Kalil Armstrong
CSCE 489-501
Spring 2015

Assignment 5 Report and Documentation

Introduction

Assignment five built largely off of the work that was done in assignment four. Assignment four gave us XED, a robust decoder and encoder tool that allows us to do deep analysis of binary files. To extend on this assignment, I've decided to build a graphical user interface to accompany the usage of XED, as well as implement a search feature that allows you to find particular registers or HEX strings within the assembly dump file.

I originally meant to provide support for both the .text and .data sections, but in the current implementation, the only thing that is supported is the text section. In place of the data section is the console output from XED's statistical analysis. This left the number of tools utilized to be fairly few - instead of using a wrapper, we simply used XED and wrote our own wrapper within a program.

Disassembler Features:

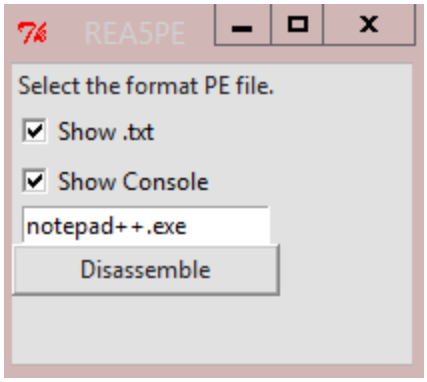
- Decodes any PE file.
- Provides console output for the decoding process.
- Provides statistical information regarding the decoding process, including, but not limited to, number of instructions decoded, number of decode errors, and the number of invalid instructions.
- Built-in wrapper for XED from a Python-based script.
- Search feature implemented that allows you to search for a decoded instruction or HEX string and it will highlight all occurrences within the file.

Disassembler Requirements:

- Python 2.7 (not Python 3)
- XED (the EXE available from Intel)
- PE file for testing
- Permissions to create text files.

Usage of this disassembler is extremely straight-forward and simple. The Python script handles most of the work for you, so you simply have to navigate to the directory containing the Python script and XED and execute it. It's imperative that the XED executable be in the same directory as the Python script.

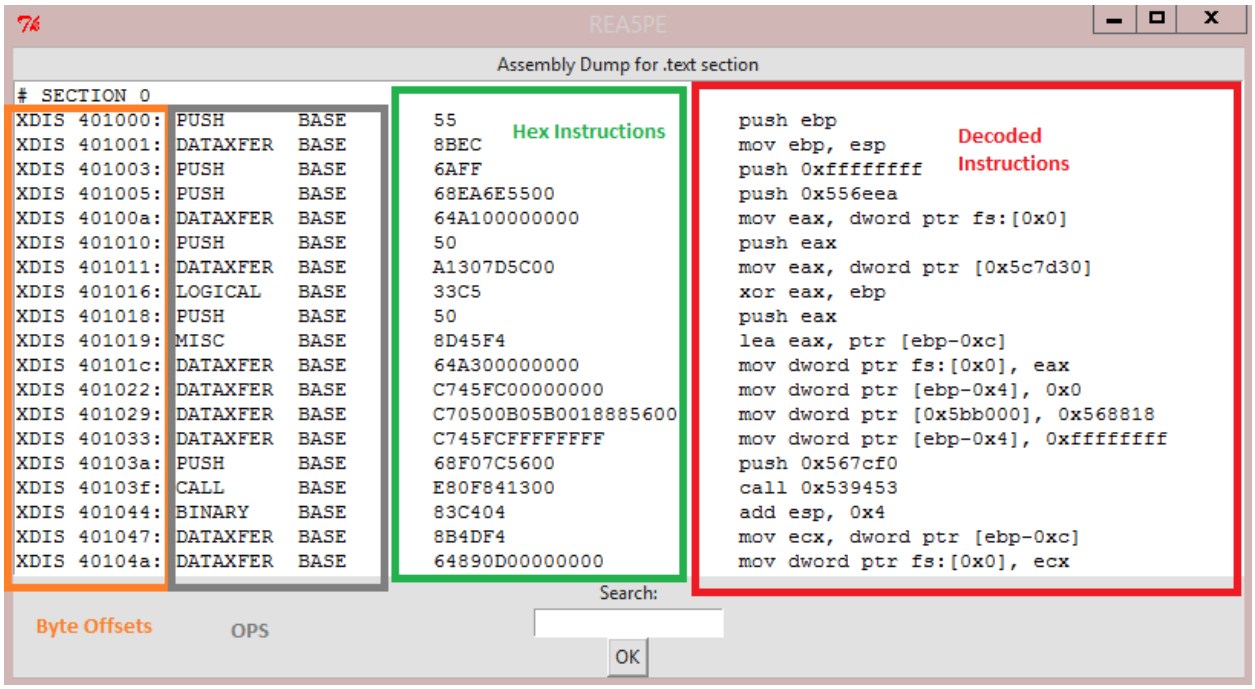
Upon successful launching, it will provide you with a dialog box:




```
PS C:\Users\Noc> python REfile.py
Attempting to launch file: notepad++.exe
Saving to file....
done
Displaying text section only.
```

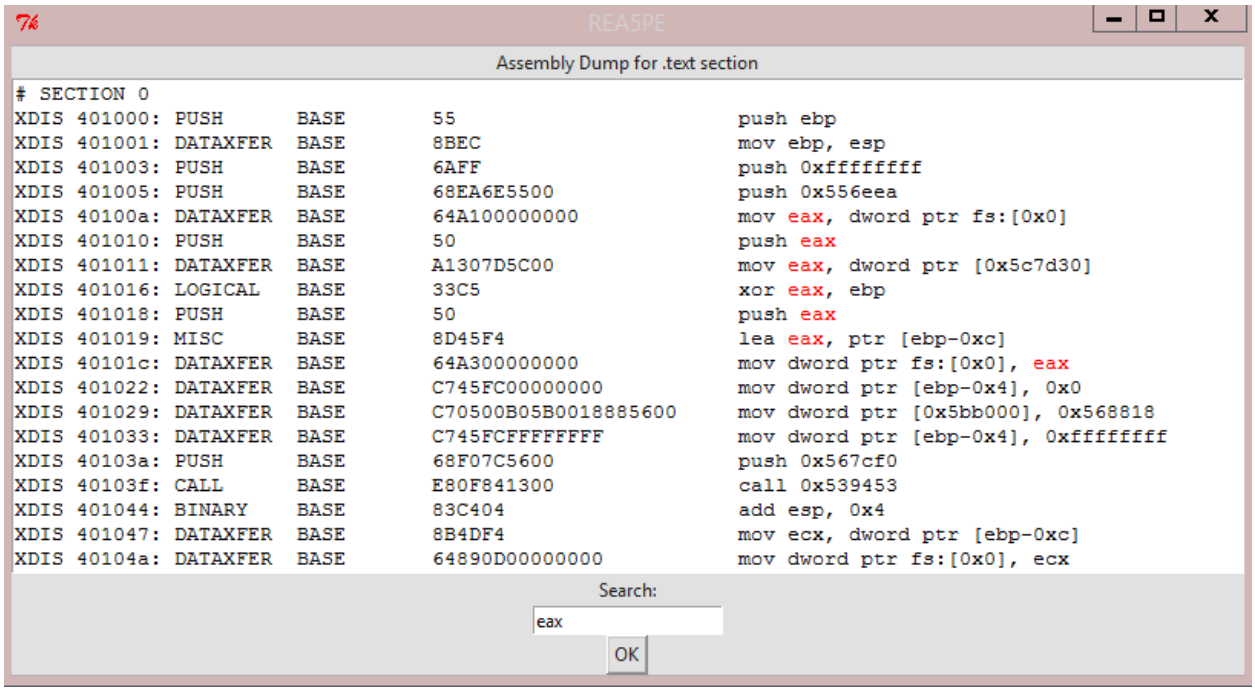
The first option allows you to see the text section, while the second option will allow you to see the statistical analysis by means of the console output. If you select both, both options will be provided for your use. The entry bar is for the name of the executable you would like to disassemble. In my preliminary testing, I disassembled Notepad++.

Upon clicking disassemble, you will get a confirmation of sorts depending on which option you chose, whether it be text, console, or text and console. In this particular walkthrough, we chose text only. By selecting this option, we're prompted with dialog that contains several aspects of the PE file. We can see the Hex Instructions, the operations, the byte offsets, and the decoded instructions. These are all found by use of XED, which is executed in the background by the Python application.




```
# SECTION 0
XDIS 401000:  PUSH       BASE      55                  push ebp
XDIS 401001:  DATAXFER   BASE      8BEC                mov ebp, esp
XDIS 401003:  PUSH       BASE      6AFF                push 0xffffffff
XDIS 401005:  PUSH       BASE      68EA6E5500          push 0x556eea
XDIS 40100a:  DATAXFER   BASE      64A100000000        mov eax, dword ptr fs:[0x0]
XDIS 401010:  PUSH       BASE      50                  push eax
XDIS 401011:  DATAXFER   BASE      A1307D5C00          mov eax, dword ptr [0x5c7d30]
XDIS 401016:  LOGICAL    BASE      33C5                xor eax, ebp
XDIS 401018:  PUSH       BASE      50                  push eax
XDIS 401019:  MISC       BASE      8D45F4              lea eax, ptr [ebp-0xc]
XDIS 40101c:  DATAXFER   BASE      64A300000000        mov dword ptr fs:[0x0], eax
XDIS 401022:  DATAXFER   BASE      C745FC00000000      mov dword ptr [ebp-0x4], 0x0
XDIS 401029:  DATAXFER   BASE      C70500B05B0018885600 mov dword ptr [0x5bb000], 0x568818
XDIS 401033:  DATAXFER   BASE      C745FCFFFFFFFF      mov dword ptr [ebp-0x4], 0xffffffff
XDIS 40103a:  PUSH       BASE      68F07C5600          push 0x567cf0
XDIS 40103f:  CALL       BASE      E80F841300          call 0x539453
XDIS 401044:  BINARY     BASE      83C404              add esp, 0x4
XDIS 401047:  DATAXFER   BASE      8B4DF4              mov ecx, dword ptr [ebp-0xc]
XDIS 40104a:  DATAXFER   BASE      64890D00000000      mov dword ptr fs:[0x0], ecx
```

Furthermore, when we issue a search query at the bottom to the application, we're given the ability to find particular registers or HEX instructions within the dump for our own browsing purposes. If we find any occurrences of the string within the dump file, it gets highlighted in red.



Final Project

This assignment is quite a bit different from my final project, but it establishes a good background for the flexibility and dependency one can have on their tools. For my project in particular, I ended up using PANDA, which is a virtualization software capable of emulating different processor architectures. However, this particular assignment gives good insight on the structure of PE files.