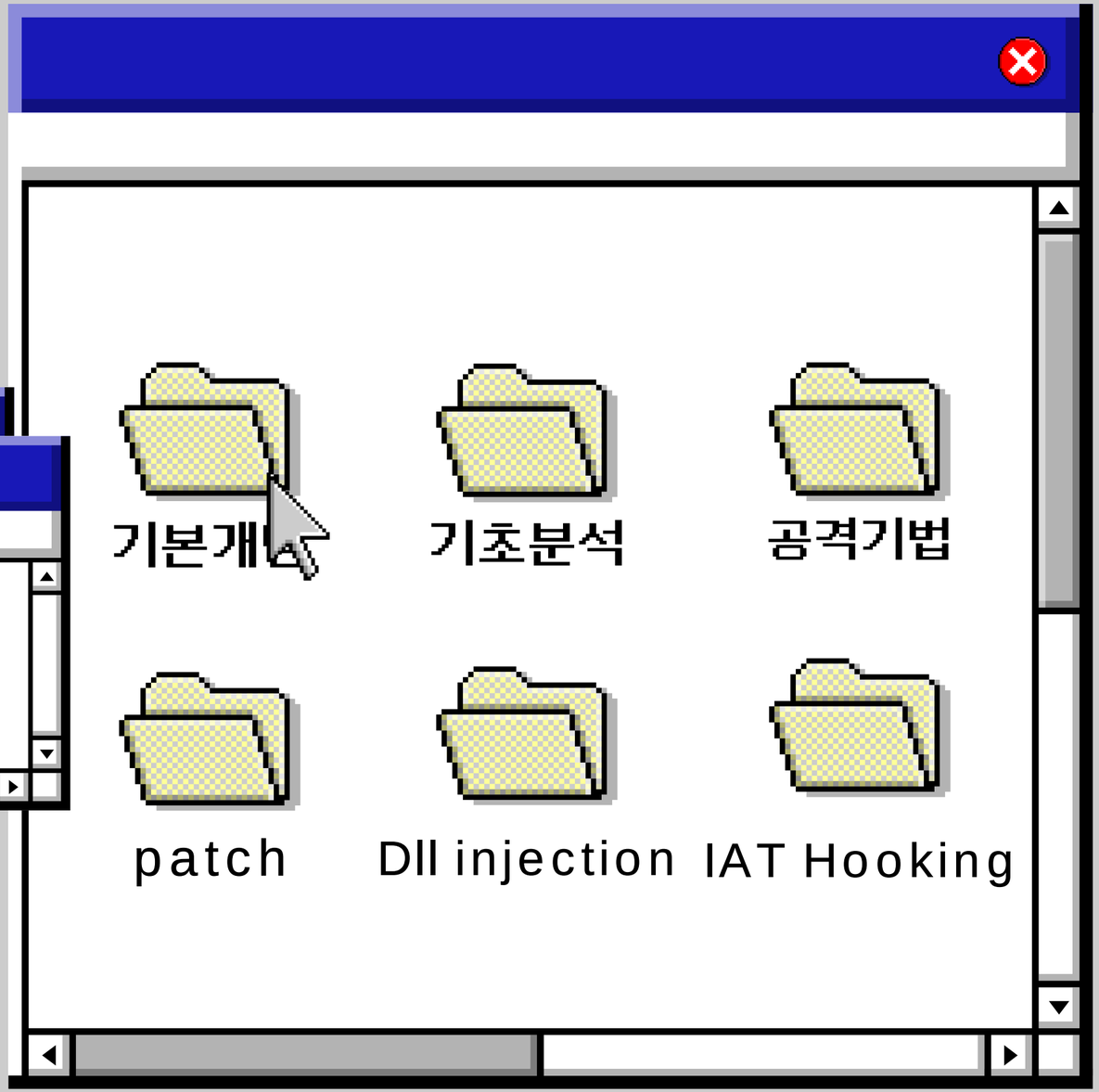
Minesweeper
해킹해보자

GUnT_0x9 | ogh09 | Mas$y_J!

05:23
PM

목차



기본개념

winmine.exe

reversing

Back to Agenda Page

기본개념

winmine.exe ?

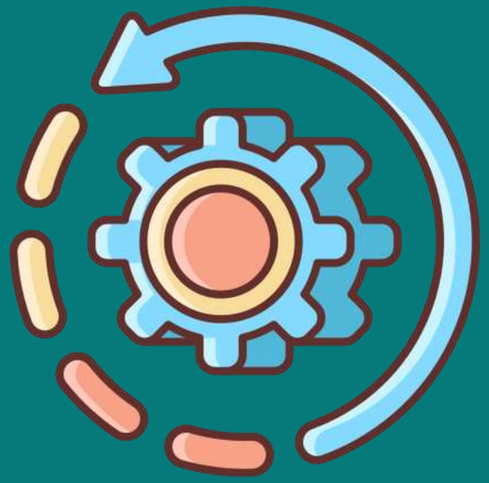
winmine.exe는
윈도우xp 버전에 내장된
지뢰찾기게임의 실행파일로
이번에 해킹할 파일이다.

Back to Agenda Page

기본개념

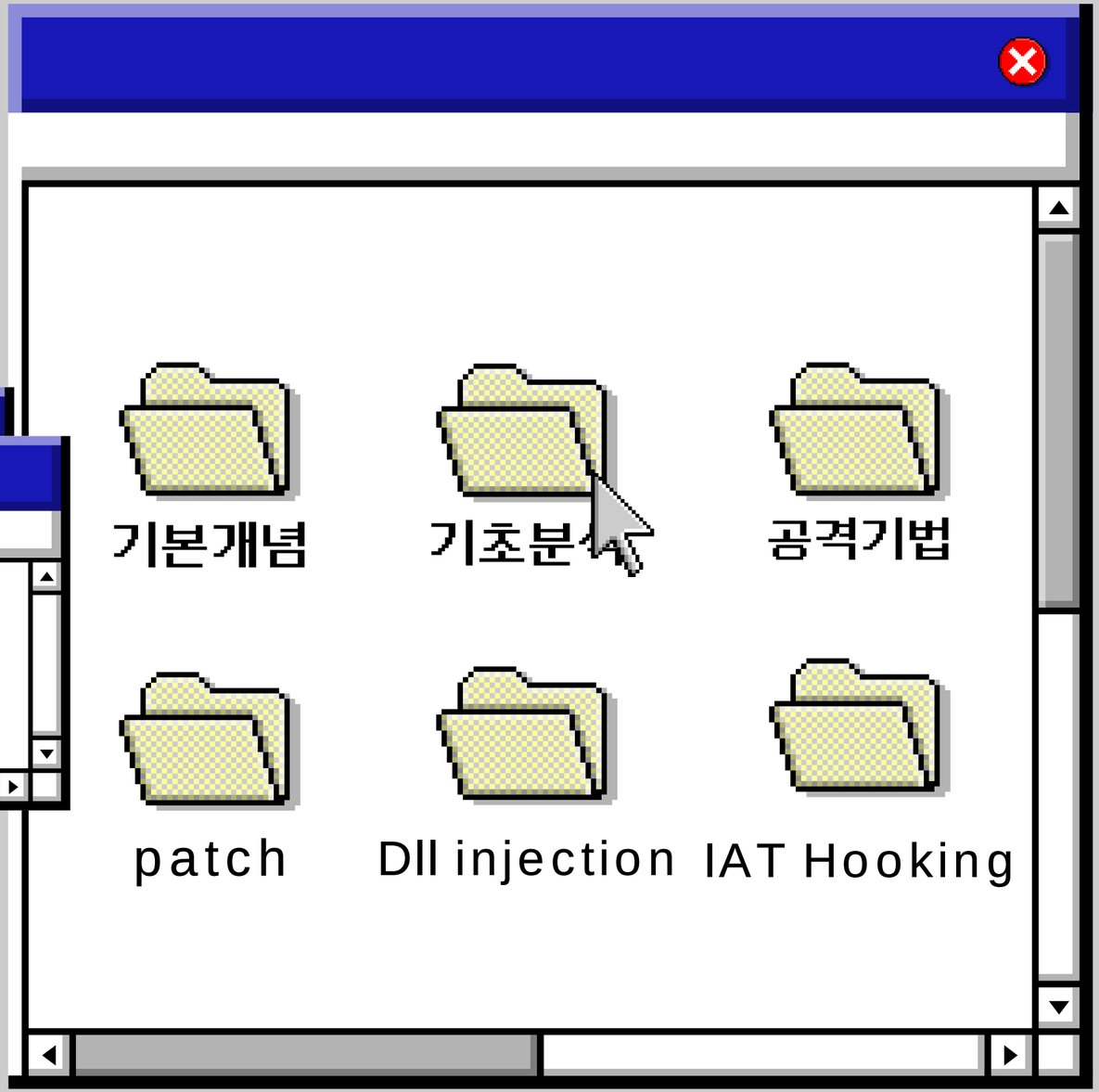
reversing ?

리버싱이란 말그대로
역공학이라고 할수있다.
완성된 파일을 뜯어봐서
내가 원하는 형태로
수정해볼수있다.



Back to Agenda Page

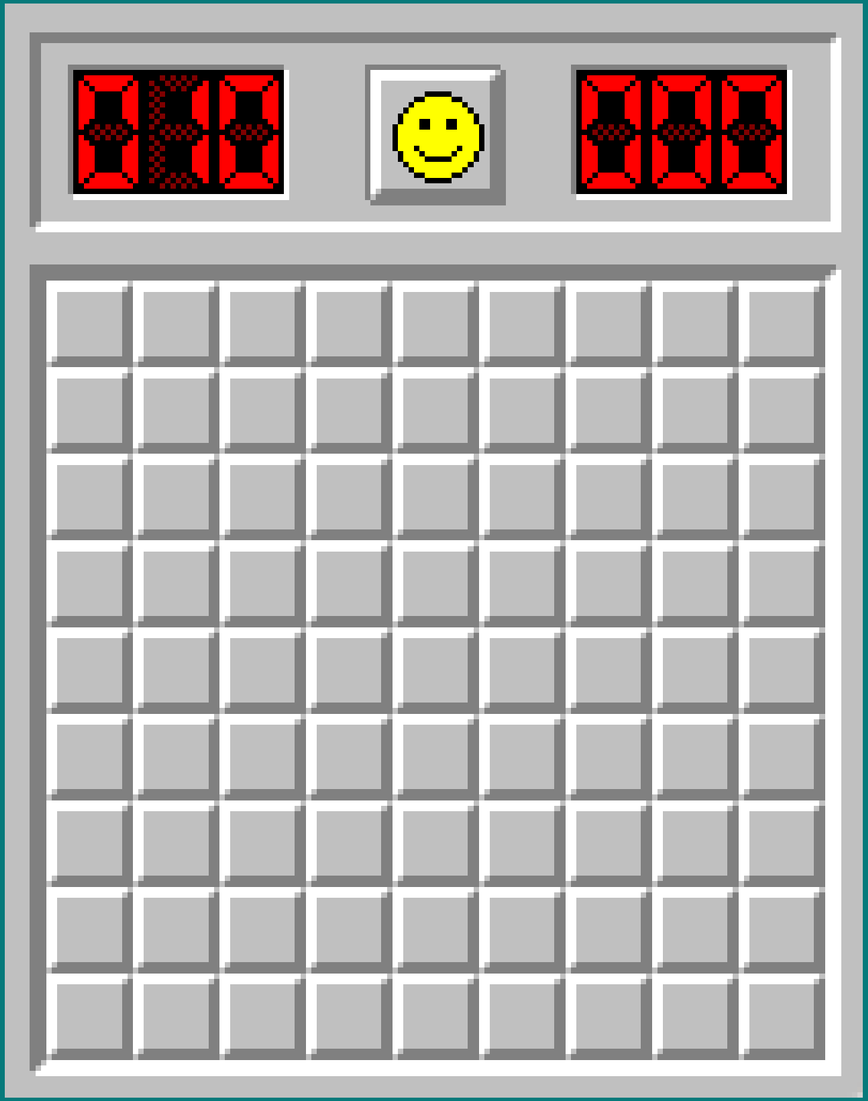
목차



BitBlt 함수

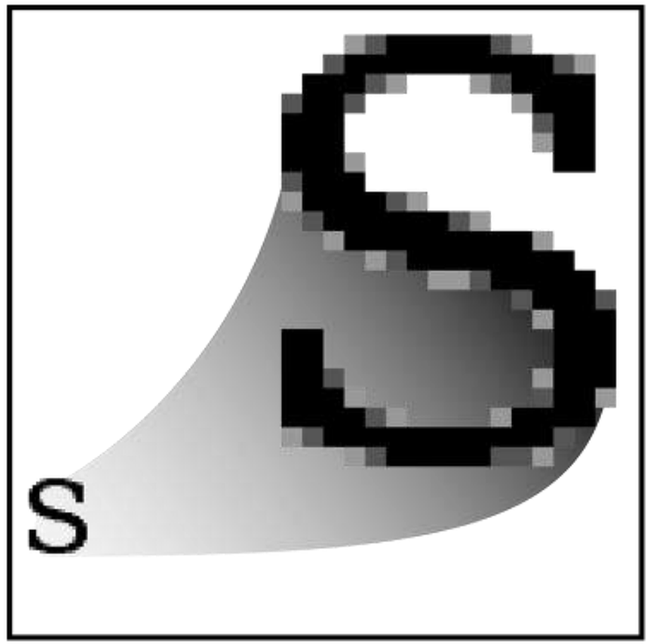
비트맵을 불러오는 함수이다.

지뢰찾기 전체에 걸쳐 사용된다.



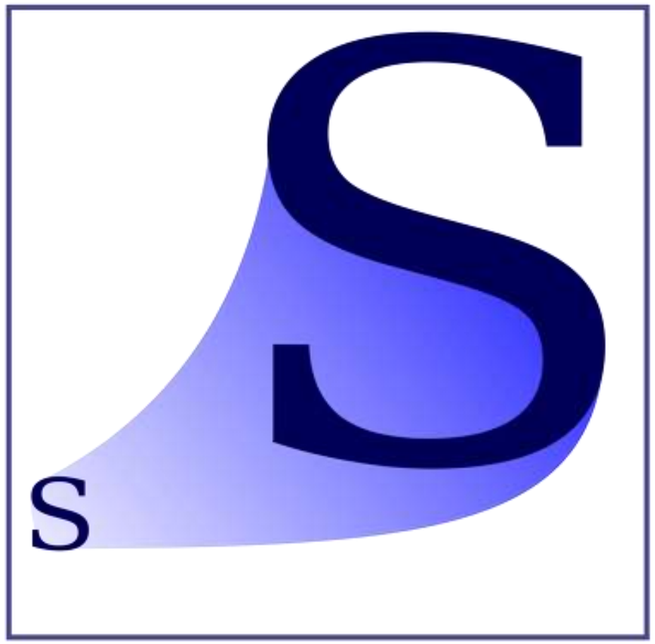
Back to Agenda Page

비트맵



Raster(Bitmap)

.jpeg .gif .png



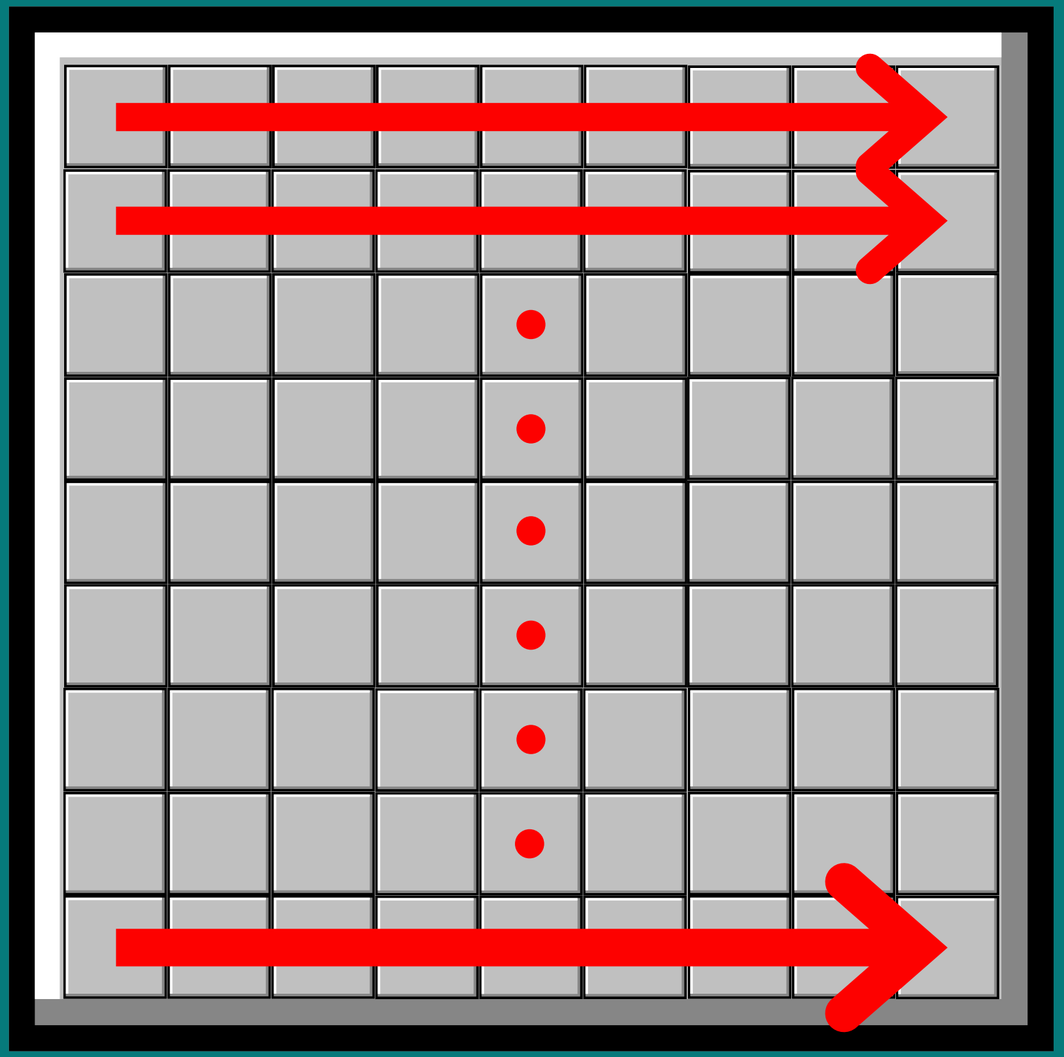
Vector

.svg

Back to Agenda Page

탐색/생성 방식

왼쪽 → 오른쪽



Back to Agenda Page

1005340

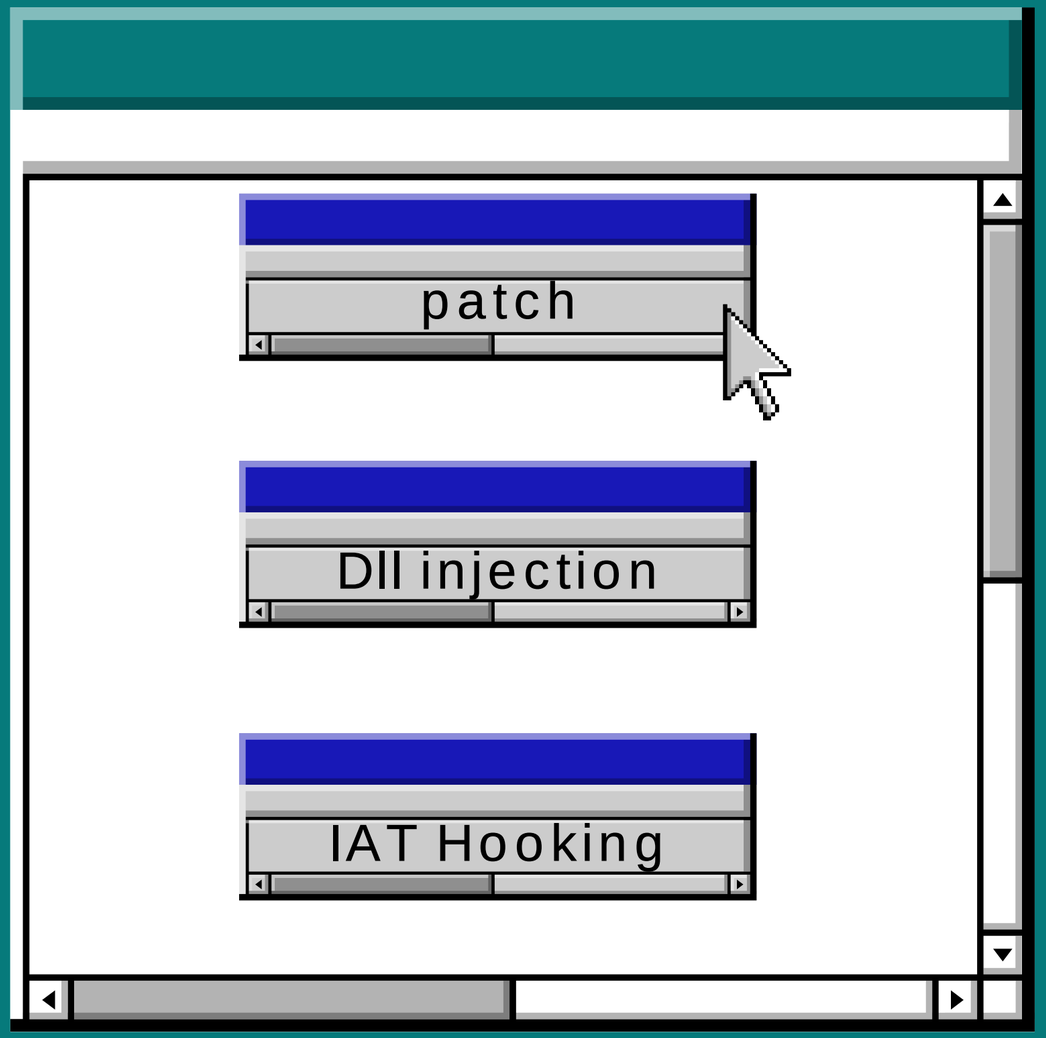
005340	10	10	10	10	10	10	10	10	10	10	10
005360	10	0F	0F	0F	0F	0F	8F	0F	0F	0F	10
005380	10	0F	0F	8F	0F	0F	8F	0F	8F	0F	10
0053A0	10	0F	0F	0F	0F	0F	0F	8F	8F	0F	10
0053C0	10	0F	0F	0F	8F	0F	0F	0F	0F	0F	10
0053E0	10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
005400	10	8F	0F	0F	0F	0F	0F	0F	8F	0F	10
005420	10	0F	8F	0F	0F	0F	0F	0F	0F	0F	10
005440	10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
005460	10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
005480	10	10	10	10	10	10	10	10	10	10	10

지뢰의 배열이
저장되는 위치

Back to Agenda Page

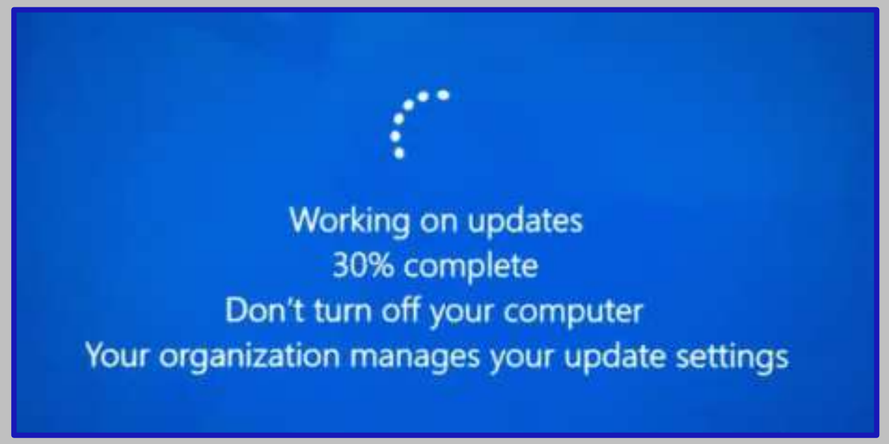
목차

공격기법



Back to Agenda Page

Patch(패치)



Windows의
hotfix 업데이트

게임의
신규 업데이트

Back to Agenda Page

Patch(패치)

간단히 말해, 완성된 프로그램 수정하는것!

Back to Agenda Page

Patch(패치)

Back to Agenda Page

Patch(패치)의 장/단점

· 장점 · ex.에어본 제로클릭

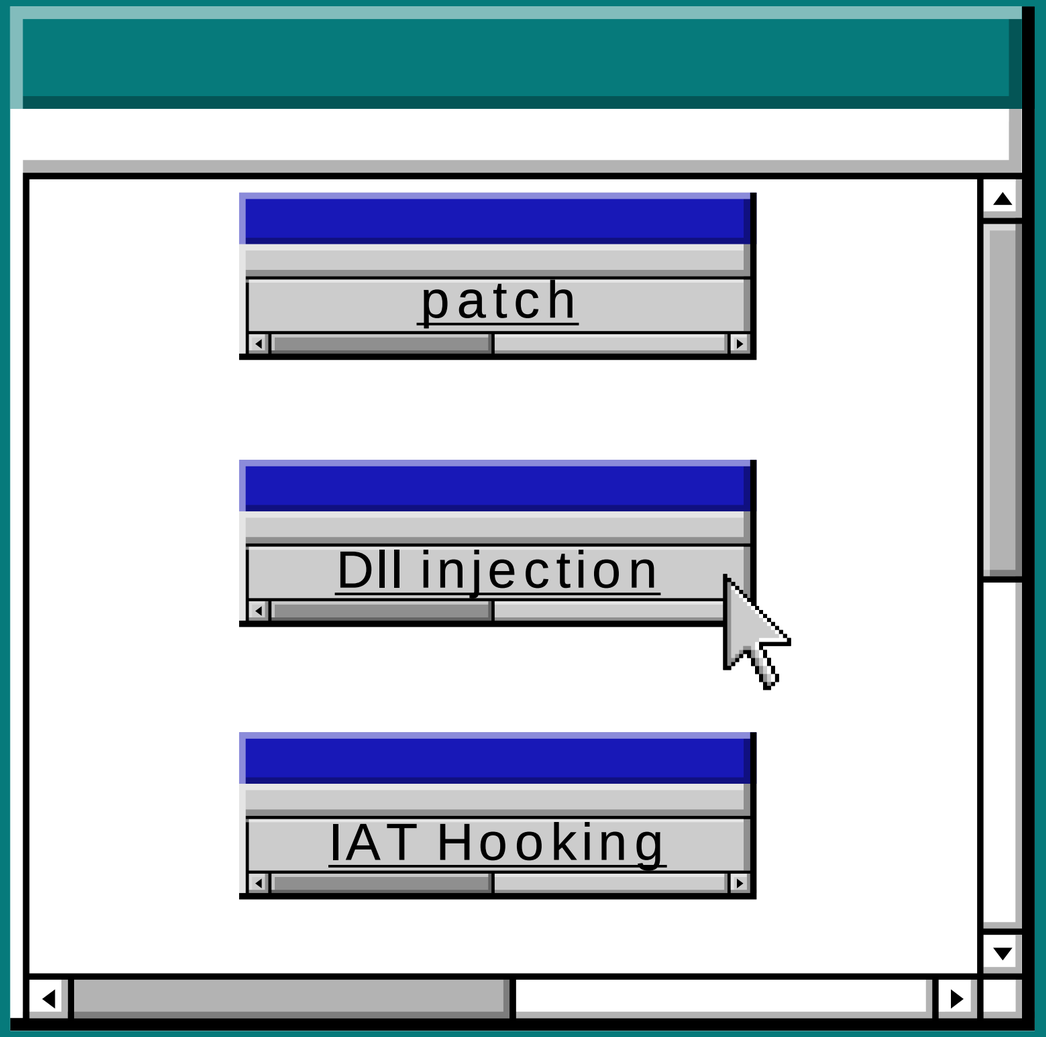
빠른 업데이트로 보안취약점등을 막을수 있다

· 단점 · ex.크라우드 스트라이

임의로 패치를 진행하게되면 프로그램이 정상적으로 작동하지 않을수도 있다

Back to Agenda Page

공격기법



Back to Agenda Page

<DLL 이란?>

explanation

DLL이란 윈도우운영체제에서 사용되는 동적연결라이브러리로 여러 프로그램이 공통으로 사용하는 함수

Back to Agenda Page

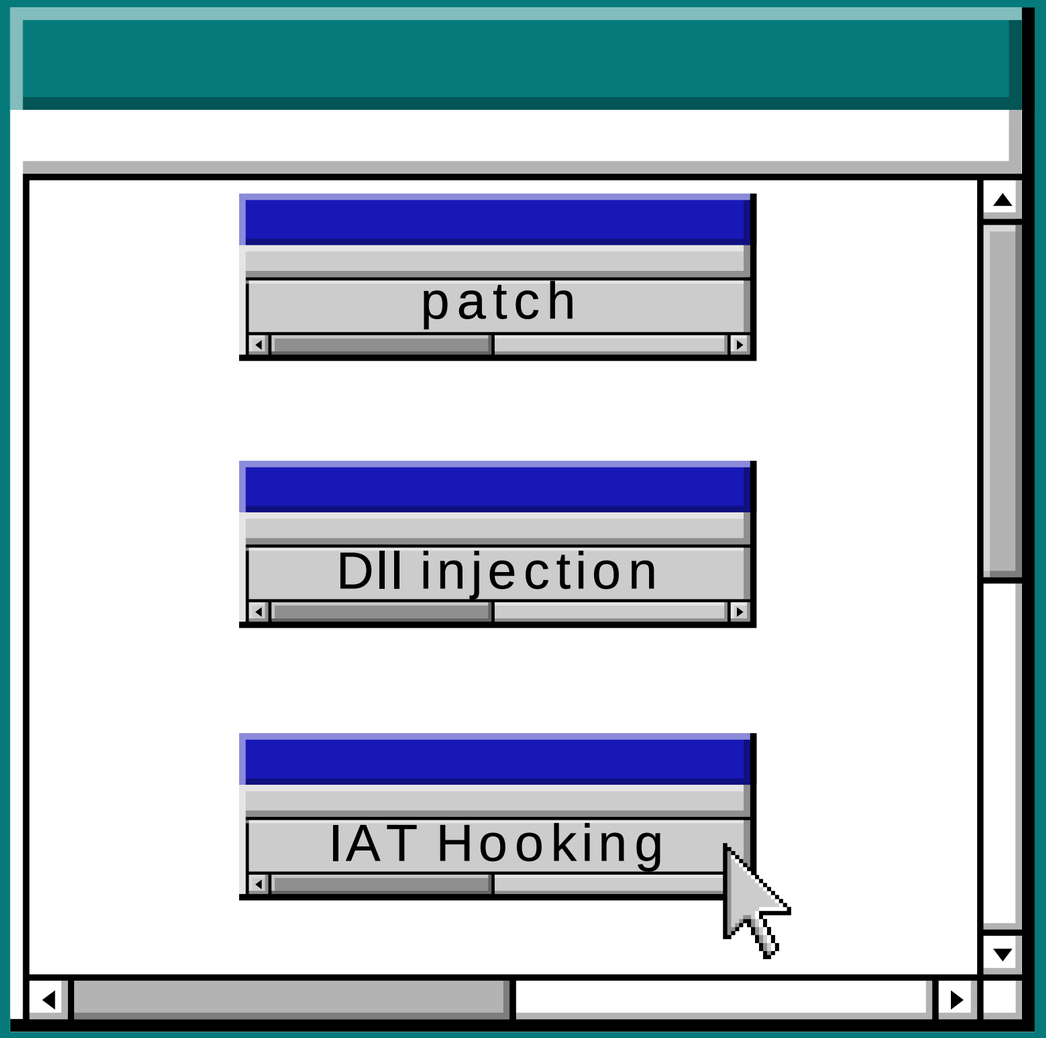
<DLL injection이란?>

explanation

DLL Injection은 실행 중인 다른 프로세스의 내가 만든 파일을 강제 삽입

Back to Agenda Page

공격기법



Back to Agenda Page

hooking?

Hooking이란 프로그램에 이벤트나 메세지에 흐름을 가로채서 자신이 원하는 코드로 바꿀수있는 해킹공격기법이다

Back to Agenda Page

후킹 개념

A

Back to Agenda Page

후킹 개념

A

정상적인 흐름

B

Back to Agenda Page

후킹 개념

A

점삼(?)적인 흐름

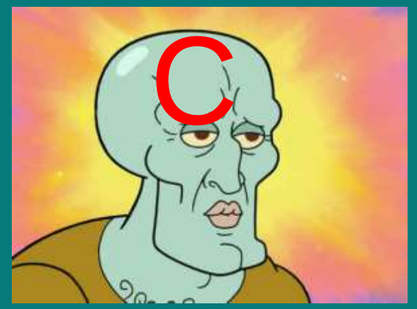
B

Back to Agenda Page

후킹 개념

A

내가 만든 코드



B

비정상적인 흐름

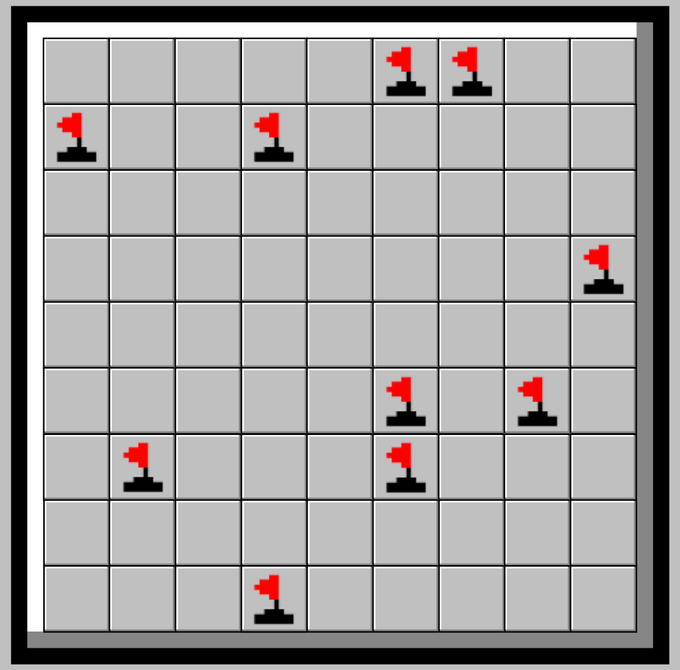
Back to Agenda Page

목차

explanation

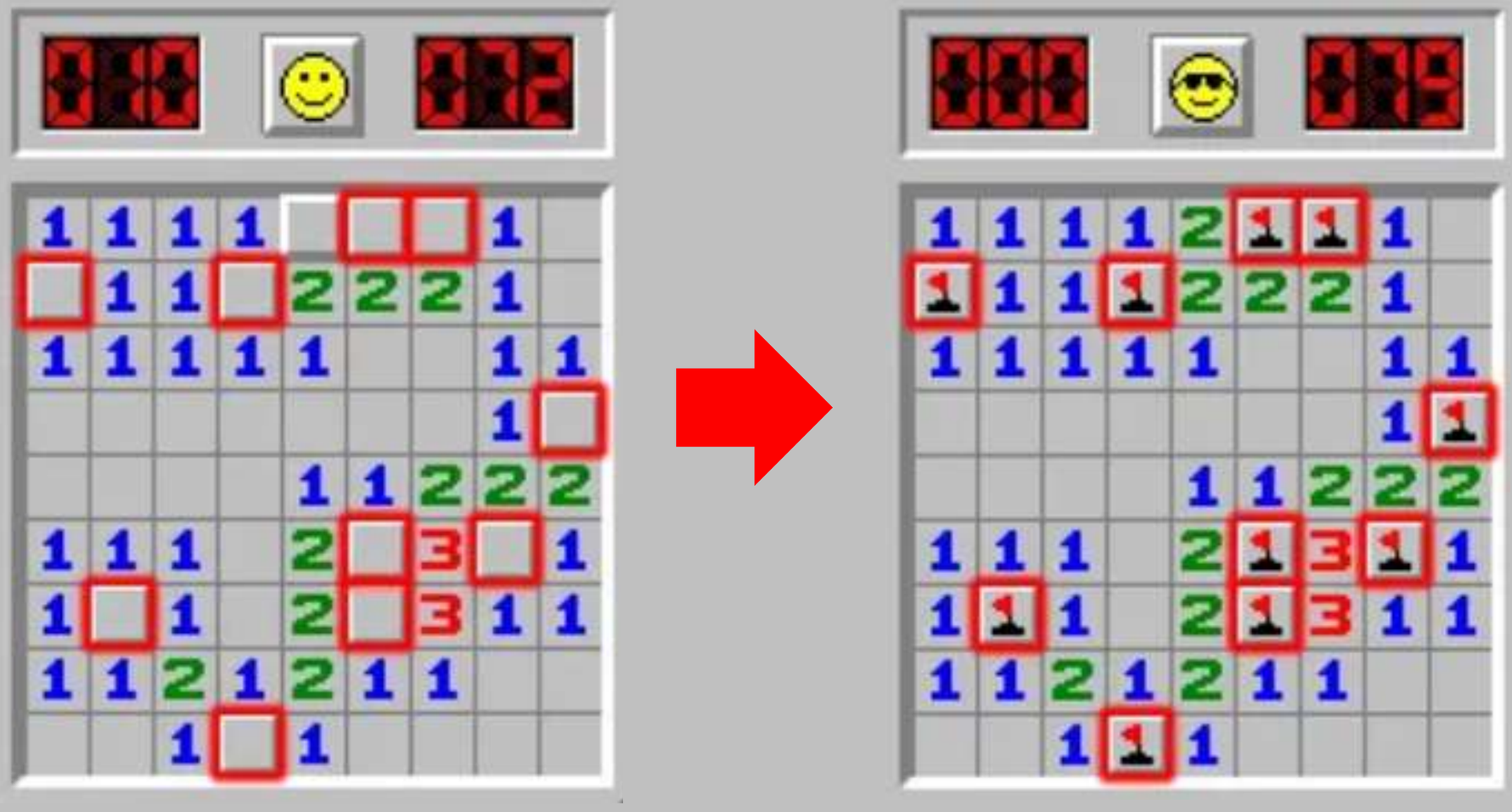
개요

모든 지뢰에 깃발 세우기 (클릭시)



바로 승리 하기 (클릭시)

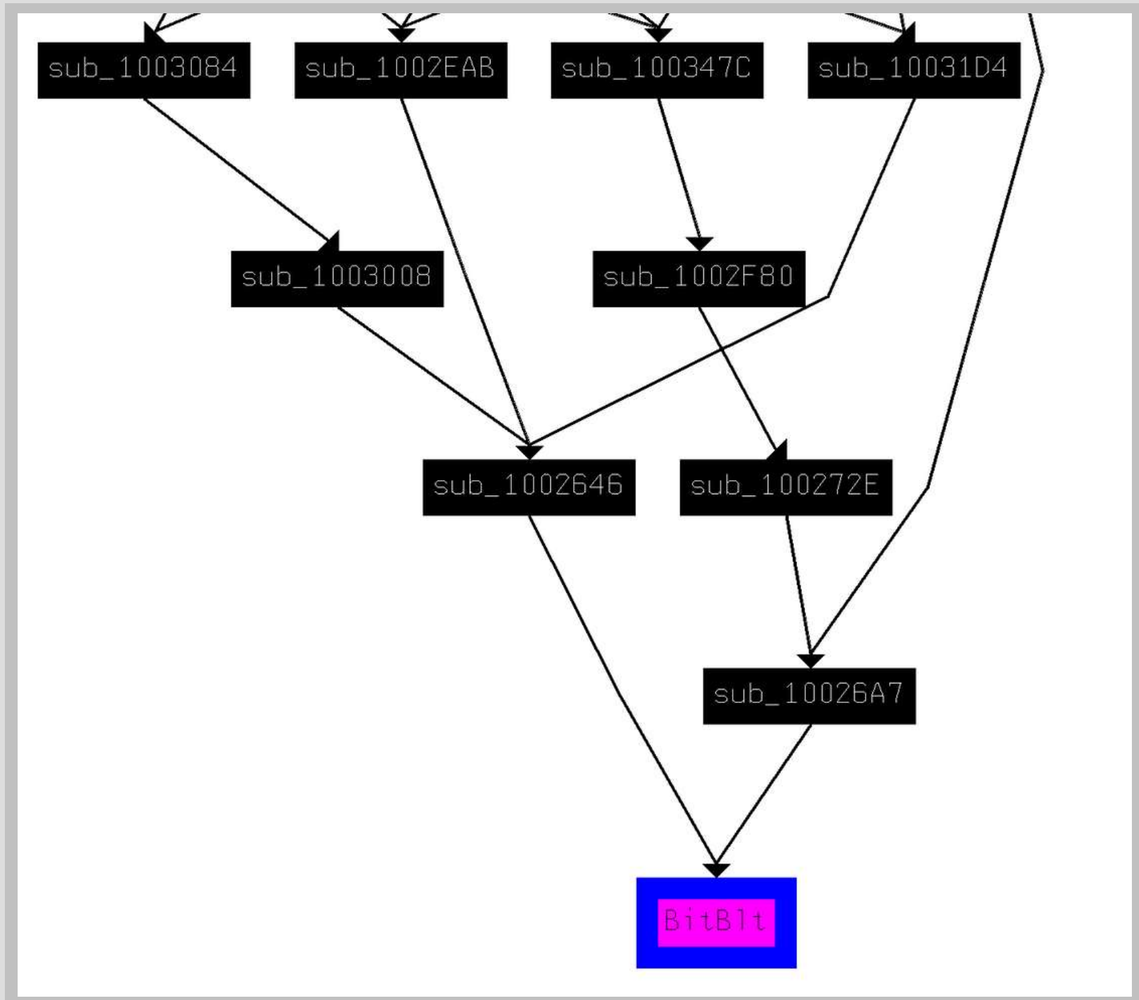
모든 지뢰에 깃발이 세워지는 조건



1. 지뢰가 아닌 모든 칸 클릭

2. 지뢰찾기 게임 승리

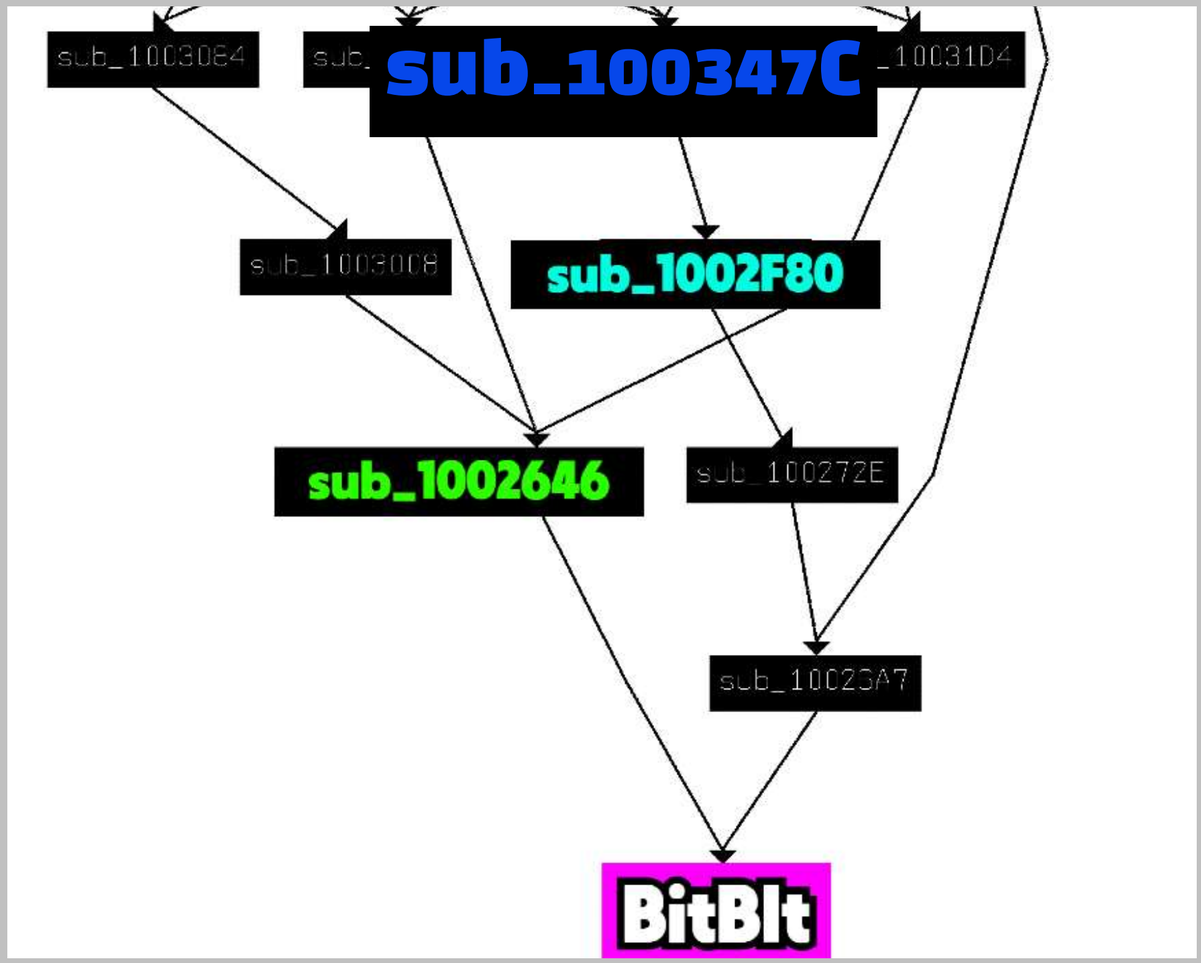
BitBlt 함수를 사용하는 함수



BitBlt을 호출하는 함수
와
상위 함수

BitBlt를 호출하는 클릭, 깃발, 승리와 관련한 함수

- ★ 347C
- ★ 2F80
- ★ 2646



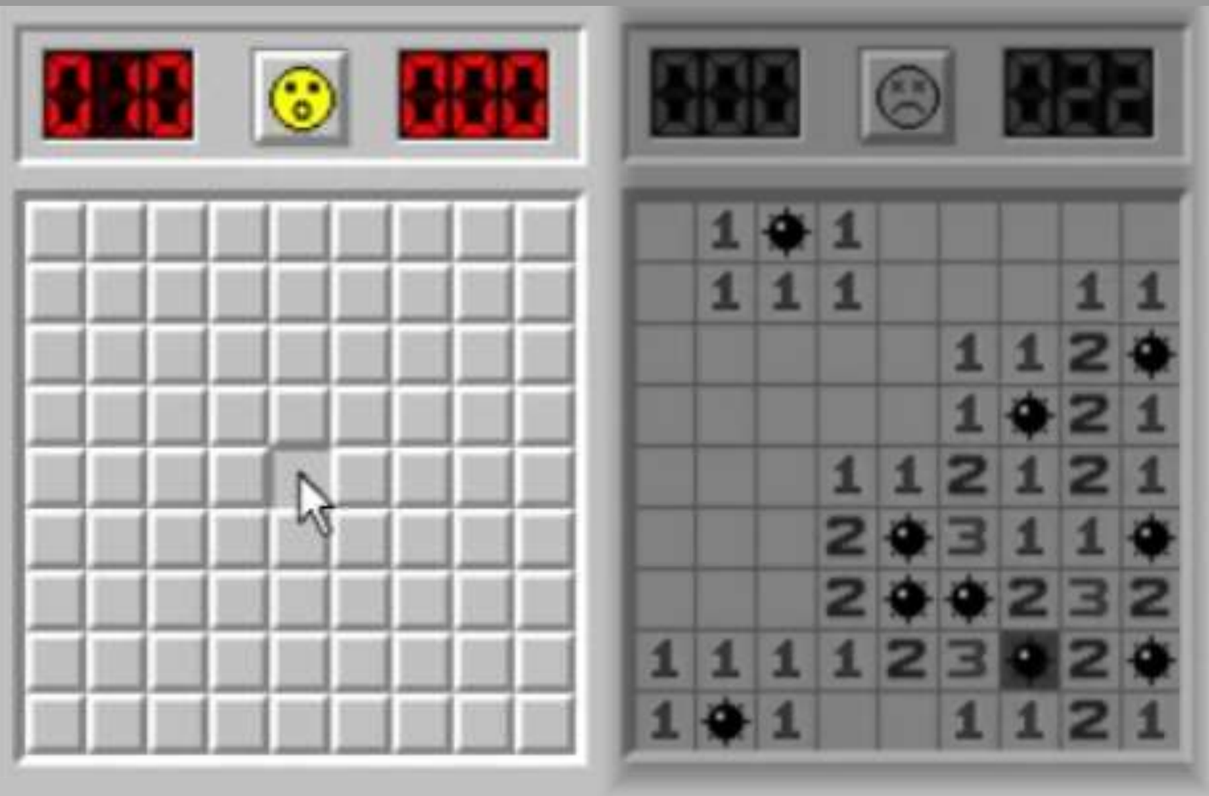
분석 & 결과 - 1

★ 264 - 1칸 비트맵 교체

```
short loc_1003415

.text:01003400 mov     edi, dword_100511C
.text:01003401 mov     esi, dword_1005118
.text:01003407 push    edi
.text:01003408 push    esi
.text:01003409 call    sub_100316B
.text:0100340D push    edi
.text:0100340E push    esi
.text:01003410 call    sub_1002646      ; 클릭
```

sub_1002646

이 위치는 클릭과 관련됨!



분석 & 결과 - 2

★ 347 - 게임에서 승리했을때만

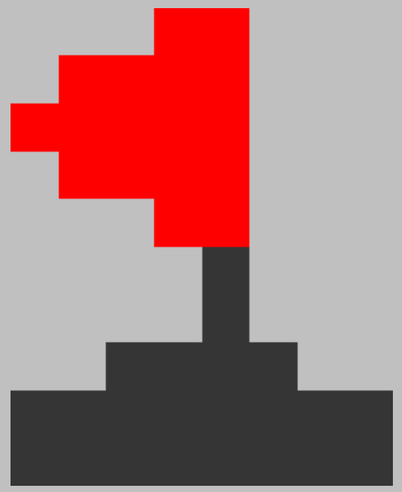
sub_100347C 내부

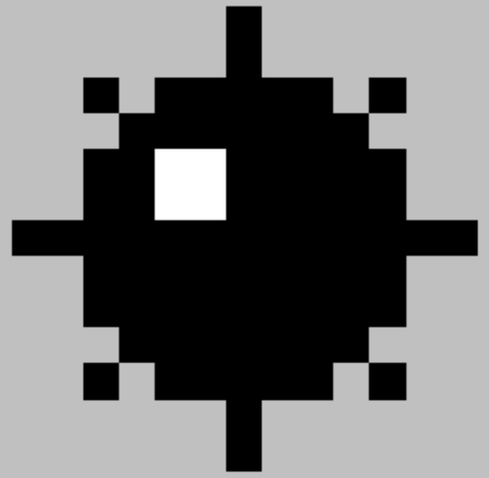
```
sub_100347C proc near

arg_0= dword ptr  4

and     dword_1005164, 0
push    esi
mov     esi, [esp+4+arg_0]
xor     eax, eax
test    esi, esi
setnz   al
inc     eax
inc     eax
push    eax
mov     dword_1005160, eax
call    sub_1002913
xor     eax, eax
test    esi, esi
setnz   al
lea     eax, ds:0Ah[eax*4]
push    eax
call    sub_1002F80
```

sub_1002F80

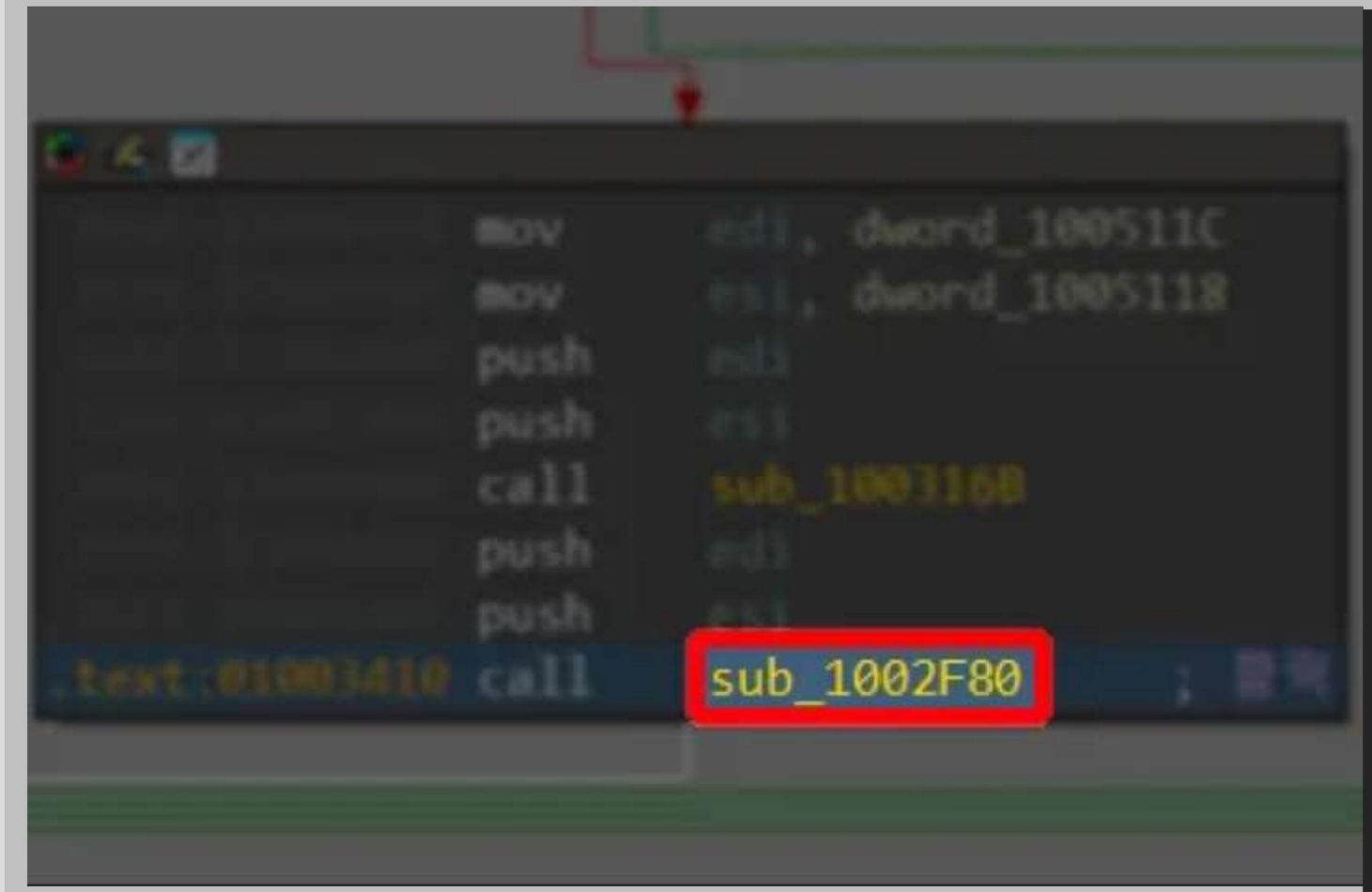
분석 & 결과 - 3

☆ 2F8 - 지뢰위치에 특정한 비트맵 가져옴





패치

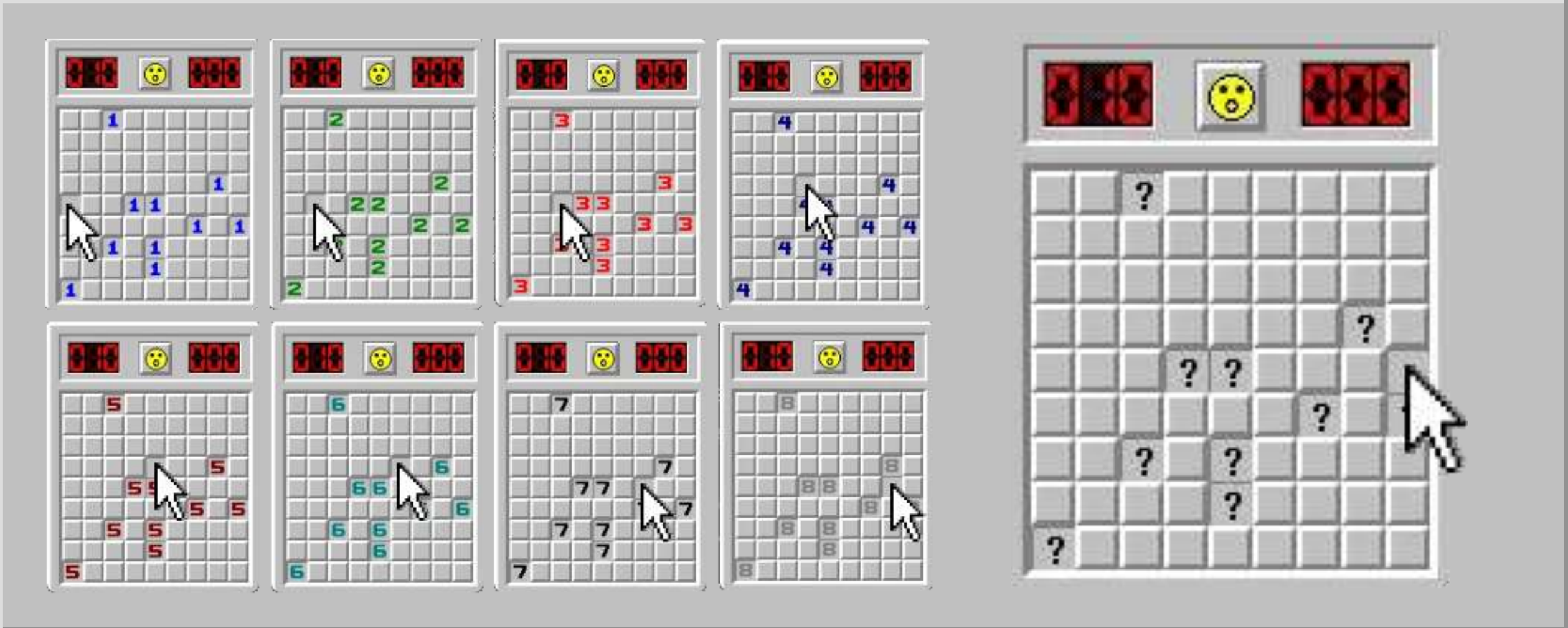


아까 찾았던 두 함수

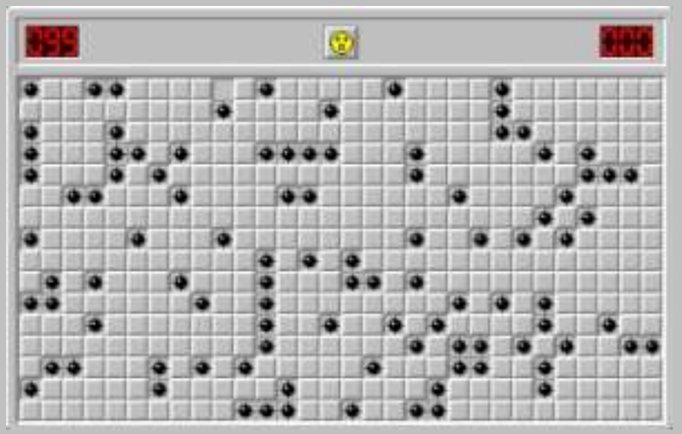
2646 → 2F80

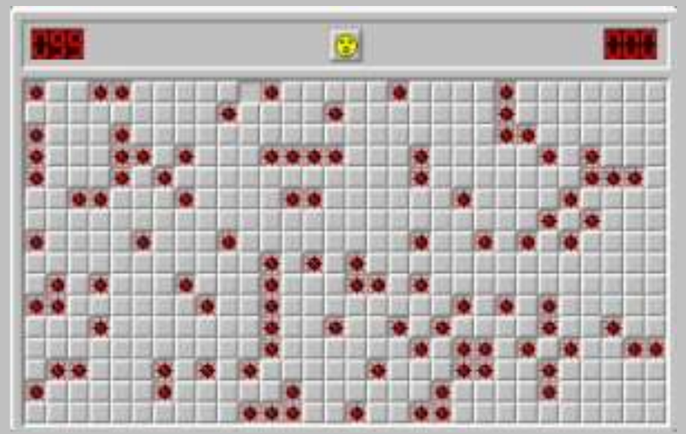
으로 패치

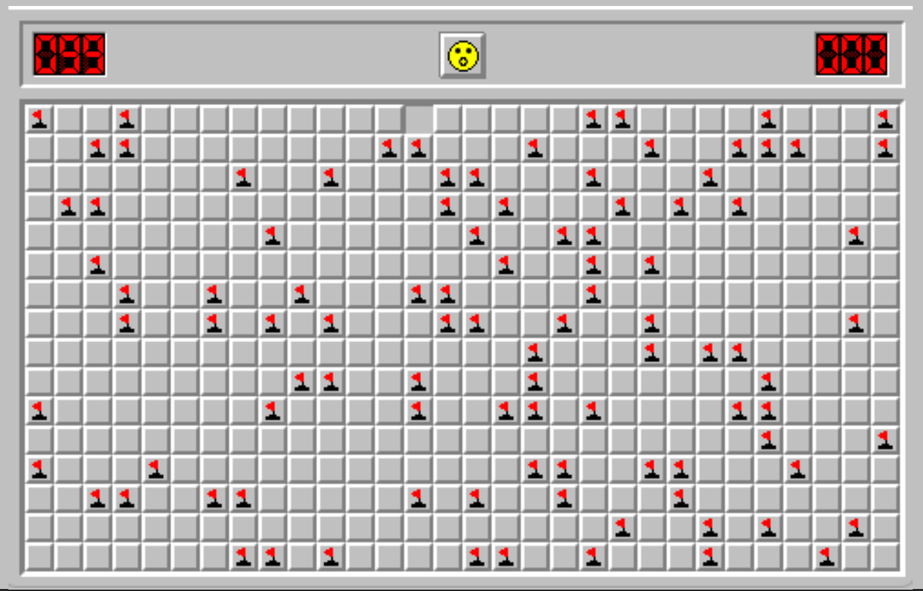
1차 패치 결과

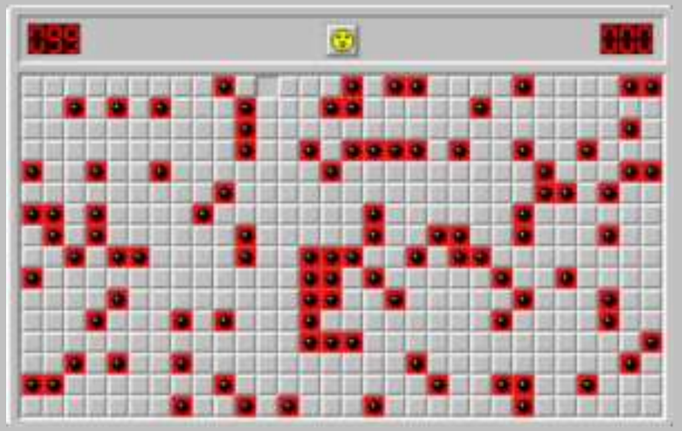


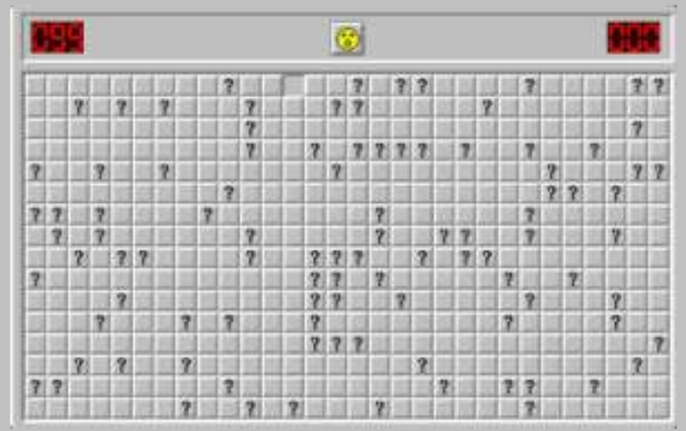
1차 패치 결과











총 비트맵은 14개

2F80함수 재분석

```
sub_1002F80 proc near

arg_0= byte ptr  4

mov     eax, dword_1005338
cmp     eax, 1
jl      short loc_1002FD8
```

```
push    ebx
push    esi
mov     esi, dword_1005334
push    edi
mov     edi, offset unk_1005360 ; 어떤 주소
mov     edx, eax
```

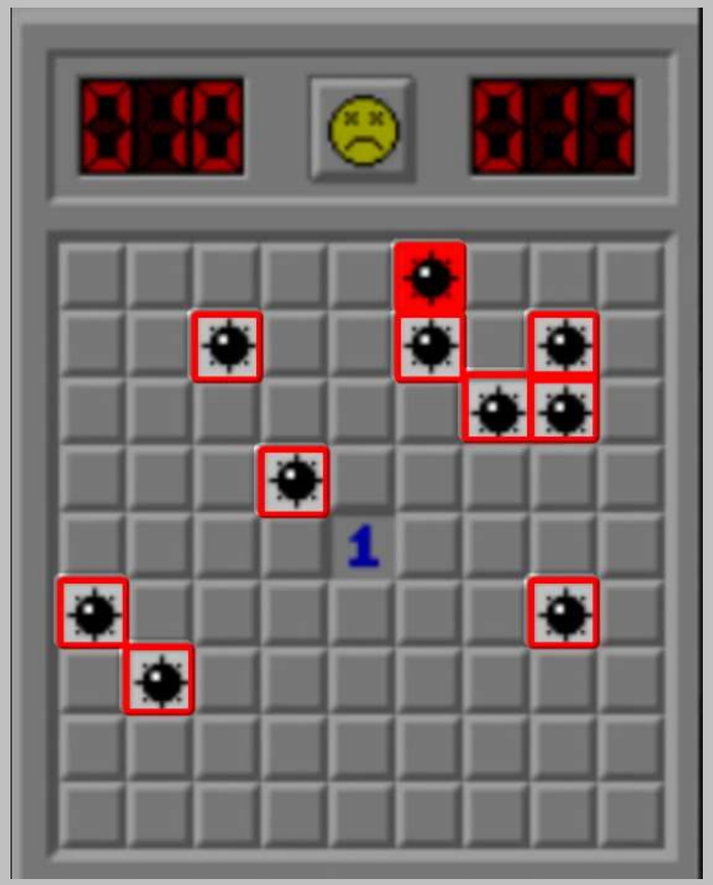
```
005340  10 10 10 10 10 10 10 10 10 10 10
005360  10 0F 0F 0F 0F 0F 8F 0F 0F 0F 10
005380  10 0F 0F 8F 0F 0F 8F 0F 8F 0F 10
0053A0  10 0F 0F 0F 0F 0F 0F 8F 8F 0F 10
0053C0  10 0F 0F 0F 8F 0F 0F 0F 0F 0F 10
0053E0  10 0F 0F 0F 0F 0F 0F 0F 0F 0F 10
005400  10 8F 0F 0F 0F 0F 0F 0F 8F 0F 10
005420  10 0F 8F 0F 0F 0F 0F 0F 0F 0F 10
005440  10 0F 0F 0F 0F 0F 0F 0F 0F 0F 10
005460  10 0F 0F 0F 0F 0F 0F 0F 0F 0F 10
                                 10 10 10 10
```

unk_1005360

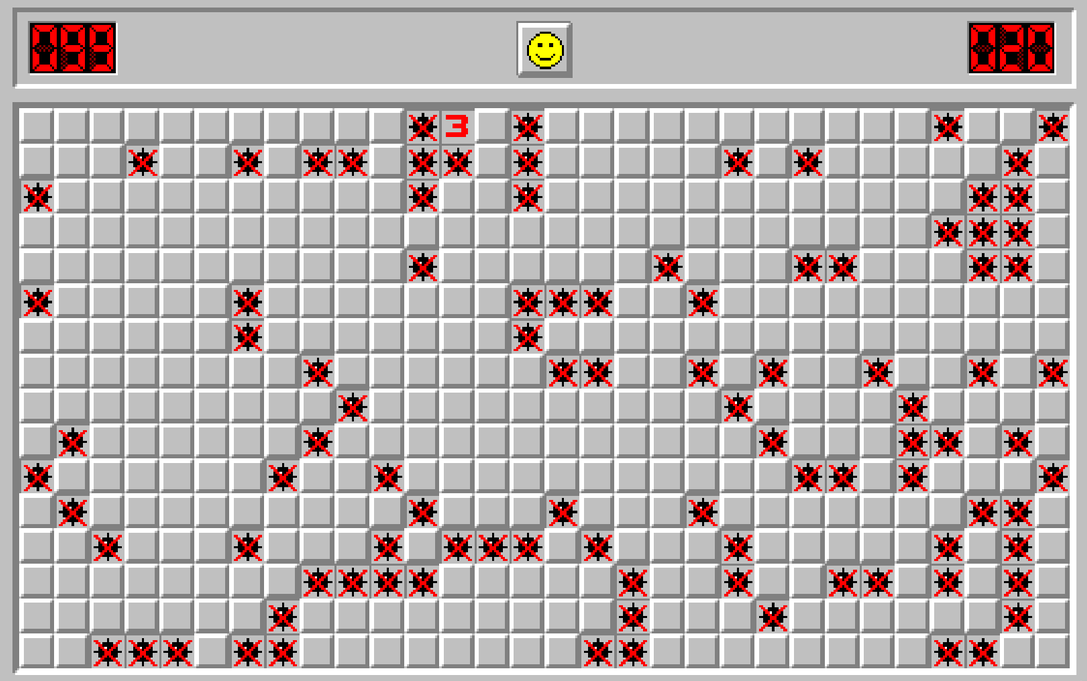
2F80함수 재분석

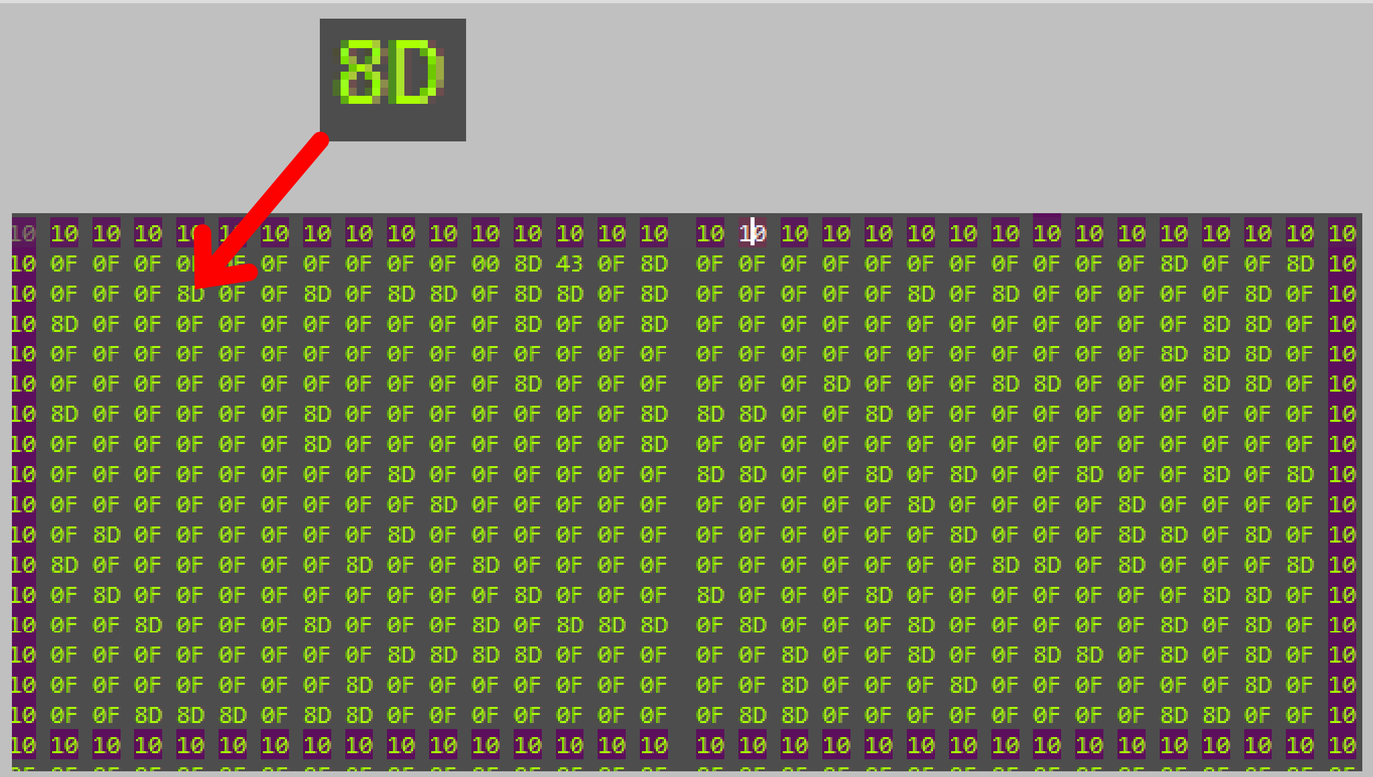
10	10	10	10	10	10	10	10	10	10	10
10	0F	0F	0F	0F	0F	8F	0F	0F	0F	10
10	0F	0F	8F	0F	0F	8F	0F	8F	0F	10
10	0F	0F	0F	0F	0F	0F	8F	8F	0F	10
10	0F	0F	0F	8F	0F	0F	0F	0F	0F	10
10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
10	8F	0F	0F	0F	0F	0F	0F	8F	0F	10
10	0F	8F	0F	0F	0F	0F	0F	0F	0F	10
10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
10	0F	0F	0F	0F	0F	0F	0F	0F	0F	10
10	10	10	10	10	10	10	10	10	10	10

=

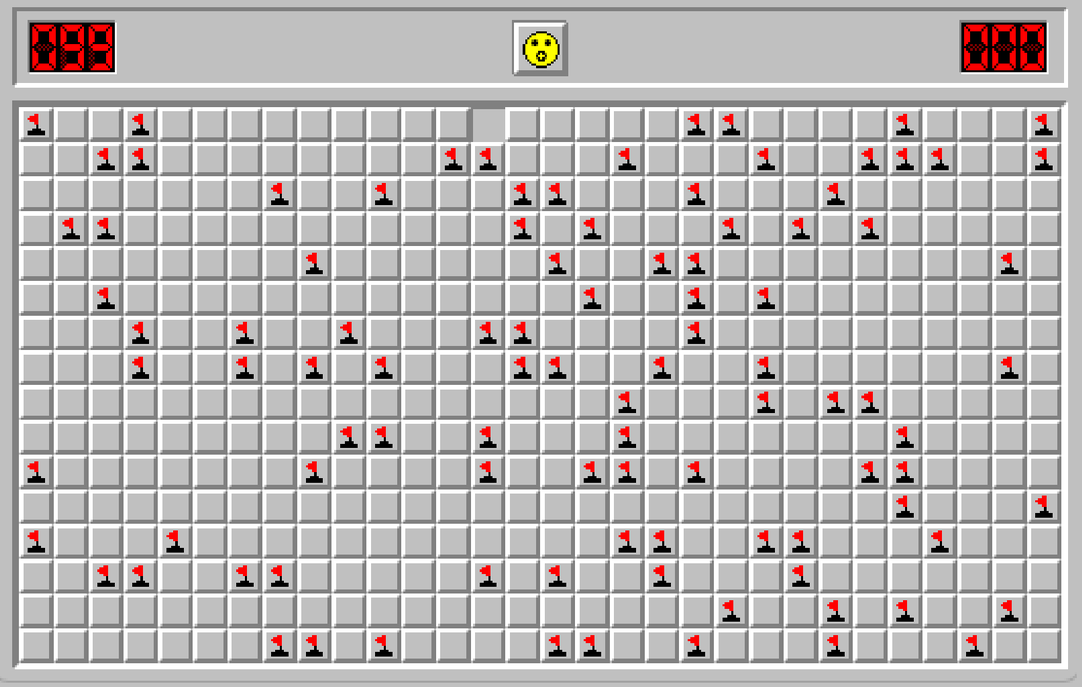


2F80함수 재분석

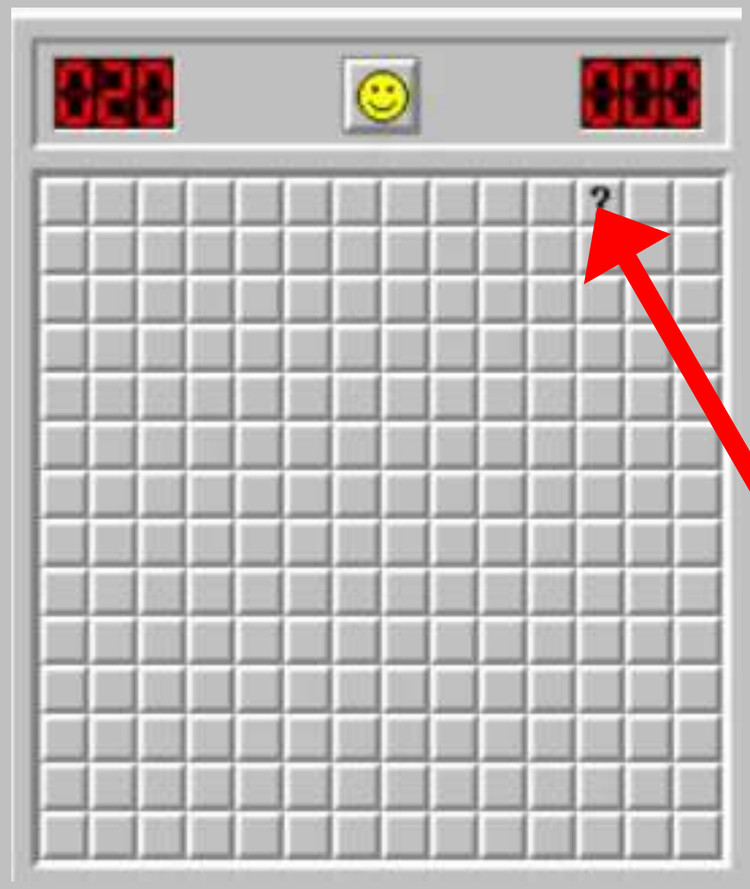




2F80함수 재분석



2F80함수 재분석



```
and     al, 0E0h
or      al, [esp+0Ch+arg_0]
jmp     short loc_1002FC7
```

지뢰일때만 오는 곳

EAX 00000080

EAX 0000008C

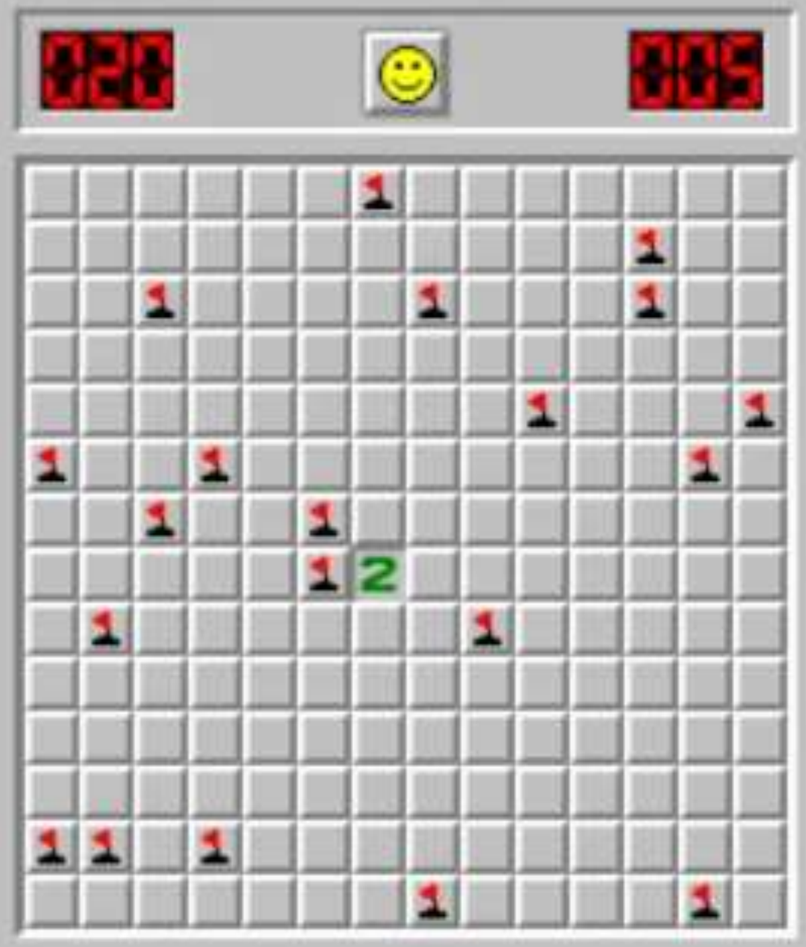
12번째 C (16진수)

수정

```
xor     al, al
mov     al, 8Eh
nop
nop
jmp     short loc_1002FC7
```

모든 부분을 지우고
8E 만 저장하게
패치하면?

성공



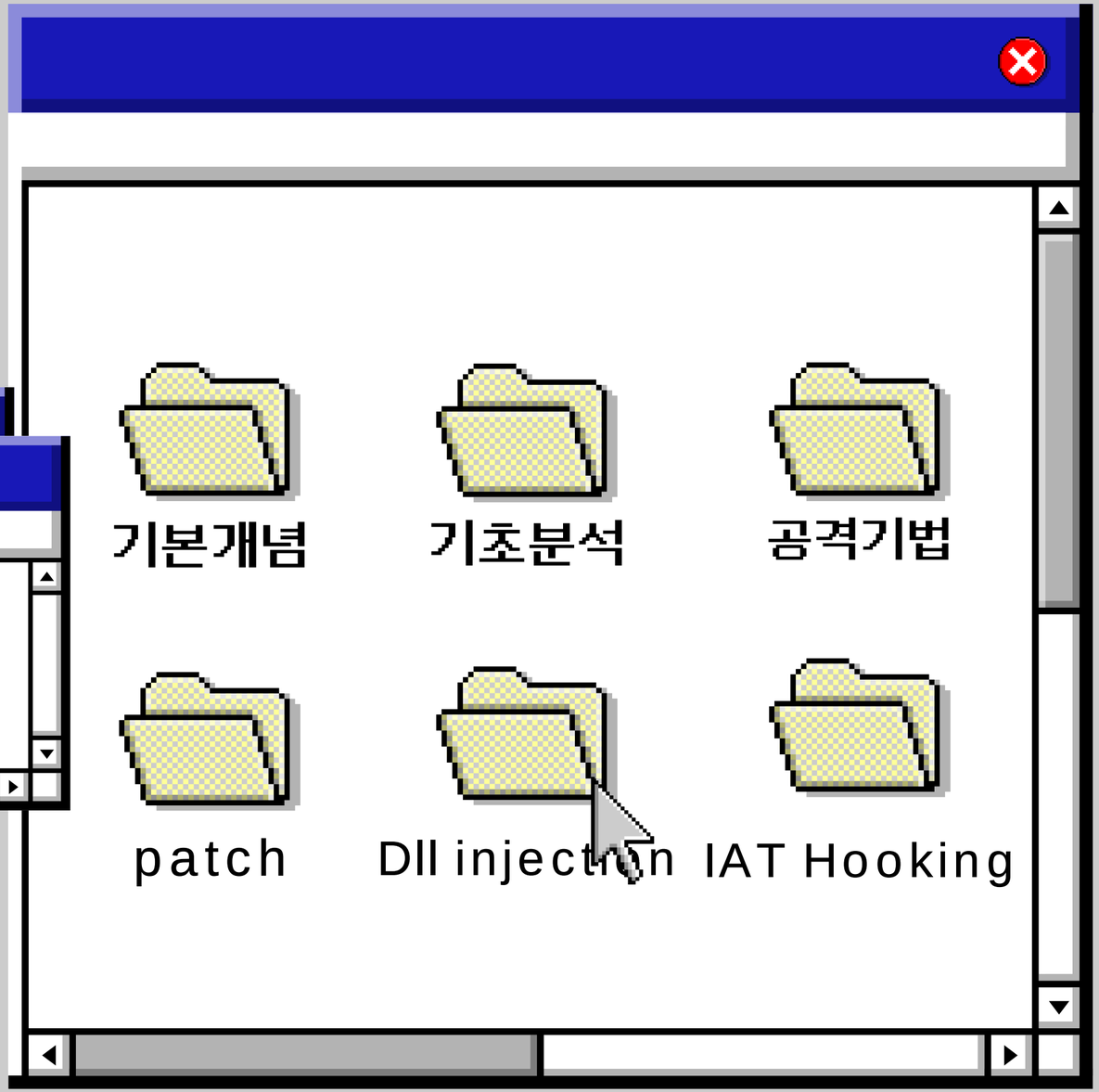
이제는 어디를 눌러도
지뢰에 깃발이 세워짐!

활용

같은 방법으로

2646 위치에 347C 패치

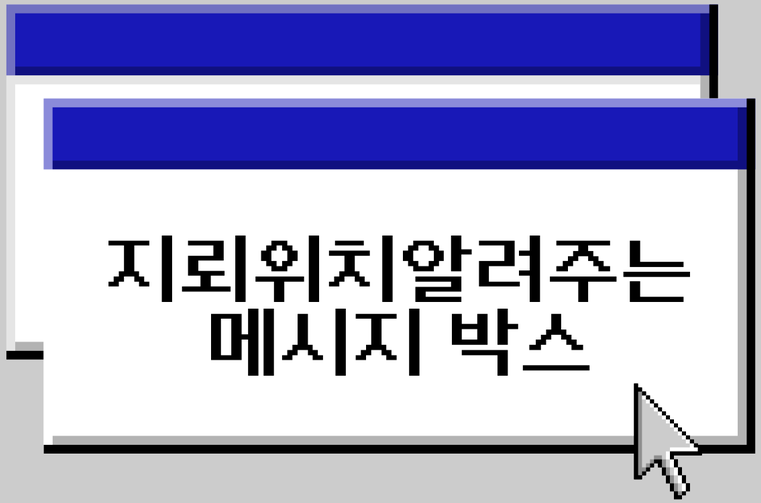
목차

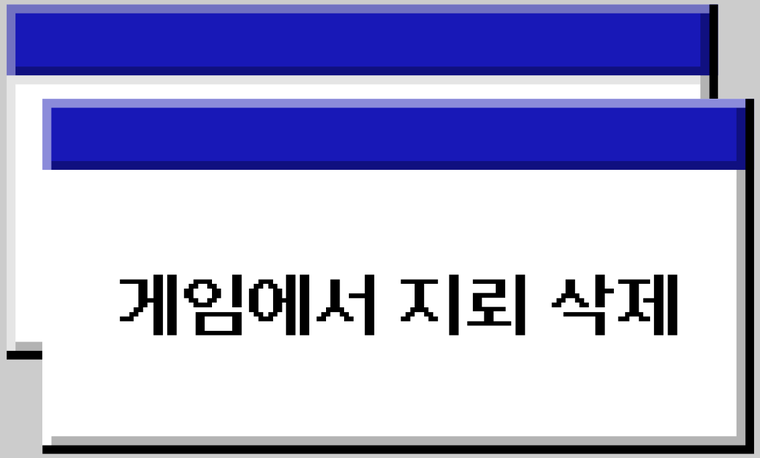


</>코드 및 결과

DLL injection

헤더와 소스코드 설명


지뢰위치알려주는
메시지 박스


게임에서 지뢰 삭제

헤더파일(코드)

```
// 원하는 맵 크기로 수정하세요!
#define MAP_WIDTH  32
#define MAP_HEIGHT 18

DWORD WINAPI HotkeyThread(LPVOID lpParam);
void ShowMines();
```

Back to Agenda Page

소스파일(코드)

변수명
:GetCellSymbol()

1.지뢰는 '★' 표시
2.벽은 '■' 표시
3.기본칸 '□' 표시
4.깃발 '♣' 표시
5.나머지 칸 '◇' 표시

```
// 원하는 값에 따라 문자 지정
wchar_t GetCellSymbol(BYTE val) {
    if (val == MINE_VALUE)      return L'★';
    else if (val == 0x10)       return L'■';
    else if (val == 0x0F)       return L'□';
    else if (val == 0x0E)       return L'♣';
    else                        return L'◇';
}
```

Back to Agenda Page

소스파일(코드)

- y: 세로(행), x: 가로(열)
- mineMap[y * 32 + x]: y행 x열의 실제 메모리 주소에서 값 읽기
- 읽은 값을 GetCellSymbol()로 변환해서 문자열에 추가
- 각 칸마다 공백 추가
- 한 줄이 끝나면 줄바꿈 추가

```
for (int y = 0; y < MAP_HEIGHT; ++y) {
    for (int x = 0; x < MAP_WIDTH; ++x) {
        mines += GetCellSymbol(mineMap[y * 32 + x]);
        mines += L' ';
    }
    mines += L"\r\n";
}
```

Back to Agenda Page

소스파일(코드)

- 무한 반복문으로 계속 실행
- GetAsyncKeyState(VK_F3) & 0x8000: F3키가 눌렸는지 체크
  - F3키가 눌리면 ShowMines() 실행
  - 0.5초(500ms) 대기(중복 입력 방지)
- 0.05초(50ms)마다 반복

```
DWORD WINAPI HotkeyThread(LPVOID lpParam)
{
    while (true) {
        if (GetAsyncKeyState(VK_F3) & 0x8000) {
            ShowMines();
            Sleep(500); // 중복 입력 방지
        }
        Sleep(50);
    }
    return 0;
}
```

Back to Agenda Page

소스파일(코드)

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH) {
        CreateThread(NULL, 0, HotkeyThread, NULL, 0, NULL);
    }
    return TRUE;
}
```

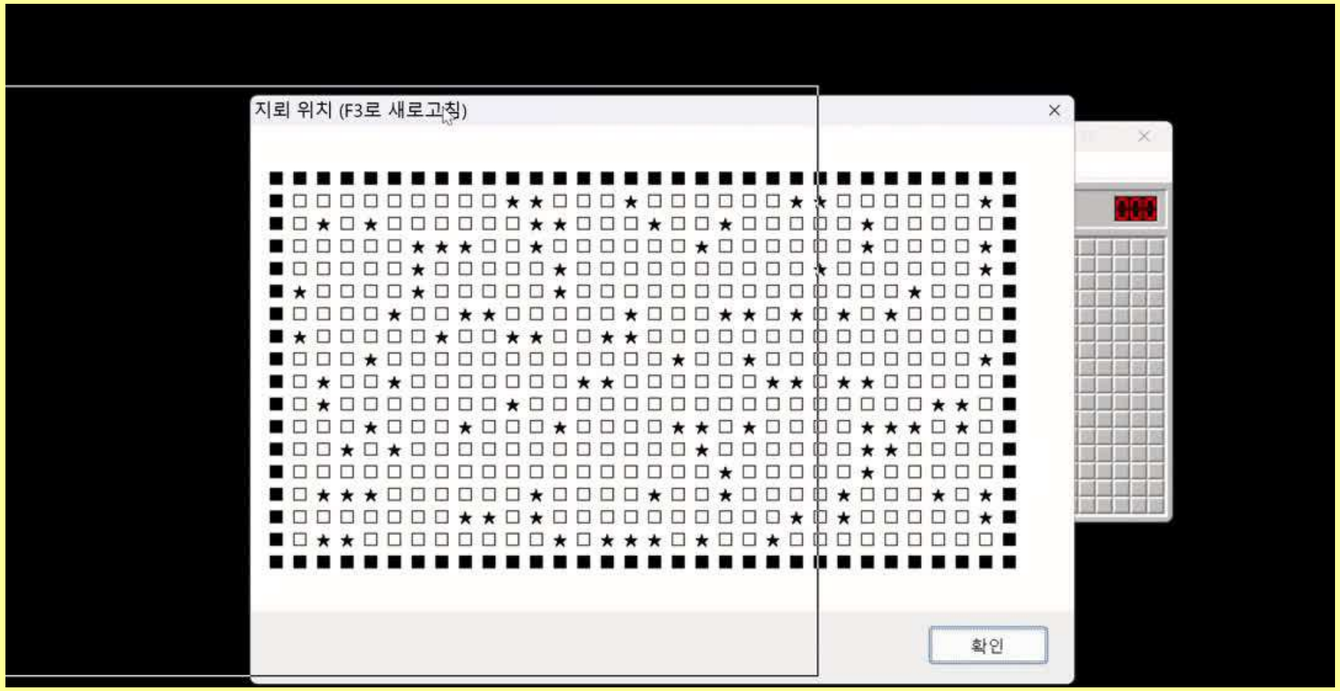
Dll main 과 핫키감지 함수 분류

- DLL이 처음 삽입될 때
- 별도의 스레드를 만들어
- 단축키 감지 등 반복 작업을 처리하게 하고
- DllMain 함수는 빠르게 종료하는 구조입니다

.

Back to Agenda Page

결과



Back to Agenda Page

지뢰위치알려주는
메시지 박스

게임에서 지뢰 삭제

헤더파일(코드)

```
#pragma once
#include <windows.h>

// 원하는 맵 크기로 수정하세요!
#define MAP_WIDTH  32
#define MAP_HEIGHT 18

DWORD WINAPI HotkeyThread(LPVOID lpParam);
void RemoveAllMines();
```

Back to Agenda Page

소스파일(코드)

- y: 세로(행), x: 가로(열)
- mineMap[y * 32 + x]: y행 x열의 실제 메모리 주소에서 값 읽기
- 읽은 값을 RemoveAllMines()로 변환해서 문자열에 추가
- 각 칸마다 공백 추가
- 한 줄이 끝나면 줄바꿈 추가

```
// 모든 지뢰를 빈 칸으로 변경
void RemoveAllMines()
{
    BYTE* mineMap = (BYTE*)MINE_MAP_ADDRESS;
    for (int y = 0; y < MAP_HEIGHT; ++y) {
        for (int x = 0; x < MAP_WIDTH; ++x) {
            if (mineMap[y * 32 + x] == MINE_VALUE) {
                mineMap[y * 32 + x] = EMPTY_VALUE;
            }
        }
    }
    MessageBoxW(NULL, L"모든 지뢰가 사라졌습니다!", L"지뢰 제거", MB_OK);
}
```

Back to Agenda Page

소스파일(코드)

- y: 세로(행), x: 가로(열)
- mineMap[y * 32 + x]: y행 x열의 실제 메모리 주소에서 값 읽기
- MINE_VALUE(지뢰칸)를 EMPTY_VALUE(빈칸)로
- 바꿈

```
// 모든 지뢰를 빈 칸으로 변경
void RemoveAllMines()
{
    BYTE* mineMap = (BYTE*)MINE_MAP_ADDRESS;
    for (int y = 0; y < MAP_HEIGHT; ++y) {
        for (int x = 0; x < MAP_WIDTH; ++x) {
            if (mineMap[y * 32 + x] == MINE_VALUE) {
                mineMap[y * 32 + x] = EMPTY_VALUE;
            }
        }
    }
    MessageBoxW(NULL, L"모든 지뢰가 사라졌습니다!", L"지뢰 제거", MB_OK);
}
```

Back to Agenda Page

소스파일(코드)

- 무한 반복문으로 계속 실행
- GetAsyncKeyState(VK_F3) & 0x8000: F3키가 눌렸는지 체크
  - F3키가 눌리면 RemoveAllMines() 실행
  - 0.5초(500ms) 대기(중복 입력 방지)
- 0.05초(50ms)마다 반복

```
// F3키 감지 스레드
DWORD WINAPI HotkeyThread(LPVOID lpParam)
{
    while (true) {
        if (GetAsyncKeyState(VK_F3) & 0x8000) { // F3키 감지
            RemoveAllMines();
            Sleep(500); // 중복 입력 방지
        }
        Sleep(50);
    }
    return 0;
}
```

Back to Agenda Page

소스파일(코드)

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH) {
        CreateThread(NULL, 0, HotkeyThread, NULL, 0, NULL);
    }
    return TRUE;
}
```

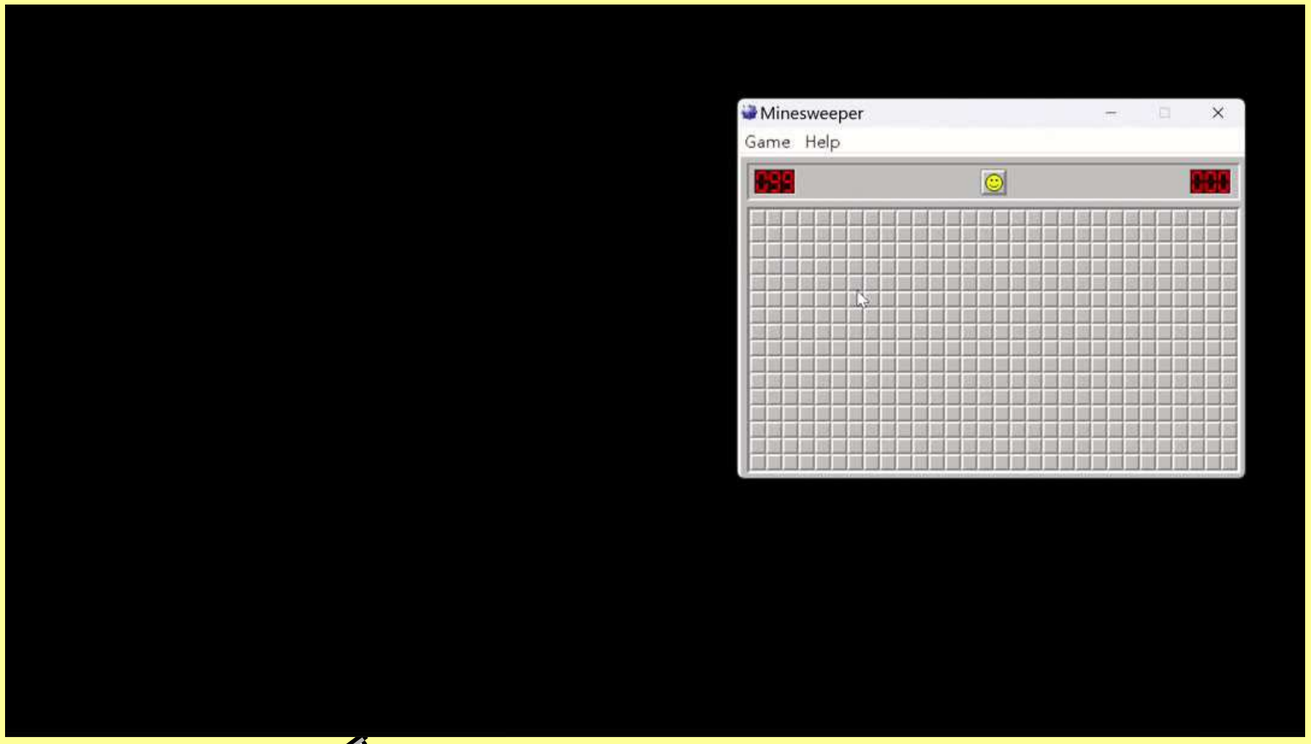
Dll main 과 핫키감지 함수 분류

- DLL이 처음 삽입될 때
- 별도의 스레드를 만들어
- 단축키 감지 등 반복 작업을 처리하게 하고
- DllMain 함수는 빠르게 종료하는 구조입니다

.

Back to Agenda Page

결과



[Back to Agenda Page]

목차

API 선택

공격할 API를 선정한다.

코드분석

코드를 분석한다.

실행결과

실행결과를 확인한다.

API 선택

SetCapture

HotkeyThread

Back to Agenda Page

API 선택

SetCapture

SetCapture는 마우스에 반응을 감지하는 마이크로소프트 API

Back to Agenda Page

API 선택

HotkeyThread

HotkeyThread는 특정 키보드에 반응하는 마이크로소프트 API

Back to Agenda Page

API 선택

공격할 API를 선정한다.

코드분석

코드를 분석한다.

실행결과

실행결과를 확인한다.

헤더파일(코드)

window API를 사용하기 때문에
window.h를 가져와준다.

```
#pragma once
#include <windows.h>

#ifdef __cplusplus
extern "C" {
#endif

    // 핫키 감지 쓰레드
    DWORD WINAPI HotkeyThread(LPVOID lpParam);

    // IAT 후킹 트리거 함수
    void TriggerIATHook(void);

    // 후킹 대상 함수
    DWORD WINAPI ShowMessage(LPVOID lParam);

#ifdef __cplusplus
}
#endif
```

Back to Agenda Page

헤더파일(코드)

C++에서 컴파일할 때, 함수 이름이 mangling으로 변형되는데, extern "C"는 이를 방지하고 C 스타일 함수명으로 유지시켜 준다

.

```
#pragma once
#include <windows.h>

#ifdef __cplusplus
extern "C" {
#endif

    // 핫키 감지 쓰레드
    DWORD WINAPI HotkeyThread(LPVOID lpParam);

    // IAT 후킹 트리거 함수
    void TriggerIATHook(void);

    // 후킹 대상 함수
    DWORD WINAPI ShowMessage(LPVOID lParam);

#ifdef __cplusplus
}
#endif
```

Back to Agenda Page

IATHooking 트래거(코드1)

PE(실행 파일) 구조를 따라가서, IAT(Import Address Table)를 읽기 위해 주소 계산하는 코드다

```
void TriggerIATHook(void)
{
    if (g_hooked) return;
    g_hooked = true;

    HMODULE hMod = GetModuleHandle(NULL);
    if (!hMod) return;

    PBYTE base = (PBYTE)hMod;
    DWORD OldProtect = 0;

    PIMAGE_DOS_HEADER idh = (PIMAGE_DOS_HEADER)base;
    PIMAGE_OPTIONAL_HEADER ioh = (PIMAGE_OPTIONAL_HEADER)
      (base + idh->e_lfanew + 24);
    PIMAGE_IMPORT_DESCRIPTOR iid = (PIMAGE_IMPORT_DESCRIPTOR)
      (base + ioh-  >DataDirectory[IMAGE_DIRECTORY_ENTRY_IMPORT].VirtualAddress);
```

Back to Agenda Page

IATHooking 트래커(코드2)

setcapture를 user32.dll에 import directory table에서 찾는 코드이다.

```
    for (; iid->Name; iid++) {
        PCSTR dllName = (PCSTR)(base + iid->Name);
        if (_stricmp(dllName, "USER32.dll") != 0) continue;

        MessageBox(NULL, TEXT("Found USER32.dll in Import Directory Table"), TEXT("GUnT_0x9"), MB_OK);

        PIMAGE_THUNK_DATA32 addrTable = (PIMAGE_THUNK_DATA32)(base + iid->FirstThunk);
        PIMAGE_THUNK_DATA32 nameTable = (PIMAGE_THUNK_DATA32)(base + iid->OriginalFirstThunk);

        for (; addrTable->u1.Function; addrTable++, nameTable++) {
            PIMAGE_IMPORT_BY_NAME iibn = (PIMAGE_IMPORT_BY_NAME)(base + nameTable->u1.AddressOfData);

            if (_stricmp((char*)iibn->Name, "SetCapture") == 0) {
                MessageBox(NULL, TEXT("Found SetCapture IAT Address"), TEXT("GUnT_0x9"), MB_OK);

                VirtualProtect(&addrTable->u1.Function, sizeof(DWORD), PAGE_READWRITE, &OldProtect);
                addrTable->u1.Function = (DWORD)(ULONG_PTR)ShowMessage;
                VirtualProtect(&addrTable->u1.Function, sizeof(DWORD), OldProtect, &OldProtect);
                return;
            }
        }
    }
}
```

Back to Agenda Page

IATHooking 트래커(코드2)

찾으면 showMessage에서 메세지 가져옴

```
    for (; iid->Name; iid++) {
        PCSTR dllName = (PCSTR)(base + iid->Name);
        if (_stricmp(dllName, "USER32.dll") != 0) continue;

        MessageBox(NULL, TEXT("Found USER32.dll in Import Directory Table"), TEXT("GUnT_0x9"), MB_OK);

        PIMAGE_THUNK_DATA32 addrTable = (PIMAGE_THUNK_DATA32)(base + iid->FirstThunk);
        PIMAGE_THUNK_DATA32 nameTable = (PIMAGE_THUNK_DATA32)(base + iid->OriginalFirstThunk);

        for (; addrTable->u1.Function; addrTable++, nameTable++) {
            PIMAGE_IMPORT_BY_NAME iibn = (PIMAGE_IMPORT_BY_NAME)(base + nameTable->u1.AddressOfData);

            if (_stricmp((char*)iibn->Name, "SetCapture") == 0) {
                MessageBox(NULL, TEXT("Found SetCapture IAT Address"), TEXT("GUnT_0x9"), MB_OK);

                VirtualProtect(&addrTable->u1.Function, sizeof(DWORD), PAGE_READWRITE, &OldProtect);
                addrTable->u1.Function = (DWORD)(ULONG_PTR)ShowMessage;
                VirtualProtect(&addrTable->u1.Function, sizeof(DWORD), OldProtect, &OldProtect);
                return;
            }
        }
    }
}
```

Back to Agenda Page

Hotkey 트래거

GetAsyncKeyState(VK_F2)로
F2를 사용하면 발동되도록 설정함

```
DWORD WINAPI HotkeyThread(LPVOID lpParam)
{
    while (true) {
        if (GetAsyncKeyState(VK_F2) & 0x8000) {
            TriggerIATHook();
            Sleep(500); // 중복 입력 방지
        }
        Sleep(50);
    }
    return 0;
}
```

Back to Agenda Page

Hotkey 트래거

F2가 발동하면 IAT후킹 트래거로
넘어가는걸 볼수있다

```
DWORD WINAPI HotkeyThread(LPVOID lpParam)
{
    while (true) {
        if (GetAsyncKeyState(VK_F2) & 0x8000) {
            TriggerIATHook();
            Sleep(500); // 중복 입력 방지
        }
        Sleep(50);
    }
    return 0;
}
```

Back to Agenda Page

dll메인

dll을 로드할때
핫키 감지용 스레드 를생성한다
.

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH) {
        DisableThreadLibraryCalls(hinstDLL);
        CreateThread(NULL, 0, HotkeyThread, NULL, 0, NULL);
    }
    return TRUE;
}
```

Back to Agenda Page

dll메인

사용자가 F2를 누르면
후킹이 진행되도록 만들었다

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
{
    if (fdwReason == DLL_PROCESS_ATTACH) {
        DisableThreadLibraryCalls(hinstDLL);
        CreateThread(NULL, 0, HotkeyThread, NULL, 0, NULL);
    }
    return TRUE;
}
```

Back to Agenda Page

API 선택

공격할 API를 선정한다.

코드분석

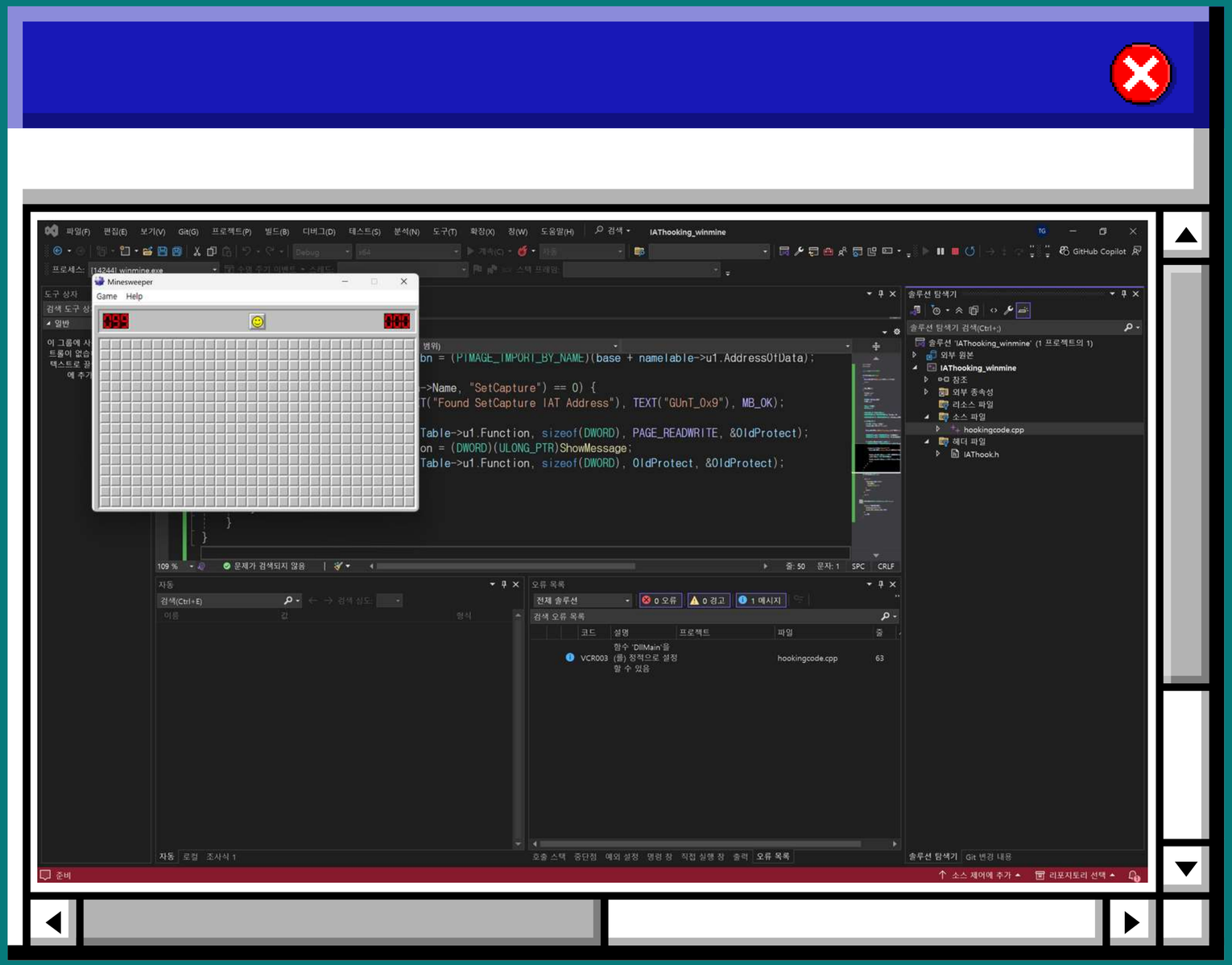
코드를 분석한다.

실행결과

실행결과를 확인한다.

실행하기

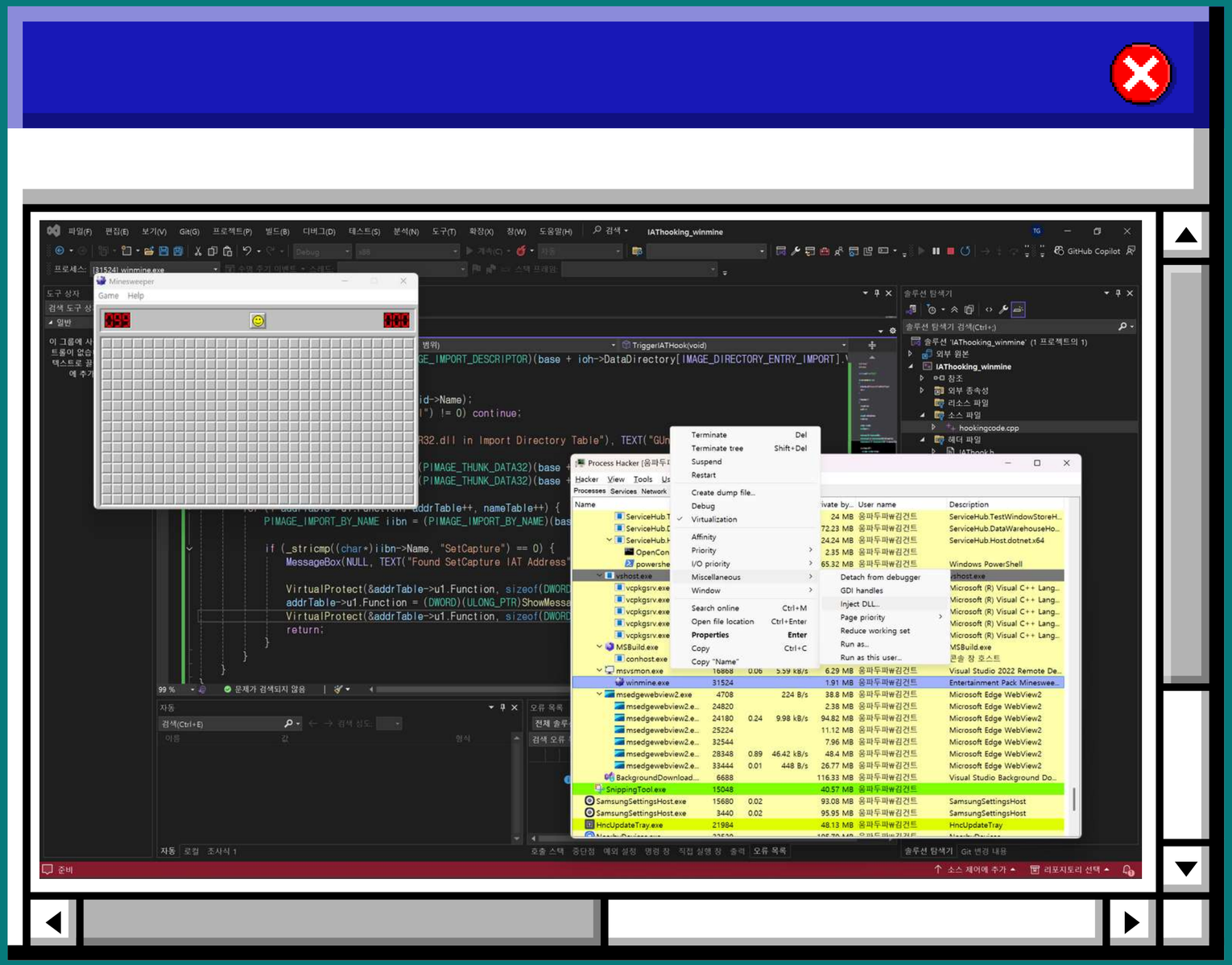
실행하면 지뢰찾기가 실행된다.
하지만 아직 dll을 삽입하지
않았기 때문에 후킹이
진행되지는 않는다.



Back to Agenda Page

실행하기

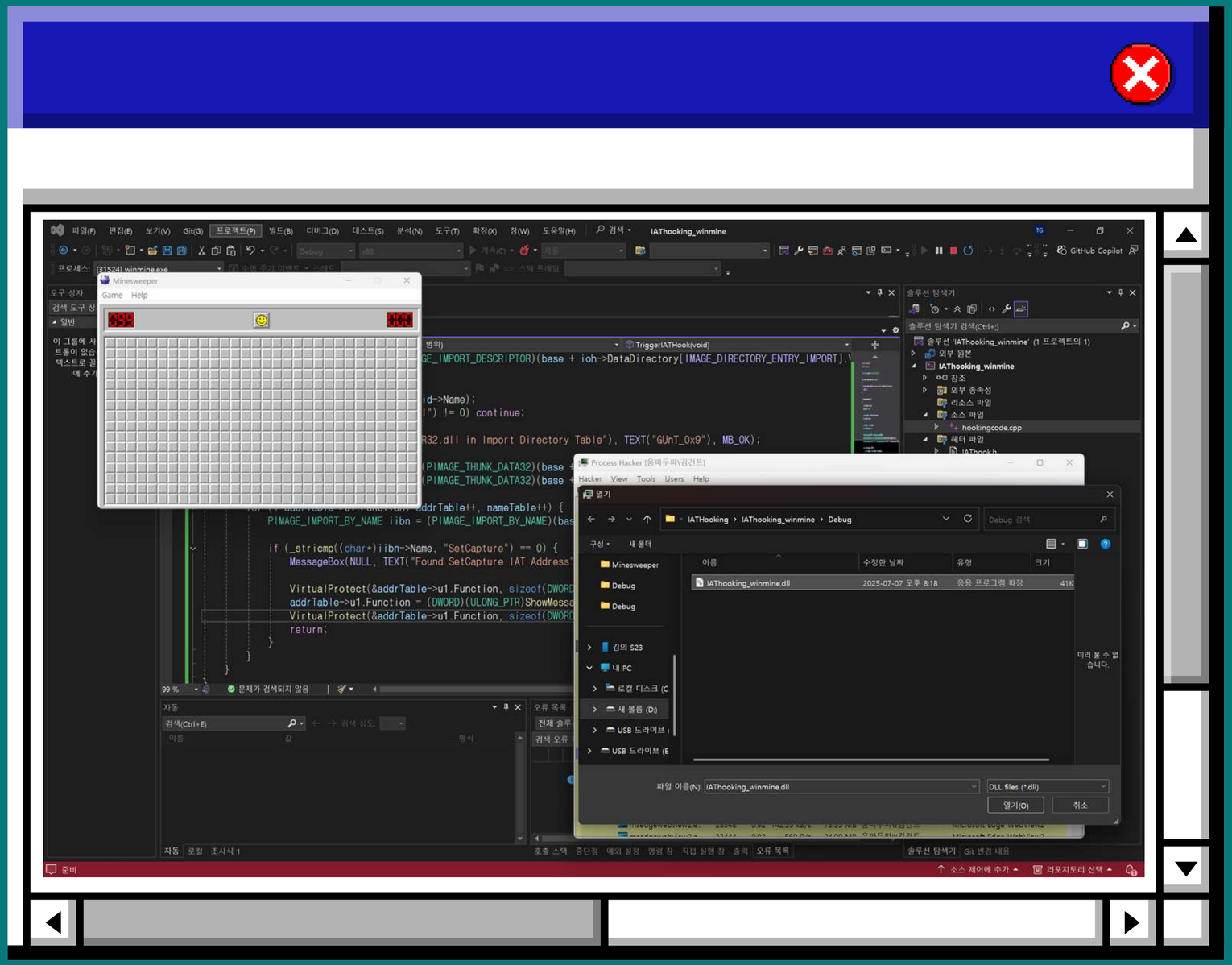
ProcessHacker.exe에서
Miscellaneous에서
inject dll를 들어가준다



Back to Agenda Page

실행하기

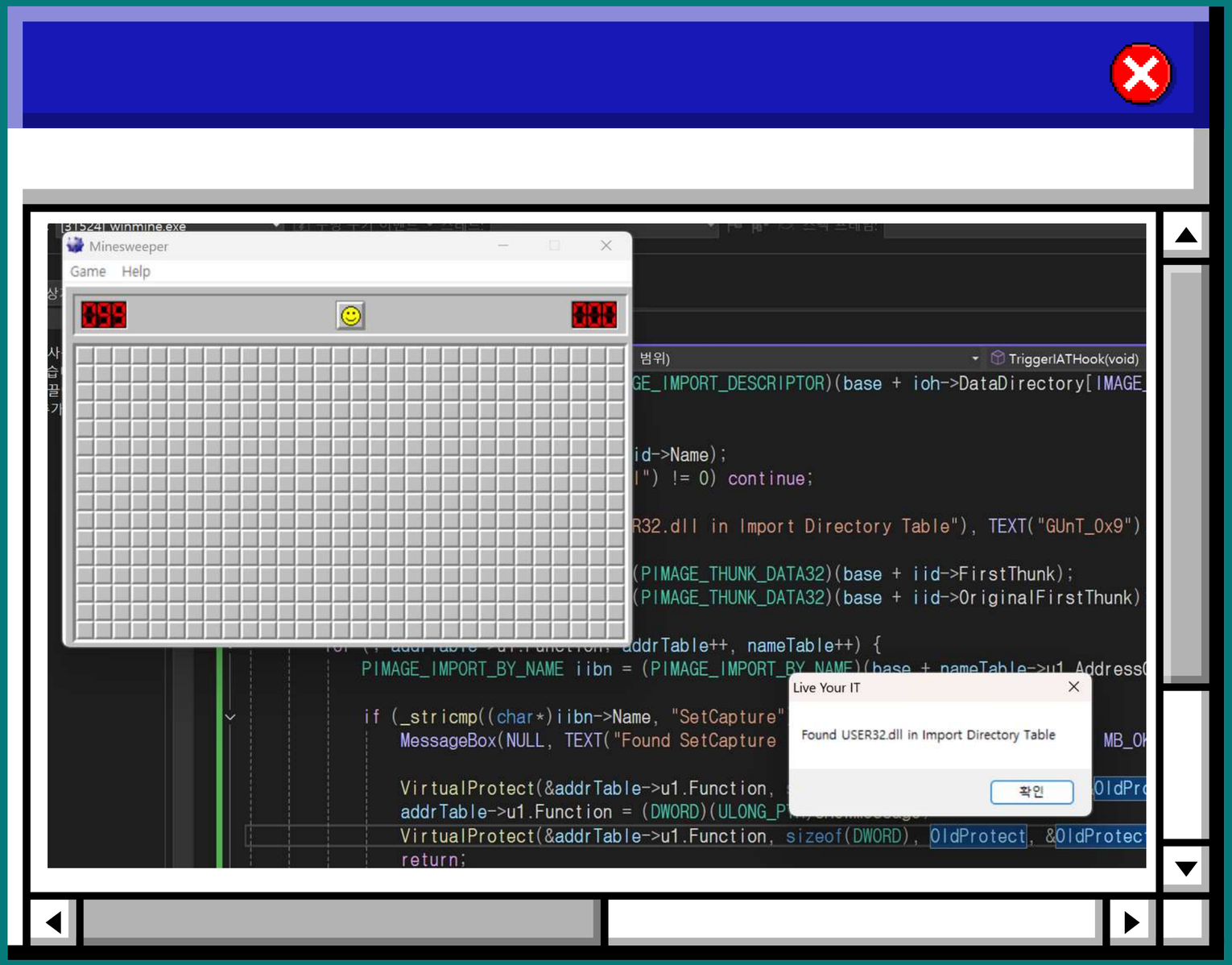
ProcessHacker.exe로
직접 inject dll로 dll을 삽입함



Back to Agenda Page

실행하기

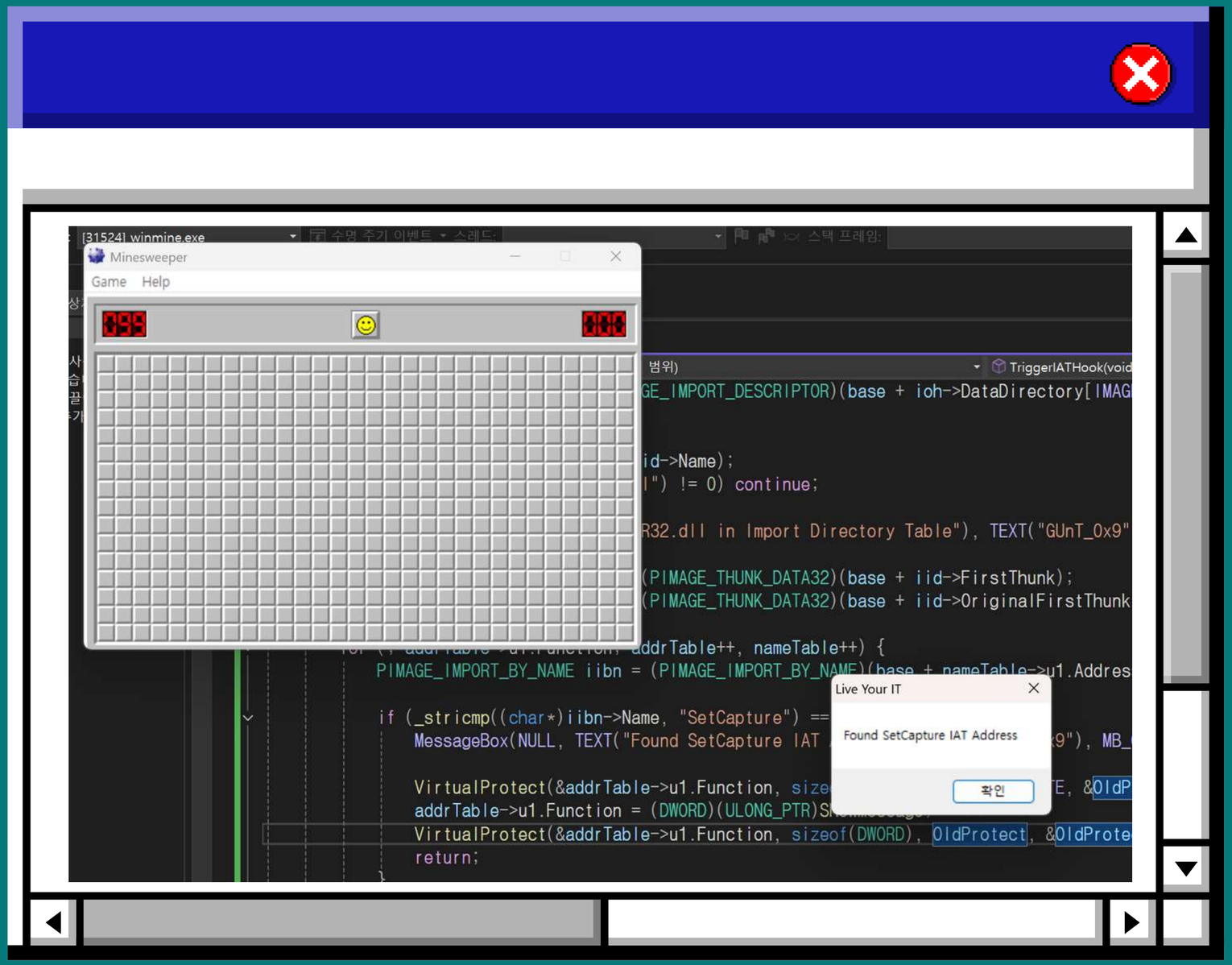
F2누르면 바로
import Directory Table를
찾는다는 메세지가 뜬다.



Back to Agenda Page

실행하기

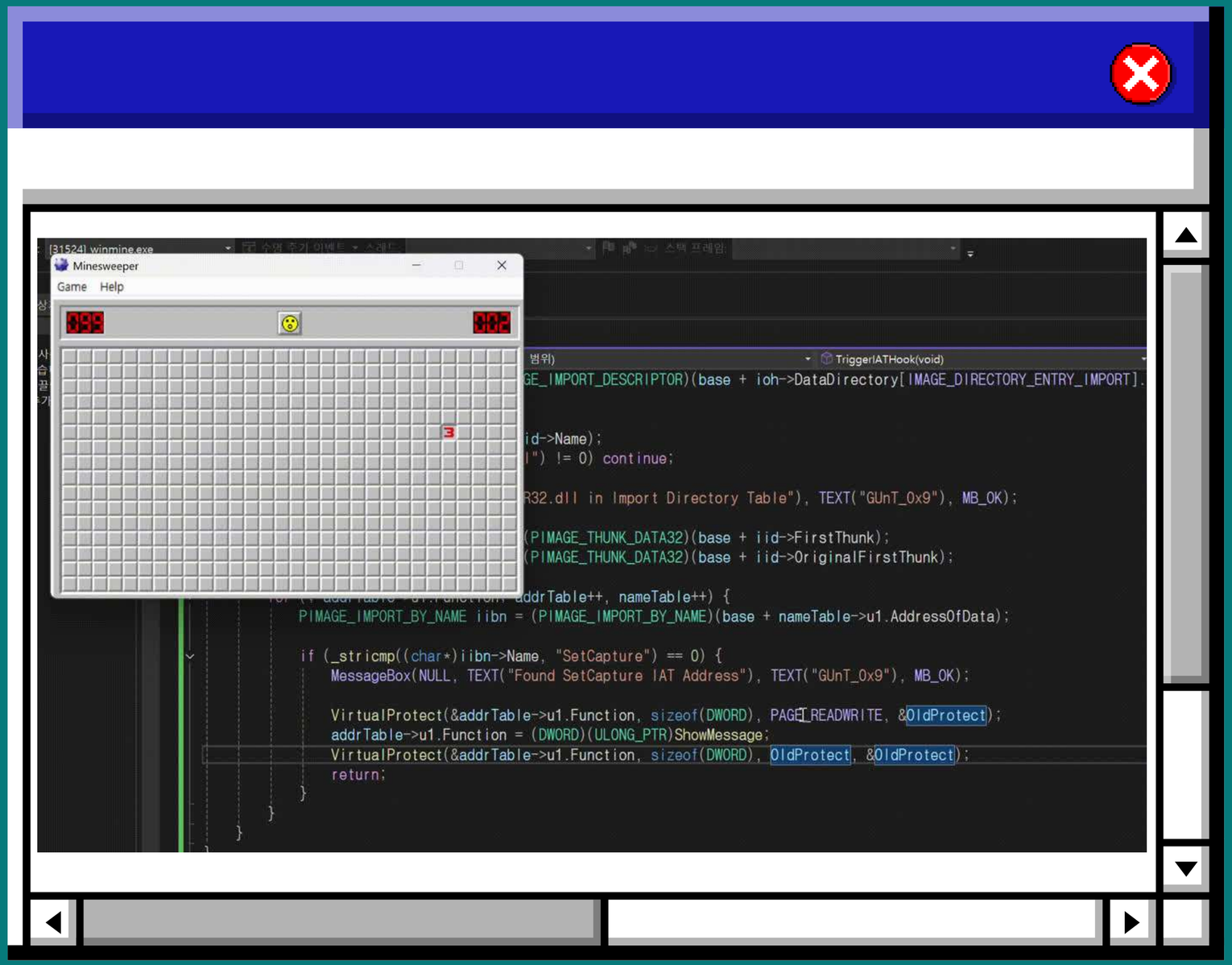
Table을 찾으면 바로 SetCapture를 찾는다는 메세지가 뜬다.



Back to Agenda Page

실행하기

SetCapture를 찾으면
지뢰찾기에서 어떤 클릭을 해도
메세지박스 내용이 뜨게 된다



Back to Agenda Page

배우게 된것 AND 더알고 싶은 점

GUnT_0x9 | ugh09 | Mas$y_J!

05:23 PM

Patch

이번 세미나를 준비하면서 처음사용하는 IDA의 기능을 익히게 되었고, 처음으로 패치를 진행하여 작동하는 핵이 만들어 진
것에 뿌듯함을 느꼈습니다.

Back to Agenda Page

Dll injection

배운점

Dll의 계념과 Dll injection

vscode2020 사용법

여러가지 환경설정 방법

더 알아가고 싶은점

dll injection을 활용한 다양한 해킹 방법

dll injection이외의 다른 공격기법

Back to Agenda Page

Hookin

인터넷에 지뢰찾기 hooking관련 정보가 없어
힘들었지만 공격을 성공해 매세지박스가
성공적으로 실행할때 보람을 느꼈다.

Back to Agenda Page

감사합니다

GUnT_0x9 | ogh09 | Mas$y_J!

05:23
PM

Q&A

GUnT_0x9 | ogh09 | Mas$y_J!

05:23
PM