안드로이드 앱 도용 탐지를 위한 API 유사도 비교 도구 구현

최성하, 이현영, 조승민, 박희완
한라대학교 정보통신방송공학부
shoutme1991@nate.com, suddlze@hanmail.net, alonejsm@nate.com,
heewanpark@halla.ac.kr

API Similarity Comparison Tool Development for Detecting Theft of Android Application

Sung-Ha Choi, Hyun-Young Lee, Seung-Min Cho, Heewan Park
Communication and Broadcasting Engineering, Halla University

요 약

최근 오픈 소스 커뮤니티가 활성화되고 수많은 오픈 소스들이 공개되고 있어서 많은 개발자들이 오픈 소스를 활용하고 있다. 그러나 오픈 소스도 정해진 라이선스 기반으로 공개되므로 오픈 소스를 사용할 때는 반드시 라이선스를 확인해야 한다. 본 논문에서는 안드로이드 앱의 라이선스 위반이나 코드 도용을 확인할 수 있는 방법으로서 안드로이드 앱 사이의 API 메소드 호출 유사도를 측정하는 방법을 제안한다. 원본 프로그램과 도용된 프로그램은 유사한 API 메소드를 사용할 것임을 예상할 수 있기 때문에 API 메소드 호출이 유사한 것을 확인하면 간접적으로 코드 도용을 확인할 수 있다. 본 논문에서 개발한 API 유사도 측정 도구는 안드로이드 앱의 소스 코드를 필요로 하지 않고, 안드로이드 달빅(Dalvik) 바이트 코드로부터 직접 API 호출 명령어를 분석하여 유사도를 측정한다는 특징이 있다. 본 논문에서 구현한 도구의 평가를 위해서 API 호출 유사도 비교 실험을 수행하였다. 그 결과, 실제로 API 호출 유사도가 높았던 두 앱이 서로 공통된 모듈을 포함하고 있음을 밝혀내었다. 그리고 선행 연구에서 제안했었던 안드로이드 달빅 코드 전체에 대한 유사도 비교 도구보다 비교 속도가 35% 정도 향상된 것을 확인하였다.

1. 서론

최근 인터넷 환경이 발달하고 오픈 소스 활용에 대한 관심이 증가하면서 인터넷의 오픈 소스 커뮤니티가 활성화되고 있다[1]. 커뮤니티를 통해서 공개된 다양한 오픈 소스 프로젝트는 누구나 자유롭게 사용할 수 있도록 공개된 소스 코드이다. 그러나 각각의 프로젝트마다 특정한 라이선스 기반으로 배포되고 있음을 주의해야만 한다.

예를 들어, GPL 기반으로 배포되는 오픈 소스의 경우에는 GPL 오픈 소스를 활용하여 새롭게 개발된 프로젝트의 소스 코드도 반드시 GPL 기반으로 공개해야 하는 의무가 있다[2]. 이것을 어기고 소스 코드를 공개하지 않는다면 라이선스 위반이 되어 법적인 제제를 받을 수 있다[3]. 즉, 오픈 소스는 비록 누구에게나 공개된 소스이기는 하지만 오픈 소스를 사용할 때는 반드시 정해진 라이선스를 준수해야만 한다.

그러나 오픈 소스가 언제 어느 소프트웨어에서 라이선스를 무시하고 무분별하게 사용되고 있는지를 판단하기는 매우 어렵다.

코드 표절 검사 도구를 사용하여 오픈 소스와의 유사도를 비교하려면 반드시 소스 코드를 먼저 확보하여야 한다. 그러나 단순히 라이선스 위반이 의심된다고 해서 개발사에게 소스 코드를 요구하기도 어렵고, 무분별하게 저작권 분쟁 위원회를 통해서 해결하는 것도 쉽지 않다.

따라서 프로그램 소스 코드를 확보할 수 없는 경우에도 소프트웨어 바이너리만으로 간접적으로 코드 도용 여부를 판단할 수 있는 방법이 있다면 유용하게 사용될 수 있을 것이다.

소프트웨어 버스마크는 소스 코드를 필요로 하지 않으면서 소프트웨어 바이너리나 자바 클래스 파일로부터 고유한 특징을 추출하여 유사도를 비교하는 방법이다.[4,5]

만일 소프트웨어 버스마크를 통해서 코드 도용이 의심되는 두 소프트웨어를 비교했을 때 유사도가 높게 측정되었다면 코드 도용을 의심할 수 있는 근거가 될 수 있다.

본 논문에서는 안드로이드 앱에 대하여 API 메소드 호출 유사도를 비교하는 버스마크 기법을 제안한다. 그리고 안드로이드 앱의 달빅(Dalvik) 코드[6] 전체에 대한 유사도 비교를 수행하는 선행 연구[7]와의 비교를 통해서 버스마크 추출 속도와 유사도 측정 결과를 평가한다.

2. 관련연구

프로그램의 소스 코드 기반이 아닌 바이너리에 대한 직접적인 유사도 비교 방법으로 제안된 개념이 소프트웨어

버스마크[4,5]이다. 소프트웨어 버스마크는 DNA와 유사한 성격을 갖는다. 즉, 프로그램이 갖고 있는 고유한 특성이기 때문에 유사한 프로그램을 식별하는데 사용될 수 있다.

Tamada는 자바 클래스 파일에 대한 버스마크 기법[4]을 제안하였다. 이 기법은 자바 클래스에서 사용된 상수 값들의 시퀀스 정보(Constant Values in Field Variables), 자바 메소드 호출 시퀀스 정보(Sequence of Method Calls), 클래스 상속 구조 정보(Inheritance Structure), 사용된 클래스 정보(Used Class)의 4가지 버스마크로 구성된다.

클래스 상속 구조 정보와 사용된 클래스 정보는 다양한 프로그램 변환 기법에도 버스마크가 쉽게 손상되지 않는다는 장점이 있다. 그러나 클래스가 서로 비슷한 상속 구조를 갖기도 하고, 또한 사용된 클래스 정보도 유사한 경우가 많기 때문에 서로 다른 알고리즘을 구현한 두 프로그램과 같은 경우를 구별하기 힘들다는 단점이 있다.

Myles는 k-gram 버스마크[5]를 제안하였다. k-gram 버스마크에서는 자바 클래스를 이루고 있는 JVM 명령어들을 길이가 k인 조각으로 잘라서 버스마크로 사용한다.

k-gram 버스마크는 JVM 명령어 레벨에서 유사도를 비교하기 때문에 서로 다른 프로그램을 구별하는 능력은 매우 뛰어나다. 그러나 유사 기능을 수행하는 다른 명령어로 변경되거나 명령어 순서가 뒤바뀌는 경우가 발생할 수 있기 때문에 버스마크가 쉽게 손상된다.

최근 자바뿐만 아니라, 안드로이드 앱을 대상으로 하는 소프트웨어 버스마크도 새롭게 연구가 시작되었다. 안드로이드 앱을 대상으로 하는 선행 연구[7]에서는 프로그램을 구성하고 있는 모든 달빅 바이트 코드를 추출하여 유사도를 비교하였다. 따라서 버스마크 추출과 비교에 많은 시간이 소요된다는 문제가 있었다. 오픈 소스 프로젝트와 같이 수많은 소프트웨어를 대상으로 유사도 비교를 수행한다면 유사도 비교 속도가 중요한 문제가 될 수 있다.

본 논문에서는 달빅 코드를 구성하는 모든 명령어를 추출하는 것이 아니라 API 메소드 호출과 관련된 달빅 명령어만을 추출하여 유사도를 비교하는 버스마크 기법을 제안하고 실험을 통해서 속도 및 도용 탐지 능력을 평가한다.

3. 안드로이드 API 유사도 기반 버스마크

3.1 소프트웨어 버스마크

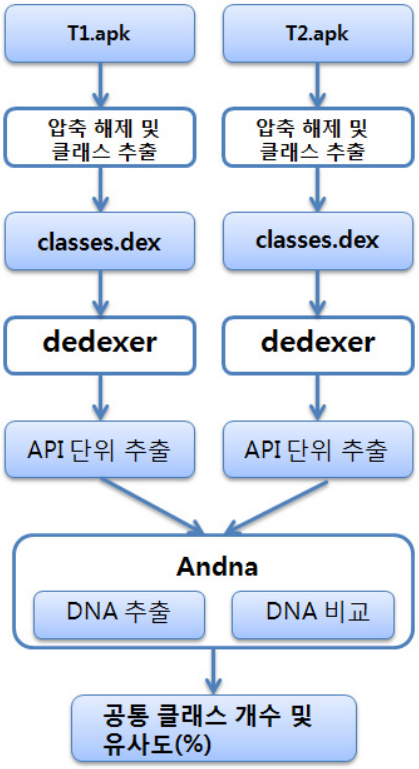
소프트웨어 버스마크에 대한 정의[4]는 다음과 같다.

정의 (소프트웨어 버스마크)
프로그램 p와 q에 대한 함수 f가 다음 조건을 만족할 때, $f(p)$를 프로그램 p의 버스마크라고 한다.
조건 1. $f(p)$는 부가적 정보 없이 p 자신에서부터 얻는다.
조건 2. 프로그램 p와 q가 서로 코드 도용 관계에 있다면 $f(p) = f(q)$ 이다.

소프트웨어 버스마크는 소프트웨어로부터 직접 추출할 수 있어야 한다. 그리고 두 소프트웨어가 서로 도용 관계에 있다면 버스마크는 서로 같다. 여기서 역은 성립하지 않는다. 즉, 버스마크가 같다고 해서 반드시 두 프로그램이 서로 도용 관계에 있다고 단정할 수는 없다.

따라서 소프트웨어 버스마크가 코드 도용을 입증하기 위한 법적인 증거 자료로서 사용될 수 없다는 한계가 있으나 충분히 도용을 의심할 수 있는 근거 자료로 사용될 수 있다는데 의의가 있다[4].

3.2 안드로이드 API 유사도 기반 버스마크



(그림 1) 안드로이드 API 유사도 비교 도구 동작 흐름도

안드로이드 API 유사도 기반 버스마크 도구의 역할은 크게 두 가지로 나눌 수 있다. 첫째, 안드로이드 앱으로부터 API 버스마크를 추출하는 것이고, 둘째, 추출된 API 버스마크를 비교하여 유사도를 구하는 것이다. (그림 1)은 추출 및 비교 과정을 보여주고 있다.

먼저 비교 대상 안드로이드 apk 파일을 압축 해제하여 classes.dex 파일을 추출한다. classes.dex 파일은 원본 자바 코드가 컴파일되어 생성되니 class파일들을 dex 형식으로 변환한 다음 한 개의 파일로 묶은 것이다. classes.dex 파일을 입력으로 dedexer[8] 도구를 실행시키면 달빅(Dalvik) 바이트 코드를 얻을 수 있다. 여기서 dedexer는 dex 파일로부터 달빅 바이트 코드를 얻을 수 있는 디스어셈블러의 일종이다.

추출된 달빅 바이트 코드로부터 invoke 명령을 탐색하

면 API 메소드 호출문을 찾을 수 있다. 이렇게 추출된 API 메소드 호출문으로 버스마크를 생성한다.

생성된 버스마크를 비교하는 방법은 연속된 버스마크를 k 크기의 조각으로 잘라서 비교하는 k-gram 방식[5]을 사용하였다. k-gram 방식을 사용하면 비록 프로그램 전체가 완전히 일치하지 않더라도 k 크기의 조각들이 일치하면 유사도 값을 높일 수 있다. 즉, 부분적인 유사도를 반영하여 전체 유사도를 계산한다.

```
.class public com/flurry/android/Flog
.super java/lang/Object
.source SourceFile
…
.line 7
        invoke-direct          ...
        return-void
.end method
…
l1a480:
        invoke-static          ...
        move-result            ...
        goto        l1a47e
.end method
…
```

(그림 2) 안드로이드 앱의 달빅(Dalvik) 바이트 코드

dedexer 도구를 이용해서 classes.dex 파일로부터 얻은 달빅 바이트 코드는 (그림 2)와 같다. 여기서 invoke로 시작하는 달빅 명령어를 확인하면 API 메소드 호출문의 위치를 쉽게 찾을 수 있다. API 호출 명령어를 추출할 때 호출문에 포함된 API 함수 이름을 함께 추출한다.

```
1: nvoke-direct <init>()V invoke-direct <init>()V invoke-direct <init>(…)V
2: invoke-direct <init>()V invoke-direct <init>(…)V invoke-virtual start()V
3: invoke-direct <init>(…)V invoke-virtual start()V invoke-virtual
4: invoke-static … invoke-direct <init>()V invoke-direct <init>()V
5: invoke-direct <init>()V invoke-direct <init>()V invoke-virtual
6: invoke-direct <init>()V invoke-virtual exists()Z invoke-direct …
```

(그림 3) 달빅 바이트 코드로부터 추출한
API 버스마크 (k=3인 경우)

(그림 3)은 연속된 API 메소드 호출문을 3개 단위로 그룹화하여 생성한 API 버스마크(k=3)를 보여준다.

k 값은 실험 환경이나 대상에 따라서 조절 가능하다. 일반적으로 k값이 작아지면 유사도가 높아지는 경향이 있고, k값이 커지면 유사도가 낮아진다. 단, 유사도가 높아지면 서로 다른 프로그램을 구별할 수 있는 신뢰도는 떨어지는 경우가 많다. 따라서 엄밀하게 두 프로그램의 유사도를 비교하고자 할 때는 k값을 높여서 실험해야 한다.

4. 실험 및 평가

본 논문에서 제안한 안드로이드 API 유사도 비교 도구를 직접 구현하였고, 선행연구[7]와 비교하여 실험하였다. 여기서 선행연구는 안드로이드 앱으로부터 추출된 모든 달빅 명령어를 비교하는 방법이다.

4.1 공통 클래스 탐지 능력 비교 실험

같은 개발사에서 만든 유사 앱은 기본적으로 공통적인 모듈을 함께 사용할 것이기 때문에 공통 클래스를 많이 포함하고 있고 앱을 구성하는 클래스 파일의 유사도가 매우 높을 것이라고 예상할 수 있다.

<표 1> 모든 달빅 명령어에 대한 비교 실험

	제작사 앱 이름 버전 (클래스 수)	Outfit7 톰 1.0.2 (1808)	Outfit7 벤 1.1.1 (1795)	Outfit7 산타 1.1.5 (1422)	Outfit7 기린 1.1.1 (1583)
Outfit7 톰 1.0.2 (1808)	공통 클래스 수	1808	964	755	818
	유사도 평균값	100%	94.15%	94.29%	97.93%
Outfit7 벤 1.1.1 (1795)	공통 클래스 수	-	1795	1010	1092
	유사도 평균값	-	100%	87.74%	81.86%
Outfit7 산타 1.1.5 (1422)	공통 클래스 수	-	-	1422	835
	유사도 평균값	-	-	100%	85.81%
Outfit7 기린 1.1.1 (1583)	공통 클래스 수	-	-	-	1583
	유사도 평균값	-	-	-	100%

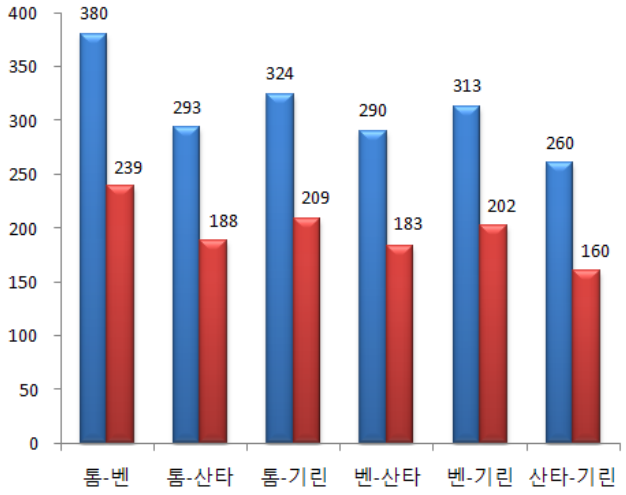
<표 1>의 실험은 모든 달빅 명령어에 대한 실험 결과이다. 같은 개발사의 유사 프로그램이기 때문에 당연히 높은 수치의 유사도 평균값과 공통 클래스 수를 실험 결과를 얻었다.

<표 2> API 메소드 호출 코드에 대한 비교 실험

	제작사 앱 이름 버전 (클래스 수)	Outfit7 톰 1.0.2 (1078)	Outfit7 벤 1.1.1 (1075)	Outfit7 산타 1.1.5 (853)	Outfit7 기린 1.1.1 (926)
Outfit7 톰 1.0.2 (1078)	공통 클래스 수	1078	971	773	835
	유사도 평균값	100%	99.20%	99.04%	99.26%
Outfit7 벤 1.1.1 (1075)	공통 클래스 수	-	1075	910	921
	유사도 평균값	-	100%	99.20%	99.21%
Outfit7 산타 1.1.5 (853)	공통 클래스 수	-	-	853	737
	유사도 평균값	-	-	100%	99.03%
Outfit7 기린 1.1.1 (926)	공통 클래스 수	-	-	-	926
	유사도 평균값	-	-	-	100%

<표 2>의 실험은 <표 1>과 같은 프로그램을 대상으로 API 호출문을 추출하여 유사도를 비교한 실험이다. 비록 API 호출문만을 비교하였음에도 불구하고 유사도 평균값은 <표 1>의 실험보다 높은 결과를 보여주었다. 다만 API 호출문이 많지 않은 클래스에 대해서는 유사도 비교가 불가능하기 때문에 공통 클래스 수는 <표 1>의 실험보다 적은 결과가 나왔다.

4.2 유사도 비교 속도 비교



(그림 3) 모든 달빅 코드 비교 방법(왼쪽)과
API 비교 방법(오른쪽)의 속도 비교(단위:초)

(그림 3)은 모든 달빅 코드를 비교하는 선행 연구와 API 메소드 호출만을 비교하는 본 연구에 대해서 실행 속도를 비교한 결과이다. API 메소드 호출에 대한 유사도 비교 도구는 선행 연구와 비교하여 실행 속도면에서 35%에서 38%정도 빠른 결과를 얻었다.

4.3 API 메소드 호출 유사도 비교 방법의 한계

API 메소드 호출에 대한 유사도 비교 방법은 모든 달빅 명령어를 대상으로 하는 선행 연구와 비교했을 때 대등한 공통 클래스 탐지 능력을 보여 주면서 빠른 실행 속도를 보여주었다. 그러나 API 메소드 호출 유사도 비교 방법도 다음과 같은 한계를 가지고 있다.

첫째, API 호출이 많지 않은 프로그램에 대해서는 적용이 불가능하다. API 호출이 적어도 k 개만큼 포함되어야만 버스마크 추출이 가능하고 k 개보다 적거나 API 호출이 없는 파일에 대해서는 버스마크 추출이 불가능하기 때문이다.

둘째, 실제 프로그램의 실행 순서를 반영하지 못했다. 현재 API 버스마크를 추출할 때 단순히 인접한 API 호출문을 k 크기의 그룹으로 묶어서 사용하고 있다. 즉, 조건문이나 반복문과 같이 제어 흐름이 순차적이지 않은 코드에 대해서도 실제 프로그램 실행 순서를 반영한 API 추출이 필요하다.

5. 결론 및 향후 연구 과제

본 논문에서는 안드로이드 코드 도용을 탐지하기 위한 방법으로서 API 호출 유사도를 측정하는 방법을 제안하였다.

선행 연구에서는 안드로이드 프로그램으로부터 추출된 모든 달빅 코드들을 대상으로 유사도를 비교하기 때문에 시간이 많이 소요되는 문제가 있었다. 본 논문에서 제안한 API 유사도 비교 방법은 선행 연구와 대등한 도용 탐지 성능을 보이면서도 비교 시간을 35% 정도 절약할 수 있음을 확인하였다.

향후 연구 과제는 다음과 같다.

본 논문에서 제안한 안드로이드 API 호출 유사도 비교 도구는 API 호출에 대해서 실행 흐름을 반영하고 있지 못하기 때문에 이에 대한 보완이 필요하다. 즉, 실행 프로그램을 나타내는 제어 흐름 그래프를 생성하고 이 그래프로부터 API 호출문을 추출하는 방법을 고려하고 있다. 그리고 API 호출 빈도가 낮은 파일에 대한 유사도 비교 방법에 대해서도 연구가 필요하다.

참고문헌

[1] "SourceForge.net: Open Source Software," http://www.sourceforge.net.

[2] "Open Source Licenses," http://www.opensource.org/licenses/.

[3] "The gpl-violations.org project," http://www.gpl-violations.org/.

[4] H. Tamada, M. Nakamura, A. Monden, K. Matsumoto, "Java birthmark Detecting the software theft," IEICE Transactions on Information and Systems, E88-D, 9 (Sept. 2005), 2148-2158.

[5] G. Myles and C. Collberg. "k-gram Based Software Birthmarks, " In Proceeding of the 2005 ACM Symposium on Applied Computing, pp. 314-318. Santa Fe, New Mexico, USA, 2005.

[6] "Bytecode for the Dalvik VM," http://www.netmite.com/android/mydroid/dalvik/docs/dalvik-bytecode.html

[7] 최성하, 박세익, 박희광, 박희완 "안드로이드 앱 도용 탐지를 위한 유사도 비교 도구 구현", 2011년도 한국정보보호학회 동계학술발표대회 논문집

[8] "dedexer, disassembler tool for DEX files," http://dedexer.sourceforge.net.