안드로이드 인앱빌링의 취약점 및 원인 분석에 대한 연구

김 상 원, 김 희 열
경기대학교 대학원 정보보호연구실
genius13684@gmail.com, heeyoul.kim@kgu.ac.kr

Vulnerability of Android In-App Billing and cause analysis

Kim Sang-won, Kim Hee-youl
Kyonggi University Information Security Laboratory

요 약

국내에 안드로이드 OS 의 점유율이 증가함에 따라 기본 결제 시스템인 인앱빌링 서비스의 비중 또한 굉장한 추세로 증가하였다. 하지만 최근에는 여러 빌링 크랙이 등장하여 판매자에게 피해가 생기고 있다. 특히 하나의 애플리케이션 단위의 공격이 아닌, 자동화된 공격이 등장하면서 그 피해 규모가 매우 커졌다. 본 논문에서는 몇 가지 자동화된 공격들을 소개하고 공격이 이루어진 원인과 구글의 보안 가이드라인에 대해서 분석한다. 또한 이런 문제점들을 개선할 수 있는 방안에 대해서 제안하고자 한다.

1. 서 론

스마트폰을 사용한 결제는 현재 IT 분야에서 가장 큰 이슈 중에 하나이다. 특히 국내에서는 안드로이드 플랫폼의 강세에 따라 안드로이드에 맞는 여러 결제 수단이 등장하고 있다. 특히 SNS 애플리케이션인 카카오톡이 등장하고, 이와 연동된 카카오 게임이 등장하였다. 많은 사용자들이 게임 내부에 있는 컨텐츠의 결제를 하였고, 그 결제 수단인 인앱빌링의 비중이 기존에 비해 훨씬 증가하였다.

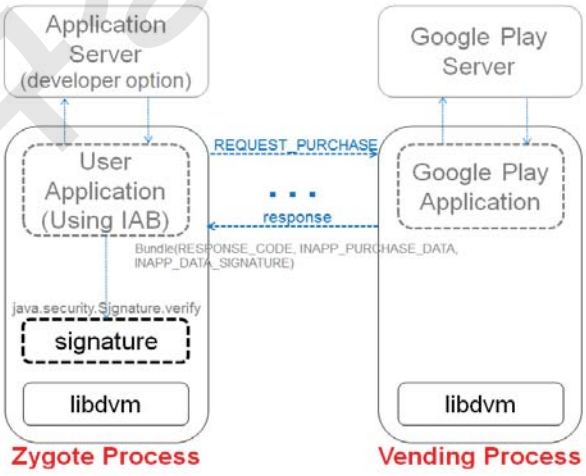
IAB 사용의 비중이 늘어난 만큼, 사용자들의 관심도 따라서 증가하였다. 이것은 긍정적인 측면이라 볼 수 있지만, 부정적인 측면 또한 야기하게 되었다. 몇몇 악의적인 사용자들이 비정상적인 경로를 통하여 제대로 결제가 이루어지지 않아도 아이템이 지급될 수 있도록 여러 가지 시도를 하기 시작하였다. 즉, 안드로이드에도 Billing crack 이 등장한 것이다.

빌링 크랙은 여러 가지가 있는데, 그 중에서 가장 쉬운 공격 방식은 recompile 이다. 스마트폰에 설치된 apk 파일을 decompile 하고, 내부의 소스코드를 항상 결제가 이루어지게끔 수정하고 다시 패키지화하여 설치하는 것이다. 이 공격은 공격자가 apk 단위로 수정해야한다는 번거로움이 있었지만, 최근에는 보다 지능화된 automated attack 이 등장하였다. 이 공격은 보통 하나의 애플리케이션 형태로 제공되고, 애플리케이션 단위가 아닌 하나의 device 에 설치된 애플리케이션들을 타겟으로 할 수 있다. 그 대표적인 예로 '프리덤' 이라는 애플리케이션이 있다. '프리덤' 은 루트 권한을 얻어 해당 device 에 인앱빌링을 사용하는 애플리케이션을 목록으로 나타내고, 그 앱을 실행하여 결제할 때 정상적인 결제를 우회할 수 있도록 한다. 이를 막기 위하여 몇몇 게임 회사에서는 루트 권한을 얻은 상태에서 애플리케이션이 실행되지 않도록 하거나, 프리덤의 패키지명을 체크하고 있지만 근본적인 해결책은 되지 않고 있다.

본 논문에서는 '프리덤' 과 유사한 automated attack 을 소개하고, 공격이 이루어지게 된 원인에 대하여 분석하고자 한다.

2. 관련 연구

2.1 In-App Billing Architecture

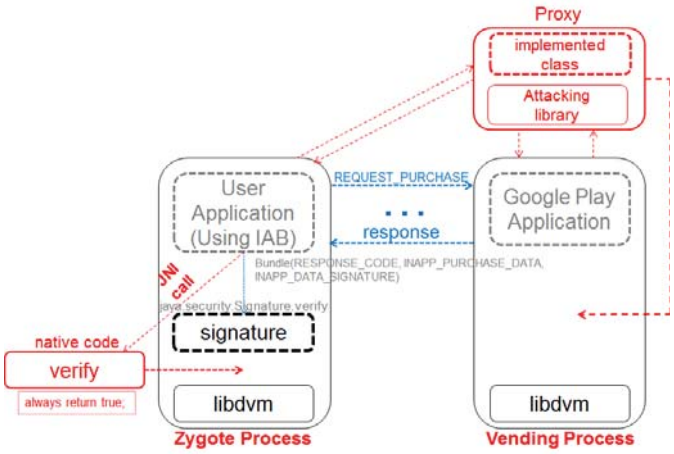


(그림 1) In-App Billing Architecture [1]

그림 1 는 정상적인 인앱빌링 구조를 나타낸 것이다. 사용자 애플리케이션은 device 내부에 있는 PlayStore 애플리케이션과 IPC 통신을 주고받는다. 사용자가 어떤 아이템에 대하여 결제를 진행하려고 하면, 해당 아이템의 정보를 포함하여 결제 요청을 하게 된다. 그리고 최종 결제 성공의 유무는 결제 정보의 무결성을 확인한다. 즉, PlayStore에서 보낸 사인된 데이터의 검증을 통하여 결정된다.

2.2 Redirecting verify method to verify function

Dalvik VM 에서는 JNI 를 사용하여 공유 라이브러리로부터 native code 를 호출할 수 있다. 그리고 인앱빌링은 최종 결제 검증에 따라 결제의 성공 유무가 결정된다. 이 두 가지를 이용하여 최종 검증 단계를 우회할 수 있는 공격이 가능해진다.



(그림 2) Redirecting verify method to verify function [2]

그림 2 는 verify method 를 우회할 수 있는 공격을 표현한 것이다. 최종 결제 검증에 결제의 성공 유무가 의존한다는 부분을 악용한 공격으로, 검증 결과를 항상 true 로 반환되도록 하는 것이다. 이 공격 방식에는 중요한 기술이 하나 있는데 바로 라이브러리 인젝션 이다 [3]. 자바 기반의 안드로이드에서는 C 레벨 함수 호출이 불가능하기 때문에, 라이브러리 형태로 만들어 인젝션 기술을 사용하여 해당 프로세스의 영역에 끼워 넣는 방식이다.

먼저 이 공격이 성공하기 위해서는 사용자 애플리케이션 프로세스와 device 내에 있는 PlayStore 애플리케이션 프로세스 모두 라이브러리 인젝션이 이루어져야 한다.

Vending Process, 즉 PlayStore 에 인젝션 되는 라이브러리는 proxy 와 유사한 동작을 하게 된다. 사용자 애플리케이션으로부터 오는 결제 요청을 비롯한 여러 요청들을 수집하고 결제가 종료되는 시점 즉, 정상적인 진행 혹은 취소 시점에 보내는 응답에 가짜 서명을 하여 전송하게 된다.

Zygote Process, 사용자 애플리케이션에는 항상 true를 반환하도록 하는 verify 함수가 포함된 라이브러리가 인젝션되어 있다. 이 라이브러리는 최초에 실행될 때, 기존 java.security.Signature의 verify 메소드가 호출되면 삽입된 verify 함수가 호출되도록 설정한다. 그리고 PlayStore로부터 온 응답에 가짜 서명이 되어 있다면, JNI 로 가짜 verify 함수를 호출한다. 즉, 결제를 정상적으로 진행하지 않더라도 마치 아무 문제가 없는 것처럼 결제가 진행된다.

3. 분석

구글에서는 개발자에게 API 뿐만 아니라 개발에 필요한 지침, 즉 가이드라인을 제공하고 있다. 이 지침에는 흥미로운 부분이 하나 있는데, 바로 'Security and Design' 이다 [4]. 보통 결제라는 민감한 부분이 관련되었다면 '보안' 측면은 필수사항으로 하지만 구글에서는 가급적 따라야 하는 지침으로 두고 있다.

하지만 이런 불분명한 지침 때문에 2절에서 보았던 공격이 가능해졌다. 이 공격이 통하게 된 지침사항은 다음과 같다.

- Perform signature verification tasks on a sever

이것은 사인된 결제 정보의 검증을 서버에서 수행하라는 권고사항이다. 하지만 서버 운용이 어려운 개발자들은 device 내부에서 최종 검증을 수행하도록 개발하고 있다. 그렇게 개발된 애플리케이션은 프리덤이나 2절에서 소개된 공격 등에 의해서 정상적인 수익을 얻지 못하고 있다. 이 지침 외에도 문제가 발생할 수 있는 부분이 하나 더 있다.

- Use secure random nonces

결제를 요청할 때 구매 요청 데이터에는 nonce가 포함되어야 한다. 구글에서는 재전송 공격의 방지를 위하여, 이 nonce의 생성을 SecureRandom 과 같이 암호학적으로 안전한 방법을 사용하여 난수 생성을 해야 하고 난수의 생성 및 검증을 서버에서 수행하라고 권고하고 있다. 하지만 이 역시 서버를 운용해야 안전하다고 할 수 있다는 점에서 한계가 있다. 따라서 현재 구글에서 제공하고 있는 가이드라인을 보다 강화할 필요성이 있고, 또한 이런 문제점들에 대응할 수 있는 방안을 제공하는 것이 필요하다.

4. 결론 및 향후 연구

구글에서는 안드로이드 플랫폼에서의 결제를 위하여 인앱빌링을 서비스하고 있다. 하지만 정상적인 결제 시퀀스를 깨고 우회하는 공격들이 등장하였다. 우리는 이번 연구에서 이런 공격이 생기게 된 원인이 구글의 불분명한 가이드라인 때문에 발생하고 있음을 파악하였고, 이런 문제점들을 방지하기 위해서 구글의 가이드라인에 개선이 필요하다는 것을 분석하였다. 우리는 앞으로 서버를 사용하지 않더라도 device 내부에서 안전하게 검증 단계를 거칠 수 있는 방법에 대하여 연구하고자 한다.

참 고 문 헌

[1] Google. In-app billing Version 3 API. http://developer.android.com/google/play/billing/api.html, August 2014.

[2] Mulliner, Collin, William Robertson, and Engin Kirda. "VirtualSwindle: an automated attack against in-app billing on android." Proceedings of the 9th ACM symposium on Information, computer and communications security. ACM, 2014.

[3] Clowes, S. injectso - Modifying and Spying on running processes under Linux and Solaris. http://www:blackhat:com/presentations/bh-europe-01/shaun-clowes/bh-europe-01-clowes:ppt, 2001.

[4] Google. In-app Billing Security and Design. http://developer.android.com/google/play/billing/billing_best_practices.html, August 2014.