정적 분석 기법을 이용한 악성 스크립트 탐지

배병우, 이성욱, 조은선, 홍만표

아주대학교, 정보통신전문대학원

Malicious Script Detection By Static Analysis

Byung-woo Bae, Seong-uck Lee, Eun-sun Cho, Manpyo Hong

Graduate school of Information and Communication Ajou Univ.

요 약

본 논문은 현재 컴퓨터 사용자들에게 많은 피해를 입히고 있는 악성 스크립트 코드에 대한 탐지기법을 제시하고자 한다. 스크립트 언어는 타 언어에 비해서 단순하며, 상위 수준의 언어로 작성된 소스를 직접 분석가능하기 때문에 기존의 이진 파일 형태의 바이러스 비해 정적 분석 기법 적용이 용이하다. 제안하는 탐지 기법은 기존의 스코어링 방식을 기반으로 한 패턴 매칭과는 달리 스크립트가 수행하는 악성 행위의 분석을 통해 행위 패턴을 생성하고, 이 패턴들을 정적 분석 기법을 통해 패턴간의 관계 분석을 통해 보다 확실한 악성 행위를 탐지하여 스크립트에 포함된 악성행위들을 보고한다. 기존 대부분의 바이러스 탐지 도구들은 이미 알려진 바이러스들만을 탐지 할 수 있다. 정적 분석 기법을 이용한 악성 스크립트 탐지 방법은 악성 행위 별 패턴 존재 여부를 판단하므로 이미 알려진 바이러스는 물론 알려지지 않은 바이러스를 탐지 할 수 있는 방안을 제시한다.

I. 서론

스크립트 바이러스는 스크립트 언어로 제작된 바이러스를 말하며, 윈도우가 일반적인 운영체제로 사용되면서부터 스크립트 바이러스가 크게 증가하고 있다. 최초의 스크립트 바이러스는 1994년 12월에 최초로 발견되었으며, 실제로 문제가 되기 시작한 것은 95년 중반부터이다. 현재 매크로 바이러스와 스크립트 바이러스가 사용자를 괴롭히는 최대 바이러스 중 하나이며, 많은 피해를 주는 악성 스크립트 코드들은 인터넷 웜의 형태로 메일이나 IRC 같은 매체를 통해 전파되고 있다. [1]

악성 스크립트를 구현하기 위해 가장 많이 사용되는 스크립트 언어는 VBS(Visual Basic Script)이다. 또한 VBS로 구현된 악성 스크립트를 자동으로 생성해 주는 웜 생성기마저 인터넷을 통해서 배포되고 있어 스크립트 언어 자체를 모르는 일반 사용자들도 악성 코드를 생성하여 배포 할 수 있다.

본 연구는 악성 스크립트 중 가장 많은 수를 차지하는 VBS로 구현된 악성 스크립트 코드를 정적 분석 기법을 통해 악성 행위 수준별로 분석하여 기존의 백신 프로그램에서 탐지 해내지 못하는 악성 스크립트를 탐지할 수 있는 방법을 제시한다.

1. 악성 코드 탐지 기법

악성 코드에 대한 일반적인 대응 방법은 해당 컴퓨터 악성 코드를 진단 및 치료 할 수 있는 안티 바이러스 프로그램을 이용한다. 이 방법은 바이러스가 발견된 후 바이러스에 대한 정확한 분석을 수행하고 이를 치료할 수 있도록 업데이트하기까지 어느 정도의 시간이 소요되며, 사용자는 그 전까지 아무런 대책도 없이 계속 피해를 입을 수 밖에 없다.

안티 바이러스 프로그램이 바이러스를 탐지하기 위해서 이용하는 방법은 바이러스의 시그니처(Signature)를 이용한 패턴 비교 방법이다. 시그니처란 특정 바이러스의 전체 부분 중 다른 코드에

는 나타나지 않으며 해당 바이러스만 지니고 있는 최소 크기의 패턴을 추출한 것이다. 임의의 파일이 바이러스에 감염되었는지 여부는 시그니처가 해당 파일에 나타나는지 비교를 통하여 이루어진다. 따라서 특정 바이러스의 출현 후 그 바이러스의 특정 패턴인 시그니처를 확보한 후에야 제작 가능한 것이 안티 바이러스 프로그램이다.

이렇듯 신종 바이러스가 출현 한 후 대응책이 만들어지기까지 발생하는 피해에 대해서는 속수무책이며 게다가 기하급수적으로 급증하고 있는 바이러스들을 감당하기 어려워지고 있다. 더군다나 바이러스 제작자들의 기술 또한 지능화되고 있어 백신 개발까지 더욱 많은 시간이 요구되고 있고 특히 동시에 수많은 바이러스들이 출현할 경우 그 피해 규모는 엄청날 것이다. 이런 단점을 보완하고, 새로운 악성 코드에 유연한 대처 능력을 확보하기 위해 면역 시스템, 신경망, 기타 휴리스틱 알고리즘을 이용한 알려지지 않은 악성 코드에 대한 탐지 기법이 연구되었다.

• 면역 시스템

면역 시스템은 새로운 바이러스가 침입할 경우 이에 대한 대응책을 생성하여 주변 네트웍에 전파시켜 전체 시스템에 바이러스에 대한 대응책을 제공하는 시스템이다.

면역시스템에서는 기존의 바이러스와 유사한 바이러스를 감지하기 위하여 근사적 패턴매칭(approximate pattern matching)을 이용하며, 알려지지 않은 침입자의 인식을 위해서는 컴퓨터 시스템의 비정상적인 행위를 감지하는 휴리스틱을 이용한다.

새로운 대응책 생성은 의심스러운 패턴이 발견될 경우 침입자에 의한 것인지 아닌지 판단하기 위하여 모종의 테스트 과정을 수행하여 알려지지 않은 바이러스가 침입했다는 것을 판단하며, 이후 테스트된 결과를 토대로 새로운 바이러스에 대한 패턴 정보를 생성한다. 이렇게 추출된 새로운 바이러스의 패턴 정보를 네트워크상에 연결된 인접한 컴퓨터에 알려줌으로서, 인접한 컴퓨터의 바이러스 패턴 정보 리스트를 갱신하고 다시 인접한 컴퓨터에 알려줌으로서 새로운 바이러스에 대해 대비할 수 있도록 한다. [2]

• 신경망

휴리스틱(heuristic) 바이러스 탐지 방법에서 가장 중요한 문제점은 바이러스 탐지 결과의 긍정 오류(False-positive)및 부정 오류(False-negative)의 발생 문제이다. 휴리스틱 알고리즘에 의하여 새로운 바이러스에 대해 생성된 바이러스 패턴을 많은 수의 바이러스 탐지 사용하면 오류가 발생할 확률이 높아진다. 이를 해결하기 위해 생성된 바이러스 패턴을 신경망을 통해 재 생성한다. 신경망은 다양한 휴리스틱 알고리즘에 의해 생성된 바이러스 패턴들을 입력으로 받아 학습을 통해 오류 발생률이 낮은 새로운 바이러스 패턴을 생성한다. [4]

• 기타 휴리스틱 탐지 방법

그 외 악성 코드 탐지 기법으로는 코드 분석을 통해서 악성 코드를 탐지하는 방법, 메일에 첨부되는 파일명을 분석하는 방법, 네트웍 흐름을 분석하는 방법 등이 있다. 코드 분석 방법은 복제 코드 탐지, 부가 기록 탐지, 자체 암호화 코드 탐지 등의 패턴을 찾아 각 패턴에 스코어링을 하여 전체 스코어링 결과를 토대로 악성 여부를 판단하는 방법이다. 코드 분석기법은 효율성이 좋으나, 각 악성 코드 파일이 이진 파일인지, 스크립트 파일인지에 따라 각각 다른 분석 방법이 필요하다. [5]

메일에 첨부되어 전파되는 악성 코드 탐지 기법으로는 첨부 파일의 파일명과 파일 타입을 분석하는 방법이 있다. 이 방법은 메일에 첨부되는 대부분의 악성 코드의 파일 이름이 복잡한 형태로 되어있어 실행되지 않는 데이터 형태의 파일인 것처럼 가장한다는 것에 주안점을 두어 악성여부를 판단한다.[5]

광범위한 규모의 악성 코드 탐지 기법으로는 네트웍 흐름을 분석하는 것으로 스팸 메일을 탐지하고 방지하는 기술과 유사하다. 이 기술은 메일을 통해 전파되는 새로운 바이러스를 탐지하는 방법으로 아주 효과적인 방법으로 인식된다. [5]

기존의 휴리스틱을 이용한 알려지지 않은 악성 코드 탐지 기술은 아직 긍정 오류가 많이 발생하며, 본 논문과 가장 유사한 형태인 스코어링 방식의 스크립트 코드 분석은 특정 패턴들의 빈도 수를 토대로 악성 여부를 판단하므로 패턴간의 관계가 결여되어 긍정 오류가 발생한다.

II. 정적 분석 기법을 이용한 악성행위 탐지

정적 분석 기법을 이용하여 새로운 악성 코드 탐지하기 위해서는 우선 기존의 악성 코드의 행동들을 분석하여 일반적인 악성 스크립트들이 갖는 행위들을 패턴으로 정의해야한다.

생성된 행위 패턴을 사용한 후 임의의 스크립트 파일들에 대한 악성 행위 존재 여부를 탐지는 스크립트 소스코드 수준에서 스캔을 통해 진행되며, 탐지된 행위 패턴들의 관계를 분석하여 악성 행위 존재 여부를 판단한다. 이 작업은 임의의 주어진 악성 스크립트에 대해 시뮬레이션 작업 없이 단지 스캔에 의해서만 이룰 수 있어 시뮬레이션에 따르는 오버헤드를 수반하지 않는 장점을 지닌다.

1. 스크립트의 악성 행위 분류

윈도우에서 제공하는 스크립트 자체는 파일 시스템이나, 레지스트리, 메일에 영향을 줄 수 없다. 하지만, 스크립트가 COM 객체와 ActiveX 객체를 사용할 수 있어 이러한 객체들을 사용해 시스템에 피해를 준다. 스크립트가 사용하는 피해를 줄 수 있는 대표적인 객체는 다음과 같다.

객체	특징
Scripting.filesystem	파일 입출력관련
WScript.Shell	윈도우 시스템정보
WScript.Network	네트웍 드라이브
Outlook.application	아웃룩 익스프레스관련

표 1 : 악성 코드에 사용되는 객체

악성 코드들은 위의 객체들을 사용하여 로컬 파일 시스템에 코드 복제 및 로컬 시스템 파일 수정, 윈도우 레지스트리 수정, 네트웍을 통한 코드 복제 및 정보 획득, 메일을 사용하여 코드 전파 등의 행위를 한다. 대부분의 악성 코드에 포함되는 악성 행위를 분류하면 표 2와 같다.

악성 행위	대상
자가 복제	로컬 시스템
	메일
	IRC 프로그램
시스템 정보 변경	레지스트리 변경
파일 수정	데이터 파일
	어플리케이션 설정 파일

표 2 : 악성행위 분류

자가 복제를 통한 악성 코드의 확산은 아웃룩의 주소록을 참조하여 메일에 자신의 파일을 첨부하여 전파시키는 방법과, 인터넷 채팅 프로그램(IRC)에서 제공하는 자동 스크립트 파일을 수정하여 인터넷 채팅 프로그램 사용 시 타 사용자들에게 전파시키는 방법을 사용한다. 시스템 정보 변경은 주로 레지스트리를 변경하여 악성 코드가 특정 시점에 자동으로 수행하게 하여 사용자가 인식하지 못하는 사이에 악성 코드를 실행시킨다. [3]

2. 악성 행위 패턴 매칭

스크립트에서 악성 행위를 실행하기 위해서는 특정 객체를 사용하여야만 한다. 때문에 스크립트가 표현 가능한 악성 행위의 패턴이 단순한 형태로 나타난다. 이러한 특징을 이용하여 각 악성 행위별 패턴을 분석하여 특정 형식으로 정의하고 이러한 패턴을 대상 스크립트에서 탐지한 후, 스코어링 방식의 패턴 매칭과는 달리 각 패턴들의 관계를 분석하여 정확한 악성 행위를 탐지한다.

악성 행위 패턴의 정의는 스크립트 소스 코드에서 행위 패턴을 탐지하는 패턴 매치(pattern match)부와 매치된 패턴간의 관계를 분석하여 적확한 행위를 탐지하는 관계 분석 부로 구분된다. 아래의 BNF에서 <Match_Rule>이 패턴 매치 정의 부분이며, <Relation_Rule>이 행위 패턴의 관계 분석 규칙 정의 부분이다.

```
<Match_Rule> ::=
        <match_ruld_id> ":" <pattern>
<pattern> ::= {<variable>  |
        <string>  |
        "*"  |
        <char> }
<Relation_Rule> ::=
        <relation_rule_id> ":"
        [<cond>|<precond>|<action>]
<cond>  ::=  "cond"  <relation_rule_id>
        ["==" | "⊂"] <relation_rule_id>
<precond> ::= <relation_rule_id>
        {"," <relation_rule_id>}
<action>  ::= [ <assignment> | alert ]
<assignment> ::=
<variable>"="<relation_rule_id>"."<variable>
<variable> ::= "$"<digit>
<relation_rule_id> ::= "R"<digit>
<match_rule_ id> ::= "M"<digit>
```

표 3 : 정적 분석 규칙 BNF

기호	설명
:	규칙 일련번호와 규칙 구분자
==	동일 여부 비교 연산자
⊂	문자열의 부분 집합 관계
*	와일드 카드
<char>	특수 문자 ('.', '&', ',', 등)
alert	행위 탐지보고
<cond>	관계 분석시 비교 조건문
<precond>	관계 분석시 기 만족 규칙
<action>	관계 조건 만족시 수행 행동

표 4 : 규칙 부가 설명

아래는 로컬 시스템상의 자가 복제 행위 중 하나의 패턴을 위의 규칙에 따라 표현한 것이다.

*M1 : $1.copyfile wscript.scriptfullname, $2 **

R1 : action $1 = M1.$2

'M1'의 패턴 규칙에서 소스코드에 매치되는 부분은 'copyfile wscript.scriptfullname'이고 '$1', '$2'는 스크립트에 존재하는 객체를 나타내는 문자열이 대입되는 변수를 나타낸다. '*'는 뒤의 문자열은 모두 무시하라는 표시이며, 'R1'은 'M1'이 만족한 경우 'R1'의 변수 '$1'에 'M1'의 변수 '$2'의 값을 대입하라는 의미이다.

이와 같은 형태의 행위 패턴들을 스크립트 코드에 적용시켜 각 행위 별로 패턴이 존재하는지 여부를 판단하여, 존재하는 악성 패턴 및 악성 패턴들의 연관 관계를 분석하여 해당 스크립트 코드가 수행 가능한 악성 행위에 대한 정보를 얻는다. 현재 기존 악성 스크립트 코드 분석 작업을 통해 얻어진 악성 행위 패턴 중 로컬 파일 시스템상의 자가 복제, 복제된 파일을 메일로 전파, 인터넷 채팅 프로그램을 통한 전파 패턴들을 아래의 표5, 6, 7에 정의하였다.

```
M1 : $1.copyfile wscript.scriptfullname, $2 *
R1 : action $1 = M1.$2
```

표 5 : 로컬 시스템상의 자가 복제 패턴

```
M8 : $1.Attachments.Add $2
M9 : $1.send
R5 : cond R1.$1==M8.$2
R6 : precond R5
     cond M8.$1 == M9.$1
     action 메일을 통한 확산 경고
```

표 6 : 메일 첨부를 통한 확산 패턴

```
M15 :  *dcc send $nick $1
R11 :  cond R1.$1 ⊂ M15.$1
R12 :  precond R11
       action : IRC를 통한 확산 경고
```

표 7 : 인터넷 채팅 프로그램상의 확산 패턴

정의된 행위 패턴별로 스크립트 소스를 분석하여 각각의 행위 패턴별 결과와 행위 패턴 사이의 관계를 바탕으로 스크립트가 지니고 있는 악성 행위에 대한 보고를 한다. 예를 들어, 표 5의 정의를 통해 로컬 파일 시스템의 자가 복제가 탐지되고, 표 6의 정의를 이용하여 표 5에서 탐지된 로컬 파일 시스템에 자가 복제된 파일이 메일 첨부를 통해 전송되는 패턴을 탐지하는 경우 악성 행위로 판단된다. 실제 이와 같은 경우는 인터넷 웜이 메일을 통해 자신을 확산하는 방법과 동일하다.

3. 실험 결과

실험을 위해 수집된 악성 스크립트는 VBS로 작성된 스크립트 파일들로 'loveletter, homepage, bubbleboy'바이러스와 같이 이미 많은 피해를 가져온 악성 코드와 웜 생성기를 통해서 생성된 악성코드 및 기타 잘 알려지지 않은 악성 코드이다. 위의 악성 코드에 로컬 시스템에 자가 복제 패턴, 메일을 통한 코드 확산 패턴 및 인터넷 채팅 프로그램을 통한 코드 확산 패턴을 사용하여 정적 분석한 결과 탐지된 행동 패턴별로 분류하였다. 또한 이 결과를 기존의 안티 바이러스 제품의 검사 결과와 비교하였다.

	메일 확산 패턴	IRC 확산 패턴
개수	11	20

표 8 : 행위 패턴별 탐지 결과

	기존AV	정적 분석	
탐지	21	탐지	18
		미탐지	3
미 탐지	9	탐지	6
		미탐지	3
총계	30	30	

표 9 : 패턴 매칭 적용 결과

위 결과에서는 정적 분석 기법을 사용하는 탐지한 것이 효율적인 결과를 보여주며, 기존 안티 바이러스 제품의 미 탐지 악성 코드 9개 중 정적 분석 방법으로 탐지한 악성 코드가 6개로 나타나 알려지지 않은 악성 코드에 대한 탐지 대안으로써의 가능성을 제시한다.

실험 결과, 기존 안티 바이러스 제품은 탐지하였으나 정적 분석을 통해 탐지하지 못한 악성 코드는 스크립트 코드 전체나 행동 패턴의 일부분이 암호화된 형태의 코드들과, 스크립트 코드의 변수를 복잡하게 사용한 스크립트로 나타났다.

III. 결론

본 연구에서는 이미 알려진 바이러스만을 탐지하는 기존의 백신과는 달리 새롭게 출현하는 악성 스크립트까지도 탐지할 수 있는 새로운 방법을 제안했다. 악성 스크립트의 피해가 늘어가는 현재 이는 새로 생성되는 미확인 악성 스크립트까지 탐지한다는 점에서 중요한 의미를 갖는다. 각 악성 스크립트마다 고유한 시그니처를 생성하지 않고도 바이러스를 탐지할 수 있다는 점에서 바이러스의 유입 후 해당 바이러스의 대안이 마련될 때까지 소모되는 시간 문제를 해결하며 또한 동시에 여러 개의 바이러스가 출현 할 경우에도 그 피해는 비해서 현저히 줄어들 것이다.

추후 연구 사항으로 부정-오류를 감소시키는 방안과 긍정-오류를 감소시키는 방안의 연구가 필요하며 부정-오류를 감소시키는 방안으로는 실험 결과 문제가 되었던 암호화된 악성 스크립트의 복호화 방안에 대한 연구와 다형성을 나타내는 악성 스크립트에 연구 및 자신의 코드를 복잡하게 하기 위해 악성 코드에서 변수의 값을 다양하게 변경한 경우 String Evaluation과정을 통해 변수의 정확한 값을 분석하는 연구가 필요하다..

긍정-오류는 행위 패턴 매칭과정에서 각 제어 블록간에 변수들의 관계 여부가 검토되지 않으면 발생한다. 때문에 각 변수들의 Use-Def chain과 제어 흐름 분석을 통해 변수들의 상관 관계를 파악하는 방법을 연구하여 각기 독립적인 변수를 사용하는 패턴들의 정보를 행위 패턴 관계 분석 과정에 적용하여 탐지과정에서 발생하는 긍정-오류를 감소시킬 수 있다.

또한 실제 시스템 구현 시 악성 스크립트를 탐지하는 시점에 대한 언급이 이루어지지 않았으며, 정적 분석이 이루어지는 시점에 따라 탐지 방법 및 속도를 최적화 시켜 적용해야 할 것이다.

참고문헌

[1] http://www.ahnlab.com/
[2] http://www.research.ibm.com/antivirus/
[3] Mark Kennedy, Script-based Mobile Threats, Virus Bulletin Conference, Sep. 2000
[4] William arnold & Gerald Tesauro, Automatically Generated Win32 Heuristic Virus Detection, Virus Bulletin Conference, Sep. 2000
[5] Alex Shipp, Heuristic Detection of Viruses within Email, virus bulletin conference, sep, 2001
[6] Tim Hollebeek and Dur Berrier, Interception, Wrapping and Analysis Framework for Win32 Scripts, Cigital Labs