ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

1

Таблица релокаций

Таблица релокаций

Напишем простую программу и дизассемблируем её

```
format PE GUI

include 'win32ax.inc'

.code

  start:
        invoke  MessageBox,HWND_DESKTOP,"Hi! I'm the example program!",invoke GetCommandLine,MB_OK
        invoke  ExitProcess,0

.end start
```

```
.00401000: 6A00                  push        0
.00401002: FF1568204000          call        GetCommandLineA
.00401008: 50                    push        eax
.00401009: E81D000000            call        .00040102B --↓1
.0040100E: 48                    dec         eax
.0040100F: 69212049276D          imul        esp,[ecx],06D274920 ;'m'I '
.00401015: 20746865              and         [eax][ebp]*2[065],dh
.00401019: 206578                and         [ebp][078],ah
.0040101C: 61                    popad
.0040101D: 6D                    insd
.0040101E: 706C                  jo          .00040108C --↓2
.00401020: 65207072              and         gs:[eax][072],dh
.00401024: 6F                    outsd
.00401025: 677261                jc          .000401089 --↓3
.00401028: 6D                    insd
.00401029: 2100                  and         [eax],eax
.0040102B: 6A00                 1push        0
.0040102D: FF1598204000          call        MessageBoxA
.00401033: 6A00                  push        0
.00401035: FF1564204000          call        ExitProcess
```

Таблица релокаций

Мы видим, что есть инструкции, где захардкожен **абсолютный** виртуальный адрес, к примеру – адрес функции GetCommandLineA = 0x00402068

```
00401000: 6A00                  push    0
.00401002: FF1568204000         call    GetCommandLineA
.00401008: 50                   push    eax
.00401009: E81D000000           call    .00040102B --↓1
.0040100E: 48                   dec     eax
.0040100F: 69212049276D         imul    esp,[ecx],06D274920 ;'m'I '
.00401015: 20746865             and     [eax][ebp]*2[065],dh
.00401019: 206578               and     [ebp][078],ah
.0040101C: 61                   popad
.0040101D: 6D                   insd
.0040101E: 706C                 jo      .00040108C --↓2
.00401020: 65207072             and     gs:[eax][072],dh
.00401024: 6F                   outsd
.00401025: 677261               jc      .000401089 --↓3
.00401028: 6D                   insd
.00401029: 2100                 and     [eax],eax
.0040102B: 6A00                 1push   0
.0040102D: FF1598204000         call    MessageBoxA
.00401033: 6A00                 push    0
00401035: FF1564204000          call    ExitProcess
```

Таблица релокаций

Линковщик помещает RVA адреса абсолютных указателей
в специальную таблицу, чтобы их изменить, в случае, если образ загрузился “не по тому imageBase” адресу.
FixUp – подоменяемый адрес

Relocation Table состоит из структуры IMAGE_BASE_RELOCATION,
Сразу за которой находится массив из fixUpов

```
typedef struct _IMAGE_BASE_RELOCATION {
    DWORD   VirtualAddress;
    DWORD   SizeOfBlock;
//  WORD    TypeOffset[1];
} IMAGE_BASE_RELOCATION;
typedef IMAGE_BASE_RELOCATION UNALIGNED * PIMAGE_BASE_RELOCATION;
```

Таблица релокаций

Имя	Описание
VirtualAddress	RVA секции в которой находятся абсолютные адреса, требующие пересчёта
SizeOfBlock	Размер всего массива из fixUpов текущей секции + sizeof(IMAGE_BASE_RELOCATION)

```
typedef struct _IMAGE_BASE_RELOCATION {
    DWORD   VirtualAddress;
    DWORD   SizeOfBlock;
//  WORD    TypeOffset[1];
} IMAGE_BASE_RELOCATION;
typedef IMAGE_BASE_RELOCATION UNALIGNED * PIMAGE_BASE_RELOCATION;
```

Таблица релокаций

Адрес таблицы релокаций получается из массива DataDirectory по 5-ому индексу

```
DWORD imageRelocationVA = NtHeader->OptionalHeader.DataDirectory[IMAGE_DIRECTORY_ENTRY_BASERELOC].VirtualAddress;
DWORD imageRelocationRAW = RVA2RAW(fileImage, imageRelocationVA);

PIMAGE_BASE_RELOCATION relocationFileBase = (PIMAGE_BASE_RELOCATION)((DWORD)fileImage + imageRelocationRAW);

while (relocationFileBase->VirtualAddress && relocationFileBase->SizeOfBlock)
```

Имя	Описание
VirtualAddress	RVA секции в которой находятся абсолютные адреса, требующие пересчёта
SizeOfBlock	Размер всего массива из fixUpов текущей секции + sizeof(IMAGE_BASE_RELOCATION)

2

Таблица экспорта

Таблица экспорта

Описывается структурой _IMAGE_EXPORT_DIRECTORY

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
    DWORD   Base;
    DWORD   NumberOfFunctions;
    DWORD   NumberOfNames;
/*+0x1C*/   DWORD   AddressOfFunctions;     // RVA from base of image
    DWORD   AddressOfNames;         // RVA from base of image
    DWORD   AddressOfNameOrdinals;  // RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

Алгоритм получения адреса функции по таблице экспорта I

OTUS

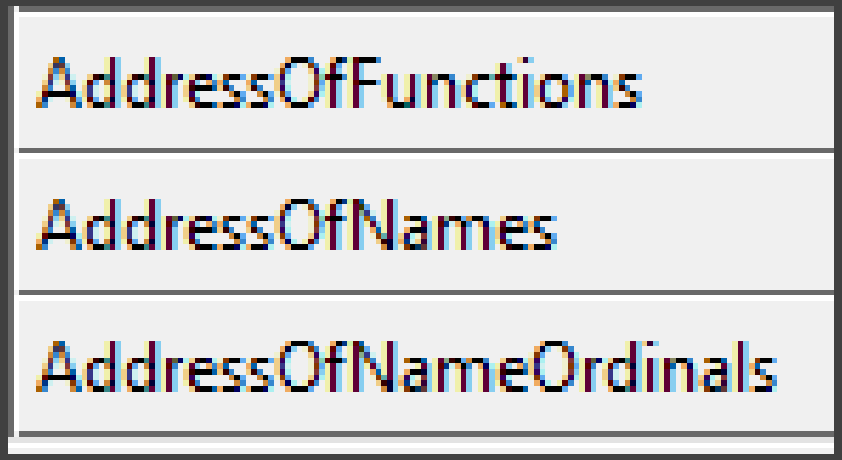
Получить указатель на _IMAGE_DOS_HEADER

По полю _IMAGE_DOS_HEADER.e_lfanew (0x3C) получить смещение до _IMAGE_NT_HEADERS

По смещению 0x78 от _IMAGE_NT_HEADERS читается смещение до таблицы экспорта

Алгоритм получения адреса функции по таблице экспорта II

Читаются три указателя на массивы



Алгоритм получения адреса функции по таблице экспорта III

OTUS

Из массива имён находится нужная функция по имени (или хешу) и запоминается её индекс N

Из массива ординалов читается N-ый элемент - M

Из массива адресов функций читается M-ый элемент - address

Таблица экспорта

Итог:

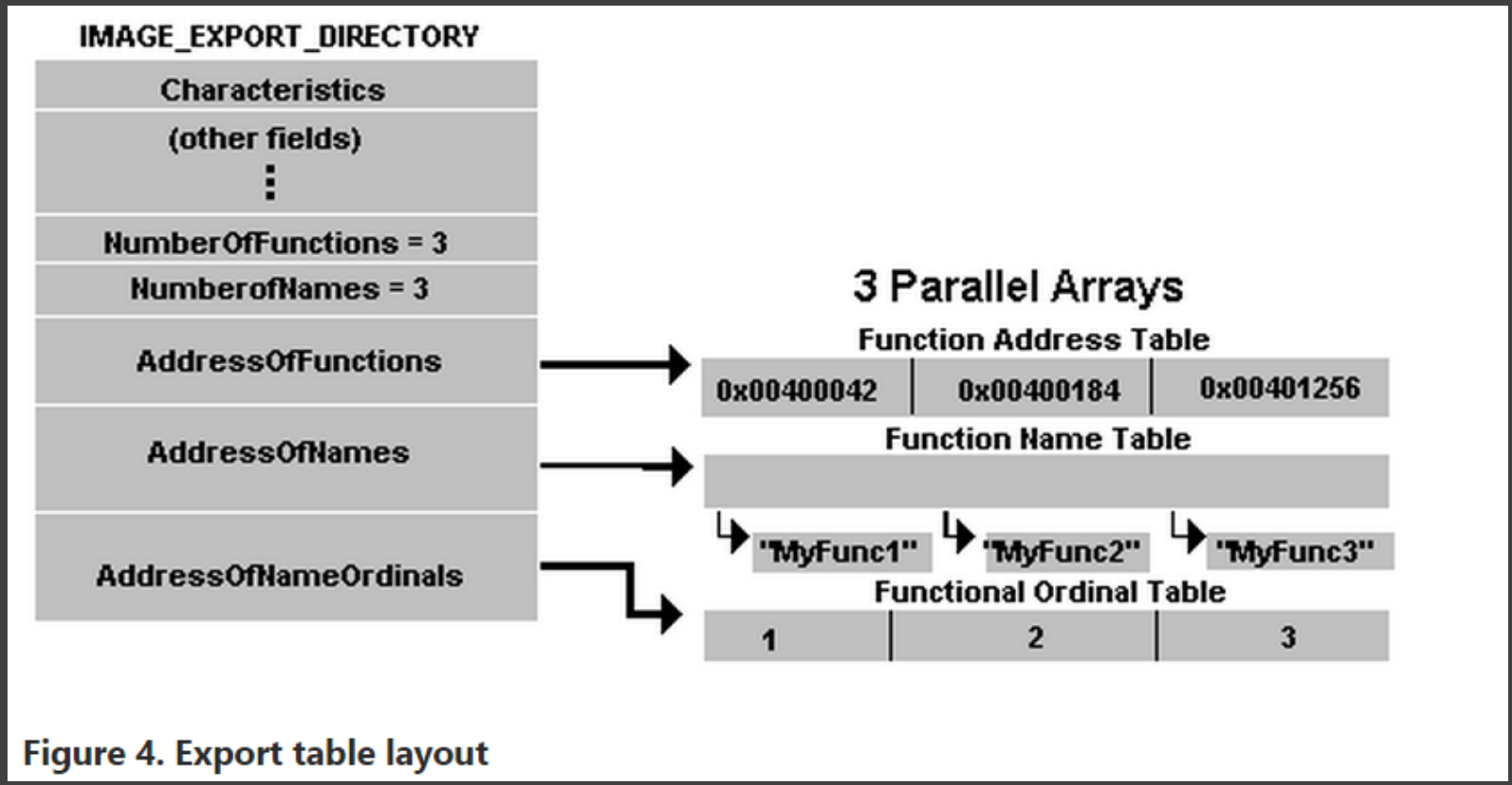


Figure 4. Export table layout

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!