동적 API 콜 그래프 기반 버스마킹 기법[a]

하재진*, 채동규**, 김상욱[b]**, 김예솔***, 조성제***
*한양대학교 컴퓨터공학과
**한양대학교 컴퓨터소프트웨어학과
***단국대학교 소프트웨어학과
e-mail:sohback@naver.com

Program Similarity Analysis based on the Dynamic API Call Graph

Jae-Jin Ha*, Dong-Kyu Chae**, Sang-Wook Kim**, Ye-Sol Kim***, SeongJae Cho***
*Dept of Computer Engineering, Hanyang University
**Dept of Computer and Software, Hanyang University
***Dept of Software, Dankook University

요 약

본 논문에서는 동적 API 콜 그래프를 기반으로 하는 버스마킹 기법을 제안한다. API 콜 그래프를 이용함으로써 기존 방법들에 비해 프로그램의 정보를 보다 많이 반영하였다. 상용 Windows 프로그램들을 대상으로 실험을 수행하였으며, 실제로 기존의 유사성 분석 기법들에 비해 신뢰성과 강인성 측면에서 모두 성능 향상을 보였다.

1. 서론

버스마킹 기법이란 프로그램과는 구별되는 그 프로그램만의 고유한 특징 정보인 버스마크를 추출한 후, 두 버스마크 간의 유사도 비교를 통해 두 프로그램 간의 유사도를 측정하는 방법이다 [1]. 버스마킹 기법을 통해 프로그램의 도용 관계를 밝혀내거나 악성 프로그램을 탐지할 수 있다. 버스마킹 기법들은 신뢰성과 강인성을 만족해야 하는 조건이 있다. 신뢰성이란 독립적으로 개발된 두 프로그램 간에는 버스마크가 상이해야 하는 조건이며, 강인성은 버스마크가 프로그램의 표절 행위에도 쉽게 변하지 않아야 하는 조건이다 [2]. 위 두 조건들 중 하나만 만족하지 못하여도 표절 탐지의 정확도가 떨어지는 문제가 발생한다.

그러나 기존의 버스마킹 기법들은 위의 조건들 중 일부만 만족하는데 그치고 있다. 프로그램을 직접 실행시키며 실행 중에 나타나는 특징 정보들을 추출하는 방식인 동적 버스마킹의 경우 신뢰성과 강인성이 떨어져서 표절 탐지의 정확도가 낮은 문제가 있다. 또한 프로그램 자체에서 특징 정보를 추출하는 정적 버스마킹의 경우 신뢰성 및 강인성은 뛰어나지만 패킹 기술이 적용된 프로그램에는 사용할 수 없다는 단점이 있다.

(a): 본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업(NIPA-2013-H0301-13-4009)과 중소기업청에서 지원하는 2014 년도 산학연협력 기술개발사업(No. C0191469) 및 문화체육관광부 및 한국저작권위원회의 2013 년도 저작권기술 개발사업의 연구결과로 수행되었음.

(b): 교신저자.

본 논문에서는 프로그램의 특징 정보를 최대한 많이 반영하여 신뢰성과 강인성이 모두 뛰어나며, 패킹된 프로그램에도 사용할 수 있는 동적 API 콜 그래프 기반의 동적 버스마킹 기법을 제안한다. 이를 위해 PIN tool 을 이용하여 동적 API 콜 그래프를 구축하며, API 콜 그래프에 *random walk with restart* (*RWR*) [4] 알고리즘을 적용하여서 이 그래프의 특징을 충분히 반영한 점수 벡터를 생성한다. 두 벡터 간의 유사도가 두 프로그램 간의 유사도로 정의된다. 실험 결과 제안하는 방법이 기존의 동적 버스마킹 기법들과 달리 신뢰성과 강인성 모두 뛰어남을 확인하였다.

2. 제안하는 방법

API 콜 그래프를 생성하기 위해 본 논문에서는 Windows 프로그램의 동적 분석 장치인 PIN 을 사용하였다. PIN 을 통해 프로그램의 실행 중 함수가 호출될 때를 추적하고, 호출되는 함수와 그 함수가 내부적으로 호출하는 모든 함수에 관한 정보를 추출한다. 이 때 추출되는 정보는 각 함수의 명칭과, 그 함수가 API 인지 사용자 정의 함수인지의 여부이다. 프로그램의 시작부터 종료까지 위 정보들을 추출한 후, 호출된 함수들로 그래프의 노드를 구성한다. 각 노드는 API 혹은 사용자 정의 함수의 이름으로 구분하며, 노드 간의 에지는 호출한 함수에서 호출된 함수로 형성한다. 동일한 에지가 여러 번 형성된다면 이는 에지의 가중치로 표현된다.

API 콜 그래프를 구축한 다음에는 이 그래프 간의 유사도를 계산해야 한다. 그러나 그래프 간의 유사도 계산은 NP-complete 한 문제로 수행시간이 매우 오래 걸리는 문제가 있다. 본 논문에서는 random walk with

restart (RWR) 알고리즘을 사용함으로써 빠른 시간 안에 그래프 간의 유사도를 계산하고자 한다. RWR 알고리즘은 그래프의 한 노드에서 출발한 random walker 가 각 단계에서 각 노드에 도달할 확률을 계산하는 방법이다. 각 노드가 차지하는 도달 확률은 그래프의 구조적 특징에 따라 결정되며, 구조적으로 중요한 위치에 있는 노드일수록 높은 점수를 받도록 설계되었다. [4] RWR 알고리즘의 결과로 인해 API 콜 그래프 안의 모든 API 들의 도달 확률이 구해지며, 이 확률들의 벡터가 실제적인 유사도 계산에 사용된다.

RWR 알고리즘은 그래프의 노드의 수를 n, 에지의 수를 e 라 할 때 O(n+e) 의 시간 복잡도를 가지므로, 수백만 개의 노드와 에지들을 가진 API 콜 그래프에 적용하더라도 빠른 시간 안에 모든 노드의 도달 확률 계산이 가능하다. 즉 그래프 간 직접 비교는 현실적으로 어렵지만, 그래프의 구조적 특징을 충분히 반영한 하나의 벡터를 생성한 후 cosine similarity 를 통해 벡터간 유사도 계산을 수행한다면 빠른 시간 안에 그래프 비교를 수행할 수 있다.

3. 실험 결과

실험 결과에 앞서 실험 프로그램의 표 1 과 같이 선정하였다.

<표 1> 실험 프로그램

Category	Program (version)	Scenario
FTP	WinSCP (5.3.3)/(5.5.1)	1)실행→종료 2)실행→연결→종료 3)실행→연결→업로드→종료
	FileZilla (3.7.4)/(3.7.1)	
	CoreFTP(2.2)/(2.1)	
압축 프로그램	7-Zip (9.2.0)/(9.2.2)	1)실행→종료 2)실행→압축→종료 3)실행→압축해제→종료
	FreeARC (0.6.6)/(0.6.0)	
	Peazip (5.4.0)/(5.4.1)	

실험 프로그램들은 FTP, 압축 프로그램 카테고리 안의 각각 3 가지 프로그램으로 구성되어있다. 또한 각 프로그램들은 서로 다른 두 가지 버전으로 구성되어있다. 프로그램의 버전 업데이트 시 프로그램의 핵심 기능은 유지하면서 내부 메커니즘을 수정하는 것은 표절 행위와 유사하므로 서로 다른 두 가지 버전 중 최신 버전 프로그램을 표절된 프로그램으로, 구버전 프로그램을 원본 프로그램이라고 가정하였다. [3] 프로그램 실행 시나리오는 각 프로그램의 카테고리마다 3 가지의 시나리오가 있으며, 프로그램을 시작시킨 직후에 바로 종료시키는 시나리오와 프로그램의 핵심 동작 시나리오 2 개로 구성되어있다.

표 2 와 3 은 각각 FTP, 압축 프로그램, 미디어 플레이어 카테고리 내에서 각 프로그램들끼리 측정된 유사도 결과이다. 굵은 글씨로 되어 있는 숫자가 제안하는 방법으로 도출한 유사도 결과이며, 나머지 숫자들은 기존 방법들을 사용하여 도출한 유사도 결과이다. SEQ 는 Tamada [1] 의 EXESEQ 버스마크를, FREQ 는 Tamada [1] 의 EXEFREQ 버스마크를 이용하여 도출된 유사도를 의미한다.

<표 2> FTP 프로그램 실험 결과

FTP	WinSCP		FileZilla		CoreFTP	
WinSCP	**97.6**	88.0	**3.6**	9.9	**15.9**	2.9
		99.9		28.4		69.7
FileZilla			**98.8**	87.1	**2.5**	6.6
				99.9		44.5
CoreFTP	**API**	SEQ			**99.9**	81.4
		FREQ				99.9

<표 3> 압축프로그램 실험 결과

ZIP	7-zip		FreeARC		Peazip	
7-zip	**99.9**	96.7	**17.0**	9.6	**2.2**	14.2
		99.9		62.8		80.5
FreeARC			**99.9**	57.9	**3.4**	3.6
				99.9		48.7
Peazip					**99.9**	66.4
						99.9

실험 결과 기존의 EXESEQ 버스마크의 경우 대부분의 경우에서 서로 다른 프로그램의 유사도를 낮게 측정하였다. 그러나 표절 프로그램 간의 유사도 또한 낮게 측정하는 문제가 있었다. 결국 신뢰성 측면에서는 좋은 성능을 보였지만, 강인성 측면에서는 나쁜 성능을 보였다. 반대로 EXEFREQ 의 경우 표절 프로그램 간의 유사도를 높게 측정하였으나 서로 다른 프로그램의 유사도 또한 전체적으로 높게 측정하였다. 결국 강인성 측면에서는 좋은 성능을 보였지만, 신뢰성 측면에서는 낮은 성능을 보였다. 반면 제안하는 방법의 경우 서로 다른 프로그램의 경우에는 전체적으로 낮은 유사도를 도출했으며, 표절 프로그램의 경우에는 높은 유사도를 도출하였다. 이는 API 콜 그래프에 API 의 호출 빈도 정보와 호출 순서 정보를 모두 반영함으로써 보다 많은 정보를 반영하였고, 각 방법들이 독립적으로 가졌던 장점들을 모두 가졌기 때문인 것으로 보인다. 위 실험을 통해 제안하는 방법이 기존의 dynamic 버스마킹 기법들과는 달리 신뢰성, 강인성 측면에서 모두 우수함을 알 수 있었다.

4. 결론

본 논문에서는 동적 API 콜 그래프를 이용한 프로그램 유사성 분석 기법을 소개하였다. 또한 그래프의 유사도 계산을 빠르게 할 수 있는 방법을 소개하였다. 실험 결과 제안하는 방법이 기존 방법들에 비해 신뢰성과 강인성 측면에서 모두 뛰어남을 확인하였다.

참고문헌

[1] H. Tamada, K. Okamoto, M. Nakamura, A. Monden, and K. Matsumoto, “Dynamic Software Birthmarks to Detect the Theft of Windows Applications,” *ISFST*, 2004

[2] G. Myles and C. Collberg, “k-Gram Based Software Birthmarks,” *ACM SAC*, pp. 314-318, 2005

[3] S Choi, H Park, H Lim, and T Han, “A Static API Birthmark for Windows Binary Executables,” *Journal of Systems and Software*, pp. 862-873, 2009

[4] T. Haveliwala, “Topic-Sensitive Pagerank,” *WWW*, pp. 517-526, 2002