

## 3주차 분석 보고서

정보보호학과 우은지

### 1. main 함수 찾기

디버거에 실행 파일을 넣고 F8로 실행을 해보면

```
004012C7 90      NOP
004012C8 $ 83EC 1C    SUB ESP,1C
004012D3 . C7D424 01000001 MOV DWORD PTR SS:[ESP],1
004012DA . FF15 88814000 CALL DWORD PTR DS:[<&nsvcrtdll.__set_app_type>]
004012E0 . E8 BBFEFFFF CALL sun.004011A0
004012E5 . 8DB426 00000001 LEA ESI,DWORD PTR DS:[ESI]
```

004011A0 함수 호출되는 부분을 실행하면 프로그램이 끝나는 것을 확인할 수 있다.

따라서 저 004011A0 함수를 F7을 이용해서 내부를 보았다.

```
004011A0 $ 53      PUSH EBX
004011A1 . 83EC 18     SUB ESP,18
004011A4 . A1 24514000 MOV EAX,DWORD PTR DS:[405124]
004011A9 . 85C0       TEST EAX,EAX
004011AB . 74 1C      JE SHORT sun.004011C9
004011AD . C74424 08 0001 MOV DWORD PTR SS:[ESP+8],0
004011B5 . C74424 04 0201 MOV DWORD PTR SS:[ESP+4],2
004011BD . C7D424 00000001 MOV DWORD PTR SS:[ESP],0
004011C4 . FFD0      CALL EAX
004011C6 . 83EC 0C     SUB ESP,0C
004011C9 > C7D424 00104001 MOV DWORD PTR SS:[ESP],sun.00401000
004011D0 . E8 FFE00000 CALL <JMP.&kernel32.SetUnhandledExceptionFilter>
004011D5 . 83EC 04     SUB ESP,4
004011D8 . E8 73060000 CALL sun.00401850
004011DD . A1 04404000 MOV EAX,DWORD PTR DS:[404040]
004011E2 . 89D424     MOV DWORD PTR SS:[ESP],EAX
004011E5 . E8 460F0000 CALL sun.00402130
004011EA . E8 01020000 CALL sun.004013F0
004011EF . A1 08704000 MOV EAX,DWORD PTR DS:[407008]
004011F4 . 85C0       TEST EAX,EAX
004011F6 . 75 4A      JNZ SHORT sun.00401242
004011F8 > E8 332D0000 CALL <JMP.&nsvcrtdll._p_fnode>
004011FD . 8B15 08404000 MOV EDI,DWORD PTR DS:[404008]
00401203 . 8910      MOV DWORD PTR DS:[EAX],EDI
00401205 . E8 960C0000 CALL sun.00401EAD
0040120A . 83E4 F0     AND ESP,FFFFFFF0
0040120D . E8 EE070000 CALL sun.00401A00
00401212 . E8 212D0000 CALL <JMP.&nsvcrtdll._p_environ>
00401217 . 8B00      MOV EAX,DWORD PTR DS:[EAX]
00401219 . 894424 08   MOV DWORD PTR SS:[ESP+8],EAX
0040121D . A1 00704000 MOV EAX,DWORD PTR DS:[407000]
00401222 . 894424 04   MOV DWORD PTR SS:[ESP+4],EAX
00401226 . A1 04704000 MOV EAX,DWORD PTR DS:[407004]
0040122B . 89D424     MOV DWORD PTR SS:[ESP],EAX
0040122E . E8 45010000 CALL sun.00401378
```

004011A0 함수의 내부 모습인데 BREAKPOINT 걸려있는 부분을 실행하면 실행 창에 해당 프로그램

결과가 출력되는 것을 확인할 수 있다. 따라서 004011A0 함수가 main 함수인 것을 알 수 있다.

## 2. 서브함수 찾기

```

00401378 55      PUSH EBP
00401379 89E5    MOV EBP,ESP
0040137B 83E4 F0 AND ESP,FFFFFFF0
0040137E 83EC 20 SUB ESP,20
00401381 E8 7A060000 CALL sun.00401A00
00401386 C74424 1C 0901 MOV DWORD PTR SS:[ESP+1C],9
0040138E C74424 18 0701 MOV DWORD PTR SS:[ESP+18],7
00401396 8B4424 18 MOV EAX,DWORD PTR SS:[ESP+18]
0040139A 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
0040139E 8B4424 1C MOV EAX,DWORD PTR SS:[ESP+1C]
004013A2 890424  MOV DWORD PTR SS:[ESP],EAX
004013A5 E8 B6FFFFFF CALL sun.00401360
004013AA 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
004013AE C70424 005040 MOV DWORD PTR SS:[ESP],sun.00405000
004013B5 E8 FE2A0000 CALL <JMP.&msvcrt.printf>
004013BA 8B4424 18 MOV EAX,DWORD PTR SS:[ESP+18]
004013BE 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
004013C2 8B4424 1C MOV EAX,DWORD PTR SS:[ESP+1C]
004013C6 890424  MOV DWORD PTR SS:[ESP],EAX
004013C9 E8 9FFFFFFF CALL sun.00401360
004013CE 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
004013D2 C70424 0B5040 MOV DWORD PTR SS:[ESP],sun.0040500B
004013D9 E8 DA2A0000 CALL <JMP.&msvcrt.printf>
004013DE B8 00000000 MOV EAX,0
004013E3 C9      LEAVE
004013E4 C3      RETN

```

ASCII "sun : Zd %"  
printf

ASCII "minus : Zd %"  
printf

main 함수의 내부이다. call이 되는 부분에서 F7를 이용해 실행하여 보면 BREAKPOINT 걸린 부분이 sum과 minus 함수를 call하고 있는 부분임을 알 수 있다.

## 3. 레지스터 변경 확인

EAX: 산술 연산 결과 저장

EIP: 다음 명령어 저장

```

004012E3 50      MOV ESP,ECX
004012E4 83EC 1C SUB ESP,1C
004012E7 C70424 010000 MOV DWORD PTR SS:[ESP],1
004012EB FF35 80814000 CALL DWORD PTR DS:[&msvcrt.__set_app_type]
004012F0 E8 B0FEFFFF CALL sun.00401100
004012E5 80B426 000000 LEA ESI,DWORD PTR DS:[ESI]
004012EC 807426 00 LEA ESI,DWORD PTR DS:[ESI]
004012F0 83EC 1C SUB ESP,1C
004012F3 C70424 020000 MOV DWORD PTR SS:[ESP],2
004012F8 FF35 80814000 CALL DWORD PTR DS:[&msvcrt.__set_app_type]
00401300 E8 90FEFFFF CALL sun.00401100

```

EAX 0061FCE2  
ECX 00401200 sun.<ModuleEntryPoint>  
EDX 00401200 sun.<ModuleEntryPoint>  
EBX 0039C000  
ESP 0061F580  
EBP 0061FF80  
ESI 00401200 sun.<ModuleEntryPoint>  
EDI 00401200 sun.<ModuleEntryPoint>  
EIP 00401200 sun.<ModuleEntryPoint>

F8을 이용해서 한줄한줄 실행할 때마다, EIP의 값이 다음 명령어로 바뀌는 것을 확인했고,

sum 함수에서 mov를 통해 스택에 있는 값 7을 가져와서 EAX에 저장하는 것을 확인했고,

ADD를 통해 EAX 값인 7과 EDXD에 있던 값인 9를 더한 16을 EAX에 저장한 것을 확인했다. (16진수로 저장되어 10이라고 저장됨)

#### 4. 스택 변경 확인

여기에서 CALL로 004011A0으로 갈 때 스택을 확인해보면

스택에 004011A0 밑에 있는 명령어의 주소가 저장되어있는 것을 알 수 있다.

(004011A0 함수를 끝내고 돌아가려고)

MAIN 함수를 들어가는 부분인데 이때도 MAIN 함수를 들어가서 스택을 처음 확인해보면

다음 명령어 주소인 00401233이 스택에 저장되어 있는 것을 확인할 수 있었다.

이뿐만 아니라 CALL 명령어를 실행하면 바로 밑 명령어의 주소가 스택에 저장되는 것을 확인 할 수 있었다.

다음은 스택 관련 레지스터 EBP, ESP에 대해 서브루틴 호출 전 후를 확인해봤다.

```

00401386 | C74024 1C 0000 MOV EBP, PTR SS:[ESP+10],9
0040138E | C74024 10 0700 MOV EBP, PTR SS:[ESP+10],7
00401396 | 8B4024 10      MOV EAX, DWORD PTR SS:[ESP+10]
00401398 | 8B4024 04      MOV EAX, DWORD PTR SS:[ESP+4],EAX
0040139E | 8B4024 1C      MOV EAX, DWORD PTR SS:[ESP+10],EAX
004013A2 | 8B4024 04      MOV EAX, DWORD PTR SS:[ESP+4],EAX
004013A8 | EB 00FFFFFF    CALL EBX,00000000
004013B0 | C70024 005040 MOV EBP, PTR SS:[ESP],sun.00405000
004013B5 | EB FE200000    CALL <JMP .405000>
                                printf
  
```

Registers (FPU)

EAX	00000000
ECX	00401960 sun.00401960
EDX	00000000
EBX	0035C000
ESP	0061FF20
EBP	0061FF20
ESI	00401200 sun.<ModuleEntryPoint>
EDI	00401200 sun.<ModuleEntryPoint>

MAIN 함수 속 SUM 함수 호출 직전 EBP, ESP에 저장된 값이다.

SUM 함수를 호출하면 어떻게 변하는지 확인해봤다.

```

00401360 | 55      PUSH EBP
00401361 | B9E5    MOV EBP, ESP
00401363 | 8B55 00 MOV EAX, DWORD PTR SS:[EBP+0]
00401366 | 8B55 0C MOV EAX, DWORD PTR SS:[EBP+4]
00401369 | 8B00    MOV EAX, EAX
0040136B | 5D      POP EBP
0040136D | C3      RET
  
```

Registers (FPU)

EAX	00000000
ECX	00401960 sun.00401960
EDX	00000000
EBX	00294000
ESP	0061FF20
EBP	0061FF20

먼저 PUSH EBP를 통해 MAIN 함수의 EBP를 스택에 저장해서 ESP의 값만 변한 것을 알 수 있다.

```

00401360 | 55      PUSH EBP
00401361 | B9E5    MOV EBP, ESP
00401363 | 8B55 00 MOV EAX, DWORD PTR SS:[EBP+0]
00401366 | 8B55 0C MOV EAX, DWORD PTR SS:[EBP+4]
00401369 | 8B00    MOV EAX, EAX
0040136B | 5D      POP EBP
0040136D | C3      RET
  
```

Registers (FPU)

EAX	00000000
ECX	00401960 sun.00401960
EDX	00000000
EBX	00294000
ESP	0061FF20
EBP	0061FF20

다음은MOV EBP, ESP를 통해 서브 루틴의 EBP의 값을 만들어주는 것을 확인했다.

```

0040135E | 90      NOP
0040135F | 90      NOP
00401360 | 55      PUSH EBP
00401361 | B9E5    MOV EBP, ESP
00401363 | 8B55 00 MOV EAX, DWORD PTR SS:[EBP+0]
00401366 | 8B55 0C MOV EAX, DWORD PTR SS:[EBP+4]
00401369 | 8B00    MOV EAX, EAX
0040136B | 5D      POP EBP
0040136D | C3      RET
0040136E | 55      PUSH EBP
  
```

Registers (FPU)

EAX	00000000
ECX	00401960 sun.00401960
EDX	00000000
EBX	00294000
ESP	0061FF20
EBP	0061FF20
ESI	00401200 sun.<ModuleEntryPoint>
EDI	00401200 sun.<ModuleEntryPoint>

POP EBP의 결과 EBP와 ESP의 값이 서브루틴 호출 전과 같아진 것을 확인할 수 있었다.