7. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1) Гордеев, А.В., Молчанов, А.Ю. Системное программное обеспечение – Спб.: Питер, 2001. – 736с. ил.

2) Рихтер, Дж. Windows для профессионалов: создание эффективных WIN32 приложений с учетом специфики 64-х разрядной версии Windows. – СПб.: Издательский дом «Питер», 2001.

3) Брой, М. Информатика. Основополагающее введение. Структуры систем и системное программирование.– М.: Диалог-МИФИ, 1996

4) Ганеев, Р.М., Проектирование интерфейса пользователя средствами Win32 API. – М.: Горячая линия–Телеком, 2001. – 336 с.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«__» _____2017 г.

Обзор PE-формата исполняемых файлов платформы Win32

Методические указания по выполнению лабораторной работы по дисциплине «Безопасность операционных систем» для студентов укрупненной группы специальностей 10.00.00

Курск 2017

Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 09.09.2021 14:38:09
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

В нашем примере нам нужны только имена импортируемых функций. Их можно найти во второй секции (рис. 13)

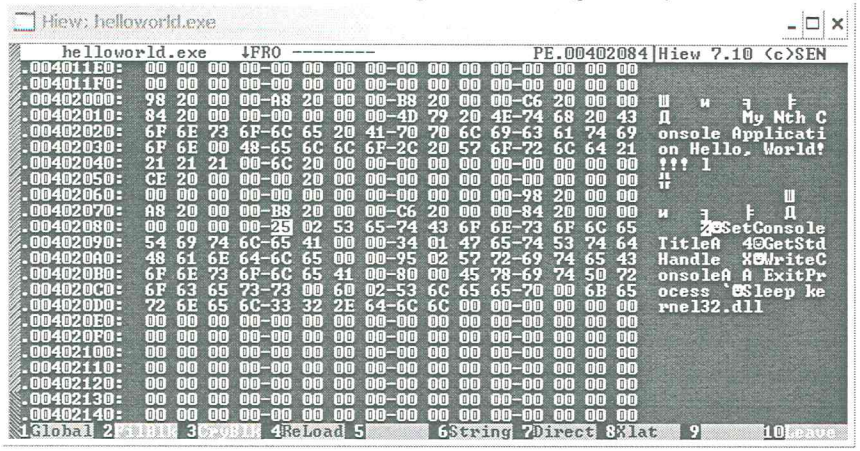

Рис. 13 – Название импортируемых функций в коде программы

Каждая импортируемая функция содержит слово-индекс для быстрого нахождения функции в IAT загрузчиком и её имя. Наше приложение использует 5 импортируемых функций: SetConsoleTitleA, GetStdHandle, WriteConsoleA, ExitProcess, Sleep. Все они импортируются из библиотеки kernel32.dll. Список импортируемых библиотек можно увидеть, нажав F8 (Header) и F7 (Import) в HIEW (рис. 14)

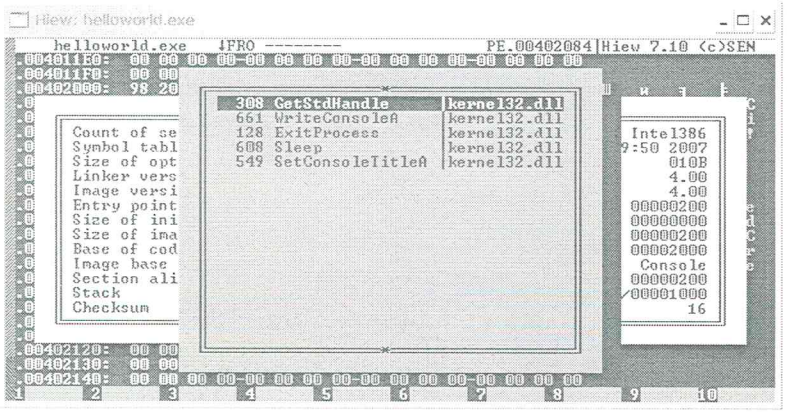

Рис. 14 – Список импортируемых функций.

549

ОГЛАВЛЕНИЕ

1. Цель работы. 4
2. Сведения о структуре исполнеяемых на платформе Win32 файлов. 4
3. Исследование формата исполняемого файла. 8
3.1. Анализ заголовка файла. 8
3.2. Секции. 15
3.3. Таблица импорта функций 17
4. Задание на лабораторную работу 19
5. Содержание отчёта. 19
6. Вопросы для самоконтроля. 19
7. Библиографический список 20

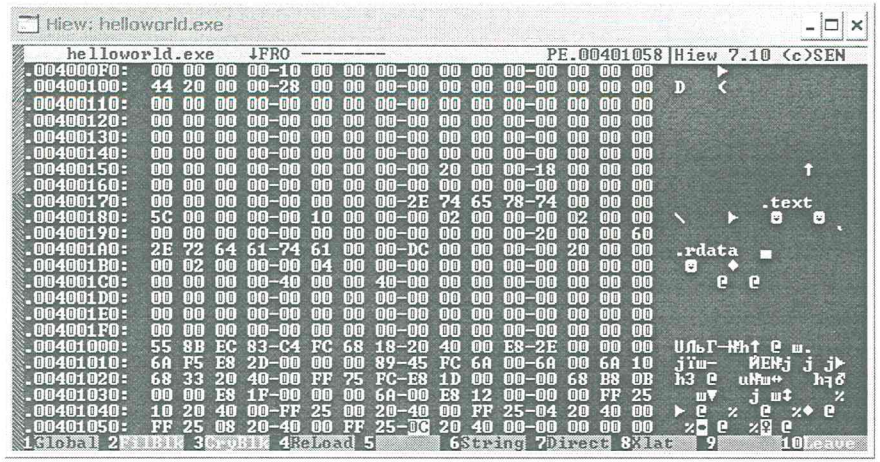

Рис. 11 – Таблицы секций приложения

Рассмотрим первую секцию. Следующие за её именем двойные слова представляют собой её виртуальный размер, относительный виртуальный адрес, физический адрес, внутрифайловое смещение и характеристики (как указано в структуре).

Выписываем необходимые данные:

- Имя секции: .text
- RVA начала: 0x00001000
- Внутрифайловое смещение начала: 0x00000200
- Виртуальный размер секции: 5Ch = 92 байт
- Физический размер секции: 200h = 512 байт

Аналогично записываем в отчет данные о второй секции:

- Имя секции: .rdata
- RVA начала: 0x00002000
- Внутрифайловое смещение начала: 0x00000400
- Виртуальный размер секции: DCh = 220 байт
- Физический размер секции: 200h = 512 байт

Встроенный в HIEW анализатор секций позволяет проверить правильность найденных данных. Для этого необходимо нажать F8 (Header) и F6 (ObjTbl) (рис. 12)

DOS mode". Но поскольку DOS-заглушка – это полноценное DOS-приложение, не исключено другое её применение.

После DOS-stub'a идет PE-заголовок. PE-заголовок – это общее название структуры под названием IMAGE_NT_HEADERS. Эта структура содержит много основных полей, используемых PE-загрузчиком. В случае, если программа запускается операционной системой, которая знает о PE-формате, PE-загрузчик может найти смещение PE-заголовка в заголовке DOS-MZ. После этого он может пропустить DOS-stub и перейти напрямую к PE-заголовку, который является настоящим заголовком исполняемого файла.

Настоящее содержимое PE-файла разделено на блоки, называемые секциями. Секция – это блок данных с общими атрибутами, такими как код/данные, чтение/запись и т.д. Группирование данных производится на основе их атрибутов. Не играет роли, как используются код/данные, если данные/код в PE-файле имеют одинаковые атрибуты, они могут быть сгруппированы в секцию, т.е. секции могут содержать и данные и код одновременно, главное, чтобы те имели одинаковые атрибуты. Если у вас есть данные, и вы хотите, чтобы они были доступны только для чтения, вы можете поместить эти данные в секцию, помеченную соответствующим атрибутом. Таблица секций, соответственно, располагает данными о каждой секции PE-файла.

Остановимся подробнее на заголовке PE-файла. Адрес PE-заголовка загрузчик получает из DOS-MZ заголовка, который, в свою очередь тоже является структурой. Сама структура IMAGE_NT_HEADERS, представляющая PE-заголовок, определена следующим образом:

```
IMAGE_NT_HEADERS STRUCT
      Signature dd ?
      FileHeader IMAGE_FILE_HEADER <>
      OptionalHeader IMAGE_OPTIONAL_HEADER32 <>
IMAGE_NT_HEADERS ENDS
```

Поле Signature представляет собой двойное слово 50 45 00 00 (ASCII-символы "PE", за которыми следуют два нулевых байта).

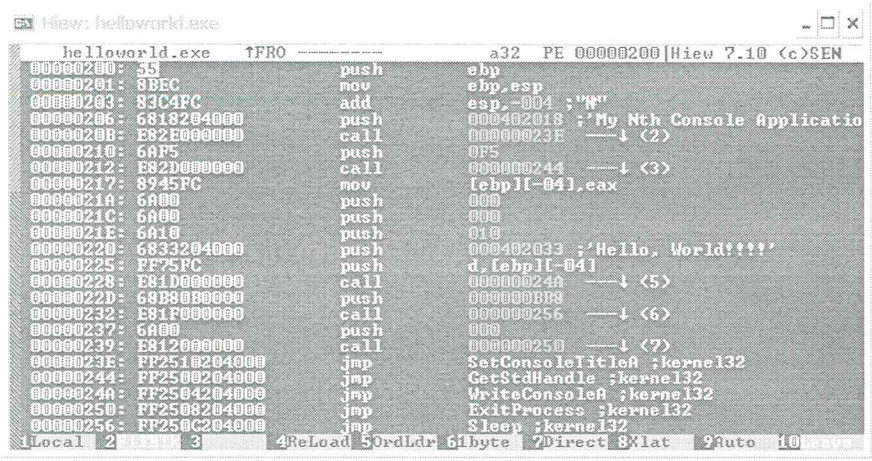

Рис. 9 – Дизассемблированный код приложения

Поскольку по смещению 0x0200 находится точка входа нашего приложения, первой инструкцией является операция помещения в стек указателя базы кадра стека:

```
00000200: 55        push        ebp
```

Системе безразлична конечная цель применения тех или иных данных. Это означает, что если точка входа указывает, скажем, на секцию с данными, то процессор будет трактовать, например, строковые данные как код и пытаться его исполнить, что может привести к непредсказуемым последствиям. Так же и дизассемблер будет пытаться дизассемблировать любой код, начиная с указанного байта, и если инструкция имеет, например, 3-хбайтный опкод, а дизассемблирование начинается со второго байта, то результаты дизассемблирования будут неверными.

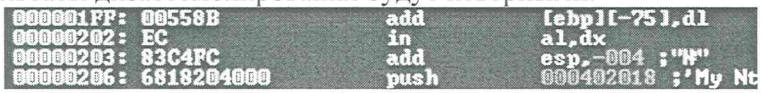

Рис. 10 – Результат неверного дизассемблирования

На рисунке показан пример неверного дизассемблирования. Точка входа имеет адрес 0x200, а дизассемблирование начинается с адреса 0x1FF трёхбайтной инструкцией. Для того, чтобы получить верное отображение файла в декодированном виде, следует

Имя поля	Описание
SectionAlignment	Размер выравнивания секций в памяти. Например, если значение в этом поле равно 4096 (1000h), каждая секция должна начинаться по адресу, кратном этому значению. Если первая секция находится в 401000h и его адрес равен 10 байтам, следующая секция должна начинаться в 402000h, даже если адресное пространство между ними останется неиспользованным.
FileAlignment	Размер выравнивания секций в файле. Например, если значение в этом поле равно 512 (200h), каждая секция должна начинаться на расстоянии от начала файла кратном 512 байтам. Если первая секция в файле находится по смещению 200h и ее размер 10 байт, следующая секция должна быть расположена со смещением 400h: пространство между смещениями 522 и 1024 будет неиспользовано/неопределено.
Minor/Major Subsystem Versions	Версия подсистемы win32. Если PE-файл спроектирован для Win32, версия подсистемы должна быть 4.0, в противном случае диалоговое окно не будет иметь 3D-вида.
SizeOfImage	Общий размер образа PE в памяти. Это сумма всех заголовков и секций, выровненных по SectionAlignment.
SizeOfHeaders	Размер всех заголовков + таблицы секций. То есть это значение равно размеру файла минус комбинированный размер всех секций в файле.
Subsystem	Указывает для какой из подсистем NT предназначен этот PE-файл. Для большинства win32-программ используется только два значения: Windows GUI и Windows CUI (консоль).

смещений между виртуальными адресами и внутрифайловыми смещениями (рис. 6):

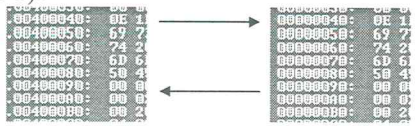

Рис. 6 – Виртуальные и адресные смещения

Имея теоретические сведения о формате PE-файлов и заголовков, можно кратко проанализировать файл (обратите внимание на то, что для указания любых значений используется обратный порядок байтов):

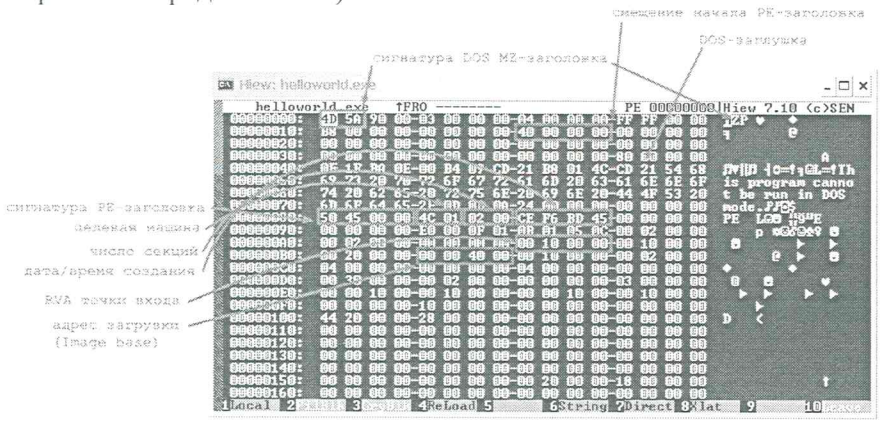

Рис. 7 –Элементы PE-файла в окне дизассемблера HIEW

Встроенный в HIEW анализатор заголовков позволяет проверить правильность найденных вручную значений. Для этого необходимо нажать F8 (Header):


```
.386
 .model flat,stdcall
option casemap:none

includelib kernel32.lib

SetConsoleTitleA PROTO :DWORD
GetStdHandle PROTO     :DWORD
WriteConsoleA PROTO
:DWORD,:DWORD,:DWORD,:DWORD,:DWORD
ExitProcess PROTO      :DWORD
Sleep PROTO            :DWORD
 .const
sConsoleTitle  db 'My Nth Console Application',0
sWriteText  db 'Hello World!!!!'
 .code
Main PROC
  LOCAL hStdout :DWORD
  push offset sConsoleTitle
  call SetConsoleTitleA
  push -11
  call GetStdHandle
  mov hStdout, EAX

  push 0
  push 0
  push 16d
  push offset sWriteText
  push hStdout
  call WriteConsoleA
  push 2000d
  call Sleep
  push 0
  call ExitProcess
Main ENDP
end Main
```