

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.

File Settings View Compare Info

Lesson1.exe

- DOS Header
- NT Headers
 - Signature
 - File Header
 - Optional Header
- Section Headers
 - .text
 - EP = 840
 - .rdata
 - .data
 - .pdata
 - .src
 - .reloc

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
400	7C	48	83	84	F0	E8	C0	00	00	00	41	51	41	50	52	51	0H	aa	aa	A	A	O	A	P	R	O					
410	56	48	31	D2	45	48	58	52	40	48	58	52	18	40	58	52	V	H	i	O	=	H	.	R	.	H	.	R	.	H	.	R			
420	20	48	8B	72	50	48	0F	B7	4A	4A	4D	31	C9	40	31	C0	.	H	.	x	F	H	.	.	J	M	I	E	H	I	A				
430	AC	3C	41	7C	02	2C	20	41	C1	C9	0D	41	01	C1	E2	ED	=	<	A	I	A	A	E	.	A	.	A	A			
440	52	41	51	48	8B	52	20	8B	42	9C	48	01	D0	8B	80	88	R	A	Q	H	B	<	H	.	B	.	.				
450	00	00	00	48	85	C0	74	E7	48	01	D0	50	8B	40	18	44	.	.	.	H	.	A	.	t	g	H	.	B	.	P	.	H			
460	8B	40	20	45	01	D0	E3	56	48	FF	C9	41	8B	34	88	48	.	.	.	B	.	E	.	V	H	.	y	E	.	A	.	.			
470	01	D6	4D	31	C9	40	AC	41	C1	C9	0D	41	01	C1			.	O	H	I	E	H	I	A	~	A	A	E	.	A	.	A			
480	38	E0	75	F1	4C	03	4C	24	08	45	39	D1	75	D8	58	44	B	.	A	u	a	R	.	L	.	s	.	E	9	B	.	u	O	X	D
490	8B	40	24	45	01	D0	6C	41	8B	0C	48	44	8B	40	1C	45	.	.	E	.	I	.	B	.	E	.	A	.	.	H	D	.	.	.	
4A0	01	D0	41	8B	04	88	01	D0	41	58	41	58	41	58	5X	59	5A	.	D	A	.	.	.	H	.	.	D	A	X	A	X	^	Y	Z	
4B0	41	58	41	59	41	5A	48	83	EC	20	41	52	FF	E0	58	41	A	X	A	Y	A	Z	.	.	.	A	.	R	.	A	.	Y	A	X	A
4C0	59	5A	48	8B	12	E9	57	FF	FF	5D	48	8A	01	00	00		Y	Z	H	
4D0	00	00	00	00	00	48	8D	8D	01	01	00	00	41	8A	31	8B	
4E0	6F	87	FF	D5	8B	E0	1D	2A	0A	41	8A	A6	95	8D	9D	FF	
4F0	D5	48	83	C4	28	3C	06	7C	0A	80	FB	E0	75	05	BB	47	O	H	.	A	
500	13	72	6F	4A	00	55	41	89	DA	FF	D5	43	61	4C	63	00	.	E	.	o	
510	48	8D	05	19	25	00	00	C3	CC	CC	CC	CC	CC	CC	CC	CC	H	
520	48	89	4C	24	00	48	89	54	24	10	4C	89	44	24	18	4C	H	.	L	.	s	
530	89	4C	24	20	53	56	57	48	83	EC	30	48	8B	T9	48	8D	.	L	.	s	

Disasm: .text General DOS Hdr Rich Hdr File Hdr Optional Hdr Section Hdrs Imports Resources Exception BaseReloc. Debug LoadConfig

	Hex	Disasm	Hint
1000	9C	CJD	
1001	4882E4F0	AND ESP, 0xFFFFFFFFFFFFF0	
1005	E9C0000000	CALL 0X1400010CA	
100A	4151	PUSH B9	
100C	4150	PUSH B9	
100E	52	PUSH RDX	
100F	51	PUSH RDX	
1010	54	PUSH RSI	
1011	4831D0	XOR RDX, RDX	
1014	66498B81240	MOV RDX, QWORD PTR GS:[RDX + 0X40]	
1019	488B210	MOV RDX, QWORD PTR [RDX + 0X18]	
101D	488B220	MOV RDX, QWORD PTR [RDX + 0X20]	
1021	488B230	MOV RDX, QWORD PTR [RDX + 0X30]	
1025	488B74A4A	MOVZX RDX, WORD PTR [RDX + 0X4A]	
102A	4D81C9	XOR R9, R9	
102D	4981C0	XOR RAX, RAX	
1030	AC	LOCKB AL, BYTE PTR [RDI]	
1031	3C41	CMF AL, 0X41	
1033	7C02	JL SHORT 0X140001037	
1035	2C20	SUB AL, 0X20	
1037	41010000	INC RAX, RAX	