# 연산자 요약을 이용한 k그램 소프트웨어 버스마크

이기화*, 우균*
*부산대학교 컴퓨터공학과
e-mail:{dlrlghkekk, woogyun}@pusan.ac.kr

# Software Birthmark Based on k-gram Using Operator Abstraction

Kihwa Lee*, Gyun Woo*
*Dept. of Computer Science and Engineering, Pusan National University

요 약

소프트웨어 버스마크 기법은 도용이 의심되는 소프트웨어의 소스 코드를 얻을 수 없을 때 사용할 수 있는 소프트웨어 도용 탐지 기법이다. 이 기법은 프로그램의 바이너리나 자바 클래스 파일에서 프로그램 고유의 특징인 버스마크를 추출한 다음 프로그램간 버스마크 유사도 측정을 통해 도용을 탐지한다. 이 논문에서는 선행 연구된 k그램 버스마크 기법에 연산자 요약이라는 아이디어를 접목한 연산자 요약 k그램 버스마크 기법을 제안한다. 연산자 요약이란 연산자 우선순위가 같은 연산자의 JVM 명령어를 묶어 요약번호로 나타내는 것이다. 연산자 요약 k그램 버스마크 기법은 연산자 요약과 제어 흐름을 고려하여 생성한 연속된 k개의 요약번호 시퀀스 집합을 버스마크로 정의한다. 버스마크를 평가하기 위해 선택 정렬 메소드와 버블 정렬 메소드를 대상으로 신뢰도 실험과 강인도 실험을 하였다. 실험 결과 연산자 요약 k그램 버스마크 기법이 선행 연구된 Tamada 버스마크 기법과 k그램 버스마크 기법보다 높은 신뢰도와 강인도를 보였다.

## 1. 서론

소프트웨어는 저작권법에 의해 저작물로 규정되고 보호되고 있지만 소프트웨어 도용 사례는 날로 증가하고 있다. 소프트웨어 도용 사례는 다양한 형태로 우리 주변에서 흔히 발생한다. 상용 프로그램을 불법 복제하여 공유하거나 프로그래머가 프로젝트의 개발 시간과 비용을 절감하기 위해 저작권이 있는 프로그램의 소스 코드를 일부라도 도용하는 것은 소프트웨어 도용에 해당된다.

소프트웨어 도용 여부를 검사하는 기법은 도용이 의심되는 소프트웨어의 소스 코드를 얻을 수 있는 경우와 얻을 수 없는 경우로 나누어 생각해 볼 수 있다. 소스 코드를 얻을 수 있는 경우에는 소스 표절 검사 기법을 사용하고, 얻을 수 없는 경우에는 소프트웨어 버스마크 기법을 사용한다[4].

도용이 의심되는 소프트웨어가 상용일 경우 소스 코드를 얻을 수 있는 경우는 거의 없기 때문에 소프트웨어 버스마크 기법을 사용한다. 소프트웨어 버스마크 기법은 바이너리나 자바 클래스 파일을 분석하여 도용 여부를 판별한다. 먼저 원본 프로그램과 도용이 의심되는 프로그램에서 프로그램 고유의 특징인 버스마크를 추출한 다음 프로그램간 버스마크 유사도를 측정한다. 유사도와 프로그램 도용 여부는 비례한다.

이 논문에서는 연산자 요약을 이용한 k그램 버스마크 기법을 제안한다. 그리고 선행 연구된 버스마크 기법과 비교 실험을 통해 신뢰도와 강인도를 평가한다.

## 2. 관련 연구

Tamada는 자바 클래스 파일의 도용을 탐지하기 위해 Tamada 버스마크 기법[1]을 제안했다. 이 기법은 프로그램의 버스마크로서 필드 변수의 상수 값, 메소드 호출 순서, 상속 구조, 사용된 클래스 정보를 이용하였다. 프로그램 변환 기법에도 버스마크가 크게 손상되지 않지만 클래스 구조가 동일하고 알고리즘이 다른 두 프로그램은 구별할 수 없다는 단점이 있다[4]. Tamada가 오픈 소스로 배포하는 stigma 툴[2]에서 Tamada 버스마크 기법과 k그램 버스마크 기법을 사용해 볼 수 있다.

Myles는 k그램 버스마크 기법[3]을 제안했다. 이 기법은 자바 클래스 파일을 이루고 있는 연속된 JVM 명령어를 길이가 k인 창을 슬라이딩하여 생성한 연속된 k개의 JVM 명령어 시퀀스 집합을 프로그램의 버스마크로 정의했다. 두 프로그램을 구별하는 능력은 뛰어나지만 프로그램 변환 기법에 버스마크가 쉽게 손상되는 단점이 있다[4].

박희완은 API 함수 호출을 이용한 정적 API 트레이스 버스마크 기법[4]을 제안했다. 이 기법은 프로그램 실행 중에 호출될 수 있는 자바 API 함수 시퀀스 집합을 프로

그램의 버스마크로 정의했다. 두 프로그램을 구별하는 능력이 뛰어나고 프로그램 변환 기법에도 버스마크가 쉽게 손상되지 않는다. 하지만 조건문이 많은 프로그램을 대상으로 했을 때는 버스마크 추출 자체가 불가능하고 API 호출이 빈번하지 않은 작은 클래스 파일의 경우에는 비교 자체를 할 수 없다는 단점이 있다[6].

## 3. 소프트웨어 버스마크

소프트웨어 버스마크는 프로그램 고유의 특징으로서 어떠한 정보의 가감 없이 프로그램 자신으로부터 얻을 수 있어야 한다. 그리고 두 프로그램이 복제관계에 있다면 버스마크는 동일해야 한다. 소프트웨어 버스마크 정의는 다음과 같다.

정의 1. 버스마크
프로그램 $p$, $q$가 다음의 조건을 만족할 때, $f(p)$를 프로그램 $p$의 버스마크라 한다.
조건 1. $f(p)$는 어떠한 정보의 가감 없이 프로그램 자신으로부터 얻을 수 있어야 한다.
조건 2. 프로그램 $p$와 $q$가 복제관계에 있다면 $f(p)=f(q)$을 만족해야 한다.

다음 두 가지 속성은 버스마크를 평가하는 기준이다.

속성 1. 신뢰도 (credibility)
프로그램 $p$와 $q$가 독립적으로 작성되었다면 $f(p)\neq f(q)$을 만족해야 한다.
속성 2. 강인도 (resilience)
프로그램 $p$와 프로그램 $p$에 프로그램 변환 기법을 적용한 프로그램 $p'$이 있다고 할 때, $f(p)=f(p')$을 만족해야 한다.

제안하는 버스마크 기법이 다른 여타의 기법보다 뛰어나다는 것을 보이기 위해 신뢰도 실험과 강인도 실험을 한다. 신뢰도가 뛰어난 버스마크는 독립적으로 작성된 두 프로그램의 버스마크 유사도를 낮게 측정하고, 동일한 두 프로그램의 버스마크 유사도를 높게 측정한다. 즉, 프로그램을 구별하는 능력이 뛰어나다. 강인도가 뛰어난 버스마크는 프로그램 변환 기법을 적용해도 프로그램의 버스마크가 변하지 않는다.

## 4. 연산자 요약 k그램 소프트웨어 버스마크

연산자 요약 k그램 버스마크 기법은 연산자 요약과 제어 흐름을 고려하여 생성한 연속된 k개의 요약번호 시퀀스 집합을 버스마크로 정의한다.

### 4.1 버스마크 추출

연산자 요약 k그램 버스마크 기법을 사용하여 프로그램의 버스마크를 추출하기 위해서는 다음과 같은 과정이 필요하다. 첫 번째로 자바 클래스를 분석하여 메소드에 대한 제어 흐름 그래프를 생성한다. 이 논문에서는 제어 흐름 그래프를 생성하기 위해 선행 연구된 API 트레이스 버스마크 기법[4,5]에서 사용한 방법을 따랐다. 두 번째로 생성된 제어 흐름 그래프에 연산자 요약을 적용한다. 연산자 요약 정의는 다음과 같다.

정의 2. 연산자 요약
연산자 우선순위가 같은 연산자의 JVM 명령어를 묶어 요약번호로 나타내는 것을 연산자 요약이라 정의한다. 예를 들면, 연산자 우선순위가 제일 높은 단항 연산자와 타입 캐스트 연산자의 JVM 명령어는 요약번호 1이다. 연산자 요약을 적용하는 단계는 이미 제어 흐름 그래프를 생성한 후이기 때문에 비교 연산자 등 JVM 명령어에서 분기문을 사용하면 연산자 요약에서 제외했다. 연산자 요약을 표로 정리하면 표 1과 같다.

<표 1> 연산자 요약 – 연산자 우선순위가 같은 연산자의 JVM 명령어를 묶어 요약번호로 나타냄.

| 요약번호 | JVM 연산자 명령어 집합 |
|---|---|
| 1 | {iinc, ineg, ifeq, i2l, i2f, i2d, l2i, l2f, l2d, f2i, f2l, f2d, d2i, d2l, d2f, i2b, i2c, i2s} |
| 2 | {imul, idiv, irem} |
| 3 | {iadd, isub} |
| 4 | {ishl, ishr, iushr} |
| 5 | {instanceof} |
| 6 | {iand} |
| 7 | {ixor} |
| 8 | {ior} |

연산자 요약을 적용한 제어 흐름 그래프의 기본 블록은 JVM 명령어 시퀀스 대신 연산자 요약번호 시퀀스를 가진다. 즉, 연산자 요약에 사용되지 않는 JVM 명령어는 지우고, 사용되는 JVM 명령어는 요약번호로 변경한다. 만일 기본 블록이 요약번호를 하나도 포함하지 않는다면 기본 블록을 삭제하고, 삭제된 기본 블록으로 진입하는 에지와 나가는 에지를 연결시킨다[4]. 연산자 요약을 적용한 제어 흐름 그래프 정의는 다음과 같다.

정의 3. 연산자 요약을 적용한 제어 흐름 그래프
제어 흐름 그래프 $G(V,E)$에 대해 다음 조건을 만족하는 제어 흐름 그래프 $G'(V',E')$를 연산자 요약을 적용한 제어 흐름 그래프라고 정의한다[4]. 조건 1의 집합 $O$는 연산자 요약에서 사용되는 JVM 명령어를 원소로 갖고, 함수 $f$는 JVM 명령어를 입력으로 받아 요약번호를 출력한다.

조건 1. $V' = \{f(v) | v \in V, v \in O\}$
조건 2. $E' = \{(v,u) | (v,w) \in E \wedge (w,u) \in E \wedge w \notin V'\}$

마지막으로 연산자 요약을 적용한 제어 흐름 그래프의 모든 경로에서 길이가 k인 창을 슬라이딩하여 연속된 k개의 요약번호 시퀀스를 생성한다. 이렇게 생성한 시퀀스 집합이 연산자 요약 k그램 버스마크이다. 연산자 요약 k그램 버스마크 정의는 다음과 같다.

정의 4. 연산자 요약 k그램 버스마크
연산자 요약을 적용한 제어 흐름 그래프의 모든 경로에 대해 길이가 k인 창을 슬라이딩하여 연속된 k개의 요약번호 시퀀스를 생성한다. 이렇게 생성된 시퀀스 집합을 연산자 요약 k그램 버스마크라 정의한다.

### 4.2 버스마크 유사도 측정

4.1에서 추출한 버스마크에 대한 유사도 측정은 k그램 버스마크 기법[3,6]에서 사용한 유사도 측정법을 따른다. 두 메소드로부터 추출한 버스마크를 각각 $P$와 $Q$라고 할 때, 유사도 $M_{sim}$는 다음과 같다.

$$M_{sim}(P,Q) = max\left(\frac{|P \cap Q|}{|P|}, \frac{|P \cap Q|}{|Q|}\right)$$

$max$함수의 첫 번째 인자의 의미는 버스마크 $P$가 원본 메소드로부터 추출된 버스마크일 경우 유사도이고, 두 번째 인자의 의미는 버스마크 Q가 원본 메소드로부터 추출된 버스마크일 경우 유사도이다. 어느 것이 원본인지 알 수 없는 경우에는 두 인자 값 중 큰 값을 취한다.

## 5. 실험 및 평가

연산자 요약 k그램 버스마크 기법으로 추출한 버스마크를 평가하기 위해 신뢰도 실험과 강인도 실험을 하였다. 실험 대상은 선택 정렬 메소드와 버블 정렬 메소드로 선택했다. 두 메소드는 정렬을 한다는 목적은 같지만 내부 알고리즘은 다르다. 신뢰도가 뛰어난 버스마크는 두 메소드를 분명히 구별해야 한다. 다음 표 2는 버스마크 신뢰도 실험 결과이다.

<표 2> 버스마크 신뢰도 실험 - 선택 정렬 메소드와 버블 정렬 메소드를 대상으로 버스마크 신뢰도를 실험함.

| k | Tamada | k그램 | 연산자 요약 k그램 |
|---|---|---|---|
| 3 | 1.000 | 0.739 | .0.750 |
| 4 | 1.000 | 0.634 | 0.500 |
| 5 | 1.000 | 0.510 | 0.250 |

표 2의 버스마크 실험 결과에 의하면 Tamada 버스마크 기법은 두 메소드를 전혀 구별하지 못한다. 반면, k그램 버스마크 기법과 연산자 요약 k그램 버스마크 기법은 두 메소드를 구별한다. k값이 3일 경우에는 두 버스마크 기법의 신뢰도가 비슷하지만 k값이 증가할수록 연산자 요약 k그램 버스마크 기법의 신뢰도가 큰 폭으로 증가하였다.

프로그램 변환 기법에도 동일한 버스마크를 유지하는지 Jikes 컴파일러[7]를 사용하여 강인도 실험을 하였다. Jikes 컴파일러로 컴파일한 결과는 Javac 컴파일러로 컴파일한 결과와 다르게 표현되는 부분이 있다. 다음 표 3은

<표 3> 버스마크 강인도 실험 - Jikes를 이용하여 버스마크 강인도를 실험함.

| | Tamada | 4그램 | 연산자 요약 k그램 |
|---|---|---|---|
| 선택 정렬 | 1.000 | 0.928 | 1.000 |
| 버블 정렬 | 1.000 | 0.925 | 1.000 |

버스마크 강인도 실험 결과이다.

표 3의 버스마크 강인도 실험에서 주목할 것은 k그램 버스마크 기법과 연산자 요약 k그램 기법이다. k그램 버스마크 기법의 단점은 프로그램 변환 기법에 의해 버스마크가 쉽게 손상되는 것이다. 하지만 연산자 요약 k그램 버스마크 기법은 k그램 버스마크 기법을 기반으로 하면서도 버스마크가 동일하게 유지되고 있다. 이는 연산자 요약 k그램 버스마크 기법이 k그램 버스마크 기법의 단점을 어느 정도 보완하고 있음을 보여준다.

## 6. 결론 및 향후 연구과제

이 논문에서는 소프트웨어 버스마크 기법으로서 연산자 요약 k그램 버스마크 기법을 제안하였다. 선택 정렬 메소드와 버블 정렬 메소드를 대상으로 한 실험 결과 연산자 요약 k그램 버스마크 기법이 선행 연구된 Tamada 버스마크 기법과 k그램 버스마크 기법보다 높은 신뢰도와 강인도를 보였다. 특히, 연산자 요약 k그램 버스마크 기법은 k그램 버스마크 기법을 기반으로 하면서도 높은 강인도를 보였다. 또한, 선택 정렬 메소드와 버블 정렬 메소드를 대상으로 했기 때문에 작은 클래스 파일을 대상으로는 비교 자체가 불가능한 정적 API 트레이스 버스마크 기법[4]의 단점도 보완할 수 있을 것으로 기대된다.

향후 연구과제로 다양한 크기의 프로그램을 대상으로 한 신뢰도 실험과 강인도 실험을 할 것이다. 특히, 강인도 실험은 다양한 난독화 도구를 사용하여 엄밀하게 평가할 것이다.

## 참고문헌

[1] Haruaki Tamada, Masahide Nakamura, Akito Monden, and Ken-Ichi Matsumoto. Java birthmarks--detecting the software theft--. *IEICE transactions on information and systems*, 88(9):2148-2158, 2005.

[2] Tamada. stigma. http://stigmata.sourceforge.jp/. (2013년 03월 20일 방문).

[3] Ginger Myles and Christian Collberg. K-gram based software birthmarks. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 314-318. ACM, 2005.

[4] Heewan Park, Seokwoo Choi, Hyun-il Lim, and Taisook Han. Detecting java theft based on static api trace birthmark. In *Advances in Information and Computer Security*, pages 121-135. Springer, 2008.

[5] OW2 Consortium. ASM. http://asm.ow2.org/index.html. (2013년 03월 20일 방문).

[6] 박희완, 최석우, 임현일, 한태숙. 제어 흐름을 고려한 API k-gram 소프트웨어 버스마크. *한국정보처리학회 2008년 추계학술발표대회 논문집*, 15(2):523-526, 2008.

[7] IBM. Jikes Java Compiler. http://jikes.sourceforge.net. (2013년 03월 20일 방문)