|  |
| --- |
|  |
| **악성코드분석** |
| 1. 악성 코드 stuxnet이 사용한 취약점 서술  2. Metamorphic malware로 분류되는 악성 코드의 난독화 방법 종류 |

|  |
| --- |
|  |

**목 차**

[**1. 악성 코드 stuxnet이 사용한 취약점 서술 2**](#_Toc453778794)

[**1.1) 스턱스넷(Stuxnet) 2**](#_Toc453778795)

[**1.2) stuxnet의 공격 과정 3**](#_Toc453778796)

[**1.3) stuxnet이 사용한 취약점 6**](#_Toc453778797)

[**1.3.1) Windows shell LNK 취약점, 바로가기 아이콘 로드로 인한 원격 코드 실행 문제점** 6](#_Toc453778798)

[**1.3.2) Windows Printer Spooler 취약점, 인쇄 스풀러 서비스의 취약점으로 인한 원격 코드 실행 문제점** 7](#_Toc453778799)

[**1.3.3) Windows Server Service 취약점, 서버 서비스의 취약점으로 인한 원격 코드 실행 문제점** 8](#_Toc453778800)

[**1.3.4) Windows kernel-mode Drivers (keyboard layout) 취약점, 권한 상승 문제점** 9](#_Toc453778801)

[**1.3.5) Windows Task Scheduler 취약점, 권한 상승 문제점** 9](#_Toc453778802)

[**1.3.6) Siemens SIMATIC WinCC Default Password Security Bypass 취약점**  10](#_Toc453778803)

[**2. Metamorphic malware로 분류되는 악성 코드의 난독화 방법 종류 11**](#_Toc453778804)

[**2.1) Metamorphic 기법 개요 11**](#_Toc453778805)

[**2.2) Metamorphic malware의 난독화 방법 12**](#_Toc453778806)

[**2.2.1) 데이터 난독화 기법 (Data Obfuscation)** 12](#_Toc453778807)

[**2.2.2) 죽은 코드 삽입 기법 (Dead Code Insert)** 13](#_Toc453778808)

[**2.2.3) 레지스터 재할당 기법 (Register Reassignment)** 14](#_Toc453778809)

[**2.2.4) 흐름 난독화 기법 (Control flow obfuscation)** 15](#_Toc453778810)

# **1. 악성 코드 stuxnet이 사용한 취약점 서술**

## **1.1) 스턱스넷(Stuxnet)**

스턱스넷(Stuxnet)이란 2010년 7월에 발견된 ‘슈퍼 산업 시설 바이러스 웜’ 이다. 스턱스넷이라는 이름은 코드 안에 ‘Stuxnet’으로 시작하는 이름의 파일이 유난히 많아 이 존재하여 붙여지게 되었다. 스턱스넷은 독일 siemens사의 산업 자동화 제어 시스템(PCS 7)을 공격 목표로 주요 산업 시반 시설의 제어 시스템 (SCADA 시스템)에 침투하여 오작동을 유발하여 시스템에 치명적인 손상을 입힐 수 있다. 일반 PC에서는 SCADA 시스템에 접속할 수 없기 때문에 악성코드에 감염만 될 뿐 결과적으로는 오직 특정 산업 자동화 시스템을 공격한다.

또한 스턱스넷은 하나뿐만 아니라 여러 가지의 다양한 취약점을 사용하였다. 이 중에서 알려 지지 않았던 취약점들도 대다수이다.

※ 윈도우 운영체제 취약점

- Windows Shell LNK 취약점

- Windows Server Service 취약점

- Windows Printer Spooler 취약점

- Windows kernel-mode Drivers (keyboard layout) 취약점

- Windows Task Scheduler 취약점

※ SCADA 시스템 취약점

- Siemens SIMATIC WinCC Default Password Security Bypass 취약점

## **1.2) stuxnet의 공격 과정**

스턱스넷은 파일이 여러 개로 구성되며, 알려지지 않은zero-day 취약점들을 이용하여 SCADA 시스템을 제어하는 PC에 핵심 모듈 파일을 제공하는 dropper가 실행되도록 한다. dropper는 정상 ‘s7otbxdx.dll’ 파일의 이름을 변경하여 기존 정보를 백업하고 동일한 이름으로 악성 파일을 생성한다. 그리고 SCADA 시스템을 통합 관리하는 ‘Step7’을 실행하면 기존의 정상적인 파일이 아닌 스턱스넷 악성코드 파일이 실행된다. ‘Step7’은 ‘s7otbxdx.dll’ 파일을 이용하여 제어 PC와 산업 자동화 제어 시스템 간에 Block 파일을 교환하는 것이다. 이 파일을 스턱스넷 악성코드의 DLL로 변경하면 정상적인 동작이 아닌 악성코드를 실행하게 되어 산업 자동화 제어 시스템을 모니터링 하거나 제어가 가능하다.

※ WinCC/Step 7 : STL또는 SCL과 같은 언어로 PLC를 실행/제어/모니터링 할 수 있는 코드를 작성할 수 있는 통합환경의 관리 도구

**<과정>**

① 감염된 PC에서 USB를 통해 메인 PC에 전파

- 스턱스넷은 ‘~WTR4141.tmp’, ‘~WTR4132.tmp’ 2개의 파일을 사용하여 메인 악성코드를 설치하고 최초 실행을 위하여 Window Shell LNK 취약점을 사용한다.

② 감염된 PC에서 C&C 서버로 시스템 정보를 전송

- 스턱스넷은IExplorer.exe 프로세스에 injection되어 감염PC의 OS버전, 감염 시간 등 시스템 정보들을 C&C 서버에게 전송한다. 그리고 C&C 서버의 명령에 따라RPC 서버로 동작하여 감염된 다른 시스템들의 악성 코드 버전 업데이트를 수행한다.

③ 악성코드 유포를 목적으로 내부 네트워크의 다른 시스템 공격

- WinCC database를 이용하여 감염시키거나, Windows printer spooler 취약점 등의 방법을 통하여 내부 네트워크의 다른 시스템들도 감염시킨다.

④ 감염된 메인 PC와 추가 감염된 내부 시스템 사이의 공격 명령 공유

- 감염 시 RPC 서버가 동작하여 네트워크 내의 감염된 다른 시스템들로부터 감염된 버전을 체크하고 버전이 낮은 경우에는 상위 버전의 악성코드를 받아 설치한다.

⑤ 공격자의 공격 명령 생성

- 공격자는 임의의 공격 명령을 생성, C&C 서버에 전송한다.

⑥ 공격 명령 전송

- C&C 서버는 암호화된 바이너리 코드를 받아와 실행한다.

⑦ PLC 제어 명령 생성

- Step7 프로그램은 데이터와 코드 Block들을 MC7 형태의 파일로 컴파일 하여 PLC 장치에 전송하고 받은 Block들은 메모리에 저장한 후 로드 한다.

※ PLC(Programmable Logic Controllers) : 산업 자동 제어 시스템에서 실제 device들을 제어하는 장치

⑧ PLC 제어 명령 변조

- PLC 장치에 공격자의 임의의 명령어를 삽입하기 위해 공격자는Step7 프로그램에서 PLC 장비와 관리 프로그램 간에 데이터 교환을 수행하는 ‘s7otbxdx.dll’ 파일을 이용한다. 공격자는 직접 제작한 악성 DLL 파일을 동일한 파일명(s7otbxsx.dll)으로 생성하여 기존의 것과 교체한다. 변조된 ‘s7otbxdx.dll’ 파일은 원본 DLL 파일의 모든 정보에 임의의 코드가 추가된 형태이기 때문에 업무 수행에 문제가 없어 감염이 되더라도 이상 징후를 쉽게 발견하기가 힘들다.

⑨ 공격 대상 시스템 공격

- 변조된 명령어를 통해 공격 대상이 되는 시스템에 공격을 시도한다.

⑩ TLC 장비 감염

- 제어 장애가 발생한다.

정리하자면, 스턱스넷에 감염된 USB 메모리를 윈도우 운영체제를 사용하는 컴퓨터에 연결하면 자동으로 침투가 된다. 침투가 된 컴퓨터 내에서는 해당 산업 시설을 제어하는 Siemens 사 소프트웨어를 검색하고 발견하지 못했을 경우 해당 컴퓨터의 내부 네트워크(LAN)을 통해 다른 컴퓨터로 침투하여 다시 검색을 수행한다. Siemens 사 소프트웨어를 검색하여 찾아낼 수 있는 경우는 다시 프로그래밍하는 방법으로 해당 시설의 자동 제어 장치를 장악하여 기능을 정지시키거나 오작동을 일으킨다.

## **1.3) stuxnet이 사용한 취약점**

### **1.3.1) Windows shell LNK 취약점, 바로가기 아이콘 로드로 인한 원격 코드 실행 문제점**

스턱스넷은 USB를 이용하여 전파할 때 자동 실행 목록 ’Autorun.inf’ 파일을 이용하기도 하지만, 자동 실행 기능이 비활성화 되어 있는 경우에도 감염을 전파하기 위해 LNK 파일에 대한 Windows shell 아이콘 처리 내부에 존재하는 취약점을 이용하였다.

이미 감염된 USB를 Windows탐색기 및 내 컴퓨터를 통해 장치 드라이브를 열었을 때 USB 내의 파일과 디렉터리를 검색하기 위해 바로 가기 아이콘이 로드되는 순간, 악성 DLL파일이 저장되고 실행됨과 동시에 감염이 되어 지정된 대상 파일이 사용자 권한으로 로드 된다. 또한 사용자가 웹 브라우저(Iexplorer.exe)나 Windows 탐색기와 같은 파일 관리자로 웹 사이트를 방문할 때에도 바로 가기 아이콘을 로드할 때 악성 DLL 파일이 저장되고 실행할 수 있다.

<감염된 USB 내 생성된 디렉터리 목록>

*%DriveLetter%\~WTR4132.tmp (스턱스넷의 메인 DLL이 포함된 파일)*

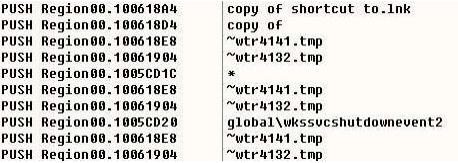
*%DriveLetter%\~WTR4141.tmp (~WTR4132.tmp를 로드하는 파일)*

*%DriveLetter%\Copy of Shortcut to.lnk*

*%DriveLetter%\Copy of Copy of Shortcut to.lnk*

*%DriveLetter%\Copy of Copy of Copy of Shortcut to.lnk*

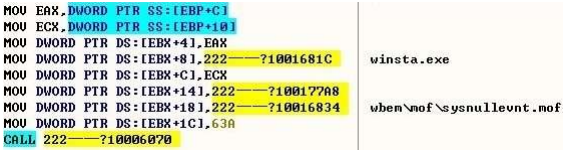
*%DriveLetter%\Copy of Copy of Copy of Copy of Shortcut to.lnk*



**[그림 1-1] USB 내 복사된 파일**

USB 상 모든 LNK 파일은 ~WTR4141.tmp 파일을 가리키고 있고 DLL로 로드 된다. 각각 다른 형태의 LNK 파일은 다양한 운영체제를 지원한다. ~WTR4141.tmp 파일은 LoadLibararyA를 호출하여 드롭퍼를 로드하기 위한 메인 Entrypoint를 실행시켜 스턱스넷을 설치한다.

### **1.3.2) Windows Printer Spooler 취약점, 인쇄 스풀러 서비스의 취약점으로 인한 원격 코드 실행 문제점**

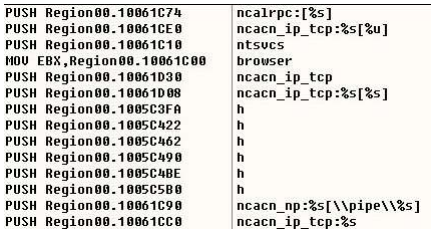


**[그림 1-2] printer spooler 취약점 공격**

스턱스넷은 공격 대상 시스템에 원하는 데이터를 복사하기 위하여 공유된 프린터를 사용하였다. 이 취약점은 시스템에서 파일과 프린터 공유 옵션을 사용해야 하고, 프린터는 반드시 공유되어야 한다는 조건이 있다. 스턱스넷은 Dropper 기능을 가지는 메인 DLL을 ‘winsta.exe’라는 이름으로 대상 시스템의 디렉터리(%SystemRoot%System32\)에 복사한다. 이 때 guest 권한으로 로그인한 사용자는 파일을 쓸 수 있으나 실행하지 못한다. 이 때문에, 스턱스넷은 파일을 실행하기 위해서 어떠한 방법을 사용한다.

그 방법은 ‘winsta.exe’를 실행하는 WMI 코드가 포함되어 있는 WMI BMF(Binary Managed Object Format) 파일인 ‘sysnullevnt.mof’ 파일을 기본 MOF 자체 설치 디렉터리(%SystemRoot%System32\wbem\mof\)에 저장한다. 이곳에 저장된 파일은 자동적으로 컴파일 되고 등록이 된다. 다시 설명하자면 감염된 PC에 사용자가 프린트 작업 요청 시 PC의 버퍼에 있던 악성 코드가 프린터로 이동된다. 여기서 정상적인 PC가 프린터에게 인쇄를 요청했을 때 프린터에 있던 악성 코드가 해당 PC로 복사되는 것이다.

### **1.3.3) Windows Server Service 취약점, 서버 서비스의 취약점으로 인한 원격 코드 실행 문제점**



**[그림 1-3] RPC 취약점 공격 시작**

공격자는 이 취약점을 사용하여 원격에서 권한과 무관하게 네트워크에 있는 다른 감염 대상 PC에게 스턱스넷은 RPC(Remote Procedure Call) 서버를 설치하고 연결을 한다. 그리고 RPC 서버에서 스턱스넷 버전을 보낸다. 만약 클라이언트에 존재하는 버전이 새로운 버전보다 낮다면 RPC 서버는 클라이언트로 DLL 파일을 복사해서 보낸다. 그 후 클라이언트는 새로운 버전을 받고 선택된 프로세스 안에 인젝션을 하여 설치를 시작한다. 이 방식은 고립된 PC에서 자체로 스턱스넷을 업데이트를 행하도록 하나 인터넷이 연결이 되어 있어야 한다. 감염된 모든 파일은 UUID 값이 000204e1-0000-0000-c000-000000000046을 사용하도록 설정이 된다. 이 값을 사용하면 스턱스넷에 감염된 시스템들 간에 P2P 형태로 서로를 확인하고 통신하여 업데이트가 가능하다.

또한 stuxnet은 이 취약점을 사용하는데 있어서 다음의 조건을 만족해야 한다.

*1. 감염 시스템의 날짜가 2030. 1. 1 이전*

*2. 몇 가지 안티 바이러스의 시그니처 버전을 조사하여 날짜가 2009. 1. 1 이전*

*3. Kernel32.dll과 Netapi32.dll이 패치된 날짜가 2008. 10. 12 이전*

### **1.3.4) Windows kernel-mode Drivers (keyboard layout) 취약점, 권한 상승 문제점**

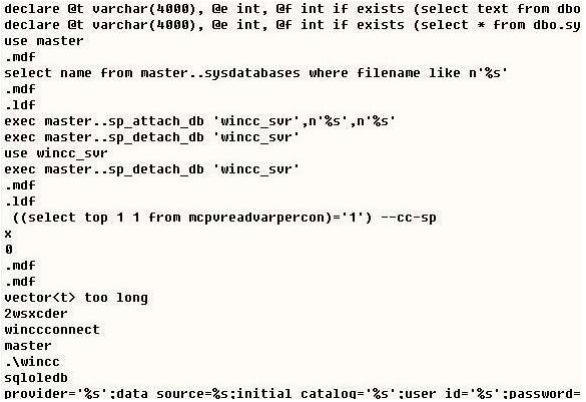
이 취약점은 아래에 설명할 Windows Tash Scheduler 취약점과 같은 맥락으로 권한 상승을 하고 새로운 프로세스(Win32k.sys와 같은 경우에 “csrss.exe”) 및 Task Scheduler case에 있는 새로운 작업을 관리자 권한으로 실행하기 위하여 사용한다. Window 2000/XP 32bit 시스템 환경에 영향을 주며, Windows커널 모드 드라이버가 ‘NtSendUserInput’ 시스템 요청 내에 함수 포인터를 위치시키는데 이용하는 값에 대해 충분히 확인을 하지 않았기 때문에 발생한다. 그 결과, 커널 모드로 임의 코드를 수행할 수 있다. 스턱스넷은 ‘win32k.sys’를 이용하여 메모리 주소에 위치하고 있는 쉘 코드로 점프하여 커널 모드로 실행한다. 그리고 나서 쉘코드는 “csrss.exe”를 구성하고 있는 스턱스넷 메인 DLL을 로드한다.

### **1.3.5) Windows Task Scheduler 취약점, 권한 상승 문제점**

이 취약점은 Windows7과 Windows Server 2008에 영향을 주며, Task Scheduler 서비스에 영향을 준다. CRC32 무결성 검사를 우회할 목적으로, 스키마 XML 내에 알려지지 않은 특정 필드가 COM(Component Object Model) 인터페이스를 통해 조작될 수 있다. Windows Task Scheduler가 실행하는 작업 파일을CRC32 충돌을 일으키도록 수정하면, 공격자는 이 취약성을 이용해 SYSTEM권한으로 임의의 코드를 실행할 수 있다. 공격을 성공할 경우에는 시스템을 장악할 수 있고 실패할 경우는 서비스 거부가 발생된다. 스턱스넷은 앞에서 언급한 것과 같이 상승된 권한으로 “rundll32.exe”을 사용하여 자신을 실행시키기 위해 이것을 사용한다.

### **1.3.6) Siemens SIMATIC WinCC Default Password Security Bypass 취약점**

Siemens의 WinCC와 PCS 7 SCADA 시스템은 로컬 사용자가 back-end 데이터베이스에 접근하고 권한을 얻기 위해 default 암호를 사용한다. 이 취약점을 이용하여 공격자는 SQL데이터베이스와 WinCC가 사용하는데 필요한 Step7 프로젝트 정보를 볼 수 있다.



**[그림 1-4] WinCC database 쿼리문**

그리고 스턱스넷은 쿼리문을 실행하여 Step7소프트웨어를 구성 설정하고 있는DLL 파일들을 변경할 수 있다. 또한, %ALL USERS PROFILE%\sql%05x.dbi. 에 있는 파일에 접근하거나 삭제가 가능하다. 이에 따라 backdoor와 비슷한 맥락으로 .DBI 파일이 데이터베이스 explorer 정보 파일이기 때문에 삭제를 하게 되면 데이터 베이스에서 악성 코드가 수행한 흔적을 지울 수 있다.

# **2. Metamorphic malware로 분류되는 악성 코드의 난독화 방법 종류**

## **2.1) Metamorphic 기법 개요**

Metamorphic 기법이란 두 개의 다른 실행 파일이 동일한 행위를 하지만, 실행 파일에 존재하는 code 영역의 형태를 변형 할 수 있는 기법을 말한다. 위와 같은 기법이 나온 이유는 악성 코드가 백신에 검출 되지 않고 지속적인 행위를 하기 위해서 개발 되었다. 우리가 사용하는 안티 바이러스 제품의 기존 악성코드 탐지 방법은 매번 파일의 행위를 검사하는 방식이 아닌 악성코드에서 악성 행위를 하는 코드의 특징이나 함수, Data, 실행 파일의 해시 값 등을 추출한다. 이러한 정보들로 database를 만든 후, 파일을 검사하는 방식으로 악성코드를 탐지한다. 수행하는 역할이 같은 코드를 다르게 표현하고, 코드의 배치를 다르게 구성했지만 실행 순서는 같게 하는 등 특징을 추출 하는 것이 가능한 부분을 변경하게 될 경우에는 기존의 탐지 방안은 한계를 가지게 된다. 즉, 어느 정도 악성코드의 생존 가능성과 시간을 증가 시켜서 원하는 활동을 할 수 있게 되는 것이다.

|  |  |
| --- | --- |
| **Metamorphic 기법** | **설명** |
| Data Obfuscation | 해당 명령어의 동일한 값을 대체 |
| Dead Code Insert | 연산에 상관없는 Dummy 코드 삽입 |
| Register Reassignment | 기존 레지스터를 다른 레지스터로 변경 |
| Control flow obfuscation | 순서에 상관없는 명령어들의 배치를 변경 |

**[표 1] Metamorphic 기법 요약**

## **2.2) Metamorphic malware의 난독화 방법**

### **2.2.1) 데이터 난독화 기법 (Data Obfuscation)**

데이터 난독화 기법은 동일한 명령을 수행 할 수 있는 명령어로 기존의 코드를 변경 하는 것을 말한다. 이를 통해서 같은 역할을 수행 하지만, 코드 영역의 코드가 달라 지기 때문에, 탐지를 우회 할 수 있다. 예를 들어 0을 표현 하기 위한 방법으로 “1-1=0”의 표현을 “상수 미분”과 같은 다른 방식으로 표현 하는 것이다.

**[그림 2-2] 데이터 난독화 기법 2**

mov dl, [eax+1]

mov cl, dl

cmp dl, [esi+1]

jnz short loc\_40110E

add eax, 2

add esi, 2

**xor ebx,ebx**

test cl, cl

**[그림 2-1] 데이터 난독화 기법 1**

mov dl, [eax+1]

mov cl, dl

cmp dl, [esi+1]

jnz short loc\_40110E

add eax, 2

add esi, 2

mov ebx, 0

test cl, cl

[그림-1]

[그림 2-1]의 mov ebx,0의 코드가 실행 되면 ebx에 0이 들어 갈 것이다. 그리고 해당 코드를 [그림 2-2]의 xor ebx,ebx로 변환을 하면 mov ebx,0의 결과 값과 같게 ebx에 0이 들어가게 된다. 반대의 경우에도 가능하다. 이와 같은 기법은 시그니처 기반의 탐지를 피할 수 있다.

※ XOR 명령어 : operand1, operand2 두 값의 각 bit가 XOR 연산을 수행한다.

같은 값 ebx를 XOR 연산으로 수행하기 때문에 결과 값이 0이 된다.

### **2.2.2) 죽은 코드 삽입 기법 (Dead Code Insert)**

죽은 코드 삽입 기법은 연산 과정과 상관없는 쓰레기 코드(Dummy)를 삽입하는 기법이다. 해당 Dummy 코드를 이용하면 특징을 추출한 검사 방법을 우회 할 수 있다. 뿐만 아니라 Dummy 코드를 다량으로 삽입하게 되면 분석가의 분석을 방해 할 수 있다.

다만, 해당 기법은 기존의 실행 시간에 있어서 차이가 있다.

mov dl, [eax+1]

mov cl, dl

add eax, 0

cmp dl, [esi+1]

jnz short loc\_40110E

or esi, 0

add eax, 2

add esi, 2

mov esi, esi

mov ebx, 0

and ebx, -1

test cl, cl

mov dl, [eax+1]

mov cl, dl

cmp dl, [esi+1]

jnz short loc\_40110E

add eax, 2

add esi, 2

mov ebx, 0

test cl, cl

**[그림 2-3] Dummy 코드 삽입 기법 1**

**[그림 2-4] Dummy 코드 삽입 기법 2**

|  |  |
| --- | --- |
|  |  |

위에 있는 [그림 2-3]를 수행한 결과 값과, [그림 2-4]을 수행한 결과 값은 동일 하다.

[그림 4]에 추가된 빨간색 코드들을 확인해 보면 명령을 수행하지만 전혀 의미 없다는 것을 알 수 있다.

\*add eax, 0 → eax = eax + 0

or esi, 0 → esi = esi | 0

mov esi, esi → esi = esi

and ebx, -1 → ebx = ebx & -1(0xffffffff)

### **2.2.3) 레지스터 재할당 기법 (Register Reassignment)**

레지스터 재할당 기법은 기존에 사용되던 레지스터를 다른 레지스터로 변경 하는 기법이다. 해당 기법 같은 겨우 모든 코드 영역에 적용 하는 것은 어려움이 따른다. 하지만 탐지의 특징이 될 수 있는 부분에 적용하면 효과적인 기법이다. 그리고 기존 레지스터의 life time 동안 변경될 레지스터가 사용되지 않아야 한다.

mov ebx, eax

mov dl, [eax+1]

mov cl, dl

cmp dl, [esi+1]

jnz short loc\_40110E

add eax, 2

add esi, 2

mov ebx, 0

test cl, cl

leave

ret

mov ebx, eax

push ebx

mov dl, [ebx+1]

mov cl, dl

cmp dl, [esi+1]

jnz short loc\_40110E

add ebx, 2

add esi, 2

mov ebx, 0

test cl, cl

pop ebx

leave

ret

**[그림 2-5] 레지스터 재할당 기법 1**

**[그림 2-6] 레지스터 재할당 기법 2**

|  |
| --- |
|  |
|  |

예를 들어 [그림 2-5] 의 표시된 부분이 변형해야 할 부분이라고 가정한다. 그리고 [그림 2-6]과 eax 레지스터가 사용된 코드를 전부 ebx 레지스터로 변경한 것이다. 이러한 경우에는 eax 레지스터로 특징을 잡았을 경우에 해당 루틴은 검출 되지 않게 된다.

### **2.2.4) 흐름 난독화 기법 (Control flow obfuscation)**

흐름 난독화 기법은 실행되는 특정 영역에서 명령어들의 순서는 같다. 다시 말해, 루틴의 실행 결과는 동일하다. 하지만, 명령어들의 배치는 변경되어 기존에서 특징을 잡았던 순서를 무력화 시킬 수 있다. 이러한 방법에는 간단한 jmp 명령어를 이용하는 방법도 존재한다. 뿐만 아니라 [그림 2-8]에서 보여 주듯이, stack 저장과 retn 명령어를 사용하여서 흐름을 변경 시킬 수 있다.

1

mov dl, [eax+1]

mov cl, dl

cmp dl, [esi+1]

mov eax,eax

add eax, 2

add esi, 2

test cl, cl

mov edx, Lable\_1

mov ebx, Lable\_2

mov edi, Lable\_3

push edi

push ebx

push edx

retn

Label\_2:

cmp dl, [esi+1]

mov eax,eax

add eax, 2

retn

Label\_1:

mov dl, [eax+1]

mov cl, dl

retn

Label\_3:

add esi, 2

test cl, cl

2

3

2

**[그림 2-7] 흐름 난독화 기법 1**

1

3

**[그림 2-8] 흐름 난독화 기법 2**

[그림 2-7]을 보면 코드의 배치와 실행 순서가 1,2,3으로 같은 것을 알 수 있다. 하지만 [그림 2-8]의 코드 배치는 2,1,3 순성 이지만 실행 순서는 1,2,3으로 진행되고 [그림 2-7]과 흐름이 같음을 알 수 있다.