Санкт-Петербургский государственный университет

Математико-механический факультет

Кафедра системного программирования

Дешифрация образа диска с защитой BItlocker To Go инструментами анализа дампа памяти

Курсовая работа студента 344 группы
Грабового Филиппа Николаевича

Научный руководитель
ст. преп. Ю. А. Губанов

Санкт-Петербург
2015

Оглавление

1 Введение

В современном мире средства обеспечения безопасности информации стремительно развиваются и становятся доступны простому пользователю. Так, возможность шифрования данных на жестких дисках — BitLocker — была добавлена компанией Microsoft в линейку ее операционных систем с выходом версии Vista, в 2007-ом году. Конечно же, такими средствами защиты может воспользоваться и злоумышленник, пытающийся скрыть важные материалы. По этой причине важно уметь интерпретировать информацию с зашифрованных носителей, такая задача может возникнуть в ходе криминалистических исследований.

Важно отметить, что в сравнении с другими средствами шифрования дисков (например, TrueCrypt) очевидны простота и удобство использования утилиты от Microsoft. Ее интегрированность в операционную систему обеспечивает ряд преимуществ с пользовательской точки зрения:

1. Отсутствие необходимости установки дополнительных программ.
2. Автоматическое обновление.
3. При работе со съемными носителями — отсутствие особенных аппаратных требований к компьютеру.

Это побуждает нас в первую очередь исследовать именно BitLocker, в особенности режим To Go, представленный с выходом ОС Windows 7 и позволяющий шифровать съемные носители.

1.1 BitLocker. Понятия и определения

1.1.1 Ключи BitLocker

Рассмотрим типы ключей BitLocker и их назначения.

1. Full Volume Encryption Key (FVEK) — ключ шифрования всего тома, использующийся в алгоритмах шифрования и шифрующий диск. При алгоритме AES-128 занимает 128 бит, при AES-256 — 256 бит. Хранится на носителе в зашифрованном виде, шифруется с помощью Volume Master Key.

2. TWEAK — ключ, использующийся совместно с FVEK, если алгоритм шифрования диска использует технологию Elephant Diffuser. Размер совпадает с размером ключа шифрования всего тома, и в структурах, содержащих ключи шифрования (в памяти ОС, в метаданных на диске), FVEK и TWEAK хранятся последовательно, друг за другом. Также шифруется с помощью Volume Master Key.

3. Volume Master Key (VMK) — ключ, использующийся для шифрования FVEK и TWEAK. Размера 256 бит, также хранится на зашифрованном носителе в нескольких экземплярах, которые тоже могут быть зашифрованы ключом восстановления, внешним ключом, пользовательским ключом, с помощью TPM (Trusted Platform Module). Если копия не зашифрована, такой ключ называется чистым.

4. Ключ восстановления (Recovery Key) — ключ, предусмотренный для восстановления данных при потере внешнего ключа или другого способа расшифровать носитель. Пароль восстановления хранится на диске, состоит из 48 цифр, образующих последовательно в 8 групп. Делением каждой из них как целого числа на 11 получаются 16-битные значения, которые и образуют ключ длиной 128 бит.

5. Внешний ключ (External Key, Startup key) — ключ, зачастую хранящийся на дополнительном носителе и использующийся для доступа к данным. Таких ключей может быть несколько.

6. Пользовательский ключ (User key) — ключ, задающийся пользователем и использующийся для доступа к носителю.

1.1.2 Алгоритмы шифрования BitLocker

BitLocker использует разные методы шифрования для работы с исходными данными тома и ключами. Для решения первой задачи разработчики выбрали алгоритм, подходящий под следующие требования[1]:

1. Защищает конфиденциальность данных.

2. Шифрует сектора данных размером 512, 1024, 2048, 4096 или 8192 байт (при этом размер зашифрованных данных не должен превышать размера исходных — в противном случае это приводит к уменьшению вместимости диска).

3. Зависит от порядкового номера сектора.

4. Достаточно быстрый в сравнении со скоростями чтения данных с устройства.

5. Необходимо, чтобы при изменении зашифрованных данных нельзя было понять/предсказать изменения в исходной информации (этим можно было бы воспользоваться для расшифровки).

AES-CBC (Advanced Encryption Standard with Cipher Block Chaining) имеет один важный недостаток, не позволивший использовать его напрямую: из-за последовательности преобразований изменение одного блока зашифрованных данных приведет к понятному изменению следующих, что может быть использовано для расшифровки данных. Алгоритм был улучшен добавлением дополнительного ключа (TWEAK), которым преобразуется исходный сектор операцией xor, и двумя диффузорами. Результат далее шифруется алгоритмом AES-CBC.

Для шифрования ключей (VMK или FVEK, хранимые в метаданных тома) используется алгоритм AES с режимом CCM (Counter with CBC-MAC).

1.1.3 Шифрование секторов

Рассматриваемая утилита шифрует весь том, кроме трех компонент[2]:

1. Заголовок тома — имеет длину 512 бит, совпадает с заголовком загрузочного сектора. Содержит смещения, относящиеся к трем FVE блокам метаданных.

2. Поврежденные сектора — если они отмечены как нечитаемые.

3. Метаданные тома — блоки данных на носителе информации, хранящие ключи BitLocker'а в зашифрованном виде. Рассмотрим их подробнее.

1.1.4 Метаданные BitLocker

Зашифрованный том содержит три блока метаданных, для обеспечения избыточности (в случае повреждения одного можно воспользоваться другими). Смещения каждого из них могут быть найдены в заголовке тома и в заголовке любого из самих блоков. Сигнатура "-FVE-FS-" обозначает начало метаданных (с помощью нее же можно вычислить смещения самостоятельно, имея образ тома). Внутри блок метаданных состоит из трех важных составляющих[3]:

1. Заголовок блока, содержит смещения на начала трех блоков метаданных и заголовка тома.

2. Дополнительный заголовок, хранящий GUID, код шифрующего алгоритма, время и дату установки BitLocker To Go.

3. Данные — массив структур, содержащих ключи в зашифрованном виде: FVEK и VMK. При этом VMK, шифрующийся при необходимости несколькими ключами, содержится в нескольких экземплярах, по-разному зашифрованный.

1.2 Структуры процессов.

В ходе работы потребуется проанализировать файл дампа памяти компьютера, поэтому необходим механизм, позволяющий убедится в его валидности. Необходимо проверить, есть ли в нем структуры, описывающие процессы, запущенные на момент снятия дампа. В современных операционных системах Windows как правило постоянно функционирует ряд системных процессов— System.exe, explorer.exe, dwm.exe и т. д. Если в файле есть записи, соответствующие им, он может быть интерпретирован как дамп памяти.

Ядро Windows использует структуру EPROCESS для описания процесса и хранения необходимой информации о нем[4]. Такие структуры отведены для всех процессов, запущенных в системе, и исполняемых в ядре в том числе. Для удобного хранения определенные поля структур содержат ссылки на соседние и тем самым формируют списки: ActiveProcessLinks образует список из всех структур процессов (так как каждый является активным), SessionProcessLinks —процессы, исполняемые внутри сессии, JobLinks — для процессов, принадлежащих объектам Job.

Важной частью EPROCESS структуры является структура KPROCESS (хранится в поле PCB — Process Control Block, в памяти ядра). Она используется планировщиком ОС Windows и содержит важные параметры процесса для его идентификации, в т. ч. системных: код диспетчера, планировщика, а также код учета прерываний и времени.

Параметры структуры, позволяющие идентифицировать процесс, которые вычисляются в ходе анализа файла дампа памяти:

- Process ID — уникальный номер, присваеваемый процессу ядром.
- Process Name — имя процесса.
- Eprocess Size — размер структуры.
- Directory Table Base — значение, позволяющее вычислить адрес таблицы виртуальных адресов процесса в директории таблиц.
- Thread List Heads — ссылка на начало списка с потоками процесса (их описывают структуры ETHREAD).
- Active Process Links — ссылки на следующий и предыдущий процесс в списке активных процессов (на структуры EPROCESS).
- VAD Tree — дерево ссылок на дескрипторы виртуальных адресов.

1.3 Формулировка задачи

Полученные сведения о методах шифрования BitLocker нужно применить, чтобы расшифровать образ диска съемного носителя информации (USB Flash накопителя, именно их шифрует BitLocker To Go). В ходе криминалистических экспертиз этот функционал будет полезен, так как на портативных компьютерах преступники (или наоборот, жертвы преступлений) могут хранить важную информацию, которая может помочь при расследовании происшествия. При этом удобство использования решения от Microsoft приводит к повышению количества зашифрованных носителей в целом, и, в частности, повышается вероятность столкнуться именно с этим типом шифрования.

Для расшифровки внешнего диска нужен FVEK, хранящийся в том числе на носителе в защищенный виде другими ключам. Однако подбирать VMK, внешний ключ или ключ восстановления для получения FVEK мы не будем. Вместо этого предполагается, что ключ шифрования всего тома и TWEAK могут быть найдены уже в расшифрованном виде. Например, в оперативной памяти компьютера, который использует этот внешний накопитель. В таком случае, если пользователь вводил пароль или восстанавливал доступ ключом восстановления, то искомые ключи могут быть найдены в оперативной памяти компьютера. Для ее анализа потребуется снять дамп оперативной памяти машины.

Такой подход требует дополнительных данных, но при этом имеет ряд преимуществ:

1. Ключ доступен в явном виде (очевидное преимущество перед решением с помощью перебора ключа и аргумент искать сразу FVEK — нахождение любого другого потребовало бы дополнительной операции расшифровки главного ключа и в целом количество ситуаций, при которых меняются ключи носителя, для FVEK меньше)

2. Ключ может быть найден в оперативной памяти и после использования накопителя в течение какого-то времени (пока он там будет сохранен).

Если ключей не будет в представленном дампе памяти, следует сделать другой дамп (в нужный момент времени) и проанализировать его. Если ключ присутствует, то расшифровка образа диска завершится успешно за сравнительно небольшой промежуток времени. Возможно, что наборов ключей в оперативной памяти будет несколько — в таком случае обнаружатся они все и возможна расшифровка нескольких носителей.

Итак, требуется найти в данном дампе памяти ключи и с помощью них расшифровать имеющийся образ диска.

2 Используемые решения

В этой части подробно описываются используемые в ходе исследования и разработки средства.

2.1 Снятие образов диска и дампов памяти

Рассмотрим способы создания файлов дампа памяти и образа USB накопителя, необходимые для работы. Приведенные решения сохраняют данные без большого увеличения их объема, в файлах типа .mem.

2.1.1 Belkasoft Live RAM Capturer

Эффективное программное обеспечение для снятия дампов оперативной памяти компьютера, разработано компанией Belkasoft. Утилита позволяет обходить активные системы противодействия отладке процессов, запущенных в операционной системе (запуском в привилегированном режиме) и получать актуальные, а не нулевые или случайные данные для таких защищенных процессов. Подходит для снятия слепка оперативной памяти.

2.1.2 AccessData Forensic Toolkit Imager

Удобное решение от компании AccessData для создания и просмотра файлов образов. К преимуществам стоит отнести возможность добавления комментариев к файлу, сегментирование, поддержку нескольких форматов и подтверждение целостности данных через контрольные суммы. Воспользуемся им для создания образа USB диска.

2.2 *Инструменты для анализа структур EPROCESS*

В ходе работы для анализа структур EPROCESS использовался инструментарий, созданный Овчинниковым Антоном в ходе его курсовой работы за 2011-2012 годы[5]. Он основан на сигнатурном анализе, т.е. поиске сигнатур, соответствующих EPROCESS, в файле и применении эвристик к значениям, найденным по смещениям от вхождений (чтобы убедится, что эта сигнатура действительно описывает процесс, а значения являются допустимыми значениями соответствующих параметров). При таком подходе есть моменты, за которыми необходимо следить с течением времени:

1. Необходимо проверять и исправлять смещения, по которым находятся значения параметров. Более того, для разных версий операционных систем они различны — нужно знать, какой ОС соответствует данный файл дампа памяти.

2. Набор эвристик также нуждается в постоянном мониторинге, их изменение более вероятно с выходом даже небольшого обновления для операционной системы.

3. Может поменяться способ хранения некоторых параметров, и если они участвовали в вычислений каких-то значений внутри программы, их необходимо пересмотреть. Например, структура VAD-дерева, хранимая в EPROCESS, поменялась с выходом Windows 2000.

Эти пункты были исправлены в ходе выполнения работы. Для проверки полученных значений использовались две утилиты: Volatility Framework и Windbg.

2.2.1 Volatility Framework

В 2007 году на конференции Black Hat был представлен набор утилит volatools, написанных на Python, предназначенных для анализа памяти и основанных на материалах публичных академических исследований. Позднее он был дополнен и перевыпущен под названием Volatility Framework. На сегодняшний день проект поддерживается компанией Volatility Foundation и активно используется как в академических исследованиях, так и в цифровых расследованиях.

Программа работает с образами памяти многих операционных систем (Windows XP и далее, Server с модели 2003; Mac OSX версии 10.5.x и далее, Linux с версиями ядра от 2.6.11 до 3.5 и др.), предоставляя возможность извлечь информацию о запущенных процессах, сетевых соединениях, открытых файлах и записей реестра и пр.

Утилита анализирует файлы образа памяти, полученные на предыдущем шаге (файлы .mem).

2.2.1 Windbg

Отладчик для операционных систем Windows, выпущенный компанией Microsoft. Позволяет отлаживать пользовательские приложения, драйвера и саму операционную систему в режиме ядра. Поддерживает возможность автоматической загрузке символьных файлов с серверов (PDB файлов), позволяющих ставить в соответствие машинным кодам исполняемых файлов конструкции языка программирования. Иногда разработчики включают такие файлы вместе с программой на диск.

В последнее время Windbg распространяется в пакете “Debugging Tools for Windows”, включающем и другие средства отладки (KD, CDB, NSTD), предназначенные для разных целей, но использующие схожий back-end.

Windbg широко применяется при отладке дампов памяти режима ядра, созданных операционной системой при появлении синих экранов смерти (BSoD), когда нужно выяснить причину остановки системы. Можно воспользоваться менее информативным приложением dumpchk для получения кода ошибки и ее возможного источника, однако стоит использовать windbg, если, например, необходимо понять, какая инструкция приводит к ошибке или увидеть стек вызовов режима ядра.

Для нашей задачи необходимо проанализировать структуры процессов, имеющиеся в памяти. Так как с помощью этой утилиты можно не только проверить актуальные смещения, но и получить значения параметров, нужно получить дамп памяти в какой-то момент времени и потом его исследовать, подключения к работающей системе будет недостаточно. Для этого была активирована возможность аварийно завершать работу системы с помощью клавиатуры, подключенной по USB (или PS/2 по порту i8042ptr), по комбинации клавиш “Right Control” + “Scroll Lock” (это можно сделать добавлением в реестр флага CrashOnCtrlScroll).В таком случае система сохранит файлы дампа памяти (если эта опция включена) с кодом ошибки 0xE2 (MANUALLY_INITIATED_CRASH)[6].

2.3 Поиск ключа и расшифровка образа

Поиск ключа осуществлялся с помощью уже реализованного алгоритма поиска подстроки в строке — алгоритма Карпа-Рабина (этим способом находятся и сигнатуры EPROCESS). После нахождения ключа в памяти нужно применить его для расшифровки образа диска. Существует библиотека, которая решает эту задачу — LibBDE (Library of Bitlocker Drive Encryption). Она используется в ходе этой работы.

2.3.1 Алгоритм Карпа-Рабина

Алгоритм, разработанный в 1987 году Ричардом Карпом и Майклом Рабином, решает задачу поиска вхождений шаблонов в исходный текст с помощью хеширования. В данной задаче одновременно будет проводиться поиск одного шаблона в тексте, причем размер шаблона много меньше размера исходных данных. Сложность работы алгоритма в среднем — O(n), в худшем случае — O(nm), где n — размер шаблона, m — размер текста.

2.3.2 LibBDE

Библиотека была создана в 2011 году Иоахимом Метцем, экспертом в области цифровых расследований, создавшем также libewf, liblnk и др.

Библиотека поддерживает все форматы дисков, шифруемых Bitlocker: для систем Windows Vista, Windows 7, Windows 8, а также Bitlocker To Go. Диски каждой из этих файловых систем описываются по-разному в заголовке тома (отличаются сигнатуры, указанные там), одна из задач — нахождение этой сигнатуры как точки входа диска.

С помощью этого решения носитель может быть расшифрован каждым из ключей: FVEK + TWEAK, VMK в виде чистого ключа, паролем восстановления, одним из внешних ключей или пользовательским паролем. В данной работе важен первый способ (так как именно эти ключи искались), однако диск может быть расшифрован и по-другому.

3 Описание решения

3.1 Анализ структур EPROCESS

Перед обработкой образа памяти в зависимости от вида операционной системы выбираются значения, соответствующие сигнатурам структур EPROCESS — они ищутся в памяти — и смещения, по которым хранятся значения параметров. Рассмотрим их в программе Windbg.

3.3.1 Смещения

Структуры EPROCESS и KRPOCESS легко отображаются в Windbg (командами вида dt !_structure virtAddress)[7]:

```
0: kd> dt !_eprocess fffffa80071c4060
nt!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_I
   +0x168 CreateTime       : _LARGE_INT
   +0x170 ExitTime         : _LARGE_INT
   +0x178 RundownProtect   : _EX_RUNDOW
   +0x180 UniqueProcessId  : 0x00000000
   +0x188 ActiveProcessLinks : _LIST_EN
   +0x198 ProcessQuotaUsage : [2] 0x654
   +0x1a8 ProcessQuotaPeak : [2] 0x6798
   +0x1b8 CommitCharge     : 0xb4e
   +0x1c0 QuotaBlock       : 0xfffffa80
   +0x1c8 CpuQuotaBlock    : (null)
   +0x1d0 PeakVirtualSize  : 0x611e000
   +0x1d8 VirtualSize      : 0x5931000
   +0x1e0 SessionProcessLinks : _LIST_E
   +0x1f0 DebugPort        : (null)
   +0x1f8 ExceptionPortData : 0xfffffa8
   +0x1f8 ExceptionPortValue : 0xfffffa
   +0x1f8 ExceptionPortState : 0y000
   +0x200 ObjectTable      : 0xfffff8a0
   +0x208 Token            : _EX_FAST_R
```

```
0: kd> dt !_kprocess fffffa80071c4060
nt!_KPROCESS
   +0x000 Header           : _DISPATCHE
   +0x018 ProfileListHead  : _LIST_ENTR
   +0x028 DirectoryTableBase : 0x000000
   +0x030 ThreadListHead   : _LIST_ENTR
   +0x040 ProcessLock      : 0
   +0x048 Affinity         : _KAFFINITY
   +0x070 ReadyListHead    : _LIST_ENTR
   +0x080 SwapListEntry    : _SINGLE_LI
   +0x088 ActiveProcessors : _KAFFINITY
   +0x0b0 AutoAlignment    : 0y0
   +0x0b0 DisableBoost     : 0y0
   +0x0b0 DisableQuantum   : 0y0
   +0x0b0 ActiveGroupsMask : 0y0001
   +0x0b0 ReservedFlags    : 0y00000000
   +0x0b0 ProcessFlags     : 0n8
   +0x0b4 BasePriority     : 8 ''
   +0x0b5 QuantumReset     : 6 ''
   +0x0b6 Visited          : 0 ''
   +0x0b7 Unused3          : 0 ''
   +0x0b8 ThreadSeed       : [4] 3
   +0x0c8 IdealNode        : [4] 0
```

Для каждого параметра есть смещение, по которому он хранится, название и значение. Такими сравнениями были проверены значения смещений параметров, интересующих нас, отдельно для операционных систем Windows разных разрядностей.

3.1.2 Эвристики

В ходе работы с образами памяти выяснилось, что часть предположений о значениях параметров потеряли свою актуальность. Так, параметр Directory Table Base не обязательно должен быть выровненным по значению 0x1000. Значение виртуального адреса структуры не содержится в ней самой, но важно для анализа процесса. Способ вычислений с помощью списков активных процессов содержал ошибку, добавлен вариант вычислений через VAD-дерево[8][9].

3.2 Поиск и работа с ключом

3.2.1 Поиск ключа

Сигнатура ключа, по которой осуществляется поиск, одинаковая для всех операционных систем и видов шифрования — “FVEc”. Одинаковы и смещения, по которым ищутся FVEK и TWEAK относительно этой сигнатуры. Код шифрующего алгоритма (который также находится по одному смещению) указывает, был ли носитель зашифрован с применением диффузора, какой длины ключ и блоки:

0x8000	AES-CBC 128-bit encryption with Elephant Diffuser
0x8001	AES-CBC 256-bit encryption with Elephant Diffuser
0x8002	AES-CBC 128-bit encryption
0x8003	AES-CBC 256-bit encryption

Другие значения, которые может принимать это поле (0x2000 – 0x2005) обозначают AES-CCM 256-битный алгоритм, который используется для шифрования ключей (VMK с помощью Recovery key, External Key и т.п.) и поэтому не рассматриваются в этой работе. Дополнительно проверяется, что ключи для шифрования и дешифрования (которые также должны храниться с этой структуре данных), совпадают и для FVEK и для TWEAK.

Наборы данных, удовлетворяющие этим условиям, могут быть интерпретированы как ключи к каким-то носителям информации. Если их несколько, верную копию следует выбрать, интерпретируя расшифрованные данные.

3.2.2 Расшифровка образа

После нахождения ключа необходимо передать его структурам libbde, предоставляющим интерфейс для чтения/записи данных. Поиск входной точки (entry point, которая зависит от модели устройства) можно было бы осуществить алгоритмом поиска Карпа-Рабина, но так как длина сигнатуры небольшая (6 байт) и нужная копия должна быть выровнена по 0x1000, лучше последовательно проверить каждое подходящее смещение в небольшом диапазоне. Тогда точка входа будет найдена и библиотека сможет считать с тома три блока метаданных. Получив FVEK и TWEAK (или один из других дополнительных ключей), она выполнить расшифровку тома.

4 Тестирование

Работа алгоритма тестировалась на зашифрованном USB носителе “Verbatim” объемом 8 гигабайт, ключ искался в образе оперативной памяти операционной системы Windows 7 SP1, объемом 4 гигабайта. Объем буффера для чтения данных — 128 килобайт. Время работы:

1. Поиск процессов в памяти — 1,25 минут
2. Поиск ключа в памяти — 1,25 минут
3. Последовательное чтение данных с образа диска (без записи) — 22 минуты

5 Заключение

Рассмотренное в ходе работы решение расшифровки внешнего носителя информации обладает определенными преимуществами и недостатками. С одной стороны, алгоритм зависим от архитектурных особенностей операционной системы и постоянно нуждается в тестировании и расширении (при выпуске новых ОС с поддержкой технологии Bitlocker To Go), что было выполнено в ходе работы. Но с другой стороны реализованный алгоритм решает поставленную задачу за приемлемое время, без проведения сложных вычислений, которые потребовались бы при подборе ключа.

Дальнейшее развитие решения состоит в исследовании области хранения ключа в терминах виртуальной памяти — можно ли считать, что он хранится в памяти определенного процесса, если да, то какого? Возможно, FVEK все время хранится в виртуальной памяти процесса System.exe. Такое предположение в случае его истинности позволяет улучшить алгоритм, сужая область поиска. Но вместе с этим возникают и другие вопросы, касательно фрагментации структуры ключа и др.

6 Список литературы

1. N. Ferguson, “AES-CBC + Elephant diffuser. A Disk Encryption Algorithm for Windows Vista”, 2006

2. P. Shabana Subair, C. Balan, S. Dija, K. L. Thomas, ”Forensic Decryption of FAT BitLocker Volumes”, Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2014

3. J. Metz, “BitLocker Drive Encryption format specification”, URL: https://github.com/libyal/libbde/blob/master/documentation/BitLocker%20Drive%20Encryption%20(BDE)%20format.asciidoc#bitlocker-drive-encryption-bde-format-specification

4. М. Руссинович, Д. Соломон, А. Ионеску, “Внутреннее устройство Microsoft Windows. 6-е изд. Основные подсистемы ОС.”, Питер, 2014

5. А. Овчинников, “Восстановление адресного пространства процесса из расширенного образа памяти на платформе Windows”, СПбГУ, 2012

6. “Forcing a System Crash from the Keyboard”, URL: https://msdn.microsoft.com/en-us/library/ff545499.aspx

7. “Translating Virtual To Physical Address On Windows: Physical Addresses”, URL: http://resources.infosecinstitute.com/translating-virtual-to-physical-address-on-windows-physical-addresses/

8. B. Dolan-Gavitt, “The VAD tree: A process-eye view of physical memory”, URL: http://dfrws.org/2007/proceedings/p62-dolan-gavitt.pdf

9. “Криминалистический анализ памяти: Исследуем процессы в Windows 7”, URL: https://xakep.ru/2011/02/14/54714/