

지뢰 찾기 분석

REV&PWN 17TH RUNA

홍 영우

목차

1. reverse engineering intro

2. 지뢰 찾기 (winemine.exe) 분석

정적 분석: Drawtime (x), DoTimer()

-> 시간 값을 참조하는 함수

draw Bomb count (x), update Bomb count(x)

동적 분석: count Bombs(arg1, arg2)

→ 지뢰개수를 참조하는 함수

reverse engineering이란?

역 공학: 완성된 제품(소프트웨어)로 부터 구조와 원리를 알아내는 기술

사용되는 어셈블리어

Push: 스택에 값을 넣는다(저장)

Pop: 스택에 저장된 값을 꺼내 레지스터 등에 저장한다.

Inc: 피연산자의 값을 1증가시킨다

사용되는 어셈블리어

Call: 함수/코드 등을 불러온다.

Jmp ,Jz, Jge,jnz → 특정 조건을 만족시키면 지정된 함수로 점프한다.

사용되는 용어

Ebx: 추가적인 저장소 역할을 하는 레지스터

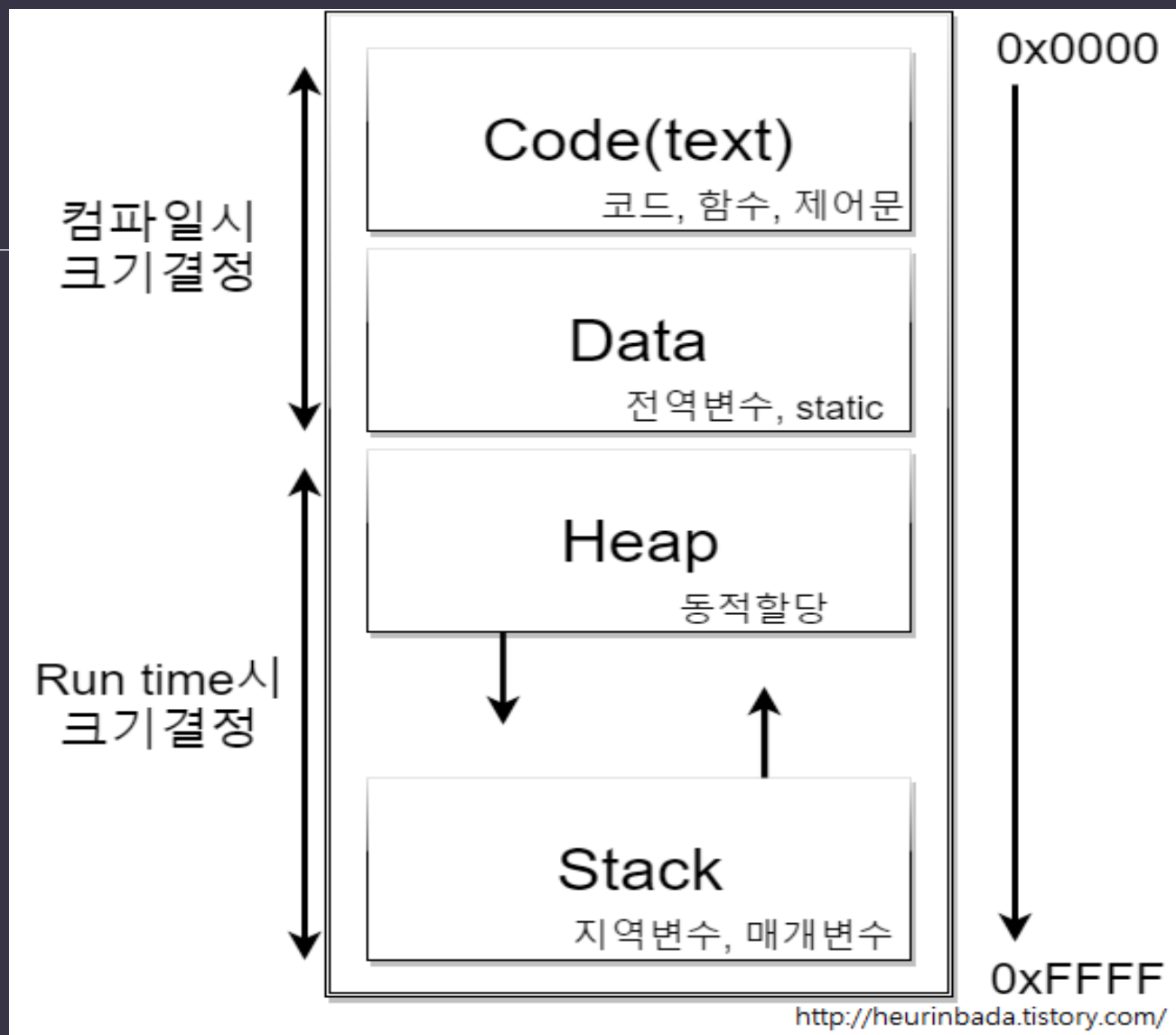
esi,edi: 파일 복사시 소스데이터/목적지 데이터의 주소를 저장

Esp: 스택 포인터, 스택의 가장 높은 곳을 가리킨다.

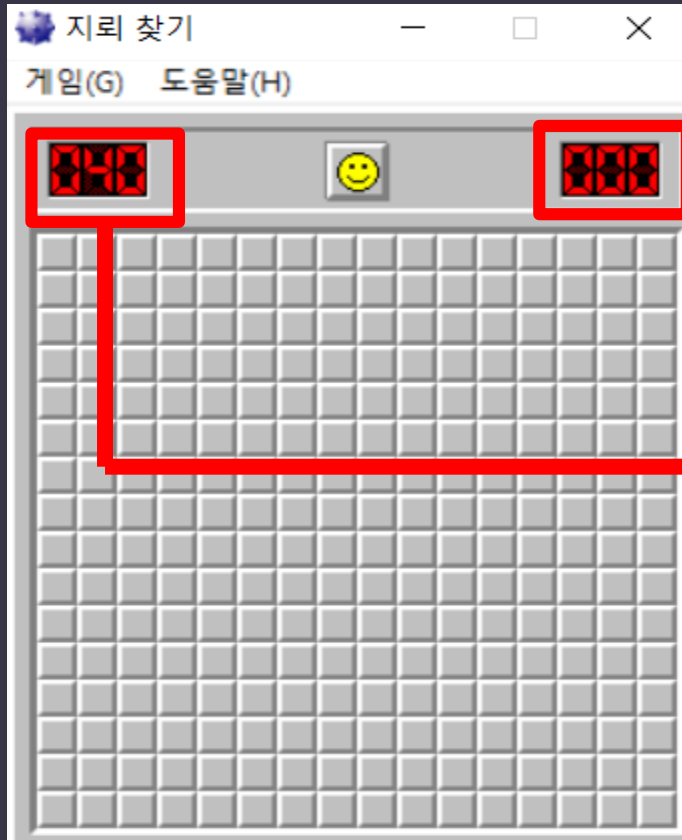
Ebp: 베이스 포인터, 스택의 가장 낮은 곳을 가리킨다.

메모리 구조

그림 참고



지뢰 찾기



시간 값

지뢰 개수 값

시간(data)값을 참조 하는 함수 분석

```
data:01005798 _stepmac dd 0 ; DATA XREF: stepXY(x,x)+501r
data:01005798 ; StepXY(x,x)+671w ...
data:0100579C cSec dd 0 ; DATA XREF: DrawTime(x)+81r
data:0100579C ; DoTimer()+91r ...
data:010057A0 _cBoxVisitMac dd 0 ; DATA XREF: StepSquare(x,x)+8F1r
```

dd -> double(실수)형 data로 시간 값을 저장

DrawTime 함수와 DoTimer함수 가 참조 중

```
; int __stdcall DrawTime(HDC hdc)
_DrawTime@4 proc near
```

```
hdc= dword ptr 4
```

```
push ebx
push ebp
push esi
mov esi, [esp+0Ch+hdc]
```

```
push edi
mov edi, _cSec
push esi ; hdc
```

```
call ds: __imp_GetLayout@4 ; GetLayout(x)
```

```
mov ebp, eax
mov ebx, ebp
and ebx, 1
jz short loc_100284C
```

```
push 0 ; 1
push esi ; hdc
call ds: __imp__SetLayout@8 ; SetLayout(x,x)
```

```
loc_100284C:
```

```
push 64h
mov eax, edi
cdq
pop ecx
idiv ecx
push eax ; int
mov eax, _dxWindow
sub eax, _dxpBorder
mov edi, edx
sub eax, 38h
push eax ; xDest
push esi ; hdc
call _DrawLed@12 ; DrawLed(x,x,x)
```

```
push 0
mov eax, edi
cdq
pop ecx
idiv ecx
push eax ; int
```

Draw time 함수

저장한 시간 값 -> draw LED 함수 전달.

<GUI 환경에 맞추어 시간 값을 보여준다>

Do Timer 함수

_cSec -> 카운터 하고 있는
시간값을 저장하는 변수

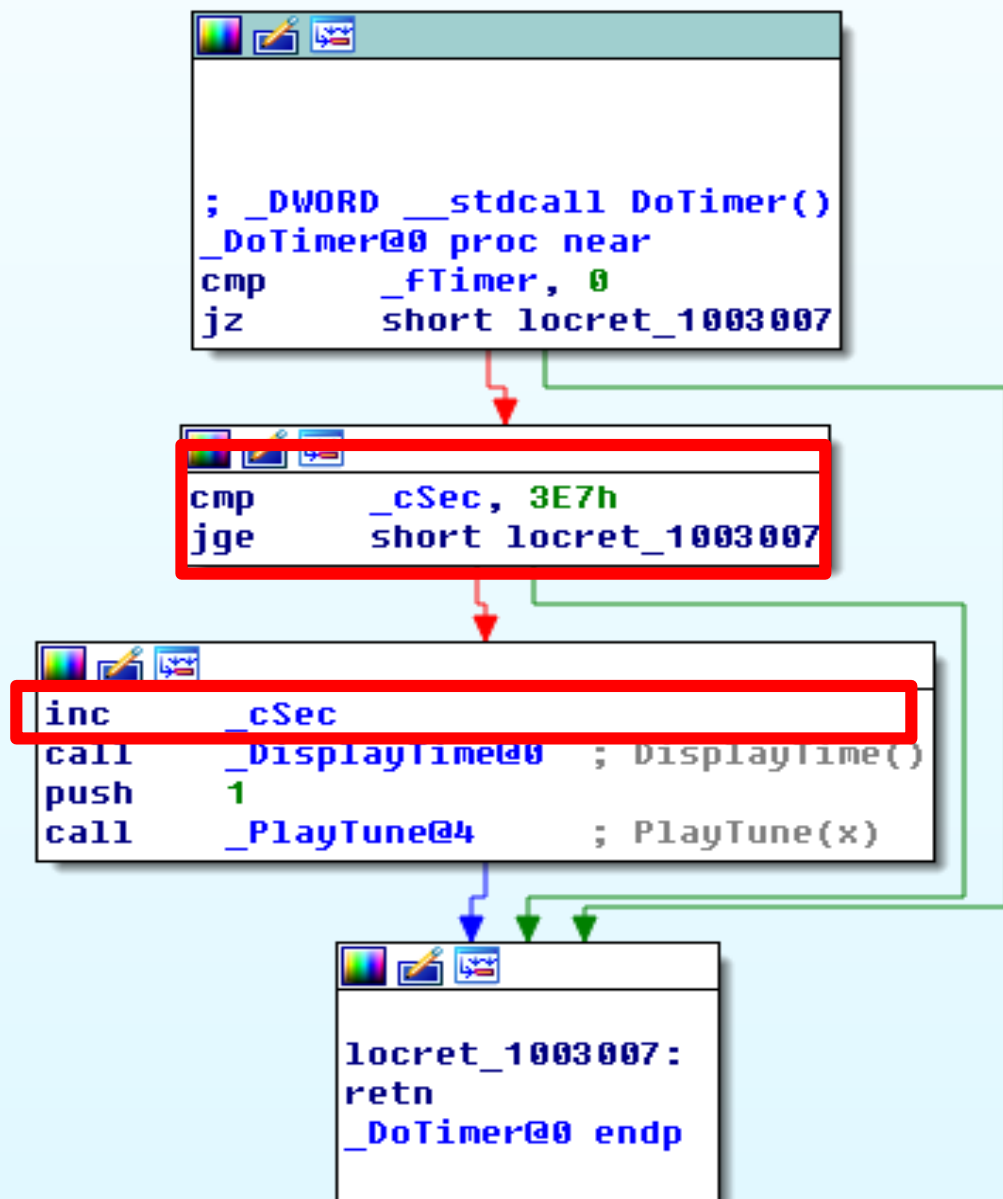
cmp -> 두 값을 비교 <분기점>

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 함수들 분석

```
.data:01005190                ; HtmlHelpA(x,x,x,x)+71w  
.data:01005194 cBombLeft      dd 0                ; DATA XREF: DrawBombCount(x):loc 10027AAir  
.data:01005194                ; UpdateBombCount(x)+41w ...  
.data:01005198                align 10h  
.data:010051A0 _rgStepX       dd 0                ; DATA XREF: StepXY(x,x)+551w  
.data:010051A0                ; StepBox(x,x)+291r
```

지뢰 값(data)을 참조하는 함수들을 분석

```
data:01005330 _cBombStart dd ? ; DATA XREF: StartGame()+481w
data:01005330 ; StartGame()+831w ...
data:01005334 _xBoxMac dd ? ; DATA XREF: AdjustWindow(x):loc_10019DB1r
data:01005334 ; MainWndProc(x,x,x,x)+5451r ...
data:01005338 _yBoxMac dd ? ; DATA XREF: AdjustWindow(x)+9B1r
data:01005338 ; MainWndProc(x,x,x,x)+5511r ...
data:0100533C align 10h
```

지뢰 값(data)을 참조하는 함수들을 분석

```
.data:010056A4 ; UINT dword_10056A4
```

```
.data:010056A4 dword_10056A4 dd ? ; DATA XREF: PrefDlgProc(x,x,x,x)+931w
```

```
.data:010056A4 ; PrefDlgProc(x,x,x,x)+D21r ...
```

```
.data:010056A8 ; UINT uValue
```

```
.data:010056A8 uValue dd ? ; DATA XREF: PrefDlgProc(x,x,x,x)+601w
```

```
.data:010056A8 ; PrefDlgProc(x,x,x,x)+6A1r ...
```

```
; int __stdcall DrawBombCount(HDC hdc)
_DrawBombCount@4 proc near
```

```
hdc= dword ptr 4
```

```
push    ebx
push    ebp
push    esi
mov     esi, [esp+0Ch+hdc]
push    edi
call    ds:imp_GetLayout@4 ; GetLayout(x)
mov     ebp, ds:imp_SetLayout@8 ; SetLayout(x,x)
mov     ebx, eax
mov     [esp+10h+hdc], ebx
and     ebx, 1
jz      short loc_10027AA
```

draw Bomb count

폭탄의 개수를 전광판에 표시하는 함수
레이아웃 함수를 호출

```
push    0 ; 1
push    esi ; hdc
call    ebp ; SetLayout(x,x) ; SetLayout(x,x)
```

```
loc_10027AA:
mov     eax, _cBombLeft
test    eax, eax
jge     short loc_10027C0
```

```
push    0Bh
pop     ecx
neg     eax
push    64h
cdq
pop     edi
idiv    edi
jmp     short loc_10027C8
```

```
loc_10027C0:
push    64h
cdq
pop     ecx
idiv    ecx
mov     ecx, eax
```

draw Bomb count

```
push    0Bh
pop     ecx
neg     eax
push    64h
cdq
pop     edi
idiv    edi
jmp     short loc_10027C8
```

```
loc_10027C0:
push    64h
cdq
pop     ecx
idiv    ecx
mov     ecx, eax
```

```
loc_10027C8:                ; int
push    ecx
push    11h                 ; xDest
push    esi                 ; hdc
mov     edi, edi
call    _DrawLed@12         ; DrawLed(x,x,x)
mov     eax, edi
pop     ecx
cdq
idiv    ecx
push    eax                 ; int
push    1Eh                 ; xDest
push    esi                 ; hdc
mov     edi, eax
call    _DrawLed@12         ; DrawLed(x,x,x)
push    edi                 ; int
push    2Bh                 ; xDest
push    esi                 ; hdc
call    _DrawLed@12         ; DrawLed(x,x,x)
test    ebx, ebx
jz      short loc_10027FA
```

```
push    [esp+10h+hdc]       ; 1
push    esi                 ; hdc
call    ebp ; SetLayout(x,x) ; SetLayout(x,x)
```


draw Bomb count

```
push    [esp+10h+hdc]    ; 1  
push    esi               : hdc  
call    ebp ; SetLayout(x,x) ; SetLayout(x,x)
```

```
loc_10027FA:  
pop     edi  
pop     esi  
pop     ebp  
pop     ebx  
retn    4  
_DrawBombCount@4 endp
```

Count Bombs (동적 분석)

01002F38	\$ 8B4C24 08	MOV ECX,DWORD PTR SS:[ARG.2]	winmine.01002F38(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	• 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	• 8B06	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	• EB 08	JMP SHORT 01002F70	
01002F68	> F60432 80	TEST BYTE PTR DS:[ESI+EDX],80	
01002F6C	• 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 0000	RETN 8	
01002F80	\$ A1 38530001	MOV EAX,DWORD PTR DS:[1005338]	winmine.01002F80(guessed Arg1)
01002F85	• 83F8 01	CMP EAX,1	
01002F88	• 7C 4E	JL SHORT 01002FD8	
01002F8A	• 53	PUSH EBX	

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	✓ 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	✓ EB 08	JMP SHORT 01002F70	
01002F68	> F60432 80	TEST BYTE PTR DS:[ESI+EDX],80	
01002F6C	✓ 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	✓ 7C 4E	JL SHORT 01002FD8	
01002F8A	. 53	PUSH EBX	

01002F38	\$ 8B4C24 08	MOV ECX,DWORD PTR SS:[ARG.2]	winmine.01002F38(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	• 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	• 8B06	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	• EB 08	JMP SHORT 01002F70	
01002F68	> F60432 80	TEST BYTE PTR DS:[ESI+EDX],80	
01002F6C	• 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 0000	RETN 8	
01002F80	\$ A1 38530001	MOV EAX,DWORD PTR DS:[1005338]	winmine.01002F80(guessed Arg1)
01002F85	• 83F8 01	CMP EAX,1	
01002F88	• 7C 4E	JL SHORT 01002FD8	
01002F8A	• 53	PUSH EBX	

Count Bombs

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

count Bombs

영상 참조

update Bomb count

```
; stdcall UpdateBombCount(x)
UpdateBombCount@4 proc near

arg_0= dword ptr 4

mov     eax, [esp+arg_0]
add     _cBombLeft, eax
call    _DisplayBombCount@0 ; DisplayBombCount()
retn    4
_UpdateBombCount@4 endp
```

_cBomb Left 에서 지뢰
값을 전달 받아
displayBombCount 함수에
전달 한다.

arg_0 : count bomb 에서
지뢰의 개수 값으로 사용하던
인수

발견했던 취약한 부분

변수 값들이 평문으로 저장 되어있어 쉽게 검색됨

동적 분석 시 매번 같은 주소에 데이터나 코드가 저장되어
분석이 용이함

보안 방법

1.xor암호화

변수의 연산/암호화를 통해 검색 /값 노출 방지

2. ASLR

공격자가 대상 주소를 예측하기 어렵게 만들어 메모리상의 공격/분석을 어렵게 합니다.

Thanks you!

A solid purple bar at the bottom of the slide, transitioning from a lighter shade on the left to a darker shade on the right.