정적 API 트레이스 버스마크를 이용한 자바 클래스 도용 탐지*

박희완 최석우O 임현일 한태숙
KAIST 전자전산학과 전산학전공
{hwpark, swchoi, hilim}@pllab.kaist.ac.kr, han@cs.kaist.ac.kr

Detecting Java Class Theft using Static API Trace Birthmark

Heewan Park Seokwoo ChoiO Hyun-il Lim Taisook Han
Division of Computer Science, Dept. of EECS, KAIST

1. 서론

소프트웨어는 쉽게 복제 및 배포될 수 있어서 표절 및 도용이 빈번하게 발생할 수 있다. 소프트웨어 도용은 프로그램 개발자나 소프트웨어 산업에 심각한 피해를 주기 때문에 불법적인 도용으로부터 지적 재산권을 보호하는 방법을 고려해야만 한다. 만일 도용이 의심되는 프로그램의 소스 코드를 얻을 수 있는 상황이라면 소스 코드 표절 검사 기법이 가장 효과적으로 사용될 수 있다. 그러나 일반적으로 프로그램은 컴파일되어 배포되기 때문에 항상 소스를 얻을 수 있는 것은 아니다. 이런 상황에서 도용을 탐지할 수 있는 방법으로 제시된 것이 소프트웨어 버스마크이다.

소프트웨어 버스마크는 생물 정보학 분야의 유전자와 유사한 개념으로서 프로그램을 인식하거나 확인하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 버스마킹 시스템은 프로그램으로부터 버스마크를 추출하고, 추출된 버스마크를 비교하여 유사도를 측정하는 시스템이다. 측정된 유사도에 따라서 프로그램 사이의 도용을 검사할 수 있다. 본 논문에서는 자바의 정적 API 트레이스를 이용한 새로운 버스마크 기법을 제안한다. 그리고 기존의 버스마크와 비교하여 프로그램 변환에 대해서 강인한 버스마크임을 실험으로서 평가한다.

2. 정적 API 트레이스 버스마크

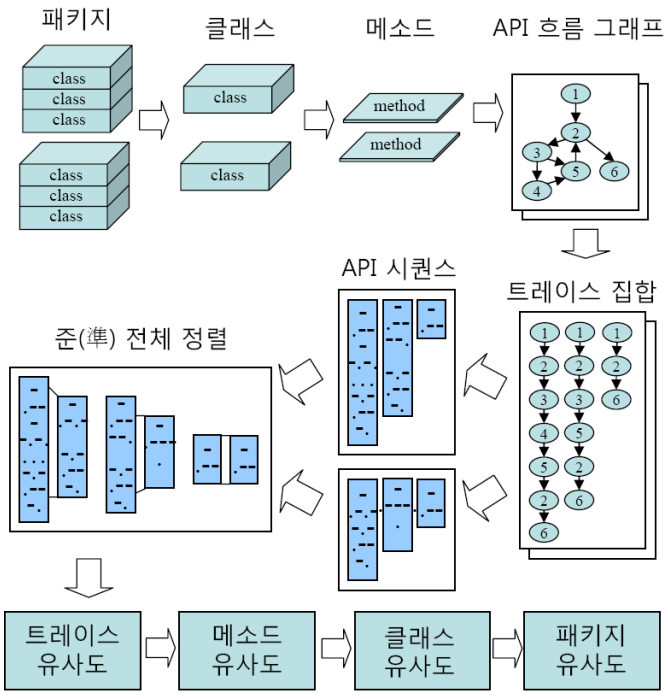

그림 1 정적 API 트레이스 버스마킹 시스템

정적 API 트레이스 버스마크는 메소드가 실행될 때 메소드 시작 지점에서 종료 지점까지 호출될 수 있는 모든 API 함수 시퀀스의 집합이다.

그림 1은 정적 API 트레이스 버스마킹 시스템의 구조도이다. 버스마크를 추출하기 위해서 자바 바이트코드를 정적으로 분석하여 메소드 단위로 API 흐름 그래프를 생성한다. API 흐름 그래프는 일반적인 제어 흐름 그래프에서 API가 포함된 노드만 추출하여 구성한 그래프를 의미한다. API 흐름 그래프로부터 트레이스를 추출하기 위해서 시작 노드에서부터 종료 노드까지 실행 가능한 경로를 모두 추출한다. 단 가능한 모든 경로의 개수는 무한하기 때문에 반복문의 경우는 반복 회수를 1회로 제한한다. 또한 버스마크의 크기를 줄이기 위하여 인접한 중복 API 호출은 1개로 축약한다.

API 시퀀스 사이의 유사도는 준(準)전체 정렬 알고리즘을 사용하여 계산한다. 메소드 단위의 유사도가 계산되면 가장 유사도가 높은 메소드끼리 매칭시켜서 클래스 레벨 유사도를 계산한다. 그리고 계산된 클래스 레벨 유사도로부터 가장 유사도가 높은 클래스끼리 다시 매칭시켜서 최종적으로 패키지 레벨 유사도를 얻는다.

* 이 논문은 2008년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임 (No. R01-2008-000-11856-0)

3. 실험 및 평가

소프트웨어 버스마크는 신뢰도와 강인도를 측정함으로 평가할 수 있다. 신뢰도는 서로 다른 프로그램을 구별할 수 있는 능력으로 신뢰도가 크다면 서로 다른 프로그램 사이의 유사도가 낮게 측정된다. 강인도는 같은 프로그램에 대해 난독화나 다른 컴파일러의 사용 등의 프로그램 변환을 통해서 변경할 때 버스마크가 손상되지 않는 정도를 의미한다. 강인도가 크다면 프로그램 변환 전후의 유사도가 높게 측정되어야 한다.

정적 API 트레이스 버스마크를 검증하기 위해서, 실험 대상으로 XML Parser 패키지 중에서 소스가 공개된 Aelfred 7.0, Crimson 1.1.3, Piccolo 1.04, XP 0.5를 선택하였다. 실험 대상 패키지에 포함된 클래스 중에서 API 호출이 2개 이하인 클래스는 제외하고, 남은 클래스에 대해서 강인도와 신뢰도 실험을 하였다.

[표 1] Tamada, k-gram, 정적 API 트레이스 버스마크에 대한 난독화 및 컴파일러 변경에 대한 강인도 실험

	Smokescreen			Jikes		
	Tamada	k-gram	정적 API 트레이스	Tamada	k-gram	정적 API 트레이스
Aelfred	0.758	0.671	0.980	0.857	0.892	0.975
Crimson	0.689	0.563	0.996	0.839	0.871	0.998
Piccolo	0.733	0.614	0.998	0.832	0.891	0.998
XP	0.720	0.589	1.000	0.920	0.894	0.984
평균	**0.725**	**0.609**	**0.994**	**0.862**	**0.887**	**0.989**

표 1은 강인도 실험 결과를 보여준다. API 트레이스는 제어 흐름을 반영하여 API 시퀀스를 추출하기 때문에 난독화 변환이나 컴파일러 변경에 대해서도 다른 버스마크보다 월등한 99% 수준의 높은 강인도 값을 얻었다.

신뢰도를 실험하기 위해서 4개의 예제 프로그램에 대해서 2개를 선택하는 모든 조합으로 6(=$_4C_2$)번의 유사도 실험을 하였다. 그 결과, Tamada, k-gram, API 트레이스 버스마크들은 모두 Crimson과 Piccolo 패키지의 유사도를 가장 높게 계산하였다. 그 원인은 Crimson과 Piccolo가 모두 오픈 소스 라이브러리인 xml.parsers와 xml.sax를 포함하고 있기 때문이다. 코드를 도용하는 경우 소스 코드를 그대로 사용하지 않고 난독화를 통해서 코드 도용을 은닉하는 것이 일반적이다. 따라서 Crimson을 Smokescreen으로 난독화한 Crimson-smoke 패키지를 Piccolo 패키지와 비교하였다.

[표 2] Crimson-smoke와 Piccolo의 유사도 분포

클래스 유사도 구간	해당 유사도 구간에 포함된 클래스 개수					
	Tamada		k-gram		정적 API 트레이스	
	원본	난독화	원본	난독화	원본	난독화
$0.0 \le C_{sim} < 0.1$	0	0	4	10	7	10
$0.1 \le C_{sim} < 0.2$	0	0	12	12	4	3
$0.2 \le C_{sim} < 0.3$	11	12	6	1	0	1
$0.3 \le C_{sim} < 0.4$	4	9	0	1	3	2
$0.4 \le C_{sim} < 0.5$	7	3	1	3	1	1
$0.5 \le C_{sim} < 0.6$	0	2	0	7	5	3
$0.6 \le C_{sim} < 0.7$	2	3	1	2	2	2
$0.7 \le C_{sim} < 0.8$	2	9	3	1	2	2
$0.8 \le C_{sim} < 0.9$	2	0	2	1	2	2
$0.9 \le C_{sim} < 1.0$	1	0	3	0	1	1
$C_{sim} = 1.0$	9	0	6	0	11	11

표 2는 난독화된 Crimson-smoke와 Piccolo의 유사도 분포 결과이다. 난독화 이후에 Tamada와 k-gram 버스마크는 100% 일치하는 클래스를 하나도 찾지 못했다. 그러나 API 트레이스는 난독화 전과 마찬가지로 100%일치하는 클래스를 11개 찾았다. 실제로 Crimson과 Piccolo에서 사용한 xml 공통 클래스 개수는 총 16개이다. 5개를 찾지 못한 이유는 두 패키지가 서로 다른 버전을 사용하였기 때문이다. 나머지 5개의 클래스는 유사도 70% 이상을 고려할 때 같은 클래스라는 것을 발견할 수 있다.

4. 결론

소프트웨어 버스마크는 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 정적 API 트레이스에 기반을 둔 자바 버스마킹 기법을 제안하였다. 정적 API 트레이스 버스마크의 특징은 다음과 같이 요약된다. API 트레이스를 생성할 때 메소드의 제어 흐름을 분석하기 때문에 난독화나 컴파일러 변경에도 강인성이 보장된다. API 트레이스를 비교할 때 시퀀스 정렬 기법 중 준전체 정렬 방법을 사용했기 때문에 서로 다른 프로그램을 구별하는 신뢰도를 높이면서 트레이스의 전단부와 후단부에 코드를 추가하는 것에 대해서도 강인한 결과를 얻는다. 본 논문에서 제안한 버스마크는 4개의 공개 XML Parser에 대한 실험을 통해 이전의 버스마크 기법인 k-gram과 Tamada에 비해 신뢰도와 강인도가 높다는 것을 보였다.