МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Восстановление данных из дампа оперативной памяти ОС Windows

АВТОРЕФЕРАТ
дипломной работы

студента 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Сергеева Сергея Евгеньевича

Научный руководитель
доцент, к. ю. н. _____ А. В. Гортинский
19.01.2026 г.

Заведующий кафедрой
д. ф.-м. н., профессор _____ М. Б. Абросимов
19.01.2026 г.

Саратов 2026

ВВЕДЕНИЕ

В современных вычислительных системах Windows информация, находящаяся в оперативной памяти, существует временно и постоянно изменяется в процессе работы системы. Снятие дампа физической памяти позволяет зафиксировать содержимое оперативной памяти на определённый момент времени, что открывает возможности для анализа и восстановления данных, которые в обычных условиях остаются недоступными для пользователя.

Данная работа посвящена исследованию методов восстановления информации из дампа оперативной памяти операционной системы Windows. Основными целями исследования являются изучение механизмов управления виртуальной памятью в операционных системах семейства Windows, разработка механизма реконструкции виртуального адресного пространства отдельного процесса для 64-битных версий данных систем, а также применение разработанного механизма для восстановления данных из дампа оперативной памяти.

Дипломная работа состоит из введения, 2 разделов, заключения, списка использованных источников и 6 приложений. Общий объем работы – 68 страниц, из них 30 страниц – основное содержание, включая 16 рисунков и 0 таблиц, список использованных источников из 14 наименований.

КРАТКОЕ СОДЕРЖАНИЕ

В разделе 1 «Работа с оперативной памятью в ОС Windows» рассматриваются теоретические основы организации и управления оперативной памятью в операционных системах семейства Windows.

В подразделе 1.1 «Физическая и виртуальная память» вводятся основные понятия, связанные с организацией памяти в операционной системе Windows. Физическая память определяется как совокупность реально установленных в системе модулей оперативной памяти, доступ к которым осуществляется по физическим адресам. Виртуальная память рассматривается как программно-аппаратный механизм, обеспечивающий каждому процессу собственное изолированное линейное адресное пространство, абстрагированное от физического расположения данных в оперативной памяти. Также кратко описывается принцип постраничного отображения виртуальных адресов в физические и использование страничной подкачки для эффективного управления ресурсами памяти.

В подразделе 1.2 «Процессы и потоки» вводятся основные понятия, связанные с выполнением программ в операционной системе Windows. Процесс определяется как выполняемая программа, обладающая собственным закрытым виртуальным адресным пространством, набором системных ресурсов и уникальным идентификатором. Подчёркивается, что каждому процессу соответствует как минимум один поток выполнения. Поток рассматривается как минимальная единица исполнения, включающая контекст процессора, пользовательский и ядровый стеки, локальное хранилище потока и уникальный идентификатор.

Также рассматриваются внутренние структуры ядра Windows, используемые для описания процессов. Отмечается, что каждый процесс представлен структурой EPROCESS, содержащей сведения, необходимые для идентификации и управления процессом, а также вложенную структуру

KPROCESS, в которой хранится значение DTB (Directory Table Base), используемое при трансляции виртуальных адресов в физические. Описывается механизм связывания всех активных процессов в двусвязный список с помощью поля ActiveProcessLinks, который применяется операционной системой и используется при анализе дампов памяти для восстановления перечня активных процессов.

В подразделе 1.3 «Режимы доступа» рассматриваются режимы выполнения кода в операционной системе Windows. Пользовательский режим определяется как режим работы процессора, в котором выполняются прикладные программы и доступ к системной памяти и привилегированным инструкциям ограничен. Режим ядра рассматривается как привилегированный режим, обеспечивающий операционной системе полный доступ ко всей памяти и инструкциям процессора, что необходимо для выполнения системных служб и драйверов устройств. Описывается разделение виртуального адресного пространства на пользовательскую и системную области для 32- и 64-битных версий Windows. Также выделяется системный процесс System, использующий особое DTB ядра и предназначенный для выполнения потоков режима ядра.

В подразделе 1.4 рассматривается механизм постраничного преобразования виртуальных адресов в физические в операционной системе Windows. Для этого используется четырёхуровневая иерархия таблиц страниц: PML4T (Page Map Level 4 Table), PDPT (Page Directory Pointer Table), PDT (Page Directory Table) и PT (Page Table). Каждая запись таблицы указывает либо на следующую таблицу уровня, либо на физическую страницу памяти. DTB (Directory Table Base) процесса хранится в регистре CR3 и используется как указатель на PML4T.

Виртуальный адрес разбивается на индексные поля, соответствующие уровням таблиц, и смещение внутри страницы. Для стандартной страницы размером 4 КБ младшие 12 бит образуют смещение внутри страницы, для больших страниц (2 МБ и 1 ГБ) используется соответствующее смещение, а

обход нижних уровней таблиц пропускается при установленном бите PS (Page Size).

Иерархическая структура таблиц страниц позволяет адресовать до 256 ТБ виртуального пространства в 64-битных системах. Процесс трансляции включает логические операции маскирования и сдвига для выбора записей таблиц страниц и получения физического адреса. При отсутствии допустимой записи или нарушении каноничности виртуального адреса трансляция прерывается и генерируется исключение обращения к памяти, что обеспечивает корректность и безопасность работы процессов.

В разделе 2 «Восстановление данных из дампа памяти» рассматриваются методы анализа дампов оперативной памяти в операционной системе Windows. Особое внимание уделяется реализации механизма реконструкции виртуального адресного пространства отдельного процесса для 64-битных версий систем, позволяющего корректно сопоставлять виртуальные и физические адреса. Демонстрируется применение разработанного механизма для восстановления данных из дампа памяти Windows.

В подразделе 2.1 «Дамп памяти» рассматривается определение и назначение дампа оперативной памяти как побайтовой копии физической памяти системы на конкретный момент времени. Дамп сохраняет состояние RAM, включая данные пользовательских процессов, структуры ядра, таблицы страниц и системные буферы. Применение дампов актуально в цифровой криминалистике, анализе вредоносного ПО и расследовании компьютерных инцидентов, где требуется доступ к временным данным, недоступным при обычной работе системы.

Существует два метода получения дампа: аппаратный и программный. Аппаратный метод использует устройства с прямым доступом к памяти (DMA), например через FireWire, Thunderbolt или PCI-Express, обеспечивая минимальное вмешательство в работу ОС. Программный метод выполняется средствами операционной системы в режиме ядра, например с помощью FTK

Imager, но может влиять на состояние памяти и потенциально искажать данные.

Для анализа дампов применяются специализированные инструменты. Среди них наиболее известен Volatility — фреймворк на Python с модульной архитектурой, поддерживающий различные ОС и форматы дампов, позволяющий восстанавливать информацию о процессах, потоках, файлах, сетевых соединениях и регистрах процессора. Популярны команды pslist, filescan и netscan. Также существует Rekall Memory Forensic Framework, форк Volatility, ориентированный на повышение модульности и скорости анализа, извлекающий аналогичные данные о процессах, потоках и системных структурах.

В подразделе 2.2 ставится проблема анализа аналитически значимых данных в дампах памяти операционной системы Windows. Данные могут быть разделены на отдельные физические страницы и размещены в произвольных участках дампа, что нарушает их исходную логическую целостность. Для корректного анализа требуется восстановить взаимное расположение этих данных.

Для решения этой проблемы реализуется программное обеспечение, которое позволяет автоматически реконструировать виртуальное адресное пространство процесса, восстанавливая соответствие между виртуальными и физическими адресами, определяя принадлежность данных конкретному процессу и упрощая последующий анализ. Детали реализации рассматриваются в последующих подразделах.

В подразделе 2.3 «Перечень активных процессов» рассматривается механизм формирования перечня активных процессов системы на момент снятия дампа памяти, поскольку на первоначальном этапе необходимо идентифицировать процесс, виртуальное адресное пространство которого требуется восстановить.

В подразделе 2.3.1 «Поиск системного процесса» описывается разработанная программная реализация, предназначенная для

автоматизированного обнаружения системного процесса System в дампе оперативной памяти. Программа выполняет сканирование дампа с использованием известных смещений полей структуры EPROCESS и характерных признаков системного процесса, что позволяет выявить корректный экземпляр структуры, соответствующий процессу System. Найденный системный процесс используется как исходная точка для дальнейшего формирования перечня активных процессов.

В подразделе 2.3.2 «Получение DTB ядра» рассматривается разработанная программная реализация, предназначенная для автоматического получения значения DTB ядра из дампа физической памяти. Описывается принцип работы программы, основанный на сканировании физических страниц памяти и анализе структуры таблиц страниц с применением набора эвристик и фильтров. Реализованный подход позволяет надёжно определить корректное значение DTB ядра и используется в дальнейшем для трансляции виртуальных адресов внутри структур EPROCESS.

В подразделе 2.3.3 «Получение активных процессов через системный процесс» рассматривается программная реализация, позволяющая сформировать перечень активных процессов из дампа памяти Windows. Разработанная программа pslist.py использует физический адрес структуры EPROCESS системного процесса System и DTB ядра для получения PID, PPID, имени процесса и физического адреса структуры EPROCESS каждого активного процесса.

Алгоритм выполняет обход двусвязного списка ActiveProcessLinks, начиная с System EPROCESS, и преобразует виртуальные адреса указателей Flink и Blink в физические с помощью DTB ядра и иерархии таблиц страниц Windows x64. Для предотвращения повторной обработки используется множество посещённых адресов. Таким образом программа обеспечивает полную и корректную информацию об активных процессах, необходимую для дальнейшей реконструкции виртуального адресного пространства.

В подразделе 2.4 «Получение виртуального пространства процесса по DTB» описывается алгоритм реконструкции виртуального адресного пространства процесса, реализованный в программе mapping.py на языке Python. Алгоритм использует DTB процесса и выполняет постраничный обход виртуального адресного пространства с последующим преобразованием виртуальных адресов в физические на основе иерархии таблиц страниц Windows x64.

Виртуальное адресное пространство 64-битных систем Windows разделено на пользовательскую и системную области. В рамках практического анализа основное внимание уделяется нижней пользовательской области, содержащей данные процесса. Для ускорения работы и сокращения объёма обрабатываемых данных в реализации предусмотрен обход только данной части адресного пространства.

Поскольку полный побайтовый перебор виртуальных адресов нецелесообразен, используется постраничный обход с учётом размеров страниц (4 КБ, 2 МБ и 1 ГБ). При обнаружении отсутствующих или некорректных записей таблиц страниц соответствующие диапазоны виртуальных адресов пропускаются целиком, что позволяет эффективно обходить большие неотображённые области памяти. Дополнительно в алгоритме реализовано кэширование таблиц страниц, снижающее количество обращений к дампу физической памяти и повышающее общую производительность реконструкции.

В подразделе 2.5 «Применение реконструкции виртуального адресного пространства для восстановления данных из дампа» демонстрируется использование разработанных программ для восстановления данных из дампа памяти Windows. Для проверки корректности работы механизма решается задача восстановления секретного текста, введённого в программу text_hasher.exe, который существует исключительно в оперативной памяти.

Для этого последовательно используются разработанные инструменты:

1) dtb_finder.py — получение DTB ядра из дампа;

2) system_process_finder.py — определение физического адреса системного процесса System;

3) pslist.py — формирование перечня активных процессов и выбор целевого процесса text_hasher.exe;

4) mapping.py — реконструкция виртуального адресного пространства выбранного процесса и формирование дампа proc1.dmp.

После реконструкции виртуального адресного пространства поиск секретного текста значительно упрощается: данные представлены в корректном логическом порядке, а анализ ограничен памятью конкретного процесса. Для идентификации текста используется эталонный дамп процесса с заранее известным входом (proc2.dmp), который позволяет выделить характерные контекстные признаки структуры, содержащей текст.

Далее программа dump_word_search на Rust выполняет поиск аналогичных участков в proc1.dmp и определяет смещение секретного текста. Извлечённый текст проверяется через text_hasher.exe на совпадение хэша с ожидаемым значением. Результат показал полное совпадение, что подтверждает успешное восстановление исходной информации.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено детальное изучение структуры оперативной памяти в операционных системах Windows. Были разработаны программные инструменты для анализа дампов памяти с использованием языков Python и Rust.

Для проверки корректности работы реализованных инструментов был продемонстрирован сценарий, в ходе которого использование этих инструментов позволило автоматизировать восстановление секретных данных из дампа. По результатам сверки восстановленных данных с эталонным хэш-значением было подтверждено, что разработанные программы отработали успешно, обеспечивая точное восстановление исходной информации.