안드로이드 기반 자바 어플리케이션 보안 취약성 개선

박 성 준*, 김 정 웅*, 양 해 술*
*호서벤처전문대학원 컴퓨터응용기술학과
e-mail : popo8402@hanmail.net, jwk@korea.com, hsyang@office.hoseo.ac.kr

Java Application Security Vulnerability Improvement based on Android

Sung June Park,*, Jeong Woong Kim*, Hae Sool Yang*
*Dept. Application of Computer Technology,
Hoseo Graduate School of Venture

요 약

최근 들어 안드로이드폰, 아이폰, 옴니아폰 등의 스마트폰 사용자가 많이 증가하고 있고 전자상거래 등 그 이용분야 또한 점점 증가되고 있는 추세이다. 많은 사용자들이 스마트폰 환경에 접어들면서 삶의 질이 향상되어지고 있다. 하지만 이렇게 모바일 환경으로 변화되면서 여러 가지 많은 문제점도 발견되어지고 있는데 그중 대표적인 이슈로 논의되어지고 있는 문제 중 하나가 스마트폰에서의 보안취약성에 대한 내용인데 본 논문은 스마트폰 중 안드로이드 환경에서의 보안문제를 짚어보고 악성코드 같은 바이러스로부터 이를 보호하기위해서는 어떤 점이 필요한지 개선안을 제시하고 이를 해결하고자 한다.

1. 서론

최근 스마트폰 시장에서는 Google 안드로이드 환경이 보안에 취약하다는 지적이 제기되고 있다. 스마트폰 이용자가 급속히 증가하면서 안드로이드에 대한 보안문제에 관한 관심 또한 높아지고 있다.

안드로이드에 대한 보안문제로는 비교적 보안에 강한 유선환경과 전달매체, 기술, 디바이스가 현저히 다르다는 것과 안드로이드 프로그램 개발 시 사용되는 Java언어로 만들어진 프로그램이 악성코드의 위협으로부터 자유롭지 못하다는 것 등이 있다.

안드로이드 어플리케이션 개발환경은 Java 개발환경과 비슷한 점이 많이 존재하는데 그중 대표적으로 보안에 다소 취약한 Java 가상머신과 비슷한 Dalvik 이라는 가상머신을 사용한다는 것이 있다.

이와 관련해 많은 연구자들은 악성코드 유입 등 보안의 위협으로부터 안전하지 않은 Java 가상머신과 비슷한 Dalvik 가상머신 또한 보안에 문제가 있지 않겠느냐라는 이의를 제기하고 있다.

본 논문에서는 위와 같은 문제와 관련하여 안드로이드가 보안으로부터 자유로워 지기위한 개선방법을 모색하고자 한다.

2. Dalvik VM

안드로이드에 사용되는 Dalvik VM은 레지스터 기반으로 스택 기반의 Java와 형태는 다르지만 실제 명령은 Java와 거의 일대일로 대응되는 구조다.

Dalvik VM은 모바일 디바이스용으로 최적화 되도록 구현되어 있으며 안드로이드에서 사용되는 모든 코드는 이 Dalvik VM내에서 실행이 이루어진다.

Dalvik VM은 Java VM에 대비해 다음과 같은 중요한 두 가지 차이점이 있다. 첫째, Dalvik VM은 .dex 파일을 실행시키는데 이 파일은 표준 .class와 .jar 파일이 컴파일 시점에 변환된 것인데 이 .dex 파일은 클래스 파일보다 콤팩트하고 효율성이 높기 때문에 안드로이드가 사용될 적은 메모리와 배터리로 작동하는 환경에 적합하다는 것과 둘째, 안드로이드에 들어있는 코어 Java 라이브러리는 Java Standard Edition Library와 Java Mobile Edition Library 와는 다르지만 중복되는 부분이 상당히 있다는 것이다.

즉, 다시 말해 Dalvik VM은 Java Virtual Machine과 차이는 있지만 상당부분 비슷한 부분이 있다는 것이라고 볼 수 있다.

3. 자바의 보안 취약성

지금 현재까지 Java는 다양한 취약사항이 발견되었고 개선되어왔지만 아직까지 미비한 보안취약점이 존재하고 있다. 가장 큰 보안 문제로는 외부 시스템에서 만들어진 프로그램이 내 컴퓨터에서 실행되는 것인데 그중

웹브라우저에서 Local과 Remote 환경으로 실행되는 Java Applet 프로그램은 각종 Game, Utility 들로 널리 이용되고 있다. Java Applet의 보안의 취약점은 크게 3가지로 나뉘어서 살펴볼 수 있다.

① applet developer 에 의한 보안 위협
② applet transmission 시의 보안 위협
③ applet executable 시의 보안 위협

첫째, applet developer에 의한 보안위협으로는 Applet에 트로이목마 등의 악성코드를 삽입하거나 Applet이 실행되는 컴퓨터의 자원에 불법 access 기능을 삽입하는 행위 그리고 시스템 수행에 방해되는 기능이 담긴 코드를 삽입하여 갑자기 시스템이 느려지게 하는 행위 마지막으로 불쾌감을 유발하는 화면을 담아서 사용자를 위협하고 사운드를 생성하는 행위 등이 있다.

둘째, applet transmission 시의 보안 위협으로는 Applet을 전송하는데 방해나 가로채기를 한다거나 다른 Applet으로 바꾸는 위·변조 등의 행위가 있다.

셋째, applet executable 시의 보안위협으로는 server의 정보유출(disclosure)과 불법 수정·복제 행위가 있다.

4. 모바일의 보안 취약성

기존의 유선 네트워크와 마찬가지로 무선인터넷, 디지털 방송 등을 이용할 수 있는 모바일환경 또한 개방형 OS를 탑재하고 있는 특성상 여러 가지 보안위협에 노출되어 있다. 이러한 무선 인터넷을 위한 기술들은 유선환경과는 달리 전혀 다른 전달매체와 기술, 디바이스, OS등을 가지고 있기 때문에 유선과는 차별화된 보안 기술이 필요하게 된다. 무선 환경은 유선보다는 그 특성상 도청, 감청, 위변조에 한층 더 취약성을 내포하고 있으며, 최근 휴대용 단말기들에 대한 악성코드들이 급속도로 증가하고 있는 추세이고 무선 환경에서 동작되는 단말기 중 스마트폰이 주요대상이 되고 있다. 이는 수년전부터 기존의 컴퓨터와 서버들을 대상으로 하던 악성코드들의 위협이 모바일 기기들을 대상으로 급속도로 발전되고 있는 것 이라고 다시 말할 수 있다. 모바일 악성코드들은 각각의 모바일 단말기들뿐만 아니라 서비스를 제공하는 서버 또는 Sync를 수행하는 Desktop 컴퓨터에도 피해를 줄 수 있으며, 이는 전체 네트워크로의 전파도 우려되고 있어 새로운 위협요소로 떠오르고 있다.

또한 스마트폰의 경우 와이파이(WiFi) 기능이 기본 탑재되어 있어서 공중 또는 사설 무선랜 사용빈도가 높은데 이에 대한 보안 대책이 유선 인터넷에 비해 상대적으로 허술하다는 것도 위협요소의 원인으로 꼽히고 있다. 최근 스마트폰 금융거래 빈도가 높아지는데다 전화번호나 사진 등 민감한 개인정보가 많은 주소록 등의 개인정보유출이 우려되는 민감한 상황이 계속 이어지고 있고 심지어 다른 사람의 문자메시지(SMS)를 훔쳐본다거나 타인의 스마트폰 번호로 인터넷 상품을 구매하는 피해사례들이 늘어남에 따라 모바일 보안취약성에 따른 개선안 마련이 시급한 상황이다.

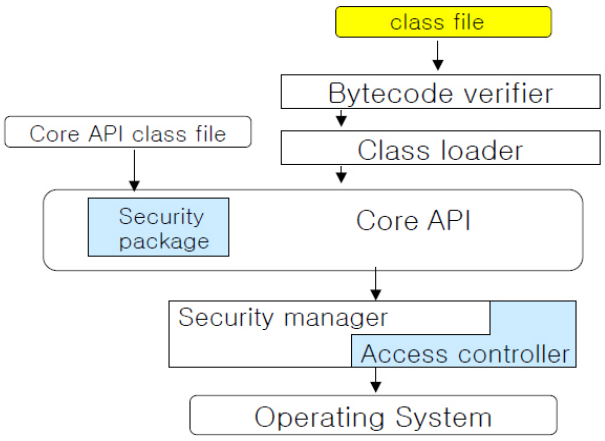
5. 안드로이드의 보안 취약성

Java의 가상머신인 'JVM'과 유사한 리눅스 커널 기반에서 동작하는 Dalvik 가상머신을 사용하는 안드로이드는 휴대성 등의 특성으로 인하여 보안에 취약할 수밖에 없다. 안드로이드가 보안에 약한 이유를 더 자세히 살펴보면 애플리케이션에 대한 검증 절차가 없다는 것과 멀티태스킹이 가능한 환경을 악용한 악성코드 삽입 등 안드로이드는 현재까지 보안에 많이 취약한 상태이다.

간단한 퍼즐 게임으로 위장한 정보유출 프로그램을 사용자가 다운로드 했을 경우, 실제로 게임을 종료하더라도 백그라운드 단에서 정보유출을 위한 코드들이 동작할 수가 있고 이 악성코드가 사용자의 주소록을 읽어와 해당 프로그램의 설치를 유도하는 문자메시지(SMS)를 주소록에 있는 사람들에게 전송되어 악성코드가 확산될 수 있다.

6. 안드로이드의 보안성 개선방안

가상머신, 자바, 모바일환경, 안드로이드 등 지금까지 각 환경에서의 보안취약점에 대해서 살펴보았다. 본 항에서는 위에서 나열한 각 취약사항을 최소화하기 위한 개선방안에 대해서 설명하고자한다. 첫 번째 개선방안으로는 보안에 대비한 Security Layer Model을 만드는 것이다. 이는 기존 안드로이드 프로그램의 실행과정에서 API부분이 실행될 적에 보안클래스들이 들어있는 패키지를 추가하고 Security Manager가 실행되는 과정에 Access Controller추가를 통하여 Access를 통제하는 방법이다.



(그림 1) Security Layer in Android

또한, Security Layer는 <표 1>과 같이 4단계로 구성하였으며 상위 계층으로 올라갈수록 개념적이고 높은 보안내용을 담고 있다.

<표 1> Security Layer in Android

L4	Protecting the F.S. & N/W
L3	Class loader
L2	Verify the byte code
L1	The language & compiler

Layer1은 The Language나 Compiler 차원을 나타내는 레이어로서 Language 수준에서 프로그래머의 의도적인 혹은 실수에 의한 error의 유발을 최소화시킨다.

그 특징을 살펴보면, 첫째로 pointer를 지원하지 않는다는 것이다. pointer는 상당히 편리한 기능을 가지고 있는 반면 pointer가 가리키는 데이터를 우연히 지우거나 새 데이터를 그 지점에 저장하는 오류를 범할 수 있다. 이 문제는 pointer 대신에 real array을 사용하면 쉽게 해결할 수 있다.

두 번째로 garbage collection이 있는데 개발자가 프로그래밍을 할 때 변수 선언 후 사용하지 않는 등 관리 소홀로 인한 메모리누출 방지를 제공한다.

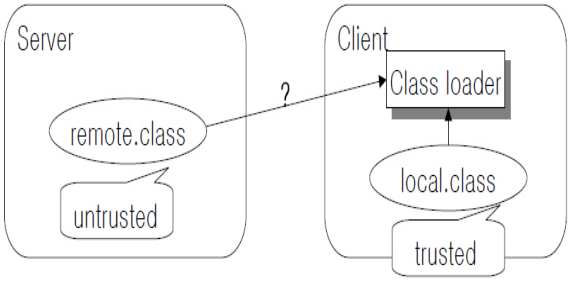
세 번째로 엄격한 type checking이 있다. 엄격한 type checing이란 프로그램 내에서 사용되는 변수나 오브젝트와 같이 다른 클래스에서 이용하면 안 되는 것들을 private type이라고 정의를 해두면 function내에서만 사용되고 다른 곳에서는 이용할 수 없게 되는데 예를 들어 외부에 알려지면 안 되는 password 객체 등이 있다.

네 번째는 bound checking이다. bound checking은 배열을 사용할 때 배열의 사이즈를 넘어 객체를 이용하고자 할 때 사용되는데 예를 들면 array size가 10 인데 11번째 포인팅을 하면 에러가 발생되는 것 들이 있다.

다섯 번째는 late binding이다. late binding은 객체들이 필요하기 전까지는 메모리 위치로 load 되거나 지정되지 않게 함으로서 클라이언트 머신에서 프로그램이 물리적 메모리 위치를 예측하는 것이 불가능하도록 만듦으로서 이러한 것들에 대한 공격시도를 차단시킬 수 있다. 끝으로 late binding은 중요한 class에 대한 계승을 방지하고 주요 method에 대한 overriding을 방지함으로서 중요한 메소드나 클래스들이 잘 보존될 수 있는 기능들을 제공한다.

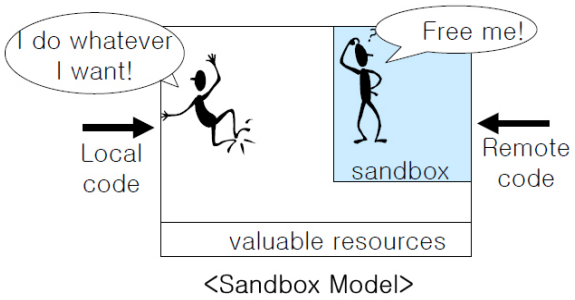
다음으로 Layer2는 Verify the byte code, 즉 바이트 코드를 검증하는 레이어로서 악의적 공격자가 compiler를 수정하여 safety rule을 범하거나 정상적 code가 N/W 전송도중 변경 되는 것에 대한 검증이 이루어진다.

그 특징을 살펴보면, byte code가 실행되기 전에 Verifier 가 byte code에 대한 완전성 여부를 check 하여 보안성을 보장하는 내용을 담고 있으며 verification이 끝나서 정상적으로 이루어지면 컴파일러를 고의적으로 수정해서 악의적인 코드가 만들어지지 않았다는 것과 stack overflow나 underflow가 이루어지지 않는다는 것이 보장이 되게 된다. 그리고 모든 opcode에 대한 parameter의 type이 올바르지 않아 발생되는 현상이 있다. Layer2를 거치게 되면 runtime시 code의 실행 속도를 빠르게 하는 부수적인 효과를 얻게 된다. Layer1과 Layer2만으로는 보안검증이 부족한데 이를 뒷받침 해줄 수 있는 내용으로는 Class가 로드될 때 보안을 점검하는 것과 파일시스템, 네트워크 접근을 보호하는 내용이 있다. 먼저 클래스가 로드될 때 보안적인 측면을 점검하게 되는 Layer3은 Data가 Network로부터 온 것인지, local file 시스템에서 부터 온 것인지를 구별해서 서로 독립된 실행 환경을 갖게 함으로서 네트워크 source로 부터의 보안 위협을 방지하게 된다. (그림 2)



(그림 2) Layer3 - The Class Loader

이를 다른 표현방법으로 Sandbox Model로 표현할 수 있는데 Sandbox Model은 특정한 보안영역을 할당하여 실행되는 구조를 가진다. 그리고 이 Sandbox Model은 보안영역이 할당될 때 고유의 UID, GUID를 부여받게 된다.

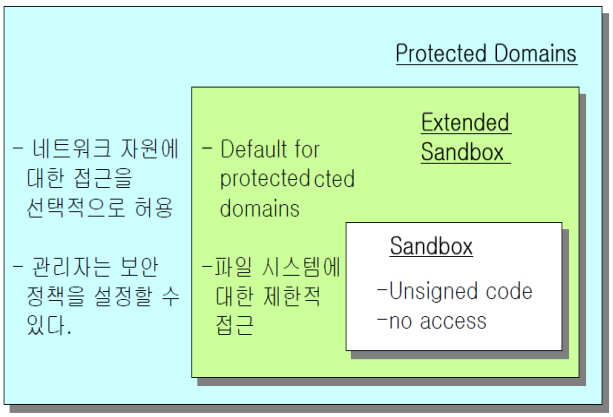


(그림 3) Sandbox Model

Layer4는 파일시스템, 네트워크 접근 보호를 위한 레이어로서 파일시스템과 네트워크 접근 보호를 위한 내용을 담는다. 그 내용으로는 첫 번째, remote code가 local 시스템을 access 하려고 할 때 이를 방지하는 파일

시스템보호가 있다. 파일시스템을 보호하는 이유는 프로그램이 정상적인 것같이 위장하고 들어와서 사용자의 정보를 읽어와 외부로 유출시킬 수 있기 때문이다.

위에서 언급한 첫 번째 개선방안은 Sandbox라는 특정한 보안영역을 할당하여 보안적인 측면을 개선한 방법이라면 두 번째 개선방안은 전자서명을 이용한 방법인데 전자서명이 된 프로그램은 소속 domain에 따라 N/W, file 시스템을 선택적으로 access 가능하게 하는 방법으로 이루어진다. (전자서명이 되지 않은 프로그램은 기존의 sandbox 모델을 적용시킨다) 본 방법은 첫 번째 방법과 유사하긴 하나 permission의 내용이 포함된 것이 그 차이점이라고 할 수 있다.



(그림 4) Protected Domains 보안 모델

전자서명 확인은 프로그램이 설치되는 시점에서 이루어지게 된다.

전자서명은 응용프로그램 개발자를 구분하는 기능을 하며, 소프트웨어 업데이트를 위해서는 같은 키로 서명이 이루어져야 한다는 특징을 가지고 있다.

프로그램에 Signature-Based Permission을 명시하면 같은 인증서로 서명된 응용프로그램은 프로그램 코드와 데이터를 공유할 수 있게 된다.

7. 결론

지금까지 스마트폰에서 사용되는 주요 OS중 안드로이드 환경에서의 보안을 최소화하기 위한 개선방안에 대해서 설명하였다. 안드로이드의 보안 취약성에서도 기술 했듯이 보안결함이 발생되는 주원인으로는 애플리케이션에 대한 검증절차가 없다는데 있다. 이와 관련해서 본 논문에서는 자체적인 검증절차를 수립하여 보안취약점 개선을 맞추는데 초점을 두었다. 사용자가 스마트폰으로 전자상거래 서비스를 이용한다거나 게임을 할 적에 개인정보가 유출되고 악성코드와 같은 바이러스가 유포된다면 사용자에게 큰 손실을 가져다 줄 것이다. 몇 년 전부터 이슈화가 이루어지고 있는 Ubiquitous가 처음에 보급될 때만 해도 보안적인 측면이 많이 제기되었었다. 하지만 지금 현재는 시스템이 상당부분 안정화가 이루진 상태여서 보안 적으로 상당히 개선된 상태이다. 이처럼 안드로이드 환경역시 현재는 시스템이 보급된 지 얼마 되지 않아 인프라 부족 등 여러 전문가들이 보안적인 측면에 대해서 많은 우려를 하고 있지만 Ubiquitous가 처음에 보급되었을 때와 마찬가지로 안전성과 신뢰성이 점차 개선되어 지면서 보안적인 요소도 하나하나 줄어들 것이다. 지금 이 시간에도 본 저자를 포함한 많은 연구팀과 연구자들이 스마트폰에서의 보안취약성 강화를 위해 노력하고 있으며 이를 통해 앞으로 스마트폰에서의 보안적인 결함은 상당부분 많이 개선될 것이라고 생각된다.

참고문헌

[1] "HotJava™: The Security Story", http://www.javasoft.com/sfag/may95/security.html

[2] "Java Security", http://www.javasoft.com/security/

[3] "Java Security API Overview", http://www.javasoft.com/security/

[4] "FAQ's - JAVA™ Protected Domains Security Model", http://java.sun.com/marketing/collcteral/prot_dom.html

[5] M.Erdos, B.Hartman, M.Mueller, "Security Reference Model for the Java Developer Kit 1.0.2", http://www.javasoft.com/security/SRM.html

[6] "Android Emulator의 Open GL 연산 효율 개선에 관한 연구", 김정웅, 이동렬, 양해술, 한국정보처리학회 춘계학술발표대회, 제 16 권 제 1호, Apr. 23-24, 2009, pp. 1109-1111

[7] 안드로이드 프로그래밍, 김정훈, 성안당, 2009

[8] "안드로이드 기반 모바일 정보공유 시스템", 배성호, 김우생, 전자공학회논문지 제46권 제2호 통권 제326호, Mar, 2009, pp. 28-41

[9] 구글의 오픈소스 모바일 OS인 안드로이드 오픈, 컴퓨터프로그램보호위원회, 2008

[10] What is Android? Android Runtime, 구글, 2007

[11] 구글, 모바일 세계를 뒤흔든다, 윤건일, 따뜻한 디지털세상 제128호, Dec, 2007, pp. 22-23

[12] 안전한 자바 프로그래밍은 꿈인가?:자바의 보안 취약점 살펴보기, 서광열 마이크로소프트웨어 통권 278호, June, 2006, pp. 76-81

[13] 무선 확대와 모바일 보안위협, 정연서, 김기영, 장종수, IITA 주간기술동향, 2006

[14] "Java환경에서의 보안 위협과 메커니즘", 이강수 한국정보과학회지, 1972. 7, pp. 48~56