情報保護學會論文誌
第12券 第3號, 2002. 6

메서드 치환을 이용한 악성 자바 애플릿 탐지 기법

이 승 수*, 오 형 근*, 배 병 철*, 고 재 영*, 박 춘 식*

A Technique for Detecting Malicious Java Applet Using Java-Methods Substitution

Seung-soo Lee*, Hyung-geun Oh*, Byung-chul Bae*,
Jae-young Koh*, Choon-sik Park*

요 약

웹 서버에서 HTTP 프락시 서버를 경유하여 사용자의 웹 브라우저에서 실행되는 자바 애플릿은 클라이언트의 파일이나 자원에 접근할 수 있어 악성 자바 애플릿에 대한 보안이 요구되고 있다. 기존의 악성 자바 애플릿에 대한 보안 대책으로는, 프락시 서버에서 알려진 악성 자바 애플릿을 탐지할 수 있는 필터를 구축하는 방법과 별도의 보안 자바 가상머신을 구축하는 방법을 사용하였다. 필터를 이용한 보안대책은 알려지지 않은 악성 자바 애플릿을 탐지할 수 없으며, 보안 자바가상머신을 이용한 보안대책은 프락시 서버에서 자바 애플릿을 실행시킨 후 악성 판정을 하기 때문에 프락시 서버의 부하가 가중된다. 본 논문에서는 클라이언트에서 자바 애플릿을 실행시키지만, 메서드 치환 기법을 이용하여 프락시 서버에서 자바 애플릿에 감시 기능을 삽입한 후, 사용자에 전달하여 악성 자바 애플릿을 탐지하도록 하여 서버의 부하를 줄이면서 알려지지 않은 악성 자바 애플릿을 탐지할 수 있는 악성 탐지 기법을 제안한다.

ABSTRACT

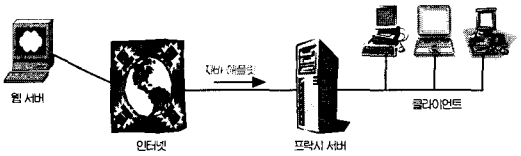
Java applet, executed in user's web browsers which is via proxy server on web sever, can approach client files or resources, so it is necessary to secure against malicious java applet. Currently, the previous security countermeasures against malicious java applet use two ways: one is making a filter system to detect malicious java applet known in proxy, the other is that establishes another security java virtual machine. However, the first one can not detect unknown malicious java applet, and the other one may increase loads, because it decides whether there is malicious or not after implementing java applet on proxy server. In this paper, after inserting monitoring function to java applet on proxy server using java-methods substitution and transfer it to user to detect malicious java applet, we propose a technique for detecting malicious java applet that can detect the unknown malicious java applet with reducing loads

Keyword : *Java Applet, Java-Methods Substitution*

Ⅰ. 서 론

악성 자바 애플릿과 같은 이동실행코드에 의한 피해는 코드레드와 같이 치명적인 피해 사례로 보고되진 않았지만 인터넷 사용의 증가와 함께 무수히 많은 웹 서버가 존재하게 되면서 그 피해 사례가 늘고 있다.[1,2] 사용자가 임의의 웹 서버에 접속하여 각종 자바 애플릿을 프락시 서버를 통해 전송받는 경우가 많아지고 있기 때문이다. 악의의 웹 서버가 그 서버에 접속하는 모든 불특정 다수의 개인 정보 획득, 시스템 정보 획득 및 시스템 파괴를 목적으로 접속한 사용자에게 악성 자바 애플릿을 전송하여 자동으로 실

* ETRI부설 국가보안기술연구소 응용기술연구부(kadan@etri.re.kr)



〔그림 1〕 자바 애플릿과 프락시 서버.

행될 수 있도록 한다면 그 피해가 크다. 현재 발견된 악성 자바 애플릿의 침해는 크게 4가지 유형으로 분류할 수 있으며, 유형별로 살펴보면, 사용자의 자원을 불법적으로 획득하는 비밀성 침해, 사용자의 시스템 자원을 불법적으로 수정 및 변경하는 무결성 침해, 사용자 시스템의 자원을 과도하게 사용하여 사용자의 정상적인 사용을 방해하는 가용성 침해, 그리고 윈도우의 화면 독점 등으로 사용자의 작업을 방해할 수 있는 사용자불편 침해이다.

웹 페이지와 함께 전송되는 자바 애플릿은 〔그림 1〕과 같이 웹 서버에 접속한 사용자에게 전송되기 전에 HTTP 프락시 서버를 경유한다. 따라서 악성 자바 애플릿의 침해를 방지하기 위해서는 사용자마다 악성코드를 탐지할 수 있는 도구를 설치할 수 있지만, 프락시 서버에서 악성 자바 애플릿을 사전에 탐지하고 제거할 수 있는 방법이 더 비용 효과적이다. 기존의 프락시 서버에서 악성 자바 애플릿에 대한 대책은 악성 자바 애플릿을 가지고 있는 유해한 사이트를 차단하거나 이미 악성으로 알려진 자바 애플릿에 대한 패턴 정보를 DB로 관리하여 사용자가 요청한 자바 애플릿에 대해 사전 필터링을 수행하는 것이었다. 기존 대책에서는 알려지지 않은 악성 자바 애플릿에 대해서는 사전에 탐지할 수 없었다. 또한 기존의 프락시 서버는 HTTP 기능뿐만이 아니라 SMTP, FTP 기능 등을 같이 수행하고 있기 때문에 프락시 서버에서 악성 자바 애플릿 탐지 및 제거를 위한 기능을 추가시킬 경우 부하가 가중되어 전송 속도가 떨어지고 사용자의 불편을 유발할 수 있다.

따라서 본 논문에서는 악성 자바 애플릿의 침해를 사전에 탐지하여 피해를 방지할 수 있으면서 프락시 서버의 부하를 가중시키지 않는 악성 자바 애플릿에 대한 대책으로, 프락시 서버에서의 악성 탐지기 구조와 감시 기능을 위한 메서드 치환 기법을 제안하며, 제안한 방법은 악성으로 의심스러운 자바 애플릿에 감시 기능을 수행할 수 있는 코드로 치환한 후 클라이언트에서 수행되도록 하여 프락시 서버의 부하를 줄일 수 있다.

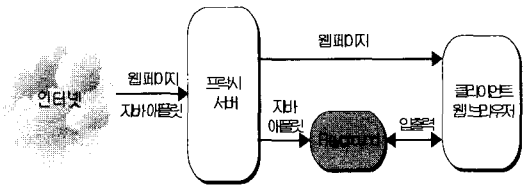
Ⅱ. 관련 악성 자바 애플릿 탐지 기법

자바 애플릿의 보안은 Sandbox를 통해 이루어지기 때문에 다른 시스템에 비해 비교적 안전하지만 악성 자바 애플릿이 사용자의 시스템에 피해를 끼칠 수 있는 취약점이 학계를 통해 계속 보고되고 있으며, 자바 1.2 버전부터는 자바 애플릿에 서명 값을 추가하여 그 보안성을 강화하였다.[3,4] 그러나 악의의 웹 서버를 통한 악성 자바 애플릿의 제작 및 유포가 점점 증가하고 있는 추세이어서 더욱 더 악성 자바 애플릿 보안 대책이 요구되며 이에 대한 탐지 전용 도구 개발도 필요하다.

기존의 악성 자바 애플릿 전용 탐지 도구들을 살펴보면, Finjan 사는 서버용과 클라이언트용의 악성이동코드 탐지 도구 제품을 출시하였으며, 이 기술은 보안성을 강화한 자체 보안 Sandbox(SafeZone)를 통해 악성 자바 애플릿을 탐지할 수 있도록 하였다.[5~8] Trend Micro 사에서는 스캐너 구조를 제안하여 HTTP 프락시 서버에서 악성 자바 애플릿을 탐지할 수 있는 도구를 개발하였다.[9] 프락시 서버의 탐지 도구에서 악성 자바 애플릿을 탐지하기 위해 사용된 기존의 악성 탐지 기법에는 Playground 기법과 스캐너 기법 등이 있다.[7,8,10~12]

2.1 Playground 기법

Playground 기법은 〔그림 2〕와 같이 프락시 서버에서 악성 자바 애플릿을 클라이언트에게 전달하기 전에 별도로 분리된 Playground에서 자바 애플릿을 실행시켜 보고 악성 여부를 판정하여 웹 브라우저에 알려주는 기법이다. 동작 원리는 클라이언트의 시스템 자원을 접근할 수 없는 다른 곳(Playground)에서 자바 애플릿을 실행하고, 자바 애플릿 실행코드는 제한된 환경으로부터의 클라이언트와 상호작용을 하도록 하는 것이다. 이것은 프락시 서버에서 Playground를 통해 자바 애플릿을 실행시킴으로써 이루어질 수 있다.



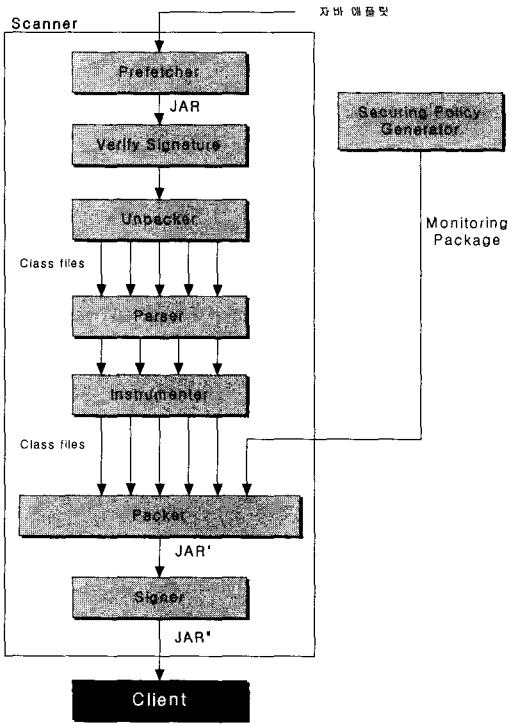
〔그림 2〕 Playground 기법

[그림 2]와 같이 구성 단계에서 이와 같은 보호가 이루어져야 하기 때문에 기존의 시스템과는 구별되는 별도의 Playground 장치가 필요하다. 또한 자바 애플릿이 실행되는 동안 클라이언트와 상호 작용을 해야 하기 때문에 클라이언트는 마치 그래픽 터미널처럼 작동하게 된다. 하지만 어느 시점에서도 자바 애플릿은 클라이언트의 웹 브라우저에서는 직접 실행되지 않고 Playground라는 공간상에서 실행이 되며 단지 입출력 결과만을 클라이언트에 보여주기 때문에 클라이언트 시스템에 직접적인 피해를 일으킬 수 없다. 하지만 이 기법의 단점은 입출력 결과를 알려주기 위한 웹 브라우저와 서버간의 별도의 통신 프로토콜이 설계되어야 하는 것이다. 또한 이러한 물리적인 분리를 통해 웹의 유연성을 지원하기 위해 만들어진 자바 애플릿의 이동성을 제한할 수 있으며, 프락시 서버의 부하를 가중시킬 수 있다. Digitivity사의 Cage 제품은 Playground 기법을 이용한 것으로 Playground를 위한 CageServer를 두고 있다.[2]

2.2 스캐너 방법

스캐너 기법은 Playground와 같은 기법을 이용할 경우 프락시 서버에 부하가 가중되는 문제점을 해결할 수 있는 방법이다. [그림 3]은 기존의 악성 자바 애플릿 감시를 위한 스캐너 구조로서 그 절차를 살펴보면 먼저 스캐너의 입력으로 서명된 자바 애플릿을 받아 Prefetcher를 통해 클래스 형태의 자바 애플릿을 하나의 JAR파일로 압축하고 서명을 검사한다. 서명을 검사한 후 Unpacker를 통해 클래스 파일로 다시 변환하고 각각의 클래스 파일들은 Parser를 통해 각각의 인스트럭션 형태로 만들어 Instrumenter를 통해 의심스러운 행위를 수행할 수 있는 인스트럭션에 대해서는 앞뒤에 감시용 코드(Pre- and Post- filter)를 삽입하고, 보안정책에 맞는 모니터링 패키지를 덧붙여 Packer를 통해 다시 하나의 JAR′ 파일로 압축한다. 스캐너를 통해 원래의 자바 애플릿이 변환되었기 때문에 서명을 다시 하여 클라이언트로 전달한다. 이렇게 재서명된 자바 애플릿은 클라이언트에서 실행되며 악성 여부를 판정하게 된다.

스캐너 기법의 장점은 악성 자바 애플릿을 탐지하기 위해 감시 코드를 삽입하여 클라이언트에서 실행되도록 하기 때문에 프락시 서버에서의 부하 가중을



[그림 3] 악성 자바 애플릿 탐지용 스캐너 구조

막을 수 있다는 점이다. 그러나 트랜드 마이크로사의 프락시 서버의 스캐너 기법은 서명된 자바 애플릿에 한정하여 적용되며, 특허로 등록되어 보호되고 있고 구체적인 핵심 구현기술은 공개하지 않고 있기 때문에 스캐너 기법을 이용한 활용이 어렵다.[10] 또한 스캐너 구조가 복잡하기 때문에 탐지에 시간이 많이 소요되는 단점이 있다.

Ⅲ. 제안한 악성 탐지기와 메서드 치환 기법

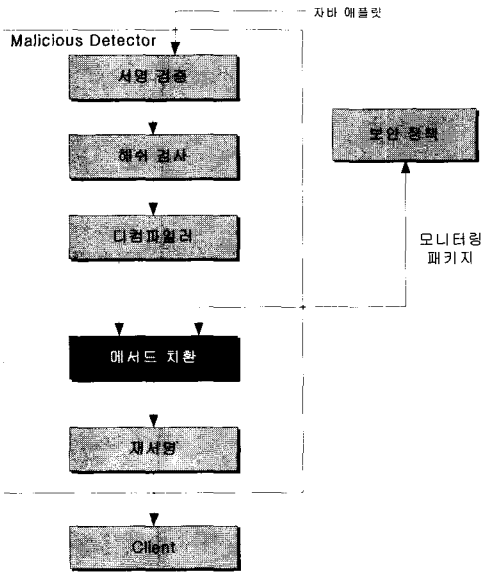
본 장에서는 제안한 악성 탐지기와 메서드 치환 기법에 대해 알아본다. 악성 탐지기는 프락시 서버에 부하를 가중시키지 않도록 클라이언트에서 실행되도록 하며 악성 행위를 탐지하기 위한 감시 기능을 메서드 치환 기법을 이용하였다. 먼저 제안한 악성 탐지기의 구조를 살펴보고 메서드 치환 기법에 대해 설명한다.

3.1 악성 탐지기 구조

제안한 악성 탐지기를 구현하기 위해서는 악성 자바 애플릿 탐지를 위한 악성 탐지기의 각 절차를 설

계하여야 한다. 자바 애플릿이 바이트 코드 형태의 클래스 파일로 프락시 서버를 경유하기 때문에 먼저 메서드 치환을 이용한 감시 코드를 삽입하기 위해서는 [그림 4]와 같이 디컴파일러가 필요하다. 메서드란 자바 소스 코드에 사용되는 함수를 의미하여 자바가 제공하는 패키지의 클래스이다. 디컴파일러의 수준은 메서드 레벨 수준으로 치환할 메세드들의 목록을 찾을 수 있는 수준이어야 한다. 디컴파일러를 통해 생성된 코드를 분석하여 의심스러운 행위 가능한 메서드 부분을 치환하여야 한다. 치환 메서드 목록은 침해 유형별로 무결성, 가용성, 비밀성, 사용자 불편으로 분류하여 각각의 해당 메서드를 관리하고 치환할 메서드는 악성행위에 대한 감시 기능을 수행할 수 있는 형태로 관리한다. 또한 치환 메서드를 작성하기 위해 대상 메서드의 치환용 코드를 작성하여야 한다.

[그림 4]는 본 논문에서 제안한 악성 탐지기 동작 절차를 나타낸 것으로 프락시 서버에 들어온 클래스 파일에 대해 서명 검증과 해쉬 검사를 수행하고 디컴파일러를 수행하여 치환할 메서드가 있는지를 찾고 치환할 메서드가 있을 경우에는 메서드를 치환하고 치환한 메서드에 감시용 코드를 삽입해야 할 경우에 설정된 보안정책에 따라 모니터링 패키지를 삽입하고 재서명을 통해 클라이언트에 전달한다. 본 논문에서는 침해유형별 가중치를 두어 3단계의 보안정책(사용안함, 높음, 매우높음)을 적용하였다.

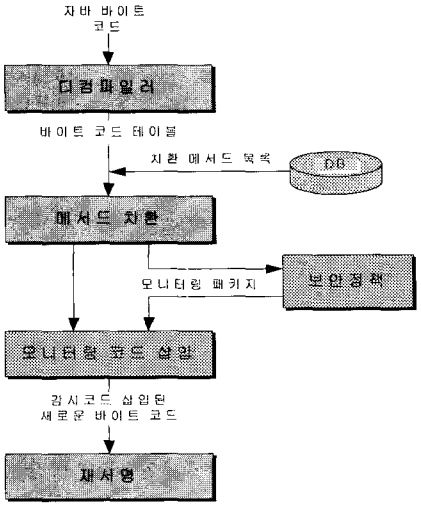


[그림 4] 제안한 악성 탐지기 구조

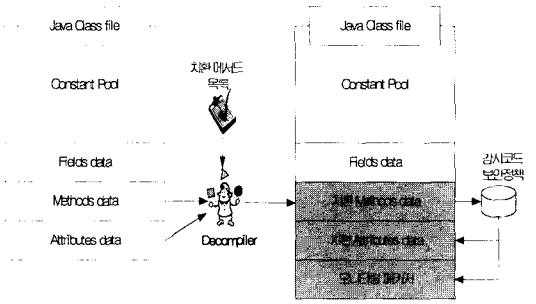
3.2 메서드 치환 기법

본 논문에서 제안하는 악성 자바 애플릿 탐지를 위한 프락시 서버에서의 메서드 치환 기법은 메서드 치환 기법이 적용되는 시스템 절차와 모니터링 코드 삽입을 수행하는 절차로 크게 나누어 살펴본다. [그림 5]는 [그림 4]의 메서드 치환 부분에 대한 상세 그림으로써, 본 논문에서 제안한 메서드 치환 및 모니터링 코드의 삽입 절차를 나타낸 것이다. 먼저 자바 바이트 코드를 디컴파일러를 통해 치환할 대상 메서드를 찾고 해당 메서드는 치환 메서드 목록에 있는 메서드로 치환한다. 치환 메서드에는 악성 행위를 탐지할 수 있는 감시용 코드와 악성 행위를 일으키지 않을 경우 원래의 메서드를 호출하는 코드가 있다. 치환된 메서드에서 악성 여부를 판단하기 어려운 경우, 모니터링 감시 코드에 정보를 넘겨주고, 보안정책에 따라 각 클래스마다 모니터링 패키지를 삽입한다. 모니터링 코드 삽입에서는 프로세스 상태 전이나 자원 사용량을 감시하여, 악성코드 여부를 판별하고 처리한다. 감시 코드에는 악성으로 판정이 되었을 경우 프락시 서버에 보고할 수 있는 코드가 있다. 감시코드가 삽입된 새로운 코드에 대해서는 재서명을 수행하여 클라이언트에 전달한다.

[그림 6]은 자바 애플릿 실행코드 배열을 나타낸 것으로 왼쪽은 원래 전송된 자바 클래스 파일이고 오른쪽은 메서드 치환을 통해 새롭게 만들어진 자바 클래스 파일의 배열을 나타낸 것이다. [그림 5]의 메서드 치환 기법이 적용되면 [그림 6]의 오른쪽과 같은



[그림 5] 메서드 치환 기법



〔그림 6〕 메서드 치환에 의한 자바 바이트 코드

자바 실행코드가 된다. 〔그림 6〕에서와 같이 바이트 코드 형태의 클래스 파일의 배열에서 자바 애플릿의 행위와 관련된 Methods(메서드)와 Attributes(속성) 데이터에서 디컴파일러를 통해 치환할 메서드나 속성이 있는지를 찾고 치환 메서드 목록에 있을 경우 감시 기능을 수행할 수 있도록 해당 메서드와 속성을 치환하고 보안정책이 반영된 모니터링 패키지를 삽입한다.

3.3 메서드 치환 기법 설계

메서드 치환 방법은 먼저 악성 자바 애플릿의 침해 유형별로 메서드들을 분류하고 이에 대한 치환 메서드를 생성해 놓은 후 디컴파일러를 통해 나온 소스에서 치환할 메서드가 있는 경우 해당 메서드를 치환한다. 치환할 메서드에는 자체적으로 악성 여부를 판별 및 처리 가능한 메서드도 있고 악성 판별 및 처리를 위해 정보를 전달해야 하는 메서드도 있다. 메서드 치환 방법을 예제를 통해 살펴보면 다음과 같다.

① 사용자불편 유발 가능한 메서드 치환

윈도우 크기가 비정상적으로 확대된 자바 애플릿은 화면 독점으로 인한 클라이언트의 작업을 방해할 수 있는 악성 자바 애플릿이다. 윈도우 크기를 변경하는 메서드인 setSize()는 그 입력값이 너무 큰 경우 화면 독점으로 인한 작업 방해를 유발시킬 수 있는 메서드이다. 이를 감시하기 위해 메서드 치환으로 새로운 monitor_setSize() 메서드를 만들어 해당 메서드 부분을 치환한다. 치환된 monitor_setSize()는 파라미터 값을 검사한 후 이상이 없으면 원래의 setSize() 메서드를 호출한다. 다음 글상자의 모의 코드는 monitor_setSize() 치환 메서드의 작성 예제이다.

```
monitor_setSize()
{
모니터 크기에 대한 정보를 획득한다;
if(자바애플릿이 모니터 화면보다 크다면) {
자바애플릿의 화면을 재조정한다;
} else {
      원래의 setSize() 메서드를 호출한다;
}
}
```

② 무결성 침해 가능한 메서드 치환

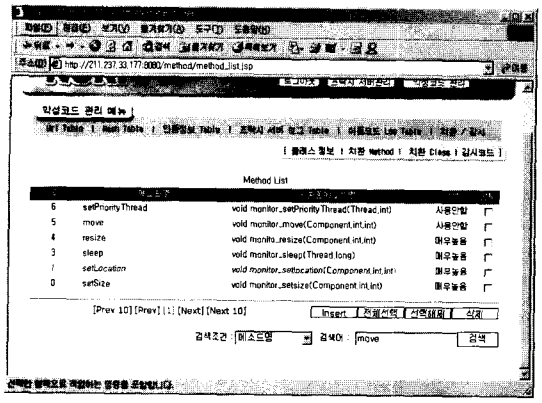
파일 접근을 통해 자료의 복사, 삭제 등의 공격이 행해질 수 있는 java.io 패키지의 클래스 종류들을 살펴보면 File, FileDescriptor, StreamTokenizer, InputStream, OutputStream, RandomAccessFile 등이 있다. 이를 메서드 치환으로 감시코드를 삽입할 수 있다. 디컴파일된 메서드 코드에 악성을 유발할 수 있는 FileInputStream() 메서드가 존재할 경우 monitor_FileInputStream() 메서드로 치환하여 만약 악성 행위를 수행할 경우에는 설정한 보안정책에 따라 SecurityException() 메서드를 호출하고, 그렇지 않은 경우에는 원래의 FileInputStream() 메서드를 호출하게 된다. 다음 글상자의 모의 코드는 monitor_FileInputStream() 치환 메서드의 작성 예제이다.

메서드 치환용 코드를 작성할 때 고려사항은 실시간으로 치환 기법이 이루어져 자바 애플릿을 요청한 클라이언트가 불편을 느끼지 않도록 해야 한다.

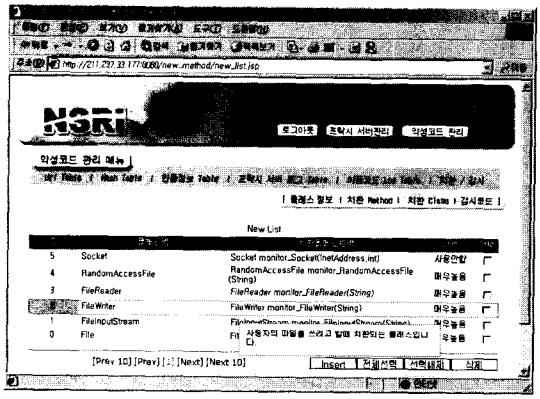
```
monitor_FileInputStream() {
try{
if(설정한 보안정책에 위반되는 파일 접근시){
보안 정책에 따라 경고 에러 메시지를 띄우고 실행을
중지시킨다;
New  SecurityException();
}else {
원래의 FileInputStream() 메서드를 호출한다;
}
}catch (SecurityException ){
alert();
}
}
```

3.4 구현 및 시험

본 논문에서 제안한 메서드 치환 기법의 구현 결과를 살펴보면 〔그림 7, 8〕과 같으며, 치환 대상 메서드는 감시코드가 삽입된 새로운 메서드로 치환한다.

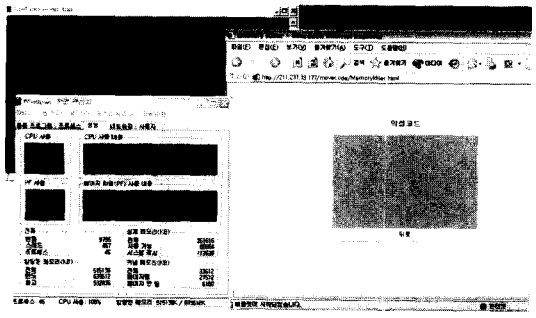


〔그림 7〕 사용자불편 메서드 치환 구현

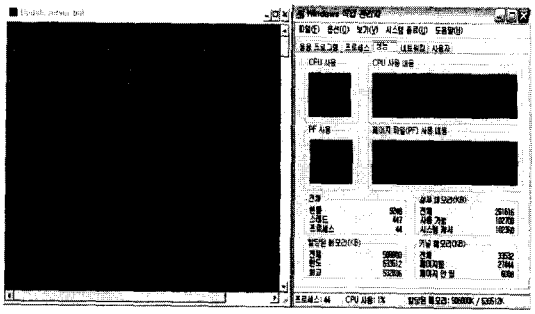


〔그림 8〕 무결성 침해 메서드 치환 구현

메서드 치환을 통해 감시 기능이 삽입된 자바 애플릿이 클라이언트에서 CPU의 메모리를 점유할 때 탐지하는 과정을 살펴보면 〔그림 9, 10〕과 같다. 〔그림 9〕는 Memorykiller 악성 자바 애플릿에 감시 기능이 있는 Memorymonitor 메서드를 삽입하여 클라이언트에서 자바 애플릿을 처리하는 화면이며, 〔그림 10〕은 감시 기능에 의해 악성 자바 애플릿을 탐지한 후 그 악성 정보를 프락시 서버에 전송하는 화면이다.



〔그림 9〕 감시 기능이 삽입된 악성 자바 애플릿



〔그림 10〕 악성 자바 애플릿 탐지 결과

Ⅳ. 결 론

본 논문에서 제안한 기법은 프락시 서버에서 알려지지 않은 자바 애플릿에 대해 악성 여부를 판정하기 위해 메서드 치환 기법을 통해 감시 기능을 자바 애플릿에 삽입하고 클라이언트를 통한 탐지 보고를 수행하도록 하여 프락시 서버의 부하 가중을 일으키지 않도록 한 기법이다. 본 논문을 통해 자바 애플릿의 행동 모니터링을 할 수 있는 메서드 치환 기법을 제안하였으며, 기존의 감시 코드를 삽입하는 스캐너 기법과 비교하면 감시 기능을 수행하기 위한 탐지 절차 및 기법이 다르며 또한 바이트 코드 형태에서 디컴파일러를 통해 메서드 부분을 찾아 치환하기 때문에 구조적으로 단계가 줄어 보다 효율적으로 탐지할 수 있도록 하였다. 악성 자바 애플릿의 기존 탐지 기법과 본 논문에서 제안한 메서드 치환 탐지 기법을 비교하면 〔표 1〕과 같다.

〔표 1〕 악성 탐지 기법 성능 비교

항목 / 탐지 기법	구현 난이도	탐지 대상	동작 속도	비 고
Playground 기법	복 잡	모든 자바 애플릿	준실시간 탐지	서버 부하 큼
스캐너 기법	복 잡	서명 자바 애플릿	실시간 탐지	구조 복잡
메서드 치환 기법	용 이	모든 자바 애플릿	실시간 탐지	※ 제안한 기법

본 논문에서 제안한 메서드 치환을 이용한 악성 탐지 기법은 기존의 탐지 기법들과 비교하여 구현 난이도와 탐지 대상 및 동작 속도에서 우수하다. 메서드 치환 기법을 이용한 악성 탐지 기법은 특허로 출원하였으며, Visual C++과 Java 언어를 통해

핵심 모듈로 개발하였다.

향후 상용화를 위해서는 악성 자바 애플릿의 4가지 침해 유형별로 좀 더 많은 치환 대상 메서드에 대한 체계적인 분류가 필요할 것 있으며, 감시 기능이 중복되어 삽입되지 않도록 보안정책을 설정하는 것이 필요할 것으로 사료된다.

참 고 문 헌

[1] 이정효, "Executable Contents 보안," http://www.kisa.or.kr/.

[2] 이병각, "이동코드 보안," http://www.kisa.or.kr/.

[3] 임영주, "자바 시큐리트에 대한 고찰," http://www.javastudy.co.kr/docs/.

[4] 이인영, "Java Security와 Cryptography Architecture," http://www.javastudy.co.kr/docs/.

[5] Surfingate, http://www.finjan.com/product_home.cfm, *Data Sheet*.

[6] SurfinShield, http://www.finjan.com/product_home.cfm, *Data Sheet*.

[7] Shlomo Touboul and Nachshon Gal, "System and Method for Attaching a downloadable Security Profile to a downloadable," *Patent US6154844*, Nov. 28, 2000.

[8] Shlomo Touboul, "Sytem and Method for Protecting a Computer and a Network from Hostile Downloadables," *Patent US6092194*, Jul. 18, 2000.

[9] InterScan AppletTrap, http://www.antivirus.com/products/isat/, *Data Sheet*.

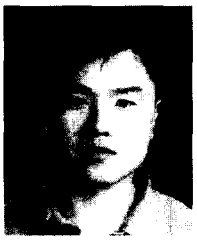
[10] Shuang Ji, Santa Clara and Calif, "Computer Network Malicious Code Scanner," *Patent US5983348*, Nov. 9, 1999.

[11] Gilad Golan, "Security Monitor," *Patent US5974549*, Oct. 26, 1999.

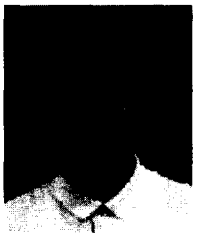
[12] Ajay Chander, John C.Mitchell and Insik Shin, "Mobile Code Security by Java Bytecode Instrumentation," *IEEE*, 2001.

〈著者紹介〉

이 승 수 (Seung-soo Lee) 정회원
1995년 2월 : 전북대학교 전자공학과 졸업
1997년 2월 : 전북대학교 전자공학과 석사
2002년 2월 : 전북대학교 전자공학과 박사
2001년 3월~현재 : ETRI부설 국가보안기술연구소 선임연구원
〈관심분야〉 이동통신보안, 정보보호, 회로 설계



오 형 근 (Hyung-geun Oh) 정회원
1998년 2월 : 순천향대학교 전산학과 졸업
2000년 2월 : 순천향대학교 전산학과 석사
2000년 2월~2000년 8월 : (주)한국사이버페이먼트
2000년 8월~현재 : ETRI부설 국가보안기술연구소 연구원
〈관심분야〉 악성코드, 네트워크보안, 전자화폐



배 병 철 (Byung-chul Bae) 정회원
1994년 2월 : 홍익대학교 컴퓨터공학과 졸업
1996년 2월 : 홍익대학교 전자계산학과 석사
1996년 1월~1999년 1월 : 국방정보체계연구소
1999년 1월~2000년 1월 : 국방과학연구소
2000년 2월~현재 : ETRI부설 국가보안기술연구소 선임연구원
〈관심분야〉 정보보증, 시스템공학

고 재 영 (Jae-young Koh) 정회원
1984년 2월 : 전북대학교 전자공학과 졸업
1992년 2월 : 전북대학교 전자공학과 석사
1998년 8월 : 전북대학교 전자공학과 박사
1984년 3월~2000년 1월 : 국방과학연구소
2000년 2월~현재 : ETRI부설 국가보안기술연구소 부장, 책임연구원
〈관심분야〉 네트워크보안, 정보보증

박 춘 식 (Choon-sik Park) 종신회원
1981년 : 광운대학교 전자통신공학과 졸업
1983년 : 한양대학교 전자통신공학과 석사
1995년 : 일본 동경공업대학 전기전자공학과 박사
1982년~1999년 : 한국전자통신연구원 부장, 책임연구원
2000년~현재 : ETRI부설 국가보안기술연구소 연구위원, 책임연구원
2002년~현재 : 한국정보보호학회 국제이사
〈관심분야〉 암호이론, 전자투표, 보안정책