Journal of Korea Game Society 2013 Dec; 13(6): 85-94
http://dx.doi.org/10.7583/JKGS.2013.13.6.85

게임 콘텐츠,산업,마케팅

축약된 인스트럭션 시퀀스를 이용한 안드로이드 게임 리패키징 탐지 기법

이기성, 김휘강
고려대학교 정보보호대학원
{sakabatou, cenda}@korea.ac.kr

Android Game Repackaging Detection Technique using Shortened Instruction Sequence

Gi Seong Lee, Huy Kang Kim
Graduate School of Information Security, Korea University

요 약

모바일 기기의 성능 향상과 사용자 증가에 따라 모바일 게임 시장이 확대되고 다양한 모바일 게임들이 등장하고 있다. 하지만 이와 더불어 최근 모바일 게임에 대한 다양하고 심각한 보안 위협들이 나타나고 있으며 이에 대한 대응방안 마련이 필요한 상황이다. 특히 안드로이드 환경에서 모바일 게임의 리패키징은 모바일 게임 사용자와 제작자 그리고 생태계에 심각한 문제를 야기시킨다. 본 논문에서는 축약된 인스트럭션 시퀀스를 이용하여 안드로이드 게임의 리패키징 여부를 탐지하는 기법을 제안하고 구현하였으며 실험을 통해 제안한 기법이 효과적으로 리패키징 여부를 탐지할 수 있음을 보였다. 제안한 기법은 축약된 인스트럭션 시퀀스를 이용하기 때문에 여러 제약사항이 많은 모바일 기기에 적용이 가능하며 이를 통해 다양한 경로에서 유입되는 리패키징된 앱을 탐지 및 차단하고 리패키징으로 발생하는 악성 앱의 확산과 불법복제를 예방할 수 있다.

ABSTRACT

Repackaging of mobile games is serious problem in the Android environment. In this paper, we propose a repackaging detection technique using shortened instruction sequence. By using shortened instruction sequence, the proposed technique can be applicable to a mobile device and can block repackaged apps coming from various sources. In the experiment, our technique showed high accuracy of repackaging detection.

Keywords : Android Game Security, Repackaging Detection, Instruction Sequence

Received: Oct. 25, 2013 Accepted: Nov. 19, 2013
Corresponding Author: Huy Kang Kim (Korea University)
E-mail: cenda@korea.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

모바일 기기의 성능향상과 사용자 증가에 따라 모바일 게임시장이 확대되고 다양한 모바일 게임이 등장하고 있다. 한국의 스마트폰 보급은 2013년도에 들어서 3천 만 대 시대에 돌입하였으며 모바일 게임시장은 33.8%라는 높은 성장률로 전체 게임시장에서 차지하는 비중이 크게 증가하고 있다[1,2].

하지만 모바일 게임의 증가와 더불어 모바일 게임에 대한 다양하고 심각한 위협들이 나타나고 있다. 악성코드를 포함한 모바일 게임 배포, 모바일 게임의 불법복제가 대표적이며 이에 따른 피해들이 나날이 증가하고 있는 실정이다.

주니퍼 네트웍스가 발표한 모바일 보안 위협 보고서에 따르면 2012년 3월부터 2013년 3월까지 1년간 전체 모바일 플랫폼으로부터 수집한 모바일 악성 앱이 276,259개로 전년 대비 614% 늘어난 수치를 보였으며 최근 미래창조과학부에서는 악성코드가 은닉된 다수의 모바일 게임 앱이 블랙 마켓으로 불리는 사설 앱 스토어에서 유포되고 있다는 것을 발표하였다[3,4].

늘어나는 악성 앱과 더불어 불법복제 역시 큰 문제가 되고 있다. 게임은 일반적인 컴퓨터 프로그램 이상의 가치를 지니며 저작물로 보호되어야 하지만 모바일 환경에서는 PC 환경에 비하여 저작권을 잘 보호받지 못하고 있는 상황이다[5]. 모바일 앱을 개발하는 업체를 대상으로 조사한 한국저작권위원회 설문 결과에서는 전체의 16%가 저작권 침해를 당한 적이 있다고 응답했고 54%는 저작권 침해가 위험 수준이라고 응답했다[6].

문제가 되고 있는 모바일 악성 앱의 배포와 모바일 앱의 불법복제는 대부분 리패키징을 통해 이루어진다. 리패키징이란 모바일 앱을 리버스 엔지니어링하여 코드 또는 리소스 등에 변화를 주고 다시 패키징하여 유통하는 것이며 이러한 리패키징 앱은 공식 마켓에서도 소수 발견되기도 하지만 주로 비공식 마켓, 블랙마켓, 인터넷 등에서 배포되어지고 있다.

리패키징은 안드로이드 자체의 구조적인 문제에 기인하여 발생하는 취약점을 이용하기 때문에 리패키징에 의한 피해는 대부분 안드로이드 플랫폼에서 나타난다[8,9]. 2010년 8월부터 2011년 11월까지 발견된 안드로이드 악성 앱 샘플 중 약 86%가 리패키징 기반의 앱으로 확인되었으며 안드로이드 앱 불법복제 역시 주로 리패키징을 통한 재배포에 의하여 발생한다[10].

악의적인 목적의 사용자는 리패키징을 통해 앱 내의 광고를 조작하여 금전적인 수익을 얻을 수 있고 유료 앱을 무료로 공개할 수 있으며 악의적인 코드를 추가하여 사용자의 개인정보와 위치정보 등을 유출 시킬 수 있다[11]. 특히 안드로이드 환경에서 게임은 일반 앱보다 비교적 인기가 많고 유료 앱 비율이 높기 때문에 리패키징의 주요대상이 되며 실제로 다수의 유명 게임 앱이 리패키징 되어 블랙마켓 등에서 유통되고 있다[12,13].

모바일 악성 앱의 배포와 모바일 앱의 불법복제를 막기 위해 구글은 실시간으로 공식 마켓의 앱들의 악성여부를 검사하는 Bouncer를 적용하고 앱 실행 시 사용자의 구매내역을 통해 앱의 라이선스를 인증하는 LVL(License Validation Library)를 도입하였지만 Bouncer와 LVL의 검증을 우회하는 방법들이 공개되었으며[14,15], 악성 또는 불법복제 앱들은 대부분 상대적으로 규제가 느슨한 전 세계 500개 이상의 공개 마켓과 인터넷상에 존재하는 웹 사이트 등 다양한 유통경로를 통해 배포되기 때문에 공식 마켓 차원의 대응으로는 한계가 있다.

다양한 경로에서 유입되는 리패키징 게임의 확산과 이에 따른 피해를 방지하기 위해서는 마켓이 아닌 모바일 기기에서 게임의 리패키징 여부를 탐지하여 차단하는 방법이 필요하다. 따라서 본 논문에서는 축약된 인스트럭션 시퀀스의 유사도를 이용하여 모바일 앱의 리패키징 여부를 탐지하는 기법을 제안하고 다수의 모바일 게임 앱을 리패키징하여 구현한 기법의 탐지수준을 평가한다.

해당 기법은 기존의 연구들과 달리 축약과정을 거친 인스트럭션 시퀀스를 사용하기 때문에 저장공

간, 통신비용 등 여러 제약사항이 많은 모바일 기기에 적용이 가능하며 다양한 경로에서 다운로드되는 모바일 게임의 리패키징 여부를 효과적으로 탐지할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 배경 및 관련 연구에 대해 기술한다. 3장에서는 리패키징 탐지 기법을 제안 및 설명하고 4장에서는 실험을 통해 구현한 기법의 탐지수준을 평가한다. 5장에서는 결론을 기술한다.

2. 배경 및 관련 연구

구글은 악성 앱의 배포를 차단하기 위해 마켓에 Bouncer를 도입하였다. Bouncer는 마켓에 새로 등록된 앱을 자동 분석해 악성코드 포함여부를 탐지하는 시스템으로 가상화 기술을 이용하여 실행 환경을 구성하고 직접 새 앱을 설치 및 실행하여 분석한다. 하지만 곧 Bouncer의 가상 실행 환경을 파악하여 이를 우회방법이 보안 컨퍼런스 Summer-con 2012에서 Jon Oberheide에 의해 발표되었다[14].

더불어 구글은 앱 불법복제를 막기 위해 LVL을 도입하였다. LVL은 사용자가 앱을 사용할 수 있는 권한이 있는지 인증하는 기능을 지원하는 라이브러리로, 사용자가 앱을 구매할 때 구글의 마켓 라이센스 서버에 구매 내역이 저장되고 사용자가 앱을 실행할 때 마켓 라이센스 서버가 마켓 앱과 연동하여 사용자가 해당 앱을 실행할 정당한 권한을 가졌는지 확인한다. 하지만 마켓 앱으로부터 수신한 메시지를 확인하는 부분의 실행 코드를 간단히 조작하는 방법으로 우회할 수 있다[15].

공식 마켓이 아닌 비공식 마켓이나 기타 공개 마켓에서 앱을 보호하기 위해서는 난독화 도구를 사용할 수 있다. 구글이 제공하는 난독화 도구 Proguard는 클래스명, 필드명, 메소드명 등을 식별자 변환 기법을 통해 난독화한다. 이 기법은 리버스 엔지니어링을 통한 코드 분석을 어렵게 하지만 실행코드의 제어흐름은 변경하지 않기 때문에 리버스 엔지니어링을 통한 코드 분석을 완전하게 방지할 수는 없다[16].

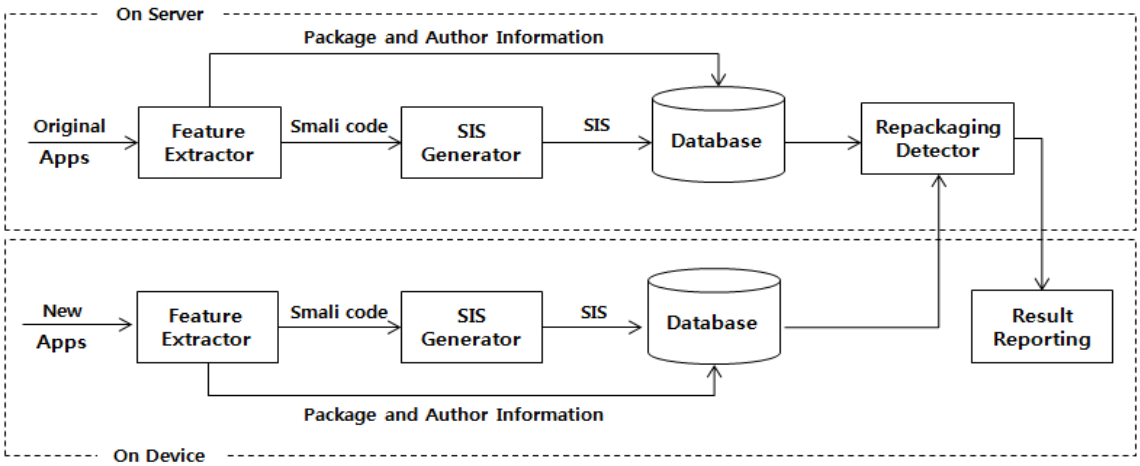
안드로이드 실행파일을 암호화된 상태로 저장하여 리버스 엔지니어링을 막는 기술은 매번 복호화에 따른 성능의 오버헤드가 발생하는 단점이 있고, 앱의 불법 실행을 차단하기 위해 앱의 일부분을 실행 시 마다 서버로부터 전송받아 동적적재하여 실행하는 기법은 네트워크에 의존적이기 때문에 불안정한 모바일 무선 네트워크 환경에서는 실행되지 못하거나 지연될 수 있는 단점이 존재한다[17].

이찬희 등의 연구[17]에서는 접근 제어 기법을 사용하여 기기에서 앱을 추출하는 것을 방지하며 이를 통하여 앱의 리패키징을 사전에 예방한다. 하지만 해당 접근 제어 기법을 모든 기기들에 일괄적으로 적용시키기 어려우며 이미 추출되거나 유통되고 있는 앱들에 대한 대응을 할 수 없다.

AppInk[18]는 개발자가 원본 앱에 동적 워터마킹 코드를 삽입하고 검증 시 원본 앱과 쌍이 되는 앱을 워터마크 인식기를 운영하는 중재자에게 제출하여 리패키징의 여부를 탐지하는 기법이다. 해당 기법은 개발 과정에서 쉽게 적용할 수 있는 장점이 있지만 이미 다양한 경로로 유통되고 있는 많은 수의 앱에 대한 리패키징 여부를 탐지할 수 없다.

DroidMOSS[11]는 공식 마켓에 존재하는 앱과 비공식 마켓의 앱에서 인스트럭션 시퀀스를 추출하고 퍼지 해싱을 적용한 결과의 유사도를 비교하여 리패키징 여부를 탐지한다. 해당 시스템은 변조된 구획을 탐지할 수 있는 장점을 지니지만 마켓 차원의 탐지 시스템이기 때문에 다양한 경로에서 유입되는 리패키징된 앱을 효과적으로 차단할 수 없다.

다양한 경로에서 유입되는 리패키징된 앱을 차단하기 위해 RePAD[19] 기법은 정상 앱의 정보를 유지하고 있는 서버와 연동하여 사용자의 기기에서 앱이 다운로드 될 때 패키지 명과 앱의 해시 값을 비교하며 패키지 이름이 같지만 앱의 해시 값이 다른 경우를 리패키징된 앱으로 탐지한다. 하지만 패키지명은 변조될 수 있으므로 해당 기법에 의한



[Fig. 1] Overview of Repackaging Detection Technique

탐지를 쉽게 우회할 수 있다.

따라서 쉽게 탐지를 우회할 수 없고 다양한 경로로부터 유입되는 리패키징된 앱을 차단할 수 있는 방안이 필요하며 본 논문에서는 축약된 인스트럭션 시퀀스를 이용하여 기기에 존재하는 앱의 리패키징 여부를 탐지하는 기법을 제안한다.

3. 본 론

3.1 탐지 기법 개요

리패키징 탐지 기법의 전체 구성은 [Fig. 1]과 같다. 서버는 보호대상 앱들을 입력으로 취하고 각 앱의 패키지 정보, 저자 정보, 축약된 인스트럭션 시퀀스를 유지 및 관리하는 데이터베이스를 운영하며 기기에서 리패키징 여부 검사에 대한 요청이 들어오면 기기에서 전송된 축약된 인스트럭션 시퀀스의 유사도와 저자 정보를 서버 데이터베이스의 정보와 비교하여 리패키징 여부를 판별하고 기기로 결과를 전달한다.

해당 기법은 세 가지 주요 구성요소를 기반으로 동작한다. Feature Extractor는 앱의 패키지 정보, 저자 정보, 디스어셈블된 코드를 추출하며 SIS Generator는 디스에셈블된 코드를 축약된 인스트럭션 시퀀스로 만들어준다. Repackaging Detector는 서버에 저장된 앱들의 정보와 기기에서 요청한 앱의 정보를 비교하여 리패키징 여부를 돌려준다.

사용자의 모바일 디바이스에서 새로운 앱이 설치되면 앱의 패키지 정보, 저자 정보, 축약된 인스트럭션 시퀀스를 서버로 전송하게 되고 서버는 리패키징 여부를 사용자에게 알려준다. 추가적으로 저작권을 소유하고 있는 게임사 또는 모바일 백신을 운영하는 보안업체에게 해당 정보를 전달하여 게임 앱의 불법복제 현황을 파악하고 악성 및 불법복제 앱의 확산을 차단할 수 있다.

3.2 Feature Extractor

안드로이드 앱인 APK 파일은 확장자가 '.APK'인 압축파일(ZIP)이며 여러 파일들로 구성되어 있다. Feature Extractor는 APK 파일의 압축을 해제하고 앱의 패키지 정보와 저자 정보를 추출하여 데이터베이스에 저장하며 해당 앱의 디스어셈블된 코드들을 생성한다.

AndroidManifest.xml 파일은 패키지 이름과 버전 등 어플리케이션에 대한 설명 및 실행권한 등의 정보를 지니는 XML 형태의 파일이다. Feature Extractor는 AndroidManifest.xml 파일을 바이너리 형태에서 텍스트 형태로 변환하고 패키지 이름

을 추출하여 데이터베이스에 저장한다.

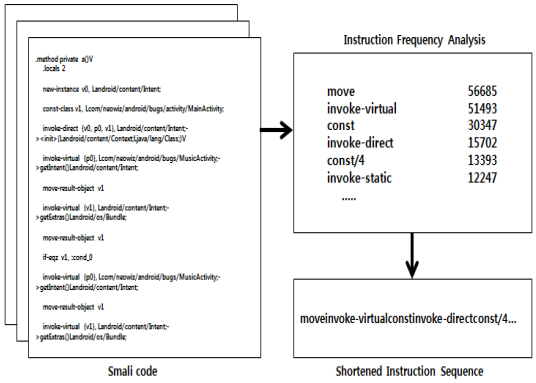
META-INF 폴더에는 일반적으로 확장자가 '.MF'인 Manifest File과 '.SF'인 Signature File, '.RSA' 또는 '.DSA'인 Signature Block File이 포함되어 있다. Manifest File은 앱 내의 파일들의 이름과 파일 별 다이제스트가 기록되어 있으며 Signature File에는 Manifest File과 관련된 다이제스트들이 포함되어 있다. Signature Block File은 저자의 개인키로 만들어진 SF 파일에 대한 전자서명과 저자의 공개키를 포함한 X.509 형식의 인증서가 합쳐진 형태의 바이너리 파일이며 Feature Extractor는 이 파일에서 저자 공개키의 해시 값을 추출하여 데이터베이스에 저장한다.

Classes.dex 파일은 JAVA 기반의 클래스 파일을 DEX(Dalvik Executable) 형태로 변환한 것으로 가상 머신인 Dalvik에서 동작하는 바이너리 실행 파일이다. Feature Extractor는 Classes.dex 파일을 디스어셈블하여 확장자가 '.SMALI'인 Smali 코드를 가진 파일들을 생성해낸다.

3.3 SIS Generator

Smali 코드는 분석이 용이하도록 DEX 파일의 바이너리를 디스어셈블하여 텍스트 형태로 변환한 코드이다. DEX 파일의 바이너리는 Dalvik 바이트코드로 이루어져 있으며 이는 Dalvik 가상머신이 실행하는 인스트럭션의 형태이다. 각 바이트코드는 1 바이트의 크기를 가지므로 256 종류의 인스트럭션을 사용할 수 있지만 일부만 실제로 사용된다.

SIS Generator는 Feature Extractor가 생성한 Smali 파일들의 Smali 코드에서 인스트럭션을 종류별로 검색하여 각 인스트럭션의 출현 빈도를 계산한 후 인스트럭션을 빈도가 높은 순으로 나열하여 인스트럭션 시퀀스를 생성한다. [Fig. 2]는 축약된 인스트럭션의 생성 절차를 보여준다.



[Fig. 2] Generation procedure of SIS

기존의 인스트럭션 시퀀스는 인스트럭션이 출현할 때마다 나열하여 시퀀스로 만들기 때문에 생성된 인스트럭션 시퀀스의 길이가 매우 길어 인스트럭션 시퀀스 간의 유사도를 비교하기 위해서는 퍼지 해싱 등의 추가적인 처리절차를 거쳐야만 했다. 하지만 빈도 순으로 나열된 인스트럭션 시퀀스는 인스트럭션의 종류별로 한 번씩 나타나기 때문에 인스트럭션 시퀀스의 길이가 큰 폭으로 줄어들게 된다. 이와 같이 축약된 인스트럭션 시퀀스는 저장공간, 통신비용 등 여러 제약사항이 많은 모바일 디바이스에 적용하기에 적합하다.

또한, 대부분 앱에서 인스트럭션마다 빈도 수치가 매우 높게 나타나기 때문에 일부의 코드만을 추가하거나 변경하거나 삭제하는 리패키징에 큰 영향을 받지 않고 인스트럭션 시퀀스의 형태를 유지할 수 있다. 악의적인 목적의 사용자가 각 인스트럭션의 빈도 순을 이용한 해당 기법의 축약된 인스트럭션 시퀀스를 바꾸려면 여러 종류의 인스트럭션을 많은 양으로 추가하거나 삭제해야만 한다. 따라서 탐지를 우회하는 것이 어렵다.

3.4 Repackaging Detector

리패키징 앱은 기존 앱의 코드 등을 변조하여 자신의 서명으로 다시 패키지하는 과정을 거치므로 일반적으로 같거나 유사한 코드를 가지고 다른 저자 정보를 가진다. 따라서 Repackaging Detector

에서는 디바이스에서 전달된 축약된 인스트럭션 시퀀스를 서버의 데이터베이스에 저장된 축약된 인스트럭션 시퀀스들과 유사도를 비교하여 점수를 측정한 후, 가장 높은 점수를 가진 축약된 인스트럭션 시퀀스의 앱을 선정하고 선정된 앱과 디바이스에서 전달된 앱에서 추출된 저자 정보를 비교하여 최종적으로 리패키징 여부를 판단한다.

축약된 인스트럭션 시퀀스는 각 인스트럭션의 빈도 수치를 통해 정렬된 인스트럭션의 순서 정보가 가장 중요하다. LCS(Longest Common Subsequence) 알고리즘은 연속되지는 않지만 순서 정보가 일치하는 가장 긴 부분 시퀀스를 추출하기 때문에 축약된 인스트럭션 시퀀스의 유사한 부분을 추출하는데 적합한 방법이다. Repackaging Detector에서는 [Fig. 3]과 같이 축약된 인스트럭션 시퀀스에 적합하게 개선된 LCS 알고리즘[20]을 사용하여 두 시퀀스 간의 유사 시퀀스를 추출한다.

```
LCS(X, Y) : Y is the shorter one
m=Y.length
n=X.length
k=SYMBOL_SIZE
let map[1...k] and c[1...m] be new arrays
for i=1 to m
    if map[xi]!=0
        max=0
        for j=1 to map[xi]-1
        if max<x[j]
            max=c[j]
        c[map[xi]]=max+1
rerurn c

PRINT_LCS(c, Y)
m= Y.length
max=1
for i=1 to m
    if c[max] < c[i]
        max=I
for i=max to 1
    if c[i]==c[max]
        print xi
        max=max-1
```

[Fig. 3] Improved LCS Algorithm

또한, 추출된 유사 시퀀스을 이용하여 유사도를 도출하기 위해 Jaccard Coefficient를 변형한 (Formular 1)을 사용한다. 해당 식은 결과로 0과 100사이의 값을 가지게 되며 해당 값이 가장 높은 축약된 인스트럭션 시퀀스가 가장 유사한 코드를 가진 앱이 된다. Repackaging Detector는 유사도 결과 중 가장 높은 유사도를 효율적으로 탐색하기 위해 일정 유사도 이하의 값은 탐색에서 제외하는 임계치를 둔다. 임계치가 높으면 미탐이 발생할 수 있으며 임계치가 낮으면 탐색의 효율이 감소하기 때문에 적정한 수준의 임계치를 선정하는 것이 필요하다.

$$S = \frac{|LCS(X, Y)|}{|X| + |Y| - |LCS(X, Y)|} \times 100$$

(Formular 1) Calculation of similarity score

4. 실험 및 결과

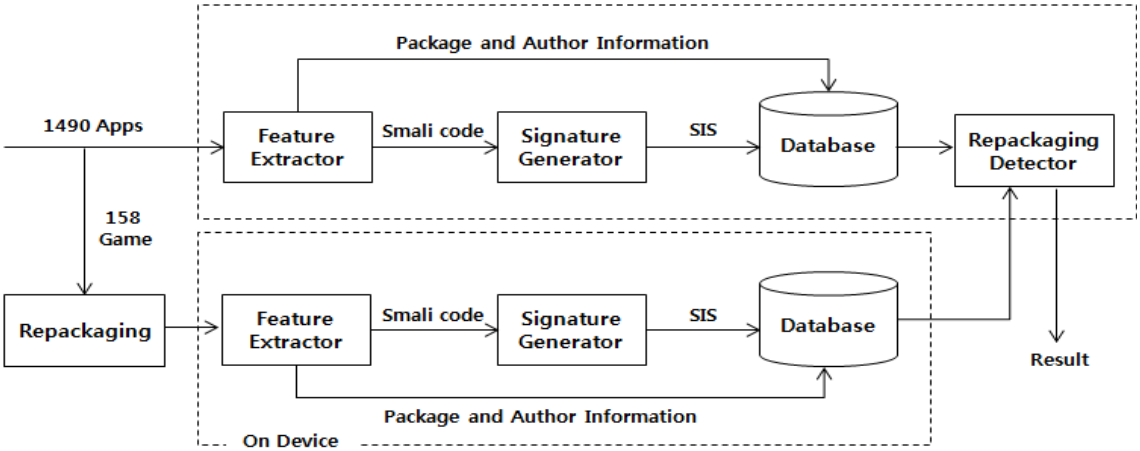
4.1 실험 개요

본 실험에서는 [Fig. 4]와 같이 공식 마켓에서 수집한 1490개의 정상 앱과 정상 앱 중 158개의 게임을 4가지 방법으로 리패키징한 앱을 제안한 기법에 적용하여 리패키징 여부 탐지를 수행하고 결과를 분석하였다.

제안한 기법의 서버는 Feature Extractor, SIS Generator, Repackaging Detector를 파이썬으로 구현하였으며 데이터베이스는 MySQL을 사용하였다. 제안한 기법의 디바이스는 Feature Extractor, SIS Generator를 자바를 사용하여 앱의 형태로 구현하였으며 데이터베이스는 SQLite를 사용하였다.

1490개의 정상앱은 한국 구글 플레이스토어에서 무료 인기 앱 목록을 바탕으로 구글 API를 이용하여 2013년 1월 6일 4시 9분부터 2013년 7월 18일 11시 38분까지 수집하였으며 이 중 158개의 게임 앱을 모바일 백신의 스트레스 테스트 용도로 제작된 자동화 도구 ADAM[21]을 이용하여 리패키징하였다.

ADAM은 원본 앱의 인스트럭션을 변경하지 않고 리패키징을 수행하는 3가지 방법과 난독화를 적용하여 인스트럭션을 변경하는 4가지 방법을 사



[Fig. 4] Experimental overview

용하여 리패키징된 앱을 생성한다. 본 실험에서는 인스트럭션 변경에 대한 제안한 기법의 저항성을 평가해야 하므로 4가지 난독화를 적용한 리패키징된 앱을 생성하고 제안한 기법에 적용하여 정확도와 오류율을 평가한다.

4.2 실험 결과

[Table 1] Repackaging result

Obfuscation method	Sucess number	Sucess rate
Insert defunct methods	154	97.47
Renaming methods	155	98.10
Changing control flow graphs	126	79.75
Encrypting constant strings	61	38.61

[Table 1]은 리패키징에 성공한 개수와 비율을 나타내고 있으며 [Table 2]는 리패키징 탐지의 정확도와 오류율이다. ADAM은 앱의 인스트럭션에 변화를 주지만 패키지명은 변경하지 않으므로 본 실험에서는 리패키징된 것으로 판별된 게임 앱의 패키지명과 원본 게임 앱 패키지명의 추가 비교를 통해 제안한 기법의 정확도를 평가하였다. 임계치는 70으로 두고 실험을 진행하였으며 해당 임계치에서 미탐은 발생하지 않았고 일부 오탐이 발생하여 이를 오류로 처리하였다.

[Table 2] Repackaging detection result

Obfuscation method	Accuracy	Error rate
Insert defunct methods	93.51	6.49
Renaming methods	93.55	6.45
Changing control flow graphs	92.06	7.94
Encrypting constant strings	88.52	11.48

[Table 3]은 각 리패키징 방법마다의 최고 유사도의 평균값과 전체 유사도의 평균값을 나타내고 있다. 제안한 기법은 인스트럭션만을 사용하기 때문에 앱 내에 포함된 메소드의 이름을 변경하는 Renaming method의 경우 최고 유사도가 완전히 일치하였으며 나머지 리패키징 방법들에서는 최고 유사도 평균이 약 90에서 98로 나타났다. 앱 내의 문자열들을 암호화하여 인스트럭션의 변경을 발생시키는 Encrypting constant strings 방법은 90.57로 가장 낮은 유사도를 기록하였다.

[Table 3] Similarity average

Obfuscation method	Maximum similarity	Total similarity
Insert defunct methods	95.661	41.59
Renaming methods	100	41.77
Changing control flow graphs	98.37	41.68
Encrypting constant strings	90.57	41.44

최고 유사도의 차이는 난독화에 따른 인스트럭션 빈도 순위의 변경에 의하여 발생한다. 하지만 최고 유사도 평균의 가장 낮은 수치인 90.57과 해당 기법의 전체 유사도 평균 수치인 41.44와는 간극이 크기 때문에 제안한 기법은 안정적으로 리패키징 여부를 판별할 수 있다. 기존의 인스트럭션 시퀀스는 출현 순서대로 인스트럭션을 나열하기 때문에 제어흐름을 변경하는 등의 난독화에 의한 변화가 심하지만 축약된 인스트럭션 시퀀스는 빈도 순위에 기반하기 때문에 이에 강한 특성을 보인다.

하나의 앱 당 인스트럭션 종류별 빈도 수치는 매우 큰 값을 가지기 때문에 빈도 순위를 변경시켜 제안한 기법을 우회하려면 다양한 종류의 인스트럭션을 매우 많은 수치로 삽입하거나 변경하거나 삭제해야만 한다. 따라서 제안한 기법은 게임 앱의 리패키징 여부를 안정적이고 효과적으로 탐지할 수 있다.

5. 결 론

본 연구에서는 다양한 경로에서 유입되는 리패키징된 앱을 탐지하기 위해 축약된 인스트럭션 시퀀스를 이용하여 안드로이드 게임의 리패키징 여부를 탐지하는 기법을 제안하고 구현하였으며 실험을 통해 제안한 기법이 효과적으로 리패키징 여부를 탐지할 수 있음을 보였다.

제안한 기법은 축약된 인스트럭션 시퀀스를 이용하기 때문에 저장공간, 통신비용 등 여러 제약사항이 많은 모바일 디바이스에 적용이 가능하며 이를 통해 다양한 경로에서 다운로드되는 악성 앱 또는 불법복제 앱의 확산을 선제적으로 차단할 수 있다.

또한, 제안한 기법에서 사용하는 축약된 인스트럭션 시퀀스는 인스트럭션의 빈도 순위를 사용하기 때문에 기존의 인스트럭션 시퀀스에 비하여 난독화에도 강한 면모를 보이며 인스트럭션의 빈도 순위를 크게 변경하기 힘들기 때문에 기존의 인스트럭션 시퀀스를 사용하는 시스템에 비하여 탐지를 우회하기 어려운 특성을 가진다.

본 연구에서 제안하는 기법은 단순히 리패키징의 여부만을 신속하게 판별하기 때문에 변조된 부분을 식별하거나 변조된 코드를 구분하기 어렵다. 효율적인 분석 및 대응을 위해서는 리패키징 여부 탐지와 더불어 변조된 부분을 식별하고 변조된 코드의 유형을 분류하는 과정이 요구된다.

ACKNOWLEDGMENTS

This research was supported by the MSIP(Ministry of Science, ICT & Future Planning), Korea, under the “Employment Contract based Master's Degree Program for Information Security” supervised by the KISA(Korea Internet Security Agency) (H2101-13-1001).

REFERENCES

[1] Jae-Hong Lee, “The Study on Arbitration of Contents Dispute in Mobile Game”, Journal of Korea Game Society, Vol. 13, No. 3, 2013.
[2] Ministry of Culture, Sports and Tourism, “White Paper on Korean Games”, Oct. 2012.

[3] Juniper Networks Mobile Threat Center, “Third Annual Mobile Threats Report: March 2012 through March 2013”, May 2013.
[4] Microsoftwre, “Concealing malicious code... mobile game in Black Market”, August 2013.
[5] Jisun Choi, “Protecting Game Developers Under the Works-for-hire Clause of Copyright Law”, Journal of Korea Game Society, Vol. 11, No 4, 2012.
[6] DigitalDaily, “Copyright Committee, Defend Smart Industry”, Dec. 2011
[8] Huy Kang Kim, Young Jun Kum, “Security issue of mobile game service on Android environment”, Review of Korea Institute of Information Security and Cryptology, Vol. 23, No. 2, April 2013.
[9] Soonil Kim, Sunghoon Kim, Dong Hoon Lee, “A Study on the vulnerability of integrity verification functions of android-based smartphone banking application”, Journal of Korea Institute of Information Security and Cryptology, Vol. 23, No. 4, August 2013.
[10] Y. Zhou, X. Jiang, “Dissecting Android Malware: Characterization and Evolution”, In Proc. of the 33rd IEEE Symposium on Security and Privacy, May 2012.
[11] Wu Zhou, Yajin Zhou, Xuxian Jiang, Peng Ning, “Detecting Repackaged Smartphone Application in Third-Party Android Marketplaces”, In Proc. of the 2nd ACM Conference on Data and Application Security and Privacy, February 2012.
[12] Xuxian Jiang, “Security Alert: New Sophisticated Android Malware DroidKungFu Found in Alternative Chinese App Markets”, June 2011.
[13] INCA Internet, “Repackaged FastRacing game application leaks your smartphone information”, June 2011.
[14] Kaspersky Lab. Security News, “Researchers Find Methods For Bypassing Google's Bouncer Android Security”, June 2012.
[15] Justin Case, “Report: Google's Android Market License Verification Easily Circumvented, Will Not Stop Pirates”, August 2010.
[16] Yuxue Piao, Jin-hyuk Jung, Jeong Hyun Yi, “Structural and Functional Analyses of Proguard Obfuscation Tool”, The Journal of Korea Information and Communication Society, Vol. 38, No. 8, August 2013.
[17] Chan-Hee Lee, Yeong-Ung Park, Ji-Hyeng Lim, Hong-Geun Kim, Choong-Hyun Lee, Seong-Je Cho, Jaesoo Yang, “Access Control Mechanism Preventing Application Piracy on the Android Platform”, Journal of Computing Science and Engineering, Vol. 18, No. 10, Oct. 2012.
[18] Wu Zhou, Xinwen Zhang, Xuxian Jiang, “AppInk: Watermarking Android Apps for Repackaging Deterrence”, In Proc. of the 8th ACM SIGSAC symposium on Information, computer and communications security, May 2013.
[19] Young Nam Joun, Woo Hyun Ahn, “Detecting Repackaged Applications using the Information of App Installation in Android Smartphones”, Journal of Convergence Security, Vol. 12, No. 4, September 2012.
[20] BooJoong Kang, Yeoreum Lee, Eul Gyu Im, “Malware Family Detection Method using Instruction Frequency Sequence”, Journal of Security Engineering, Vol. 10, No. 1, February 2013.
[21] Min Zheng, Patrick P.C.Lee, John C.S.Lui, “ADAM: An Automatic and Extensible Platform to Stress Test Android Anti-Virus System”, In Proc. of the 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 2012.

이 기 성 (Lee, Gi Seong)

2012년 2월 공주대학교 정보통신공학과 졸업
2012년 3월-현재 고려대학교 정보보호대학원 석사과정

관심분야 : 모바일게임 보안, 시스템 보안,
네트워크 보안, 악성코드 분석

김 휘 강 (Kim, Huy Kang)

1998년 2월 KAIST 산업경영학과 학사
2000년 2월 KAIST 산업공학과 석사
2009년 2월 KAIST 산업및시스템공학과 박사
2004년 2월-2010년 2월 엔씨소프트 정보보안실장
2010년 3월-현재 고려대학교 정보보호대학원 조교수

관심분야 : 온라인게임 보안, 네트워크 보안,
네트워크 포렌식