Санкт-Петербургский государственный университет

Кафедра системного программирования

Медведев Андрей Александрович

Анализ файла гибернации ОС Windows 10

Курсовая работа

Научный руководитель:
ст. пр. Губанов Ю. А.

Научный консультант:
Тимофеев Н. М.

Санкт-Петербург
2017

Оглавление

Введение

Одну из важнейших частей криминалистики составляет цифровая криминалистическая экспертиза, потому как в большинстве совершаемых преступлений так или иначе используются цифровые устройства.

Для более детального проведения расследования от специалистов требуются высокие навыки обращения с компьютерной техникой и умение извлекать нужную информацию из цифровых устройств в условиях ограниченного доступа к данным. Одним из возможных сценариев является задержание подозреваемого совместно с изъятием у него персонального компьютера. В качестве доказательства или опровержения его причастности к совершению преступления могут быть использованы данные, найденные на компьютере. Однако, не всегда у специалистов есть прямой доступ к таким данным, поэтому, одним из возможных способов поиска доказательств является анализ файла гибернации.

Файл гибернации является частью процесса гибернации, позволяющего сохранять состояние устройства в энергонезависимое хранилище. При этом оперативная память компьютера и данные о состоянии ОС сохраняются в сжатом виде в системный файл hiberfil.sys, затем питание компьютера полностью отключается. При возобновлении работы компьютера состояние операционной системы восстанавливается из этого файла.

Вместе с выходом новых версий Windows формат файла гибернации менялся, и инструменты для его анализа переставали быть актуальными. Специалистам приходилось восстанавливать формат файлов гибернации новых версий, чтобы иметь возможность анализировать их содержимое. На данный момент последней версией ОС является Windows 10, формат её файла гибернации также не изучен до конца.

Постановка задачи

Целью данной работы является исследование формата файла гибернации ОС Windows 10. Для достижения этой цели были поставлены следующие задачи:

- провести обзор существующих решений для анализа файла гибернации Windows 10;
- изучить структуру файла гибернации Windows 10;
- провести сравнительный анализ файла гибернации последней версии ОС с предыдущими версиями;
- разработать прототип программы на C++, позволяющей восстанавливать образ оперативной памяти из файла гибернации ОС Windows 10;

1. Обзор существующих решений

В связи с изменениями, внесёнными в структуру файла гибернации с последними вышедшими версиями ОС Windows, большинство известных решений для анализа файла гибернации утратили свою актуальность, например Volatility Project [5], Blade [2], Belkasoft Evidence Center [1]. На данный момент известно лишь о нескольких инструментах, позволяющих проводить анализ файла гибернации версии ОС Windows 8 или 10.

Одно из таких решений – Hibr2Bin, автором которого является Matthew Suiche. Данный проект является бесплатным и общедоступным, к тому же, обладает открытым исходным кодом. Несмотря на перечисленные плюсы, результат работы Hibr2Bin оставляет желать лучшего, потому как подвергается анализу лишь один набор данных из файла гибернации, номер первой страницы которого указан в заголовке; как следствие, часть данных может быть упущена из виду.

Второй вариант – Hibernation Recon [3]. Это инструмент, предоставленный командой Arsenal Recon, способен проводить углублённый анализ файлов гибернации как младших версий ОС Windows, так и Windows 8 и 10. Купленная лицензия на этот продукт открывает доступ к полному функционалу, который позволяет производить анализ нескольких файлов гибернации одновременно и восстанавливать данные из прошлых сеансов гибернации, если такие в файле имеются. Есть также и бесплатная версия, функциональность которой ограничивается восстановлением образа из файла гибернации.

2. Исследование структуры файла гибернации

Структура файла гибернации является закрытой информацией Microsoft. Поэтому вся информация о его структуре и особенностях была восстановлена специалистами с помощью методов обратной разработки.

Основная информация о формате файле гибернации, структурах, которые в нём содержатся и алгоритмах сжатия данных была найдена в статье [4] посвящённой исследованию файла гибернации. Однако, найденной информации оказалось недостаточно для того, чтобы получить полное представление о формате этого файла. Поэтому для восстановления структур, использующихся в файле гибернации были использованы средства отладки среды Windows, а именно, инструмент WinDBG [6]. Содержимое структуры можно получить благодаря использованию символов отладки ядра соответствующей версии ОС Windows и команды dt (DisplayType) [7], которая служит для отображения информации о типе данных.

Файл гибернации делится на непрерывные сегменты памяти размером 4096 байт, при этом он представляет из себя набор из нескольких структур.

В первом сегменте hiberfil.sys находится заголовок – структура, в которой содержится основная информация, касающаяся процесса гибернации: состояние, системное время на момент гибернации, номера страниц, с которых начинаются наборы восстановления, и т.д. Далее следует сегмент, содержащий данные о состоянии процессоров на момент гибернации. Всё остальное пространство занимают наборы восстановления, содержащие данные из оперативной памяти в сжатом виде.

2.1. Заголовок файла гибернации

Заголовок файла гибернации представляет из себя структуру, содержащую информацию о состоянии системы на момент гибернации и о самой процедуре гибернации. Как говорилось ранее, содержимое струк-

туры может различаться в зависимости от версии ОС, но меняется по большей части лишь относительное расположение полей структуры.

После успешного восстановления системы из файла гибернации вся хранимая информация в нём, кроме первой страницы, заполняется нулями.

В Таб. 1 представлен пример расположения нескольких основных значений в структуре.

Значение	**Относительный адрес**	**Размер**
Сигнатура	+0×000	4 бита
Системное время	+0×020	64 бита
Кол. страниц для загрузчика	+0×058	8 бита
Адрес первой загрузочной страницы	+0×068	8 бита

Таблица 1: Основные значения в заголовке

2.2. Контекст процессора

Данная структура служит для корректного и полноценного восстановления состояния процессоров после гибернации. Она хранит в себе все значения регистров и полностью зависит от модели процессора.

2.3. Хранение данных

Данные в файле гибернации группируются в один или несколько наборов восстановления. Каждый набор сохраняет очерёдность страниц из памяти, но загружается на определённой стадии восстановления состояния системы.

В файле гибернации, как правило, встречается как минимум один набор данных, номер страницы которого указывается в соответствующем значении заголовка файла гибернации (см. Таб. 1). Это набор основных загрузочных данных, он хранит в себе непосредственно данные из оперативной памяти. Общее количество страниц в несжатом виде, содержащихся в этом наборе, опять же, указано в заголовке (см. Таб. 1).

Наборы восстановления данных делятся на наборы сжатых данных, в каждом из которых хранится максимум 64 килобайт данных. Каждый такой набор начинается с 32-битного заголовка, продолжается определённым количеством дескрипторов страниц и заканчивается непосредственно сжатыми данными.

В младших 8 битах (или 4 битах в 32-битных системах) заголовков наборов сжатых данных хранится количество дескрипторов страниц. Следующие 22 бита хранят размер сжатых данных. Самый старший бит служит для определения алгоритма сжатия, который использовался при сохранении данных.

Для того чтобы определить, какой алгоритм сжатия применялся в конкретном наборе сжатых данных, нужно, в первую очередь, сравнить размер сжатых данных с количеством дескрипторов страниц, умноженным на размер страницы. Если значения равны, то данные были сохранены без каких-либо изменений. Если же значения различны, то в случае, если соответствующий старший бит заголовка сегмента выставлен, использовался алгоритм LZ77+Huffman, иначе — Plain LZ77.

За заголовком набора сжатых данных следует набор из дескрипторов страниц, которые хранят информацию о непрерывной последовательности страниц физической памяти. В каждом из таких дескрипторов указывается количество страниц в последовательности и номер первой страницы в физической памяти.

2.4. Алгоритмы сжатия XPRESS

С целью минимизации использования памяти при создании hiberfil.sys используются вариации алгоритма сжатия XPRESS. Существует три варианта, основными характерными различиями которых являются скорость обработки данных и сложность применения:

- самый быстрый по исполнению — Plain LZ77, который представляет из себя алгоритм Абрахама Лемпеля и Якоба Зива опубликованный в 1977 году;

- LZ77+Huffman — более медленный вариант алгоритма за счёт дополнительного использования кодировки кодами Хаффмана;
- третий вариант — LZNT1, реализует алгоритм LZ77, но с упрощённой процедурой сжатия. На данный момент известно [4], что он не применяется при сжатии;

2.4.1. Plain LZ77

Основная идея этого алгоритма заключается в замене повторного вхождения последовательности символов ссылкой на одно из предыдущих вхождений. Для этого используется метод скользящего окна: совпадения в строке кодируются длиной совпадения и смещением, при необходимости восстановления закодированных данных программа обращается по положению указанному в параметрах и копирует нужное количество символов.

В реализации этого алгоритма для хранения смещения используется 13 бит, поэтому максимальное смещение может составлять не более 8192 символов. Для хранения длины совпадающей подстроки может использоваться по необходимости 3, 4, 8 или 16 битов. Кроме того, перед каждой последовательностью символов, содержащих сжатые данные, располагается 32 бита флагов, которые нужны для обнаружения ссылок на предыдущие вхождения.

2.4.2. LZ77+Huffman

В отличие от обычного LZ77 алгоритма эта модификация использует для большей эффективности сжатия коды Хаффмана. Кроме того, в этом алгоритме длина и смещение повторного вхождения могут принимать значения вплоть до 65535. Алгоритм сжатия данных делится на три последовательных этапа:

- составление набора символов алфавита Хаффмана, подсчёт количества их вхождений и создание промежуточного сжатого буфера данных с помощью LZ77

- построение кодов Хаффмана на основании частоты появления символов
- преобразование ранее созданного промежуточного буфера согласно построенным кодам

Этап LZ77-кодирования На этом этапе из входного буфера генерируется стандартный, полученный с помощью LZ77-алгоритма, буфер сжатых данных. Каждому символу или последовательности символов, ранее встречавшимся в буфере, ставится в соответствие символ алфавита Хаффмана. По мере применения алгоритма LZ77 также ведётся подсчёт количества повторений символов.

Символ алфавита Хаффмана в контексте этого алгоритма – значение в пределах от 0 до 511. Если необходимо закодировать литерал, то в качестве символа алфавита Хаффмана используется он сам (значение от 0 до 255 включительно, в шестнадцатеричном представлении). В случае, когда необходимо закодировать последовательность символов, встречавшуюся ранее, используется длина и смещение повторного вхождения, где функция GetHighBit служит для вычисления индекса старшего ненулевого бита. Алгоритм представлен на Рис. 1.

```
If (Length – 3) < 15
    HuffmanSymbol = 256 + (Length – 3) + (16 * GetHighBit(Distance))
Else
    HuffmanSymbol = 256 + 15 + (16 * GetHighBit(Distance))
```

Рис. 1: Вычисление символа алфавита Хаффмана

Генерация кодов Хаффмана Основная цель этого этапа — закодировать информацию о повторном вхождении последовательности символов наиболее эффективным образом. Для этого применяется кодировка Хаффмана. Каждому символу алфавита Хаффмана ставится в соответствие последовательность битов, получаемая согласно алгоритму кодирования Хаффмана и частоте вхождений символа.

Алгоритм кодирования применяется с модификациями. При построении дерева кодов символов, если глубина дерева оказывается больше 15, то значения количества вхождений каждого символа делятся на 2 с оценкой сверху и дерево строится заново; этот метод применяется для того, чтобы ограничить длину кодов. При окончательном построении дерева каждый уровень дерева сортируется по значению символа Хаффмана, что в дальнейшем позволяет по длине кода определить соответствующий символ.

Заключительный этап На этом этапе формируется окончательный вид сжатых данных.

Прежде всего, в выходной поток записывается таблица длин соответствующих кодов Хаффмана. Благодаря ранее описанным ограничениям на коды Хаффмана, длина каждого кода будет выражаться не более чем в 4 битах. Длина кодов символов Хаффмана с чётным значением сохраняется в младшие четыре бита, а с нечётным – в старшие. Таким образом, для 512 символов создаётся таблица размером в 256 байт. Во время расшифровки данных сжатых этой вариацией алгоритма по этой таблице восстанавливаются соответствия символов Хаффмана и их кодов.

Затем данные, полученные на первом этапе и модифицированные кодами Хаффмана, записываются в выходной поток непосредственно после таблицы длин кодов.

3. Сравнительный анализ

Гибернация была доступна ещё с ранних версий ОС Windows. С каждым выпуском новой версии продукта этот файл структурно изменялся, при этом его предназначение сохранялась таким же. Наибольшим изменениям файл гибернации подвергся с появлением ОС Windows 8.

В версиях ОС Windows 7 и младше файл гибернации мог содержать множество артефактов из давно прошедших сеансов гибернации этого устройства. Происходило это из-за того, что файл гибернации располагается в фиксированном месте памяти и при успешном восстановлении из него данные восстановления не удаляются намеренно. Могли остаться нетронутые промежутки с неактуальными, но полезными для экспертов цифровой криминалистики данными. Этим можно было воспользоваться и попытаться восстановить информацию о состоянии компьютера из давно прошедших сеансов гибернации. В последних версиях ОС Windows эта возможность была утрачена в силу того, что при успешном восстановлении из файла гибернации все страницы, составляющие его, кроме заголовка, заполняются нулями.

Кроме того, принцип хранения данных в файле гибернации версий ОС Windows 8 и 10 отличается от способа хранения в более ранних версиях. Раньше информация о состоянии ОС сохранялась при помощи таблиц адресации и наборов XPRESS-данных. Такой подход делал процесс восстановления из файла гибернации излишне трудоёмким за счёт того, что адрес каждой новой порции информации нужно было вычислять согласно множеству таблиц адресации. Начиная с ОС Windows 8 способ хранения данных был упрощён в угоду простоте восстановления состояния операционной системы. В последних версиях файлов гибернации используются несколько наборов восстановления, содержащих последовательность дескрипторов страниц и следующих за ними данных.

Все остальные характерные черты файл гибернации ОС Windows 10 унаследовал от своих предыдущих версий. Между тем, даже в разных

выпусках 10 версии ОС Windows появляются небольшие изменения в структуре заголовка. Поэтому при анализе файла гибернации нужно обязательно уточнить версию ОС, под управлением которой находилось устройство.

4. Разработка прототипа

На основе полученных знаний о структуре файла гибернации ОС Windows 10 автором данной работы был разработан прототип программы, способной восстанавливать образ физической памяти по содержимому файла гибернации 64-битной версии ОС Windows 10. Прототип был создан для проверки гипотез о структуре файла и его содержимого.

Прототип программы реализован на языке C++ и предоставляет функционал для восстановления физической памяти согласно информации, хранящейся в файле гибернации.

В процессе обработки файла гибернации необходимо постоянное обращение к его содержимому. Учитывая потенциально большой размер файла гибернации было решено использовать буфер объёмом равным объёму страницы, 4096 байт, для более быстрого обращения к данным.

Обработка файла гибернации делится на два этапа: анализ заголовка файла гибернации и обработка сжатых данных.

На этапе анализа заголовка производится восстановление информации касательно процесса гибернации и содержимого файла. Например, номера первых страниц наборов восстановления, дата создания файла гибернации, состояние и т.д. Информация, полученная на этом этапе, по большей части несёт информативный характер и практически не влияет на этап декомпрессии сжатых данных.

После получения информации из заголовка начинается декомпрессия набора восстановления, адрес которого был получен на первом этапе. В соответствии с данными, указанными в дескрипторах страниц, по кусочкам восстанавливается изначальный образ оперативной памяти компьютера.

Результатом работы программы является образ оперативной памяти компьютера, с которого был получен файл гибернации.

Для проведения тестирования прототипа программы был проведён сравнительный анализ результата обработки программой файла гибернации компьютера с установленной на нём 64-битной ОС Windows 10

и соответствующего образа оперативной памяти. Анализ происходил путём сравнения соответствующих участков памяти образов.

Данные в образе, полученном при выполнении программы, частично совпали с данными в снимке оперативной памяти. Причём несовпадение выражалось в отсутствии цельных порций данных в восстановленном образе памяти, что говорит о том, что часть данных из файла гибернации хранилась в наборах восстановления которые не были восстановлены программой.

Из полученных результатов можно сделать вывод, что внутреннее содержимое файла гибернации и способы хранения данных были установлены верно, но часть данных не была восстановлена за счёт того, что в прототипе программы анализируется лишь один набор восстановления.

Заключение

В рамках данной работы были получены следующие результаты:

- проведён обзор существующих решений для анализа файла гибернации Windows 10
- изучена структура файла гибернации Windows 10
- проведён сравнительный анализ файла гибернации последней версии ОС с предыдущими версиями
- разработан прототип программы на C++, позволяющей восстанавливать образ оперативной памяти из файла гибернации ОС Windows 10

Список литературы

[1] Belkasoft. — 2017. — URL: https://belkasoft.com.

[2] Blade. — 2017. — URL: http://www.bladeforensics.com.

[3] Hibernation Recon. — 2017. — URL: https://arsenalrecon.com/apps/hibernation-recon/.

[4] Joe T. Sylve Vico Marziale Golden G. Richard. Modern windows hibernation file analysis // Elsevier. — 2016.

[5] Volatility Project. — 2017.

[6] Microsoft. — WinDBG, 2017. — URL: https://msdn.microsoft.com/en-us/library/windows/hardware/ff551063.aspx.

[7] WinDBG cmnd. -dt. — 2017. — URL: https://msdn.microsoft.com/en-us/library/windows/hardware/ff542772.aspx.