리버싱 이 정도는 알아야지

Chapter 04. 어셈블리와 친해지기

http://securityfactory.tistory.com

목 차

Chapter 04

어셈블리와 친해지기

SECURITY FACTORY

1. 예제 분석_Sample 02.exe

1.1. 도전 과제

Sample 02.exe를 분석해보겠습니다. 단순히 분석하고 끝내려는 건 아니고요. 나름의 도전 과제가 있습니다.

> ※ 도전 과제
>
> - Sample 02.exe 가 "If Code!!" 문자열을 출력하도록 만들어보세요.
> - 전체적인 동작을 이해하고, 문제 해결을 위해서 어디를 어떻게 고쳤는지 파악해야 합니다.
> - 힌트: 'if 조건문'을 사용해서 만들었습니다.

먼저 한번 실행시켜보겠습니다. 아무 것도 출력되지 않네요.

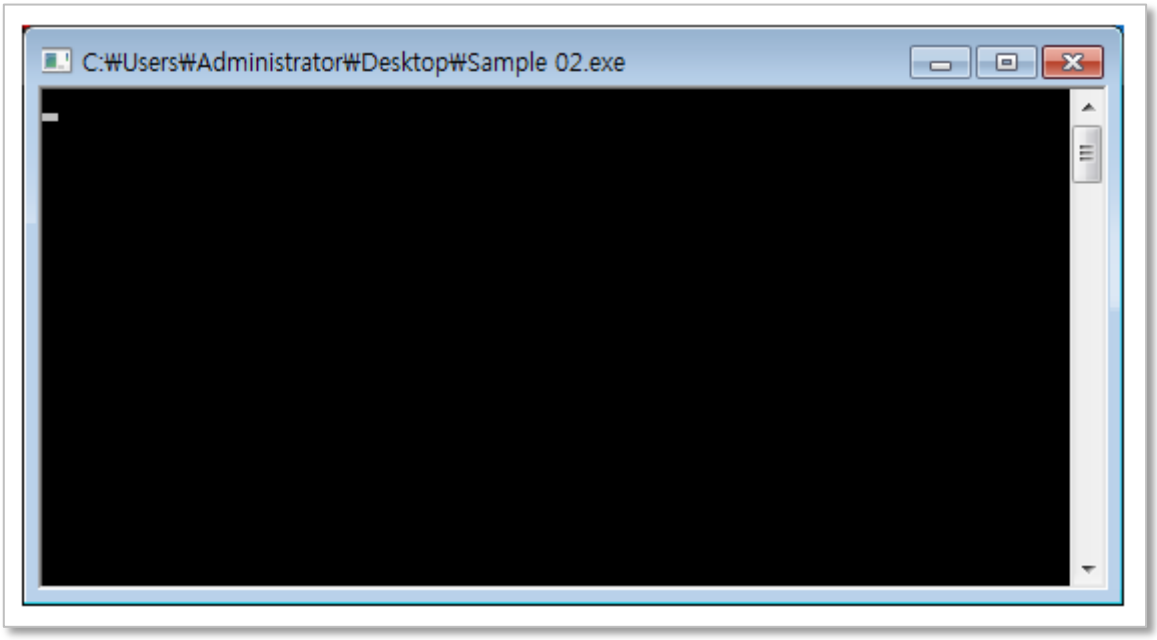

그림 1-1 Sample 02.exe 실행 화면

우리는 Sample 02.exe를 분석해서 문자열이 출력되지 않는 원인을 파악한 다음에 [그림 1-2]와 같이 만들어야 합니다. 여기서 1차적인 목표는 "If Code!!" 문자열 출력이지만, 본질은 근본적인 원인을 파악하고 그 과정에서 어셈블리를 공부하려는 것 입니다.

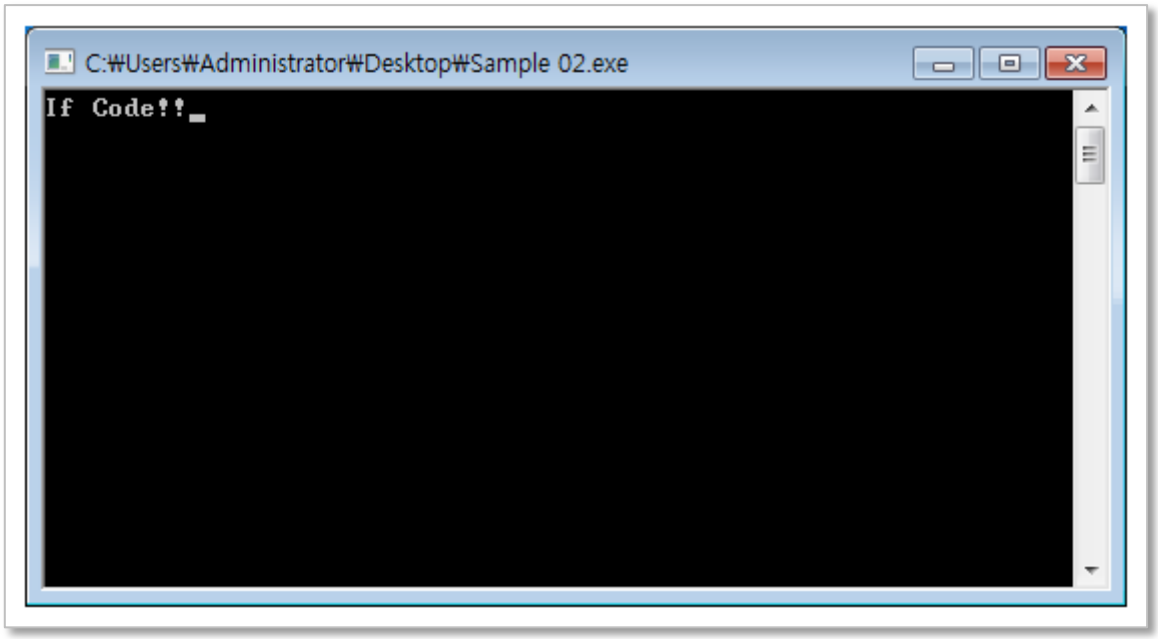

그림 1-2 문제 해결시 출력되는 문자열

1.2. 코드 분석_Level.1 | 흐름 파악하기

분석에 앞서, 문제가 무엇일지 고민을 좀 해봅시다. 도전 과제를 보면, “Sample 02.exe는 ’if 조건문’을 사용해서 만들었다.”고 나와있습니다. 음, 왠지 ’if 문’에 들어가는 조건이 일치하느냐에 따라 문자열 출력 여부가 결정될 것으로 보입니다. (너무 당연해 보이나요? 그래도 여러 가지 가능성을 고민해보고, 확인하는 습관이 분석 능력을 향상시키는데 아주 큰 역할을 합니다.)

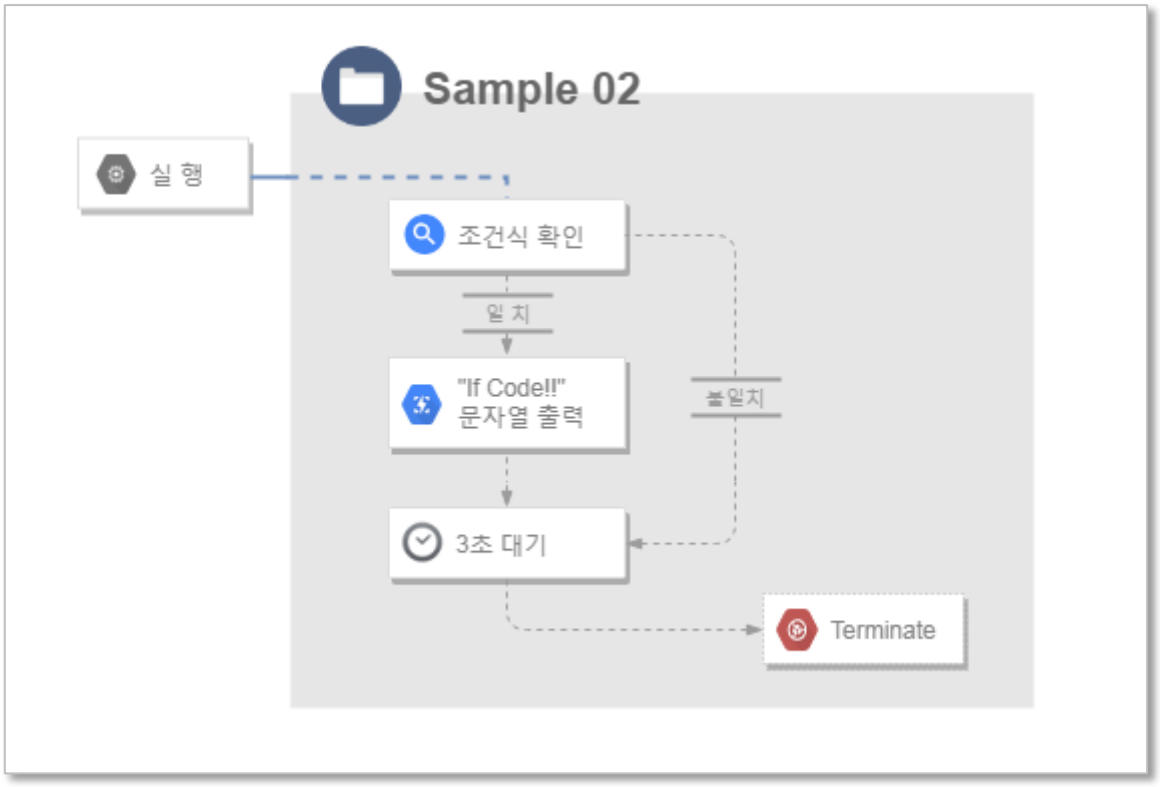

그림 1-3 Sample 02.exe 예상 동작흐름

이렇게 원인을 유추하고 나니까, 어떻게 접근해야 할지도 떠오르네요. 우선 조건에 따라 문자열 출력 여부가 결정된다면, 그 조건을 확인하는 지점이 있을 것입니다. 그 지점을 찾아가서 예상이 맞는지 확인합니다. 그리고 조건이 맞지 않는 원인을 파악한 후에 문제를 해결하면 될 것으로 보입니다. 이대로 해보겠습니다.

1.3. **코드 분석_Level.2 | 원인 도출**

문제 발생 지점을 찾고, 확인하는 시간을 가지겠습니다. 메인 시작 지점부터 하나씩 살펴봅시다.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

1.4. 코드 분석_Level.3 | 문제 해결

문제의 원인을 파악하고, 해결 방법을 고민해보는 시간입니다.

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

GetLocalTime()은 현재의 날짜와 시간 정보를 얻을 수 있는 API입니다.

GetLocalTime function

Retrieves the current local date and time.

To retrieve the current date and time in Coordinated Universal Time (UTC) format, use the GetSystemTime function.

Syntax

```
void WINAPI GetLocalTime(
      _In_  LPSYSTEMTIME  lpSystemTime
);
```

Parameters

lpSystemTime [out]
A pointer to a SYSTEMTIME structure to receive the current local date and time.

SYSTEMTIME은 년, 월, 일 등 다양한 시간 정보를 입력할 수 있도록 구성되어 있는 구조체입니다.

```
typedef struct _SYSTEMTIME {
  WORD wYear;                  // 년
  WORD wMonth;                 // 월
  WORD wDayOfWeek;             // 요일
  WORD wDay;                   // 일
  WORD wHour;                  // 시
  WORD wMinute;                // 분
  WORD wSecond;                // 초
  WORD wMilliseconds;          // 1/1000 초
} SYSTEMTIME, *PSYSTEMTIME;
```

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

2. 예제 분석_Sample 03.exe

2.1. 도전 과제

Sample 03.exe를 분석해보겠습니다. 다음은 도전 과제입니다.

※ 도전 과제

- Sample 03.exe 가 "For Code!!" 문자열을 출력하도록 만들어보세요.
- 전체적인 동작을 이해하고, 문재 해결을 위해서 어디를 어떻게 고쳤는지를 파악해야 합니다.
- 힌트: 'for 반복문'을 사용해서, 문자가 하나씩 출력되도록 만들었습니다.

실행시켜보니 이번에는 알 수 없는 쓰레기 문자들을 출력하고, 3초 뒤에 종료됩니다.

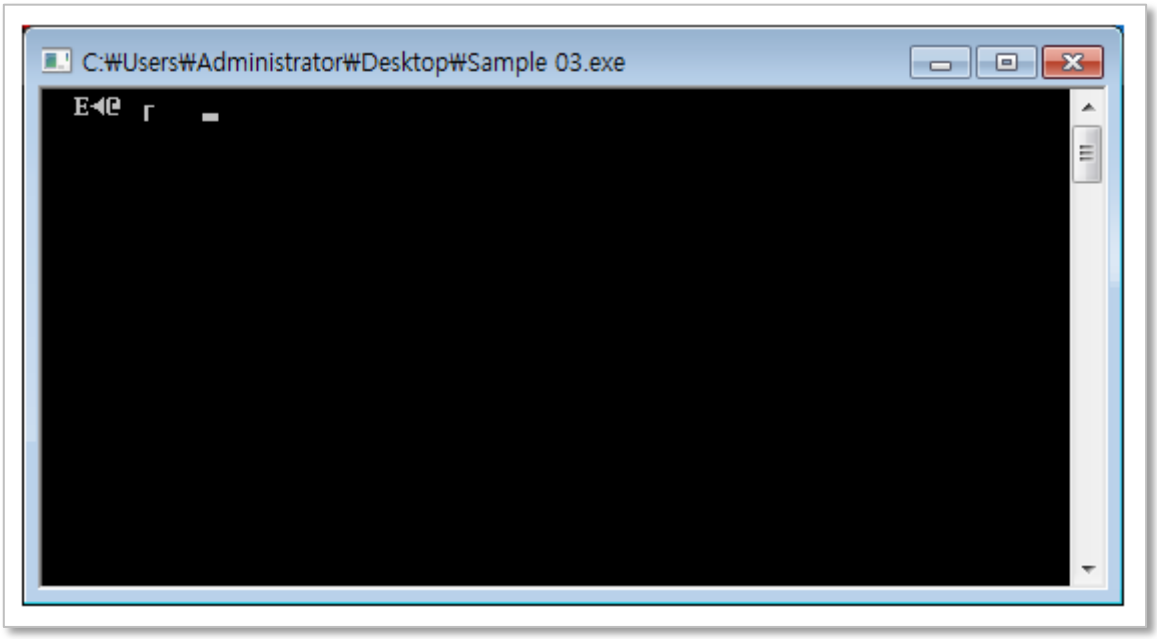

그림 2-1 Sample 03.exe 실행 화면

Sample 03.exe을 분석해서 근본적인 원인을 파악한 뒤에 문자열을 출력하도록 만들겠습니다.

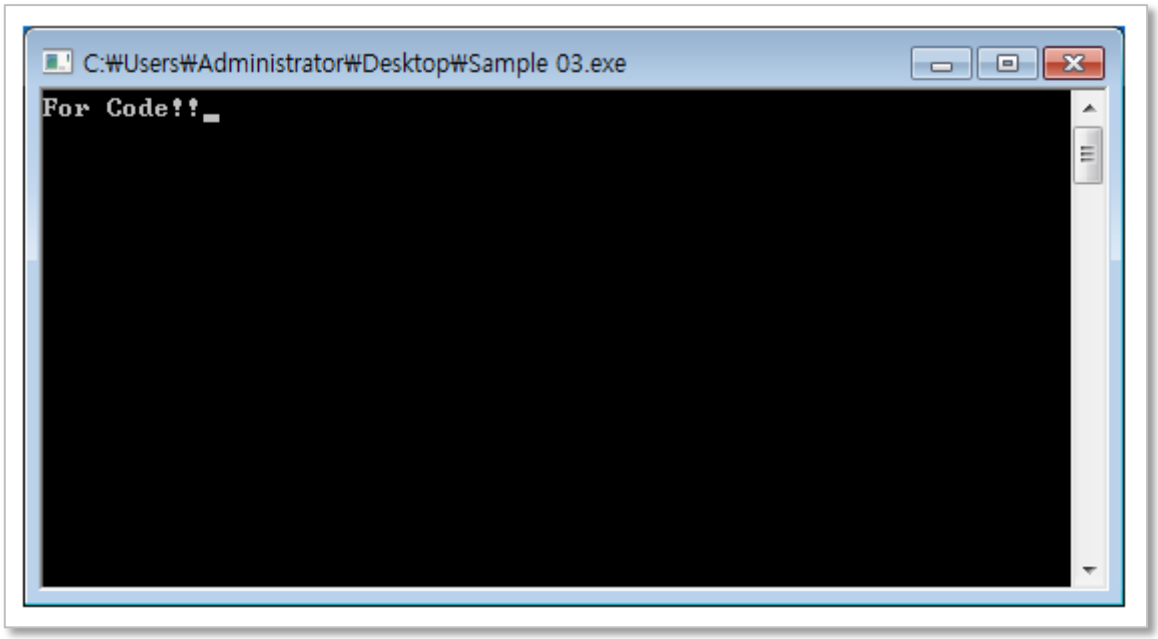

그림 2-2 문제 해결시 출력되는 문자열

2.2. 코드 분석_Level.1 | 흐름 파악하기

Sample 03.exe는 문자열 길이만큼 문자를 반복 출력하도록 작성되어 있습니다.

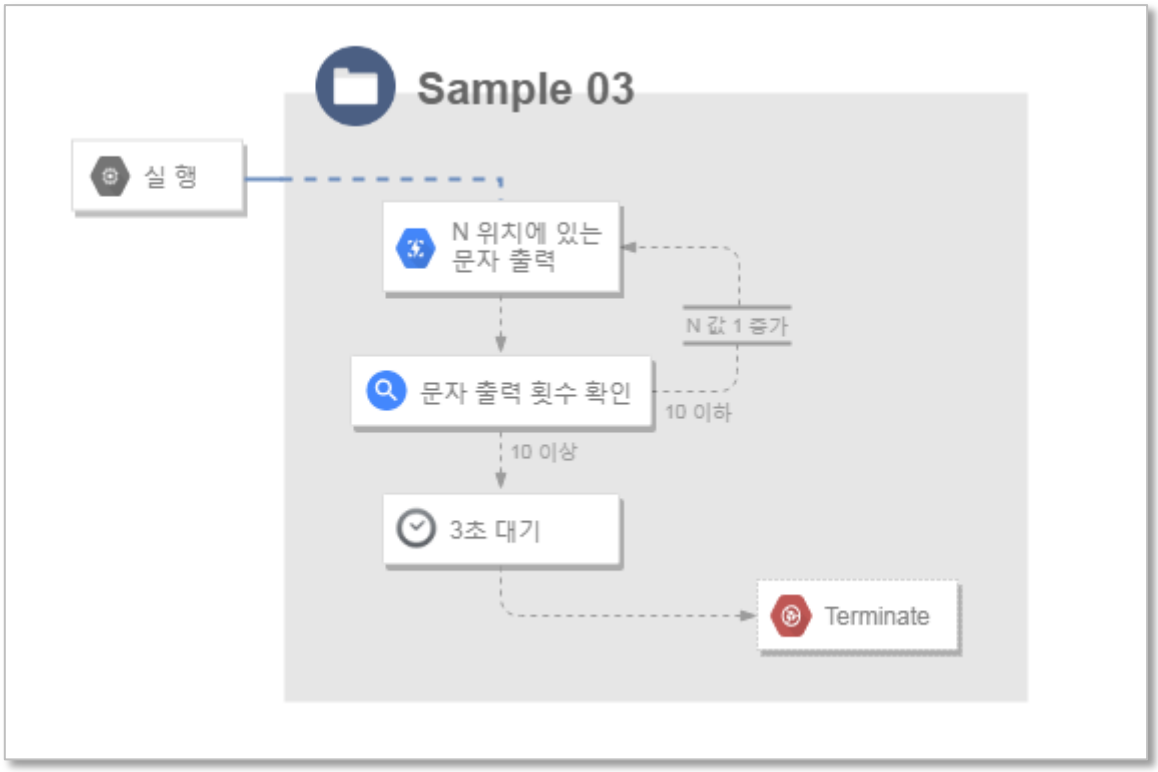

그림 2-3 Sample 02.exe 예상 동작흐름

출력 문자열이 “For Code!!”이기 때문에 문자 출력 카운트는 10이 맞습니다. 아무래도 문자를 출력하는 방식에 문제가 있어 보입니다. 대략 두 가지 가능성이 떠오르네요.

① “For Code!!” 문자열을 특정 값과 연산해서 다른 문자열로 만든 후에 출력한다.
② Sample 03.exe의 .data 섹션에는 “For Code!!” 문자열이 기록되어 있다. 그리고 이 외에도 다양한 데이터가 기록되어 있을 것이다. 문자 출력 주소가 바뀌었다면, “For Code!!” 문자열이 아닌 다른 데이터가 출력될 수 있다.

2.3. 코드 분석_Level.2 | 원인 도출

문제 발생 지점을 찾고 확인하는 시간입니다. 먼저 메인함수를 찾아가겠습니다. [그림 2-4]는 메인함수 시작 지점입니다.

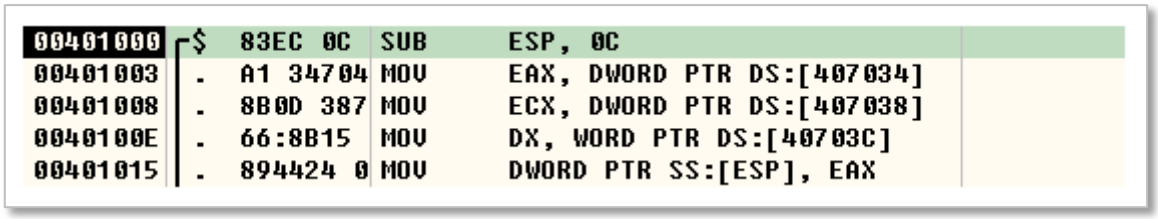

그림 2-4 Sample 02.exe 메인 시작지점

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

2.4. 코드 분석_Level.3 | 문제 해결

※ Note

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

블로그: http://bitly.kr/udWs
페이스북: http://bitly.kr/OrHQ

※ 참고

실제 분석에서는 어셈블리를 한 줄씩 모두 분석하지 않습니다. 중점적으로 봐야 할 부분이 아니라고 판단되면 빠르게 넘깁니다. 그런데 이게 하루 아침에 되진 않겠죠? 다양한 파일을 많이 분석하면서 스스로 체득해야 합니다. 그렇다고 너무 걱정하지 마세요. 같이 분석하면서 자연스럽게 익히게 될 것입니다.

리버싱 이 정도는 알아야지

발행일 | 2018 년 09 월
발행자 | SecurityFactory
페북 주소 | http://bitly.kr/OrHQ
이메일 | itseeyou@naver.com


SECURITY FACTORY
http://securityfactory.tistory.com