코드 수정을 통한 스크립트 형태의 악성 이동 코드 대응 기법*

윤영태, 예홍진, 조은선, 고재영
아주대학교 정보통신전문대학원
anmerong@ajou.ac.kr

Code Rewriting Technique to Detect Script-based Malicious Mobile Codes

Yung-tae Yoon, Hong-Jin Yeh, Eun-Sun Cho, Jae-Young Koh
Graduate School of Information and Communication, Ajou University

요 약

현재의 스크립트 형태의 악성코드에 대한 대응 기법으로는 자바의 샌드박스(Sandbox) 모델과 흡사한 자원 제한 기법이 주로 사용된다. 하지만 이 기법의 경우 악의가 없는 스크립트 또한 제한되어 있는 자원을 사용할 수 없으며 이로 인하여 스크립트 본연의 취지인 전송된 스크립트를 통한 편의성 제공의 기능이 많은 제약을 받고 있다. 따라서 각각의 스크립트 코드의 유해 여부를 코드 수행 중에 결정하여 자원 사용의 허용 여부를 결정하는 기법이 절실이 요구되며 본 논문에서는 전송되어온 코드의 위험 요소 부위를 자체 면역성을 가지도록 코드를 수정하는 기법을 제안하고자 한다.

1. 서론

웹 페이지나 이-메일에 삽입되어 있는 자바스크립트나 비주얼베이직 스크립트는 웹 서버나 메일 서버로부터 브라우저나 아웃룩(Outlook)로 전송된 후 클라이언트 측의 CPU time, Memory등의 자원을 사용하여 해석·실행된다. 이를 통하여 전송된 스크립트는 사용자의 마우스 클릭이나 또는 키보드 입력 등에 대하여 즉각적으로 반응할 뿐 아니라 입력된 데이터의 에러 체크 등의 기능을 클라이언트에서 수행해 줌으로써 네트워크 트래픽과 서버 측의 자원 소모를 줄여주는 등 많은 이점을 제공한다.

하지만 클라이언트 측의 자원을 사용하여 동작한다는 특징으로 인하여 일부에서는 서비스 거부공격이나 시스템 변경 등의 목적으로 사용될 수 있다는 문제점을 안고 있으며 이에 대한 대책으로서 자바의 샌드박스와 같은 자원 제한 기법, 악성 코드내의 특정 시그니쳐를 바탕으로 악성코드를 필터링하는 패턴 매칭 기법이 주로 사용되고 있다. 그러나 자원 제한 기법의 경우 악성이 아닌 코드에 대하여서도 일률적으로 자원의 사용이 제한된다는 문제점이 있으며 두 번째 기법의 경우 이미 알려진 악성 코드에만 적용할 수 있다는 문제점이 있다.

이러한 문제점을 해결하기 위하여 런타임에 악성 여부를 결정할 수 있는 메커니즘이 필요하다. 본 논문에서는 전송된 스크립트를 자체 보안성을 가지도록 코드를 수정한 후 별도의 모니터링 모듈과 연계하여 동작하는 방식을 제안하고자 한다.

2. 악성 스크립트 언어의 특징과 기본 아이디어

요즈음 일반적으로 많이 사용되고 있는 스크립트 언어에는 자바스크립트(JavaScript)와 비주얼베이직스크립트(VBScript)가 있다. 이들 두 스크립트 언어는 언어 자체적인 특성만을 봤을 때에는 윈도우 시스템의 레지시트리(Registry)나 파일 시스템 수정, 아웃룩(Outlook)을 통한 이-메일 전송 등의 일들을 할 수 없다.

하지만 마이크로 소프트는 사용자들에게 편의성을 제공하기 위하여 스크립트를 사용하여 COM 오브젝트나 Active X control등의 자원을 사용할 수 있도록 하였으며 이를 통하여 스크립트로 작성된 프로그램은 위에 언급된 레지스트리나 파일 시스템 변경, 아웃룩(Outlook)을 통한 메일 전송[3] 등의 기능들 외에도 많은 강력한 기능들을 수행하게 되었다.

2.1장의 스크립트 언어의 특징에서 언급했듯이 스크립트 언어로 작성된 프로그램이 시스템 변조나 이-메일 전

* 본 연구는 한국전자통신연구원 과제의 지원으로 수행되었음

송 등의 기능을 수행하기 위해서는 스크립트 언어 자체의 능력만으로 불가능하며 따라서 COM이나 Active X Control과 연계되어야만 한다. [2]

예제 1: 멜리사 바이러스(Melissa virus)

```
Dim UngaDasOutlook, DasMapiName, BreakUmOffASlice
Set UngaDasOutlook = CreateObject("Outlook.Application")
Set DasMapiName = UngaDasOutlook.GetNameSpace("MAPI")
```

그림 1 Melissa virus에서 Outlook 객체 생성 부분

```
        Set BreakUmOffASlice = UngaDasOutlook.CreateItem(0)
        For oo = 1 To AddyBook.AddressEntries.Count
            Peep = AddyBook.AddressEntries(x)
            BreakUmOffASlice.Recipients.Add Peep
            x = x + 1
            If x > 50 Then oo = AddyBook.AddressEntries.Count
        Next oo
        BreakUmOffASlice.Subject = "Important Message From "
        BreakUmOffASlice.Body = "Here is that document you a
        BreakUmOffASlice.Attachments.Add ActiveDocument.Full
        BreakUmOffASlice.Send
        Peep = ""
```

그림 2 객체를 이용하여 메일을 전송하는 부분

멜리사 바이러스는 사용자의 주소록을 검색하여 받은 편지함에서 랜덤하게 하나의 메일을 선택하여 자기 자신을 첨부하여 보내는 형태의 바이러스로 99년 엄청난 피해를 야기하였으며, 이때 사용하였던 기법이 스크립트 내에서 Outlook객체를 이용하여 메일을 전송하는 방법을 취하였다.[3]

예제 2: 바라쿠다 바이러스(VBS.Baracuda)

```
Set fso = CreateObject("Scripting.FileSystemObject")
Set file = fso.OpenTextFile(WScript.ScriptFullName,1)
mecopy = file.ReadAll

Set Windir = fso.GetSpecialFolder(0)
Set Sysdir = fso.GetSpecialFolder(1)
Set cf = fso.GetFile(WScript.ScriptFullName)
```

그림 3 FileSystemOjbect를 생성하는 부분

위의 예제의 경우 FileSystemObject를 이용하여 특정 디렉토리를 얻어오고 있는 경우이다.

정리하면, 스크립트 형태의 악성 코드에서 주로 사용되는 COM object들은 다음과 같다고 밝혀진 바 있다.[2]

- Scripting.FileSystemObject : 파일 시스템에 직접 접근하기 위해 사용하는 오브젝트로서 파일의 생성, 삭제, 수정, 파일 위치 파악 등 파일과 관련된 다양한 기능을 수행한다.
- WScript.Shell : 레지스트리의 생성, 삭제, 수정 등의 기능을 수행할 목적으로 사용된다.
- WScript.Network : 이-메일 등의 매체 없이 직접 네트워크를 통하여 증식할 목적으로 사용된다.
- Outlook.Application : 아웃룩이 설치되어 있을 경우 존재하는 오브젝트로서 주로 악성코드의 증식을 위하여 메일을 생성, 발신할 목적으로 사용된다.

3. 제안하는 대응 기법

3.1 작동 원리

스크립트로서 구현할 수 있는 이외의 기능을 구현하기 위해서는 2장에서 언급했듯이 COM 오브젝트나 Active X Control을 사용하여야만 한다. 따라서 COM 오브젝트나 Active X 컨트롤을 사용하는 부분을 보안 검증 모듈을 거치도록 수정한 후 일반 실행 루틴을 돌도록 처리하게 되면 스크립트는 외부의 모니터링 도구의 도움 없이 자체적인 보안 검증 기능을 갖추게 된다.

이를 구조적으로 나타내면 그림 4와 같다.

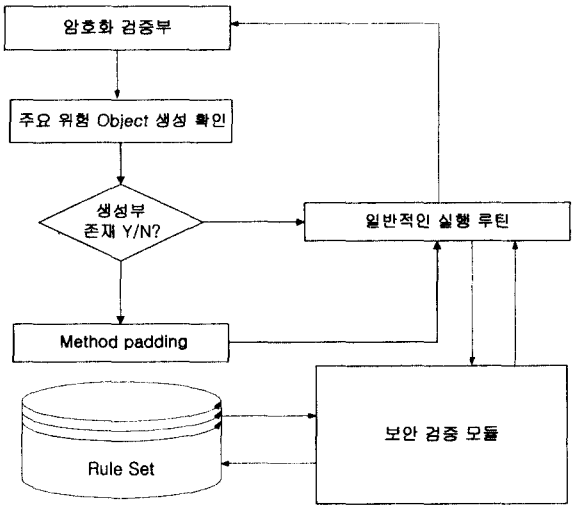

그림 4 본 대응 기법 개념도

스크립트가 실행되게 되면 우선 암호화 검증부를 거치게 되어 일단 암호화가 풀리게 된다. 이렇게 암호화가 풀리거나 또는 암호화되지 않은 코드는 주요 위험 Object및 Method의 존재 여부를 확인하게 되며 만약 존재하게 되면 보안 검증 모듈과 통신할 수 있는 Padding code가 삽입되게 되어 일반적인 실행루틴을 실행하게 된다.

3.2 탐지 과정의 예

멜리사 바이러스의 경우 아웃룩 객체를 이용하여서 자기 자신을 증식 시키는 방법을 취하고 있다. 따라서 Outlook.Application 객체 생성 부분, 주소록 검색 부분, 메일 생성 부분, 메일 발송 부분의 Method를 그림 5와 같이 Padding처리하면 Padding된 코드를 통하여 스크립트 작동 시 보안 검증 모듈에서 스크립트의 작동 형태를 추적할 수 있다. 또한 Rule set 데이터 베이스에 멜리사와 같이 메일을 생성·발송하는 형태의 Rule set이 존재하고 있다면 멜리사 바이러스는 악성인 것으로 판정 내려지며 그 실행이 정지되게 된다.

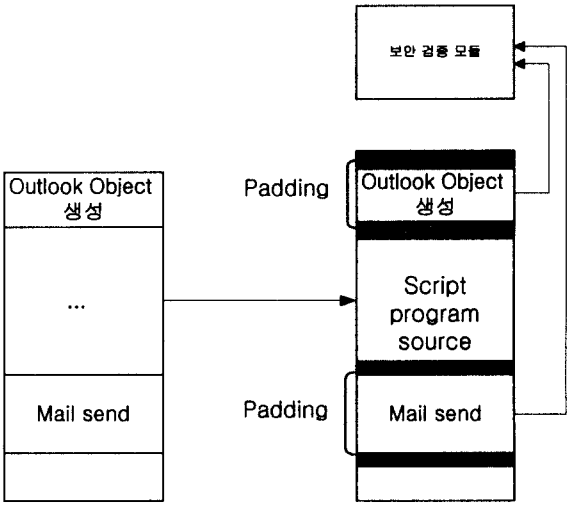

그림 5 위험 Method Padding

4. 결론 및 향후 과제

. 본 논문에서 제안하고 있는 방식은 일반적인 시그니쳐 기반의 탐지 기법과 달리 알려지지 않은 악성의 진위를 런타임에 행위에 기반하여 판단하므로 알려지지 않은 악성 이동 코드에 대하여도 사용할 수 있다. 또한 악성 여부의 판단이 각각의 서버에서 독자적으로 이루어지므로 서버 단위의 보안 메커니즘과 쉽게 연계시킬 수 있다. 뿐만 아니라 수동적으로 코드의 작동 형태를 외부에서 보고 일반적인 모니터링 기법과 달리 면역성을 가지도록 코드를 수정 후 실행함으로 코드의 실행 형태를 보다 세밀하게 관찰할 수 있다.

향후 과제로서는 본 논문에서 제시하고 있는 기법을 스크립트만이 아닌 바이너리 형태의 악성 코드로 확대하는 것이며 또한 보안 검증 모듈의 효율적인 구현 방안을 강구해 보고자 한다

6. 참고 문헌

[1] Flexible Policy-Directed Code Safety, David Evans, 1999

[2] Script-based mobile threats, VIRUS BULLETIN CONFERENCE, 335-354, 2000

[3] Melissa virus source code

[4] VBS.Love letter virus source code

[5] I_worm.WinXP.vbs source code

[6] VBS.Baracuda source code

[6] Mail Filter, http://www.trans-cosmos.co.jp/prod_f/sendmail/switch21demo/helplets/en/MAIL_FILTER.html

[7] Java security architecture (JDK 1.2). L. Gong. Technical report, JavaSoft, July 1997

[8] The Saft-Tcl Security Model. Sun Micro-systems Laboratories, 1997

[9] MFC: A Malicious Code Filter, Raymond W. Lo, 1994