한국컴퓨터정보학회 하계학술대회 논문집 제18권 제2호 (2010. 7)

스마트 폰 어플리케이션의 코드서명*

유재성°, 김학현*, 최동현*, 원동호*, 김승주*‡
°*성균관대학교 정보보호연구소
e-mail: {jsyou, hhkim, dhchoi, dhwon, skim}@security.re.kr

Code Signing of Smart Phone Application

Jaesung Yoo°, Hakhyun Kim*, Donghyun Choi*, Dongho Won*, Seungjoo Kim*‡
°*Information Security Group, Sungkyunkwan University

● 요 약 ●

스마트 폰 사용자의 증가와 개발자들의 참여가 확대되면서, 다양한 스마트 폰 어플리케이션들이 배포되고 있다. 스마트 폰의 운영체제 공급자들은 개발된 어플리케이션을 직접 또는 개발자에게 위임하여 테스트하고 어플리케이션 설치파일의 코드를 서명하여 사용자에게 배포한다. 여기서 코드 서명은 개발자의 확인과 동시에 어플리케이션이 배포과정에서 수정되지 않았음을 보장한다. 사용자 측면에서는 이런 서명이 어플리케이션의 안전성을 판단 할 수 있는 유일한 기준이 된다. 하지만, 코드 서명을 우회하거나 어플리케이션의 설치파일 코드를 수정할 수 있는 방법이 나타나게 되었고, 이것은 사용자가 악성 프로그램을 설치하는 보안 문제로 이어질 수 있다. 본 논문에서는 각 스마트 폰 운영체제별, 어플리케이션의 안전하지 못한 코드 서명으로 발생하는 보안문제를 서술하고, 스마트 폰 어플리케이션의 안전한 코드 서명을 위해 필요한 요구사항에 대해서 논의한다.

키워드: Smart phone(스마트 폰), Codesigning (코드 서명), Android (안드로이드), Windows Mobile(윈도우 모바일), iPhone (아이폰)

I. 서론

일반 컴퓨터에 버금가는 처리능력으로 전화뿐만 아니라, 인터넷과 멀티미디어를 지원하는 스마트 폰의 사용자가 많아지고 있다. 스마트 폰 사용자의 증가로, 애플과 구글, 그리고 마이크로소프트에서는 개인 개발자들도 스마트 폰 어플리케이션을 체계적으로 배포할 수 있도록 개발에 필요한 툴과 테스트 환경 및 배포과정을 지원한다. 이에 힘입어 현재, 다양한 어플리케이션들이 쏟아지고 있다. 하지만, 일부 어플리케이션의 설치파일 코드가 악의적으로 변경되어 재배포 되는 문제가 생겨났다. 스마트 폰은 사용자들의 개인 정보를 가지고 있는 경우가 많기 때문에, 악성 어플리케이션 설치는 개인 정보 유출 등의 보안 사고로 이어질 수 있다.

본 논문의 2장에서는 스마트 폰 운영체제 별로 어플리케이션의 개발과 코드 서명, 배포에 이르는 절차에 대해 서술하며, 3장에서는 배포과정에서 발견된 취약점에 대해서 살펴본다. 4장에서는 스마트 폰 어플리케이션의 안전한 코드 서명을 위해 필요한 요구사항을 도출하고, 5장에서는 향후 연구 계획을 언급하며 결론을 맺는다.

* 본 연구는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(계약번호 UD100002KD)
* 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었습니다. (NIPA-2010-(C1090-1031-0005))
‡ 교신저자, skim@security.re.kr

II. 관련 연구

1. 안드로이드

안드로이드 기반의 어플리케이션은 개발자가 직접 테스트한 후, RSA 서명 알고리즘을 이용해서 어플리케이션 설치파일을 개발자의 서명키로 코드 서명한 후, 안드로이드 마켓에 등록된다.[1]

그림 1. 안드로이드 어플리케이션 배포절차
Fig. 1. The distribution process of Android Application

안드로이드 어플리케이션에 대해서 구글이 코드 서명이나, 악성 프로그램 검사 등의 안전성 검증에 직접 관여하지 않고 테스트 툴만 제공하기 때문에, 안드로이드 어플리케이션 개발자는 구글에서 제공하는 테스트 툴을 이용해서 자신이 개발한 어플리케이션을 직접 테스트하게 된다. 테스트를 거쳐 최종적으로 생성된 설치 파일을 개발자가 가지고 있는 서명키로 코드 서명하게 되는데, 이 때

사용되는 서명키는 구글이나 다른 외부 CA(Certificate Authority)에서 발급된 것이 아닌 개발자가 JDK(Java Development Kit)에서 제공되는 Keytool 명령을 이용해서 생성한 것이다.

이러한 코드 서명은 어플리케이션 개발자를 구분하는 역할을 한다. 하지만, 개발자의 서명을 증명해 줄 수 있는 CA와 같은 기관이 없기 때문에, 사용자는 개발자의 서명을 신뢰하기 어렵다. 또한, 어느 개발자든지 안드로이드 어플리케이션에 서명만 하면, 사용자의 스마트 폰에 아무런 검증 절차 없이 설치 될 수 있다. 따라서, 사용자는 설치하려는 어플리케이션이 안전한 것인지에 대한 여부를 확인할 수 없고, 악성 어플리케이션 설치 후에 발생되는 문제는 전적으로 사용자의 책임으로 돌아간다.

2. 윈도우 모바일

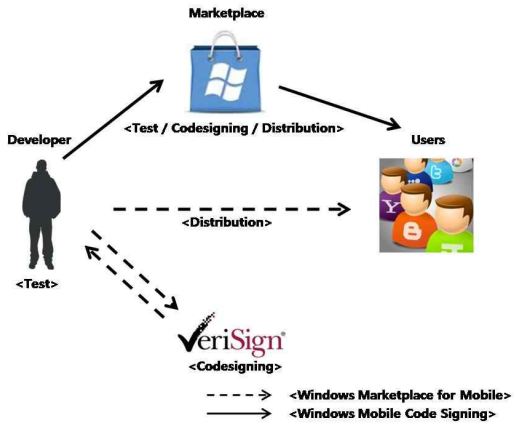

그림 2. 윈도우 모바일 어플리케이션 배포절차
Fig. 2. The distribution process of Windows Mobile Application

윈도우 모바일 기반의 어플리케이션이 서명되어 배포되는 방법은 'Windows Marketplace for Mobile'과 'Windows Mobile Code Signing'의 두 가지가 있다.[2] 첫 번째 방법은, 마이크로소프트에서 제공하는 Marketplace 웹 사이트에 개발자가 어플리케이션을 제출하면 마이크로소프트에서 테스트 및 코드 서명, 그리고 배포에 이르는 일련의 과정을 통합해서 진행한다. Marketplace로 제출된 어플리케이션은 마이크로소프트가 직접 기술테스트, 정책 부합여부, 그리고 시장 적합성을 판단하기 때문에, 악성 프로그램이 Marketplace를 통해 사용자들에게 제공되는 것을 미연에 방지한다.

두 번째 방법은 윈도우 모바일 환경에서 개발된 어플리케이션을 마이크로소프트가 인증한 CA인 Verisign에서 서명을 하고, 개발자 또는 ISV(Independent Software Vendor)가 자신들의 유통 채널을 통해 배포하는 것이다. 이 방법은 윈도우 모바일 6.5까지만 지원되며, 윈도우 모바일 7은 첫 번째 방법을 통해서만 배포가 가능하다.

이와 같이, 윈도우 모바일 어플리케이션의 안전성을 마이크로소프트가 직접 검증하기 때문에, 사용자들은 이를 믿고 어플리케이션을 설치하게 된다.

3. 아이폰 OS

그림 3. 아이폰 어플리케이션 배포절차
Fig. 3. The distribution process of iPhone Application

아이폰 어플리케이션 개발자들은 iPhone Developer Program에 가입한 후, 개발한 어플리케이션을 테스트하고 Apple store에 등록할 수 있다.[3] 개발자들은 iPhone Developer Program에서 제공되는 테스트 도구를 통해 아이폰이나 시뮬레이터를 이용해서 어플리케이션을 테스트 하게 되며 특히, iPhone Developer Program에 등록된 다른 개발자들과 함께 팀을 구성해서, 자신이 개발한 어플리케이션을 팀 구성원들과 함께 테스트할 수 있는 환경을 제공받는다. 애플은 배포되는 어플리케이션의 명확한 출처를 위해 개발자의 철저한 등록을 요구한다. 또한, 개발자들은 애플로부터 교부 받은 인증서로 자신들이 개발한 어플리케이션의 코드를 서명하므로, 어플리케이션의 안전성을 증명한다. 이렇게 코드 서명된 어플리케이션이 제출되면 애플에서 코드 검증을 수행한 후, Apple store로 등록 시킨다.

철저한 개발자 등록으로 개발자는 까다로운 절차를 밟아야 하지만, 사용자 측면에서는 어플리케이션에 대한 출처를 신뢰할 수 있다. 또한, 애플의 어플리케이션 코드 검증으로 사용자는 비교적 안심하고 Apple store에서 어플리케이션을 다운받아 설치할 수 있다.

III. 스마트 폰 어플리케이션의 보안문제

안드로이드의 경우, 사용자에게 배포되는 어플리케이션의 테스트나 코드 서명에 구글이 직접 관여하지 않고, 개발자가 전담한다. 단지, 악성 어플리케이션이 발견되면 구글이 강제적으로 해당 어플리케이션을 안드로이드 마켓에서 삭제하는 정책이다. 이에 대한 피해로 지난 1월, 09Droid라는 개발자가 배포한 50여개 은행의 모바일뱅킹 서비스를 통해 많은 사용자들의 개인정보가 유출되었던 보안사고가 있었다.[4] 사용자들이 설치했던 모바일뱅킹 서비스 어플리케이션은 해당은행 화면으로 Redirection하는 역할을 수행하면서, 사용자들의 개인정보를 유출하였다. 이것은, 안드로이드 어플리케이션의 안전성 검증을 제대로 수행하지 못하기 때문에 발생한 보안사고다. 또한, CA가 아닌 개발자가 직접 어플리케이션을 서명하기 때문에, 사용자는 설치되는 어플리케이션이 신뢰할 수 있는 것인지 판단하기가 어렵다.

윈도우 모바일의 경우, 개발된 어플리케이션이 Marketplace를 통해 제출 되면, 마이크로소프트에서 테스트와 코드 서명, 그리고 배포에 이르는 전 과정을 수행한다. 위와 같은 절차를 통해서 배포되는 어플리케이션은 비교적 안전하지만, Marketplace를 통하지

않고 외부 사이트에서 배포되는 어플리케이션들은 여전히 악성코드가 추가 또는 수정 될 수 있는 보안문제에 노출되어 있다. 최근에는 DoDownload와 GearDownload 같은 유명 다운로드 사이트에서 '3D Anti-Terrorist Action'이라는 게임 어플리케이션에 악성코드가 추가되어 재배포 되었고, 많은 사용자들이 의도하지 않은 국제 통화료를 내야하는 피해가 발생하였다[5]. 이것은 배포된 어플리케이션에 대한 코드 변경 여부를 검증하는 무결성 검사가 제대로 이뤄지지 않았기 때문에 발생한 문제다. 이러한 피해를 막기 위해서, 어플리케이션이 배포되어 설치된 후에도, 어플리케이션 코드의 변경여부를 검증할 수 있는 무결성 검사가 제공되어야 한다.

아이폰의 경우, 다른 스마트 폰에 비해 다소 폐쇄적인 어플리케이션 배포 정책을 가지고 있다. 철저한 개발자 등록과 코드 서명, 그리고 어플리케이션 테스트는 사용자에게 신뢰를 주는 부분이다. 하지만, 아이폰에도 어플리케이션 서명 부분에 있어서 보안문제가 있었다. 아이폰 SDK에서 Codesign Utility가 어플리케이션을 서명할 때, Symbolic Link 되어 있는 파일을 제대로 서명하지 못하는 것이 발견되었다[6]. Symbolic Link는 윈도우의 바로가기 아이콘과 같이, 복사된 파일이 원본파일의 포인터 역할을 하는 것을 말한다. Codesign Utility는 Symbolic Link의 원본 파일은 서명하지만 복사본 파일은 서명하지 못하는 것이 문제였다. 서명되지 않은 Symbolic Link를 이용하면 악의적인 해커가 어플리케이션을 업데이트하고 악성 코드를 삽입 할 수 있다. 이러한 Codesign Utility의 문제점은 코드 서명을 수행하는 알고리즘에도 보안 문제가 발생할 수 있음을 의미한다.

Ⅳ. 스마트 폰 어플리케이션의 코드 서명을 위한 요구사항 도출

본 장에서는 스마트 폰 어플리케이션의 코드 서명을 위한 요구사항들을 도출하고, 배포 된 어플리케이션의 코드가 변경되지 않도록 제공되어야 할 서비스에 대해서 기술한다.

먼저 스마트 폰에 적용될 수 있는 PKI(Public Key Infrastructure)의 요구사항을 도출하였다.[7][8] 그리고 스마트폰 보안시스템 보호프로파일[9]과 앞서 언급한 보안 취약점을 토대로, 어플리케이션의 코드 서명을 우회하거나 어플리케이션 코드가 악의적으로 변경되는 것을 방지하기 위해서 아래의 세 가지 항목을 요구사항으로 추가하였다.

- Prevent Forgery in Signature Making
- Prevent Forgery in Signature Verifying
- Verify Signature in Run Time

첫 번째 요구사항은 코드 서명과정에서, 신뢰할 수 없는 어플리케이션이 신뢰할 수 있는 어플리케이션으로 위조 서명되는 것을 방지하는 것이다. 두 번째는 신뢰할 수 없는 어플리케이션의 코드 서명이 검증과정을 우회하거나, 검증 프로그램을 조작하여 신뢰할 수 있는 서명으로 잘못 인식되는 것을 방지하는 것이다. 세 번째는 어플리케이션이 설치된 후에도, 코드가 변경 되지 않았음을 검증하기 위한 것이며, 어플리케이션 실행시마다 코드 서명을 검증하는 것이다. 이렇게 도출된 요구사항들을 현재 스마트 폰 운영체제들이 제공하는지에 대한 내용을 <표1>에 나타내었다. '○'는 요구사항을 지원하고, '×'는 지원하지 않음을 말한다.

표 1. 스마트 폰 환경에서 어플리케이션 코드 서명을 위한 요구사항[1][10][11]

Table 1. The requirements of application code signing in smart phone environment.[1][10][11]

	Android	Windows Mobile	iPhone OS
Cross-certification	×	○	○
Existence of Certification Authority	×	○	○
Key Backup & Recovery	○	○	○
Certificate Revocation	×	○	○
Automatic Key Update	×	○	○
Certificate Repository	×	○	○
Timestamping	○	○	○
Support for non-repudiation	×	○	○
Prevent Forgery in Signature Making	×	-	-
Prevent Forgery in Signature Verifying	×	-	-
Verify Signature in Run Time	×	×	×

'-'는 요구사항의 충족여부가 확인되지 않은 것임

위와 같은 요구사항들을 만족할 경우, 개발자의 신원 확인이 가능하고 배포되는 어플리케이션의 무결성을 보장한다. 안드로이드의 경우, CA가 따로 없기 때문에 도출한 요구사항들의 대부분을 만족하지 못하므로, 악성 어플리케이션이 배포되는 것을 막기가 힘들다. 윈도우 모바일과 아이폰 OS는 철저한 어플리케이션 테스트와 CA를 통한 서명으로 악성 어플리케이션의 배포를 방지할 수 있다. 하지만, 어플리케이션이 설치 된 후, 코드가 변경되었는지 확인하기 위한 코드 서명 검증은 세 운영체제 모두 지원하지 않는다. 무결성 확인을 위한 서명 검증이 지속적으로 수행되지 않으면, 설치된 어플리케이션에 악성 코드가 추가되어 보안문제를 일으킬 수 있다.

<표1>에서 'Prevent Forgery in Signature Making'과 'Prevent Forgery in Signature Verifying' 요구사항의 일부는 '-'로 표시했다. 이는 현재까지 Marketplace와 Apple Store에서, 어플리케이션 코드의 서명검증을 우회하거나 조작하는 보안사고가 발생하지 않았고, 마이크로소프트와 애플에서 코드 서명에 관한 자세한 내용을 명시하지 않기 때문에 확인되지 않은 부분이다. 하지만, 스마트 폰 어플리케이션의 무결성을 보장하기 위해서 반드시 충족되어야 할 요구사항들이다.

사용자 보안 측면에서, 어플리케이션의 설치시는 물론, 설치 후에도 어플리케이션의 무결성이 보장되어야 한다. 이를 위해, <표1>의 요구사항 중 3개 운영체제에서 모두 만족시키지 못하는 'Verify Signature in Run Time' 항목이 만족될 수 있도록, 실시

간 코드 서명 검증 서비스가 마련되어야 한다. 또한, 배포된 어플리케이션의 서명 검증을 우회하지 못하도록 코드 난독화 기술도 함께 제공되어야 한다.

V. 결론 및 향후 계획

지금까지 각 스마트 폰 별 어플리케이션이 코드 서명되어 사용자에게 전달되는 과정과 어플리케이션 배포 후 나타난 보안 문제와 취약점에 대해서 살펴보고, 스마트 폰 환경에서 어플리케이션 코드 서명을 위한 요구사항을 도출하였다.

코드 서명의 가장 큰 목적은, 사용자가 안전한 어플리케이션을 제공받는 것을 보장하는 것이다. 마이크로소프트의 Marketplace와 애플의 Apple Store에서는 비교적 안전하게 어플리케이션이 배포되고 있다. 하지만, 안드로이드를 포함한 세 운영체제 모두 설치된 어플리케이션에 대해서는 실시간 코드 서명 검증을 제공하지 않기 때문에, 향후 악의적인 코드 변경의 위험성이 존재한다.

결론적으로, 스마트 폰 어플리케이션이 안전하게 배포되고 무결성을 보장하기 위해서, <표1>에 제시된 요구사항을 만족하는 서비스가 제공되어야 한다. 향후 계획으로, 이러한 서비스가 구체적으로 실현될 수 있는 방법에 대해서 연구해 보고자 한다.

참고문헌

[1] Android Developer Web pages *http://developer.android.com/guide/http://developer.android.com/guide/*

[2] Windows Mobile code signing http://msdn.microsoft.com/en-us/windowsmobile/dd569132.aspx

[3] Apple Developer Web pages http://developer.apple.com/

[4] Android banking apps deemed insecure, Goolge pulls them from Android market place http://www.mobilemag.com/2010/01/17/andriod-banking-apps-deemed-unsecure-google-pulls-them-from-android-marketplace/

[5] Malware found lunking in apps for Windows Mobile *http://news.cnet.com/8301-27080_3-20006882-245.html*

[6] Intrepid iPhone developers bypass security for functionality *http://www.avertlabs.com/research/blog/index.php/category/iphone/*

[7] Wireless Application Protocol Public Key Infrastructure Definition Open Mobile Allinance OMA-WAP-WPKI-V1_1-20050322-C

[8] What is PKI? *http://www.entrust.com/whatispki.htm*

[9] 스마트폰 보안시스템 보호프로파일 V1.0 *(Protection Profile for Security Enhancement System for Smart Phone V1.0) 2008.8.22*

[10] Certificate for Winodws Mobile 5.0 and Windows Mobile 6 http://technet.microsoft.com/en-us/library/cc182301.aspx

[11] Apple Inc. Certification Authority Certificate Policy *Version 1.2*