находит MFT-запись, соответствующую искомому файлу, разбирает ее, понимает, в каких кластерах находится содержимое файла и сжато ли оно. Далее исследователь вручную восстанавливает содержимое файла, при необходимости выполняя декомпрессию.

4.2.1 Недостатки.

“Ручной” труд. Медленно, долго, кропотливо, чревато ошибками. Подход совершенно неприменим при поиске большого количества файлов.

4.3 Восстановление раздела

Существуют приложения для восстановления поврежденных разделов. Эти приложения не обеспечивают никаких возможностей поиска данных, то есть они специализируется на восстановлении данных, но не на цифровых расследованиях. Некоторые исследователи применяют такие приложения для своих целей: они восстанавливают поврежденный NTFS-раздел (при этом получается набор файлов, какие получилось восстановить, в «чистом» виде, т.е. не сжатые и не разреженные). Далее к полученным данным применяют стандартные подходы карвинга.

4.3.1 Достоинства

Данные, которые удалось восстановить, уже не сжаты и не разрежены, поэтому на них достаточно эффективно работают стандартные техники карвинга.

4.3.2 Недостатки

Требуется много дополнительного места для сохранения восстановленных данных, возможно, даже больше, чем объем исследуемого носителя, что неприемлемо при больших объемах данных.

Также увеличивается время проведения карвинга (время на восстановление и время на сам карвинг).

“Восстановиться” может не все, поэтому имеет смысл провести карвинг исходных данных, а является уже третьим проходом по исследуемым данным (первый проход при восстановлении, второй – по восстановленным данным). В итоге, на проведение карвинга требуется крайне много времени.

5. Предлагаемый метод

5.1 Мотивация

Несмотря на то, что изначально карвинг – это осуществления поиска данных на основании их содержимого или структуры, без использования метаданных файловой системы, в случае NTFS можно попытаться использовать эту информацию для проведения карвинга. Это возможно благодаря тому, что в NTFS информация о расположении данных хранится в записях MFT, обладающих следующими свойствами:

1. Каждому файлу соответствует одна или более записей.
2. MFT-запись обладает сигнатурой “FILE”
3. MFT-запись обладает фиксированным размером (1024 байта или меньше), никогда не фрагментируется
4. Содержимое MFT-записи не сжимается и не шифруется
5. При удалении файла соответствующие ему MFT-записи не удаляются (всего лишь помечаются как неиспользуемые и при необходимости выделяются новым файлам).
6. При форматировании раздела NTFS MFT-записи также не удаляются (перезаписывается одна главная MFT-запись $MFT, делаются другие операции, которые не затирают MFT-записи).

Первые 4 пункта говорят о том, что MFT-записи становятся “легкой добычей” для самого простого карвера, работающего по принципу Header/Length (поиск сигнатуры “FILE”, при нахождении - копирование последующих 1024 байт). Другими словами, получение информации о расположении и сжатии данных относительно просто. Дополнительную сложность представляет разбор MFT-записи, но о нем речь пойдет позже.

Пункты 5 и 6 говорят о том, что от MFT-записей избавиться совсем непросто. Удаление файлов и форматирование раздела не помогает. Если злоумышленник будет применять так называемые “шредеры” – программы, которые заполняют заданные области диска нулями или каким-нибудь “мусором”, то такие действия также расцениваются как цифровая улика и используются цифровыми криминалистами для доказательств. Как сказано выше, при удалении MFT-запись помечается как неиспользуемая и позже может быть выделена другому файлу, в результате чего ее содержимое изменится. Этот факт можно использовать для попытки уничтожить информацию. При большом количестве файлов такая попытка успеха иметь не будет, так как требуется большое количество файлов, чтобы их создание на разделе перезаписало существующие MFT-записи. Несколько десятков тысяч копий одного файла на разделе также привлечет внимание цифрового криминалиста.

5.2 Идея

Общая идея состоит в проведении карвинга сжатого NTFS раздела в два прохода:

- При первом проходе выполняется поиск и извлечение информации из MFT-записей о размещении файлов на разделе и их сжатии
- При втором проходе выполняется чтение раздела, используя накопленную информацию о размещении и сжатии файлов для:
    - Отслеживания фрагментации файлов
    - Компенсации разреженных блоков
    - Декомпрессии сжатых данных

  По ходу извлечения данных, применение к ним существующих карверов.

Таким образом, карверы будут применяться к не фрагментированным, не сжатым данным, без пропусков нулевых блоков, вследствие чего карвинг будет более продуктивным.

Так как карверы применяются к получаемым данным ”на лету” (то есть в каждый момент времени в памяти хранится только часть прочитанных данных, обрабатываемая карверами – как правило, несколько мегабайт), то не требуется дополнительного внешнего хранилища этих данных.

6. Архитектура

Ключевыми элементами в реализации предложенного подхода являются:

- Карвер MFT-записей (MFT Carver)
- Синтаксический анализатор MFT-записей (MFT Parser)
- Источник данных, читающий диск в соответствии с информацией, полученной в результате разбора MFT

6.1 MFT Carver

Задачей этого компонента является карвинг исходного носителя и излечение последовательностей байт, соответствующих записям MFT. Как говорилось выше, это простой *Header/Length* карвер (заголовок – ASCII-строка “FILE”, длина – 1024 байта (длина MFT-записи)), который применятся к исходному носителю.

6.2 MFT Parser

Задачей синтаксического анализатора MFT является разбор полученных карвером MFT данных. На вход синтаксическому анализатору поступает последовательность байт, возвращаемых MFT-карвером, а результатом работы является объект, содержащий извлеченную информацию. Возможны ситуации, при которых карвер возвращает последовательности байт, не являющиеся MFT-записями, следовательно, никакой полезной информации из них извлечь нельзя. Таким образом, синтаксический анализатор осуществляет проверку, являются ли полученные карвером данные корректной MFT-записью.

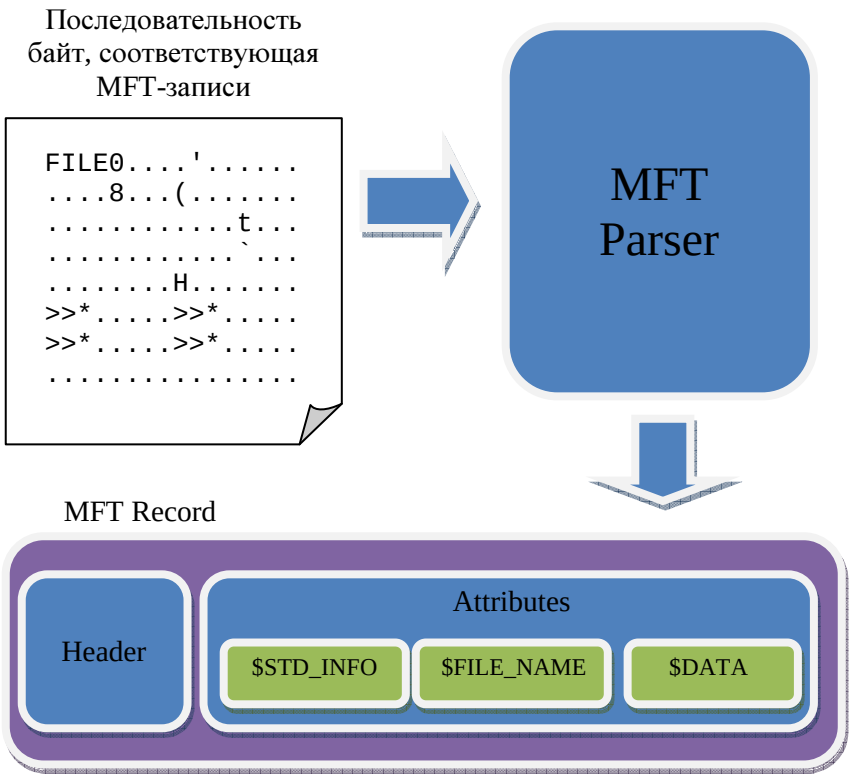

Рисунок 7: MFT Parser

6.3 Virtual File

Синтаксический анализатор извлекает довольно много элементов MFT-записи, но далеко не вся информация, полученная им, требуется для правильного чтения файла.

Поэтому из MFT-записей, полученных парсером, извлекается только нужная информация: имя файла, списки серий, размер файла, сжат/не сжат. Для хранения этой информации служат объекты, которые далее будем называть виртуальными файлами (Virtual File).

6.4 Fragmented Data Source

Этот компонент получается на вход список серий и читает исходный носитель в соответствии с этим список серий:

- возвращает содержимое кластеров, описанных в сериях
- компенсирует разреженные блоки – возвращает последовательности нулевых байт нужной длины

6.5 Compressed Data Source

Этот компонент, как и Fragmented Data Source, получает на вход список серий и читает исходный носитель в соответствии с этим списком серий, при этом производя декомпрессию содержимого. Для своей работы он использует Fragmented Data Source, управляет им.

В этом компоненте реализуется алгоритм декомпрессии сжатых данных NTFS.

6.6 Structural Reader

Задачей такого источника данных является «правильное» чтение диска на основании полученной информации из MFT-записей. «Правильное» чтение означает:

- Чтение с учетом фрагментации
- Компенсация разреженных блоков
- Автоматическая декомпрессия сжатых данных

Такой источник данных имеет стандартный интерфейс, основу которого составляет метод *int Read(byte[] buffer, int count)*. Метод принимает на вход количество байт, которое следует прочитать из источника и массив *buffer*, куда следует вернуть прочитанное, и возвращает количество реально прочитанных байт.

Structural Reader связывает вместе описанные выше элементы: управляющую информацию (набор виртуальных файвов), Fragmented Data Source, Compressed Data Source. Ниже приведено его схематичное изображение.

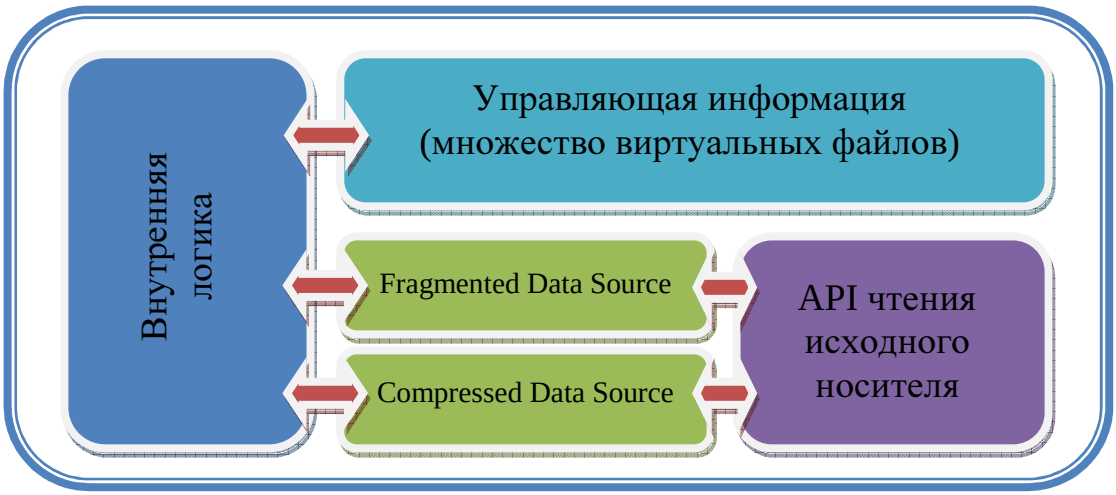

Рисунок 8: Structural Reader

Внутренняя логика работы Structural Reader (логика работы метода *Read*) заключается в следующем:

- Чтение управляющей информации (виртуальных файлов)
- «Нацеливание» Fragmented Data Source или Compressed Data Source (в зависимости от того, сжаты данные или нет, что тоже является частью управляющей информации) на нужную позицию исходного носителя
- Определение того, сколько нужно прочитать данных с исходного носителя, выполнение этого чтения средствами Fragmented Data Source или Compressed Data Source и возвращение результата.

6.7 Схема работы

Ниже представлена схема работы всей системы.

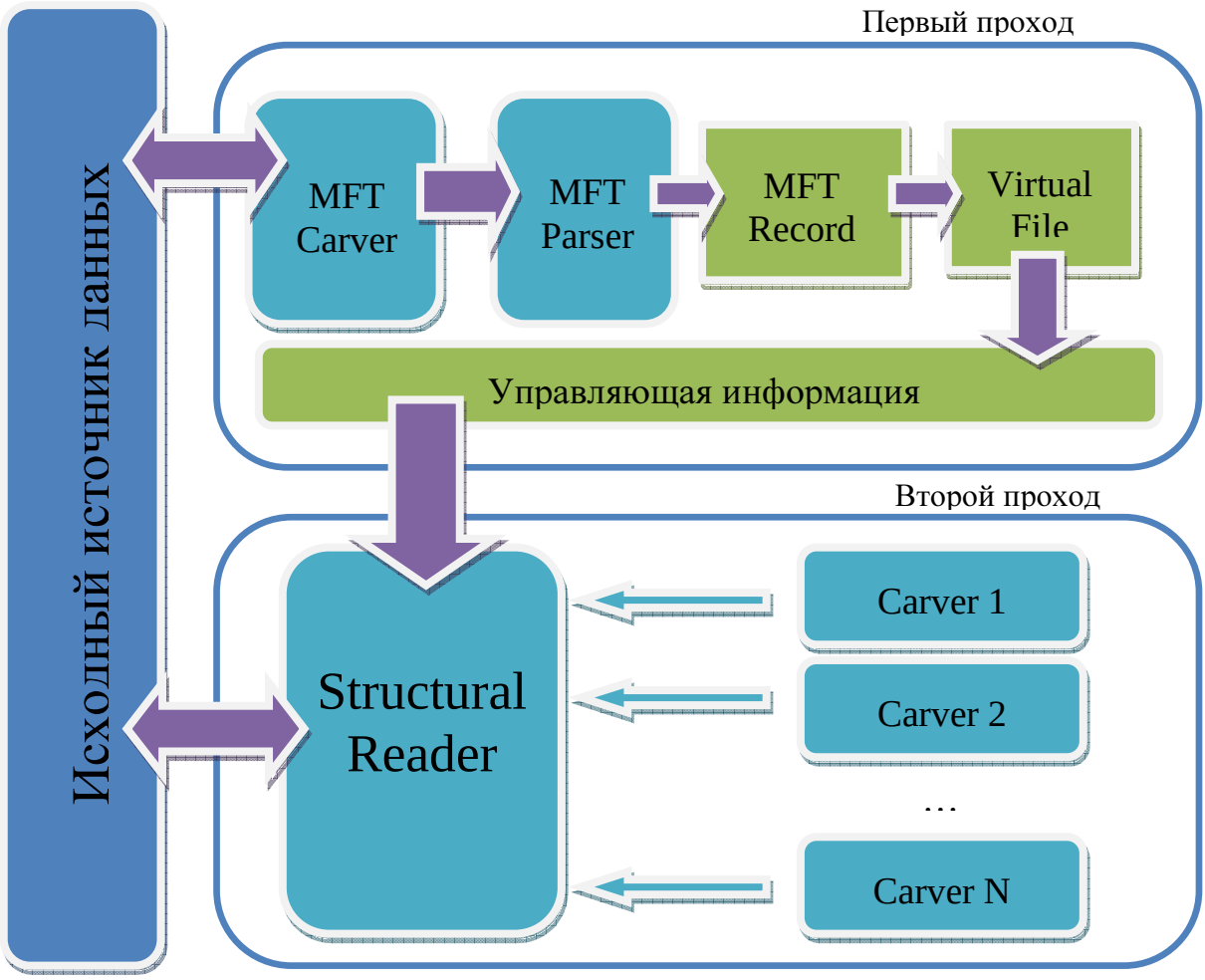

Рисунок 9: Общая схема работы

Карвинг сжатого NTFS-раздела происходит в два прохода:

- При первом проходе собирается нужная служебная информация для “правильного” чтения диска
  - MFT-карвер извлекает из раздела последовательности байт, соответствующие MFT-записям
  - Синтаксический анализатор MFT разбирает полученные MFT-карвером данные, возвращая при этом объект (MFT Record), который предоставляет простой доступ ко всем элементам MFT-записи.
  - Из полученной MFT-записи извлекается только нужная для правильного чтения носителя информация, которая сохраняется в Virtual File
  - Полученные виртуальные файлы накапливаются, в результате чего образуется управляющая информация
  - Полученная управляющая информация используется для проведения второго прохода

- Второй проход включает в себя чтение исходного носителя в соответствии с управляющей информацией, полученной при первом проходе
    - Управляющая информация передается компоненту Structural Reader, который служит для корректного чтения исходного источника данных
    - Существующие карверы в качестве источника данных используют Structural Reader. Таким образом, карверы получают на вход корректные данные (не фрагментированные, без пропусков нулевых блоков, не сжатые), в результате чего показывают хорошие результаты: находят и извлекают больше искомых файлов или записей, нежели при применении к ”сырому” носителю.

7. Применение разработанного подхода

Описанный подход был применен в программном продукте, предназначенном для проведения цифровых расследований. Данный продукт реализует достаточно широкий набор карверов некоторых форматов файлов. При применении карверов к NTFS-разделам с установленным сжатием, продукт показывал неудовлетворительные результаты.

В результате реализации предложенного подхода появилась возможность успешно проводить карвинг сжатых NTFS-разделов, а также улучшились результаты карвинга разделов без сжатия. В следующей таблице представлены количества найденных записей некоторых типов с применением подхода и без него, полученные при карвинге сжатого NTFS-раздела.

Тип искомой записи	**Без применения подхода**	**С применением подхода**
История активности веб-браузера FireFox	0	576
История активности веб-браузера Internet Explorer	3	41
История звонков и бесед Skype	2	602

Таблица 2: Результаты применения подхода

В продукте был реализован “комбинированный подход”. Его отличие от описываемого ранее заключается в проведении первого прохода. Напрямую к исходному разделу применяется не только карвер MFT-записей, но и реализуемые продуктом карверы. Такое решение повышает эффективность карвинга, так как:

- данные могут находиться в кластерах носителя, которые не описываются ни одной найденной при первом проходе MFT-записью – Structural Reader никогда не вернет содержимое таких кластеров
- данные могут находиться в самой MFT-записи (если их размер очень маленький, не более 700 байт), а ее заголовки будут повреждены и структурный анализатор MFT посчитает ее некорректной
- заранее неизвестно, насколько “плох” исходный раздел для карверов: насколько фрагментирован, установлено или нет сжатие

Таким образом, первый проход также может вносить непосредственный вклад в результаты карвинга, а не быть только вспомогательным для второго.

Заключение

В рамках курсовой работы была рассмотрена и изучена проблема карвинга сжатых NTFS-разделов. Был предложен способ повышения эффективности проведения такого карвинга. Описанный подход с успехом применен в существующем программном продукте.

Список литературы

1. Anandabrata Pal, N. M. (2009). *The Evolution Of File Carving [The benefits and problems of forensics recovery]*.
2. Cohen, M. (б.д.). Advanced Carving techniques.
3. Garfinkel, S. L. (2007). Carving contiguous and fragmented files with fast object validation. *Digital investigation 4S* .
4. Merola, A. (2008). *Data Carving Concepts.*
5. Metz, J. (2006). *A smart carving approach.*
6. Mikus, N. (2005). *An analysis of disk carving techniques.*
7. Pereira, M. T. (2009). Forensic Analysis of the Firefox 3 Internet History and Recovery. *Digital Investigation Journal, May* , 93–103.
8. Richard Russon, Yuval Fledel. NTFS Documentation.
9. Sonntag, M. (2009). *File carving.*
10. Кэрриэ, Б. (2007). *Криминалистический анализ файловых систем*. Издательство "Питер".