УДК 004.451.5
ББК 32.371
Э64

Эндриесс Д.
Э64 Практический анализ двоичных файлов / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2022. – 460 с.: ил.

ISBN 978-5-97060-978-1

В книге представлено подробное описание методов и инструментов, необходимых для анализа двоичного кода, который позволяет убедиться, что откомпилированная программа работает так же, как исходная, написанная на языке высокого уровня.

Наряду с базовыми понятиями рассматриваются такие темы, как оснащение двоичной программы, динамический анализ заражения и символическое выполнение. В каждой главе приводится несколько примеров кода; к книге прилагается сконфигурированная виртуальная машина, включающая все примеры.

Руководство адресовано специалистам по безопасности и тестированию на проникновение, хакерам, аналитикам вредоносных программ и всем, кто интересуется вопросами защиты ПО.

УДК 004.451.5
ББК 32.371

Title of English-language original: Practical Binary Analysis: Build Your Own Linux Tools for Binary Instrumentation, Analysis, and Disassembly Reversing Modern Malware and Next Generation Threats, ISBN 9781593279127, published by No Starch Press Inc. 245 8th Street, San Francisco, California United States 94103. The Russian-Language 1st edition Copyright © 2021 by DMK Press Publishing under license by No Starch Press Inc. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

ISBN 978-1-59327-912-7 (англ.)
ISBN 978-5-97060-978-1 (рус.)

© Dennis Andriesse, 2021
© Перевод, оформление, издание, ДМК Пресс, 2022

СОДЕРЖАНИЕ

Вступительное слово ... 17

Предисловие ... 20

Благодарности ... 21

Введение ... 22

ЧАСТЬ I. ФОРМАТЫ ДВОИЧНЫХ ФАЙЛОВ

Глава 1. Анатомия двоичного файла ... 32

1.1 Процесс компиляции программы на C ... 33
- 1.1.1 Этап препроцессирования ... 33
- 1.1.2 Этап компиляции ... 35
- 1.1.3 Этап ассемблирования ... 37
- 1.1.4 Этап компоновки ... 38

1.2 Символы и зачищенные двоичные файлы ... 40
- 1.2.1 Просмотр информации о символах ... 40
- 1.2.2 Переход на темную сторону: зачистка двоичного файла ... 42

1.3 Дизассемблирование двоичного файла ... 42
- 1.3.1 Заглянем внутрь объектного файла ... 43
- 1.3.2 Изучение полного исполняемого двоичного файла ... 45

1.4 Загрузка и выполнение двоичного файла ... 48

1.5 Резюме ... 50

Глава 2. Формат ELF ... 52

2.1 Заголовок исполняемого файла ... 54
- 2.1.1 Массив e_ident ... 55
- 2.1.2 Поля e_type, e_machine и e_version ... 56

2.1.3 Поле e_entry 57
2.1.4 Поля e_phoff и e_shoff 57
2.1.5 Поле e_flags 57
2.1.6 Поле e_ehsize 58
2.1.7 Поля e_*entsize и e_*num 58
2.1.8 Поле e_shstrndx 58
2.2 Заголовки секций 59
2.2.1 Поле sh_name 60
2.2.2 Поле sh_type 60
2.2.3 Поле sh_flags 61
2.2.4 Поля sh_addr, sh_offset и sh_size 61
2.2.5 Поле sh_link 62
2.2.6 Поле sh_info 62
2.2.7 Поле sh_addralign 62
2.2.8 Поле sh_entsize 62
2.3 Секции 62
2.3.1 Секции .init и .fini 64
2.3.2 Секция .text 64
2.3.3 Секции .bss, .data и .rodata 66
2.3.4 Позднее связывание и секции .plt, .got, .got.plt 66
2.3.5 Секции .rel.* и .rela.* 70
2.3.6 Секция .dynamic 71
2.3.7 Секции .init_array и .fini_array 72
2.3.8 Секции .shstrtab, .symtab, .strtab, .dynsym и .dynstr 73
2.4 Заголовки программы 74
2.4.1 Поле p_type 75
2.4.2 Поле p_flags 76
2.4.3 Поля p_offset, p_vaddr, p_paddr, p_filesz и p_memsz 76
2.4.4 Поле p_align 76
2.5 Резюме 77

Глава 3. Формат PE: краткое введение 78
3.1 Заголовок MS-DOS и заглушка MS-DOS 79
3.2 Сигнатура PE, заголовок PE-файла и факультативный заголовок PE 79
3.2.1 Сигнатура PE 82
3.2.2 Заголовок PE-файла 82
3.2.3 Факультативный заголовок PE 83
3.3 Таблица заголовков секций 83
3.4 Секции 84
3.4.1 Секции .edata и .idata 85
3.4.2 Заполнение в секциях кода PE 86
3.5 Резюме 86

Глава 4. Создание двоичного загрузчика с применением libbfd 88
4.1 Что такое libbfd? 89
4.2 Простой интерфейс загрузки двоичных файлов 89

4.2.1 Класс Binary ... 92
4.2.2 Класс Section ... 92
4.2.3 Класс Symbol ... 92
4.3 Реализация загрузчика двоичных файлов ... 93
4.3.1 Инициализация libbfd и открытие двоичного файла ... 94
4.3.2 Разбор основных свойств двоичного файла ... 96
4.3.3 Загрузка символов ... 99
4.3.4 Загрузка секций ... 102
4.4 Тестирование загрузчика двоичных файлов ... 104
4.5 Резюме ... 106

ЧАСТЬ II. ОСНОВЫ АНАЛИЗА ДВОИЧНЫХ ФАЙЛОВ

Глава 5. Основы анализа двоичных файлов в Linux ... 109
5.1 Разрешение кризиса самоопределения с помощью file ... 110
5.2 Использование ldd для изучения зависимостей ... 113
5.3 Просмотр содержимого файла с помощью xxd ... 115
5.4 Разбор выделенного заголовка ELF с помощью readelf ... 117
5.5 Разбор символов с помощью nm ... 119
5.6 Поиск зацепок с помощью strings ... 122
5.7 Трассировка системных и библиотечных вызовов с помощью strace и ltrace ... 125
5.8 Изучение поведения на уровне команд с помощью objdump ... 129
5.9 Получение буфера динамической строки с помощью gdb ... 131
5.10 Резюме ... 134

Глава 6. Основы дизассемблирования и анализа двоичных файлов ... 135
6.1 Статическое дизассемблирование ... 136
6.1.1 Линейное дизассемблирование ... 136
6.1.2 Рекурсивное дизассемблирование ... 139
6.2 Динамическое дизассемблирование ... 142
6.2.1 Пример: трассировка выполнения двоичного файла в gdb ... 143
6.2.2 Стратегии покрытия кода ... 146
6.3 Структурирование дизассемблированного кода и данных ... 150
6.3.1 Структурирование кода ... 151
6.3.2 Структурирование данных ... 158
6.3.3 Декомпиляция ... 160
6.3.4 Промежуточные представления ... 162
6.4 Фундаментальные методы анализа ... 164
6.4.1 Свойства двоичного анализа ... 164
6.4.2 Анализ потока управления ... 169
6.4.3 Анализ потока данных ... 171
6.5 Влияние настроек компилятора на результат дизассемблирования ... 175
6.6 Резюме ... 177

ŋqdbɣeɎ bɆ dedɆbɎɜe d ɕɑɑbɕɣ ɟqddɎqɣè

Глава 7. Простые методы внедрения кода для формата ELF 178

7.1 Прямая модификация двоичного файла с помощью шестнадцатеричного редактирования 178
 7.1.1 Ошибка на единицу в действии 179
 7.1.2 Исправление ошибки на единицу 182
7.2 Модификация поведения разделяемой библиотеки с помощью LD_PRELOAD 186
 7.2.1 Уязвимость, вызванная переполнением кучи 186
 7.2.2 Обнаружение переполнения кучи 189
7.3 Внедрение секции кода 192
 7.3.1 Внедрение секции в ELF-файл: общий обзор 192
 7.3.2 Использование elfinject для внедрения секции в ELF-файл 195
7.4 Вызов внедренного кода 198
 7.4.1 Модификация точки входа 199
 7.4.2 Перехват конструкторов и деструкторов 202
 7.4.3 Перехват записей GOT 205
 7.4.4 Перехват записей PLT 208
 7.4.5 Перенаправление прямых и косвенных вызовов 209
7.5 Резюме 210

ЧАСТЬ III. ПРОДВИНУТЫЙ АНАЛИЗ ДВОИЧНЫХ ФАЙЛОВ

Глава 8. Настройка дизассемблирования 212

8.1 Зачем писать специальный проход дизассемблера? 213
 8.1.1 Пример специального дизассемблирования: обфусцированный код 213
 8.1.2 Другие причины для написания специального дизассемблера 216
8.2 Введение в Capstone 217
 8.2.1 Установка Capstone 218
 8.2.2 Линейное дизассемблирование с помощью Capstone 219
 8.2.3 Изучение Capstone C API 224
 8.2.4 Рекурсивное дизассемблирование с помощью Capstone 225
8.3 Реализация сканера ROP-гаджетов 234
 8.3.1 Введение в возвратно-ориентированное программирование 234
 8.3.2 Поиск ROP-гаджетов 236
8.4 Резюме 242

Глава 9. Оснащение двоичных файлов 244

9.1 Что такое оснащение двоичного файла? 244
 9.1.1 API оснащения двоичных файлов 245
 9.1.2 Статическое и динамическое оснащение двоичных файлов 246
9.2 Статическое оснащение двоичных файлов 248
 9.2.1 Подход на основе int 3 248
 9.2.2 Подход на основе трамплинов 250

ŋqdbɣeɎ bɆ dedɆbɎɜe d ɕɑɑbɕɣ ɟqddɎqɣè

ɳqdqyɐʎ bƐ dədƐbʎɜə d ɕɑɒɕɑʎ Ɉqdqʎqɐè

9.3 Динамическое оснащение двоичных файлов 255
9.3.1 Архитектура DBI-системы 255
9.3.2 Введение в Pin 257
9.4 Профилирование с помощью Pin 259
9.4.1 Структуры данных профилировщика и код инициализации 259
9.4.2 Разбор символов функций 262
9.4.3 Оснащение простых блоков 264
9.4.4 Оснащение команд управления потоком 266
9.4.5 Подсчет команд, передач управления и системных вызовов 269
9.4.6 Тестирование профилировщика 270
9.5 Автоматическая распаковка двоичного файла с помощью Pin 274
9.5.1 Введение в упаковщики исполняемых файлов 274
9.5.2 Структуры данных и код инициализации распаковщика 276
9.5.3 Оснащение команд записи в память 278
9.5.4 Оснащение команд управления потоком 280
9.5.5 Отслеживание операций записи в память 280
9.5.6 Обнаружение оригинальной точки входа и запись распакованного двоичного файла 281
9.5.7 Тестирование распаковщика 283
9.6 Резюме 287

Глава 10. Принципы динамического анализа заражения 289
10.1 Что такое DTA? 290
10.2 Три шага DTA: источники заражения, приемники заражения и распространение заражения 290
10.2.1 Определение источников заражения 291
10.2.2 Определение приемников заражения 291
10.2.3 Прослеживание распространения заражения 292
10.3 Использование DTA для обнаружения дефекта Heartbleed 292
10.3.1 Краткий обзор уязвимости Heartbleed 292
10.3.2 Обнаружение Heartbleed с помощью заражения 294
10.4 Факторы проектирования DTA: гранулярность, цвета политики заражения 296
10.4.1 Гранулярность заражения 296
10.4.2 Цвета заражения 297
10.4.3 Политики распространения заражения 298
10.4.4 Сверхзаражение и недозаражение 300
10.4.5 Зависимости по управлению 300
10.4.6 Теневая память 302
10.5 Резюме 304

Глава 11. Практический динамический анализ заражения с помощью libdft 305
11.1 Введение в libdft 305
11.1.1 Внутреннее устройство libdft 306
11.1.2 Политика заражения 309

ɳqdqyɐʎ bƐ dədƐbʎɜə d ɕɑɒɕɑʎ Ɉqdqʎqɐè

11.2 Использование DTA для обнаружения удаленного перехвата управления 310
11.2.1 Проверка информации о заражении 313
11.2.2 Источники заражения: заражение принятых байтов 315
11.2.3 Приемники заражения: проверка аргументов execve 317
11.2.4 Обнаружение попытки перехвата потока управления 318
11.3 Обход DTA с помощью неявных потоков 323
11.4 Детектор утечки данных на основе DTA 324
11.4.1 Источники заражения: прослеживание заражения для открытых файлов 327
11.4.2 Приемники заражения: мониторинг отправки по сети на предмет утечки данных 330
11.4.3 Обнаружение потенциальной утечки данных 331
11.5 Резюме 334

Глава 12. Принципы символического выполнения 335

12.1 Краткий обзор символического выполнения 336
12.1.1 Символическое и конкретное выполнение 336
12.1.2 Варианты и ограничения символического выполнения 340
12.1.3 Повышение масштабируемости символического выполнения 347
12.2 Удовлетворение ограничений с помощью Z3 349
12.2.1 Доказательство достижимости команды 350
12.2.2 Доказательство недостижимости команды 354
12.2.3 Доказательство общезначимости формулы 354
12.2.4 Упрощение выражений 356
12.2.5 Моделирование ограничений для машинного кода с битовыми векторами 356
12.2.6 Решение непроницаемого предиката над битовыми векторами 358
12.3 Резюме 359

Глава 13. Практическое символическое выполнение с помощью Triton 361

13.1 Введение в Triton 362
13.2 Представление символического состояния абстрактными синтаксическими деревьями 363
13.3 Обратное нарезание с помощью Triton 365
13.3.1 Заголовочные файлы и конфигурирование Triton 368
13.3.2 Конфигурационный файл символического выполнения 369
13.3.3 Эмуляция команд 370
13.3.4 Инициализация архитектуры Triton 372
13.3.5 Вычисления обратного среза 373
13.4 Использование Triton для увеличения покрытия кода 374
13.4.1 Создание символических переменных 376
13.4.2 Нахождение модели для нового пути 377

ŋqdpʎeʎ bƐ dǝdƎbʎɜǝ d ɕɑɒɔɑʎ Ɉqddʎqʎè

13.4.3 Тестирование инструмента покрытия кода ... 380
13.5 Автоматическая эксплуатация уязвимости ... 384
13.5.1 Уязвимая программа ... 385
13.5.2 Нахождение адреса уязвимой команды вызова ... 388
13.5.3 Построение генератора эксплойта ... 390
13.5.4 Получение оболочки с правами root ... 397
13.6 Резюме ... 400

ЧАСТЬ IV. ПРИЛОЖЕНИЯ

Приложение A. Краткий курс ассемблера x86 ... 402

A.1 Структура ассемблерной программы ... 403
A.1.1 Ассемблерные команды, директивы, метки и комментарии ... 404
A.1.2 Разделение данных и кода ... 405
A.1.3 Синтаксис AT&T и Intel ... 405
A.2 Структура команды x86 ... 405
A.2.1 Ассемблерное представление команд x86 ... 406
A.2.2 Структура команд x86 на машинном уровне ... 406
A.2.3 Регистровые операнды ... 407
A.2.4 Операнды в памяти ... 409
A.2.5 Непосредственные операнды ... 410
A.3 Употребительные команды x86 ... 410
A.3.1 Сравнение операндов и установки флагов состояния ... 411
A.3.2 Реализация системных вызовов ... 412
A.3.3 Реализация условных переходов ... 412
A.3.4 Загрузка адресов памяти ... 413
A.4 Представление типичных программных конструкций на языке ассемблера ... 413
A.4.1 Стек ... 413
A.4.2 Вызовы функций и кадры функций ... 414
A.4.3 Условные предложения ... 419
A.4.4 Циклы ... 420

Приложение B. Реализация перезаписи PT_NOTE с помощью libelf ... 422

B.1 Обязательные заголовки ... 423
B.2 Структуры данных, используемые в elfinject ... 423
B.3 Инициализация libelf ... 425
B.4 Получение заголовка исполняемого файла ... 428
B.5 Нахождение сегмента PT_NOTE ... 429
B.6 Внедрение байтов кода ... 430
B.7 Выравнивание адреса загрузки внедренной секции ... 431
B.8 Перезапись заголовка секции .note.ABI-tag ... 432
B.9 Задание имени внедренной секции ... 437
B.10 Перезапись заголовка программы PT_NOTE ... 439
B.11 Модификация точки входа ... 441

ŋqdpʎeʎ bƐ dǝdƎbʎɜǝ d ɕɑɒɔɑʎ Ɉqddʎqʎè

Приложение C. Перечень инструментов анализа двоичных файлов 443
C.1 Дизассемблеры 443
C.2 Отладчики 445
C.3 Каркасы дизассемблирования 445
C.4 Каркасы двоичного анализа 446

Приложение D. Литература для дополнительного чтения 447
D.1 Стандарты и справочные руководства 447
D.2 Статьи 448
D.3 Книги 450

Предметный указатель 451