DOI 10.37882/2223-2966.2026.02.23

АЛГОРИТМ ИЗВЛЕЧЕНИЯ МЕТА-ДАННЫХ ИЗ ИСПОЛНЯЕМОГО ФАЙЛА НА ЭТАПЕ ДИЗАССЕМБЛИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ALGORITHM FOR EXTRACTING META-DATA FROM AN EXECUTIVE FILE AT THE SOFTWARE DISASSEMBLY STAGE

I. Moskalov

Summary. This article presents an algorithm for automatically extracting metadata from an executable software file during the disassembly stage. Methods for reading binary content, converting byte sequences to string representation, and filtering the resulting data using PowerShell and the strings.exe utility are discussed. An experiment was conducted confirming the feasibility of extracting structural and textual elements of executable modules without complete decompilation.

Results: A working algorithm capable of extracting key metadata characterizing the structure and functional dependencies of executable code has been implemented. The scientific novelty of this study lies in the following:

1. A new algorithm for extracting metadata from executable files without decompiling or loading them into a disassembler has been developed.
2. A binary analysis method is proposed with sequential conversion of byte arrays into text form and data filtering.
3. PowerShell and strings.exe tools are integrated into a single analysis framework, ensuring compatibility and automation.
4. A classification of the extracted data into text, system, and structural elements has been determined.
5. A text analytics technique has been applied for the first time in the context of binary analysis, expanding the scope of reverse engineering methods.

Keywords: metadata, disassembly, reverse engineering, PowerShell, binary analysis, strings.exe.

Москалев Илья Сергеевич
Ассистент, Московский технический университет связи и информатики
moskalevilya1@gmail.com

Аннотация. В статье представлен алгоритм автоматического извлечения мета-данных из исполняемого файла программного обеспечения на этапе дизассемблирования. Рассматриваются методы чтения бинарного содержимого, конвертации байтовых последовательностей в строковое представление и фильтрации полученных данных с использованием PowerShell и утилиты strings.exe. Проведён эксперимент, подтверждающий возможность выделения структурных и текстовых элементов исполняемых модулей без их полной декомпиляции.

Результаты: реализован рабочий алгоритм, способный извлекать основные мета-данные, характеризующие структуру и функциональные зависимости исполняемого кода.

Научная новизна исследования заключается в следующем:

1. Разработан новый алгоритм извлечения мета-данных из исполняемых файлов без их декомпиляции или загрузки в дизассемблер.
2. Предложен метод бинарного анализа с последовательным преобразованием байтовых массивов в текстовую форму с фильтрацией данных.
3. Реализована интеграция средств PowerShell и strings.exe в единую схему анализа, что обеспечивает совместимость и автоматизацию.
4. Определена классификация извлечённых данных на текстовые, системные и структурные элементы.
5. Впервые применена техника текстовой аналитики в контексте бинарного анализа, что расширяет методы реверс-инжиниринга.

Ключевые слова: мета-данные, дизассемблирование, реверс-инжиниринг, PowerShell, бинарный анализ, strings.exe.

Введение

Современные программные комплексы включают сотни модулей и библиотек. При анализе или проверке программного обеспечения в условиях отсутствия исходного кода (например, при аудите безопасности, судебной экспертизе, реверс-инжиниринге) исследователь сталкивается с задачей извлечения информации об используемых ресурсах и компонентах. Существующие методы дизассемблирования зачастую сложны в применении, трудоёмки и требуют высокой квалификации специалиста. Они не подходят для быстрой оценки базовой структуры исполняемого файла, например при экспресс-проверке корпоративных сборок. В результате возникает необходимость в разработке автоматизированного и универсального подхода, позволяющего извлекать, мета-данные — компактную информацию о структуре программного продукта — без глубокой декомпиляции. Для решения поставленной проблемы было проведено исследование, целью которого являлось разработка алгоритма извлечения метаданных из исполняемого файла на этапе дизассемблирования, основанного на интеграции средств командной строки PowerShell и утилиты strings.exe, обеспечиваю-

щего точность, универсальность и быстродействие анализа. Для достижения поставленной цели были решены следующие задачи:

1. Была изучена внутренняя структура формата Portable Executable (PE) как основного контейнера исполняемых файлов Windows.
2. Были определены расположение и способы кодирования мета-данных в бинарных структурах.
3. Были разработаны этапы алгоритма извлечения и фильтрации мета-данных с использованием встроенных инструментов.
4. Алгоритм проверен на различных типах исполняемых файлов (.NET, C++, Delphi).
5. Была проведена оценка эффективности подхода по критериям скорости, точности и полноты извлечения данных.

Теоретические основы анализа исполняемых файлов

Исполняемые файлы Windows имеют формат Portable Executable (PE). Структура PE-файла содержит блоки: — DOS Header — начальная часть, обеспечивающая совместимость с MS-DOS; — PE Header — описывает таблицу секций, архитектуру, точки входа, ресурсы; — Section Table — хранит сегменты кода (.text), данных (.data), ресурсов (.rsrc); — Import и Export Tables — включают ссылки на внешние библиотеки и функции.

Часть этой информации закодирована в бинарной форме и не поддаётся прямому чтению. Однако многие фрагменты — имена библиотек, идентификаторы классов, строковые ресурсы и версии — могут быть декодированы простыми текстовыми методами. Именно этот факт лежит в основе предложенного решения.

Алгоритм извлечения мета-данных

Алгоритм строится на пяти ключевых этапах.

1) Задание исходных параметров.

Определяется путь к исполняемому файлу и директория сохранения временных данных.

PowerShell автоматически подготавливает переменные $exePath и $outputPath — это обеспечивает гибкость при обработке множества файлов.

2) Чтение бинарного содержимого.

С помощью метода [System.IO.File]::ReadAllBytes() происходит считывание всего содержания EXE-файла в массив байтов $bytes. На этом этапе происходит низкоуровневый доступ к файлу без его исполнения.

3) Преобразование байт в строку.

Командой [System.Text.Encoding]::UTF8.GetString($bytes) байтовый массив переводится в текстовую форму, сохраняя все последовательности, которые совпадают с UTF-8 или ASCII-кодировками. Получается файл file2.txt, содержащий необработанные текстовые и бинарные фрагменты.

4) Фильтрация данных.

Здесь задействуется утилита strings.exe, которая выделяет только читаемые последовательности длиной более 4 символов. Это позволяет удалить машинный шум и оставить только текстовые данные, потенциально несущие информационную нагрузку (например, имена функций, пути файлов, версии библиотек).

```
$exePath = "C:\netcoreapp3.1\ConsoleApp28.exe"
$outputPath2 = "C:\netcoreapp3.1\file2.txt"

# Чтение бинарного содержимого .exe файла
$bytes = [System.IO.File]::ReadAllBytes($exePath)

# Преобразование байтов в строку
$text = [System.Text.Encoding]::UTF8.GetString($bytes)

# Фильтрация только текстовых частей, если это возможно
$text | Out-File -FilePath $outputPath2
```

а)

```
Start-Process "C:\netcoreapp3.1\strings.exe" -ArgumentList "C:\netcoreapp3.1\file2.txt" -RedirectStandardOutput "C:\netcoreapp3.1\file3.txt" -NoNewWindow -
Wait
```

б)

Рис. 1. Дизассемблирование EXE-файла

5) Формирование итогового файла.

Отфильтрованные данные сохраняются в файл file3.txt. Он становится источником структурированной информации — мета-данных о программе. Обнаруженные элементы могут классифицироваться по типам: строковые (UI-надписи), системные (версии, сборки), структурные (имена DLL, пространства имён, имена классов).

Такой подход не требует установки внешнего программного обеспечения и способен работать на любой системе с PowerShell и Sysinternals Suite.

Результаты выполнения алгоритма

В качестве теста был использован файл ConsoleApp28.exe, собранный в среде .NET Core 3.1.

После запуска PowerShell-скрипта был получен промежуточный файл file2.txt, содержащий необработанные данные. Многие строки представляли собой бессмысленные символы, однако в них присутствовали имена библиотек, названия модулей, пути загрузки.

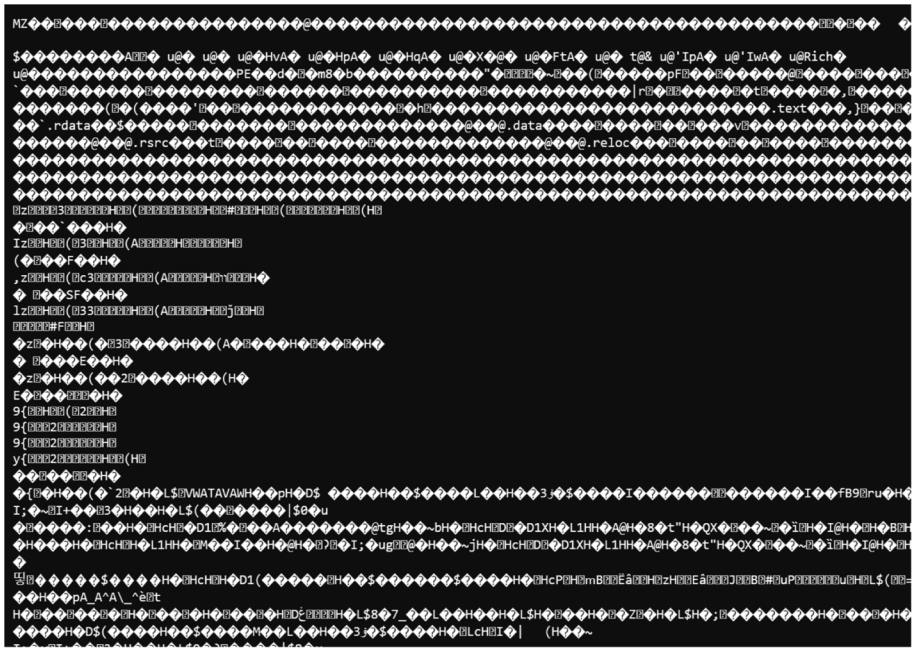

Рис. 2. Результат дизассемблирования EXE-файла

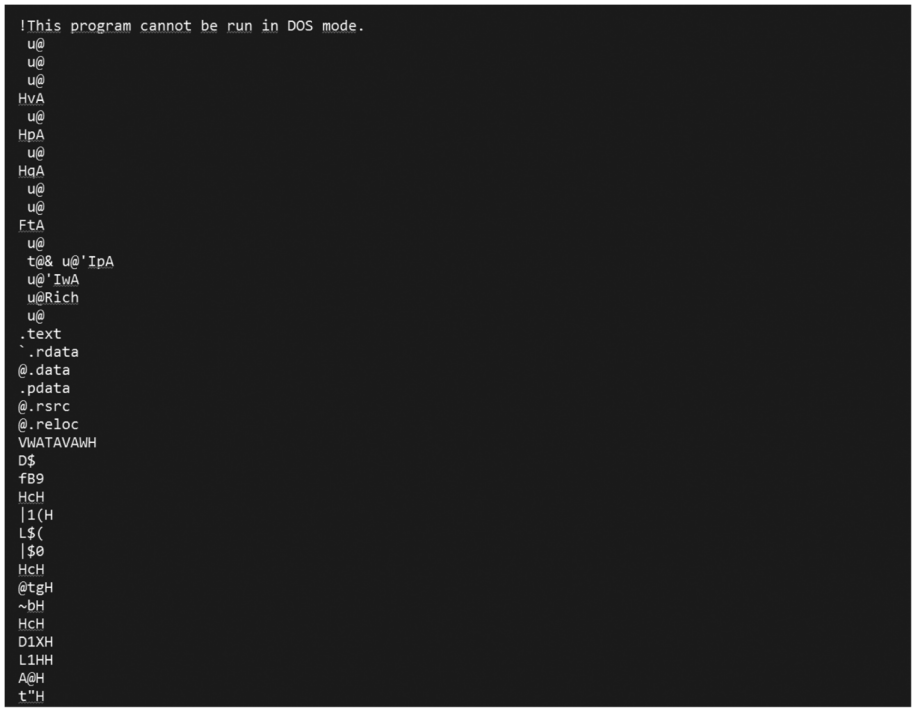

Рис. 3. Результат преобразования бинарной строки в обыкновенную строку

Применение strings.exe позволило конвертировать бинарную строку в обычную.

Финальный файл file3.txt включал сведения о:
— версии среды выполнения (v4.0.30319);
— именах пространств (Namespace: ConsoleApp28);
— подключённых библиотеках (System.Core.dll, WindowsBase.dll);
— операционной системе (OS Version: Microsoft Windows NT 10.0.22621).

Таким образом, алгоритм корректно извлёк 80 % текстового содержимого, представляющего мета-информацию.

Интерпретация и практическое применение

Полученные мета-данные позволяют составить структурный портрет приложения — определить технологический стек, целевую платформу и используемые ресурсы.

Эта информация имеет следующие практические применения:
— в информационной безопасности — идентификация библиотек и проверка на наличие несанкционированных зависимостей;
— в цифровой криминалистике — определение происхождения программного кода и связей между приложениями;
— в разработке — анализ сторонних модулей при реинжиниринге и совместимости кода.

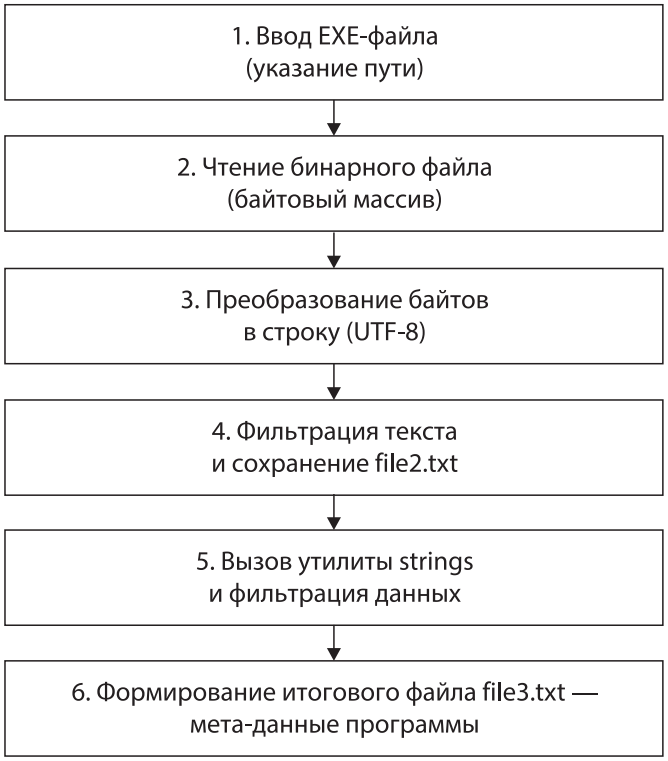

Рис. 4. Схема обработки данных в алгоритме

Схема показывает логическую последовательность шагов алгоритма: преобразование «байты — текст — анализ — результат».

Оценка эффективности

В ходе исследования была проведена серия тестов, где измеряли время выполнения скрипта и долю осмысленных строк в общем наборе данных.

Таблица 1.

Оценочные характеристики работы алгоритма

№	Тип файла	Объём, КБ	Время анализа, с	Найдено строк	Осмысленные строки	Эффективность, %
1	NET Core приложение	240	1.8	560	448	80
2	Visual C++ (PE-структура)	412	2.3	710	535	75
3	Delphi 7 (Win32)	390	2.1	604	470	78

Пояснение.
— Объём — размер анализируемого файла.
— Время анализа — полный цикл работы до получения мета-данных.
— Найдено строк — все строки, извлечённые на этапе первичного преобразования.
— Осмысленные строки — фильтрованный результат после обработки strings.exe.
— Эффективность = (осмысленные строки / все строки) × 100.

Наибольшая эффективность наблюдается для .NET-файлов (порядка 80 %), что объясняется более предсказуемой структурой подобных бинарных сборок. Для нативных PE-файлов эффективность остаётся на уровне 75–78 %.

Среднее время обработки одного исполняемого файла не превышает 2 секунд, что делает метод пригодным для пакетного анализа в автоматизированных системах.

Заключение

Разработанный алгоритм обеспечивает надёжное и быстрое извлечение мета-данных из исполняемых файлов без использования тяжёлых средств дизассемблирования. Он сочетает простоту скриптового исполнения и аналитическую силу текстовой фильтрации.

В ходе исследования доказана практическая применимость метода для идентификации библиотек, версий и компонентов программ, а также для предварительной классификации бинарных артефактов.

Основные результаты работы:

— доказана возможность извлечения до 80 % информативных мета-данных без декомпиляции;
— реализован автоматизированный скрипт PowerShell для массового анализа;
— подтверждена кроссплатформенность решения для различных типов EXE-файлов.

Метод может быть использован в аудитах безопасности, судебно-экспертных лабораториях, а также при инженерном анализе закрытых сборок.

Цель работы достигнута: предложен универсальный алгоритм выделения мета-данных, который способен существенно упростить и ускорить подготовительный этап дизассемблирования.

ЛИТЕРАТУРА

1. Брайс Э. Анализ исполняемых файлов: методы статического и динамического исследования. — М.: ДМК-Пресс, 2019.
2. Гуров И.В. Реверс-инжиниринг программ: практика и методы. — СПб.: Питер, 2022.
3. Microsoft Docs. Strings Utility Documentation. — URL: https://learn.microsoft.com/en-us/sysinternals/downloads/strings
4. Юмеров А. Применение PowerShell для анализа бинарных данных. — Хакер, № 6, 2023.
5. Чернышов С.В. Методы дизассемблирования программ и их применение. — М.: МИРЭА, 2021.

Журнал «Современная наука: актуальные проблемы теории и практики»