안드로이드 앱 도용 탐지를 위한 유사도 비교 연구

박세익, 박희광, 최성하, 박희완
한라대학교 정보통신방송공학부
seiks@naver.com, blue300@nate.com, shoutme1991@nate.com,
heewanpark@halla.ac.kr

A Study on Similarity Comparison for Detecting Theft of Android Application

Seik Park, Heekwang Park, Sungha Choi and Heewan Park
School of Information, Communication and Broadcasting Engineering,
Halla University

요 약

소프트웨어 버스마크는 모든 프로그램에 이미 포함되어 있으며 서로 다른 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 소프트웨어 버스마크를 이용하여 안드로이드 앱 사이의 유사도를 측정하고 코드 도용 탐지에 활용하는 방법을 제안하였고, 다양한 카테고리의 안드로이드 앱에 대한 유사도 비교 실험을 하였다.

먼저, 같은 개발사에서 만든 유사한 프로그램을 대상으로 버스마크 유사도를 측정한 결과 유사도가 매우 높다는 것을 확인하였다. 또한, 서로 다른 개발사에서 만든 유사한 카테고리의 프로그램을 비교하였고 비슷한 프로그램이라도 서로 다른 개발사에서 만든 앱이기 때문에 유사도가 낮다는 것을 확인하였다. 마지막으로, 서로 다른 개발사의 유사한 프로그램들 중에서 유사도가 높게 측정된 경우를 탐지한 실험 결과를 제시하였다. 이러한 유사 앱들은 실제로 공통 클래스를 함께 포함하고 있었다.

실험 결과들을 바탕으로 소프트웨어 버스마크가 안드로이드 앱 사이의 공통 클래스를 탐지하는데 활용될 수 있음을 확인하였고, 더 나아가 안드로이드 앱에서의 코드 도용을 탐지하는 목적으로도 활용될 수 있음을 보여주었다.

1. 서론

오늘날 세계적으로 수많은 소프트웨어들이 시시각각 작성되어 배포되고 있다. 소프트웨어는 각 기업들이 기울인 노력의 산물이기 때문에 소프트웨어 개발에 대한 노하우와 프로그램의 원본 소스 코드가 타 업체에 의해서 도용당하지 않도록 보호하는 것도 매우 중요하다. 만일 다른 업체가 자사의 소프트웨어를 도용한 사실을 발견했다면 정당한 법적 소송을 통해서 해당 소프트웨어의 권리를 주장해야 한다.

일반적으로 소프트웨어는 바이너리 상태로 배포되기 때문에 역공학이 쉽지 않다. 따라서 출시된 소프트웨어의 바이너리로부터 핵심 알고리즘을 추출하거나 소스를 얻어내는 것은 불가능에 가깝다. 그러나 자바 언어로 작성된 프로그램의 경우는 상황이 다르다. 자바 프로그램은 이식성이 높고, 엄격한 자바 가상 머신의 명세 때문에 다양한 디컴파일러까지 개발된 상태이다[1,2]. 따라서 소스가 공개되지 않은 자바 프로그램의 경우에도 대부분 디컴파일하여 원본 소스를 얻어낸 후 부분적으로 수정하여 도용하는 것이 가능하다. 이와 같이 역공학 도구만 사용한다면 누구나 쉽게 자바 프로그램을 도용할 수 있는 상황이다.

특히 안드로이드 앱은 자바를 기반으로 하기 때문에 자바 프로그램과 마찬가지로 쉽게 디컴파일러를 이용하여 원본 소스 코드를 얻어낼 수 있어서 코드 도용에 대해서 매우 취약하다.

소프트웨어 버스마크는 모든 프로그램에 이미 포함되어 있으면서 서로 다른 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 소프트웨어 버스마크를 이용하여 안드로이드 앱 사이의 버스마크 유사도를 측정하고 코드 도용 탐지에 활용하는 방법을 제안한다.

2. 관련 연구

자바 클래스를 대상으로 하는 정적 버스마크는 Tamada에 의해서 제안되었다[3]. 이 방법은 자바 클래스의 필드 변수에 사용된 상수값, 메소드 호출이 발생한 순서 정보, 클래스 상속에 사용된 클래스 정보, 프로그램에서 실제로 사용된 클래스 정보를 추출하여 특정 자바 클래스를 대표하는 버스마크로 사용한다. 그러나 이러한 특징들이 유사하더라도 클래스에서 사용된 알고리즘이 다른 경우에는 서로 다른 클래스를 구별하는 변별력이 떨어지는 단점이 있다.

Myles는 k-gram 버스마크[4]를 제안하였다. k-gram

버스마크는 자바 클래스에 포함된 바이트코드 명령어를 추출하여 연속된 k개의 시퀀스 집합을 버스마크로 사용한다. 이 방법은 실제로 구현에 사용된 바이트코드를 버스마크로 사용하기 때문에 서로 다른 프로그램을 구별하는 능력은 뛰어나다. 그러나 프로그램의 흐름을 분석하지 않아서 조건문에서 참과 거짓 조건의 순서를 바꾸는 간단한 변환에 대해서도 버스마크가 손상되는 단점이 있다.

대표적인 동적 버스마크에는 Myles가 고안한 WPP 버스마크[5]가 있다. WPP는 전체 프로그램 경로(whole program path)를 의미한다. 이 방법은 프로그램을 실행시키면서 호출된 명령어를 저장하고 이것을 이용하여 그래프 형태의 버스마크를 생성한다. 프로그램이 변환되거나 난독화가 되어도 실제로 실행되는 명령어가 동일하다는 장점이 있지만, 실행 시간에 저장해야 하는 정보의 양이 많다는 단점도 있다.

3. 소프트웨어 버스마크를 이용한 안드로이드 앱 유사도 비교

3.1 소프트웨어 버스마크

소프트웨어 버스마크는 프로그램을 인식하거나 확인하는데 사용될 수 있는 고유한 특징이다. Tamada[3]와 Myles[4]는 다음과 같이 소프트웨어 버스마크를 정의했다.

정의 1 (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 copy 관계에 있다면 f(p) = f(q) 이다.

다음 두 가지 속성은 버스마크가 만족시켜야 하는 평가 기준을 나타낸다.

속성 1 (신뢰성 Credibility)

같은 기능을 하는 두 프로그램 p와 q에 대해서, p와 q가 독립적으로 개발되었을 때, f(p)≠ f(q)을 만족해야 한다.

속성 2 (강인성 Resilience)

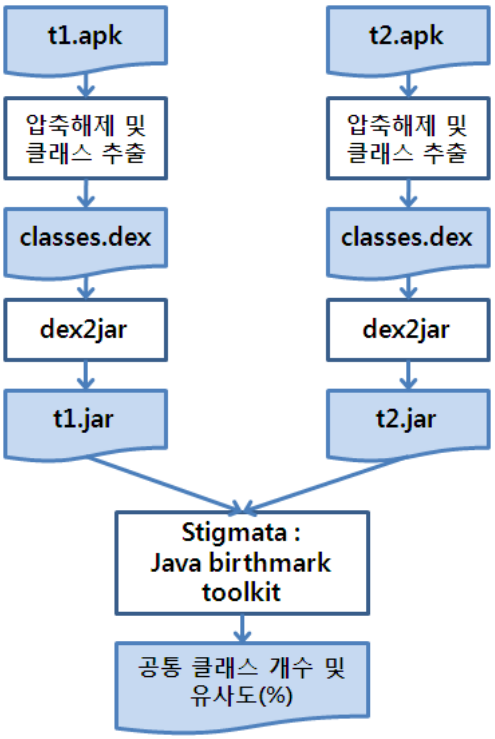
프로그램 p′이 프로그램 p로부터 변환되었다고 할 때, f(p) = f(p′)을 만족해야 한다.

속성 1은 버스마크의 신뢰성을 의미한다. 즉, 두 모듈이 비록 동일한 기능을 수행하더라도 소스 코드를 공유하지 않고 독립적으로 개발된 모듈이라면, 두 모듈로부터 추출된 버스마크는 같지 않아야 한다.

속성 2는 버스마크의 강인성을 의미한다. 즉, 어떤 프로그램이 프로그램 최적화 기법이나 난독화 기법과 같이 프로그램의 의미를 바꾸지 않는 다양한 변환을 통해서 변환되더라도 변환 전에 추출된 버스마크와 변환 후에 추출된 버스마크가 서로 같아야 한다.

3.2 안드로이드 앱 유사도 측정 방법

본 논문에서는 두 안드로이드 앱 사이의 유사도를 구하기 위해서 공개된 자바 버스마크 툴킷인 Stigmata[6]를 이용하였다. Stigmata는 두 개의 jar 파일을 입력으로 받아들이는 도구이기 때문에 안드로이드 앱에 곧바로 적용할 수는 없고 아래와 같은 일련의 변환 과정을 거쳐야 한다.



(그림 1) 안드로이드 앱 유사도 측정 절차

먼저 apk 포맷으로 압축된 안드로이드 앱의 압축을 풀고 classes.dex 파일을 추출한다. 그리고 dex2jar 도구[7]를 사용하여 dex 포맷 파일을 jar 파일로 변환한다. 이런 과정으로 생성된 두 개의 jar 파일을 stigmata 버스마크 툴킷에 입력으로 넣어서 결과적으로 공통 클래스와 유사도 값을 계산하였다.

4. 실험 및 평가

본 논문에서 제안하는 소프트웨어 버스마크 기반의 안드로이드 앱 유사도 비교 방법의 효용성을 검증하기 위해서 Stigmata를 이용한 비교 실험을 하였다. Stigmata 버스마크 툴킷은 다양한 종류의 버스마크와 옵션을 제공한다. 그 중에서 대표적인 정적 버스마크 기법인 k-gram 버스마크를 선택하였고, k는 기본 설정 값인 4로 정하였다. 그림 2는 Stigmata의 대한 실행 화면이다.

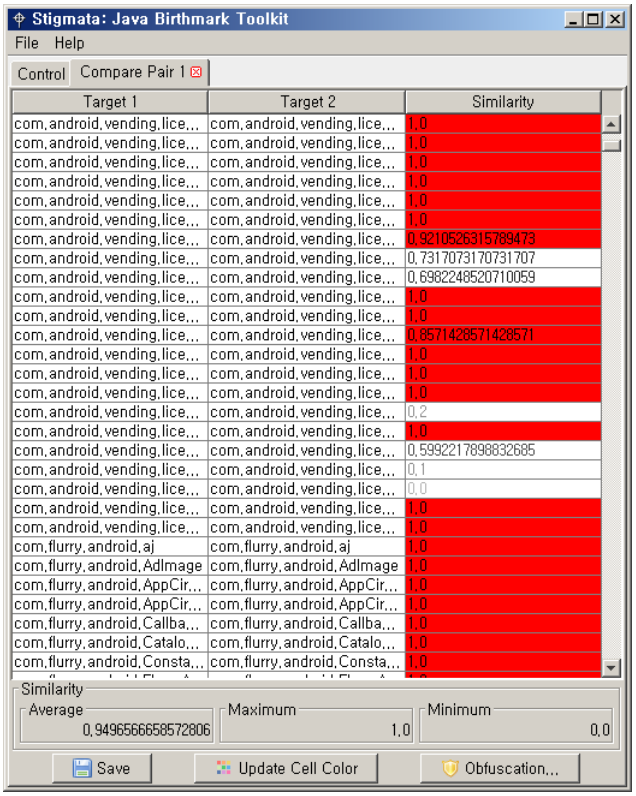


그림 2. Stigmata 버스마크 툴킷 실행화면

4.1 같은 개발사의 유사 앱에 대한 실험

같은 개발사에서 만든 유사 앱은 기본적으로 공통적인 모듈을 함께 사용할 것이기 때문에 공통 클래스의 개수와 클래스 간 유사도가 매우 높을 것이라고 예상할 수 있다.

<표 1> 같은 개발사의 유사 프로그램에 대한 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	1230	982	1070
	유사도 평균값	100%	94.9%	95.4%	97.0%
Outfit7 벤 1.1.1 (1795)	공통 클래스 수	-	1795	1344	1445
	유사도 평균값	-	100%	86.0%	82.1%
Outfit7 산타 1.1.5 (1422)	공통 클래스 수	-	-	1422	1139
	유사도 평균값	-	-	100%	84.6%
Outfit7 기린 1.1.1 (1583)	공통 클래스 수	-	-	-	1583
	유사도 평균값	-	-	-	100%

표 1의 실험에 사용된 안드로이드 앱은 Outfit7사에서 제작한 사람의 말을 따라하는 앱 시리즈이다. 실험에 사용된 Outfit7사 앱들은 서로 유사한 기능을 수행하기 때문에 공통 클래스가 많이 포함되어 있었고 유사도 또한 매우 높은 것을 확인할 수 있었다.

<표 2> 같은 개발사의 유사 프로그램에 대한 실험

	제작사 앱 이름 버전 (클래스 수)	Rovio Angry Birds 1.2.05 (155)	Rovio Angry Birds 1.4.2 (153)	Rovio Angry Birds 1.5.1 (199)	Rovio Angry Birds Rio 1.0 (382)
Rovio Angry Birds 1.2.05 (155)	공통 클래스 수	155	147	150	135
	유사도 평균값	100%	98.4%	100%	88.2%
Rovio Angry Birds 1.4.2 (153)	공통 클래스 수	-	153	152	132
	유사도 평균값	-	100%	98.5%	86.4%
Rovio Angry Birds 1.5.1 (199)	공통 클래스 수	-	-	199	136
	유사도 평균값	-	-	100%	88.0%
Rovio Angry Birds Rio 1.0 (382)	공통 클래스 수	-	-	-	382
	유사도 평균값	-	-	-	100%

표 2의 실험에서는 Rovio사에서 제작한 Angry Birds 시리즈를 버전별로 유사도 측정을 하였다. 동일한 게임의 여러 가지 버전을 비교하는 경우에는 당연히 높은 유사도가 나와야 된다는 가정 하에 실험하였다.

그 결과, 비록 연속된 버전이 아니었더라도 서로 높은 유사도를 나타내었으며, Rio버전의 Angry Birds까지도 다른 버전과 높은 유사도를 나타내는 것을 확인하였다.

4.2 다른 개발사의 유사 앱에 대한 실험

다른 개발사에서 만든 유사 앱들은 비록 유사한 기능을 수행하더라도 서로 코드를 공유하지만 않고 독자적으로 개발하였다면 공통 클래스가 존재하지 않을 것이고 앱 사이의 유사도가 낮을 것으로 예상할 수 있다.

<표 3> 다른 개발사의 유사 프로그램에 대한 실험

	제작사 앱 이름 버전 (클래스 수)	Outfit7 톰 1.0.2 (1808)	SoMabh 앵무새 2.0 (169)	Mr.Moo 토킹패럿 1.0.3 (26)
Outfit7 톰 1.0.2 (1808)	공통 클래스 수	1808	0	7
	유사도 평균값	100%	0%	6.7%
SoMabh 앵무새 2.0 (169)	공통 클래스 수	-	169	0
	유사도 평균값	-	100%	0%
Mr.Moo 토킹패럿 1.0.3 (26)	공통 클래스 수	-	-	26
	유사도 평균값	-	-	100%

표 3의 실험에서는 다른 개발사에서 만든 비슷한 기능의 앱을 비교했다. 그 결과 다른 회사, 다른 개발자가 제작하였다는 것을 증명하듯이 유사도가 매우 낮게 나왔다.

<표 4> 다른 개발사의 유사 프로그램에 대한 실험

	제작자 앱 이름 버전 (클래스 수)	FineApps 대구버스 1.9.6 (133)	구미시청 구미버스 1.0 (78)	김민수 부산버스 1.15 (116)
FineApps 대구버스 1.9.6 (133)	공통 클래스 수	133	0	0
	유사도 평균값	100%	0%	0%
구미시청 구미버스 1.0 (78)	공통 클래스 수	-	78	0
	유사도 평균값	-	100%	0%
김민수 부산버스 1.15 (116)	공통 클래스 수	-	-	116
	유사도 평균값	-	-	100%

표 4의 실험에서도 표 3의 실험과 같이 서로 비슷한 기능을 가진 앱이더라도 제작자가 다를 경우 유사도가 매우 낮은 것을 확인할 수 있었다.

<표 5> 다른 개발사의 유사 프로그램에 대한 실험

	제작자 앱 이름 버전 (클래스 수)	박영훈 서울버스 1.2.4 (374)	FineApps 대구버스 1.9.6 (133)	김민수 부산버스 1.15 (116)	김덕환 울산버스 1.4 (368)
박영훈 서울버스 1.2.4 (374)	공통 클래스 수	374	26	0	0
	유사도 평균값	100%	**98.7%**	0%	0%
FineApps 대구버스 1.9.6 (133)	공통 클래스 수	-	133	0	0
	유사도 평균값	-	100%	0%	0%
김민수 부산버스 1.15 (116)	공통 클래스 수	-	-	116	56
	유사도 평균값	-	-	100%	**96.8%**
김덕환 울산버스 1.4 (368)	공통 클래스 수	-	-	-	368
	유사도 평균값	-	-	-	100%

그러나 표 5에서는 지금까지의 실험을 통해서 입증된 '비슷한 종류의 앱이라도 다른 개발자가 만든 앱은 유사도가 높을 수 없다.' 라는 가정과는 달리, 공통 클래스가 다수 발견되었고 유사도도 높은 결과를 얻었다.

그 이유는 유사도가 높은 앱들이 공통 클래스를 함께 포함하고 있었기 때문이었다. 공통 클래스를 직접 확인해 본 결과 모바일 광고 관련 클래스로 밝혀졌다. 실제로 서울버스와 대구버스 앱은 Daum[8]에서 제공하는 모바일 광고 모듈을 함께 사용하고 있었다. 그리고 부산버스와 울산버스는 Cauly[9]에서 제공하는 광고 모듈을 함께 사용하고 있었음을 확인하였다.

위 실험을 통해서 확인한 것처럼, 서로 다른 두 앱이 만일 공통 모듈을 포함하고 있다면 본 논문에서 제안하는 유사도 측정 방법을 사용하여 공통 모듈을 탐지할 수 있다. 따라서 이 방법은 안드로이드 앱에 대한 코드 도용이 발생하였을 때에도 도용 사실을 탐지할 수 있는 도구가 될 수 있을 것으로 기대한다.

5. 결론 및 향후 과제

본 논문에서는 코드 도용에 대처하는 방법으로서, 안드로이드 앱 사이의 유사도를 측정하는 방법을 제안하였고, 유사도 측정 결과를 평가하기 위해서 다양한 카테고리의 안드로이드 앱에 대한 실험을 하였다.

먼저, 같은 개발사에서 만든 유사한 앱을 대상으로 버스마크를 통한 유사도 측정 결과 유사도가 매우 높다는 것을 확인하였다. 또한, 서로 다른 개발사에서 만든 유사한 카테고리의 앱을 비교했을 경우에는 비록 비슷한 프로그램이더라도 서로 다른 개발사에서 만든 앱이기 때문에 유사도가 매우 낮은 것을 확인하였다. 마지막으로, 서로 다른 개발사의 유사한 앱들 중에서 유사도가 예상과 달리 매우 높게 측정된 경우를 발견하였다. 이러한 앱들은 실제로 광고 관련된 모듈을 함께 포함하고 있었음을 확인했다.

이러한 실험 결과들을 바탕으로 소프트웨어 버스마크가 안드로이드 앱 사이의 공통 클래스를 탐지하는데 유용하게 활용될 수 있음을 확인하였고, 더 나아가 안드로이드 앱에서의 코드 도용을 탐지하는 목적으로도 활용될 수 있음을 보여주었다.

본 논문의 향후 연구 과제는 다음과 같다.

현재 Stigmata 도구는 클래스 이름이 같은 경우에 공통 클래스로 인식하고 유사도 비교를 수행하지만, 클래스 이름이 다르더라도 실제 프로그램 내용이 동일하다면 이것을 탐지할 수 있는 방법이 필요하다. 이것을 위해서 현재 자체 도구 개발을 계획하고 있다.

또한, 아이폰 앱이나 윈도우폰 7 앱에 대해서도 유사도 측정 방법을 제시하고, 비교 도구 개발을 계획하고 있다.

참고문헌

[1] "Jad-the fast Java Decompiler," http://www.kpdus.com/jad.html.

[2] "Mocha, the Java Decompiler," http://www.brouhaha.com/~eric/software/mocha/.

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

[4] Ginger Myles and Christian 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.

[5] Ginger Myles and Christian Collberg. Detecting Software Theft via Whole Program Path Birthmarks. ISC 2004, LNCS 3225, pp. 404-415, 2004.

[6] "stigmata - Java birthmark toolkit," http://stigmata.sourceforge.jp/.

[7] "dex2jar - A tool for converting Android's .dex format to Java's .class format," http://code.google.com/p/dex2jar/.

[8] 다음 모바일 광고, http://mobile.biz.daum.net/.

[9] 카울리 모바일 광고, http://www.cauly.net/.