ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

Таблица импорта

PE Loader

Пакулов Артур

1

PE Loader

Алгоритм загрузки PE файла

OTUS

- Парсинг DOS загаловка
- Парсинг PE заголовка
- Копирование DOS/PE заголовков по адресу imageBase
- Копирование всех секций по адресу VirtualAddress
- Обработка таблицы импорта
- Обработка таблицы перемещений
- Выставление прав на секции
- Передача управления на EntryPoint

Алгоритм загрузки PE файла

- Обработка TLS колбеков
- Обработка ресурсов
- Корректировка PEB
- Заглушка для GetModuleHandle

Парсинг заголовков

readPe считывает файл и возвращает адрес памяти

```
LPSTR fileName = "D:\\Languages\\fasm\\EXAMPLES\\HELLO\\HELLO_R2S.EXE";
LPVOID fileImage = readPe(fileName, fileSize);

printf("selfModule: 0x%08x\n", GetModuleHandle(NULL));

IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);

LPVOID peImage = VirtualAlloc(NULL, NtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
PIMAGE_SECTION_HEADER pSecHeader;

//copy image headers
MoveMemory(peImage, fileImage, NtHeader->OptionalHeader.SizeOfHeaders);
```

Парсинг заголовков

readPe считывает файл и возвращает адрес памяти

```
LPSTR fileName = "D:\\Languages\\fasm\\EXAMPLES\\HELLO\\HELLO_R2S.EXE";
LPVOID fileImage = readPe(fileName, fileSize);

printf("selfModule: 0x%08x\n", GetModuleHandle(NULL));

IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);

LPVOID peImage = VirtualAlloc(NULL, NtHeader->OptionalHeader.SizeOfImage, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
PIMAGE_SECTION_HEADER pSecHeader;

//copy image headers
MoveMemory(peImage, fileImage, NtHeader->OptionalHeader.SizeOfHeaders);
```

Заголовки секций

Сразу после заголовка PE в файле располагается массив заголовков секций

```
typedef struct _IMAGE_SECTION_HEADER {
    BYTE    Name[IMAGE_SIZEOF_SHORT_NAME];
    union {
            DWORD   PhysicalAddress;
            DWORD   VirtualSize;
    } Misc;
    DWORD   VirtualAddress;
    DWORD   SizeOfRawData;
    DWORD   PointerToRawData;
    DWORD   PointerToRelocations;
    DWORD   PointerToLinenumbers;
    WORD    NumberOfRelocations;
    WORD    NumberOfLinenumbers;
    DWORD   Characteristics;
} IMAGE_SECTION_HEADER, *PIMAGE_SECTION_HEADER;

#define IMAGE_SIZEOF_SECTION_HEADER          40
```

Заголовки секций

OTUS

Не указанные поля равны 0

Название	Описание
Misc.VirtualSize	Размер секции в памяти. Если это значение больше SizeOfRawData, то секция дополняется в памяти нулевыми байтами
VirtualAddress	**RVA** секции в памяти
SizeOfRawData	Размер секции в **файле**. Всегда кратен FileAlignment imageOptionalHeader
PointerToRawData	Смещение в **файле** до начала данных секций. Всегда кратно FileAlignment из imageOptionalHeader
Characteristics	Атрибуты секции

RVA2RAW

Перевод виртуального адреса PE файла в его файловое смещение

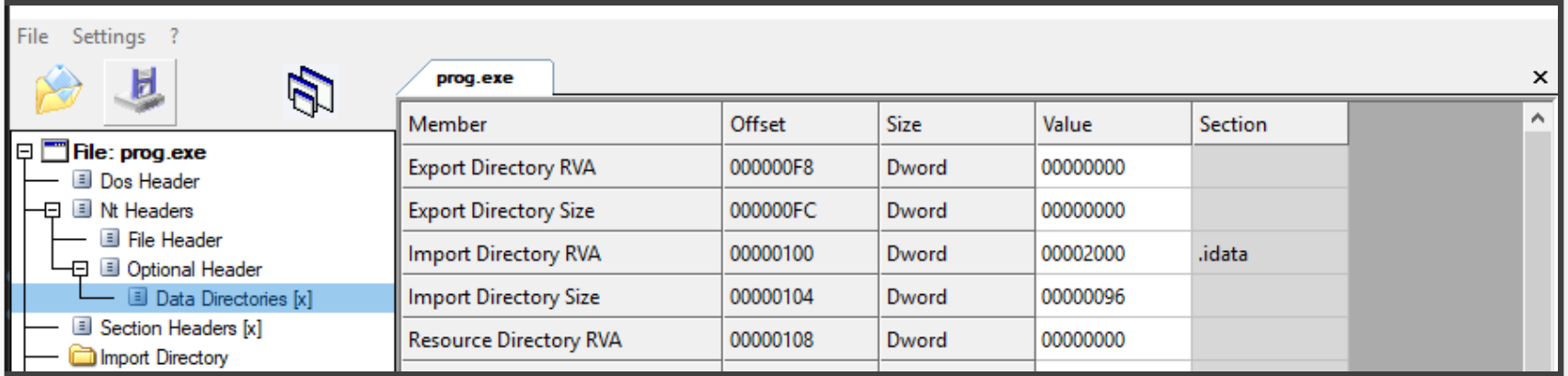
```
DWORD RVA2RAW(LPVOID fileImage, DWORD dwRVA)
{
    DWORD dwRawRVAAdr(0);
    IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
    IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);
    PIMAGE_SECTION_HEADER pSections = IMAGE_FIRST_SECTION(NtHeader);

    if (!pSections)
    {
        return dwRawRVAAdr;
    }

    while (pSections->VirtualAddress != 0)
    {
        if (dwRVA >= pSections->VirtualAddress &&
            dwRVA < pSections->VirtualAddress + pSections->SizeOfRawData)
        {
            dwRawRVAAdr = (dwRVA - pSections->VirtualAddress) + pSections->PointerToRawData;
            break;
        }
        pSections++;
    }

    return dwRawRVAAdr;
}
```

2

Таблица импорта

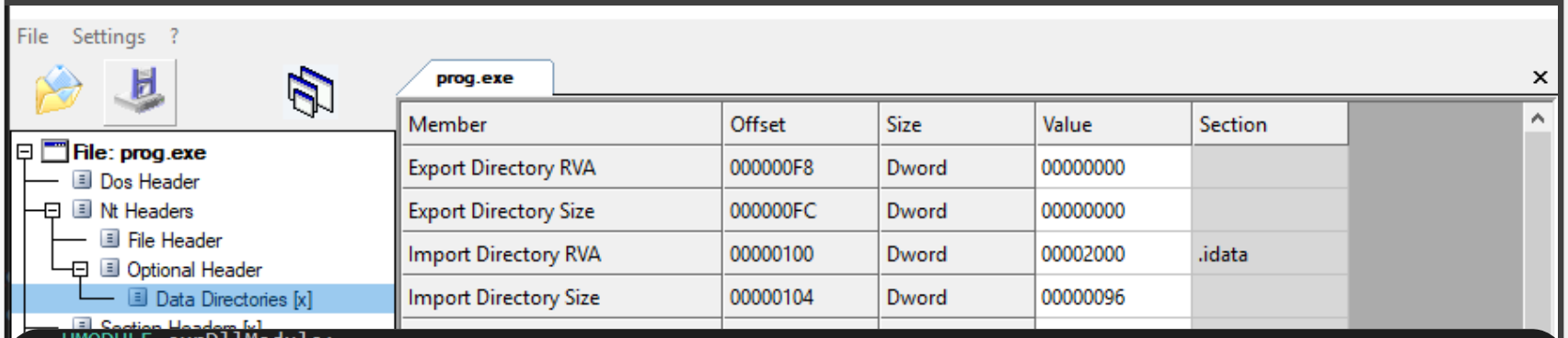
Парсинг таблицы импорта

Смешение до import Table получается из массива DataDirectory - по индексу == 1



Парсинг таблицы импорта

Смещение до import Table получается из массива DataDirectory – по индексу == 1




```
HMODULE curDllModule;
FUNC_ITEM func;
FUNCS_ARRAY result;
int imageDescrIndex = 0, thunkIndex = 0;

IMAGE_DOS_HEADER* DOSHeader = PIMAGE_DOS_HEADER(fileImage);
IMAGE_NT_HEADERS* NtHeader = PIMAGE_NT_HEADERS(DWORD(fileImage) + DOSHeader->e_lfanew);

DWORD imageImportDescrVA = NtHeader->OptionalHeader.DataDirectory[1].VirtualAddress;
DWORD imageImportDescrRAW = RVA2RAW(fileImage, imageImportDescrVA);

PIMAGE_IMPORT_DESCRIPTOR fileImageImportDescr = (PIMAGE_IMPORT_DESCRIPTOR)((DWORD)fileImage + imageImportDescrRAW);
vector<string> dllList;
```

Парсинг таблицы импорта

По полученному смещению находится массив структур типа IMAGE_IMPORT_DESCRIPTOR

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;            // 0 for terminating null import descriptor
        DWORD   OriginalFirstThunk;         // RVA to original unbound IAT (PIMAGE_THUNK_DATA)
    } DUMMYUNIONNAME;
    DWORD   TimeDateStamp;                  // 0 if not bound,
                                            // -1 if bound, and real date\time stamp
                                            //     in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
                                            // O.W. date/time stamp of DLL bound to (Old BIND)

    DWORD   ForwarderChain;                 // -1 if no forwarders
    DWORD   Name;
    DWORD   FirstThunk;                     // RVA to IAT (if bound this IAT has actual addresses)
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

IMAGE_IMPORT_DESCRIPTOR

Каждой динамической библиотеке соответствует по одной такой структуре

Название	Описание
Characteristics	RVA адрес массива, содержащего RVA адреса имён импортируемых функций (может быть 0)
TimeDateStamp	Дата/время
ForwarderChain	Обычное значение – 0xFFFFFFFF
Name	RVA строки имени динамической библиотеки
FirstThunk	Вторая копия Characteristics

Получение имён функций

Characteristics и FirstThunk указывают на массив из структур IMAGE_THUNK_DATA32

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;      // PBYTE
        DWORD Function;             // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;        // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;
```

Если старшее слово поля == 0x8000, то младшее слово содержит ординал

Получение имён функций

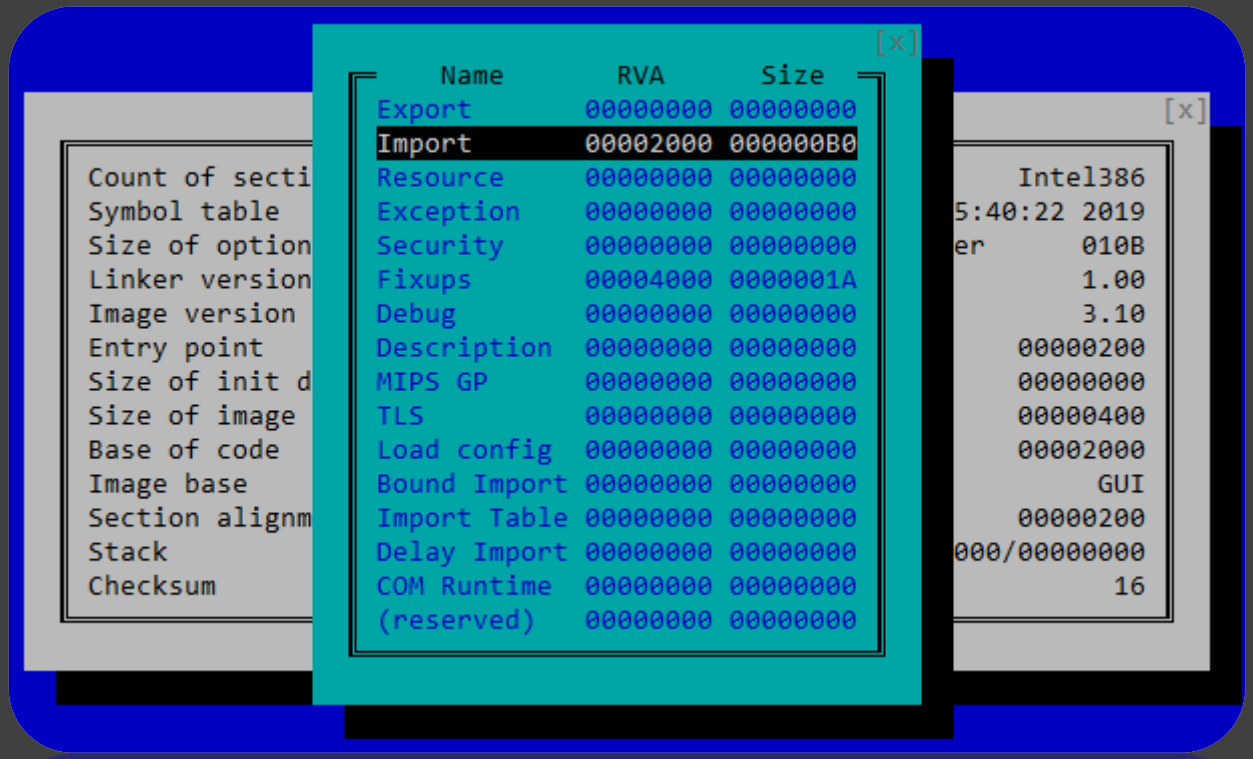
AddressOfData + 2 – адрес имени импортируемой функции

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;      // PBYTE
        DWORD Function;             // PDWORD
        DWORD Ordinal;
        DWORD AddressOfData;        // PIMAGE_IMPORT_BY_NAME
    } u1;
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;
```

Если старшее слово поля == 0x8000, то младшее слово содержит ординал

Пример: Получить имя импортируемой функции в hiew

1. Переходим на таблицу импорта: F8->F10, выбираем Import



Пример: Получить имя импортируемой функции в hiew

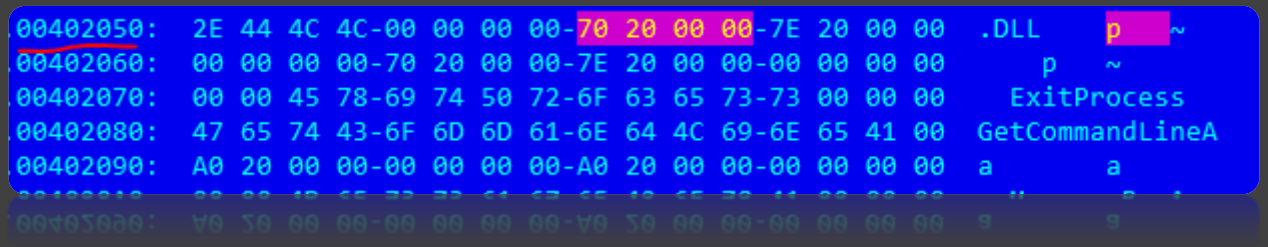
2. Первые 4 байта (поле Characteristics – 0x00002058) соответствуют первому элементу массива структур IMAGE_IMPORT_DESCRIPTOR. Это RVA структуры IMAGE_THUNK_DATA32.

Перейдём по адресу: F5, вводим 0x2058

Пример: Получить имя импортируемой функции в hiew

3. Получаем поле AddressOfData структуры IMAGE_THUNK_DATA32 = 0x00002070

По адресу 0x00002070 + 2 должно быть имя первой импотрируемой функции



Перейдём по адресу: F5, вводим 0x2072

Пример: Получить имя импортируемой функции в hiew

3. Получаем поле AddressOfData структуры IMAGE_THUNK_DATA32 = 0x00002070

По адресу 0x00002070 + 2 должно быть имя первой импотрируемой функции

Перейдём по адресу: F5, вводим 0x2072

Внимание!

Поле AddressOfData массива FirstThunk заменяется виртуальными адресами импортируемых функций загрузчиком PE файлов!

Массив Characteristics остаётся без изменения

Перейдём по адресу: F5, вводим 0x2072

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!