УДК 004.382
ББК 32.973-018
М77

Монаппа К.А.
М77 Анализ вредоносных программ / пер. с анг. Д.А. Беликова. – М.: ДМК Пресс, 2019. – 452 с.: ил.

ISBN 978-5-97060-700-8

Книга учит концепциям, инструментам и методам распознавания вредоносных программ Windows и общим элементам анализа вредоносного ПО. Для лучшего восприятия в примерах данной книги используются различные реальные образцы вредоносного ПО, зараженные образы памяти и визуальные диаграммы.

Издание предназначено для специалистов-практиков в области кибербезопасности, будет полезно студентам, аспирантам и инженерам соответствующих специальностей. Оно пригодится в работе сотрудникам служб информационной безопасности и инженерам-исследователям в области кибербезопасности.

УДК 004.382
ББК 32.973-018

ISBN 978-1-78839-250-1 (анг.)
ISBN 978-5-97060-700-8 (рус.)

Содержание

Соавторы 15
Об авторе 15
О рецензентах 16

Предисловие 17
Для кого эта книга 18
Что рассматривается в этой книге 18
Чтобы получить максимальную отдачу от этой книги 19
Скачать цветные изображения 19
Используемые условные обозначения 19

Глава 1. Введение в анализ вредоносных программ 21
1.1 Что такое вредоносное ПО? 21
1.2 Что такое анализ вредоносных программ? 23
1.3 Почему анализ вредоносных программ? 23
1.4 Типы анализа вредоносных программ 24
1.5 Настройка тестовой среды 25
1.5.1 Требования к среде 26
1.5.2 Обзор архитектуры тестовой среды 26
1.5.3 Установка и настройка виртуальной машины Linux 28
1.5.4 Установка и настройка виртуальной машины Windows 34
1.6 Источники вредоносных программ 37
Резюме 38

Глава 2. Статический анализ 39
2.1 Определение типа файла 39
2.1.1 Определение типа файла с использованием ручного метода 40
2.1.2 Определение типа файла с использованием инструментальных средств 41
2.1.3 Определение типа файла с помощью Python 41
2.2 Сличение информации с помощью цифровых отпечатков 42
2.2.1 Генерирование криптографической хеш-функции с использованием инструментальных средств 43
2.2.2 Определение криптографической хеш-функции в Python 44
2.3 Многократное антивирусное сканирование 44

2.3.1 Сканирование подозрительного бинарного файла с помощью VirusTotal 44
2.3.2 Запрос значений хеш-функций с помощью открытого API VirusTotal 45
2.4 Извлечение строк 48
2.4.1 Извлечение строк с использованием инструментальных средств 48
2.4.2 Расшифровка обфусцированных строк с использованием FLOSS 50
2.5 Определение обфускации файла 51
2.5.1 Упаковщики и крипторы 52
2.5.2 Обнаружение обфусцированного файла с помощью Exeinfo PE 54
2.6 Проверка информации о PE-заголовке 55
2.6.1 Проверка файловых зависимостей и импорт 56
2.6.2 Проверка экспорта 59
2.6.3 Изучение таблицы секций PE-файла 60
2.6.4 Изучение временной метки компиляции 63
2.6.5 Изучение ресурсов PE-файлов 64
2.7 Сравнение и классификация вредоносных программ 66
2.7.1 Классификация вредоносных программ с использованием нечеткого хеширования 66
2.7.2 Классификация вредоносных программ с использованием хеша импорта 68
2.7.3 Классификация вредоносных программ с использованием хеша секций 70
2.7.4 Классификация вредоносных программ с использованием YARA 70
2.7.4.1 Установка YARA 71
2.7.4.2 Основы правил YARA 71
2.7.4.3 Запуск YARA 72
2.7.4.4 Применение YARA 73
Резюме 77

Глава 3. Динамический анализ 78

3.1 Обзор тестовой среды 78
3.2 Системный и сетевой мониторинг 79
3.3 Инструменты динамического анализа (мониторинга) 80
3.3.1 Проверка процесса с помощью Process Hacker 80
3.3.2 Определение взаимодействия системы с помощью Process Monitor 81
3.3.3 Регистрация действий системы с использованием Noriben 83
3.3.4 Захват сетевого трафика с помощью Wireshark 84
3.3.5 Симуляция служб с INetSim 85
3.4 Этапы динамического анализа 87
3.5 Собираем все вместе: анализируем исполняемый файл вредоносного ПО 88
3.5.1 Статический анализ образца 88
3.5.2 Динамический анализ образца 90
3.6 Анализ динамически подключаемой библиотеки (DLL) 93

ŋqdqɣɐʎ bƎ dəqƎbʎзə d ɓɔɑbɔɣ Ɉqdqʎqɣè

3.6.1 Почему злоумышленники используют библиотеки DLL......................95
3.6.2 Анализ DLL с помощью rundll32.exe...95
3.6.2.1 Как работает rundll32.exe 96
3.6.2.2 Запуск DLL с использованием rundll32.exe 96
Пример 1 – Анализ DLL без экспорта 96
Пример 2 – Анализ DLL, содержащей экспорт................................98
Пример 3 – Анализ DLL, принимающей аргументы экспорта 99
3.6.3 Анализ DLL с помощью проверки процессов 100
Резюме...102

Глава 4. Язык ассемблера и дизассемблирование для начинающих 103

4.1 Основы работы с компьютером...104
4.1.1 Память 105
4.1.1.1 Как данные хранятся в памяти 105
4.1.2 Центральный процессор 106
4.1.2.1 Машинный язык 106
4.1.3 Основы программы 106
4.1.3.1 Компиляция программы 106
4.1.3.2 Программа на диске..107
4.1.3.3 Программа в памяти..108
4.1.3.4 Дизассемблирование программы (от машинного кода к коду ассемблера) 111
4.2 Регистры процессора 112
4.2.1 Регистры общего назначения 112
4.2.2 Указатель инструкций (EIP)..113
4.2.3 Регистр EFLAGS 113
4.3 Инструкции по передаче данных 113
4.3.1 Перемещение константы в регистр 113
4.3.2 Перемещение значений из регистра в регистр 114
4.3.3 Перемещение значений из памяти в регистры...........................114
4.3.4 Перемещение значений из регистров в память 116
4.3.5 Задача по дизассемблированию 116
4.3.6 Решение задачи 117
4.4 Арифметические операции...119
4.4.1 Задача по дизассемблированию 120
4.4.2 Решение задачи 120
4.5 Побитовые операции..121
4.6 Ветвление и условные операторы 123
4.6.1 Безусловные переходы 123
4.6.2 Условные переходы...123
4.6.3 Оператор if 125
4.6.4 Оператор If-Else 125

4.6.5 Оператор If-Elseif-Else 126
4.6.6 Задача по дизассемблированию 127
4.6.7 Решение задачи 127
4.7 Циклы 130
4.7.1 Задача по дизассемблированию 131
4.7.2 Решение задачи 132
4.8 Функции 133
4.8.1 Стек 134
4.8.2 Функция вызова 135
4.8.3 Возвращение из функции 136
4.8.4 Параметры функции и возвращаемые значения 136
4.9 Массивы и строки 140
4.9.1 Задача по дизассемблированию 142
4.9.2 Решение задачи 142
4.9.3 Строки 146
4.9.3.1 Строковые инструкции 146
4.9.3.2 Перемещение из памяти в память (movsx) 147
4.9.3.3 Инструкции повтора (rep) 148
4.9.3.4 Сохранение значения из регистра в память (Stosx) 148
4.9.3.5 Загрузка из памяти в регистр (lodsx) 149
4.9.3.6 Сканирование памяти (scasx) 149
4.9.3.7 Сравнение значений в памяти (Cmpsx) 149
4.10 Структуры 149
4.11 Архитектура x64 151
4.11.1 Анализ 32-битного исполняемого файла на 64-разрядной операционной системе Windows 152
4.12 Дополнительная литература 153
Резюме 154

Глава 5. Дизассемблирование с использованием IDA 155

5.1 Инструментальные средства анализа кода 155
5.2 Статический анализ кода (дизассемблирование) с использованием IDA 156
5.2.1 Загрузка двоичного файла в IDA 157
5.2.2 Изучение окон IDA 158
5.2.2.1 Окно Дизассемблирование 159
5.2.2.2 Окно Функции 161
5.2.2.3 Окно Вывод 161
5.2.2.4 Окно шестнадцатеричного представления 161
5.2.2.5 Окно Структуры 161
5.2.2.6 Окно Импорт 161
5.2.2.7 Окно Экспорт 162
5.2.2.8 Окно Строки 162
5.2.2.9 Окно Сегменты 162

5.2.3 Улучшение дизассемблирования с помощью IDA 162
5.2.3.1 Переименование переменных и функций 164
5.2.3.2 Комментирование в IDA 165
5.2.3.3 База данных IDA 166
5.2.3.4 Форматирование операндов 168
5.2.3.5 Навигация по адресам 168
5.2.3.6 Перекрестные ссылки 169
5.2.3.7 Вывод списка всех перекрестных ссылок 171
5.2.3.8 Ближнее представление и графы 172
5.3 Дизассемблирование Windows API 175
5.3.1 Понимание Windows API 176
5.3.1.1 API-функции Юникод и ANSI 179
5.3.1.2 Расширенные API-функции 180
5.3.2 Сравнение 32-битного и 64-битного Windows API 180
5.4 Исправление двоичного кода с использованием IDA 182
5.4.1 Исправление байтов программы 183
5.4.2 Исправление инструкций 185
5.5 Сценарии и плагины IDA 186
5.5.1 Выполнение сценариев IDA 186
5.5.2 IDAPython 187
5.5.2.1 Проверка наличия API CreateFile 188
5.5.2.2 Перекрестные ссылки кода на CreateFile с использованием IDAPython 189
5.5.3 Плагины IDA 189
Резюме 190

Глава 6. Отладка вредоносных двоичных файлов 191

6.1 Общие концепции отладки 192
6.1.1 Запуск и подключение к процессам 192
6.1.2 Контроль выполнения процесса 192
6.1.3 Прерывание программы с помощью точек останова 193
6.1.4 Трассировка выполнения программы 195
6.2 Отладка двоичного файла с использованием x64dbg 195
6.2.1 Запуск нового процесса в x64dbg 195
6.2.2 Присоединение к существующему процессу с использованием x64dbg 196
6.2.3 Интерфейс отладчика x64dbg 197
6.2.4 Контроль за выполнением процесса с использованием x64dbg 200
6.2.5 Установка точки останова в x64dbg 201
6.2.6 Отладка 32-битного вредоносного ПО 201
6.2.7 Отладка 64-битной вредоносной программы 203
6.2.8 Отладка вредоносной DLL-библиотеки с использованием x64dbg 205
6.2.8.1 Использование rundll32.exe для отладки библиотеки DLL в x64dbg 206

ŋqdqүeɏ bɆ dәqɆbɎзə d ɛɑqɔʎ Ɉqdqɏqүè

6.2.8.2 Отладка DLL в определенном процессе 207
6.2.9 Трассировка выполнения в x64dbg..208
6.2.9.1 Трассировка инструкций 209
2.9.2 Трассировка функций...210
6.2.10 Исправления в x64dbg..211
6.3 Отладка двоичного файла с использованием IDA...............................213
6.3.1 Запуск нового процесса в IDA 213
6.3.2 Присоединение к существующему процессу с использованием IDA...214
6.3.3 Интерфейс отладчика IDA 215
6.3.4 Контроль выполнения процесса с использованием IDA 217
6.3.5 Установка точки останова в IDA..217
6.3.6 Отладка вредоносных исполняемых файлов....................................219
6.3.7 Отладка вредоносной библиотеки DLL с помощью IDA 220
6.3.7.1 Отладка DLL в определенном процессе...221
6.3.8 Трассировка выполнения с использованием IDA..............................222
6.3.9 Написание сценариев отладки с использованием IDAPython...........225
6.3.9.1 Пример – определение файлов, доступных вредоносному ПО...228
6.4 Отладка приложения .NET..229
Резюме...231

Глава 7. Функциональные возможности вредоносного ПО и его персистентность 232
7.1 Функциональные возможности вредоносного ПО 232
7.1.1 Загрузчик 232
7.1.2 Дроппер...233
7.1.2.1 Реверс-инжиниринг 64-битного дроппера 235
7.1.3 Кейлоггер 236
7.1.3.1 Кейлоггер, использующий GetAsyncKeyState()...............................236
7.1.3.2 Кейлоггер, использующий SetWindowsHookEx()..............................238
7.1.4 Репликация вредоносных программ через съемные носители..........238
7.1.5 Управление и контроль, осуществляемые вредоносными программами (C2) 243
7.1.5.1 Управление и контроль с использованием HTTP..............................243
7.1.5.2 Осуществление команды и контроля в пользовательском режиме 246
7.1.6 Выполнение на основе PowerShell 249
7.1.6.1 Основы команд PowerShell 250
7.1.6.2 Сценарии PowerShell и политика выполнения 251
7.1.6.2 Анализ команд/скриптов PowerShell 252
7.1.6.3 Как злоумышленники используют PowerShell 253
7.2 Методы персистентности вредоносных программ 255
7.2.1 Запуск ключа реестра..255
7.2.2 Запланированные задачи 256

7.2.3 Папка запуска 256
7.2.4 Записи реестра Winlogon 257
7.2.5 Параметры выполнения файла изображения 258
7.2.6 Специальные возможности 259
7.2.7 AppInit_DLLs 261
7.2.8 Захват порядка поиска DLL 262
7.2.9 Захват COM-объекта 263
7.2.10 Служба 266
Резюме 270

Глава 8. Внедрение кода и перехват 271

8.1 Виртуальная память 271
8.1.1 Компоненты памяти процесса (пространство пользователя) 274
8.1.2 Содержимое памяти ядра (пространство ядра) 276
8.2 Пользовательский режим и режим ядра 277
8.2.1 Поток вызовов Windows API 278
8.3 Методы внедрения кода 280
8.3.1 Удаленное внедрение DLL 282
8.3.2 Внедрение DLL с использованием асинхронного вызова процедур 284
8.3.3 Внедрение DLL с использованием SetWindowsHookEx() 286
8.3.4 Внедрение DLL с использованием прокладок 288
8.3.4.1 Создание прокладки 289
8.3.4.2 Артефакты прокладки 294
8.3.4.3 Как злоумышленники используют прокладки 295
8.3.4.4 Анализ базы данных прокладки 296
8.3.5 Внедрение удаленного исполняемого файла или шелл-кода 297
8.3.6 Внедрение пустого процесса (опустошение процесса) 298
8.4 Методы перехвата 302
8.4.1 Перехват таблицы адресов импорта 303
8.4.2 Встраиваемый перехват (Inline Patching) 304
8.4.3 Исправления в памяти с помощью прокладки 307
8.5 Дополнительная литература 310
Резюме 311

Глава 9. Методы обфускации вредоносных программ 312

9.1 Простое кодирование 314
9.1.1 Шифр Цезаря 314
9.1.1.1 Как работает шифр Цезаря 314
8.1.1.2 Расшифровка шифра Цезаря в Python 315
9.1.2 Кодирование Base64 316
9.1.2.1 Перевод данных в Base64 316
9.1.2.2 Кодирование и декодирование Base64 318
9.1.2.3 Декодирование пользовательской версии Base64 319

9.1.2.4 Идентификация Base64 321
9.1.3 XOR-шифрование 322
9.1.3.1 Однобайтовый XOR 323
9.1.3.2 Поиск XOR-ключа с помощью полного перебора 326
9.1.3.3 Игнорирование XOR-шифрования нулевым байтом 327
9.1.3.4 Многобайтовое XOR-шифрование 329
8.1.3.5 Идентификация XOR-шифрования 330
9.2 Вредоносное шифрование 331
9.2.1 Идентификация криптографических подписей с помощью Signsrch 332
9.2.2 Обнаружение криптоконстант с помощью FindCrypt2 335
9.2.3 Обнаружение криптографических подписей с использованием YARA 336
9.2.4 Расшифровка в Python 337
9.3 Пользовательское кодирование/шифрование 338
9.4 Распаковка вредоносных программ 342
9.4.1 Ручная распаковка 343
9.4.1.1 Идентификация исходной точки входа 344
9.4.1.2 Выгрузка памяти процесса с помощью Scylla 347
9.4.1.3 Исправление таблицы импорта 348
9.4.2 Автоматическая распаковка 350
Резюме 353

Глава 10. Охота на вредоносные программы с использованием криминалистического анализа дампов памяти 354

10.1 Этапы криминалистического анализа дампов памяти 355
10.2 Создание дампа памяти 356
10.2.1 Создание дампа памяти с использованием DumpIt 356
10.3 Обзор Volatility 359
10.3.1 Установка Volatility 359
10.3.1.1 Автономный исполняемый файл Volatility 359
10.3.1.2 Исходный пакет Volatility 360
10.3.2 Использование Volatility 361
10.4 Перечисление процессов 362
10.4.1 Обзор процесса 363
10.4.1.1 Изучение структуры _EPROCESS 364
10.4.1.2 Понимание ActiveProcessLinks 367
10.4.2. Вывод списка процессов с использованием psscan 369
10.4.2.1 Прямое манипулирование объектами ядра (DKOM) 369
10.4.2.2 Общие сведения о сканировании тегов пула 371
10.4.3 Определение связей между процессами 373
10.4.4 Вывод списка процессов с использованием psxview 374

10.5 Вывод списка дескрипторов процесса 376
10.6 Вывод списка DLL 379
10.6.1 Обнаружение скрытой библиотеки DLL с помощью ldrmodules 382
10.7 Сброс исполняемого файла и DLL 383
10.8 Вывод списка сетевых подключений и сокетов 385
10.9 Проверка реестра 386
10.10 Проверка служб 388
10.11 Извлечение истории команд 390
Резюме 393

Глава 11. Обнаружение сложных вредоносных программ с использованием криминалистического анализа дампов памяти 394

11.1 Обнаружение внедрения кода 394
11.1.1 Получение информации о дескрипторе виртуальных адресов 396
11.1.2 Обнаружение внедренного кода с использованием дескриптора виртуальных адресов 397
11.1.3 Сброс области памяти процесса 399
11.1.4 Обнаружение внедренного кода с помощью malfind 399
11.2 Исследование внедрения пустого процесса 400
11.2.1 Этапы внедрения пустого процесса 401
11.2.2 Обнаружение внедрения пустого процесса 402
11.2.3 Варианты внедрения пустого процесса 404
11.3 Обнаружение перехвата API 407
11.4 Руткиты в режиме ядра 408
11.5 Вывод списка модулей ядра 409
11.5.1 Вывод списка модулей ядра с использованием driverscan 411
11.6 Обработка ввода/вывода 412
11.6.1 Роль драйвера устройства 414
11.6.2 Роль менеджера ввода/вывода 421
11.6.3 Связь с драйвером устройства 421
11.6.4 Запросы ввода/вывода для многоуровневых драйверов 424
11.7 Отображение деревьев устройств 427
11.8 Обнаружение перехвата пространства ядра 429
11.8.1 Обнаружение перехвата SSDT 429
11.8.2 Обнаружение перехвата IDT 432
11.8.3 Идентификация встроенных перехватов ядра 433
11.8.4 Обнаружение перехватов функций IRP 434
11.9 Обратные вызовы из ядра и таймеры 437
Резюме 442

Предметный указатель 443

ŋqddγeɏ bɇ dədɇbɏɜə d ɕɑɑbɔɏ ɟqddɏqɏè