정적 분석을 이용한 다형성 스크립트 바이러스의 탐지기법 설계

이형준° 김철민 이성욱 홍만표
아주대학교 정보통신전문대학원
prime@doit.ajou.ac.kr° ily@ajou.ac.kr suleeip@yahoo.co.kr mphong@ajou.ac.kr

The Design for a Method of Detecting Polymorphic Script Virus Using Static Analysis

Hyungjoon Lee° Choimin Kim Seong-Uck Lee Manpyo Hong
Graduated School of Information and Communication

요 약

매크로 바이러스를 비롯한 악성 스크립트 바이러스는 이진 코드와는 달리 텍스트 형식으로 코드가 저장되기 때문에 많은 수의 변종이 가능하고 다형성을 지닌 형태로의 제작이 쉬워 새로운 형태의 출현이 빈번하다[1]. 이에 따라 시그니처 기반의 감지 기법을 탈피한 다양한 기법들이 제안되고 있으나 세밀한 수준의 분석으로 인한 시간 지연과 높은 긍정 오류의 문제로 현실적으로 적용되지 못하는 실정이다. 이를 개선하여 비교적 짧은 시간에 정적 분석을 끝내고 코드 삽입 기법을 병행하여 긍정 오류 문제를 해결한 기법이 제안 되었다[2]. 그러나 이 기법에서 사용하는 정적 분석은 다형성 스크립트 바이러스에 대하여 고려하고 있지 않다. 본 논문에서는 제안된 정적 분석 기법을 확장 하여 다형성 스크립트 바이러스를 탐지할 수 있는 기법을 제시 한다.

1. 서 론

1994년 12월에 최초로 발견된 악성 스크립트 코드는 인터넷 웜의 형태로 전자우편이나 IRC(Internet Relay Chat)와 같은 매체를 통해서 전파되고 있다. 스크립트 코드는 프로그래밍에 대한 전문적인 지식이 없는 초보자도 작성과 수정이 용이하다. 따라서 많은 변종들이 생겨나며, 전파하는 과정에서 자신의 형태를 변화 시키는 다형성 스크립트 바이러스까지 빈번하게 출현 되고 있다.

이에 따라 스크립트 바이러스에 대한 다양한 탐지기법들이 제안되고 있으나, 세밀한 분석을 필요로 하여 많은 시간과 비용이 소요된다. 이러한 이유로 현재 사용되고 있는 대다수의 안티 바이러스(Anti-virus) 시스템에서는 아직까지 시그니처 기반의 기법이나 문자열 비교 방식을 적용한 휴리스틱 기법에 그치고 있어 빈번하게 생겨나는 변종에 대해 신속하게 대처하지 못하고 있다.

이에 대해 정적 분석 기법과 코드 삽입 기법을 병행하여 대부분의 긍정 오류를 줄이면서 비교적 짧은 시간에 악성 코드를 감지할 수 있는 기법이 제안 되었다[3][4]. 그러나 이 기법은 자신의 코드를 변형 없이 복제하는 악성 스크립트를 대상으로 하는 것이어서 다형성 스크립트 바이러스에 대해서는 고려하지 않고 있다.

다형성 스크립트 바이러스는 전파하는 과정에서 스스로 변형 가능한 악성 코드로서, 최근 출현 빈도가 높아져 가고 있기 때문에 본 논문에서는 기존 연구를 확장하여 다형성을 보이는 악성 스크립트까지 감지 가능한 기법을 제시 한다

2. 관련연구

2.1 다형성 스크립트 바이러스

초기 다형성 스크립트 바이러스의 형태는 바이러스의 이름을 바꾸거나, 주석이나 데이터 흐름과는 관련이 없는 대입문과 같이 의미 없는 코드를 삽입하는 형태의 간단한 것이었다. 그러나 최근 발견된 다형성 스크립트 바이러스는 변수를 추가 또는 변경하거나, 프로그램 흐름을 변경하는 것을 비롯하여 코드 자체를 암호화 하는 수준까지 발전되고 있다[4].

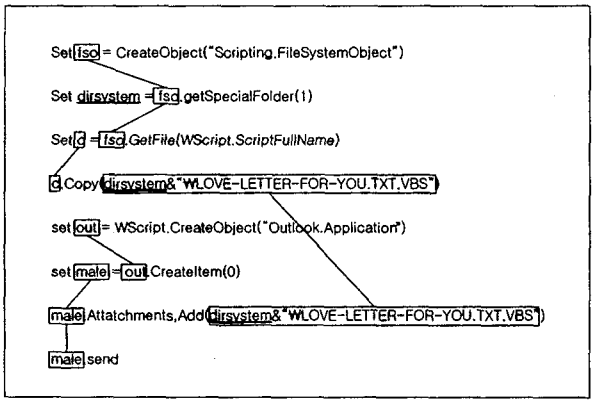
단순히 의미 없는 코드를 추가하는 것과는 달리 코드 자체를 변경하는 후자의 경우 알려져 있는 바이러스의 사전 분석을 통한 문자열 비교 방식으로는 감지할 수 없다[2]. 이에 따라 안티 바이러스들은 단순한 휴리스틱을 적용한 정적분석 방법을 많이 사용하고 있으나, 점점 더 복잡해져 가는 다형성 스크립트 바이러스에 대처하기에는 역부족이다. 따라서 다형성 스크립트 바이러스를 감지하기 위해서는 코드 자체를 에뮬레이션 하거나 컴파일 수준으로 정적분석 하는 것이 필요하다[5].

그러나 에뮬레이션이나 컴파일 수준의 정적 분석은 코드의 크기나 프로그램의 복잡도에 따라서 소요되는 시간이 현실적으로 적용할 수 없는 정도에 이를 수 있다. 이에 따라서 컴파일 최적화 단계에서 사용하는 자료 흐름 분석 기법을 활용하여 정밀한 분석을 수행하고, 정적 분석만으로는 판정이 어려운 경우에는 실행 시간에 자신을 진단하는 코드를 스크립트 코드에 삽입하는 비 다형성 스크립트 바이러스에 대한 탐지 기법이 제시되었다.[3][4]

2.2 자료 흐름 분석을 이용한 정적 분석

[그림 1]은 메일을 통해 전파되는 비쥬얼 베이직 스크립트(Visual Basic Script) 바이러스에서 주요 운장만을 일부 발췌

한 것이다. 제시된 예에서 확인 할 수 있듯이, 자기 자신을 복제하기 위해서는 Scripting.FileSystemObject 의 GetFile, Copy 메소드와 자기 자신의 파일 이름을 의미하는 Wscript.ScriptFullName 이라는 상수가 필요하다. 또한, 메일을 통해 전파되는 행위를 수행하기 위해서는 Outlook.Application 객체 내의 CreateItem, Add, Send 와 같은 메소드가 반드시 필요하다.



[그림 1] 메일을 통해 전파하는 VBS 바이러스의 일부

기존 문자열 검사 기반의 휴리스틱 알고리즘은 상술한 바와 같이 악성 행위를 위해 필요한 객체명이나 메소드의 출현 유무 또는 빈도수를 기반으로 악성 여부를 판단하기 때문에, 높은 긍정 오류를 수반하게 된다. 이를 보완하기 위해 제안된 기법에서는 컴파일 수준의 정적 분석을 통해서 메소드의 파라미터, 리턴 값들 간의 연관 관계까지도 고려하여 악성 행위를 감지한다[그림 1].

파리미터와 리턴 값들의 분석은 변수의 이름에 대한 단순한 문자열 비교와 함께, 변수의 실제 값에 대하여 분석이 이루어진다. 변수의 이름이 같은 경우라도 실행되는 과정에서 값이 바뀔 수 있기 때문이다. 변수의 값에 대한 분석을 위해서 자료 흐름 분석에 널리 사용되고 있는 상수 전파(Constant propagation)와 복사 전파(Copy propagation)를 이용한다.

상수 전파는 자료 흐름 분석에서 널리 사용되고 있는 기법으로, 프로그램의 실행 시 항상 특정 상수 값을 가지게 될 변수 또는 수식을 찾아내고, 이 상수 값을 가능한 프로그램 코드의 많은 부분으로 전파하는 것을 목적으로 한다.

복사 전파 역시 이와 유사한 기법으로, "x =: y" 형태의 복사문에 대한 분석을 통해 실행 시 항상 같은 값을 가지게 될 변수를 찾아내어 치환함으로써, 복사본의 수를 줄이는 기법을 지칭한다. 예를 들면, "x := y" 형태의 복사문이 생성 되었을 때 "z := x + 10" 문장은 "z := y + 10" 으로 대치될 수 있다.

복사문은 [그림 2]와 같이 변수의 값이 직접적으로 대입되는 간단한 형식의 대입문의 경우에 생성되며, 해당 문장에서는 다음과 같은 오직 하나의 복사문이 생성된다.

$$variable := factor$$

각 문장에서 생성된 복사문은 다음 실행 문장으로 전파 되어 집합을 이룬다. 복사 전파는 이 집합 내의 모든 복사문의 "variable" 을 현재 문장에서 사용되고 있는 변수와 비교하여, 일치할 경우 해당 변수를 "factor" 로 치환하는 방법으로 이루어진다.

복사 전파를 이용하면 대입문을 통해서 같은 값을 가지는 변수는 모두 같은 이름의 변수로 치환되기 때문에, 변수들 사이의 값에 대한 연관관계를 분석할 수 있다. 즉, 감지의 대상이 되는 문자열을 임의의 변수에 숨김으로써 감지를 회피 하려는 악성 코드도 감지할 수 있다.

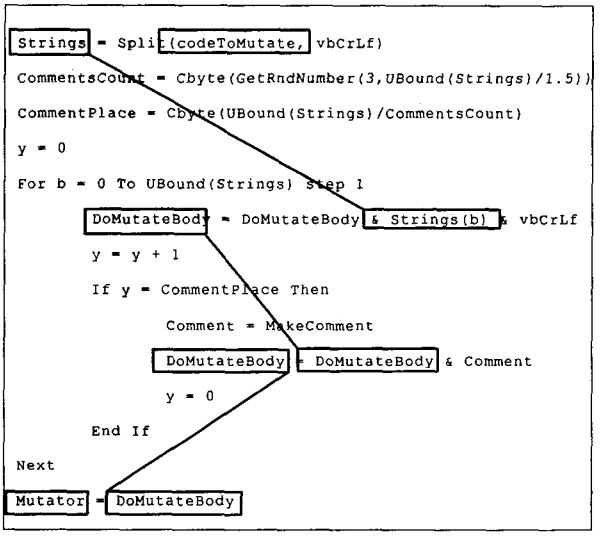
```
variable <assignop> factor
variable := <identifier>
factor := <identifier>
        | <number>
        | <literal>
```

[그림 2] 복사전파가 이루어지는 문장

3. 다형성 스크립트 바이러스 탐지기법

다형성 바이러스의 악성 행위 패턴은 자신의 코드를 읽어오는 단계와 읽어온 자신의 코드를 변형하는 단계, 변형된 코드를 전파하는 세 가지 단계로 나눌 수 있다. 이 중에서 자신을 변형하는 단계의 경우 자기 변형 과정에서 수행되는 연산이 [그림 2]의 형태를 벗어나기 때문에, 이전의 기법을 이용할 경우 상술한 복사 전파가 이뤄지지 않는다.

자기 변형 과정에서 복사 전파가 이루어지지 않게 되면, 변형된 코드의 내용을 전파하는 세 번째 단계에 이르더라도 변형 전의 코드와 변형 후의 코드 사이의 연관관계가 성립되지 않게 된다. 따라서 기존의 방법으로는 변형된 코드의 전파 또는 복제 자체를 감지하지 못하게 된다.



[그림 3] VBS/Sflus.A 바이러스 코드의 일부

[그림 3]은 VBS/Sflus.A 다형성 스크립트 바이러스의 일부를 발췌한 것이다. VBS/Sflus.A 는 코드 사이에 무작위로 주석문을 생성하여 삽입하는 방법으로 코드를 변경하는 다형성 바이러스이다. 발췌한 부분의 앞부분은 자신의 코드를 읽어 codeToMutate 변수에 저장하는 부분이고, 그림에 보이는 부분은 읽은 코드를 변

형하는 부분이다. 그림을 보면 알 수 있듯이 변형된 코드가 저장되는 변수는 Mutator 이다. 이 경우 복사 전파를 통해서 분석할 경우 codeToMutate 에서 Mutator 로 값이 전달되는 과정에서 다른 변수와 연산을 하기 때문에, [그림 2]와 같은 형태를 벗어나게 되어 복사 전파가 이루어지지 않는다. 따라서 복사 전파를 이용한 분석 방법으로는 codeToMutate 와 Mutator 변수 사이의 연관 관계를 알아낼 수 없으며, Mutator 의 값을 파일로 기록하여 전파하더라도, 악성행위를 감지할 수 없다. 이러한 경우 다른 변수와의 연산이나 함수의 리턴 값에 의해 값이 변형되어 전달되는 변수들 간의 연관관계를 분석할 필요가 있다. 따라서 [그림 4]의 형태를 가지는 문장에 대해서도 복사문을 추출하는 방법으로 복사 전파를 확장한 분석이 이루어져야 한다.

```
variable <assignop> expression
variable := <identifier>
expression := factor
      | expression <op> factor
      | expression <op> function
      | expression <op> parenthesis
function := <identifier> '(' expression ')'
parenthesis := '(' expression ')'
factor := <identifier>
      | <number>
      | <leteral>
```

[그림 4] 확장된 복사전파가 이루어지는 문장

확장된 복사 전파에서 복사문의 추출은 우변의 연산에 사용되는 모든 변수 또는 상수들에 대해서 좌변의 변수를 조합하는 방법이다. [그림 4]의 형태를 만족하는 대입문에 포함되어 있는 "factor" 들의 집합을 F 라 하고, 생성되는 복사문 "variable := factor" 의 집합을 G 라 정의할 때, 한 대입문 에서 추출되는 복사문의 집합 G 는 다음과 같다.

$$F = \{ f \mid f : \forall\ factor\ in\ assign-statement \}$$
$$G = \{ g \mid g : variable := f,\ f \in F \}$$

예를 들어 [그림 3]의 코드 중에서 첫 번째 라인을 보면, Strings 변수의 값을 결정하는 연산에서 사용된 변수 codeToMutate, vbCrLf 가 Strings 와 개별적인 복사문을 형성하게 된다. 따라서 생성되는 집합 F, G 는 각각 다음과 같다.

$$F = \{ codeToMutate, vbCrLf \}$$
$$G = \{ "Strings := codeToMutate", \quad "Strings := vbCrLf" \}$$

확장된 복사 전파 기법은 엄격한 자료 흐름 분석이 아니기 때문에, 긍정오류를 수반할 수 우려가 있다. 그러나 확장된 복사 전파 방식은 컴파일러 기법에서 적용된 코드 최적화 기법에서의 의미와는 다르다. 본 논문에서 적용한 자료 흐름 분석은 악성 행위에 이용되는 특정 메소드의 파라미터와 리턴 값들의 관계를 검증하는 데 있기 때문에 제시한 복사 전파 방법에 의해서 긍정 오류가 발생할 가능성은 매우 작다.

제시한 복사 전파를 이용할 경우 연산자나 함수를 포함한 대입문에 대해서도 복사문을 생성할 수 있기 때문에 [그림 3]과 같은 코드에서도 codeToMutate 변수와 Mutator 변수간의 연관관계를 얻을 수 있으며, 일반적인 다형성 바이러스에 대해서 제안한 정적 분석을 통해 변형된 코드를 전파하는 악성 행위를 탐지해 낼 수 있다.

4. 결론

많은 수의 변종이 발견되는 스크립트 바이러스에 대처하기 위해서 정적 분석과 코드 삽입 기법을 병행하는 기법이 제안 되었다. 제안된 기법은 긍정 오류를 줄이면서 비교적 짧은 시간에 악성 코드를 탐지할 수 있으나, 최근 증가하고 있는 다형성 스크립트 바이러스에 대해서는 고려하지 않은 기법이었다.

본 논문에서는 과거의 비 다형성 스크립트 바이러스 탐지 기법을 개선하여 다형성 스크립트 바이러스를 탐지할 수 있는 방법을 제안 하였다. 제안한 정적 분석 기법은 복사 전파 방법을 확장함으로서 자기 수정 행위를 하는 악성 코드의 자료 흐름의 분석을 가능하게 하여 다형성 스크립트 바이러스에 대해서도 감지가 가능하다.

참고문헌

[1] CERTCC-KR, "2002년 2월 바이러스 통계", http://www.certcc.or.kr", 2000.

[2] Igor Muttik, "Stripping down an AV Engine", Virus Bulletin Conference, 2000.

[3] 배병우, "정적 분석 기법을 이용한 악성 스크립트 탐지", 정보처리학회논문지 C, 9-C권, 5호, 2002.

[4] 이성욱, "정적 분석과 코드 변환을 이용한 적극적인 악성 스크립트 대응", 아주대학교 박사 학위 논문, 2003.

[4] Vesselin Bontchev, "Macro and Script Virus Polymorphism", VIRUS BULLETIN CONFERENCE, 2002.

[5] Gabor Szappanos, "Are There Any Polymorphic Macro Viruses at All? (...And What to Do with Them)", VIRUS BULLETIN CONFERENCE, 2002.

[6] Alex Shipp, "Heuristic Detection of Viruses within Email", VIRUS BULLETIN CONFERENCE, 2001.

[7] Symantec Anti-Virus Research Center, "Understanding Heuristics", Symantec White Paper, 1998.

[8] Gabor Szappanos, "VBA Emulator Engine DEsign", VIRUS BULLETIN CONFERENCE, 2001.

[9] Francisco Fernandez, "Heuristic Engines", VIRUS BULLETIN CONFERENCE, 2001.