지뢰 찾기 리버싱

17TH 홍영우

RUNA

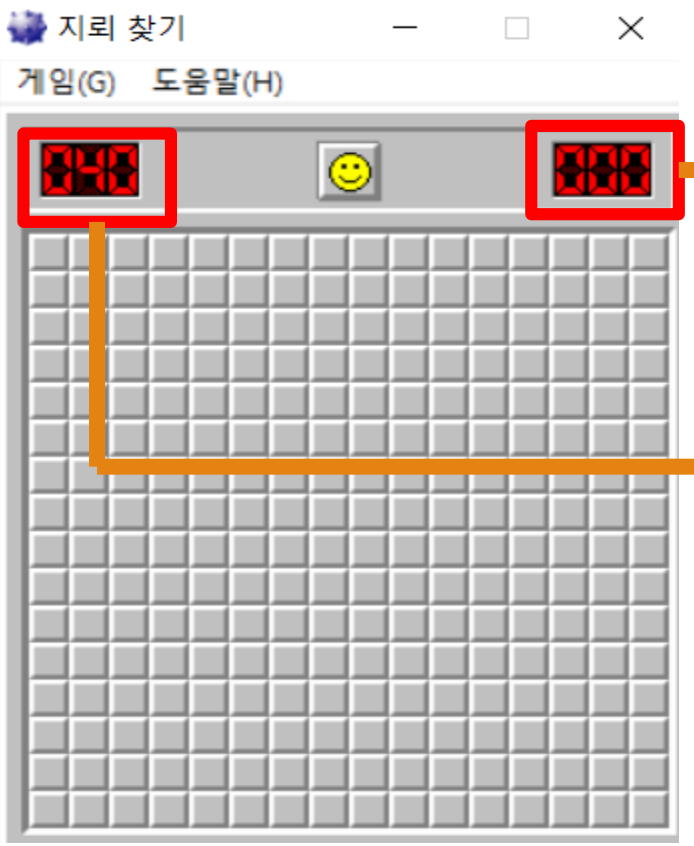
목차

툴

cheat engine 

치트 엔진은 메모리를 참조하여 변수 값 들을 검색, 수정을 하기 위해 사용됨

변수 값들이 만약 평문으로 저장 되어 있다면?

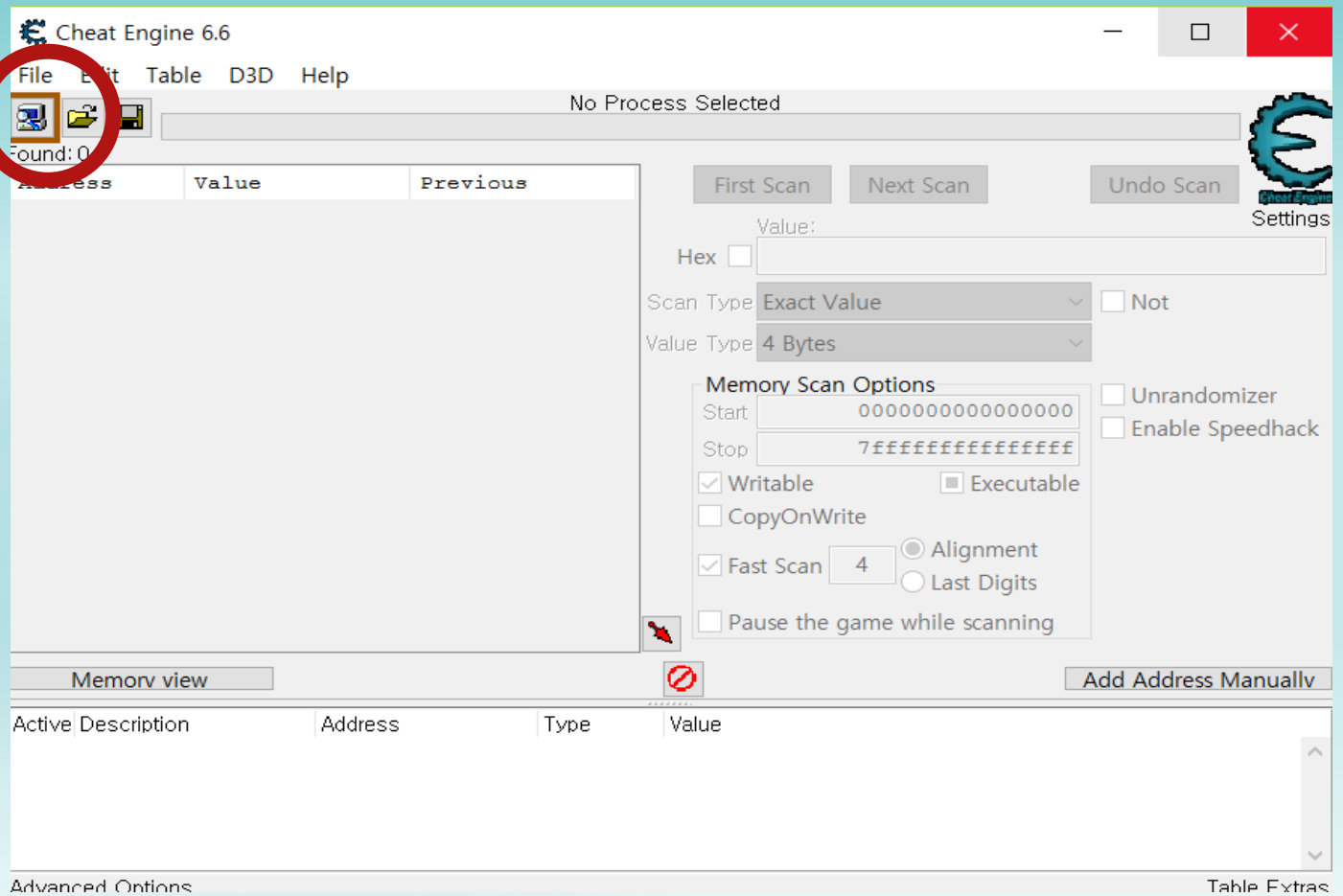


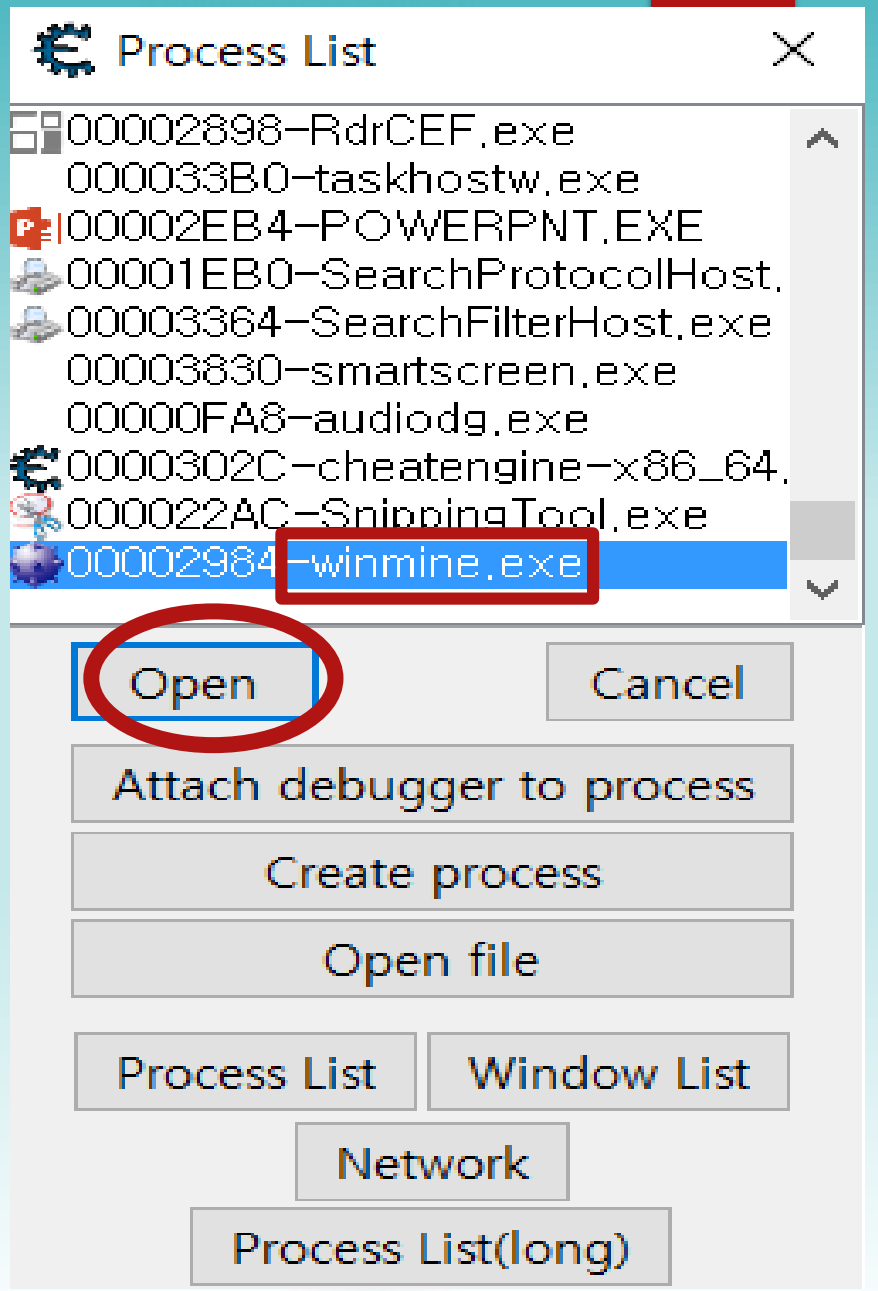
시간 값

지뢰 개수 값

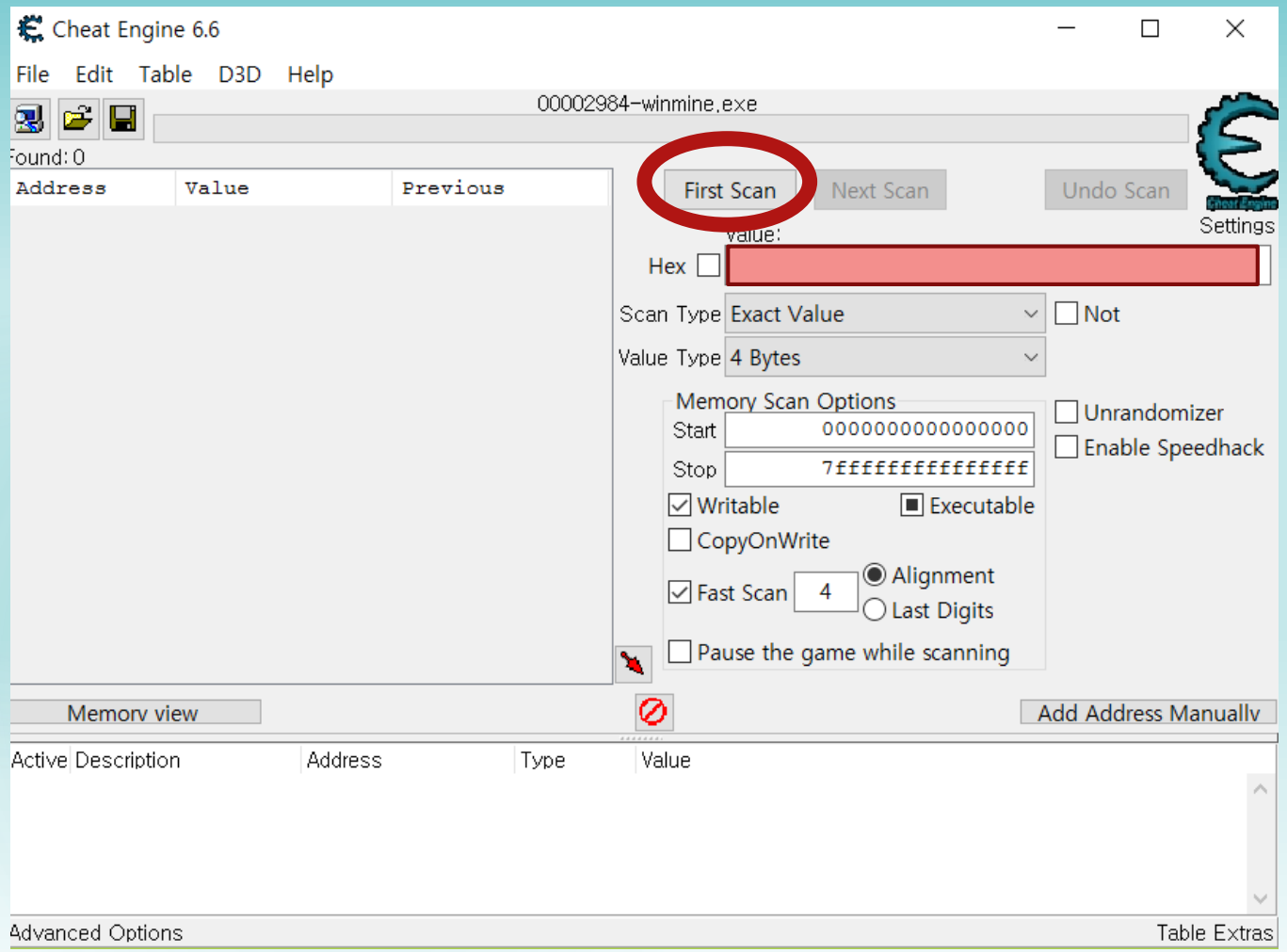
변수 값들을 검색 해내어
수정 할 수도 있을 것이다.

Cheat Engine 6.6 실행

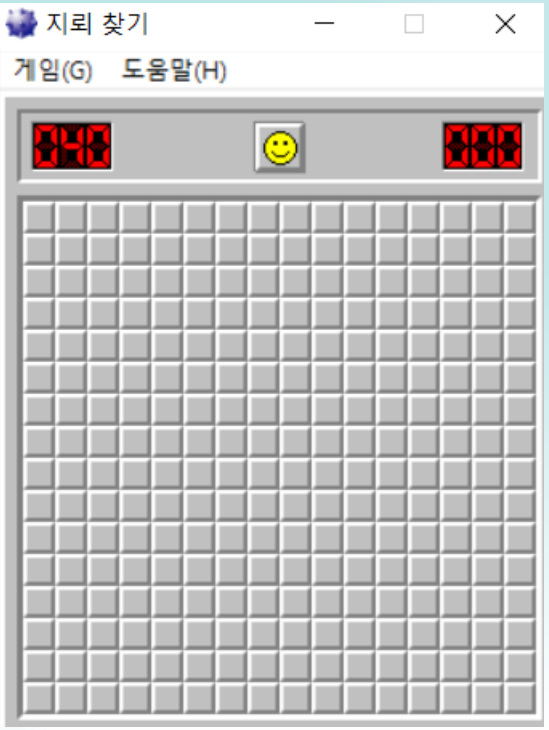




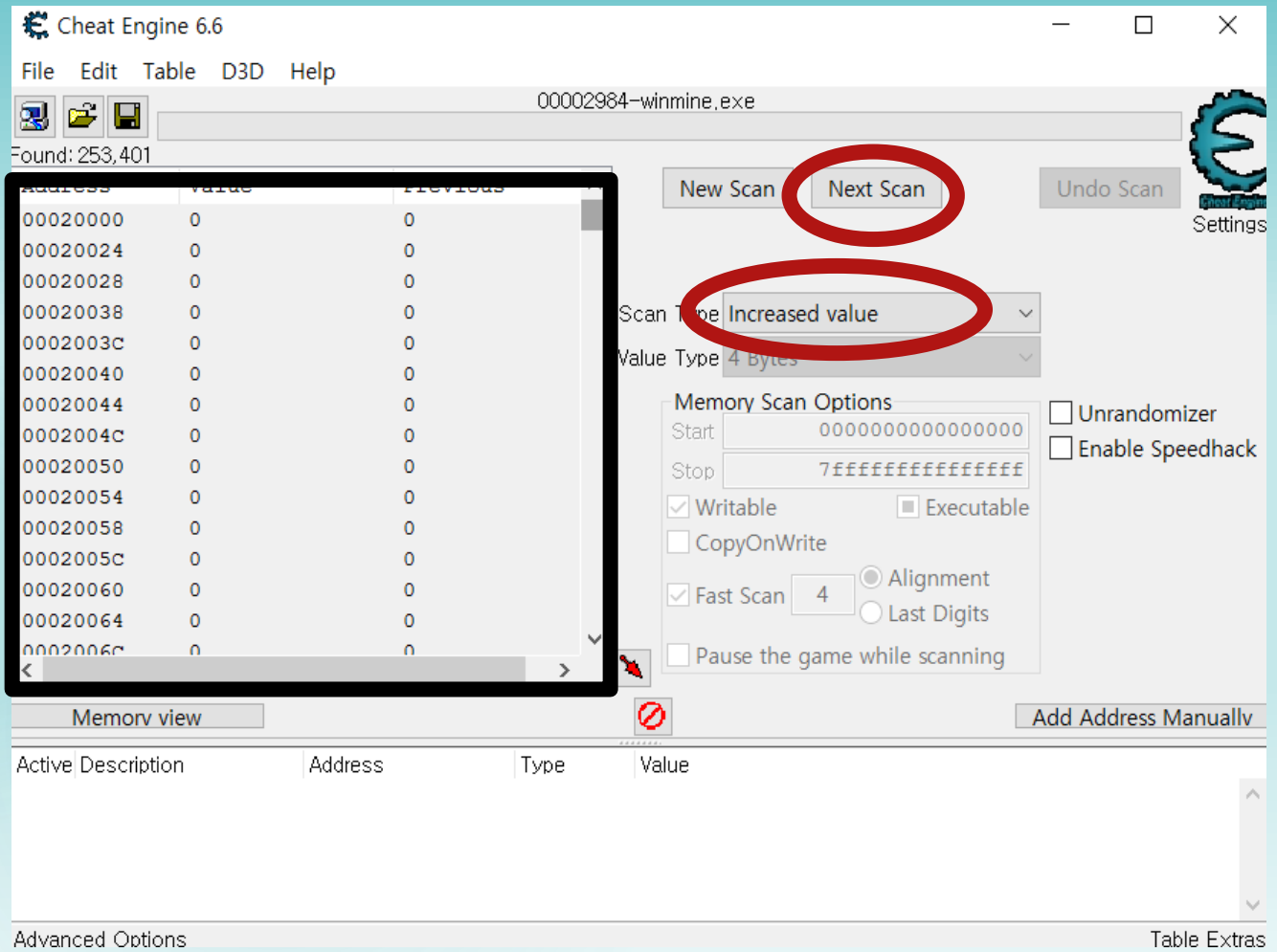
시간 값 검색하기



초기 시간 값 검색:
value에 0 넣고
First scan 클릭!



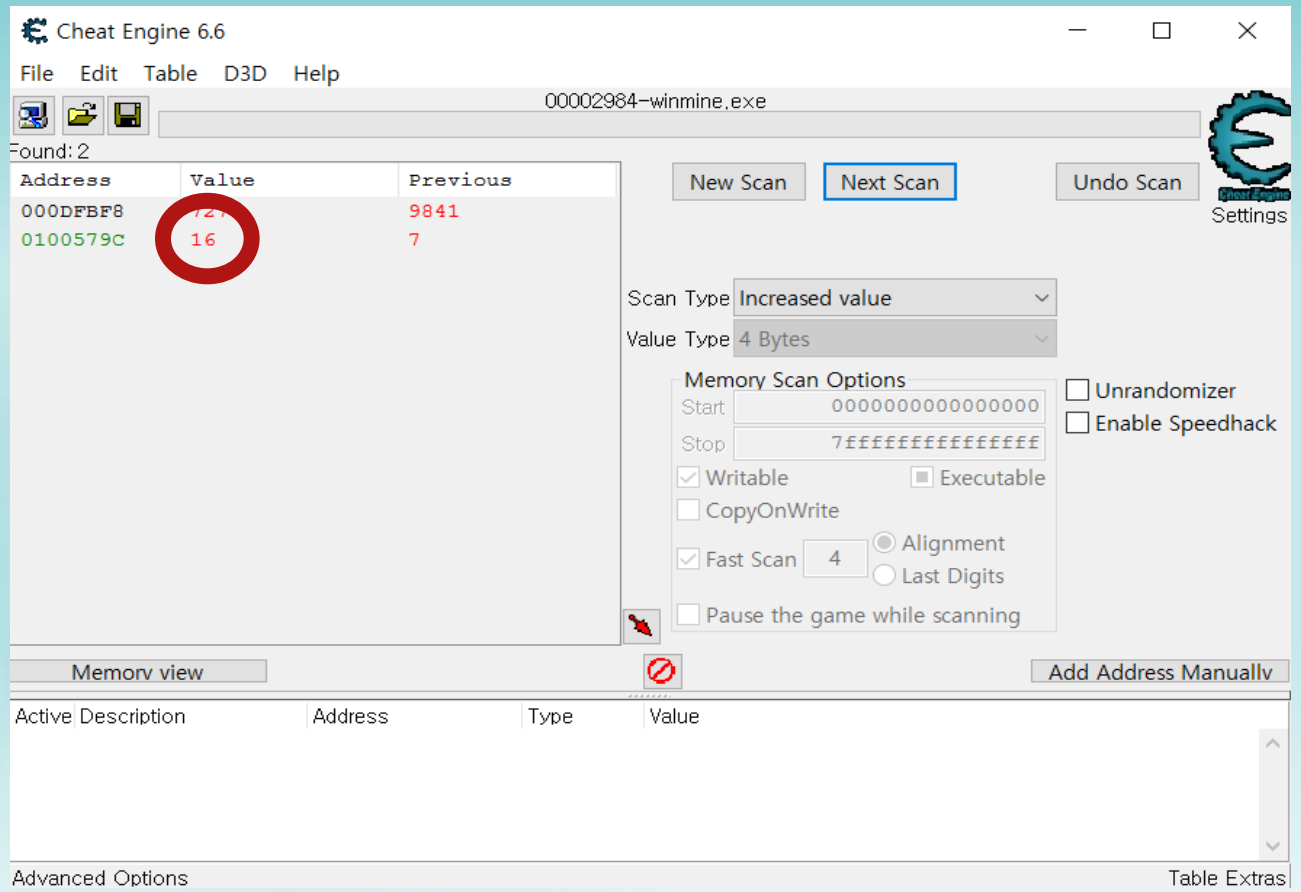
시간 값 검색하기



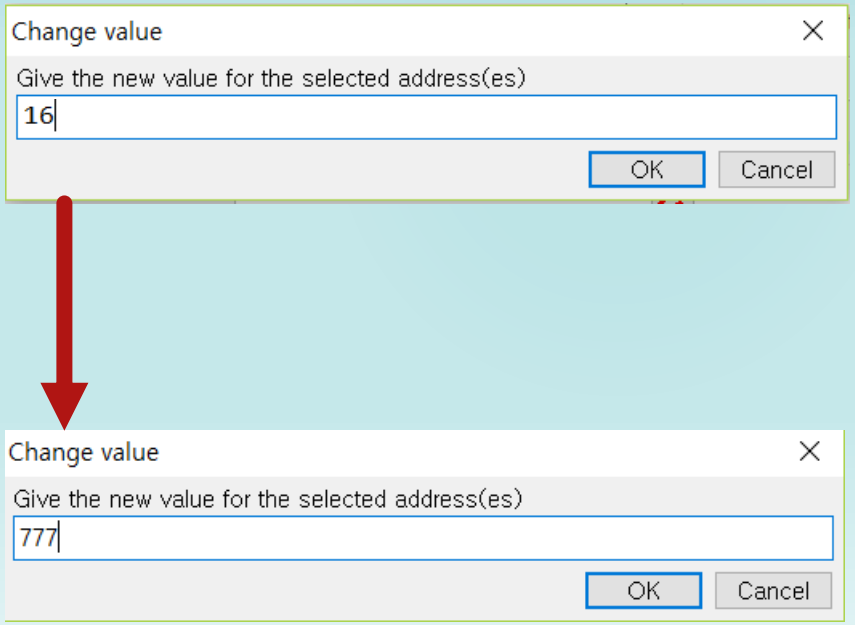
증가된 값들을 검색한다.

Next scan 을 여러 번 누르면 시간 값과 일치하는 데이터가 나온다.

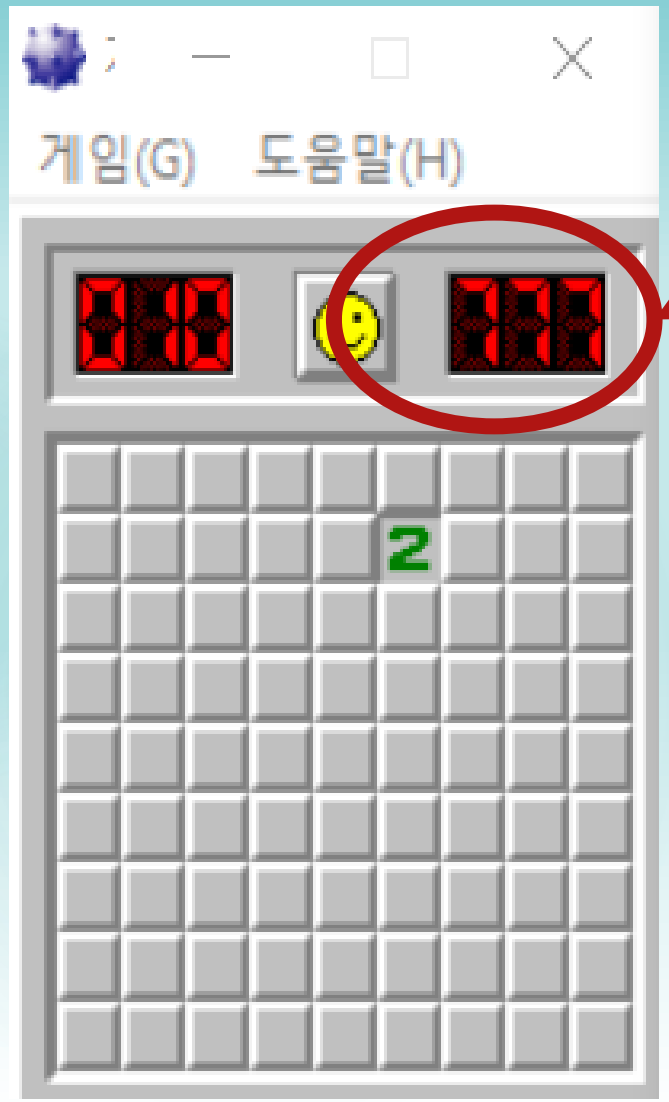
시간 값 조작하기



우 클릭 -> Change of selected address



수정된 시간 값



Cheat Engine 6.6

File Edit Table D3D Help

00003A04-winmine.exe

Found: 2

Address	Value	Previous
000DFC58	[illegible]	4293664835
0100579C	777	6

New Scan | Next Scan | Undo Scan | Settings

Scan Type: Increased value

Value Type: 4 Bytes

Memory Scan Options

Start: 0000000000000000

Stop: 7fffffffffffffff

Writable | Executable | CopyOnWrite

Fast Scan 4 | Alignment | Last Digits

Pause the game while scanning

Unrandomizer

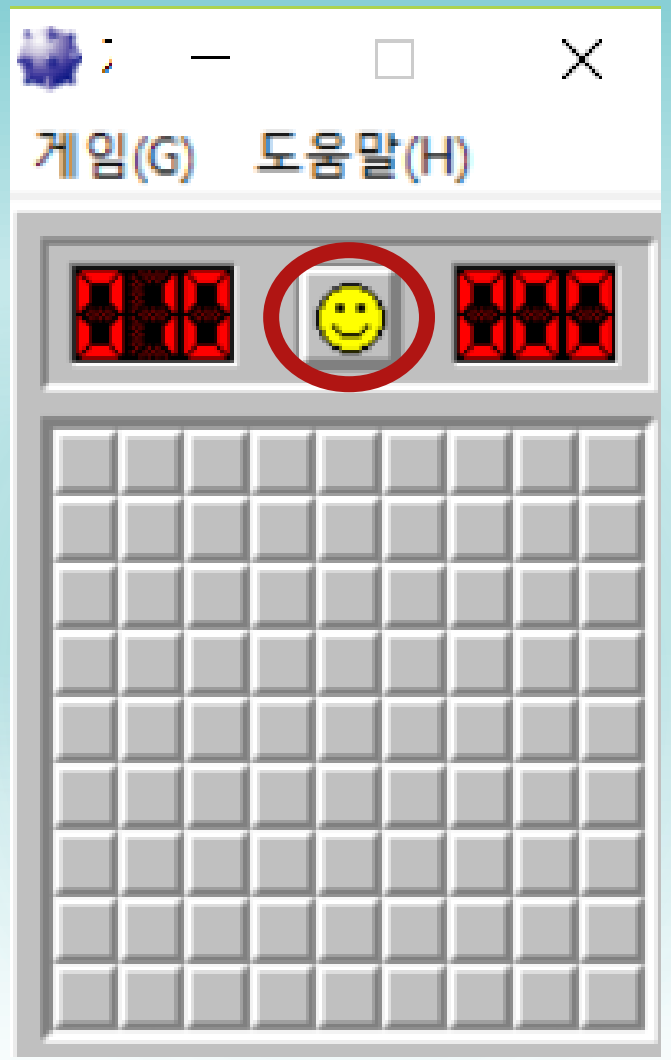
Enable Speedhack

Memory view | Add Address Manually

Active	Description	Address	Type	Value

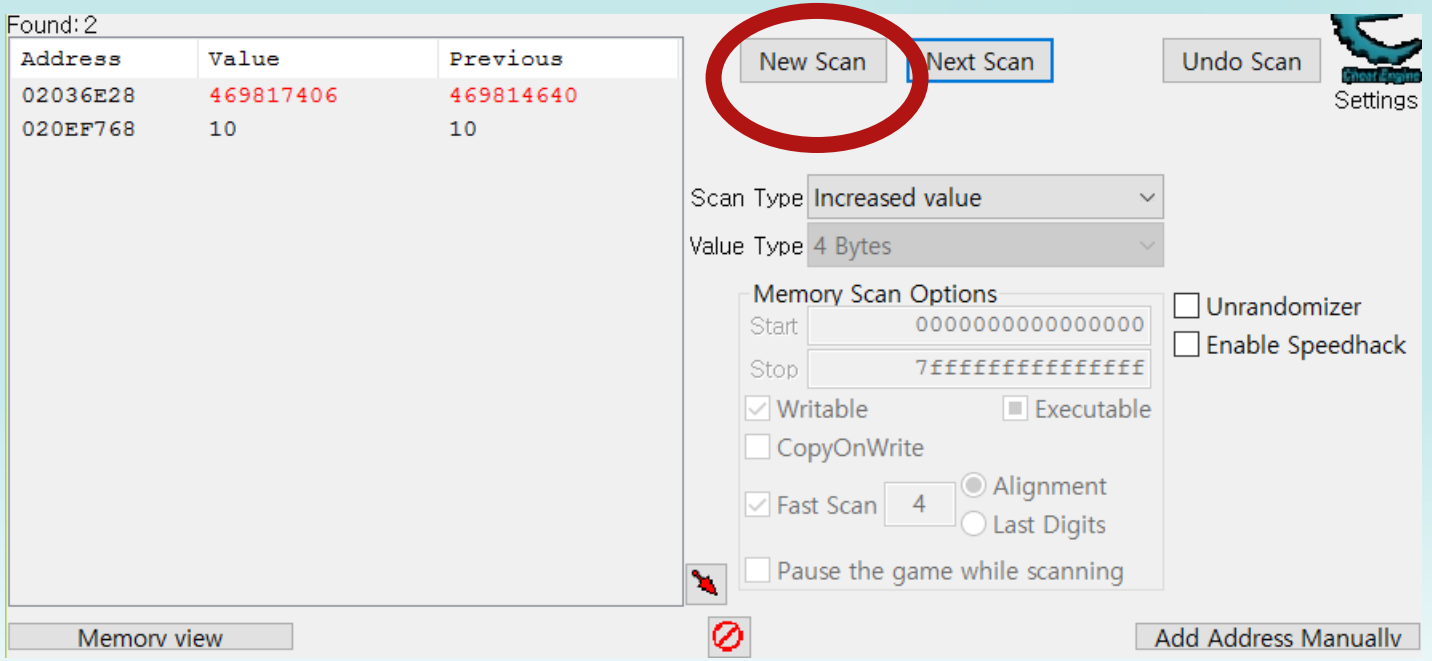
Advanced Options | Table Extras

지뢰 개수 조작하기

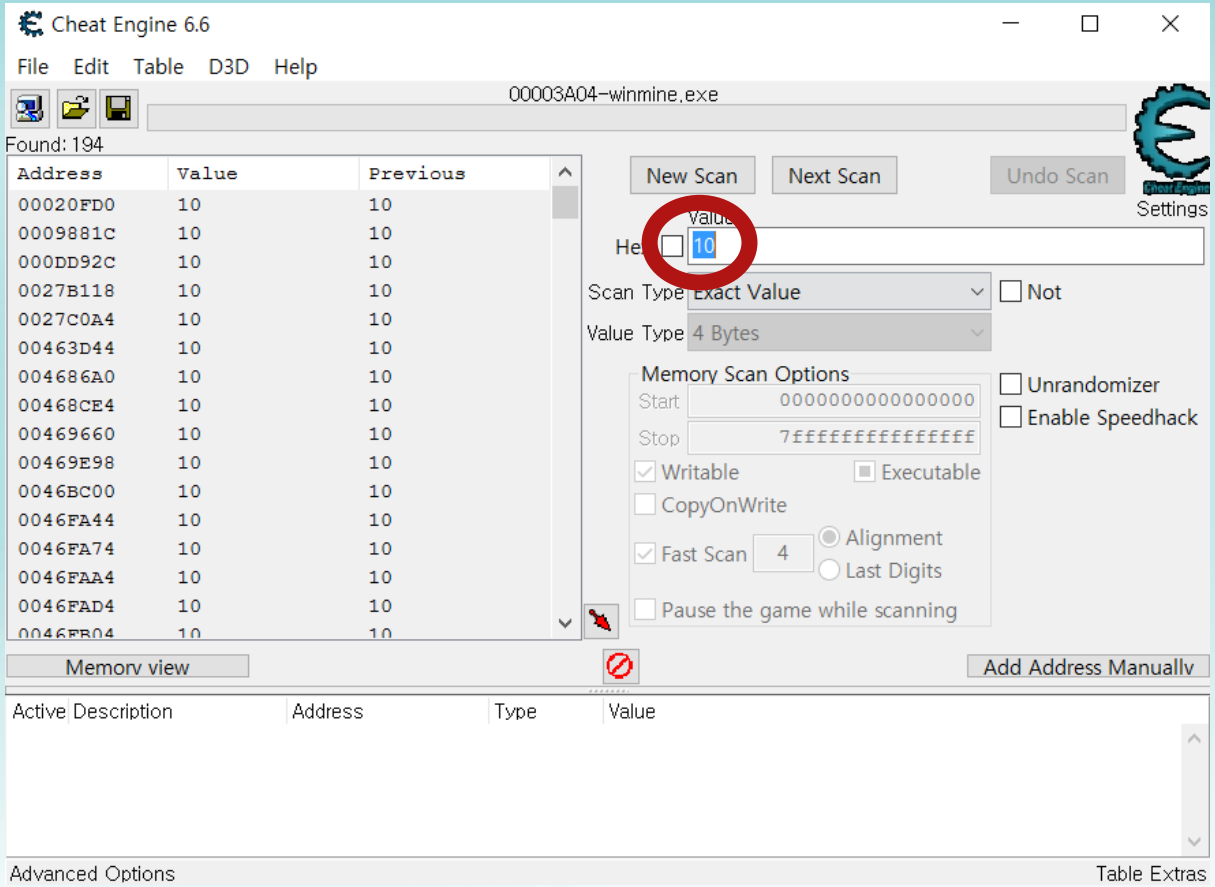


초급으로 새 게임 실행

New scan을 눌러 검색 조건 초기화

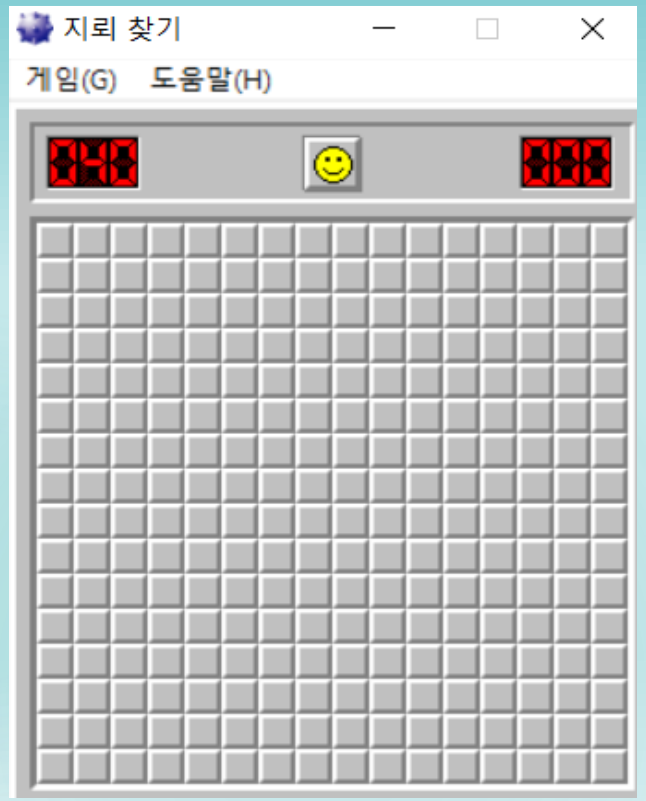


지뢰 값 찾기



Value에 지뢰 개수인
10을 넣고 Next scan한다.

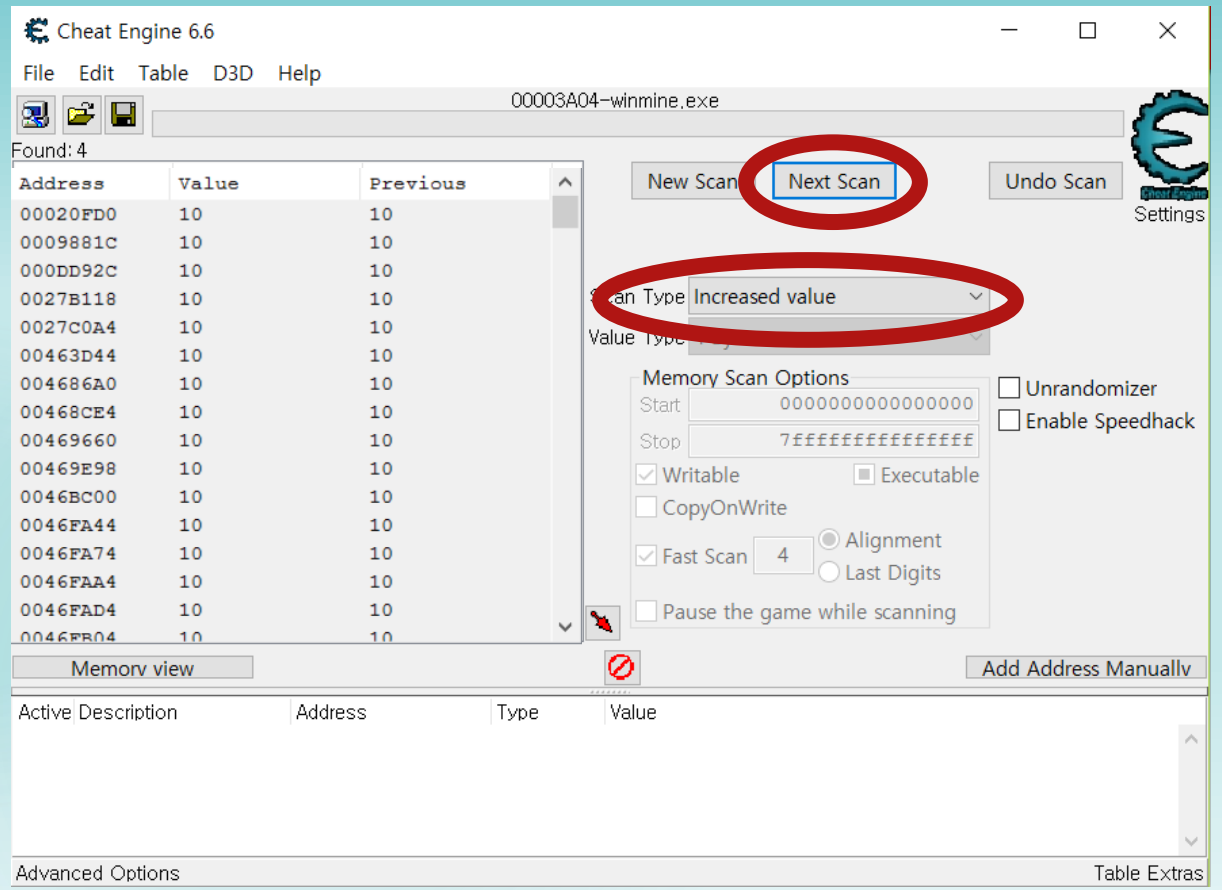
지뢰 값 찾기



좌측 상단에서 난이도를 중급으로 올린다.

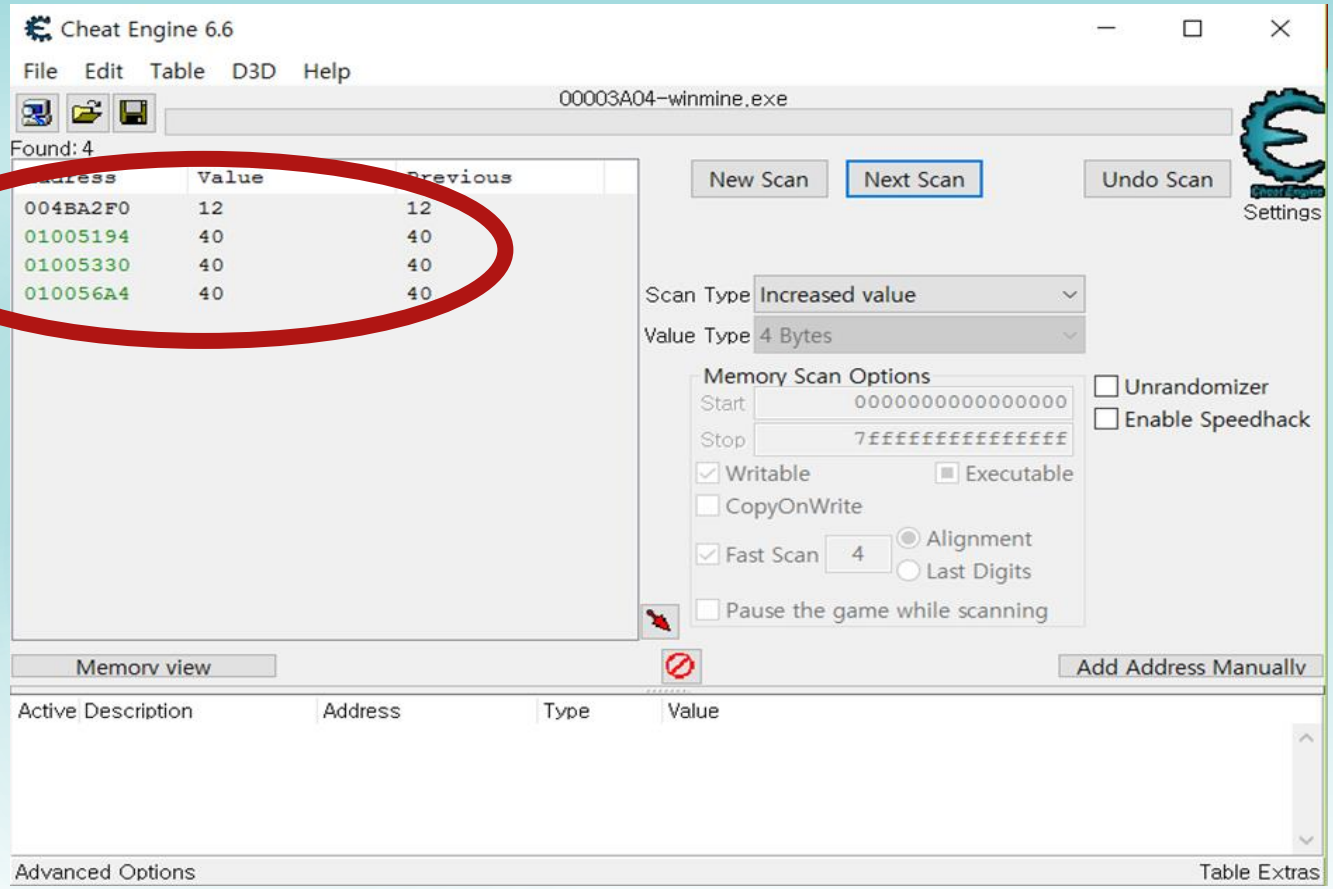
초급 -> 중급: 지뢰 개수 증가
10 -> 40

지뢰 값 찾기



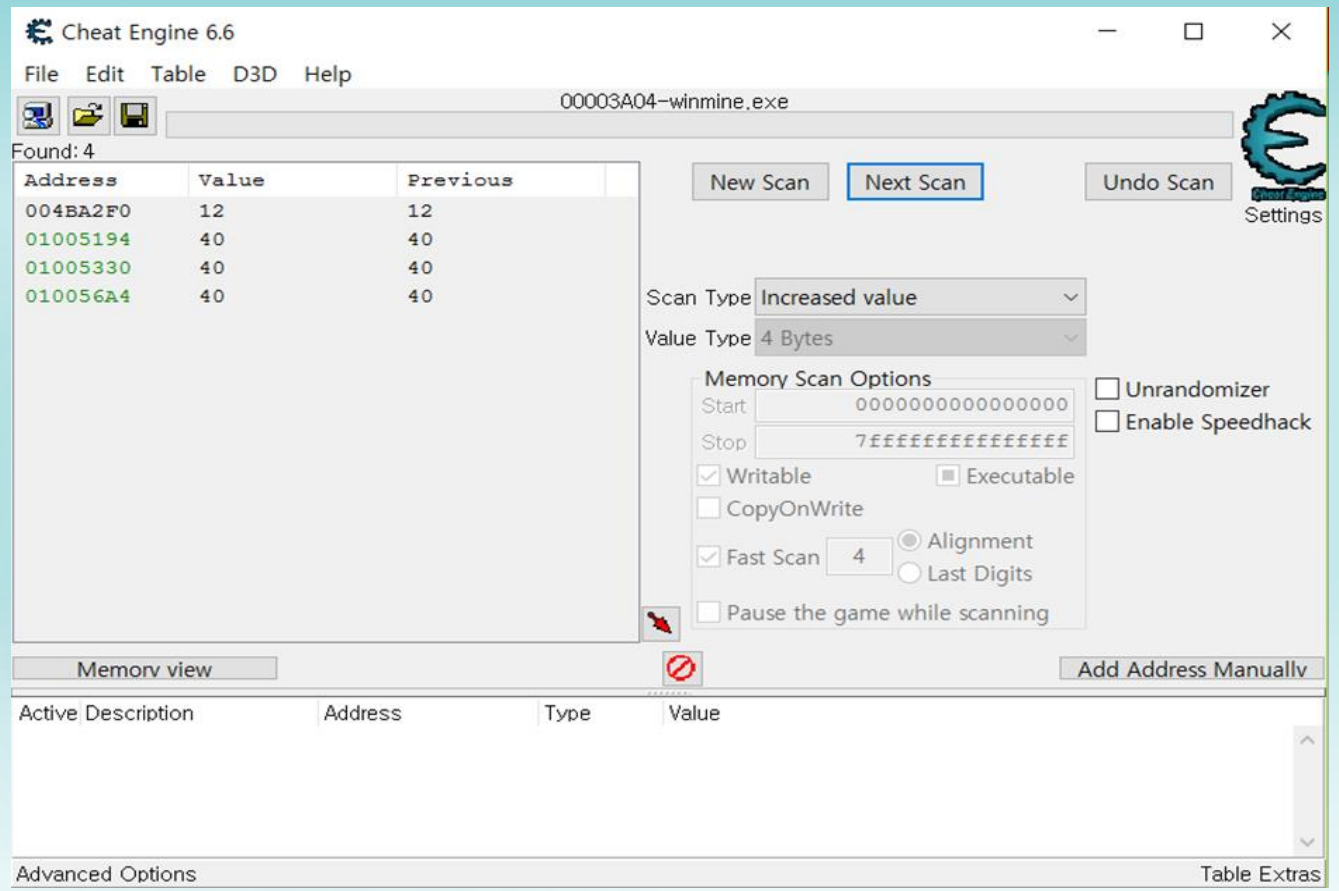
Scan type 을 increased value로 바꾸고 next scan !

지뢰 값 찾기

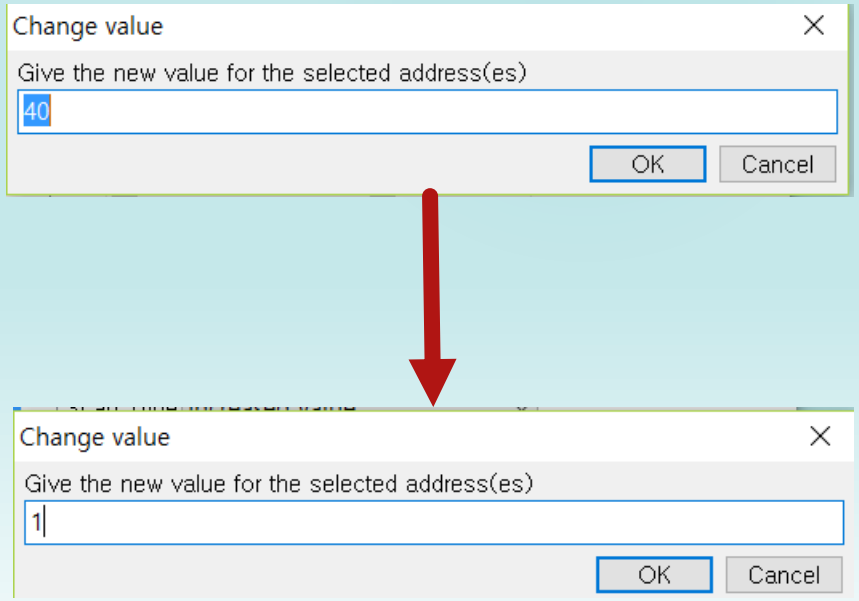


드디어 찾은 지뢰 값!!!

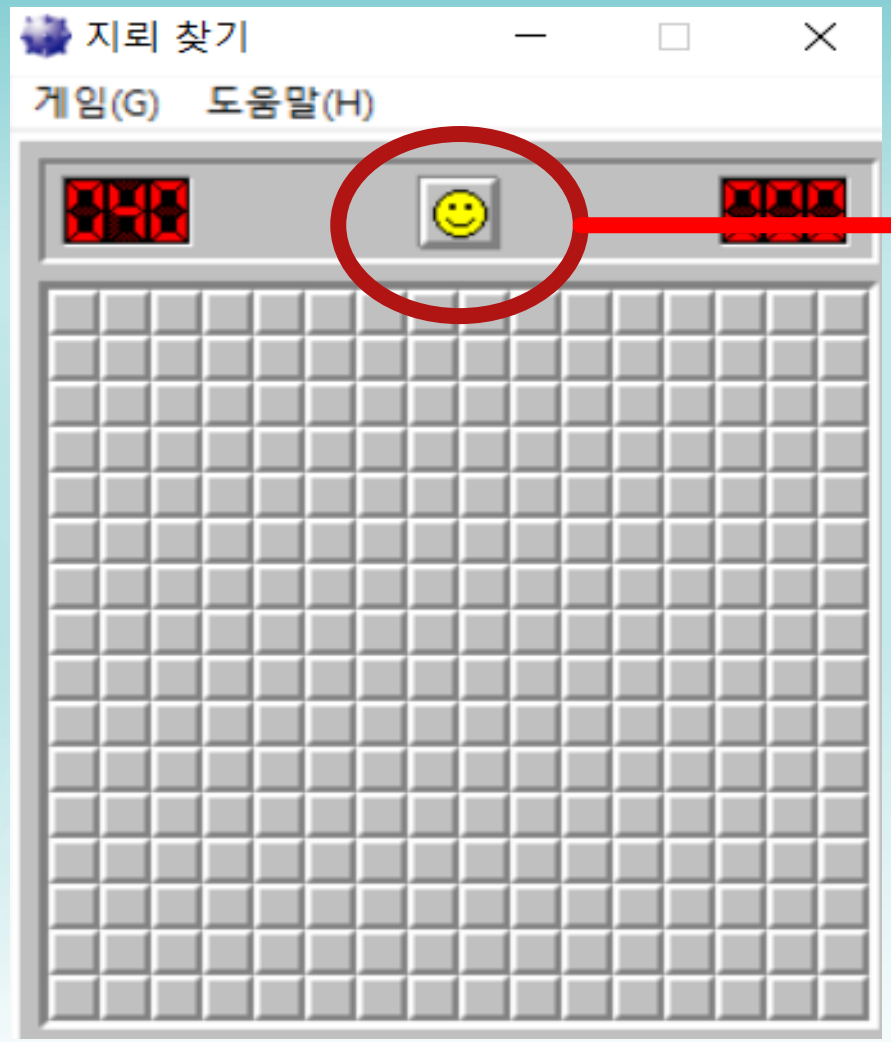
지뢰 개수 값 조작



우 클릭 -> Change of selected address -> 1로 수정

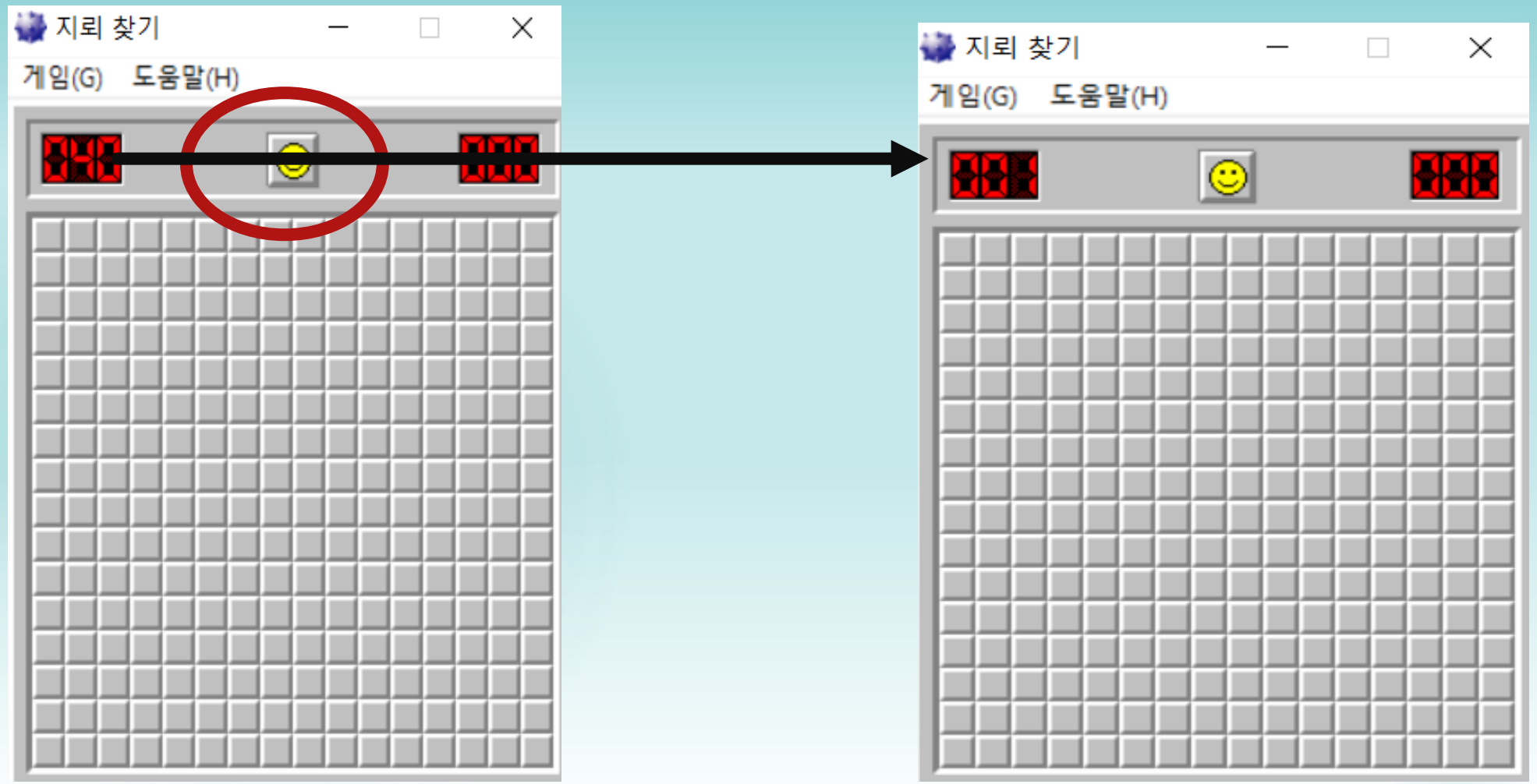


새 게임 시작

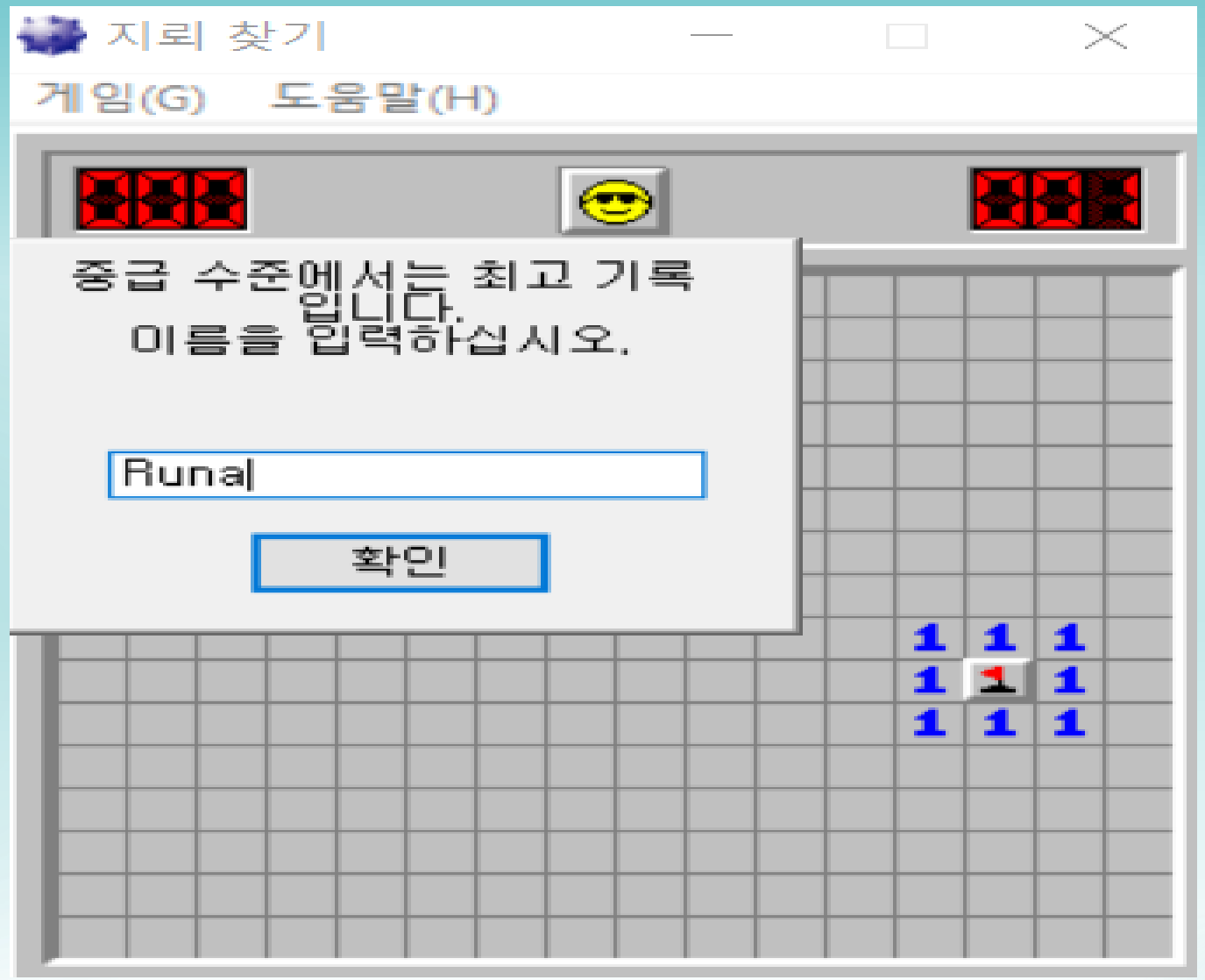


값 수정 후 새 게임 실행

수정된 지뢰 개수 값



수정 후 결과



결과

- 지뢰 찾기에서 값들이 평문으로 저장 되어 있어 쉽게 검색과 수정이 가능 했다

ida와 olly로 좀 더 분석 해보자

- ida와 olly dbg를 통해 일부 함수들을 분석해 취약한 부분 찾기

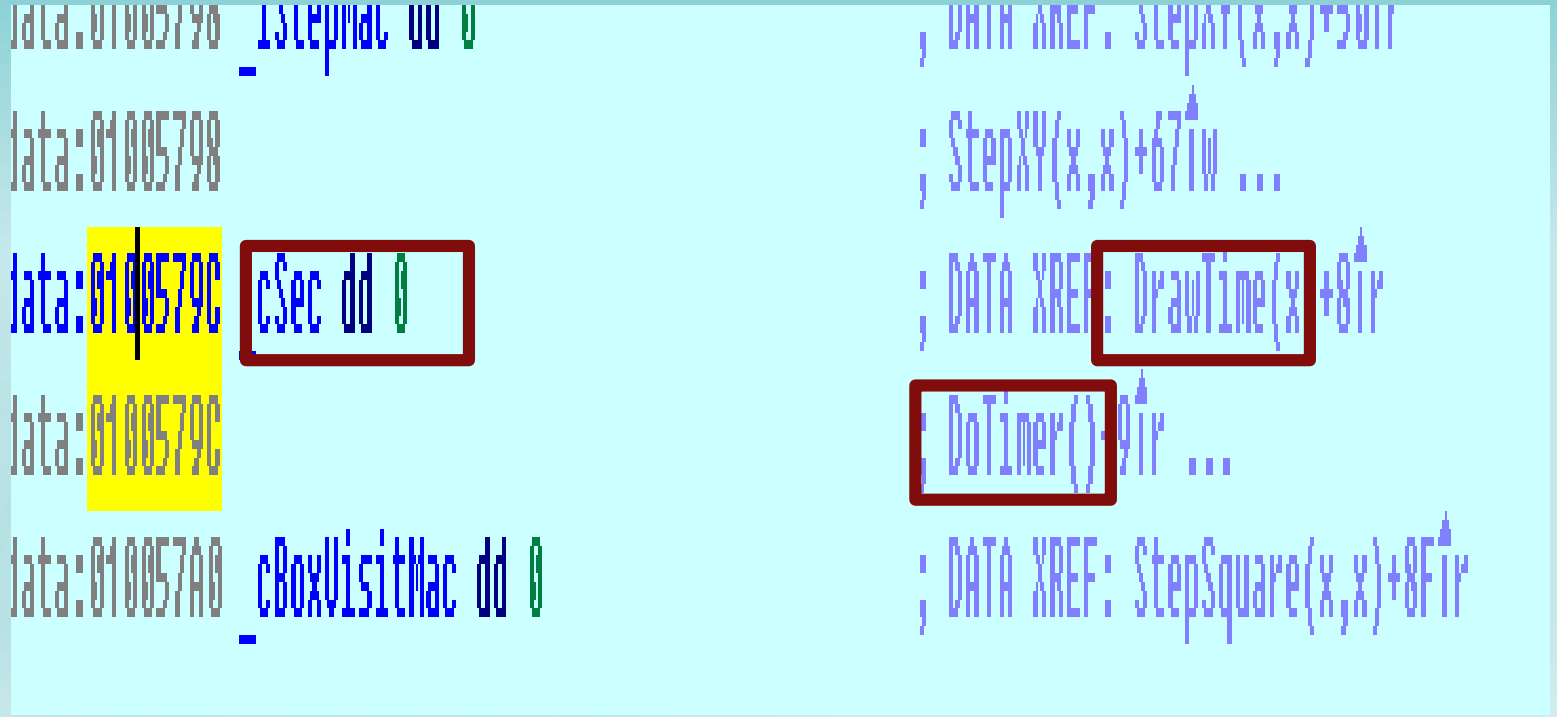
툴

IDA PRO

olly dbg

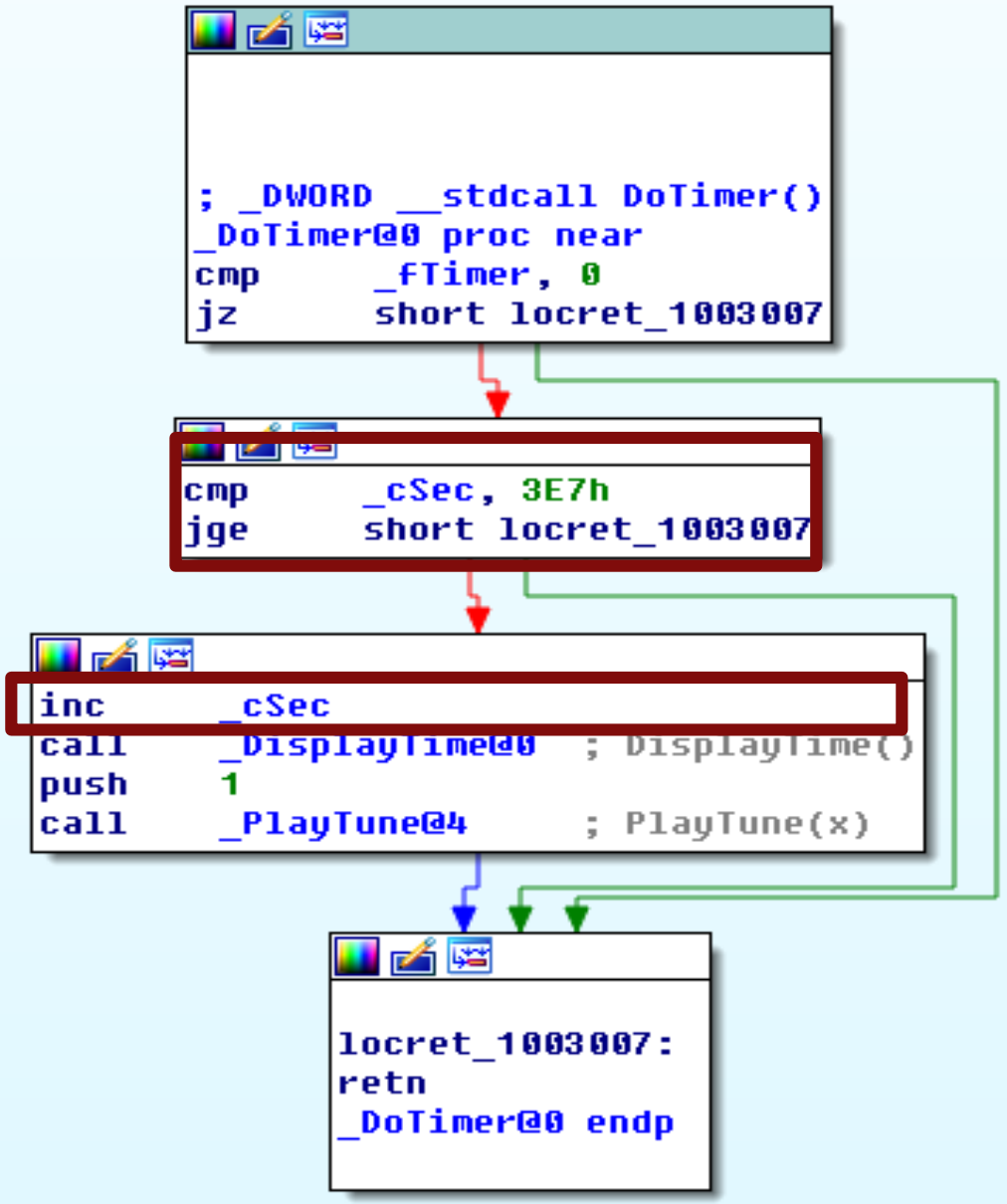
- ida는 정적 분석에 olly는 동적 분석에 사용함
- 나누어 사용한 이유는 각 분석에 있어 더 편해서

시간(data)값을 참조 하는 함수 분석



dd -> double 형태로 시간 값을 저장

DrawTime 함수와 DoTimer함수가 참조 중



Do Timer 함수

_cSec -> 카운터 하고 있는 시간값을 저장하는 변수

cmp -> 두 값을 비교 <분기점>

초록 ->true 빨강->false
inc _cSec

inc → ++ (c언어) : 값을 1증가

3E7h (999)초가 될 때 까지 시간 값을 증가하는 구문 반복

지뢰 값(data)을 참조하는 함수들을 분석

```
.data:01005190                                  ; HtmlHelpA(x,x,x,x)+71iw
.data:01005194 _cBombLeft      dd 0             ; DATA XREF: DrawBombCount(x):loc_10027AAir
.data:01005194                                  ; UpdateBombCount(x)+4iw ...
.data:01005198                 align 10h
.data:010051A0 _rgStepX        dd 0             ; DATA XREF: StepXY(x,x)+55iw
.data:010051A0                                  ; StepBox(x,x)+29ir
.data:010056A4 ; UINT dword_10056A4
.data:010056A4 dword_10056A4   dd ?             ; DATA XREF: PrefDlgProc(x,x,x,x)+93iw
.data:010056A4                                  ; PrefDlgProc(x,x,x,x)+D2ir ...
.data:01005330 _cBombStart     dd ?             ; DATA XREF: StartGame()+48iw
.data:01005330                                  ; StartGame()+83iw ...
```

Count Bombs (동적 분석)

olly 사용함

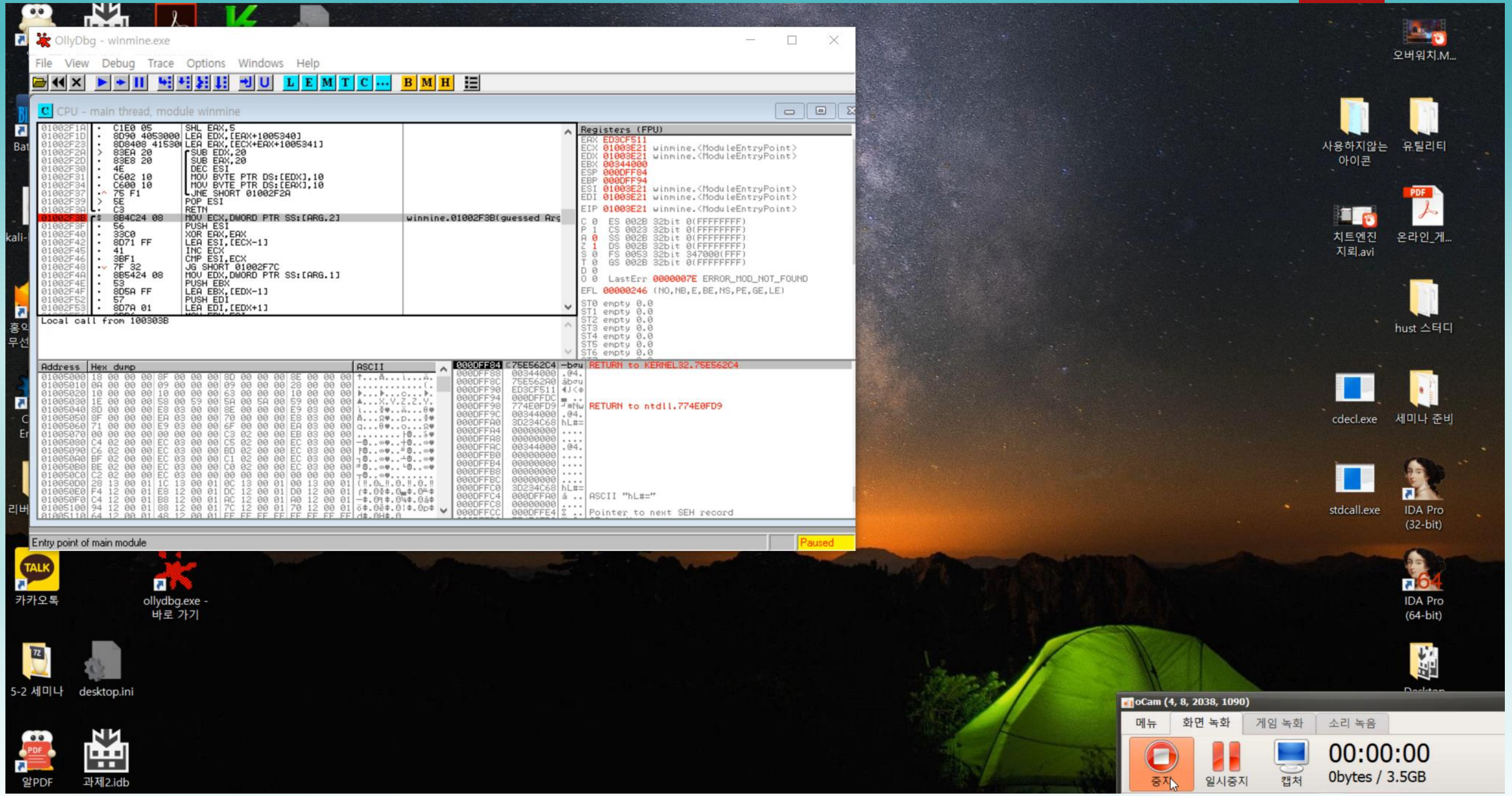
굳이 olly를 사용한 이유:
물론 ida로도 동적 분석이 가능하나 개인적 편의 상의 이유로olly를 사용함

```
01002F3B  $  8B4C24 08      MOV ECX,DWORD PTR SS:[ARG.2]      winmine.01002F3B(guessed Arg1,Arg2)
01002F3F  .  56             PUSH ESI
01002F40  .  33C0           XOR EAX,EAX
01002F42  .  8D71 FF        LEA ESI,[ECX-1]
01002F45  .  41             INC ECX
01002F46  .  3BF1           CMP ESI,ECX
01002F48  .v 7F 32          JG SHORT 01002F7C
01002F4A  .  8B5424 08      MOV EDX,DWORD PTR SS:[ARG.1]
01002F4E  .  53             PUSH EBX
01002F4F  .  8D5A FF        LEA EBX,[EDX-1]
01002F52  .  57             PUSH EDI
01002F53  .  8D7A 01        LEA EDI,[EDX+1]
01002F56  .  8BD6           MOV EDX,ESI
01002F58  .  C1E2 05        SHL EDX,5
01002F5B  .  2BCE           SUB ECX,ESI
01002F5D  .  81C2 4053000   ADD EDX,OFFSET 01005340
01002F63  .  41             INC ECX
01002F64  >  8BF3           MOV ESI,EBX
01002F66  .v EB 08          JMP SHORT 01002F70
01002F68  >  F60432 80      TEST BYTE PTR DS:[ESI+EDX],80
01002F6C  .v 74 01          JE SHORT 01002F6F
01002F6E  .  40             INC EAX
01002F6F  >  46             INC ESI
01002F70  >  3BF7           CMP ESI,EDI
01002F72  .^ 7E F4          JLE SHORT 01002F68
01002F74  .  83C2 20        ADD EDX,20
01002F77  .  49             DEC ECX
01002F78  .^ 75 EA          JNE SHORT 01002F64
01002F7A  .  5F             POP EDI
01002F7B  .  5B             POP EBX
01002F7C  >  5E             POP ESI
01002F7D  .  C2 0800        RETN 8
01002F80  $  A1 38530001    MOV EAX,DWORD PTR DS:[1005338]    winmine.01002F80(guessed Arg1)
01002F85  .  83F8 01        CMP EAX,1
01002F88  .v 7C 4E          JL SHORT 01002FD8
01002F8A  .  53             PUSH EBX
```

브레이크 포인트
걸고 실행

Count Bombs

- 빈칸을 클릭 했을 때 주변을 탐색해 주변 지뢰의 개수를 숫자로 표시한다.


OllyDbg - winmine.exe
File View Debug Trace Options Windows Help
CPU - main thread, module winmine
01002F1A · C1E0 05 SHL EAX,5
01002F1D · 8D90 4053000 LEA EDX,[EAX+1005340]
01002F23 · 8D8408 41530 LEA EAX,[ECX+EAX+1005341]
01002F2A > 83EA 20 SUB EDX,20
01002F2D · 83E8 20 SUB EAX,20
01002F30 · 4E DEC ESI
01002F31 · C602 10 MOV BYTE PTR DS:[EDX],10
01002F34 · C600 10 MOV BYTE PTR DS:[EAX],10
01002F37 · 75 F1 JNE SHORT 01002F2A
01002F39 > 5E POP ESI
01002F3A · C3 RETN
01002F3B $ 8B4C24 08 MOV ECX,DWORD PTR SS:[ARG.2] winmine.01002F3B(guessed Arg
01002F3F · 56 PUSH ESI
01002F40 · 33C0 XOR EAX,EAX
01002F42 · 8D71 FF LEA ESI,[ECX-1]
01002F45 · 41 INC ECX
01002F46 · 3BF1 CMP ESI,ECX
01002F48 · 7F 32 JG SHORT 01002F7C
01002F4A · 8B5424 08 MOV EDX,DWORD PTR SS:[ARG.1]
01002F4E · 53 PUSH EBX
01002F4F · 8D5A FF LEA EBX,[EDX-1]
01002FE2 · 57 PUSH EDI
01002FE3 · 8D7A 01 LEA EDI,[EDX+1]
Local call from 100303B
Registers (FPU)
EAX ED3CF511
ECX 01003E21 winmine.<ModuleEntryPoint>
EDX 01003E21 winmine.<ModuleEntryPoint>
EBX 00344000
ESP 000DFF84
EBP 000DFF94
ESI 01003E21 winmine.<ModuleEntryPoint>
EDI 01003E21 winmine.<ModuleEntryPoint>
EIP 01003E21 winmine.<ModuleEntryPoint>
C 0 ES 002B 32bit 0(FFFFFFFF)
P 1 CS 0023 32bit 0(FFFFFFFF)
A 0 SS 002B 32bit 0(FFFFFFFF)
Z 1 DS 002B 32bit 0(FFFFFFFF)
S 0 FS 0053 32bit 347000(FFF)
T 0 GS 002B 32bit 0(FFFFFFFF)
D 0
O 0 LastErr 0000007E ERROR_MOD_NOT_FOUND
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
Address Hex dump ASCII
000DFF84 75E562C4 RETURN to KERNEL32.75E562C4
000DFF98 774E0FD9 RETURN to ntdll.774E0FD9
000DFFC0 3D234C68 hL#=
000DFFC4 000DFFA0 ASCII "hL#="
000DFFCC 000DFFE4 Pointer to next SEH record
Entry point of main module
Paused
오버워치.M...
사용하지않는 아이콘
유틸리티
치트엔진 지뢰.avi
온라인_게...
hust 스터디
cdecl.exe
세미나 준비
stdcall.exe
IDA Pro (32-bit)
IDA Pro (64-bit)
카카오톡
ollydbg.exe - 바로 가기
5-2 세미나
desktop.ini
알PDF
과제2.idb
oCam (4, 8, 2038, 1090)
메뉴
화면 녹화
게임 녹화
소리 녹음
중지
일시중지
캡처
00:00:00
0bytes / 3.5GB

발견했던 취약한 부분

- 변수 값들이 평문으로 저장 되어있어 쉽게 검색됨
- 디버깅 모드로 컴파일 되있어 함수명,변수명등이 그대로 노출
- 프로그램이 실행 할 때마다 매번 같은 주소에 스택과 힙이 배치되어 분석이 용이함

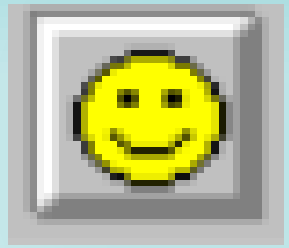
보안 방법

1. 컴파일시 디버깅 모드가 아닌 릴리즈 모드로 컴파일

   디버깅 정보가 포함되지 않아 분석에 좀 더 오래 걸린다

- 2. ASLR (Address Space Layout Randomization)
- 공격자가 메모리상의 주소를 예측하기 어렵게 만들어 메모리상의 공격을 어렵게 합니다.

- 3. 변수의 별도의 연산을 통해 읽기 어렵게 함
- 예) 데이터를 담는 변수 C_sec을 sec1, sec2, sec3로 나누어 저장 했다가 필요할때 합치는 연산을 통해 검색이 어렵도록 합니다.

Thanks you! 

Q & A