pISSN: 2287-5891

http://dx.doi.org/10.3745/KTCCS.2013.2.4.177

An Android Birthmark based on API k-gram

Heewan Park†

ABSTRACT

A software birthmark means inherent characteristics that can be used to identify a program. Because the software birthmark is difficult to remove by simple program transformation, it can be used to detect code theft. In this paper, we propose a birthmark technique based on API k-gram of Android applications. Android SDK provides various libraries that help programmers to develop application easily. In order to use Android SDK, we have to use API method calls. The API call instructions are hard to be replaced or removed, so they can be a inherent characteristics of an application. To show the effectiveness of the proposed birthmark, we compared it with previous birthmarks and evaluated it with open source applications. From the experiments, we verified that the credibility and resilience of our birthmark is higher than previous birthmarks.

Keywords : Android Application, Software Birthmark, Dalvik Bytecodes, Code Theft Detection

API k-gram 기반의 안드로이드 버스마크

박 희 완†

요 약

소프트웨어 버스마크는 프로그램을 인식하는데 사용될 수 있는 고유한 특징을 의미한다. 소프트웨어 버스마크는 단순한 프로그램 변환에 의해서 삭제되지 않기 때문에 코드 도용을 탐지하는데 사용된다. 본 논문에서는 안드로이드 앱에 대한 API k-gram 기반의 버스마크 기법을 제안한다. 안드로이드 SDK는 프로그래머가 쉽게 앱을 개발할 수 있도록 다양한 라이브러리를 제공한다. 그리고 안드로이드 SDK를 사용하기 위해서는 반드시 API 메소드 호출을 이용해야만 한다. API 메소드 호출 명령어는 다른 명령어로 바꾸거나 삭제하기 어렵기 때문에 애플리케이션의 고유한 특징으로서 사용될 수 있다. 본 논문에서 제안하는 버스마크의 효용성을 보여주기 위해서 기존의 버스마크 기법과 비교하였고 오픈소스 앱을 대상으로 평가하였다. 실험으로부터 API k-gram 버스마크가 기존 버스마크보다 신뢰도와 강인도가 높은 것을 확인하였다.

키워드 : 안드로이드 애플리케이션, 소프트웨어 버스마크, 달빅 바이트코드, 코드 도용 탐지

1. 서 론

안드로이드는 역공학 분석이 쉬운 바이트코드를 사용하기 때문에 코드 도용에 취약하다. 즉, 바이트코드 디컴파일러[1, 2]를 비롯한 역공학 도구를 사용하면 대부분 앱 바이너리로부터 자바 소스 코드를 얻을 수 있다. 안드로이드 앱의 소스 코드를 역공학에 의해서 쉽게 얻을 수 있다면 인기 있는 앱의 소스 코드를 도용해서 유사 앱을 만들 수도 있다. 이러한 위험을 예방하기 위해서 코드 난독화[3]를 적용할 수 있지만 난독화는 코드 분석을 어렵게 만들 수는 있어도 소스 코드 유출을 막을 수 있는 근본적인 대책이 되지는 못한다. 소프트웨어 버스마크[4-7]는 소프트웨어로부터 고유한 특징을 추출하여 유사도를 구하는 기법이다. 소프트웨어 버스마크는 바이너리 파일 자체를 비교하여 코드 도용 여부를 판단하거나 오픈소스 라이선스 위반을 탐지하는데 활용될 수 있다.

본 논문에서는 안드로이드 앱의 코드 도용 여부를 탐지하기 위한 *API k-gram* 기반의 버스마크 기법을 제안한다. *API k-gram*이란 API 메소드 호출 명령어를 k 크기의 조각을 자른 집합을 의미한다. 안드로이드 API 메소드는 안드로이드 앱을 개발하기 위해서 반드시 사용해야 하기 때문에 다른 것으로 대체시키거나 제거하기 어렵다는 특징이 있다. 이 사실을 근거로 하여 안드로이드 API 메소드 호출 명령어를 프로그램의 고유한 특징으로 규정하였다. 본 논문에서 제안하는 버스마크의 효용성을 보여주기 위해서 오픈소스 앱을 대상으로 기존 버스마크와 비교 실험을 하였다. 실험 결과로부터 API k-gram 버스마크가 기존 버스마크보다 신뢰도와 강인도가 높은 것을 확인하였다.

† 정 회 원 : 한라대학교 정보통신방송공학부 조교수
논문접수 : 2012년 11월 27일
심사완료 : 2013년 1월 17일
* Corresponding Author : Heewan Park(heewanpark@halla.ac.kr)

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 살펴보고, 3장에서 본 논문에서 제안하는 API k-gram 버스마크를 소개하며, 4장에서 구현 내용을 설명하고, 5장에서 실험 및 평가를 하고, 6장에서 결론과 향후 연구 과제에 대해서 논의한다.

2. 관련 연구

자바를 대상으로 하는 버스마크 중에서 가장 잘 알려진 기법은 *Tamada* 버스마크[4]와 *k-gram* 버스마크[5]이다. *Tamada* 버스마크는 자바 클래스의 필드 변수에 사용된 상수값, 메소드 호출 순서, 클래스 상속 구조, 사용된 클래스 정보를 버스마크로 사용한다. *k-gram* 버스마크는 연속된 자바 바이트코드 명령어를 *k* 크기의 조각으로 잘라서 버스마크로 사용한다. *Tamada* 버스마크는 자바 클래스의 구조적인 특징을 4가지 요소를 통해서 비교하기 때문에 서로 다른 클래스를 구별하는 신뢰도가 높다고 평가되며, *k-gram* 버스마크는 작게 조각난 명령어의 시퀀스를 비교하기 때문에 난독화에 대한 강인도가 높다고 평가된다.

최근 안드로이드 달빅(Dalvik) 코드와 API 유사도를 이용하는 버스마크 기법[6, 7]이 제안되었지만 신뢰도와 강인도 대한 평가가 고려되지 않았으며, 기존의 *Tamada, k-gram* 버스마크 기법과의 비교도 이루어지지 않았다. 본 논문은 기존 API 유사도 비교 버스마크를 확장하여 *API k-gram* 버스마크를 제안하고, 신뢰도와 강인도를 척도로 버스마크의 성능을 평가하며 기존의 *Tamada, k-gram* 버스마크와의 비교를 수행하고자 한다.

3. API k-gram 안드로이드 버스마크

3.1 API k-gram

안드로이드 앱은 클래스 파일의 집합으로 이루어지고 안드로이드 달빅 가상머신 명령어의 호출 시퀀스로 표현할 수 있다. *API k-gram*은 달빅 가상머신 명령어 중에서 자바 또는 안드로이드 API 메소드 호출 명령어만을 추출하여 생성한다. 여기서 *k-gram*은 크기가 *k*인 조각을 의미한다. 즉, API 메소드 호출 명령어 시퀀스를 *k* 크기만큼씩 잘라낸 조각들을 의미한다. 따라서 임의의 클래스 *C*에 대해서 *API k-gram(C)*는 다음과 같이 정의된다.

Definition 1. *API k-gram(C)*

$$API\ k\text{-}gram(C) = \{\langle API_i, API_{i+1}, \ldots API_{i+k-1}\rangle \mid 1 \le i \le n-k+1\}.$$

```
1: invoke-direct android/content/ContentValues/<init>()V
2: invoke-static java/lang/System/currentTimeMillis()J
3: invoke-static java/lang/Long/valueOf(J)Ljava/lang/Long;
4: invoke-virtual java/lang/String/length()I
5: invoke-virtual android/content/Intent/getData()Landroid/
   net/Uri;
```

Fig. 1. An example of API method call

예를 들어, 어떤 클래스가 Fig. 1과 같은 API 메소드 호출문을 포함하고 있다면 이 클래스로부터 추출된 *API 3-gram*은 Fig. 2와 같이 3개의 원소를 갖는 집합이 된다.

```
1: invoke-direct android/content/ContentValues/<init>()V
2: invoke-static java/lang/System/currentTimeMillis()J
3: invoke-static java/lang/Long/valueOf(J)Ljava/lang/Long;

2: invoke-static java/lang/System/currentTimeMillis()J
3: invoke-static java/lang/Long/valueOf(J)Ljava/lang/Long;
4: invoke-virtual java/lang/String/length()I

3: invoke-static java/lang/Long/valueOf(J)Ljava/lang/Long;
4: invoke-virtual java/lang/String/length()I
5: invoke-virtual android/content/Intent/getData()Landroid/
   net/Uri;
```

Fig. 2. An example of API 3-gram of Fig. 1.

3.2 API k-gram 생성

달빅 코드로부터 *API k-gram*을 생성하기 위해서는 달빅 명령어를 파싱하여 invoke로 시작하는 달빅 명령어를 먼저 구분하고, 사용자 정의 메소드 호출인지, 시스템 메소드 호출인지를 판별해야 한다. 본 논문에서는 invoke 명령문에서 *java*와 *android*로 시작하는 API 호출문을 추출하였다. 그리고 메소드 이름만 추출하지 않고 메소드 파라미터와 리턴값도 함께 추출하였는데, 이것은 메소드 이름은 동일하지만 오버로딩(overloading)된 메소드를 구분하기 위해서 사용될 수 있기 때문이다.

3.3 API k-gram 유사도 계산

임의의 두 클래스 *A*와 *B*로부터 추출된 *API k-gram*을 각각 *API k-gram(A)*와 *API k-gram(B)* 라고 할 때, 두 클래스의 유사도 *Similarity(A,B)*는 다음과 같이 구한다.

Definition 2. *Similarity(A,B)*

$$Similarity(A,B) = \frac{|API\ k\text{-}gram(A)| \cap |API\ k\text{-}gram(B)|)}{min(|API\ k\text{-}gram(A)|, |API\ k\text{-}gram(B)|)}$$

여기서 $|API\ k\text{-}gram(A)|$는 *API k-gram(A)*의 원소의 개수를 의미한다. 즉, 클래스 *A*와 *B*에서 공통된 *API k-gram*의 비율이 유사도로 표현된다. 여기서 분모에 *min* 함수를 이용한 이유는 기존의 클래스를 복제한 후에 API 메소드 호출문을 추가했을 경우에도 유사도를 높게 얻을 수 있기 때문이다. 즉, 복제 사실을 숨기기 위해서 단순하게 API 메소드 호출문을 추가한다고 해도 공통된 API 호출이 남아 있기 때문에 유사도는 낮아지지 않는다.

4. 구 현

Fig. 3은 *API k-gram* 버스마크 시스템 구조도이다. 버스마크 시스템의 입력은 두 개의 안드로이드 앱이다. 안드로

이드 앱은 apk 확장자로 압축된 형태이고, 달빅 코드로 이루어진 클래스 파일, XML 파일, 리소스 파일로 구성된다. 버스마크 시스템은 입력으로 들어온 apk 파일의 압축을 풀어서 classes.dex 파일을 추출하고, dex 디스어셈블러를 사용해서 달빅 코드 파일들을 추출한다. 본 논문에서는 달빅 코드를 추출하기 위한 전단부 도구로서 *dedexer* 디스어셈블러[8]를 사용했다.

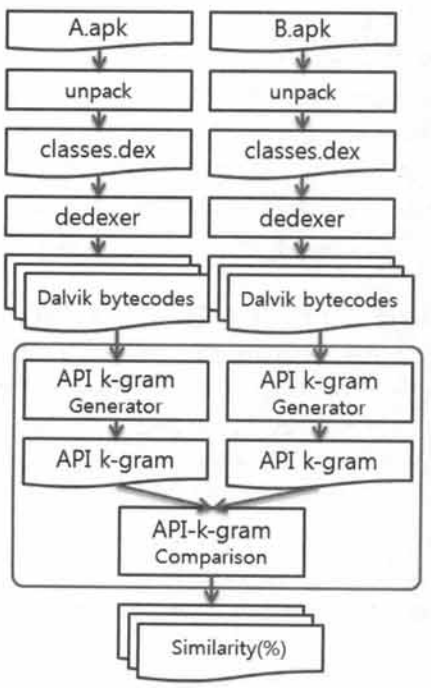

Fig. 3. API k-gram birthmark system

버스마크 시스템의 핵심 모듈은 버스마크 생성 도구와 비교 도구이다. 버스마크 생성 모듈은 3.2절에서 언급된 것과 같이 달빅 코드를 파싱하여 *invoke* 명령어를 찾아내고 메소드 이름과 메소드 파라미터 및 리턴값을 추출하여 *API k-gram*을 만드는 과정을 구현하였다. 버스마크 비교 모듈은 3.3절에서 언급된 것과 같이 생성된 *API k-gram*을 유사도 공식으로 계산하여 달빅 파일마다 유사도를 계산하고 리포트 파일을 생성하도록 구현하였다. 만일 패키지 A와 B를 비교한다고 가정하고, A로부터 N개의 달빅 파일이 추출되고, B로부터 M개의 달빅 파일이 추출되었다면, $N*M$번의 비교를 수행하여 결과를 출력한다.

5. 실험 및 평가

실험 대상 프로그램으로는 안드로이드 SDK에 포함되어 배포되는 *BluetoothChat, LunarLander, NotePad, Snake, WhetherListWidget* 5개의 앱을 선정하였다. 그리고 기존 버스마크인 *Tamada, k-gram* 버스마크, 본 논문에서 제안하는 *API k-gram* 버스마크에 대해서 신뢰도와 강인도 실험을 수행하였다. *Tamada, k-gram* 버스마크에 대한 실험은 공개된 자바 버스마크 툴킷인 *stigmata*[9]를 사용하였다. 그러나 *stigmata*는 자바를 대상으로 만들어진 도구이기 때문에 그대로 안드로이드에 적용하는 것은 불가능하다. 그래서 *dex2jar* 도구[10]를 사용하여 안드로이드 달빅 바이트코드를 자바 바이트코드로 변환한 후 *stigmata*를 이용해서 실험하였다. *k-gram*과 *API k-gram*에서의 조각의 크기를 의미하는 k값은 기존 논문[5-7]에서 공통으로 사용하고 있는 3을 선택하였다.

본 실험에서는 각 앱에 포함된 클래스 중에서 적어도 3개 이상의 API 호출문을 포함하고 있는 클래스를 선정하였다. 모든 클래스에 대해서 실험을 하면 크기가 매우 작은 클래스까지도 비교하게 되는데 이렇게 작은 클래스는 도용 가치가 적을 뿐 아니라 도용을 탐지하더라도 관용적으로 사용되는 루틴일 가능성이 많기 때문이다.

Table 1. The credibility evaluation result of different classes

유사도 범위	Tamada	k-gram	API k-gram
[0.0, 0.2)	161	46	351
[0.2, 0.4)	251	185	99
[0.4, 0.6)	44	221	14
[0.6, 0.8)	8	13	1
[0.8, 1.0)	0	0	0
1.0	1	0	0
비교 조합수 ${}_{31}C_2$=31*30/2	465	465	465
평균	0.26	0.38	0.14
표준편차	0.12	0.13	0.10

Table 1은 5개의 앱으로부터 추출된 31개의 클래스를 대상으로 버스마크의 신뢰도 비교 실험 결과이다. 서로 다른 클래스를 비교하기 때문에 ${}_{31}C_2$의 비교를 수행하며, 신뢰도의 평균값과 표준 편차는 0에 가까울수록 이상적인 버스마크이다. 3가지 버스마크의 신뢰도 평균값과 표준편차를 비교해보면 *API k-gram* 버스마크가 가장 작은 평균값인 0.14를 기록하였고, 표준편차도 0.10으로 가장 작아서 신뢰도가 가장 높은 버스마크임을 확인하였다. *API k-gram* 버스마크가 기존 *k-gram* 버스마크와 비교했을 때 신뢰도가 높은 이유는 다음과 같다. 기존 *k-gram*의 경우는 바이트코드로부터 모든 달빅 명령어만을 추출하고 파라미터와 리턴값은 제외시키기 때문에 모든 API 호출문이 *invoke*로 통일된다. 따라서 서로 다른 API를 호출하더라도 동일한 명령으로 인식하기 때문에 다른 앱을 구별할 있는 신뢰도가 낮다.

Table 2는 5개의 앱으로부터 추출된 31개의 원본 클래스와 난독화 도구 프로가드[3]를 적용한 31개의 난독화 클래스를 비교한 강인도 평가 결과이다. 강인도는 원본 파일을 난독화하더라도 버스마크가 제거되지 않고 살아남는 정도를 의미하기 때문에 강인도의 평균값은 1에 가까울수록 우수한 버스마크이다. *API k-gram*은 0.99로 가장 높은 평균값을 기록하였다. 기존 *k-gram*도 0.98로 평균값이 높지만 *API*

*k-gram*의 표준편차가 0.03으로 가장 작기 때문에 데이터가 흩어지지 않고 집중되어 있다는 것을 확인할 수 있다. *API k-gram* 버스마크가 기존 버스마크와 비교했을 때 강인도가 높은 이유는 다음과 같다. 난독화를 거치더라도 기존의 API 호출문이 다른 명령어로 바뀌지 않고 반드시 살아남기 때문이다. 그러나 난독화에 의해서 원본에 존재하지 않던 API 호출문이 추가되는 경우가 있기 때문에 API 호출문의 개수가 증가하는 현상은 발생할 수 있다.

Table 2. The resilience evaluation result of same classes after Proguard obfuscation

유사도 범위	Tamada	k-gram	API k-gram
[0.0, 0.2)	0	0	0
[0.2, 0.4)	0	0	0
[0.4, 0.6)	3	0	0
[0.6, 0.8)	12	1	0
[0.8, 1.0)	5	13	5
1.0	11	17	26
합계	31	31	31
평균	0.83	0.98	0.99
표준편차	0.16	0.05	0.03

6. 결론 및 향후 연구 과제

본 논문에서는 안드로이드 달빅 코드에 대해서 *API k-gram* 기반의 버스마크 기법을 제안하였고, 기존 버스마크와 신뢰도 및 강인도 비교 실험을 수행하였다. 실험 결과로부터 *API k-gram* 버스마크가 기존 버스마크보다 신뢰도와 강인도가 모두 높으며 표준편차가 낮아서 성능이 우수한 버스마크임을 확인하였다. 기존의 버스마크 도구는 자바 대상으로 하기 때문에 안드로이드 앱으로 실험하기 위해서는 달빅 바이트코드를 자바 바이트코드로 변환하는 과정을 거쳐야 했다. 그러나 API k-gram 버스마크는 안드로이드 달빅 바이트코드 자체를 비교하기 때문에 변환 과정이 필요 없다는 장점도 있다.

향후 연구 과제는 다음과 같다. API 메소드 호출 빈도에 따라 가중치를 부여하여 신뢰도를 높이는 방법을 고려하고 있으며, *API k-gram*을 비교할 때 완전히 일치하지 않더라도 부분 점수를 부여하여 강인도를 높이는 방법도 고려하고 있다. 그리고 현재 프로토타입 형태로 동작하는 *API k-gram* 버스마크 도구의 GUI를 보강하여 누구나 편리하게 사용할 수 있는 대표적인 안드로이드 버스마크 도구로의 업그레이드를 목표로 하고 있다.

참 고 문 헌

[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] "Proguard - a free Java class file shrinker, optimizer, obfuscator, and preverifier," http://proguard.sourceforge.net/.

[4] H. Tamada, M. Nakamura, A. Monden, K. Matsumoto, "Java birthmark Detecting the software theft," IEICE Transactions on Information and Systems, Vol.E88-D, No.9, pp.2148-2158, Sep., 2005.

[5] G. Myles and C. Collberg. "k-gram Based Software Birthmarks," Proceeding of the 20th ACM Symposium on Applied Computing, pp.314-318. New Mexico, USA, Mar., 2005.

[6] S.-H. Choi, S.-I. Park, H.-K. Park, H.-W. Park, "Similarity Comparison Tool Development for Detecting Theft of Android Application," Proceedings of the conference on Information Security and Cryptology, Vol.21, No.2, pp.201-205, Dec., 2011.

[7] S.-H. Choi, H.-Y. Lee, S.-M. Cho, H.-W. Park, "API Similarity Comparison Tool Development for Detecting Theft of Android Application," Proceedings of the 37th conference of the KIPS, Vol.19, No.1, pp.792-795, Apr., 2012.

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

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

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

박 희 완

e-mail : heewanpark@halla.ac.kr

1997년 동국대학교 컴퓨터공학과(학사)

1999년 KAIST 전산학과(공학석사)

2010년 KAIST 전산학과(공학박사)

2004년~2007년 삼성전자 무선사업부 책임연구원

2010년~2011년 ETRI 부설연구소 선임연구원

2011년~현 재 한라대학교 정보통신방송공학부 조교수

관심분야 : 프로그램 난독화, 역공학, 악성코드 분석, 소프트웨어 워터마킹, 정적 및 동적 분석