УДК 004.045

И. С. МОСКАЛЁВ

АЛГОРИТМ ИЗВЛЕЧЕНИЯ МЕТАДАННЫХ ИЗ ИСПОЛНЯЕМОГО ФАЙЛА НА ЭТАПЕ ДИЗАССЕМБЛИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Московский технический университет связи и информатики
г. Москва, Российская Федерация

Аннотация. *В статье представлен алгоритм автоматического извлечения метаданных из исполняемого файла программного обеспечения на этапе дизассемблирования. Рассмотрены методы чтения бинарного содержимого, конвертации байтовых последовательностей в строковое представление и фильтрации полученных данных с использованием PowerShell и утилиты strings.exe. Проведен эксперимент, подтверждающий возможность выделения структурных и текстовых элементов исполняемых модулей без их полной декомпиляции. Цель работы – повышение эффективности анализа бинарных приложений за счет выделения внутренних структурных данных на ранней стадии исследования. Методы исследования: статический анализ, реверс-инжиниринг, текстовая аналитика, PowerShell-скриптинг, использование средств командной строки. Результаты: реализован рабочий алгоритм, способный извлекать основные метаданные, характеризующие структуру и функциональные зависимости исполняемого кода. Научная новизна исследования состоит в: разработке нового алгоритма извлечения метаданных из исполняемых файлов без их декомпиляции или загрузки в дизассемблер; предложении метода бинарного анализа с последовательным преобразованием байтовых массивов в текстовую форму с фильтрацией данных; реализации интеграции средств PowerShell и strings.exe в единую схему анализа, что обеспечивает совместимость и автоматизацию; определении классификации извлекаемых данных на текстовые, системные и структурные элементы; применении техники текстовой аналитики в контексте бинарного анализа, что расширяет методы реверс-инжиниринга. Полученные результаты могут быть практически применены в сфере 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).*

Ключевые слова: *метаданные, 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 [5]. Существующие методы дизассемблирования зачастую сложны в применении, трудоемки 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 метаданные – компактную информацию о структуре программного продукта – без глубокой декомпиляции. Для решения поставленной проблемы было проведено исследование, целью которого являлась разработка алгоритма извлечения метаданных из исполняемого файла на этапе дизассемблирования, основанного на интеграции средств командной строки PowerShell и утилиты strings.exe, обеспечивающего точность, универсальность и быстродействие анализа. Для достижения поставленной цели были решены следующие задачи:

– изучена внутренняя структура формата PortableExecutable (PE) как основного контейнера исполняемых файлов Windows;

– определены расположение и способы кодирования метаданных в бинарных структурах;

– разработаны этапы алгоритма извлечения и фильтрации метаданных с использованием встроенных инструментов;

– алгоритм проверен на различных типах исполняемых файлов (.NET, C++, Delphi);

– проведена оценка эффективности подхода по критериям скорости, точности и полноты извлечения данных.

Основная часть

Теоретические основы анализа исполняемых файлов. Исполняемые файлы Windows имеют формат PortableExecutable (PE). Структура PE-файла содержит блоки:

– DOS Header – начальная часть, обеспечивающая совместимость с MS-DOS;

– PE Header – описывает таблицу секций, архитектуру, точки входа, ресурсы;

– SectionTable – хранит сегменты кода (.text), данных (.data), ресурсов (.rsrc);

– Import и ExportTables – включают ссылки на внешние библиотеки и функции [4].

Часть этой информации закодирована в бинарной форме и не поддается 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.

```
$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)

Рисунок 1. Дизассемблирование EXE-файла

Алгоритм извлечения метаданных. 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).

5) ***Формирование итогового файла.***

Отфильтрованные данные сохраняются в файл file3.txt. Он становится источником структурированной информации – метаданных о программе. Обнаруженные элементы могут классифицироваться по типам: строковые (UI-надписи), системные (версии, сборки), структурные (имена DLL, пространства имен, имена классов) [2, 3].

Такой подход не требует установки внешнего программного обеспечения и способен работать на любой системе с PowerShell и Sysinternals Suite. Его легко применить как к исполняемым файлам, так и другим, к которым есть возможность получить бинарный код с дальнейшим форматированием в формат слабоструктурированных данных для дальнейшего анализа данных.

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 кодированный текст, однако в них присутствовали имена библиотек, названия модулей, пути загрузки.

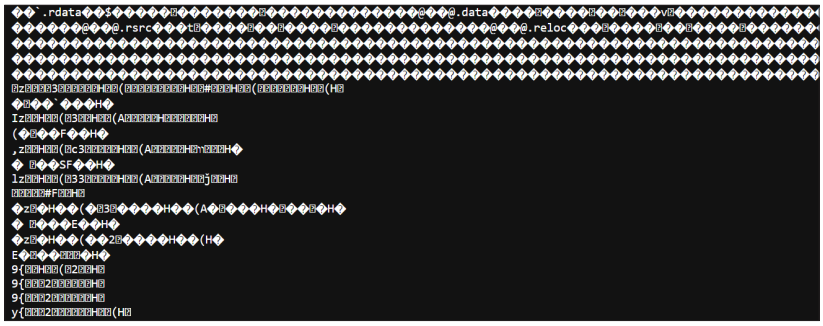

Рисунок 2. Результат дизассемблирования EXE-файла

Применение strings.exe позволило конвертировать бинарную строку в обычную.

```
 t@& u@'IpA
 u@'IwA
 u@Rich
 u@
.text
`.rdata
@.data
.pdata
@.rsrc
@.reloc
VWATAVAWH
D$
fB9
HcH
|1(H
L$(
|$0
HcH
@tgH
```

Рисунок 3. Результат преобразования бинарной строки в обыкновенную строку

Финальный файл file3.txt включал сведения о:

– версии среды выполнения (v4.0.30319);

– именах пространств (Namespace:ConsoleApp28);

– подключенных библиотеках (System.Core.dll, WindowsBase.dll);

– операционной системе (OS Version: Microsoft Windows NT 10.0.22621).

Таким образом, алгоритм корректно извлек 80 % текстового содержимого, представляющего метаинформацию [6].

Интерпретация и практическое применение. Полученные метаданные позволяют составить структурный портрет приложения – определить технологический стек, целевую платформу и используемые ресурсы. Эта информация имеет следующие возможности практического 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.

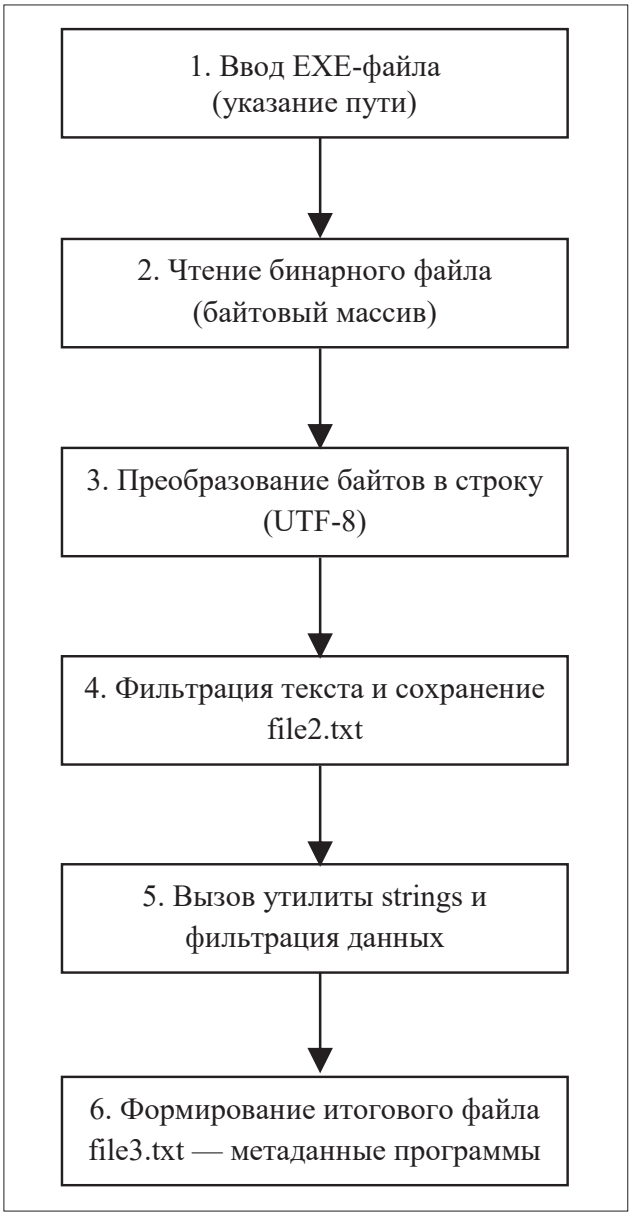

Рисунок 4. Схема обработки данных в алгоритме

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	Объем, КБ	Время анализа, с	Найдено строк	Осмысленные строки	Эффективность, %
.NET Core приложение	240	1.8	560	448	80
Visual C++ (PE-структура)	412	2.3	710	535	75
Delphi 7 (Win32)	390	2.1	604	470	78

Пояснение: Объем – размер анализируемого файла; Время анализа – полный цикл работы до получения метаданных; Найдено строк – все строки, извлеченные 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 остается на уровне 75–78 % [7].

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

Заключение

Разработанный алгоритм обеспечивает надежное и быстрое извлечение метаданных из исполняемых файлов без использования тяжелых 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 метаданных 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 метаданных, который способен существенно упростить и ускорить подготовительный этап дизассемблирования.

ЛИТЕРАТУРА

1. Баланов, А. Н. Комплексная информационная безопасность : полный справочник специалиста : практическое пособие / А. Н. Баланов. М. : Инфра-Инженерия. 2024. 154 с.

2. Барков, А. В. Влияние цифровизации на правовое обеспечение информационной безопасности государства и бизнеса в условиях современных геополитических вызовов / А. В. Барков, А. С. Киселев // Безопасность бизнеса. 2022. № 3. С. 3–7.

3. Барков, А. В. О правовом обеспечении безопасности информационно телекоммуникационной инфраструктуры банков и государственных структур / А. В. Барков, А. С. Киселев // Банковское право. 2022. № 4. С. 20–27.

4. Бондарев, В. В. Введение в информационную безопасность автоматизированных систем / В. В. Бондарев. 3-е изд. М. : Издательство МГТУ им. Н. Э. Баумана. 2021. 250 с.

5. Мельников, В. П. Информационная безопасность / В. П. Мельников, А. И. Куприянов, Т. Ю. Васильева ; под редакцией В. П. Мельникова. 2-е изд., переработанное и дополненное. М. : КноРус. 2020. 371 с.

6. Галыгина, Л. В. Социальные аспекты информационной безопасности / Л. В. Галыгина, И. В. Галыгина. М. : Лань. 2021. 64 с.

7. Гришина, Н. В. Основы информационной безопасности предприятия / Н. В. Гришина. М. : Инфра-М. 2021. 216 с.

REFERENCES

1. Balanov A.N. Kompleksnaya informatsionnaya bezopasnost. Polnyy spravochnik spetsialista. Prakticheskoye posobiye [Comprehensive Information Security. Complete reference book for specialists. Practical guide]. Moscow: Infra-Inzheneriya; 2024. 156 p. (in Russian).

2. Barkov A.V., Kiselev A.S. Vliyaniye tsifrovizatsii na pravovoye obespecheniye informatsionnoy bezopasnosti gosudarstva i biznesa v usloviyakh sovremennykh geopoliticheskikh vyzovov [The Impact of Digitalization on the Legal Regulation of the Information Security of State and Business in the Conditions of Modern Geopolitical Challenges]. Bezopasnost Biznesa. 2022;3:3–7 (in Russian).

3. Barkov A.V., Kiselev A.S. O pravovom obespechenii bezopasnosti informatsionno-telekommunikatsionnoy infrastruktury bankov i gosudarstvennykh struktur [On the Legal Support for the Security of Information and Telecommunication Infrastructure of Banks and Government Agencies]. Bankovskoye Pravo. 2022;4:20–27 (in Russian).

4. Bondarev V.V. Vvedeniye v informatsionnuyu bezopasnost avtomatizirovannykh sistem [Introduction to the Information Security of Automated Systems]. 3rd ed. Moscow: Izdatel'stvo MGTU im. N. E. Baumana; 2021. 250 p. (in Russian).

5. Melnikov V.P. Kupriyanov A.I., Vasilyeva T.Yu. Informatsionnaya bezopasnost [Information Security]. 2nd ed., revised and supplemented. Moscow: KnoRus; 2020. 371 p. (in Russian).

6. Galygina L.V., Galygina I.V. Sotsialnyye aspekty informatsionnoy bezopasnosti [Social Aspects of Information Security]. Moscow: Lan; 2021. 64 p. (in Russian).

7. Grishina N.V. Osnovy informatsionnoy bezopasnosti predpriyatiya [Fundamentals of Enterprise Information Security]. Moscow: Infra-M; 2021. 216 p. (in Russian).

I. S. MOSKALEV

ALGORITHM FOR EXTRACTING METADATA FROM AN EXECUTABLE FILE AT THE DISASSEMBLY STAGE OF SOFTWARE

Moscow Technical University of Communications and Informatics
Moscow, Russia

Abstract. *The article presents an algorithm for automatic extraction of metadata from executable software files at the disassembly stage. Methods for reading binary content, converting byte sequences into string representation, and filtering the obtained data using PowerShell and strings.exe utility are considered. An experiment was conducted confirming the possibility of extracting structural and textual elements of executable modules without their complete decompilation. The aim of this work is to improve the efficiency of analyzing binary applications by highlighting internal structural data at an early research stage. Methods used include static analysis, reverse engineering, text analytics, PowerShell scripting, and command-line tools. Results achieved include a working algorithm capable of extracting basic metadata characterizing the structure and functional dependencies of executable code. The scientific novelty of the research consists in the following points: A new algorithm has been developed for extracting metadata from executable files without decompiling or loading them into a disassembler; A method of binary analysis with sequential transformation of byte arrays into text form with data filtration has been proposed; Integration of PowerShell and strings.exe utilities into a unified analysis scheme ensuring compatibility and automation has been implemented; Classification of extracted data into textual, system, and structural elements has been defined; Textual analytics technique has been applied for the first time in the context of binary analysis, thus expanding reverse engineering methods.*

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

Москалёв Илья Сергеевич
Московский технический университет связи и информатики, Москва, Российская Федерация. Ассистент кафедры «Информатика».
ул. Народного Ополчения, 32, 123423, г. Москва, Россия.
I. Moskalev
Moscow Technical University of Communications and Informatics, Russia. Assistant of the Department of Informatics.
Narodnoe Opolchenie St., 32, 123423, Moscow, Russia
E-mail: i.s moskalev@mtuci.ru