



메모리 변조를 이용한 게임 핵 개발

with WPM, RPM



중부대학교 SCP 정보보안 동아리
1학년 부원 노무승

목 차

0. 법률

1. 기본 개념

2. 메모리 변조 시연

3. 게임 핵 프로그램 개발

(WriteProcessMemory, ReadProcessMemory API)

4. 메모리 변조 대처방안



0. 법률

0-1. 법률

형법 제314조(업무방해) ① 제313조의 방법 또는 위력으로써 사람의 업무를 방해한 자는 5년 이하의 징역 또는 1천500만원 이하의 벌금에 처한다.

② 컴퓨터등 정보처리장치 또는 전자기록등 특수매체기록을 손괴하거나 정보처리장치에 허위의 정보 또는 부정한 명령을 입력하거나 기타 방법으로 **정보처리에 장애를 발생하게 하여** 사람의 업무를 방해한 자도 제1항의 형과 같다. <신설 1995.12.29>

게임산업진흥에 관한 법률 제32조(불법게임물 등의 유통금지 등) ① 누구든지 게임물의 유통질서를 저해하는 다음 각 호의 행위를 하여서는 아니 된다.

8. 게임물의 정상적인 운영을 방해할 목적으로 게임물 관련사업자가 제공 또는 승인하지 아니한 컴퓨터프로그램이나 기기 또는 장치를 배포하거나 배포할 목적으로 제작하는 행위

게임산업진흥에 관한 법률 제46조(벌칙) 다음 각 호의 어느 하나에 해당하는 자는 1년 이하의 징역 또는 1천만원 이하의 벌금에 처한다.

3의2. 제32조제1항제8호를 위반하여 게임물 관련사업자가 제공 또는 승인하지 아니한 컴퓨터프로그램이나 기기 또는 장치를 배포하거나 배포할 목적으로 제작하는 행위를 한 자

정보통신망 이용촉진 및 정보보호 등에 관한 법률 제48조 (정보통신망 침해행위 등의 금지)

② 누구든지 정당한 사유 없이 정보통신시스템, 데이터 또는 프로그램 등을 훼손·멸실·변경·위조하거나 그 운용을 방해할 수 있는 프로그램(이하 "악성프로그램"이라 한다)을 전달 또는 유포하여서는 아니 된다.

정보통신망 이용촉진 및 정보보호 등에 관한 법률 제70조의2 (벌칙)

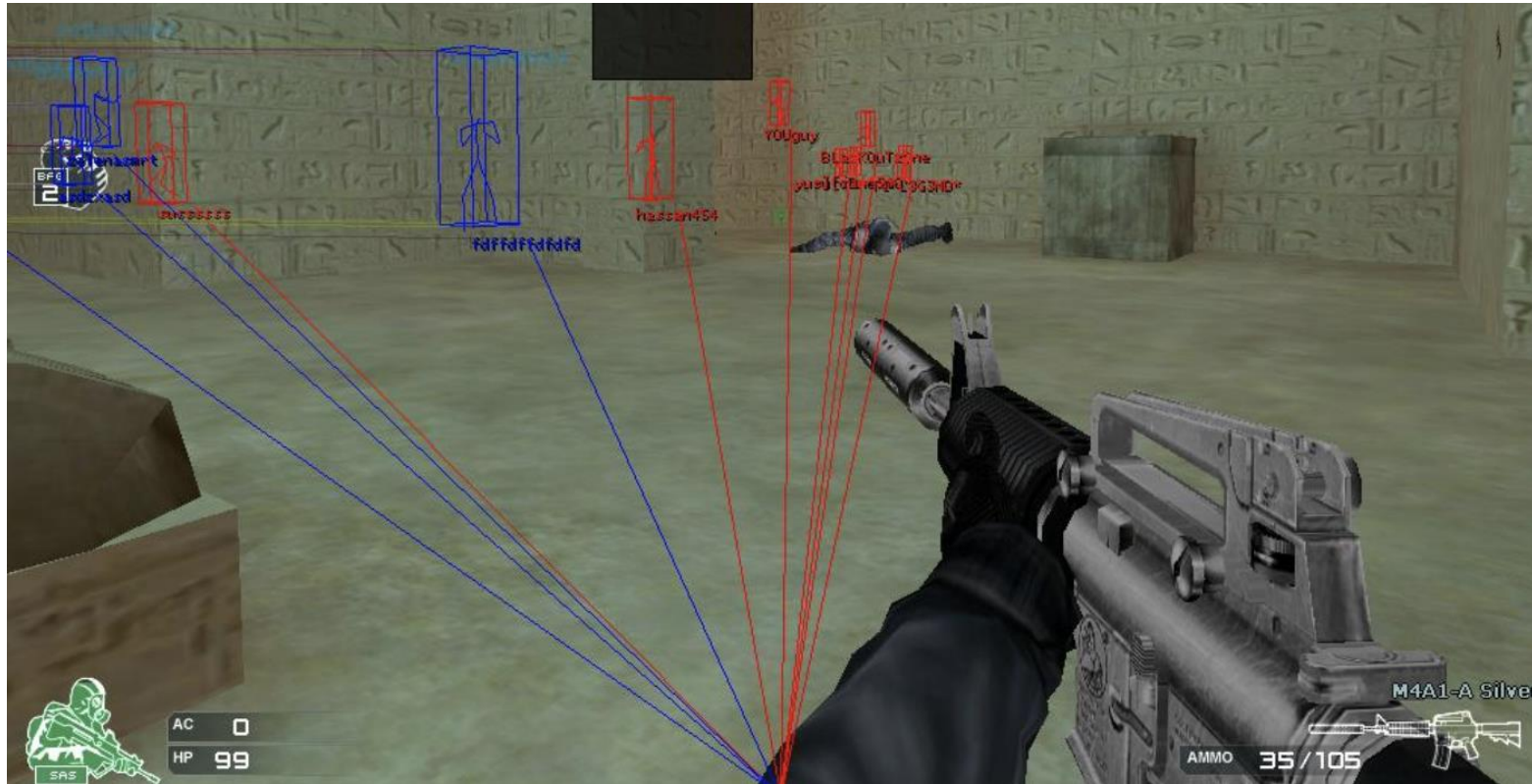
제48조 제2항 을 위반하여 악성프로그램을 전달 또는 유포하는 자는 7년 이하의 징역 또는 7천만원 이하의 벌금에 처한다.

게임핵을 제작하거나, 팔거나, 공유시, 철컹철컹



1. 기본 개념

1-1. 게임 해이란?



5. 게임별 해

- 스피드해^[12]
- 에임해^[13]
- 탄속해^[14]
- 맵해^[15]
- 고스트해^[16]
- 블랙홀해^[17]
- 대미지/쿨감해^[18]
- 관통해^[19]
- 무적해^[20]
- 아이템해^[21]
- 돈해^[22]

게임 내 비인가된 해킹 프로그램

종류에 따라 제작하는 방식이 달라 메모리 변조에 대해서만 말하고자 함.

1-2. 메모리 변조란?

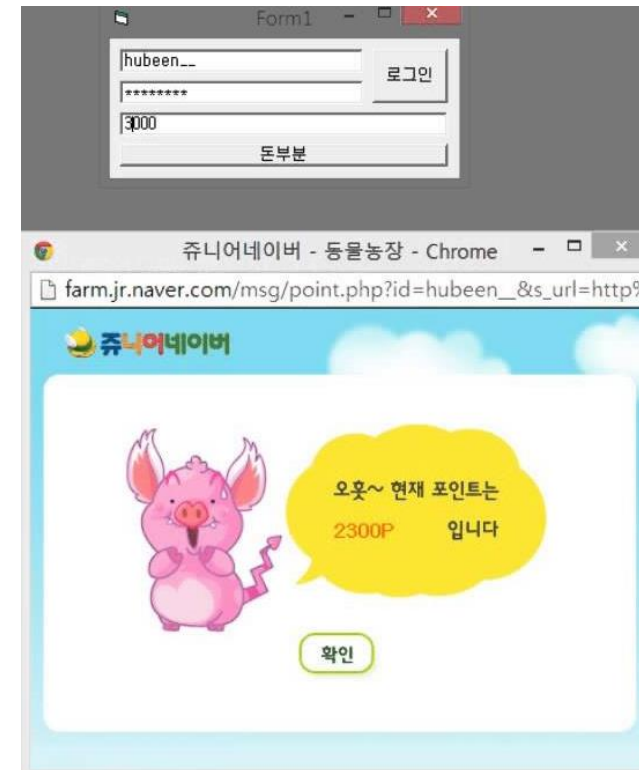
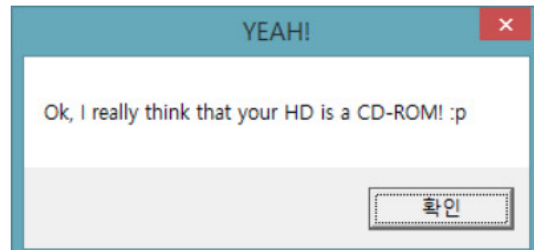
프로그램의 메모리 값을 변조하여 비정상적으로 동작하게 하는 것

```
Registers (FPU)
EAX 00000001
ECX 84109E57
EDX 00510200
EBX 7FFDE000
ESP 0018FF84
EBP 0018FF94
ESI 00401003 PTR to A
EDI 00401000 01.<Modu
```

브레이크 포인트가 걸릴때 EAX와 ESI 레지스터 모습입니다.

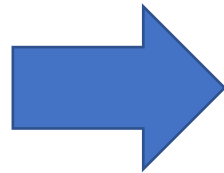
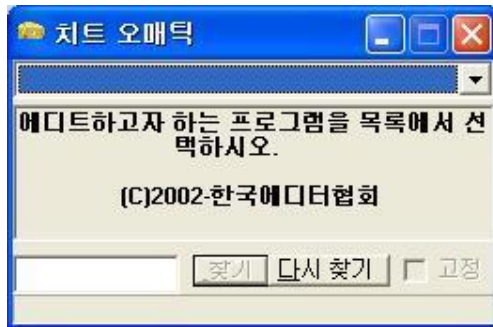
```
Registers (FPU)
EAX 00000001
ECX 84109E57
EDX 00510200
EBX 7FFDE000
ESP 0018FF84
EBP 0018FF94
ESI 00000001
EDI 00401000 01.<Modu
```

이때 EAX와 ESI 레지스터의 값을 같게 만들어 주면 됩니다.



1-3. 메모리 변조 프로그램

메모리 변조 기능을 제공하는 프로그램으로
사실상 디스어셈블러(Disassembler)



Cheat Engine



1-3. 메모리 변조 원리

<정상적인 동작 구조>

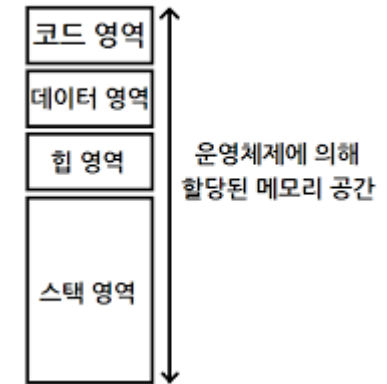
```
1  #include <stdio.h>
2  #include <stdint.h>
3
4  int main(){
5      uint8_t money;
6      // 1byte int 자료형
7      money = 10;
8      printf("%d", money);
9  }
```

0x12345678 => 0x00

0x12345678 => 0x0A

0x12345678

10



1-3. 메모리 변조 원리

<메모리 변조 후 동작 구조>

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 int main(){
5     uint8_t money;
6     // 1byte int 자료형
7     money = 10;
8     printf("%d", money);
9 }
```

0x12345678 => 0x00

0x12345678 => 0x0A

0x12345678 => 0xFF

0x12345678

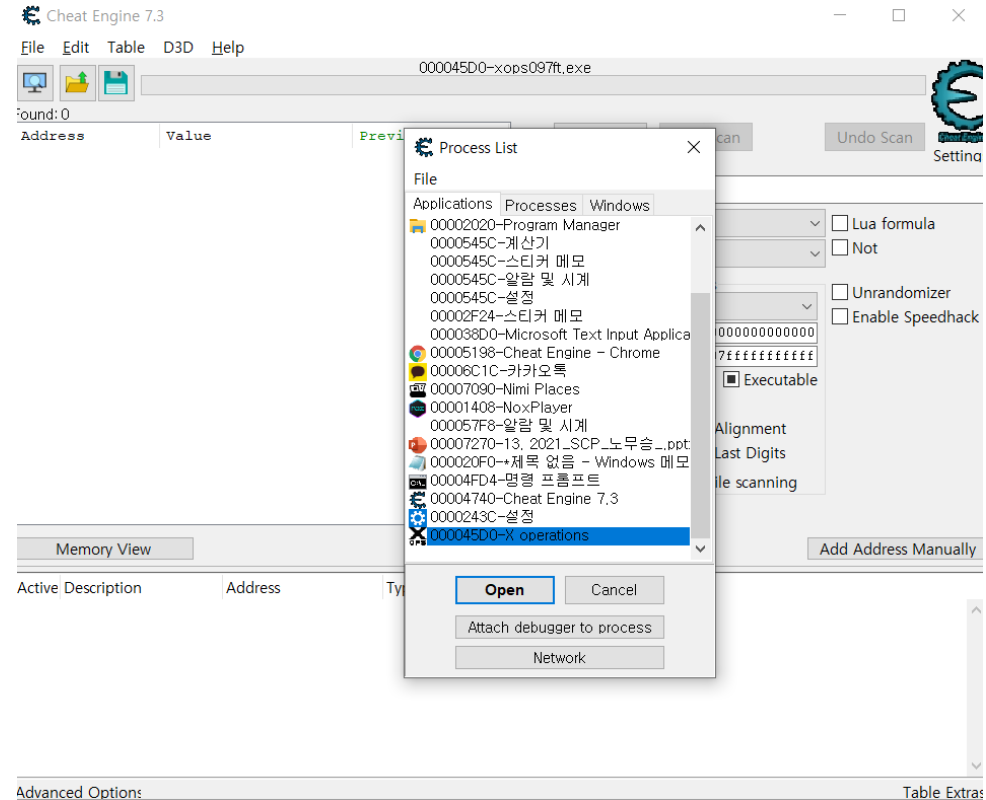
255

메모리 변조
프로그램



2. 메모리 변조 시연

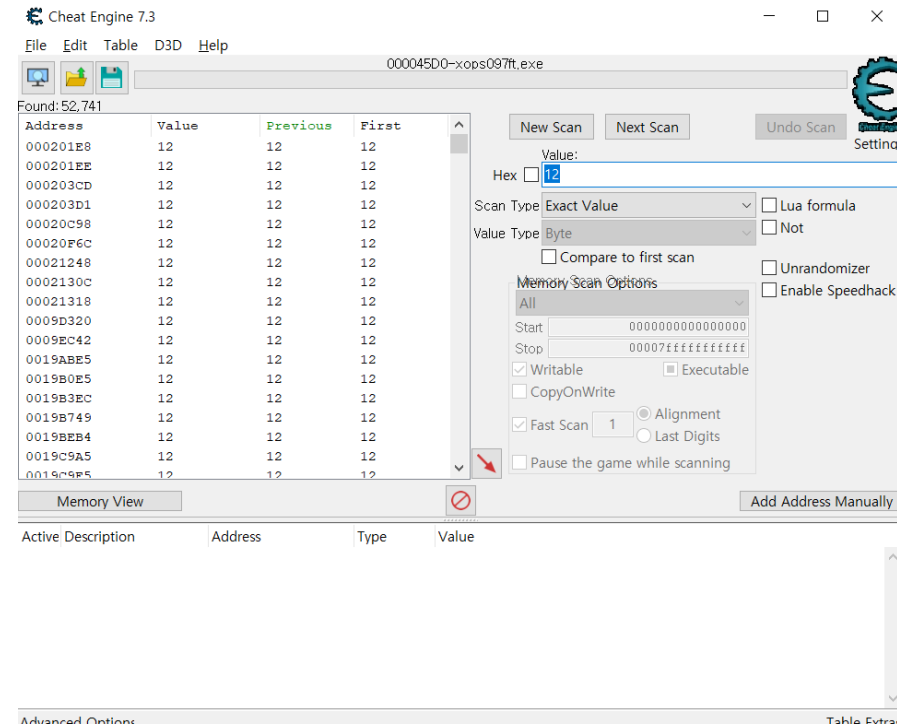
2-1. 실습 대상



실습 대상은 XOperation이라는 미니 FPS 게임.

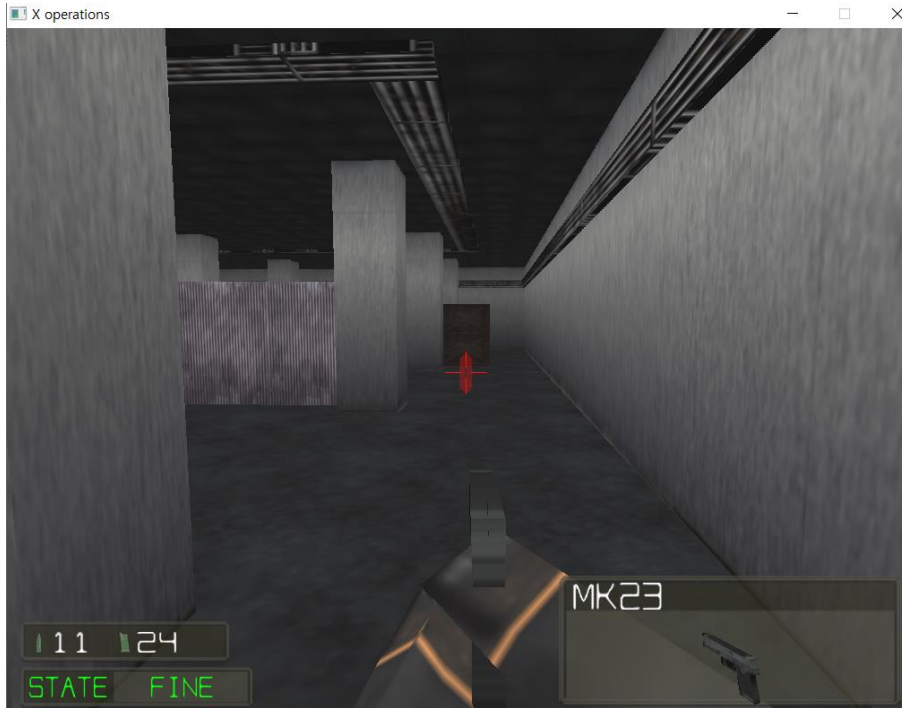
오프라인 게임이고 치트가 공식적으로 제공되기 때문에 문제의 소지가 적다.

2-2. 총알 수 : 초기 값 검색



현재 게임에서의 총알 개수는 12발로,
치트엔진에서 Byte 타입의 12 값을 가지는 메모리 주소를 찾는다

2-3. 총알 수 : 범위 축소



Found: 13

| Address | Value | Previous | First |
|-----------------|-------|----------|-------|
| xops097ft.ex... | 11 | 11 | 12 |
| 035E3A49 | 11 | 11 | 12 |
| 035E3A59 | 11 | 11 | 12 |
| 035E3A69 | 11 | 11 | 12 |
| 035E3A79 | 11 | 11 | 12 |
| 037F3A49 | 11 | 11 | 12 |
| 037F3A59 | 11 | 11 | 12 |
| 037F3A69 | 11 | 11 | 12 |
| 037F3A79 | 11 | 11 | 12 |
| 037FBA49 | 11 | 11 | 12 |
| 037FBA59 | 11 | 11 | 12 |
| 037FBA69 | 11 | 11 | 12 |
| 037FBA79 | 11 | 11 | 12 |

New Scan Next Scan

Value:

Hex ☐ 11

Scan Type

Value Type

☐ Compare to first scan

Memory Scan Options

All

Start

Stop

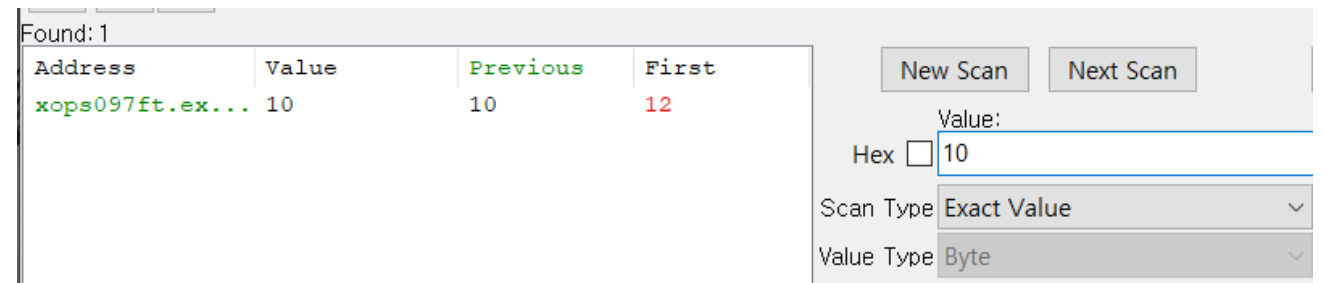
☒ Writable ☐ Executable

☐ CopyOnWrite

☒ Fast Scan ☐ Alignment ☐ Last Digits

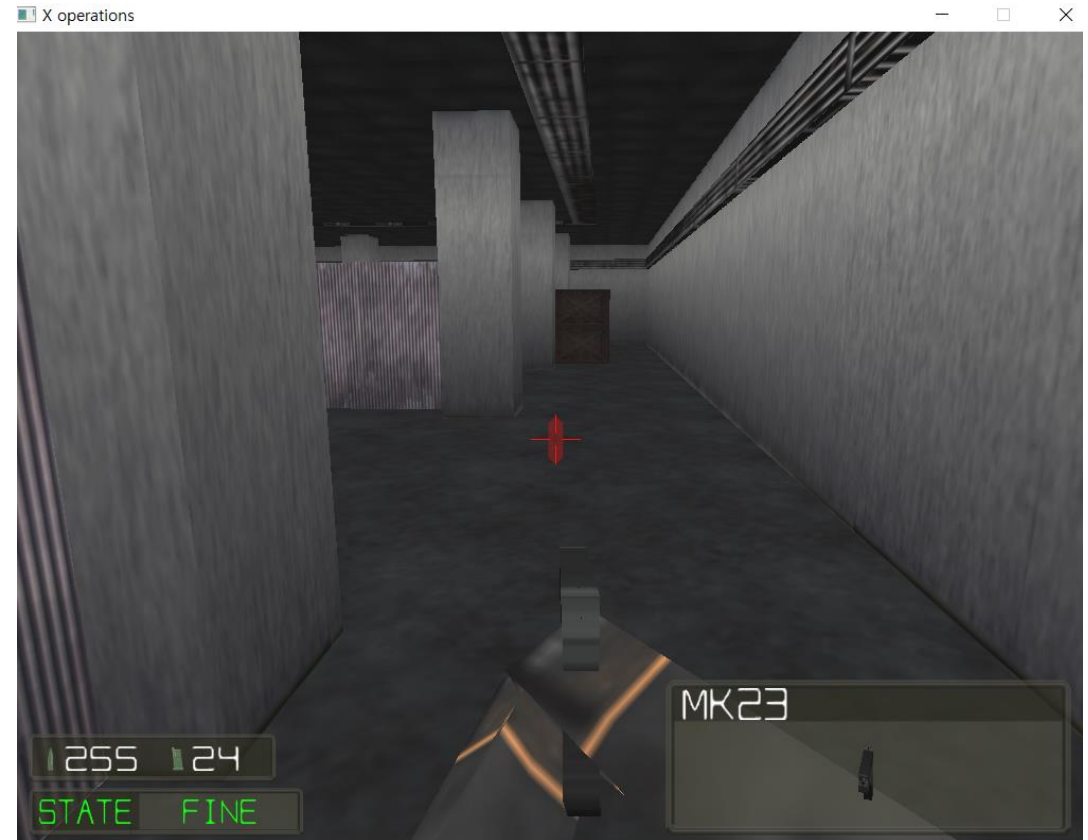
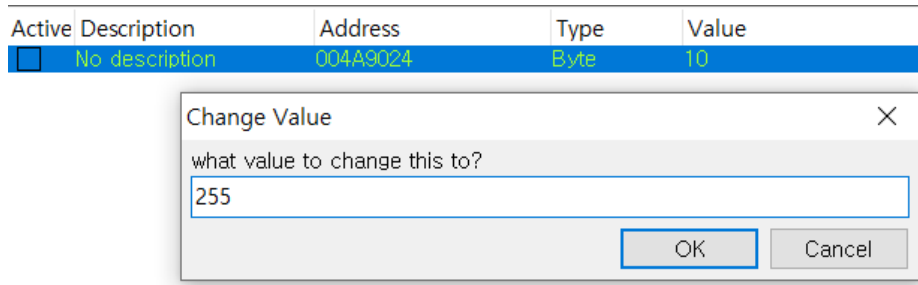
총을 한발 쏜 11발로 만든 후,
치트엔진에서 12에서 11 값으로 바뀐 메모리 주소를 찾는다.

2-3. 총알 수 : 범위 축소



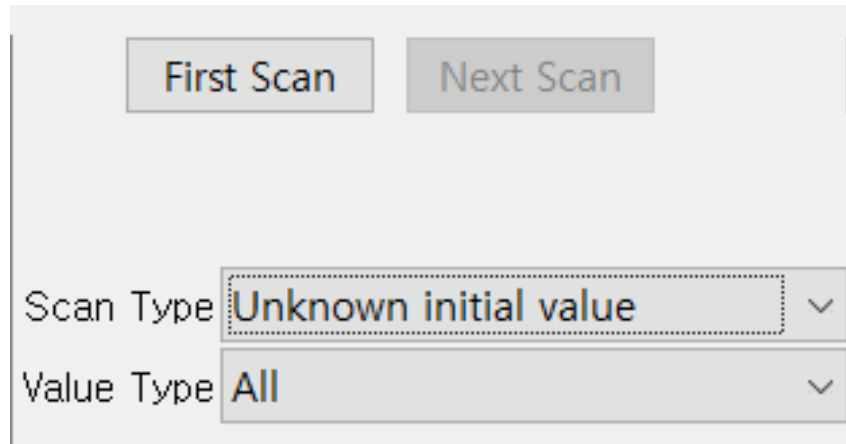
이전 작업을 반복해
총알 값을 가지는 메모리 주소 하나를 찾아낸다.

2-4. 총알 수 : 메모리 변조



치트엔진을 통해 해당 메모리 주소의 값을
255로 수정해주면 총알의 개수가 255로 변함을 알 수 있다.

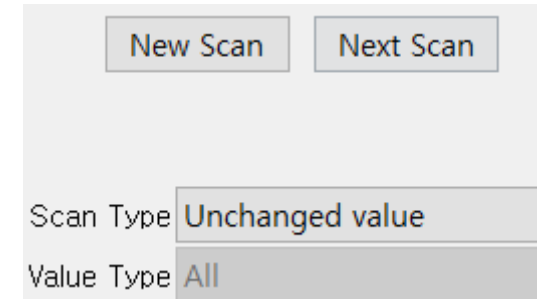
2-5. 목숨 값 : 초기 값 및 범위 축소



First Scan Next Scan

Scan Type Unknown initial value

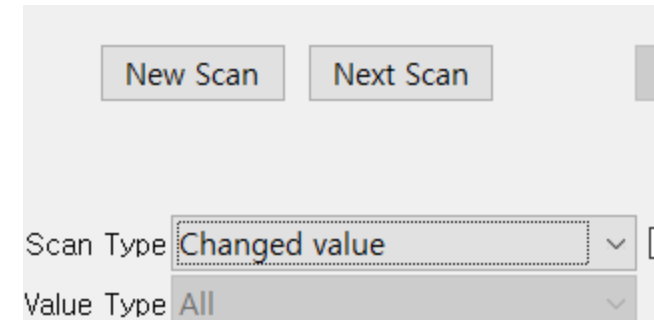
Value Type All



New Scan Next Scan

Scan Type Unchanged value

Value Type All



New Scan Next Scan

Scan Type Changed value

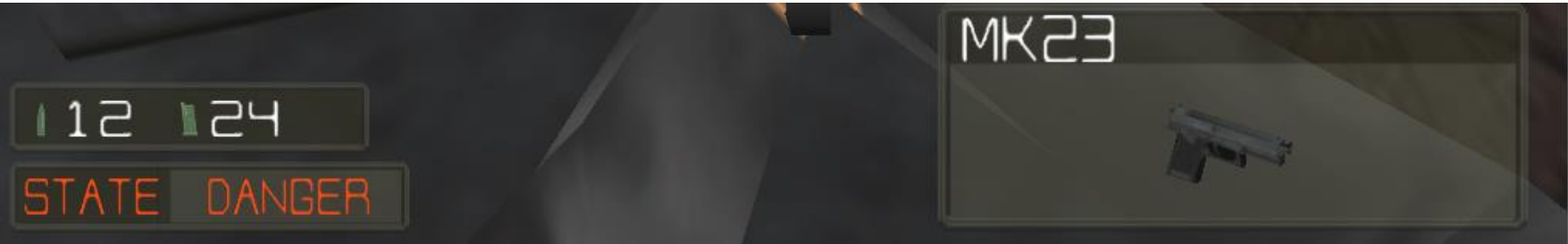
Value Type All

목숨 값의 초기 값을 모르기 때문에 Unknown으로 두고
데미지를 입는 과정에서 변하는 값만 남기는 것을 계속 반복한다.

2-6. 목숨 값 : 메모리 주소 확인



4 Bytes 120



4 Bytes 12



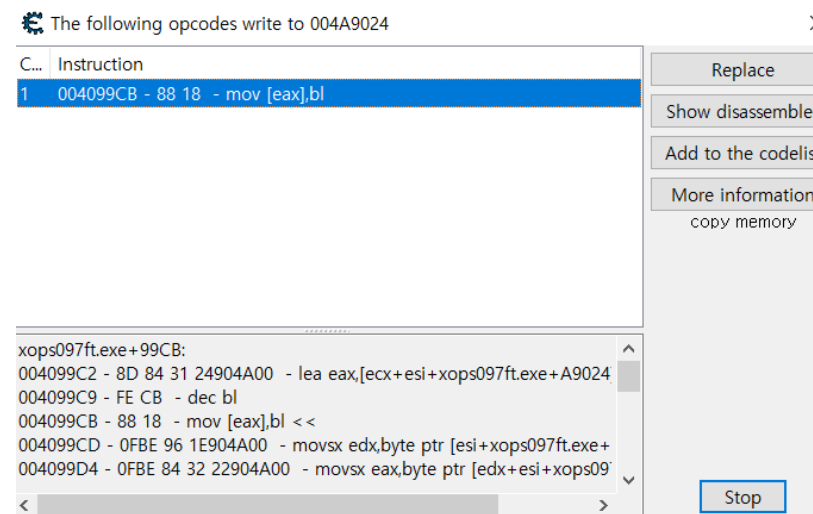
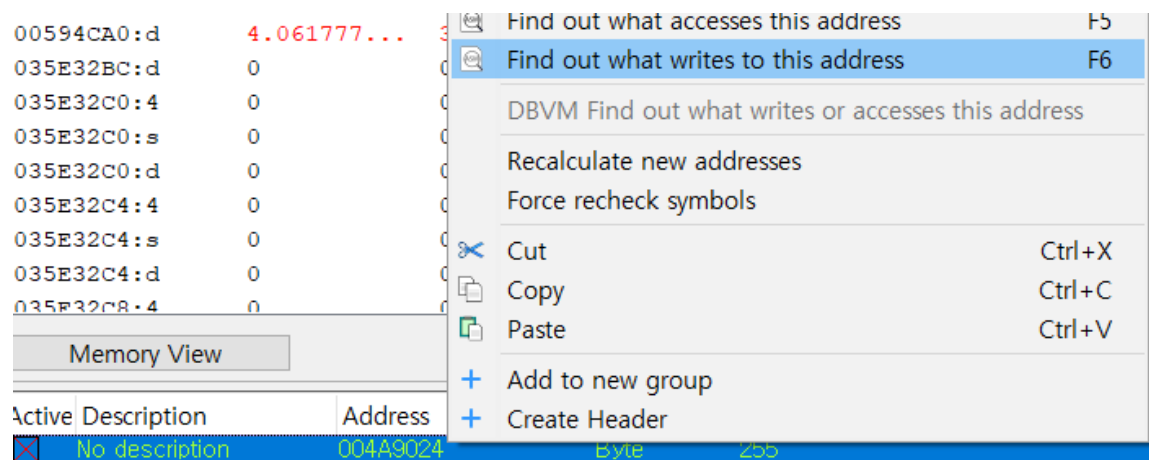
No description
No description

OUTROLET
004A901C

4 Bytes 12
4 Bytes 12

2-7. 어셈블리 코드 찾기

할당된 스택 메모리 주소는 프로그램이 실행될 때 매번 바뀜.
따라서 해당 메모리 주소에 접근하는 어셈블리 코드를 찾을 필요가 있음.



2-8. 어셈블리 코드 패치

해당 어셈블리 코드가 총알 수를 감소시키는 명령어기 때문에
기존 코드를 nop(0x90)로 바꿈.

Memory Viewer

File Search View Debug Tools Kernel tools

xops097ft.exe+99CB

| Address | Bytes | Opcode | Comment |
|--------------------|---------------------|---|---------|
| xops097ft.exe+99CB | 90 | nop | |
| xops097ft.exe+99CC | 90 | nop | |
| xops097ft.exe+99CD | 0FBE 96 1E904A00 | movsx edx,byte ptr [esi+xops097ft.exe+A901E] | |
| xops097ft.exe+99D4 | 0FBE 84 32 22904A00 | movsx eax,byte ptr [edx+esi+xops097ft.exe+A901E] | |
| xops097ft.exe+99DC | 8D 0C C5 00000000 | lea ecx,[eax+8+00000000] | |
| xops097ft.exe+99E3 | 2B C8 | sub ecx,eax | |
| xops097ft.exe+99E5 | 8D 14 88 | lea edx,[eax+ecx+4] | |
| xops097ft.exe+99E8 | 8A 04 55 30E34500 | mov al,[edx*2+xops097ft.exe+5E330] | |
| xops097ft.exe+99EF | 88 86 1F904A00 | mov [esi+xops097ft.exe+A901F],al | |
| xops097ft.exe+99F5 | 0FBE 8E 1E904A00 | movsx ecx,byte ptr [esi+xops097ft.exe+A901E] | |
| xops097ft.exe+99FC | 0FBE 84 31 22904A00 | movsx eax,byte ptr [ecx+esi+xops097ft.exe+A901E] | |
| xops097ft.exe+9A04 | 8D 14 C5 00000000 | lea edx,[eax+8+00000000] | |
| xops097ft.exe+9A0B | 2B D0 | sub edx,eax | |
| xops097ft.exe+9A0D | 8D 04 90 | lea eax,[eax+edx+4] | |
| xops097ft.exe+9A10 | D1 E0 | shl eax,1 | |

no operation

Protect:Read Only AllocationBase=00400000 Base=0045E000 Size=8000 Module=xops097f

| address | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0045E000 | 60 | E3 | 10 | 76 | 30 | E4 | 10 | 76 | F0 | FA | 10 | 76 | 00 | 00 | 00 | 00 | 70 | 5F | CA | 75 | B0 | 6D | CA | 75 | |
| 0045E018 | A0 | 58 | CA | 75 | E0 | 4C | CA | 75 | B0 | 68 | CA | 75 | 20 | 6F | CA | 75 | C0 | 6C | CA | 75 | F0 | 6F | CA | 75 | |
| 0045E030 | A0 | 5E | CA | 75 | B0 | 6E | CA | 75 | 00 | 00 | 00 | 00 | A0 | 31 | 5B | 77 | 00 | 0F | 5B | 77 | A0 | 41 | 5C | 77 | |
| 0045E048 | E0 | 31 | 5B | 77 | 10 | 33 | 5B | 77 | 50 | 32 | 5B | 77 | E0 | 0A | 5B | 77 | D0 | 0B | 5B | 77 | C0 | 1C | 5B | 77 | |
| 0045E060 | D0 | 3F | 5C | 77 | 60 | 0B | 5B | 77 | 70 | 0F | 5B | 77 | 80 | 0C | 5B | 77 | F0 | 00 | 5B | 77 | 00 | 01 | 5B | 77 | |

2-9. 코드 패치 결과

<총알 무제한>



2-9. 코드 패치 결과

<목숨 무제한>





3. 게임 핵 프로그램 개발

3-1. ReadProcessMemory Win32API

ReadProcessMemory 함수 (memoryapi.h)

2021년 10월 13일 • 읽는 데 2분

[이 페이지가 도움이 되었나요?](#)

통사론

C++

복사

```
BOOL ReadProcessMemory(  
    [in] HANDLE hProcess,  
    [in] LPCVOID lpBaseAddress,  
    [out] LPVOID lpBuffer,  
    [in] SIZE_T nSize,  
    [out] SIZE_T *lpNumberOfBytesRead  
);
```

매개변수

[in] hProcess

읽고 있는 메모리가 있는 프로세스에 대한 핸들입니다. 핸들에는 프로세스에 대한 PROCESS_VM_READ 액세스 권한이 있어야 합니다.

[in] lpBaseAddress

읽을 지정된 프로세스의 기본 주소에 대한 포인터입니다. 데이터 전송이 발생하기 전에 시스템은 지정된 크기의 기본 주소 및 메모리에 있는 모든 데이터가 읽기 액세스를 위해 액세스 가능한지 확인하고 액세스할 수 없으면 기능이 실패합니다.

[out] lpBuffer

지정된 프로세스의 주소 공간에서 내용을 수신하는 버퍼에 대한 포인터입니다.

[in] nSize

지정된 프로세스에서 읽을 바이트 수입니다.

[out] lpNumberOfBytesRead

지정된 버퍼로 전송된 바이트 수를 받는 변수에 대한 포인터입니다. 경우 *lpNumberOfBytesRead* 가 있다 NULL 매개 변수는 무시됩니다.

특정 프로세스의 메모리를 읽는 API
(주로 메모리 값을 검증하는데 쓰임)

3-2. WriteProcessMemory Win32API

WriteProcessMemory 함수 (memoryapi.h)

2021년 10월 13일 • 읽는 데 2분

[이 페이지가 도움이 되었나요?](#)

지정된 프로세스의 메모리 영역에 데이터를 씁니다. 기록할 전체 영역에 액세스할 수 있어야 하며 그렇지 않으면 작업이 실패합니다.

통사론

C++

 복사

```
BOOL WriteProcessMemory(  
    [in] HANDLE hProcess,  
    [in] LPVOID lpBaseAddress,  
    [in] LPCVOID lpBuffer,  
    [in] SIZE_T nSize,  
    [out] SIZE_T *lpNumberOfBytesWritten  
);
```

매개변수

[in] hProcess

수정할 프로세스 메모리에 대한 핸들입니다. 핸들에는 프로세스에 대한 PROCESS_VM_WRITE 및 PROCESS_VM_OPERATION 액세스 권한이 있어야 합니다.

[in] lpBaseAddress

데이터가 기록되는 지정된 프로세스의 기본 주소에 대한 포인터입니다. 데이터 전송이 발생하기 전에 시스템은 지정된 크기의 기본 주소와 메모리에 있는 모든 데이터에 쓰기 액세스에 액세스할 수 있는지 확인하고 액세스할 수 없으면 기능이 실패합니다.

[in] lpBuffer

지정된 프로세스의 주소 공간에 쓸 데이터가 들어 있는 버퍼에 대한 포인터입니다.

[in] nSize

지정된 프로세스에 쓸 바이트 수입니다.

[out] lpNumberOfBytesWritten

지정된 프로세스로 전송된 바이트 수를 수신하는 변수에 대한 포인터입니다. 이 매개변수는 선택 사항입니다. 경우 *lpNumberOfBytesWritten*가 있다 **NULL** 매개 변수는 무시됩니다.

특정 프로세스의 메모리를 쓰는 API
(주로 메모리 변조에 쓰임)

3-3. OpenProcess Win32API

OpenProcess function (processthreadsapi.h)

10/13/2021 • 2 minutes to read

[Is this page helpful?](#)

Opens an existing local process object.

Syntax

```
C++  
  
HANDLE OpenProcess(  
    [in] DWORD dwDesiredAccess,  
    [in] BOOL bInheritHandle,  
    [in] DWORD dwProcessId  
);
```

Copy

매개변수

[in] dwDesiredAccess

프로세스 개체에 대한 액세스입니다. 이 액세스 권한은 프로세스의 보안 설명자에 대해 확인됩니다. 이 매개변수는 하나 이상의 [프로세스 액세스 권한](#) 일 수 있습니다.

호출자가 SeDebugPrivilege 권한을 활성화한 경우 보안 설명자의 내용에 관계없이 요청된 액세스가 허용됩니다.

[in] bInheritHandle

이 값이 TRUE이면 이 프로세스에서 만든 프로세스가 핸들을 상속합니다. 그렇지 않으면 프로세스가 이 핸들을 상속하지 않습니다.

[in] dwProcessId

열려는 로컬 프로세스의 식별자입니다.

PID로 해당 프로세스 핸들에 대해 권한을 부여할 수 있음.
(프로그램의 PID를 알아내야 함..)

3-4. GetWindowThreadProcessId Win32API

GetWindowThreadProcessId 함수 (winuser.h)

2021년 10월 13일 • 읽는 데 2분

[이 페이지가 도움이 되었나요?](#)

지정된 창을 만든 스레드의 식별자와 선택적으로 창을 만든 프로세스의 식별자를 검색합니다.

통사론

C++

복사

```
DWORD GetWindowThreadProcessId(  
    [in]          HWND    hwnd,  
    [out, optional] LPDWORD lpdwProcessId  
);
```

매개변수

[in] hwnd

유형: HWND

창에 대한 핸들입니다.

[out, optional] lpdwProcessId

유형: LPDWORD

프로세스 식별자를 받는 변수에 대한 포인터입니다. 이 파라미터가 아닌 경우 **NULL**, **GetWindowThreadProcessId** 복사 변수는 프로세스의 식별자; 그렇지 않으면 그렇지 않습니다.

프로세스 핸들로 PID를 구할 수 있음.
(프로그램의 프로세스 핸들을 알아내야 함..)

3-5. FindWindow Win32API

FindWindowA function (winuser.h)

10/13/2021 • 2 minutes to read

[Is this page helpful?](#)

Retrieves a handle to the top-level window whose class name and window name match the specified strings. This function does not search child windows. This function does not perform a case-sensitive search.

To search child windows, beginning with a specified child window, use the [FindWindowEx](#) function.

Syntax

```
C++  
  
HWND FindWindowA(  
    [in, optional] LPCSTR lpClassName,  
    [in, optional] LPCSTR lpWindowName  
);
```

매개변수

[in, optional] lpClassName

유형: LPCTSTR

[RegisterClass](#) 또는 [RegisterClassEx](#) 함수에 대한 이전 호출에 의해 생성된 클래스 이름 또는 클래스 원자. 원자는 *lpClassName*의 하위 단어에 있어야 합니다. 상위 단어는 0이어야 합니다.

*lpClassName*이 문자열을 가리키는 경우 창 클래스 이름을 지정합니다. 클래스 이름은 [RegisterClass](#) 또는 [RegisterClassEx](#)에 등록된 이름이거나 사전 정의된 제어 클래스 이름일 수 있습니다.

경우 *lpClassName*이 NULL, 누구의 타이틀과 일치하는 모든 창을 발견 *lpWindowName*의 매개 변수를.

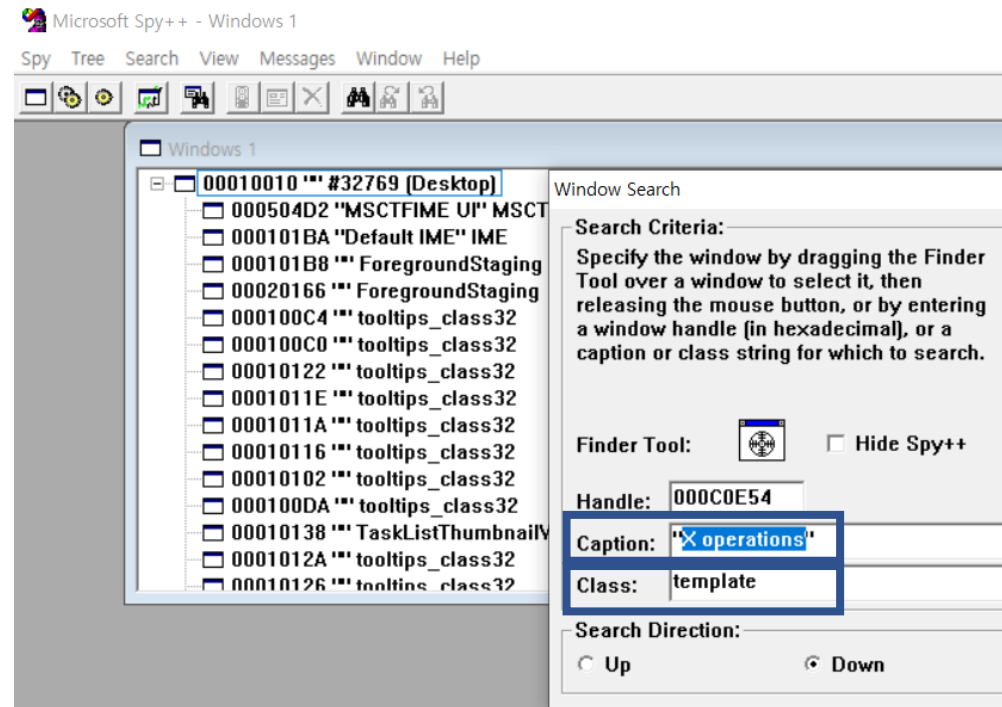
[in, optional] lpWindowName

유형: LPCTSTR

창 이름(창의 제목). 이 매개변수가 NULL이면 모든 창 이름이 일치합니다.

프로세스의 클래스 이름 또는 윈도우 이름(창 이름)으로
프로세스 핸들 값을 구할 수 있음.

3-6. Spy++ 사용



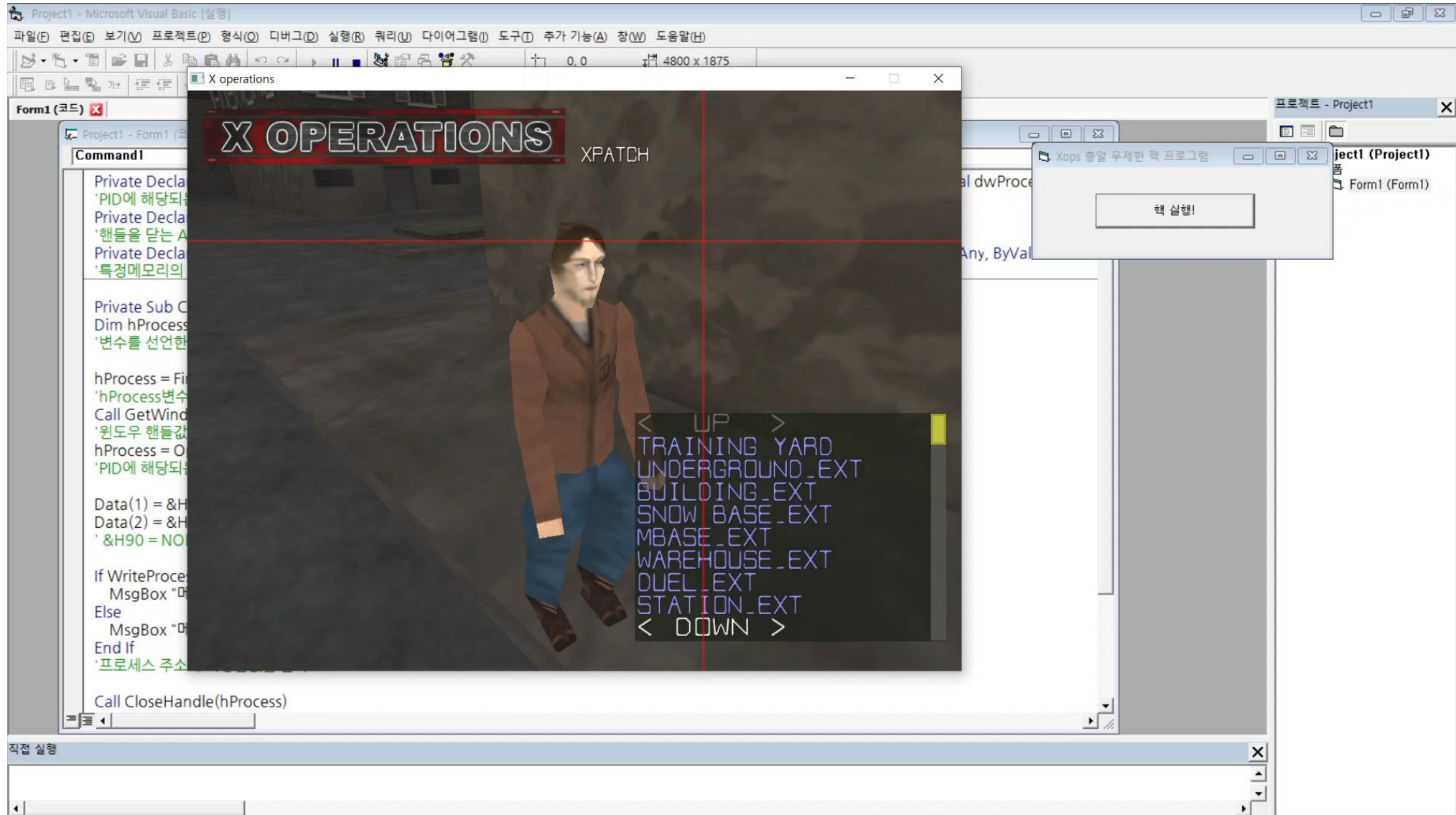
Spy++ 프로그램으로 프로세스 고유
클래스 이름과 윈도우 이름(창 이름)을 알아낼 수 있음.

3-7. 소스코드

```
Private Sub Command1_Click()  
Dim hProcess As Long, Data(2) As Byte  
'변수를 선언한다.  
  
hProcess = FindWindow(vbNullString, "X operations"  
'hProcess 변수에 특정 윈도우 캡션 이름을 사용하는 윈도우의 핸들값을 넣는다.  
Call GetWindowThreadProcessId(hProcess, hProcess)  
'윈도우 핸들값으로 PID를 구해서 hProcess에 넣는다.  
hProcess = OpenProcess(&H1F0FFF, False, hProcess)  
'PID에 해당되는 프로세스에 핸들을 요청하고 요청한 핸들의 값을 hProcess에 넣는다.  
  
Data(1) = &H90  
Data(2) = &H90  
' &H90 = NOP  
  
If WriteProcessMemory(hProcess, ByVal &H4099CB, Data(1), 2, 0) Then  
    MsgBox "메모리 변조 성공", 64, "성공"  
Else  
    MsgBox "메모리 변조 실패", 64, "실패"  
End If  
'프로세스 주소에 특정한값을 쓴다.  
  
Call CloseHandle(hProcess)  
'요청한 핸들을 닫는다.  
  
'제작자: 2N(nms200299)  
'제작날짜: 2015-01-31  
End Sub
```

제작 언어는 VisualBasic 6.0으로 6년 전에 제가 작성한 코드를 살짝 변형

3-8. 작동 영상





4. 메모리 변조 대처방안

4-1. 메모리 검증


ReadProcessMemory 함수 (memoryapi.h)

2021년 10월 13일 • 읽는 데 2분

[이 페이지가 도움이 되었나요?](#)

통사론

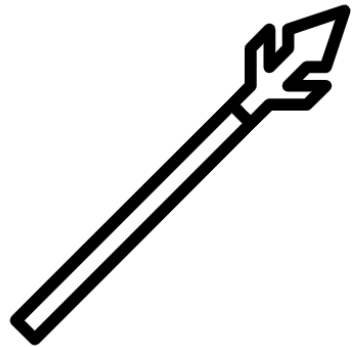
C++

 복사

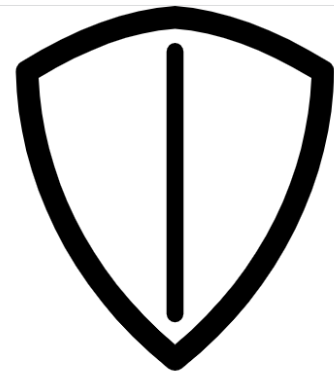
```
BOOL ReadProcessMemory(  
    [in] HANDLE hProcess,  
    [in] LPCVOID lpBaseAddress,  
    [out] LPVOID lpBuffer,  
    [in] SIZE_T nSize,  
    [out] SIZE_T *lpNumberOfBytesRead  
);
```

ReadProcessMemory API를 통해서 메모리 변조되지 않았는지
직접적인 값의 비교나 CRC 검증을 통해 알 수 있음.

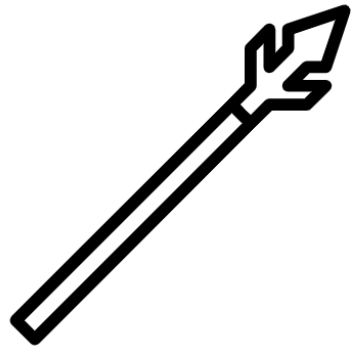
CRC 검증 루틴을 WriteProcessMemory로
코드 패치해버리면?



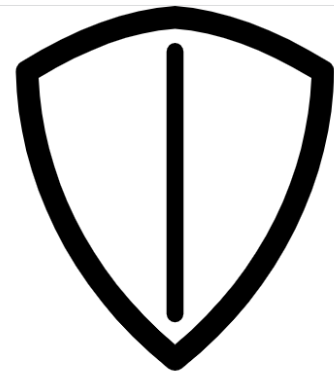
VS



따라서 게임핵은 반드시 존재할 수 밖에 없다.



VS





Q & A





감사합니다

