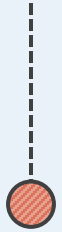
# 어셈블리어

abex' crackme + 스택 프레임

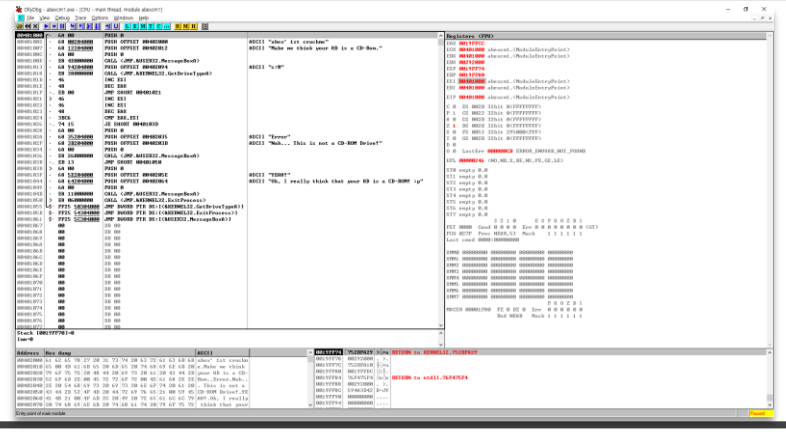


**210126 이유경**

# 어셈블리어

abex' crackme + 스택 프레임

## 01



### abex' crackme

분석 및 레지스터 정리

## 02

스택 프레임이란?

: ESP가 아닌 EBP 레지스터를 사용하여 스택 내의 로컬 변수, 파라미터, 복귀 주소에 접근하는 기법

스택 프레임 생성

```
PUSH    EBP          ⇒ 기존의 EBP값을 스택에 집어 넣어라
MOV     EBP, ESP     ⇒ ESP의 값을 EBP로 옮겨라
...
```

스택 프레임 해제

```
MOV     ESP, EBP     ⇒ ESP 값을 원래대로 복원시켜라
POP     EBP          ⇒ 스택에 백업한 EBP 값을 복원시켜라
RETN                 ⇒ 스택에 저장된 복귀 주소로 리턴하여 함수 종료
```

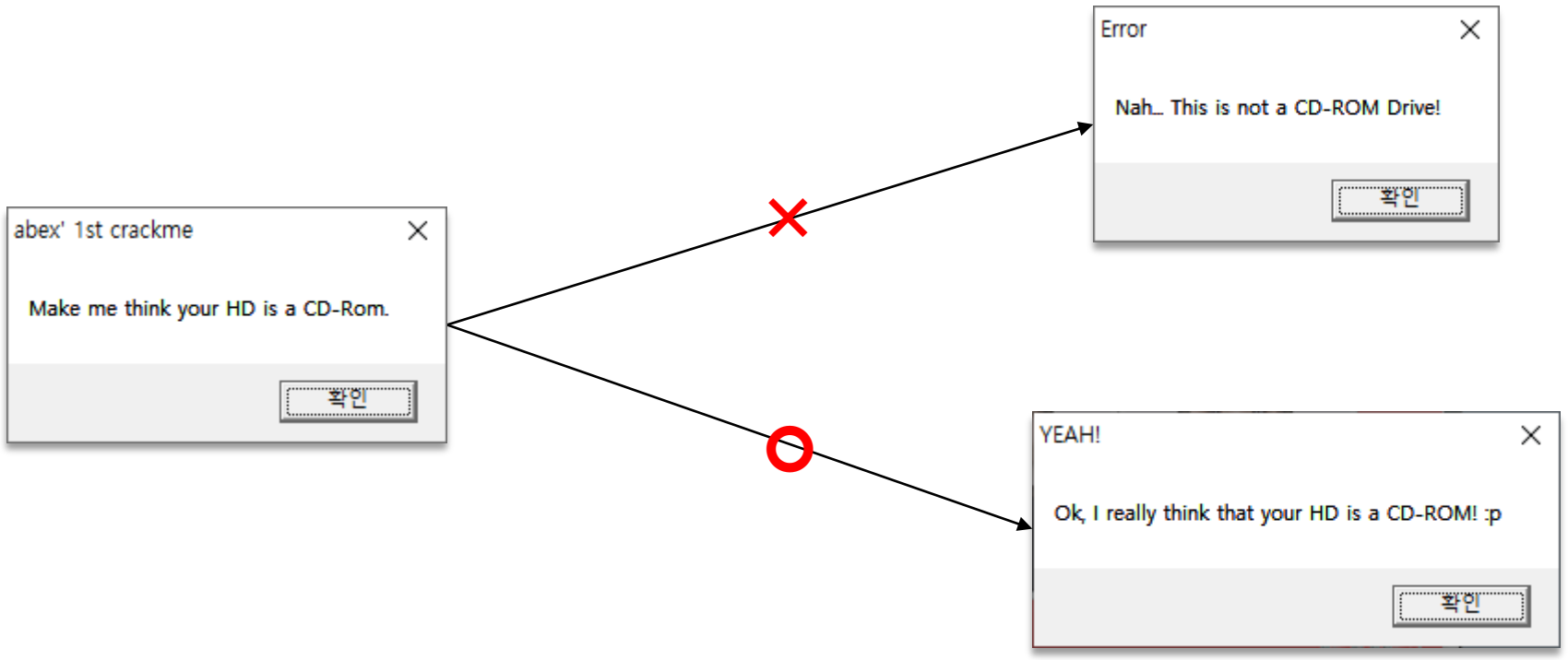
### 스택 프레임

함수 프롤로그 & 에필로그

# 어셈블리어

abex' crackme + 스택 프레임

01



# 어셈블리어

abex' crackme + 스택 프레임

01

```
00401000  ┌·  6A 00          PUSH 0
00401002  ·   68 00204000    PUSH OFFSET 00402000        ASCII "abex' 1st crackme"
00401007  ·   68 12204000    PUSH OFFSET 00402012        ASCII "Make me think your HD is a CD-Rom."
0040100C  ·   6A 00          PUSH 0
0040100E  ·   E8 4E000000    CALL <JMP.&USER32.MessageBoxA>
00401013  ·   68 94204000    PUSH OFFSET 00402094        ASCII "c:\"
00401018  ·   E8 38000000    CALL <JMP.&KERNEL32.GetDriveTypeA>
0040101D  ·   46             INC ESI
0040101E  ·   48             DEC EAX
0040101F  ·⌄  EB 00          JMP SHORT 00401021
00401021  >   46             INC ESI
00401022  ·   46             INC ESI
00401023  ·   48             DEC EAX
00401024  ·   3BC6           CMP EAX,ESI
00401026  ·⌄  74 15          JE SHORT 0040103D
00401028  ·   6A 00          PUSH 0
0040102A  ·   68 35204000    PUSH OFFSET 00402035        ASCII "Error"
0040102F  ·   68 3B204000    PUSH OFFSET 0040203B        ASCII "Nah... This is not a CD-ROM Drive!"
00401034  ·   6A 00          PUSH 0
00401036  ·   E8 26000000    CALL <JMP.&USER32.MessageBoxA>
0040103B  ·⌄  EB 13          JMP SHORT 00401050
0040103D  >   6A 00          PUSH 0
0040103F  ·   68 5E204000    PUSH OFFSET 0040205E        ASCII "YEAH!"
00401044  ·   68 64204000    PUSH OFFSET 00402064        ASCII "Ok, I really think that your HD is a CD-ROM! :p"
00401049  ·   6A 00          PUSH 0
0040104B  ·   E8 11000000    CALL <JMP.&USER32.MessageBoxA>
00401050  >   E8 06000000    CALL <JMP.&KERNEL32.ExitProcess>
```

PUSH: 스택에 값을 저장

CALL: 지정된 주소의 함수 호출

INC: 값을 1 증가

DEC: 값을 1 감소

JMP: 지정된 주소로 점프

CMP: 주어진 두 값을 비교

JE: 조건 분기(같으면 점프)

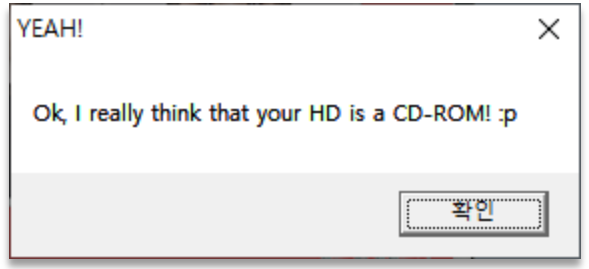
# 어셈블리어

abex' crackme + 스택 프레임

## 01

```
00401024   ·   3BC6          CMP EAX,EAX
00401026   ·⌄  74 15         JNE SHORT 0040103D
00401028   ·   6A 00         PUSH 0
0040102A   ·   68 35204000   PUSH OFFSET 00402035                ASCII "Error"
0040102F   ·   68 3B204000   PUSH OFFSET 0040203B                ASCII "Nah... This is not a CD-ROM Drive!"
00401034   ·   6A 00         PUSH 0
00401036   ·   E8 26000000   CALL <JMP.&USER32.MessageBoxA>
0040103B   ·⌄  EB 13         JMP SHORT 00401050
0040103D   >   6A 00         PUSH 0
0040103F   ·   68 5E204000   PUSH OFFSET 0040205E                ASCII "YEAH!"
00401044   ·   68 64204000   PUSH OFFSET 00402064                ASCII "Ok, I really think that your HD is a CD-ROM! :p"
00401049   ·   6A 00         PUSH 0
0040104B   ·   E8 11000000   CALL <JMP.&USER32.MessageBoxA>
00401050   >   E8 06000000   CALL <JMP.&KERNEL32.ExitProcess>
```

1. 어셈블리어를 변경: JE > JMP
2. 어셈블리어를 변경: JE > JNE
3. 조건을 변경: CMP EAX,ESI > CMP EAX,EAX



# 어셈블리어

abex' crackme + 스택 프레임

01

PUSH: 스택에 값을 저장

CALL: 지정된 주소의 함수 호출

INC: 값을 1 증가

DEC: 값을 1 감소

JMP: 지정된 주소로 점프

CMP: 주어진 두 값을 비교

JE: 조건 분기(같으면 점프)

POP: 스택에서 값을 뺌

MOV: 앞 오퍼랜드에 뒤 오퍼랜드의 값을 저장

LEA: 앞 오퍼랜드에 뒤 오퍼랜드의 주소 값을 저장

ADD: 두 개의 오퍼랜드를 더해서 앞의 오퍼랜드에 저장

SUB: 앞 오퍼랜드에서 뒤 오퍼랜드를 뺀 다음 앞의 오퍼랜드에 저장

# 어셈블리어

abex' crackme + 스택 프레임

01

## 레지스터란?

: CPU 내에서 자료를 보관하는 빠른 기억 장소, 변수와 같은 역할로 메모리에서 연산을 할 때 사용

EAX - 사칙연산 등 산술 연산에 자동으로 사용

EBX - 연산 용도 / 공간이 필요할 때 사용

ECX - 반복문 카운팅 / 연산 용도

EDX - 연산 용도(보조)

ESI, EDI - 시작지 인덱스, 목적지 인덱스

EBP - 스택의 시작 주소 값을 가리키는 포인터

ESP - 스택의 끝 주소 값을 가리키는 포인터

# 어셈블리어

abex' crackme + 스택 프레임

02

## 스택 프레임이란?

: ESP가 아닌 EBP 레지스터를 사용하여 스택 내의 로컬 변수, 파라미터, 복귀 주소에 접근하는 기법

스택 프레임 생성

```
PUSH    EBP                 ⇒ 기존의 EBP값을 스택에 집어 넣어라
MOV     EBP, ESP            ⇒ ESP의 값을 EBP로 옮겨라
...
```

스택 프레임 해제

```
MOV     ESP, EBP            ⇒ ESP 값을 원래대로 복원시켜라
POP     EBP                 ⇒ 스택에 백업한 EBP 값을 복원시켜라
RETN                        ⇒ 스택에 저장된 복귀 주소로 리턴하여 함수 종료
```