다형성 스크립트 바이러스 탐지를 위한 자료 흐름 분석기법의 확장

김 철 민[†] · 이 형 준[†] · 이 성 욱[††] · 홍 만 표[†††]

요 약

스크립트 바이러스는 제작이 쉽고 텍스트 형식으로 코드가 유포되는 특징으로 인해 변종 출현이 빈번하여, 시그너쳐에 의존하지 않고 탐지하려는 시도가 이루어지고 있다. 그러나 이러한 단순 휴리스틱 기법은 긍정 오류가 높은 단점이 있다. 이를 개선하기 위해 자료 흐름 분석을 이용하여 오류율을 낮춘 탐지 기법이 제시되었다. 그러나 이 기법은 탐지 대상이 다형성 바이러스여서 셀프를 읽어 들여 변형을 가한 후 새로운 복사본을 만드는 방식으로 전파될 경우 탐지하지 못하는 단점을 지닌다. 이를 극복하기 위해 본 논문에서는 기존의 자료 흐름 분석 휴리스틱 기법이 가지는 정적 분석 기법을 확장하여 다형성 스크립트 바이러스가 가지는 특징도 탐지할 수 있도록 하는 기법을 제안한다. 확장된 자료 흐름 분석 휴리스틱 기법은 확장된 문법을 통해 기존의 기법이 인식할 수 없었던 변형된 복사 전파를 인식할 수 있다. 또한 본 논문에서는 제안된 기법을 구현하여 제안된 기법이 가지는 탐지율에 대한 실험 결과를 제시한다.

An Extension of Data Flow Analysis for Detecting Polymorphic Script Virus

Cholmin Kim[†] · Hyoungjun Lee[†] · Seong-uck Lee[††] · Manpyo Hong[†††]

ABSTRACT

Script viruses are easy to make a variation because they can be built easily and be spread in text format. Thus signature-based method has a limitation in detecting script viruses. In a consequence, many researches suggest simple heuristic methods, but high false-positive error is always being an obstacle. In order to overcome this problem, our previous study concentrated on analyzing data flow of codes and has low-false positive error, but still could not detect a polymorphic virus because polymorphic virus loads self body and changes it before make a descendent. We suggest a heuristic detection method which expands the detection range of previous method to include polymorphic script viruses. Expanded data flow analysis heuristic has an expanded grammar to detect polymorphic copy propagation. Finally, we will show the experimental result for the effectiveness of suggested method.

키워드 : 스크립트 바이러스(Script Virus), 자료 흐름 분석(Data Flow Analysis), 바이러스 다형성(Virus Polymorphism)

1. 서 론

바이러스나 웜과 같은 악성 코드의 탐지에는 시그너쳐 기반의 탐지기법이 주로 사용되어 왔다[1]. 시그너쳐 기반 기법은 다른 기법에 비해 상대적으로 속도가 빠르며 탐지된 악성 코드가 어떤 분류에 속하는지를 잘 식별해내는 장점이 있다. 그러나 시그너쳐가 알려지지 않은 악성코드에 대응하지 못하는 단점을 지니고 있다.

한편 1994년 12월에 최초로 발견된 스크립트 바이러스는 인터넷 웜의 형태로 전자우편이나 IRC(Internet Relay Chat)와 같은 매체를 통해서 전파되고 있다[2]. 스크립트 바이러스는 고급 언어로 쓰여지므로 제작이 쉽고, 텍스트 형식으로 저장, 유포되므로 변종생성이 용이하여 특히 문제가 되고 있다[3, 4]. 변종이 많은 스크립트 바이러스의 특성상 시그너쳐 기반 방식으로 대응하기 어려워 단순 휴리스틱을 이용한 탐지가 시도되고 있으나 높은 긍정 오류 및 시간 복잡도를 가지고 있어 적극적인 활용에 어려움이 존재하는 것이 현실이다.

이러한 스크립트 악성 바이러스에 대응하기 위해 자료 흐름 분석 기법을 이용한 휴리스틱 기법이 제안되었다[5,

※ 본 연구는 한국과학재단 목적기초연구(R05-2003-000-11235-0)지원으로 수행되었음.
† 준 회 원 : 아주대학교 정보통신 전문대학원 정보통신공학과
†† 정 회 원 : 신구대학 인터넷정보과 교수
††† 종신회원 : 아주대학교 정보및컴퓨터공학부 교수
논문접수 : 2003년 5월 19일, 심사완료 : 2003년 11월 17일

6]. 이 기법은 시그너처에 의존하지 않으므로 변종 스크립트 바이러스 및 아직 알려지지 않은 스크립트 바이러스에도 대응할 수 있으며, 상대적으로 짧은 수행시간과 낮은 긍정 오류를 가진다. 그러나 이 방식은 스크립트 바이러스들이 셀프 코드를 읽어 들인 후 새로운 복사본을 생성함으로써 자기복제를 수행하는 것에 착안하여 탐지를 수행하므로 셀프 코드를 읽어 들인 후 변형을 가하여 복제를 수행하는 다형성 바이러스(Polymorphic Virus)는 탐지 할 수 없다. 스크립트 코드는 전술한 것과 같이 텍스트 형식으로 저장, 유포 되므로 처음부터 다형화된 형태로 제작하거나 기존의 것을 다형화하기 쉽다. 따라서 다형성 스크립트 바이러스를 탐지하기 위해서 기존의 자료 흐름 분석을 이용한 휴리스틱 탐지 기법을 확장할 필요가 있다.

본 논문에서는 기존의 자료 흐름 분석 휴리스틱을 통한 스크립트 바이러스 탐지기법을 확장하여 다형성 스크립트 바이러스까지 탐지 가능한 기법을 제안하고, 제안된 기법의 탐지율에 관한 실험 결과를 보인다. 실험 결과에서는 현재 일반적으로 사용되고 있는 상용 안티 바이러스 제품 및 기존 자료 흐름 분석 휴리스틱 기법과 제안된 기법의 비교 결과를 제시한다.

2. 관련 연구

2.1 다형성 스크립트 바이러스

초기의 다형성 스크립트 바이러스는 바이러스가 포함된 매크로 또는 파일의 이름을 바꾸거나, 주석이나 데이터 흐름과는 관련이 없는 대입문과 같이 의미 없는 코드를 삽입하는 형태의 간단한 것이었다. 그러나 최근 발견되고 있는 다형성 스크립트 바이러스는 복잡한 다형화 기법을 이용하여 탐지가 어려운 방향으로 계속해서 발전하고 있다[3, 4]. 현재 스크립트 바이러스 코드에서 사용되는 다형화 기법은 단순 다형화 기법과 복잡 다형화 기법으로 크게 분류 할 수 있다. 단순 다형화 기법과 복잡 다형화 기법을 나누는 기준은 다음 세대의 스크립트 코드 생성 시 구문적 의미(semantics)의 변경 여부이다.

단순 다형화 기법에는 바이러스가 포함되어 있는 스크립트 또는 파일의 이름을 세대별로 다양화 하는 다형화된 컨테이너(Container)명 기법, 한정된 개수의 형태를 각 세대가 순서대로 가지는 올리고몰피즘(Oligomorphism) 기법, 바이러스가 다음 세대를 만들기 전 현재 세대에 쓰여진 변수의 리스트를 저장한 후 저장되어 있는 변수들을 찾아 무작위로 생성된 스트링으로 치환하는 변수 치환 기법, 바이러스의 바디(Body)중 일부에 주석문이나 프로그램의 실행에 영향을 주지 않는 코드를 삽입하는 의미 없는 코드 삽입 기법 등이 있다.

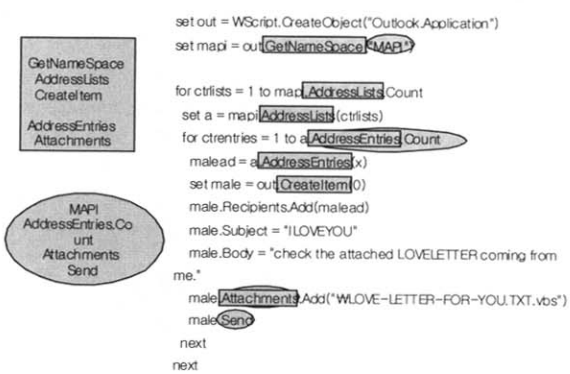
한편 복잡 다형화 기법에는 단순 기법의 의미 없는 코드 삽입과 같으나 삽입되는 코드의 길이와 위치를 무작위로 선택하는 다형화된 의미 없는 코드 삽입 기법, 바이러스의 바디 부분을 암호화하고 키를 세대별로 다르게 하여 각 세대별로 바이러스가 다르게 보이도록 만드는 암호화 기법, 프로그램 흐름에 영향을 주지 않는 범위 내에서 임의의 두 라인 순서를 바꾸는 코드 순서 섞기 기법, 다양한 암호화 기법을 각 세대별로 적용하여 복호화 하는 루틴이 다양해지도록 하는 코드 생성 기법, 지금까지 나열된 기법들이 결합되어진 것 등이 있다.

단순히 의미 없는 코드를 추가하거나 변수명을 변경하는 단순 다형화 기법과는 달리 코드의 구문적 의미를 변경하는 복잡한 다형화 기법의 경우 부모 바이러스와 다음 세대의 바이러스에 공통적으로 적용 시킬 수 있는 시그너처를 찾기 어려우므로 보고된 바이러스의 분석을 통한 시그너처 비교 방식으로는 완전한 탐지가 불가능하다.

2.2 스크립트 바이러스 탐지를 위한 기존 연구

2.2.1 단순 휴리스틱 기법

시그너처를 이용하지 않는 스크립트 바이러스 탐지 기법으로는 문자열 검사 기반의 단순 휴리스틱 기법이 있다[7, 8]. 이 기법은 바이러스에서 자주 사용되는 메서드 또는 객체들을 문자열로 정의하고 특정 스크립트에서 바이러스에 이용되는 것으로 정의된 문자열이 일정 기준 이상으로 발견되는지 탐지하는 기법이다.



(그림 1) 러브 레터 웜(love-letter worm)의 일부

(그림 1)은 전자 우편을 통해 전파되는 비주얼 베이직 스크립트 바이러스의 한 종류인 러브 레터 웜에서 주요 문장을 발췌한 것으로서 전자 우편을 통해 자기 자신을 발송하는 부분이다. 단순 휴리스틱 기법은 실제로 전자 우편을 통해 자기복제(self-replication)를 수행하는가를 정확히 판단하는 것이 아니라 특정한 메서드와 어트리뷰트의 존재만을 검색하여 바이러스 여부를 가리게 된다. 실제로 실험 과정에서 확인해 본 결과, 알려지지 않은 바이러스 진단에 성공한 한 안티바이러스는 사각형으로 표시된 단어들만을 포함

한 VBS 파일을 악성으로 진단하였으며, 다른 안티바이러스도 타원으로 표시된 단어들만을 포함하면 그것을 악성코드로 진단하였다. 하지만 이 단어들은 전자 우편 발송에도 사용될 수 있으므로 이 단어들 만으로 바이러스 여부를 판정하게 되면 정상적인 전자 우편 발송을 목적으로 작성된 스크립트를 바이러스로 진단하는 긍정 오류가 발생하게 된다.

또 다른 예로서 (그림 2)역시 러브 레터 웜에서 추출된 코드이며, 시스템에 존재하는 확장자가 VBS인 모든 파일에 자신의 내용을 겹쳐 쓰는 동작을 수행한다.

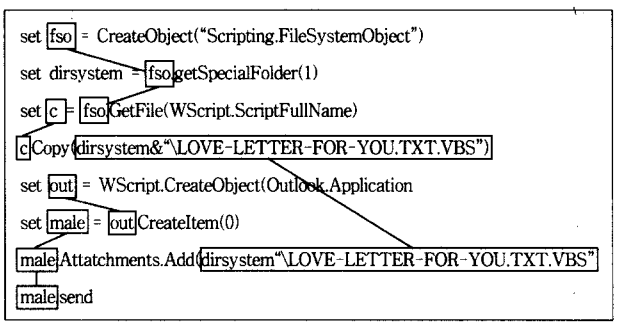
```
Set fso = CreateObject("Scripting.FileSystemObject")
set file = fso.OpenTextFile(WScript.ScriptFullname, 1)
vbscopy = file.ReadAll        '자신의 내용을 읽음
file.close
folderlist("C:\")
sub folderlist(folderspec)    '모든 폴더를 탐색하며, infectfiles를
                               호출
dim f, f1, sf
set f = fso.GetFolder(folderspec)
set sf = f.SubFolders
for each f1 in sf
  infectfiles(f1.path)
  folderlist(f1.path)
next
end sub
sub infectfiles(folderspec)   '확장자 vbs인 모든 파일에 자기복제
dim f, f1, fc, ap
set f = fso.GetFolder(folderspec)
set fc = f.Files
for each f1 in fc
  if fso.GetExtensionName(f1.path) = "vbs" then
    set ap = fso.OpenTextFile(f1.path, 2, true)
    ap.write vbscopy
    ap.close
  end if
next
end sub
```

(그림 2) 일반적인 스크립트에서 자주 사용하는 메서드들로 스크립트 바이러스의 예

이 코드는 시스템에 존재하는 모든 VBS 파일을 자신과 같은 스크립트로 만드는 행위를 수행하므로 명백히 바이러스임에도 불구하고, 파일을 열고, 폴더의 리스트를 얻는 등, 많은 스크립트에서 사용하는 일반적인 메서드들로만 이루어져 있다. 따라서 특정 단어의 존재 유무는 스크립트 바이러스와 정상적인 스크립트를 구별하는 완전한 기준이 아니며 특정 단어의 존재유무 만을 탐색하는 것만으로 특정 스크립트를 악성이라고 판정하게 되면 많은 정상 코드들이 악성으로 간주될 수 있다. 따라서 이러한 기법은 높은 긍정 오류를 수반하게 된다.

2.2.2 자료 흐름 분석을 이용한 휴리스틱 기법

기존의 단순 휴리스틱 기법이 보이는 높은 긍정 오류를 보완하기 위해 제시된 것이 자료 흐름 분석을 이용한 휴리스틱 기법이다. 자료 흐름 분석은 컴파일러에서 구문 오류 탐지(semantic error detection)또는 코드 최적화(code optimization)를 위해 사용되는 기법으로서 이 기법을 사용하면 시그너쳐에 의존하지 않고도 특정 스크립트가 바이러스인지를 높은 정확도로 구별할 수 있음이 밝혀졌다[5, 6]. 이 기법은 세밀한 정적 분석을 통해서 메서드의 파라미터, 리턴 값들간의 연관 관계까지도 고려하여 악성 행위를 탐지한다.



(그림 3) LOVE-LETTER 바이러스의 일부

(그림 3)의 각 사각형은 메서드와 파라미터, 리턴 값을 가리키고 간선들은 이것들이 서로 연관되어 있음을 나타낸다. 정상적인 스크립트 프로그램들은 자신의 코드를 읽은 뒤에 이것을 전자 우편에 첨부하는 등의 행위를 하지 않으므로 (그림 3)에서 간선으로 표현되는 연관관계를 가지지 않는다. 따라서 각 메서드와 객체간의 연관관계를 이용한 이 기법을 통해 문자열 검사 기반의 단순 휴리스틱이 가지던 높은 긍정 오류를 낮출 수 있게 되었다.

파리미터와 리턴값들의 분석은 변수의 이름에 대한 단순한 문자열 비교와 함께, 변수의 실제 값에 대하여도 이루어진다. 변수의 이름이 같은 경우라도 실행되는 과정에서 값이 바뀔 수 있기 때문이다. 자료 흐름 분석 휴리스틱 기법은 변수의 값에 대한 분석을 위해서 상수 전파(constant propagation)와 복사 전파(copy propagation)의 개념을 이용한다.

상수 전파는 프로그램의 실행 시 항상 특정 상수 값을 가지게 되는 변수 또는 수식을 찾아내고, 해당 변수 또는 수식을 상수 값으로 대치하는 것을 말한다. 이를 통해 프로그램 코드의 가능한 많은 부분이 상수로 치환되어 처리되므로 프로그램 분석이 쉬워 지고 잉여 부분을 제거 할 수 있다.

복사 전파 역시 유사한 기법으로, "x := y" 형태의 단순 대입문을 대상으로 한다. 이러한 대입문들의 분석을 통해 프로그램이 실행 되면 항상 같은 값을 가지게 될 변수들을 찾아내어 하나의 변수로 치환함으로써, 완전히 동등한 값을 가지는 잉여 변수의 수를 줄인다. 예를 들어, "x := y"와 "z := x + 10" 형태의 대입문이 순서대로 있는 프로그램이 있다면, 두 번째 문장을 "z := y + 10"으로 대치하여도 프로

그램의 무결성에 영향을 주지 않는다.

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

(그림 3) 복사전파가 이루어지는 문장에 대한 BNF 표현

복사문은 (그림 4)의 문법에 대응되는 형식의 대입문인 경우에 생성되며, 앞 문단의 x, y에 대한 예에서는 위의 문법에 의해 다음과 같은 오직 하나의 복사문이 생성된다. 이 경우 'variable'에 해당되는 변수는 x, 'factor'에 해당되는 변수는 y이다.

variable := factor

각 문장에서 생성된 복사문은 다음 실행 문장으로 전파되어 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 변수가 모두 같은 이름으로 치환되기 때문에, 변수들 사이의 값에 대한 연관관계를 분석할 수 있다. 즉, 탐지 대상이 되는 문자열을 임의의 변수에 숨김으로써 탐지를 회피하려는 악성 코드를 정상 적인 스크립트와 구별할 수 있으므로 긍정 오류를 줄이는 장점을 가진다. 다만, 이 기법은 다형성을 가지지 않는 스크립트 웜을 대상으로 연구된 것이므로, 다형성을 가진 바이러스 형태의 악성코드는 감지하지 못한다는 단점을 가지고 있다. 즉, 단순 휴리스틱 기법이 단어 검색 기반이므로 다형성 기법에 큰 영향을 받지 않음에 비해, 자료 흐름이 정확히 일치하는 것만을 악성으로 진단하므로 다형성 바이러스에 대해서는 부정 오류가 발생하게 된다.

3. 제안된 기법

3.1 자료 흐름 분석 휴리스틱 기법의 문제점

다형성 스크립트 바이러스의 악성 행위 패턴은 자신의 코드를 읽어오는 단계와 읽어온 자신의 코드를 변형하는 단계, 변형된 코드를 전파하는 세 가지 단계로 나눌 수 있다. 첫 번째 단계에서는 스크립트 바이러스가 자신의 코드를 변형하는 연산을 수행하기 위해 특정 변수에 자신의 코드를 문자열로 복사해서 저장한다. 두 번째 단계에서 위의 변수를 2장에서 나열한 여러 가지 다형화 기법을 적용하여 변경하고, 마지막으로 변경된 변수의 값 즉 다형화된 자신의 코드를 세 번째 단계에서 호스트 내에 저장하거나 전자우편등을 이용하여 외부로 전파 하게 된다. 이때 기존 자료 흐름 분석 휴리스틱 기법을 이용할 경우 두 번째 단계로 인해 상술한 복사 전파가 이루어 지지 않으므로 (그림 3)에서 보이는 간선이 끊어지게 된다. 자기 변형 과정에서 복사 전파를 탐지 하지 못하면, 변형된 코드 내용의 전파를 탐지하는 세 번째 단계에 이르러서도 변형 전의 코드와 변형 후의 코드 사이의 연관관계를 알 수 없게 된다. 따라서 기존의 방법으로는 변형된 코드의 전파 또는 복제 자체를 탐지하지 못하게 된다. (그림 5)는 이러한 스크립트 바이러스의 예를 보여 준다.

```
Strings = Split(codeToMutate, vbCrLf)
CommentsCount = Cbyte(GetRndNumber(3, UBound(Strings)/1.5))
CommentPlace = Cbyte(UBound(Strings)/CommentsCount)
y = 0
For b = 0 To UBound(Strings) step 1
        DoMutateBody = DoMutateBody & Strings(b) & vbCrLf
        y = y + 1
        If y = CommentPlace Then
                Comment = MakeComment
                DoMutateBody = DoMutateBody & Comment
                y = 0
        End If
Next
Mutator = DoMutateBody
```

(그림 5) VBS/Sflus.A 바이러스 코드의 일부

(그림 5)는 '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 변형하는 부분이다. 그림을 보면 알 수 있듯이 변형된 코드가 저장되는 변수는 'Mutator'이다. 이 경우 복사 전파를 통해서 분석하면 'codeToMutate' 변수에서 'Mutator' 변수로 값이 전달되는 도중 다른 변수와 연산을 한 결과를 다시 원래의 변수에 대입하는 연산이 있기 때문에, (그림 4)로 나타낼 수 있는 문법의 형태를 벗어나게 되어 복사 전파가 이루어지지 않는다. 따라서 기존의 복사 전파를 이용한 분석 방법으로는 'codeToMutate' 변수와 'Mutator' 변수 사이의 연관 관계를 알아낼 수 없으며, 'Mutator' 변수의 값을 파일로 기록하여 전파하더라도, 악성행위를 탐지할 수 없다.

3.2 확장된 복사 전파 휴리스틱 기법

앞 절에서 상술한 바와 같은 바이러스의 경우 다른 변수

와의 연산이나 함수의 리턴 값에 의해 값이 변형되므로 서로 다른 변수들 간의 연관관계도 분석할 필요가 있다. 즉 (그림 6)의 형태를 가지는 문장에 대해서도 복사문을 추출하는 방법으로 확장된 자료 흐름 분석이 이루어져야 한다.

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

(그림 6) 확장된 복사전파가 이루어지는 문장에 대한 BNF 표현

확장된 기법에서 복사문의 추출은 우변의 연산에 사용되는 모든 변수 또는 상수들에 대해서 좌변으로의 대입이 일어나는 경우에 이루어진다. (그림 6)의 형태를 만족하는 대입문에 포함되어 있는 'factor'들의 집합을 F 라 하고, 생성되는 복사문 'variable := factor'의 집합을 G 라 정의할 때, 한 대입문에서 추출되는 복사문의 집합 G 는 다음과 같다.

$G = \{g \mid g : \text{'variable : = f'}, f \in F\}$
when $F = \{f \mid f : \text{factor assign statement}\}$

예를 들어 (그림 5)의 코드 중에서 첫 번째 줄을 보면, Strings 변수의 값을 결정하는 연산에서 사용된 변수 code ToMutate, vbCrLf가 Strings와 개별적인 복사문을 형성하게 된다. 따라서 생성되는 집합 F와 G는 각각 다음과 같다.

F = 'codeToMutate', 'vbCrLf'
G = 'Strings := codeToMutate',
'Strings := vbCrLf'

즉 'codeToMutate' 변수의 값이 'Strings' 변수로 전달 되었음을 알 수 있다. 이러한 방식으로 생성된 복사문에서 우항의 변수가 파일로 쓰여진다면 해당 스크립트가 바이러스라고 결정할 수 있다. 따라서 복사문의 우항에 해당되는 변수들에 대한 테이블을 유지하여 이 변수들이 파일로 저장되는지 확인하여야 한다.

한편 (그림 6)의 문법에는 **'function := <identifier> '(' expression ')''**과 같이 함수를 호출하는 경우가 포함되어 있다. 현재의 상용 AV들이 이용하는 단순 휴리스틱으로는 함수를 호출하여 셀프를 전달한 후 함수 내에서 자기복제를 수행할 경우 자기복제 수행 여부를 탐지 할 수 없다. 그러나 제안된 기법은 셀프를 가진 변수가 함수로 전달되는 경우를 포함하고 있으므로 이 경우에도 대응 할 수 있다.

확장된 휴리스틱 기법의 알고리즘은 (그림 7)과 같다. 제안된 알고리즘에서 F set은 전술한 자료 흐름에서의 F set과 같고 G set은 복사문에서 우항의 변수들을 원소로 가지는 집합이다.

한편 확장된 복사 전파 휴리스틱 기법은 컴파일러에서 적용된 코드 최적화 기법의 자료 흐름 분석과는 다른 의미를 지닌다. 본 논문에서 적용한 복사 전파 기법은 악성 행위에 이용되는 특정 메서드의 파라미터와 리턴 값들의 관계를 검증하는 데 있다. 정상적인 프로그램이 고의적으로 셀프를 읽어 들인 후 이것을 변경하는 연산을 하여 연산된 값을 다시 파일의 형태로 저장하는 일은 일반적으로 일어나기 어렵다. 즉 셀프를 읽어 들여 변경하는 것은 바이러스로 의심할 수 있는 행위이기 때문에 제시한 기법이 엄격한 자료 흐름 분석을 하지 않음으로 인해 긍정 오류를 발생시킬 가능성은 매우 낮다. 그러면서도 제안된 기법을 이용할 경우 연산자나 함수를 포함한 대입문에 대해서 복사문을 생성할 수 있기 때문에 (그림 5)와 같은 코드에서도 'code ToMutate' 변수와 'Mutator' 변수간의 연관관계를 얻을 수 있다. 즉 제안된 기법은 일반적인 다형성 바이러스에서 변형된 코드를 전파하는 악성 행위를 탐지해 낼 수 있는 장점이 있다.

```
Input : Script file in text form (___.vbs)
Output : Yes/No value whether the input file is a virus or not

    Initialize this to indicate the first line of input file
    Initialize F set and G set empty
    Loop (until this reach to end of the input file)
    {
        S = Readline this
        If (self-open behavior is detected in S)
                Add self-saved variable in S to F set
        if (more than one element in F set or G set is found
            in L-value of S AND S has a form of one of a
            statement in figure 6)
                Add r-value of S to G set
        if (more than one element in G set is written to
            certain file)
                Print Yes
                Halt
        Increase this to indicate next line
    }
    Print No
    Halt
```

(그림 7) 확장된 복사 전파 휴리스틱 기법 알고리즘

4. 실험 및 결과 분석

4.1 샘플 집합 생성

제안된 기법의 효율성을 실험하기 위해서는 다양한 종류

의 다형성 바이러스 샘플이 필요하지만 학술적 목적으로 이것을 구하는 것이 현실상 어려웠으므로 실험에서는 바이러스의 다형성에 관한 논문들을 참조하여 현재 유포 되었던 일반적인 스크립트 바이러스에 다형성 기능을 첨가한 샘플을 생성하였다[3, 4]. 일반적인 스크립트 바이러스에 첨가한 다형성 기능은 주석문 삽입, 대입문 삽입, 변수 치환, 임의의 함수 호출의 4가지이다. 이러한 분류는 2장에서의 다형성 스크립트 바이러스 분류와는 다른데 이는 2장에서의 각 기법을 샘플 스크립트 바이러스를 생성하는데 사용된 방식에 따라 다시 분류하였기 때문이다. 생성 기법적 측면에서는 다형성 바이러스를 암호화, 파일명 변경, 변수명 변경, 문장 삽입/치환으로 나눌 수 있는데 암호화는 코드의 일부를 에뮬레이션 하거나 암호를 푸는 전 처리 과정을 통해 대응하는 것이 일반적이고, 파일명 변경은 스크립트 내용과는 무관하므로 감지 대상에서 제외되었다[6]. 따라서 본 논문의 실험을 위한 샘플은 변수명 변경, 문장 삽입(주석, 대입문)을 적용한 것이며, 문장 치환기법은 상대적으로 복잡한 과정을 거치므로 대부분 함수 형태로 구현된다는 점을 이용하여 함수 호출을 이용한 다형성을 적용하였다. 이 방법들은 다형성 바이러스의 분류에 대한 저서들을 참고하여 공통점을 취한 것으로 현재 분류 가능한 다형성 기법을 대부분 포함할 수 있었다. 각각의 기능은 다음과 같은 방법으로 구현 되었다.

- 주석문 삽입 : 바이러스 코드내에서 코멘트를 삽입하여도 상관없는 부분을 찾아서 해당 부분에 " '…. " 형식의 코멘트를 삽입하는 기능을 추가한다.
- 대입문 삽입 : 무작위로 스트링을 생성하여 생성된 스트링을 변수로 하는 무작위 대입문(assignment)을 생성한다. 생성된 대입문을 바이러스 코드내에서 프로그램의 흐름을 방해하지 않는 부분에 삽입하는 기능을 추가한다.
- 변수 치환 : 바이러스를 Open하여 셀프를 쥐고 있는 변수(self)를 찾아 현재 바이러스에서 사용되고 있는 임의의 변수(variable1)의 이름을 알아내고 그 변수의 이름을 무작위로 생성된 변수(variable2)로 바꾼다. 즉 Replace(self, variable1, variable2) 식의 루틴을 바이러스 내에 삽입하고 원본 바이러스 파일 내의 적절한 위치에 Replace() 함수의 바디를 삽입하도록 한다.
- 임의의 함수 호출 : 임의의 작업을 수행하는 함수(function)의 바디를 미리 만들어 두고 이 함수의 입력으로 Open된 파일을 쥐고 있는 변수(self)를 받도록 하는 call 루틴을 만든다. 즉 function(self)의 루틴을 바이러스 코드에 삽입하고 원본 바이러스 파일 내의 적절한 위치에 function() 함수의 바디를 삽입하도록 한다.

실험을 위해 확보한 스크립트 바이러스의 수는 약 150개 정도였으며, 수집 경로는 인터넷의 해킹 및 바이러스 제작 사이트들이었다. 그러나, 수집된 스크립트 바이러스들은 많은 변종을 포함하고 있고 서로 다른 스크립트 바이러스들도 실제 코드는 유사한 경우가 많았다. 따라서, 본 논문의 실험에서는 가급적 서로 다른 형태의 코드를 가지는 스크립트를 원본으로 하기 위해 수작업을 통하여 유사한 스크립트 바이러스들을 다수 배제하였고, 이러한 필터링을 통해 얻어진 샘플의 개수는 31개였다. 최종적으로, 위의 4가지 방식의 다형성 기능을 이 스크립트 바이러스들에 조합하여 총 124개의 스크립트 바이러스 샘플 집합이 완성되었다. 위의 4기법들은 서로 독립적으로 대상 스크립트 코드에 작용하므로 두 가지 이상의 기법을 하나의 스크립트에 적용하여도 단순히 샘플의 수만 증가할 뿐 실험 결과에 특이한 영향을 미치지는 않는다. 따라서 본 논문에서는 한번에 하나씩의 기법만을 적용하여 샘플 집합을 생성하였다.

4.2 제안된 기법의 다형성 스크립트 바이러스 탐지율

앞 절에서 서술된 실험 환경 및 샘플 집합에 대해 3가지의 상용화된 안티 바이러스와 기존의 기법, 제안된 기법이 보이는 탐지율을 비교하기 위한 실험이 수행되었다. 이 실험이 보여준 탐지 결과는 <표 1>과 같다.

〈표 1〉 상용 안티 바이러스 및 기존 기법과 제안된 기법의 다형성 스크립트 바이러스 탐지

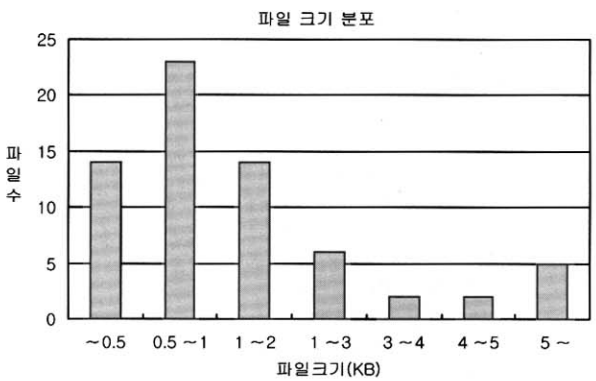
A	B	C	기존 기법	제안된 기법
100(0) / 124	44(24) / 124	28(0) / 124	8 / 124	124 / 124

상용 안티 바이러스의 경우 괄호 안의 숫자는 해당 안티 바이러스가 가지는 휴리스틱을 이용하여 탐지된 바이러스의 수이며 괄호 밖은 탐지된 전체 바이러스의 수이다. A사 제품의 경우 높은 탐지율을 보이는 이유는 방대한 양의 시그너쳐 DB를 가지고 있고 이 시그너쳐 DB가 꼭 발견된 바이러스만 포함하는 것이 아니라 웜 생성기와 같은 프로그램을 실행하였을 경우에 발생 가능한 경우도 고려하기 때문이다. A사 제품이 휴리스틱을 전혀 사용하지 않는 것은 아니지만 현재의 바이러스 샘플에 대해서 적용되는 시그너쳐가 많이 있고 휴리스틱 보다는 시그너쳐가 먼저 적용되므로 휴리스틱이 사용되지 않았음을 알 수 있다. B사 제품의 경우 탐지된 절반 이상의 바이러스가 휴리스틱을 이용하여 탐지된 것이며 C사 제품의 경우 휴리스틱이 사용되지 않고 시그너쳐 DB의 크기도 작다는 것을 알 수 있다. 기존 기법은 셀프를 읽어 들인 변수를 변경하는 경우 복사 전파가 일어나지 않으므로 바이러스를 전혀 탐지 할 수 없어야 하지만 8개의 바이러스가 탐지된 것은 이들 바이러스가 셀프를 읽어 들인 후 저장하는 방식과 객체를 복사하는 메서드를 이용하여 셀프를 그대로 복사하는 방식을 병행하기 때문이다. 기존 기법은 후자의 경우 바이러스임을 탐지

할 수 있다. 제안된 기법은 다형성 바이러스가 지니는 변수의 전달과 변형을 정확히 탐지 하므로 모든 샘플이 바이러스임을 탐지하였다.

4.3 긍정 오류 실험

앞 절에서의 실험과 함께 제안된 기법이 낮은 긍정 오류를 가짐을 보이기 위해 일반적으로 사용되는 정상 스크립트 중 바이러스가 사용하는 것과 동일한 메서드가 포함된 것을 추출하여 앞 절에서와 같은 실험을 반복하였다. 추출된 스크립트는 66개였다. 실험에 이용된 정상 스크립트들은 일반적인 스크립트 바이러스와 유사한 파일 크기 분포를 갖고 있다(그림 8).



(그림 8) 정상 스크립트의 크기 분포

이러한 정상 스크립트 샘플들에 대한 실험 결과는 <표 2>와 같다.

〈표 2〉 기존의 기법과 제안된 기법의 긍정 오류

기존 기법	제안된 기법
0/66	0/66

기존 기법과 제안된 기법 모두 자료 흐름 분석에 의해 각 변수들간의 관계를 고려함으로 낮은 긍정 오류를 가짐을 알 수 있다

5. 결 론

알려지지 않은 스크립트 바이러스에 대응하기 위해서 시그너쳐에 의존하지 않으면서 감지 오류가 적은 바이러스 탐지 기법이 필요하였다. 이를 위해 제시된 것이 자료 흐름 분석을 이용한 휴리스틱 기법으로서, 비교적 높은 정확도와 낮은 긍정 오류로 알려지지 않은 스크립트 바이러스를 탐지한다. 그러나 이 기법은 다형성 스크립트 바이러스에 대응 할 수 없었다. 본 논문에서는 자료 흐름 분석 기반의 기존 휴리스틱 기법을 확장하여 다형성 스크립트 바이러스를 탐지할 수 있는 휴리스틱 기법을 제안하였다. 제안된 기법은 복사 전파를 확장함으로써 자기 수정 행위를 하는 악성 코드에 대해서도 자료 흐름 분석을 가능하게 하여 다형성 스크립트 바이러스를 탐지할 수 있었다. 본 논문에서는 제안된 기법과 기존 기법, 현재 상용되고 있는 안티 바이러스들에 대한 비교 실험을 하였으며 그 결과로 제안된 기법이 다형성 바이러스에 대해서 높은 탐지율과 낮은 긍정 오류를 가짐을 보였다.

참 고 문 헌

[1] Igor Muttik, "STRIPPING DOWN AN AV ENGINE," Proceeding of VIRUS BULLETIN CONFERECE, pp.59-68, 2000.

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

[3] Vesselin Bontchev and Katrin Tocheva, "MACRO AND SCRIPT VIRUS POLYMORPHISM," Proceeding of VIRUS BULLETIN CONFERENCE, pp.406-438, 2002.

[4] Gabor Szappanos, "ARE THERE ANY POLYMORPHIC MACRO VIRUSES AT ALL?," Proceeding of VIRUS BULLETIN CONFERENCE, p.477, 2002.

[5] 이성욱, 배병우, 이형준, 조은선, 홍만표, "정적탐지분석 기법을 이용한 알려지지 않은 악성 스크립트 탐지," 정보처리학회 논문지C, 제9-C권 제5호, pp.765-774, 2002.

[6] 이성욱, "정적 분석과 코드 변환을 이용한 적극적인 악성 스크립트 대응," 아주대학교 박사학위 논문, 2002.

[7] Alex Shipp, "Heuristic Detection of Viruses within Email," 11th International Virus Bulletin Conference, 2001.

[8] Francisco Fernandez, "Heuristic Engines," VIRUS BULLETIN CONFERENCE, pp.407-444, 2001.

[9] Alex Shipp, "Heuristic Detection of Viruses within E-mail," Proceedings of VIRUS BULLETIN CONFERENCE, pp.467-471, 2001.

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

[11] Gabor Szappanos, "VBA Emulator Engine Design," Proceedings of VIRUS BULLETIN CONFERENCE, pp.373-388, 2001.

김 철 민

e-mail : ily@ajou.ac.kr

2000년 아주대학교 정보및컴퓨터공학부 (학사)

2002년 아주대학교 정보통신 전문대학원 정보통신공학과(석사)

2002년~현재 아주대학교 정보통신 전문대학원 정보통신공학과 재학중 (박사)

관심분야 : 정보보호, 알고리즘 등

이 형 준

e-mail : prime@ajou.ac.kr

2002년 아주대학교 정보및컴퓨터공학부 (학사)

2002년~현재 아주대학교 정보통신 전문대학원 정보통신공학과 재학중 (석사)

관심분야 : 정보보호 등

이 성 욱

e-mail : suleeip@shingu.co.kr

1994년 아주대학교 컴퓨터공학과(학사)

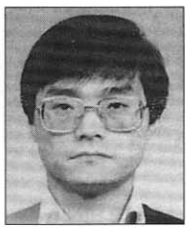
1996년 아주대학교 교통공학과(공학석사)

1996년~1997년 기아정보시스템 지능형 교통시스템팀

2003년 아주대학교 컴퓨터공학과(공학박사)

2003년~현재 신구대학 인터넷정보과 전임강사

관심분야 : 병렬처리, 컴퓨터 보안



홍 만 표

e-mail : mphong@ajou.ac.kr

1981년 서울대학교 자연과학대학 계산통계학과(학사)

1983년 서울대학교 자연과학대학 계산통계학과(석사)

1991년 서울대학교 자연과학대학 계산통계학과(박사)

1983년~1985년 울산공과대학 전자계산학과 전임강사

1993년~1994년 미네소타대학 전자공학과 교환교수

2000년~2001년 조지와싱턴대학교 컴퓨터과학과 교환교수

1985년~현재 아주대학교 정보및컴퓨터공학부 교수

관심분야 : 병렬처리, 컴퓨터 보안