**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

**основы ПОСТРОЕНИЯ**

**ЗАЩИЩЕННЫХ КОМПЬЮТЕРНЫХ СЕТЕЙ**

**домаШнее задание**

Инъекция исполняемого кода

Вариант №1

**Преподаватель:**

Борзых С. С.

**Студенты**:

Нестеров М. С.

Шикунов Ю. И.

Синяков А. С.

ИУ8-94

Москва

2016

Оглавление

Цель работы 3

Задача 3

Выполнение работы 4

I. Выбор исполняемого файла 4

II. Выбор инструментария 5

III. Структура исполняемого файла 7

IV. Написание внедряемого кода 9

V. Внедрение исполняемого кода 10

VI. Проверка работоспособности 13

Выводы 14

# Цель работы

Изучить структуру формата исполняемых файлов «Portable executable».

# Задача

* Выбрать исполняемый файл и необходимый инструментарий.
* Изучить структуру исполняемого файла.
* Написать внедряемый исполняемый код на машинном языке.
* Внедрить исполняемый код в исполняемый файл.

# Выполнение работы

## Выбор исполняемого файла

В качестве исполняемого файла был выбран свободно распространяемый клиент для различных протоколов удаленного доступа «PuTTY», написанный Саймоном Тэтхемом. Выбор данной программы обусловлен тем, что ее исходный код полностью разработан на языке «C» и для запуска не требуется сторонних библиотек. Работа программы представлена на Рисунке 1.

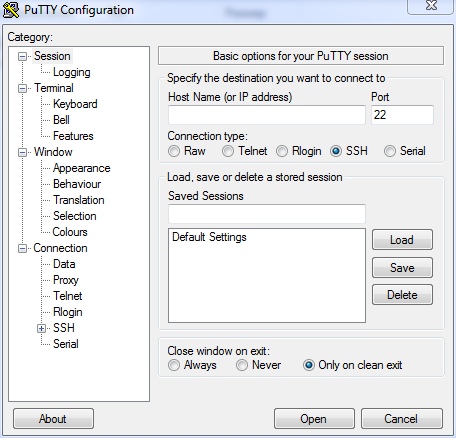


Рисунок 1 – Главное окно «PuTTY»

Название исполняемого файла: putty.exe

Версия программы: 0.66.0.0

## Выбор инструментария

Рисунок 2 – Vmware Viewer – Бесплатное ПО для виртуализации

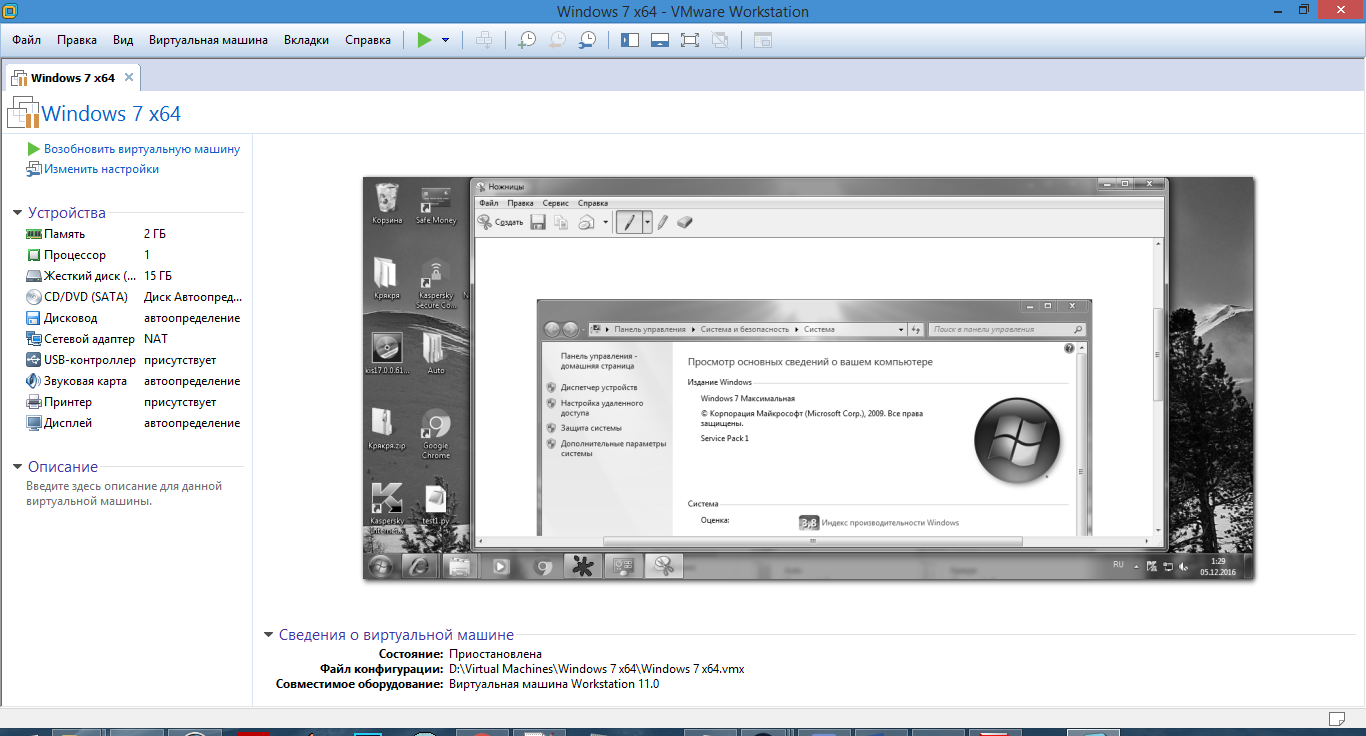
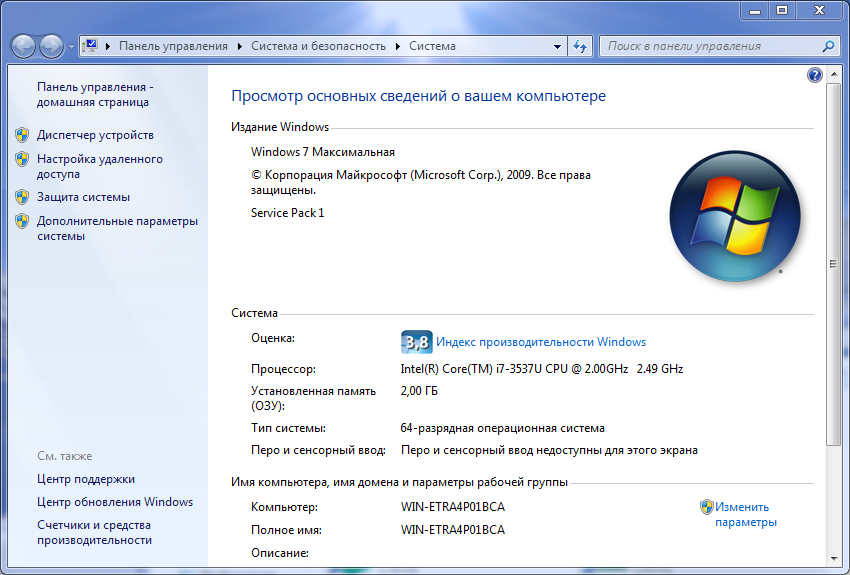


Рисунок 3 –Виртуальная машина с ОС Windows 7

Рисунок 4 – Hiew 7.10 – Редактор двоичных файлов, ориентированный на работу с кодом

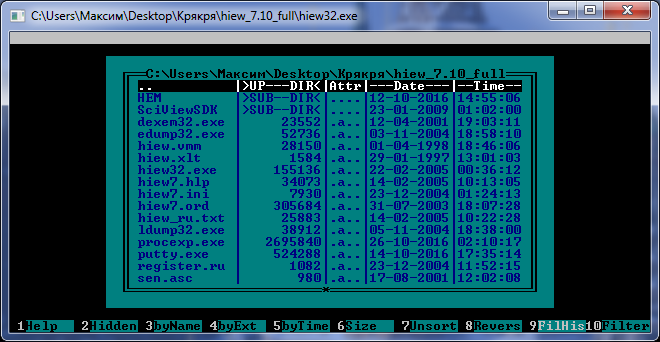


Рисунок 5 – Ht 2.1.0 – Анализатор и редактор исполняемых файлов

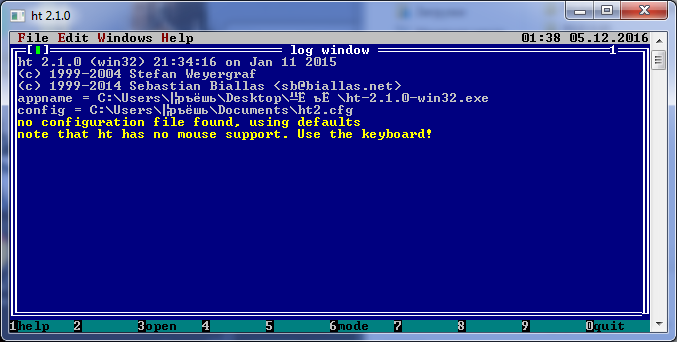
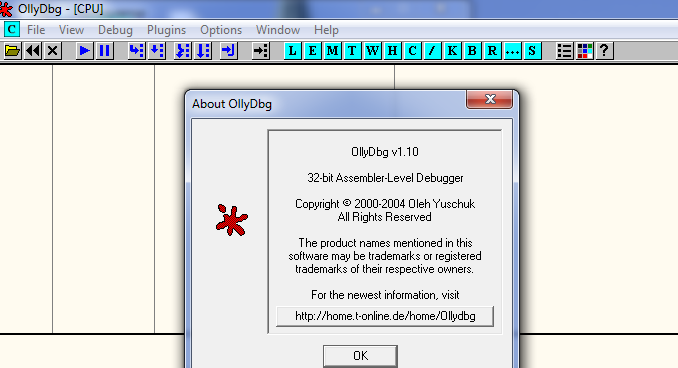
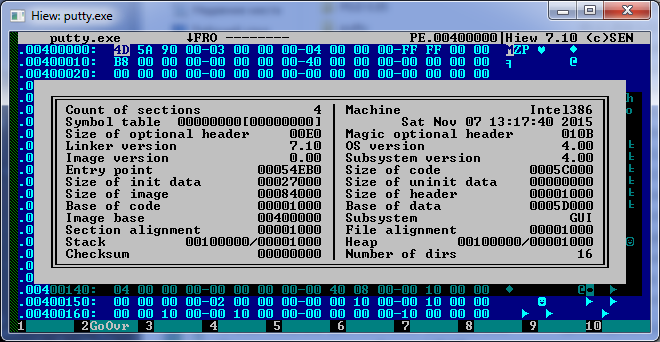


Рисунок 6 – OllyDbg 1.10 – Отладчик уровня ассемблера

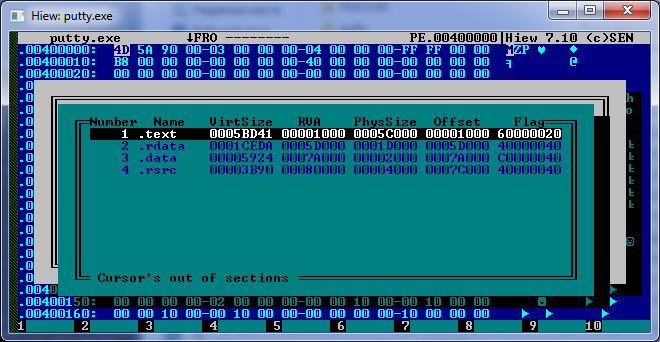
## Структура исполняемого файла

Portable Executable (PE) — формат исполняемых файлов, объектного кода и динамических библиотек, используемый в 32- и 64-разрядных версиях операционной системы Microsoft Windows. Формат PE представляет собой структуру данных, содержащую всю информацию, необходимую PE-загрузчику для отображения файла в память. Исполняемый код включает в себя ссылки для связывания динамически загружаемых библиотек, таблицы экспорта и импорта API функций, данные для управления ресурсами и данные локальной памяти потока (TLS). В операционных системах семейства Windows NT формат PE используется для EXE, DLL, SYS (драйверов устройств) и других типов исполняемых файлов.

Рисунок 7 –Точку входа можно найти в заголовке PE файла.

Файлы PE не содержат позиционно-независимого кода. Вместо этого они скомпилированы для предпочтительного базового адреса, и все адреса, генерируемые компилятором/компоновщиком, заранее фиксированы. Если PE-файл не может быть загружен по своему предпочтительному адресу (потому что он уже занят чем-то ещё), операционная система будет перебазировать его. Для процесса данный факт неизвестен, поскольку в такой ситуации он оперирует в пространстве виртуальных адресов(VA), расположение в котором совпадает с ImageBase. По-умолчанию, значение ImageBase для исполняемых файлов — 0x00400000. Также существует относительный виртуальный адрес (RVA), который упрощает процесс переноса исполняемого кода в адресном пространстве. Адрес секции рассчитывается из RVA отсчитывая от ImageBase исполняемого файла, или, в редких случаях, от “волшебных” констант.

Файл PE состоит из нескольких заголовков и секций, которые указывают динамическому компоновщику, как отображать файл в память. Исполняемый образ состоит из нескольких различных областей (секций), каждая из которых требует различных прав доступа к памяти; таким образом, начало каждой секции должно быть выровнено по границе страницы. Например, обычно секция .text, которая содержит код программы, отображена как исполняемая и доступная только для чтения, а секция .data, содержащая глобальные переменные, отображена как неисполняемая и доступная для чтения и записи.

Рисунок 8 – Секции в файле

## Написание внедряемого кода

Необходимо вызвать код и вернуть управление программе.

В качестве примера внедряемого кода использовался вызов message box

Структура winapi messageboxa

int WINAPI MessageBox(

\_In\_opt\_ HWND    hWnd,

\_In\_opt\_ LPCTSTR lpText,

\_In\_opt\_ LPCTSTR lpCaption,

\_In\_     UINT    uType

);

Дескриптор окна ставим «0».

Параметр «uType» - «0». Это кнопка «ОК».

Псевдокод на ассемблере:

**push 0**

**push lpCaption**

**push lpText**

**push 0**

**call USER32.MessageBoxA**

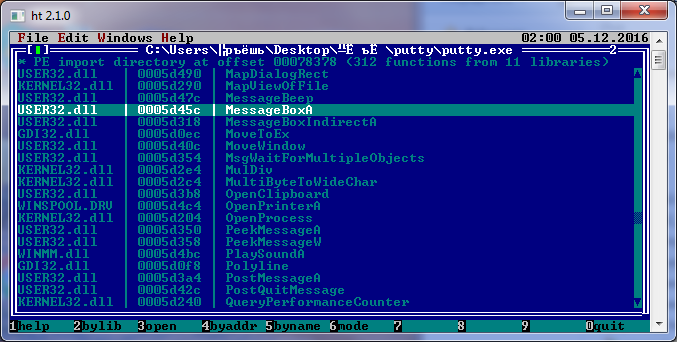
**jmp ExEntryPoint**

где:

* lpCaption, lpText – VA на нуль-терминированную ASCII строку
* USER32.MessageBoxA – адрес функции в таблице импорта
* ExEntryPoint – VA точки входа в приложении, которую мы поменяли

## Внедрение исполняемого кода

При помощи программы Ht смотрим функции которые содержатся в таблице импорта.   
Рисунок 9 – Находим MessageBox.



Запоминаем адрес данной функции.

Рисунок 10 – Находим свободное место в секции «.rdata»

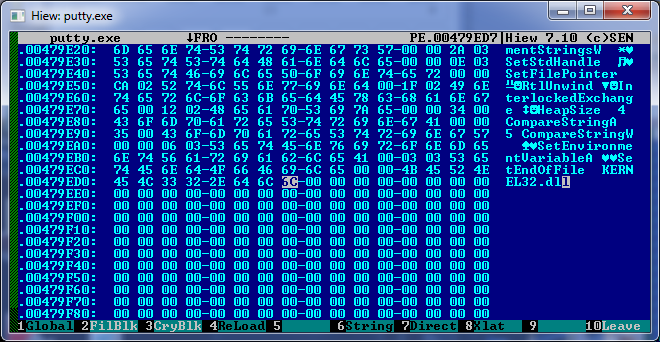
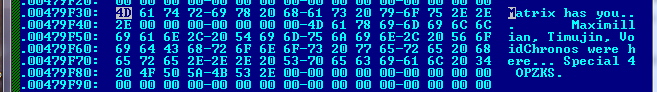


Рисунок 11–Запишем заголовок и текст, который будет напечатан в MessageBox



Запустив программу в OllyDBG узнаем что текущий адрес инструкции содержится в регистре edx.

Рисунок 12 – Отладка программы в OllyDBG

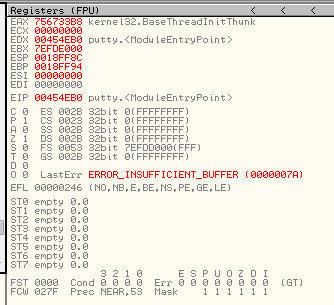
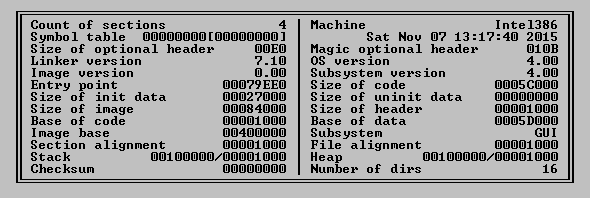
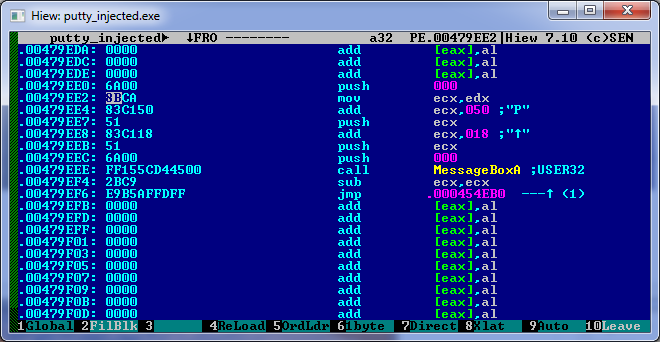


Рисунок 13 – Меняем точку входа на 79ee0



Запишем внедряемый код. Рассчитываем положение строк относительно edx. Jump на исходную точку входа – возвращаем управление.   
Рисунок 14 – Hiew автоматом переводит ассемблер в опкод.

## Проверка работоспособности

Рисунок 15 – Внедренный MessageBox с текстом

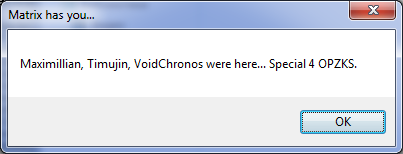
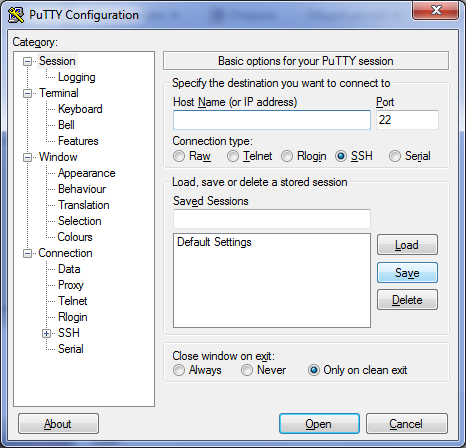


Рисунок 16 – Главное окно «PuTTY»



После запуска программы выводится текстовое окно MessageBox с указанным нами текстом, после чего запускается основная программа, сохраняя полный функционал.

# Выводы

В ходе работы была изучена структура формата исполняемых файлов ОС Windows – PE. Были получены навыки работы с инструментами для отладки и редактирования исполняемых файлов.