

시스템보안실무

중간고사

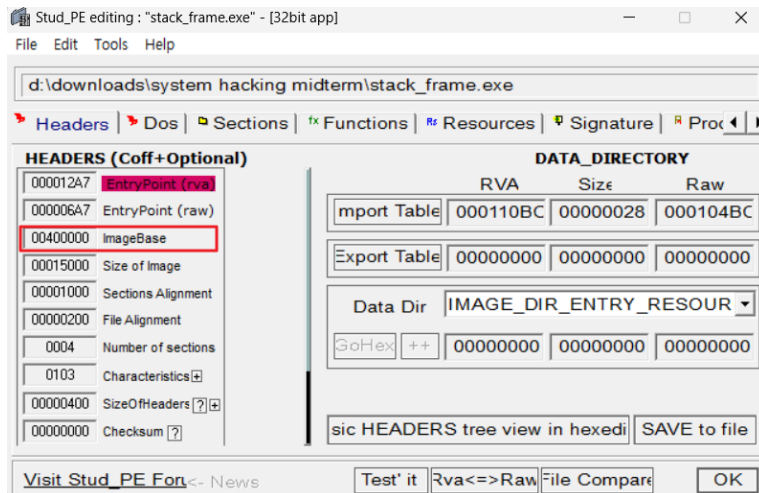


학번	2024771005
이름	박예서
제출일	2024.10.25.

Q1. 해당 실행 파일이 매핑될 메모리상의 가상주소(Imagebase)는 무엇입니까?

정답: 0x00400000

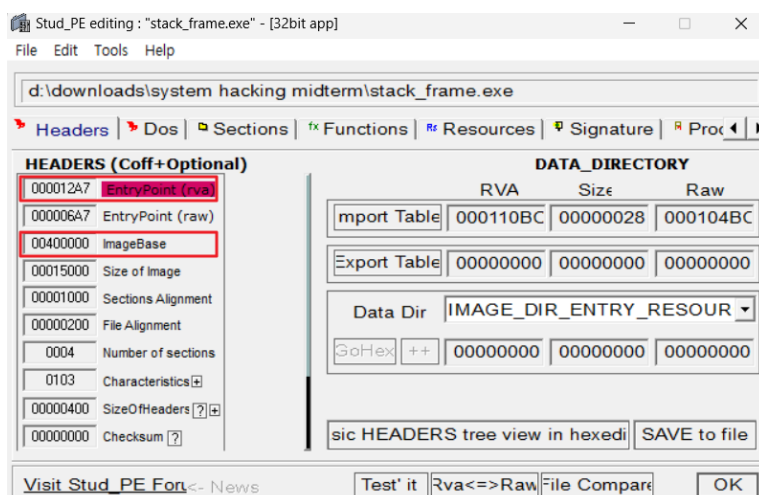
증명: stude_pe 프로그램으로 ImageBase값을 확인해 보면 0x00400000 인 것을 확인할 수 있다.



Q2. 해당 실행 파일이 메모리에 로드된 후 가장먼저 실행되는 코드의 주소(Address of Entrypoint)는 무엇입니까? (옵셋 값 말고 메모리 상의 가상주소 값을 적어 주세요)

정답: 0x004012A7

증명: Imagebase값과 EntryPoint(rva)값을 더해주면 0x004012A7이 된다.



Q3. 함수 "Func_A"가 호출되는 지점의 가상주소는 무엇입니까?

정답: 0x0040106C

증명: IDA를 통해 분석한 결과 "Func_A"는 CALL을 통해 0x0040106C지점에서 호출된다.

```
.text:00401056      mov     [ebp+var_8], 3
.text:0040105D      mov     [ebp+var_4], 4
.text:00401064      mov     eax, [ebp+var_4]
.text:00401067      push    eax
.text:00401068      mov     ecx, [ebp+var_8]
.text:0040106B      push    ecx
.text:0040106C      call    _Func_A
.text:00401071      add     esp, 8
.text:00401074      xor     eax, eax
.text:00401076      mov     esp, ebp
.text:00401078      pop     ebp
.text:00401079      retn
.text:00401079      main    endn
```

Q4. 함수 "MAIN" 에서 할당하는 지역변수 공간의 크기는 몇 바이트입니까?

정답: 8바이트

증명: "sub esp, 8" 코드가 실행되는 것을 볼 수 있고, 이 과정을 통해 8바이트를 확보한다.

```
Func_B
Func_A
main
pre_c_initialization
post_pgo_initialization
pre_cpp_initialization
__scrt_common_main_seh
_mainCRTStartup
find_pe_section
__scrt_acquire_startup_lock
__scrt_initialize_crt
__scrt_initialize_onexit_tables
__scrt_is_nonwritable_in_current_image
__scrt_release_startup_lock
__scrt_uninitialize_crt
__onexit

.text:00401050      argc    = dword ptr 8
.text:00401050      argv    = dword ptr 0Ch
.text:00401050      envp    = dword ptr 10h
.text:00401050      push    ebp
.text:00401051      mov     ebp, esp
.text:00401053      sub     esp, 8
.text:00401056      mov     [ebp+var_8], 3
.text:0040105D      mov     [ebp+var_4], 4
.text:00401064      mov     eax, [ebp+var_4]
.text:00401067      push    eax
.text:00401068      mov     ecx, [ebp+var_8]
.text:0040106B      push    ecx
.text:0040106C      call    Func_A
```

Q5. 함수 "MAIN"의 지역변수 공간을 사용하는 변수가 정수(Integer, 사이즈 4바이트)일 경우 몇 개의 지역변수가 할당 되었다고 추정되나요? 그리고 그 이유는 무엇입니까?

정답: 2개

증명: 'sub esp, 8'을 통해 8바이트의 지역변수 공간이 확보되었고(4번 사진 참조), 정수인 4바이트라고 한다면 $2(8/4=2)$ 개의 지역변수가 할당되었다고 할 수 있다.

Q6. 함수 "Func_A" 가 종료된 이후 실행되는 코드의 주소(리턴 주소)는 무엇입니까?

정답: 0x00401071

증명: "Func_A"의 retn에 도달하면 "call func_A"의 다음 줄인 0x00401071 "add esp, 8"로 리턴된다.

• .text:0040106C	call	Func_A
• .text:00401071	add	esp, 8
• .text:00401074	xor	eax, eax
• .text:00401076	mov	esp, ebp

Q7. 디버거를 이용하여 함수 "MAIN"의 "0x00401053 SUB ESP, 8" 코드까지 실행해 주세요.

1) 해당 코드까지 실행된 상태를 증명해 주세요. 이 때 EBP 레지스터는 Func_A 함수의 SFP(Saved Frame Pointer)를 가리킵니다.

2)EBP 레지스터의 값, 3)EBP 레지스터가 가리키는 SFP 값은 무엇입니까?

EBP 레지스터의 값 : 0x0019FF2C

EBP 레지스터가 가리키는 SFP 값: 0x0019FF74

증명: EBP 레지스터 값이 0x0019FF2C 인 것을 확인할 수 있고, 스택창에서 EBP 레지스터가 갖고 있는 주소 값인 0x0019FF2C가 SFP 값인 0x0019FF74를 가리키고 있다.

The screenshot shows a debugger window with the following components:

- Assembly Window:** Displays assembly code. The instruction at address 00401071 is `add esp, 8`, which is highlighted with a red box. The instruction at 0040106C is `call <stack_frame._Func_A>`.
- Registers Window:** Shows the value of the EBP register as 0019FF2C, which is also highlighted with a red box.
- Stack Window:** Shows the stack contents. The value at address 00401126 is 00401126, which is the SFP for the caller. This address is highlighted with a red box.
- Disassembly Window:** Shows the disassembly of the code, including the `call <stack_frame._Func_A>` instruction.

Q8. 디버거를 이용하여 함수 "Func_A"의 "0x00401041 POP EBP" 코드까지 실행해 주세요.

1) 해당 코드까지 실행된 상태를 증명해 주세요. 이제 해당 함수는 "RETN" 코드를 실행하기 직전 상태이며 종료되기 전입니다.

2) 현재 스택의 최상단에 있는 값은 "어떤의미"를 가집니까?

정답: "Func_A" 실행이 끝나면 돌아갈 주소이다.

증명:

The screenshot shows a debugger window with the following details:

- Assembly List:**
 - 00401041: 5D pop ebp
 - 00401042: C3 ret
 - 00401043: CC int3
 - 00401044: CC int3
 - 00401045: CC int3
 - 00401046: CC int3
 - 00401047: CC int3
 - 00401048: CC int3
 - 00401049: CC int3
 - 0040104A: CC int3
 - 0040104B: CC int3
 - 0040104C: CC int3
 - 0040104D: CC int3
 - 0040104E: CC int3
 - 0040104F: CC int3
 - 00401050: 55 push ebp
 - 00401051: 8BEC mov ebp,esp
 - 00401052: 83EC 08 sub esp,8
 - 00401053: C745 F8 03000000 mov dword ptr ss:[ebp-8],3
 - 00401054: C745 FC 04000000 mov dword ptr ss:[ebp-4],4
- Registers:**
 - EAX: 00000000
 - EBX: 0031E000
 - ECX: 00000000
 - EDX: 4159F000
 - EBP: 0019F000
 - ESP: 0019F000
 - ESI: 00412000
 - EDI: 00412000
 - EIP: 00401041
 - EFLAGS: 00
- Stack:**
 - 0019FF18: 00401071 (Ret Address)
 - 0019FF1C: 00000003
 - 0019FF20: 00000004
 - 0019FF24: 00000003
 - 0019FF28: 00000004
 - 0019FF2C: 0019FF74

Q9. 디버거를 이용하여 함수 "Func_B"의 "0x00401001 MOV EBP, ESP" 코드까지 실행해 주세요.

1) 해당 코드까지 실행된 상태를 증명해 주세요. 해당 함수의 Stack Frame이 빌드된 상태입니다. 이제 "PUSH ECX" 코드가 실행되기 직전입니다. 현재 ESP 레지스터와 EBP 레지스터는 동일한 값을 가지고 있을 것입니다.

The screenshot shows a debugger window with the following details:

- Assembly List:**
 - 00401001: 8BEC mov ebp,esp
 - 00401002: 51 push ecx
 - 00401003: C745 FC 00000000 mov dword ptr ss:[ebp-4],0
 - 00401004: 8BE5 mov esp,ebp
 - 00401005: 5D pop ebp
 - 00401006: C3 ret
 - 00401007: CC int3
 - 00401008: 55 push ebp
 - 00401009: 8BEC mov ebp,esp
 - 0040100A: 83EC 0C sub esp,C
 - 0040100B: C745 F4 00000000 mov dword ptr ss:[ebp-C],0
 - 0040100C: C745 FC 05000000 mov dword ptr ss:[ebp-4],5
 - 0040100D: C745 F8 06000000 mov dword ptr ss:[ebp-8],6
 - 0040100E: 8B45 08 mov eax,dword ptr ss:[ebp+8]
- Registers:**
 - EAX: 00000012
 - EBX: 00318000
 - ECX: 00000003
 - EDX: 509117CC
 - EBP: 0019FF00
 - ESP: 0019FF00
 - ESI: 00412CD4
 - EDI: 00412CD8
 - EIP: 00401003
- Stack:**
 - [ebp-04]: __a
 - [ebp-04]: __a

2) 두 레지스터는 현재 SFP를 포인팅 하고 있습니다. SFP 값(EBP 혹은 ESP 레지스터)은 무엇입니까?

정답: 0x0017FF14

증명: ESP 레지스터랑 EBP 레지스터는 0x0019FF00으로 똑같은 값을 가지고 있으며, 이 주소가 가리키는 SFP의 값은 0x0017FF14이다.

The screenshot shows a debugger interface with three main panes:

- Assembly Pane:** Displays assembly instructions. The instruction at address 00401037 is `mov dword ptr ss:[ebp-C], eax`. The instruction at address 0040103F is `call <stack_frame.Func_B>`.
- Register Pane:** Shows the values of various registers. EBP and ESP are both set to 0019FF00.
- Stack Pane:** Shows the current stack frame. The stack pointer is at 0019FF00. A pointer to 0017FF14 is visible in the stack.

Q10. 디버거를 이용하여 함수 "Func_A"의 "0x00401037 [EBP+var_C], EAX" 코드까지 실행해 주세요.

- 1) 해당 코드까지 실행된 상태를 증명해 주세요. 이제 함수 "Func_B"가 호출되기 직전 상태일 것입니다.
- 2) "[EBP+var_C]"는 해당 함수의 지역변수 영역입니다. 스택을 확인하여 어떠한 값이 기록되는지 증명하세요.

정답: 12가 기록된다.

증명: "mov dword ptr ss:[ebp-C], eax" 명령어는 EAX 레지스터에 저장된 값을 스택 프레임의 [EBP+var_C] 위치에 저장한다. 현재 EAX 레지스터에 12가 저장되어 있으므로, 해당 명령어가 실행되면 지역 변수 [EBP+var_C]에 12가 기록된다.

(다음 장에 사진 첨부)

주소	Hex	Dec	Comment
0040101D	C745 FC 05000000	mov dword ptr ss:[ebp-4],5	
00401024	C745 F8 06000000	mov dword ptr ss:[ebp-8],6	
0040102B	8B45 08	mov eax,dword ptr ss:[ebp+8]	
0040102E	0345 0C	add eax,dword ptr ss:[ebp+C]	
00401031	0345 FC	add eax,dword ptr ss:[ebp-4]	
00401034	0345 F8	add eax,dword ptr ss:[ebp-8]	
00401037	8945 F4	mov dword ptr ss:[ebp-C],eax	
0040103A	E8 C1FFFFFF	call <stack_frame._Func_6>	
0040103F	8BE5	mov esp,ebp	
00401041	5D	pop ebp	
00401042	C3	ret	
00401043	CC	int3	
00401044	CC	int3	
00401045	CC	int3	
00401046	CC	int3	
00401047	CC	int3	
00401048	CC	int3	
00401049	CC	int3	
0040104A	CC	int3	
0040104B	CC	int3	

Register	Value
EAX	00000012
EBX	00332000
ECX	00000003
EDX	131DCC98
EBP	0019FF14
ESP	0019FF08
ESI	00412CD4
EDI	00412CD8
EIP	0040103A
EFLAGS	00000216

주소	Hex	Dec	Comment
77CA1000	18 00 00 00	00 00 00 00	B8 19 CA 77 40 00 00 00
77CA1010	00 00 00 00	00 00 00 00	14 00 16 00 08 B0 CA 77
77CA1020	00 00 02 00	8C 59 CA 77 B0 61 CD 77	E0 5A CD 77