논문 2013-2-3

# 무결성 검증 모듈을 사용한 안드로이드 앱 필터링 기법의 개선

강성욱*, 정윤식**, 조성제***†

# Enhancing an Android App Filtering Scheme using an Integrity Verification Module

Seongwook Kang*, Youn-Sik Jeong**, Seong-je Cho***†

요 약

본 논문에서는 웹하드 등에서 불법 유포 및 유통되는 안드로이드 앱을 필터링 하는 기법을 제안한다. 기존에 제시한 classes.dex 해시값 기반 단순 필터링 기법은 해시값 변경 등의 간단한 조작만으로 우회가능하기 때문에, 이를 보완하기 위해 안드로이드 앱의 정보에 대한 무결성 검증 모듈을 추가한다. 무결성 검증 모듈은 APK 내의 2가지 정보를 이용한다. 첫 번째는, 앱의 변조 여부를 확인하기 위해 manifest.mf 파일의 해시값 정보를 데이터베이스에 최조 저장된 해시 값과 비교한다. 두 번째는, 재패키징 여부를 확인하기 위해 cert.rsa 파일의 인증서 정보를 분석한다. 실험을 통해, 제안 기법이 기존의 문제점을 해결할 수 있음을 보이고, 무결성 검증 방법의 성능을 평가하였다.

**한글키워드 :** 안드로이드 앱, 필터링, 재패키징, 무결성 검증, 해시값, 인증서

Abstract

In this paper, we propose a scheme to filter out copyrighted Android applications which are illegally distributed through online internet sites. A previous simple filtering scheme based on the hash value of a classes.dex file has a problem that the previous scheme can be bypassed by a straightforward change like modifying the hash value. To address the problem, we introduce a new module to verify the integrity of the hash value and certificate information for each Android application. The integrity verification module uses two kinds of information inside an APK file. The first information is a hash value of manifest.mf where we can determine whether the hash value is modified or not by comparing it with one initially stored a database. The second information is a certificate of cert.rsa which can be used to determine if the target file is repackaged or not. Through some experiments, we show that the proposed scheme can overcome the limitation of the previous one and evaluate performance of our scheme.

**Keyword :** Android app, Filtering, Repackaging, Integrity verification, Hash value, Certificate

※ 이 논문은 2013년 한국소프트웨어감정평가학회 추계학술발표대회에서 발표된 '웹하드에서 특징정보 기반 불법 안드로이드 앱 필터링 기법' 논문을 확장한 것임.

* 단국대학교 컴퓨터학과 소프트웨어보안전공
** 단국대학교 컴퓨터과학과
*** 단국대학교 컴퓨터학과
† 교신저자 : 조성제(Email: sjcho@dankook.ac.kr)

※ 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업 의 연구결과, "(NIPA-2013-과제번호) 그리고 미래과학창조부 및 한국인터넷진흥원의 "고용계약형 지식정보보안 석사과정 지원사업"의 연구 결과로 수행되었음 (과제번호 H2101-13-1001)
접수일자: 2013.12.20 수정완료: 2013.12.23.

## 1. 서 론

최근 안드로이드 스마트폰 사용자들이 급격히 증가함에 따라 안드로이드 애플리케이션(이하 “앱”) 시장 또한 빠르게 성장 하였다. 급격한 앱 시장 성장과 앱의 단말기 외부 유출이 비교적 쉽다는 점에 힘입어 안드로이드 앱을 불법으로 유통하는 피해 사례가 증가하고 있다[1, 2]. 한국방송통신전파진흥원에서 조사한 결과, 주로 블랙마켓을 통해 유료 앱을 불법다운로드 하는 등 관련 산업 성장에 큰 걸림돌로 작용하고 있다. 실제 일부 안드로이드 모바일게임은 불법 복제율이 97%에 달할 정도로 피해가 심각하다고 한다[3].

안드로이드 앱 불법 유포 및 유통을 막기 위해 재패키징 탐지기법과 안드로이드 앱 필터링기법이 연구 된 바 있다[4, 5, 6]. 하지만, 서드파티 마켓(Third-Party Market)을 통해 유통되는 위변조된 앱 외에도 웹하드, P2P 등 다양한 경로를 통해 불법 유통이 진행되고 있어, 기법의 활용도가 크지 않다는 단점이 있다. 또한, 웹 서버에서의 안드로이드 앱 필터링 기법[4]은 앱의 해시 값을 변경하면 간단하게 우회가 가능하다.

따라서 본 논문에서는 특징정보 기반 불법 안드로이드 앱 필터링 및 무결성 검증 기법을 제안한다. 본 연구진은 기존에 manifest.mf 파일 내부에 존재하는 안드로이드 앱 실행파일인 classes.dex파일의 해시 값을 기반으로 불법 유통되는 안드로이드 앱을 필터링하는 기법에 대해 제안하였다[4]. 하지만, 기법[4]은 필터링의 특징정보로 사용되는 해시 값을 변경하면 비교적 쉽게 우회될 수 있다. 본 논문에서는 이를 보완하기 위해, cert.rsa(인증서 정보) 파일을 대상으로 무결성 검증 정보를 추출하고, 이를 필터링의 특징정보로 사용한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 앱 재패키징 탐지 및 불법 유통을 차단하기 위한 연구에 대해 언급하고, 3장에서는 본 논문에서 제안하는 불법 앱 필터링 및 무결성 검증 기법에 대해 설명한다. 4장에서는 무결성 검증 기법의 성능평가를 보이고, 5장에서 결론을 맺는다.

## 2. 관련연구

### 2.1 안드로이드 앱 불법 유통 차단 기법

안드로이드에서 추출 가능한 특징정보를 기반으로 불법으로 유포되는 앱을 필터링 하는 기법이있다[4].

해당 기법에서는 classes.dex 파일의 해시 값을 특징정보로 사용하였다. classes.dex 파일의 해시 값은 앱 디렉토리 내의 manifest.mf 파일에서 추출 할 수 있다. 이 기법은 간단하면서도 효율적이지만, 우회가 가능하다는 단점이 있다.

[4]에서 제안된 필터링 기법을 우회하는 방법은 manifest.mf 파일의 classes.dex 파일 해시 값을 변조 후 앱을 재패키징하면 된다. 또, 다른 방법은 classes.dex 파일에서 실행과 관련 없는 부분을 일부 변경하여 재패키징 하는 것이다. 이럴 경우, 재패키징 과정에서 classes.dex의 해시 값이 원본과 다른 값으로 변경되어 manifest.mf 파일에 저장된다. 따라서, 단순 classes.dex 파일의 해시 값 외에, manifest.mf 파일 변조 및 재패키징 된 앱의 무결성 검증이 필요하다.

### 2.2 안드로이드 앱 재패키징 탐지

마켓에서 배포되는 안드로이드 앱에 악성 코드를 삽입한 후 재패키징 하여 배포하는 사례가 증가하고 있다. 이를 해결하기 위해 RePAD[5] 기법을 이용하여 재패키징 된 앱을 탐지하는 연구가 있다. 이 기법은 사용자 모바일 단말기에 탑재된 클라이언트 앱과 원격 서버로 구성되는

시스템이다. 클라이언트는 사용자가 설치한 앱의 출처와 정보를 추출하여 원격 서버로 전송하고, 서버는 전송된 정보를 바탕으로 앱의 재패키징 여부를 탐지한다.

아래 그림 1은 RePAD 전체 시스템을 도식화 한 모습이다.

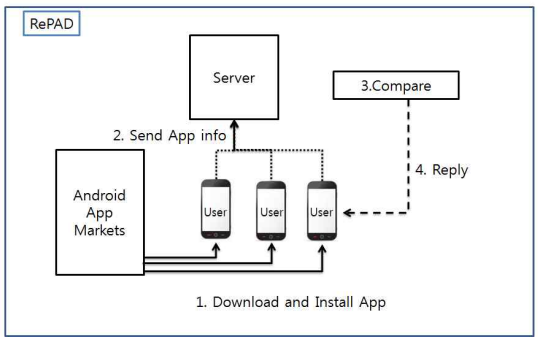


그림 1 RePAD 시스템 개요

서드파티 마켓에서 앱의 재패키징 여부를 판별하기 위해 DroidMOSS를 이용하는 연구가 있다[6]. 해당 기법은 서드파티 마켓에 등록된 앱과 공식 마켓에 등록된 앱의 유사도를 비교하여 재패키징된 앱을 탐지한다. 자세히는 원본 앱과 재패키징된 앱의 classes.dex 파일에서 Dalvik virtual machine 상에서 수행되는 명령어인 opcode만 추출하고, 추출한 opcode를 가지고 Fuzzy-Hasing을 수행한다. 산출된 해시 값의 거리를 측정하여 재패키징 앱을 탐지 한다.

### 2.3 윈도우 환경 소프트웨어 필터링

PC 환경의 소프트웨어 저작권 분쟁이 많이 발생하면서, 소프트웨어 도용 탐지 연구가 진행 되고 있다.[7, 8] MS 윈도우 환경에서의 유사한 소프트웨어 분류기법은 PE파일 포맷에서 DLL(Dynamic-link Library)들에 대한 정보를 추출하여 분류한다. 이 기법은 윈도우 시스템에서 참조 되는 DLL 정보와 특정 소프트웨어 군에서 사용되는 DLL 정보의 사용 빈도수를 활용하여 소프트웨어를 분석한다[7]. 이 기법은 DLL의 개수가 작거나 패킹 된 프로그램의 경우 DLL 정보 추출이 어렵다는 단점이 있다.

윈도우 프로그램의 정적 버스마크기반 소프트웨어식별 기법[8]은 PE파일 포맷의 IAT(Import Address Table)에서 버스마크를 추출 후, 비교를 통해 소프트웨어를 식별한다. IAT에서 DLL 명과 개수, API 명과 개수를 버스마크로 추출하고, 추출된 버스마크를 통해 불법 복제 소프트웨어를 검색 할 수 있다. 그러나 DLL과 API로 비교하기 때문에 마이너 버전의 소프트웨어 식별은 다소 어렵다.

## 3. 불법 앱 필터링 및 무결성 검증

기존 불법 안드로이드 앱 필터링 기법[4]에서는 manifest.mf 파일에 존재하는 classes.dex 파일의 해시 값을 이용하여 앱을 필터링 하였다. 하지만 공격자가 manifest.mf에 존재하는 해시 값을 변조할 경우, 필터링 시스템을 우회할 가능성이 있다.

따라서 본 논문은 필터링 우회 기법을 차단하기 위해 앱의 특징정보 추출 시 manifest.mf 파일의 무결성 정보도 함께 추출한다. 무결성 정보는 앱의 구성요소가 변조되었는지 판별하기 위한 정보로써 manifest.mf 파일의 해시 값, 인증서 값이 있다. 무결성 정보 중, 인증서 값으로 앱의 재패키징 여부를 판별한다.

앱 무결성 검증 시 앱 패키지명과 무결성 정보를 이용해서 앱의 변조 및 재패키징을 탐지한다. 그리고 본 논문에서 제안하는 기법은 공격자가 앱 패키지명을 변경하지 않고 유통하는 경우를 가정한다.

앱 필터링기법[4]에 무결성 검증 모듈을 적용하면 그림 2와 같다. 첫째, 웹 서버상에서 앱을 보관하는 앱 데이터베이스가 필터링 시스템에게 앱을 전달한다. 둘째, 무결성 검증모듈이 변조 및 리패키징 된 앱을 차단한다. 셋째, 무결성 검증 모듈을 통과한 앱은 특징정보 검색모듈에게 특징정보를 전달한다. 넷째, 특징정보 검색 모듈은 전달 받은 특징정보를 이용하여 검색을 실시하고, 결과 처리 모듈에게 결과를 전달한다. 마지막으로 검색한 결과를 웹 페이지에 표시한다.

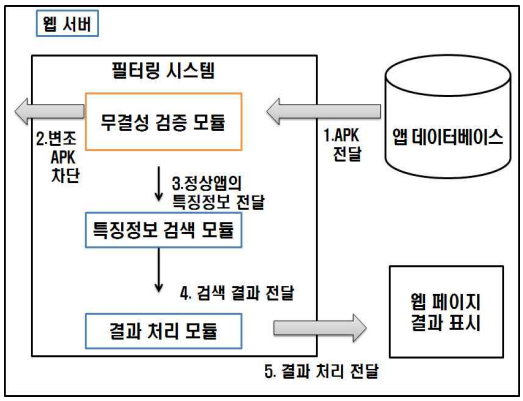


그림 2 무결성 검증 모듈이 추가된 필터링 시스템

무결성 검증 모듈은 내부적으로 두 가지 모듈로 나누어져 있다. 그림 3은 무결성 검증 모듈 내에서 2개의 무결성 정보에 대한 동작을 나타내고 있다. 무결성 검증 모듈의 동작 순서에 대한 설명은 다음과 같다.

첫째, 앱 데이터베이스가 특징정보/무결성 정보 추출 모듈에게 필터링 대상 앱을 전달하고, 앱을 필터링 할 수 있는 특징정보와 앱의 변조 여부를 판단하기 위한 무결성 정보를 추출한다. 추출된 특징정보와 무결성 정보들은 MF파일 무결성 검증모듈과 재패키징된 앱 탐지 모듈에 전달된다. 둘째, MF 파일 무결성 검증 모듈과 재패키징 된 앱 탐지 모듈은 무결성 정보를 무결성 검증 데이터베이스와 비교하여 변조 된 앱을 찾는다. 셋째, 앱의 변조 여부를 판단하여 변조된 앱은 차단한다. 앱 변조 판별은 검증 대상 앱의 패키지명, 무결성 정보와 무결성 검증 정보 데이터베이스에 저장된 앱 패키지명, 무결성 정보가 다를 경우 변조 된 앱으로 판별한다. 넷째, 재패키징 및 변조된 앱이 아닌 경우, 특징정보 추출 모듈에서 추출한 특징정보를 필터링 검색 모듈에 전달하여 그림 2와 같이 필터링 시스템 단계가 진행된다.

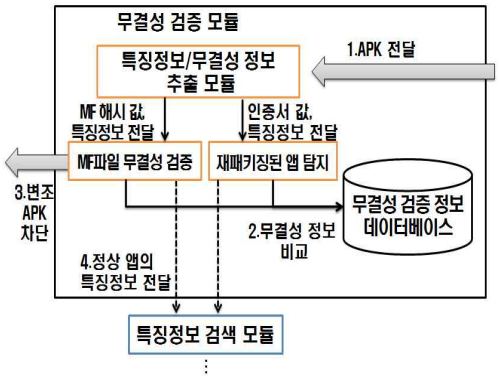


그림 3 무결성 검증 모듈의 동작원리

### 3.1 MF파일의 무결성 검증 모듈

본 논문에서 제안하는 필터링 기법에서 사용된 manifest.mf 파일과 cert.rsa 파일에 대해 살펴보면 표 1과 같다. manifest.mf, cert.rsa, cert.sf 파일은 META_INF 디렉토리에 위치하고 있다.

기존 필터링 기법[4]는 manifest.mf파일 내부에 존재하는 classes.dex 파일의 해시값으로 필터링을 하였다. 그러나 공격자가 classes.dex 파일의 해시 값을 변조하여 웹하드 서버에 앱을 업로드 할 경우, [4]의 필터링 기법을 우회할 수 있다.

표 1 META_INF 디렉토리에 위치한 파일 설명

| 파일명 | 설 명 |
|---|---|
| CERT.RSA | 서명키에 대한 인증서와 CERT.SF에 대한 서명값이 들어 있다. |
| CERT.SF | MANIFEST.MF의 각 파일에 대한 해시값을 다시 SHA1으로 해시한 값이 존재한다. |
| MANIFEST.MF | 앱 빌드 시 앱을 구성하는 파일에 대한 SHA1(Secure Hash Algorithm) 해시 값이 존재한다. |

APK(Android Package File)파일은 zip 파일 포맷 형식과 동일하기 때문에 APK파일의 확장자를 zip으로 변경하면 압축해제 할 수 있다. 압축 해제 된 APK파일은 manifest.mf 파일을 변조하여 다시 압축 할 수 있다. 이렇게 변조 된 앱은 모바일 기기에서 설치와 실행 가능하며, 정상 앱과 동일한 기능을 보인다.

본 논문에서는 manifest.mf 파일의 변조를 방지하기 위해 manifest.mf 파일에 대한 SHA1 해시 값을 산출 후, 무결성 검증 정보 데이터베이스에 저장한다. 만약 공격자가 manifest.mf 파일의 내용을 변경하게 되면 파일에 대한 SHA1 해시 값이 변경되므로, 이를 통해 manifest.mf 파일의 변조 여부를 판단할 수 있다. 즉, 추출된 manifest.mf 파일의 해시 값은 필터링 대상 앱에 대한 무결성을 보장한다.

그러나 무결성 검증 대상 앱이 버전 업데이트 시 manifest.mf 파일 해시 값도 변경되기 때문에, 각 버전 별로 manifest.mf 파일 해시 값을 추출하여 관리한다.

그림 4는 변조하기 이전의 원본 manifest.mf 파일에 대한 SHA1 해시 값을 보여주고, 그림 5는 manifest.mf 파일 속에 존재하는 SHA1 해시 값 1바이트를 변조 후, 변조된 manifest.mf 파일에 대한 SHA1 해시 값을 보여주고 있다.

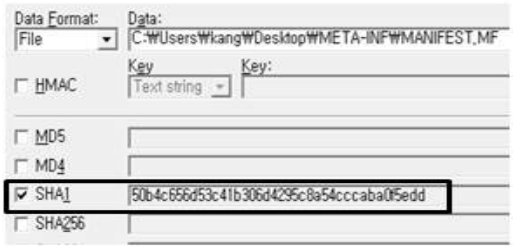


그림 4 manifest.mf 파일 내용 변경 전 SHA1 해시 값


그림 5 manifest.mf 파일 내용 변경 후 SHA1 해시 값

### 3.2 재패키징 된 앱 탐지 모듈

재패키징 된 앱 탐지 모듈은 안드로이드 앱 필터링기법의 우회를 차단하기 위해 기존 필터링 기법[4]를 보완한 것이다.

기존 앱 필터링 기법[4]에서는 3.1절에서 기술한 바와 같이 공격자가 필터링대상 앱을 변조 및 재패키징하여 웹서버에 업로드 할 경우, 필터링 시스템을 우회할 수 있다. 앱 필터링 기법[4]의 단점을 보완하기 위해, keytool[9]을 이용하여 필터링 대상 앱의 개발자의 인증서 지문을 저장하고 관리한다. keytool은 자바에서 제공하는 키 관리 유틸리티로서, 앱 서명에도 사용된다. 그림 6은 keytool로 cert.rsa 인증서를 확인한 모습이다. keytool을 사용하면 소유자, 발행자, 생산 날짜, 키 유효 기간 및 인증서 지문이 표시된다.

재패키징된 앱 탐지 모듈은 인증서 지문을 추

출 및 비교하여 수행된다. 우선 앱에 대한 인증서 지문을 추출 하여 무결성 검증 데이터베이스에 저장한다. 이후, 공격자가 불법 복제된 앱을 업로드 할 경우, 무결성 검증 데이터베이스에 저장된 인증서 지문, 앱 패키지명과 불법 복제된 앱의 인증서 지문, 앱패키지명을 비교하여 정상 앱의 지문인지 여부를 판별한다. 인증서 지문은 MD5, SHA1, SHA256으로 구성되어 있다.

```
C:₩Progran Files₩Java₩jdk1.7.0_45₩bin>keytool -printcert -file CERT.RSA
소유자: CN=NAME, OU=DAN, O=DANKOOK
발행자: CN=NAME, OU=DAN, O=DANKOOK
일련 번호: 5bf898d3
적합한 시작 날짜: Thu Dec 19 12:20:35 KST 2013, 종료 날짜: Sat Dec 12 12:20:35 K
ST 2043
인증서 지문:
        MD5: DD:33:5A:F1:E9:1F:84:C0:36:01:7F:21:A1:83:D8:B9
        SHA1: 88:87:E5:5C:C2:6C:BE:C0:35:39:06:1F:BE:8C:13:6C:66:46:03:C0
        SHA256: 5F:30:C4:E1:E8:00:26:55:53:70:D4:54:C4:8C:D8:4A:DF:DD:17:BB:5D:
B4:67:D3:3E:94:47:91:8F:55:62:51
        서명 알고리즘 이름: SHA256withRSA
        버전: 3
```

그림 6 keytool을 이용한 인증서 확인

## 4. 실험

무결성 검증 모듈을 기존 앱 필터링 기법[4]에 보완하였으므로 무결성 정보가 추출되는 소요시간과 변조 및 재패키징 된 앱의 탐지율을 측정할 필요가 있다. 성능을 측정하기 위해 구글 공식마켓인 구글 플레이에서 무료 앱 900개를 다운로드 하였다. 무결성 검증 모듈은 자바로 구현되며, Intel(R) Core(TM) i5-3570 CPU, 16GB RAM, Window 7 64bit 환경에서 성능평가를 진행 하였다. 성능 평가 실험에 사용된 프로그램은 앱이 위치한 디렉토리에서 각 앱마다 앱패키지명, 'manifest.mf', 'cert.rsa' 파일을 추출한다. 추출 된 manifest.mf 파일의 SHA1 해시 연산을 하고, cert.rsa 인증서의 지문을 keytool로 추출한다. 연산된 해시 값과, 인증서 지문은 텍스트 파일로 저장된다.

### 4.1 실험 방법

기존 앱필터링 기법에 각각의 무결성 정보(manifest.mf 해시 값, 인증서값)를 적용하여 무결성 추출 시간과 탐지율을 비교해 본다. 실험 방법은 두 가지로 나누어진다. 우선 필터링 대상 앱에서 무결성 정보를 추출하는데 소요되는 시간을 측정한다. 그리고 무결성 검증 모듈에서 각 무결성 정보에 대해 성능평가를 하였다.

첫 째, 무결성 정보를 추출하는데 소요 되는 시간을 측정을 위해서 각 필터링 대상 앱에 대해 'manifest.mf', 'cert.rsa' 파일 추출시간, 'manifest.mf' 파일의 SHA1 해시 값 산출 시간 측정, 'cert.rsa' 파일에서 keytool을 사용하여 인증서 지문 추출에 소요되는 시간을 측정 한다. 앱 개수별 특징정보 추출 소요 시간의 변화를 보기 위해 앱 개수 300개 단위로 나누어 총 3회의 실험을 진행 하였다. 실험 대상 앱 300개~900개 크기는 각각1.78GB, 3.56GB, 5.34GB 이다. 각 앱의 크기는 13KB~45MB로 다양하게 분포한다.

둘째, 필터링 대상 앱이 변조되거나 재패키징 여부를 판별을 위해 900개 앱 중 10개를 변조 및 재패키징 하였다. 무결성 검증 모듈을 통해 제안 기법이 변조된 앱을 정확하게 탐지 할 수 있는가를 살펴보았다. 단, 앱패키지명 변경은 없다고 가정한다.

### 4.2 실험 결과

첫 째, 무결성 정보를 추출한 결과는 표 2와 같다. 900개 앱에 대한 cert.rsa 지문 추출 시간은 3분 12초로, manifest.mf 파일의 SHA1 연산 시간보다 많은 소요시간이 걸렸다. 원인을 분석한 결과, cert.rsa 파일에서 지문 부분만을 추출하기 위

해선 추가적인 파싱 및 추출작업이 필요하기 때문이다.

둘째, 900개의 앱 중에서 변조 및 재패키징된 앱 10개를 100% 탐지하였다. 무결성 정보는 manifest.mf 파일의 해시 값과 cert.rsa에서 추출한 인증서 값에 대해 각각으로 이루어졌는데, 이 값들은 파일의 변조가 1byte라도 일어나는 순간 해시 값이 달라지기 때문에 보다 정확하게 변조 및 재패키징 여부를 탐지할 수 있다.

표 2는 무결성 정보 추출시간을 나타내고 있고 결과 값은 총 3회 실행 결과의 평균값이다.

표 2. 앱 개수별 무결성 정보 추출 시간(단위: 초)

| 구 분 | 300개 | 600개 | 900개 |
|---|---|---|---|
| 파일 추출 누적 시간 | 10.87 | 20.66 | 31.10 |
| 파일 추출 평균 시간 | 0.035 | 0.034 | 0.034 |
| manifest.mf 파일의 SHA1 연산 누적 시간 | 0.175 | 0.309 | 0.450 |
| manifest.mf 파일의 SHA1 연산 평균 시간 | 0.0005 | 0.0005 | 0.0005 |
| cert.rsa 지문 추출 누적 시간 | 63.92 | 127.09 | 192.14 |
| cert.rsa 지문 추출 평균 시간 | 0.21 | 0.21 | 0.21 |

표 3. 기존 특징정보 추출 시간과 무결성 정보 추출시간 비교(단위: 초)

| 구 분 | 300개 | 600개 | 900개 |
|---|---|---|---|
| 기존 특징정보 추출 시간 | 11.16 | 21.22 | 31.90 |
| manifest.mf 파일의 SHA1 연산 총 소요 시간 | 11.05 | 20.98 | 31.55 |
| cert.rsa 인증서 지문 추출 총 소요 시간 | 74.80 | 147.76 | 223.24 |

표 3은 기존 필터링 기법의 특징정보 추출시간과 각각 무결성 정보를 추출시간을 비교하였다. manifest.mf 파일은 기존 특징정보 추출시간보다 300개 앱 기준으로 0.11초 빠르게 추출되었다. 기존 특징정보 추출시간과 cert.rsa 파일의 인증서 추출 시간 보다 빠른 반면에 앱을 변조 하지 않고 공격자의 인증서로 재패키징 할 경우에는 탐지하기 어렵다.

그리고 기존 특징 정보 추출 시간보다 cert.rsa 파일에서 인증서 값을 추출하는데 300개 앱 기준으로 1분 3초 오버헤드가 발생하는 반면 cert.rsa 파일의 인증서 값을 이용하면 공격자의 인증서로 재패키징 할 경우에 탐지가 가능하다.

그림 7은 manifest.mf 파일의 해시 값 연산 결과를 나타내고 있다. 300개의 앱에 대한 해시 값 연산은 0.175초가 소요되었고, 900개의 앱에 대한 해시 값 연산은 0.450초로, 0.275초의 차이를 보이고 있다. 기존 필터링 기법의 특징정보 추출 시간보다 빠르게 추출 할 수 있다.

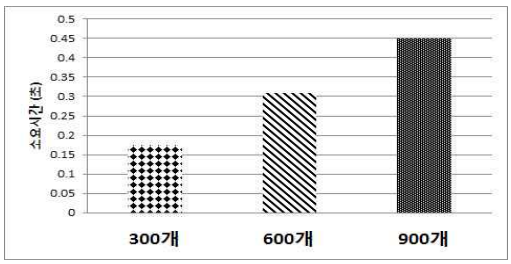


그림 7 MF 파일 SHA1 연산 누적시간

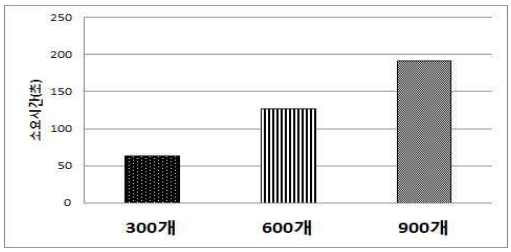


그림 8 cert.rsa 지문 추출 누적시간

그림 8은 cert.rsa 파일에서 인증서 지문을 추출되는 시간을 나타내고 있다. 300개의 앱에 대한 인증서 지문 추출 시간은 63.92초 소요되고, 900개의 앱에 대한 인증서 추출 시간은 192.14초 소요되었다. 앱의 개수가 많아질수록 인증서 지문을 추출하는데 걸리는 시간은 일정한 비율로 증가함을 알 수 있다. 즉, manifest.mf 파일의 해시 값 추출 방법이 cert.rsa 파일에서 인증서 지문 추출하는 시간보다 빠르다는 것을 알 수 있다.

## 5. 결론 및 향후 연구

최근 안드로이드 앱 시장이 확대되면서 앱의 불법 복제도 확산되고, 앱의 저작권 침해 문제가 심각하게 대두되고 있다. 이러한 문제점을 해결하기 위해 안드로이드 앱 특징정보 기반 필터링 연구가 진행 되어 왔다. 하지만 변조 및 재패키징 과정을 통해 이러한 기존의 안드로이드 앱 필터링 기법을 우회할 수 있었다. 본 논문은 필터링 대상 앱의 무결성 검증을 통해 변조 및 재패키징 된 앱을 판별하여 필터링한다. 제안된 기법은 100% 모두 판별하였다.

향후에는 기존의 안드로이드 앱 필터링 기법의 새로운 특징정보를 추가하여 보다 효과적인 필터링 기법을 연구할 예정이고, 재패키징 된 앱의 빠른 탐지를 위해 효율적인 무결성 정보를 추가할 예정이다.

## 참 고 문 헌

[1] 스마트폰 앱 불법복제 기승 --- 21.5%↑ (2012.10.10. ZDNet Korea), http://www.zdnet.co.kr/news/news_view.asp?artice_id=20121010102848&type=xml

[2] 스마트 기기를 통한 저작권 침해 실태조사 결과 발표, 박인기, 현영민, 이종섭, 한국저작권위원회 보도자료, 2011. 12.

[3] “어! 어디서 본 듯 한데” --- 소셜게임 열풍에 저작권 분쟁 불붙나“ (2012.11.19. 이데일리 뉴스) http://www.edaily.co.kr/news/NewsRead.edy?SCD=DC16&newsid=02210726599727360&DCD=A01405

[4] 웹하드 환경에서 프로그램 특징정보 기반의 불법 안드로이드 앱 필터링 기법, 정윤식, 강성욱, 조성제, 최상훈, 지현호, 제19회 한국소프트웨어감정평가학회 추계학술발표대회, 2013

[5] 안드로이드 스마트폰에서 앱 설치 정보를 이용한 리패키징 앱 탐지 기법, 전영남, 안우현, 융합보안 논문지 제12권, 제4호, 2012.09

[6] Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces, Wu Zhou, Yajin Zhou, Xuxian Jiang, Peng Ning, Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012.

[7] 동적 연결 라이브러리 기반의 MS 윈도우 애플리케이션 분류 기법, 한용만, 황윤하, 김동진, 최종천, 조성제, 유해영, 우진운, 한국소프트웨어감정평가학회 논문지, 제9권 제1호, 2013.6

[8] A Static Birthmark for MS Windows Applications Using Import Address Table, JongCheon Choi, YongMan Han, Seong-je Cho, HaeYoungYoo, Jinwoon Woo,2013 Seventh International conference on Innovative Mobile and Internet services in Ubiquitous Computing, pp.129-134, July.2013.

[9] keytool, http://docs.xrath.com/java/se/6/docs/ko/technotes/tools/solaris/keytool.html

## 저 자 소 개


**강 성 욱**

2010년 동신대학교 컴퓨터공학과 학사
2013년 3월～현재: 단국대학교 컴퓨터학과 소프트웨어보안전공 석사과정
<관심분야 : 안드로이드 애플리케이션, 악성코드 분석 등>

**정 윤 식**

2012년 단국대학교 컴퓨터과학과(이학사)
2013년 단국대학교 컴퓨터과학과 공학석사.
2013년～현재: 단국대학교 컴퓨터학과 소프트웨어보안전공 박사과정.
<관심분야 : 컴퓨터보안, 소프트웨어 보증, 시스템소프트웨어 등>

**조 성 제**

1989년 서울대학교 컴퓨터공학과 공학사
1991년 서울대학교 컴퓨터공학과 공학석사
1996년 서울대학교 컴퓨터공학과 공학박사
2001년 미국 University of California, Irvine 객원연구원
2009년 미국 University of Cincinnati 객원연구원
1997년 3월～현재: 단국대학교 소프트웨어학과/컴퓨터학과 교수
<관심분야 : 컴퓨터보안 (취약점 탐지 및 분석, 악성코드 분석), 스마트폰 보안, 소프트웨어 지적재산권 보호, 소프트웨어 보증 등>