Windows PE 파일의 임포트 테이블에 기반한 소프트웨어 버스마킹(Birthmarking) 기법*

박희완 임현일 최석우 한태숙
한국과학기술원 전자산학과 전산학전공, 첨단정보기술 연구센터(AITrc)
{hwpark, hilim, swchoi}@pllab.kaist.ac.kr han@cs.kaist.ac.kr

A Software Birthmark of Windows PE File Based on Import Table

Heewan Park, Hyun-il Lim, Seokwoo Choi, and Taisook Han
Division of Computer Science and Advanced Information Technology Research Center(AITrc).
Korea Advanced Institute of Science and Technology

요 약

소프트웨어 버스마크는 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 Windows PE(Portable Executable) 파일의 API에 대한 정보를 가지는 임포트 테이블에 기반한 프로그램 버스마킹 기법을 제안한다. 버스마크의 신뢰도를 높이기 위한 방법으로 대부분의 Windows 프로그램에서 사용되는 범용의 API는 버스마크에서 제외시키고 프로그램 개개의 특성을 나타낼 수 있는 특화된 API에 초점을 맞추어서 비교하는 방법을 사용한다.

본 논문에서 제안한 버스마킹 기법을 평가하기 위해서 다양한 카테고리의 Windows 프로그램에 대해서 실험을 하였다. 신뢰도를 측정하기 위해서 같은 프로그램에 대해서 버전별로 비교를 하였고, 프로그램의 분류에 따라서 유사한 카테고리와 다른 카테고리에 대해서 비교를 하였다. 프로그램의 변환이나 난독화에도 견딜 수 있는 강인도(Resilience)를 평가하기 위해서 서로 다른 컴파일러를 사용하여 생성된 프로그램에 대해서 비교를 하였다. 실험 결과에서 본 논문에서 제안하는 버스마크가 프로그램의 특징을 충분히 표현하고 있음을 보여준다.

1. 서 론

최근 많은 프로그램들이 오픈 소스 프로젝트로 개발되고 있다. 대부분의 오픈 소스 프로그램들은 GPL(GNU Public License) 규정에 따라 공개하고 있으며, 이 소스를 이용하는 소프트웨어 또한 이 규정에 따라 공개하는 것을 원칙으로 한다. 그러나 많은 소프트웨어 개발 업체들이 GPL 기반의 공개 소스를 사용함에도 불구하고 소스를 공개하지 않는 등 GPL 규정을 따르지 않는 것으로 알려지고 있다. 이런 경우 GPL의 오픈 소스를 이용해서 개발되었는지 검증할 수 있는 방법이 필요하다. 소프트웨어 버스마킹은 이런 소프트웨어 도용을 확인하는데 사용될 수 있는 기법이다.

소프트웨어 버스마크는 프로그램을 식별하는데 사용될 수 있는 고유한 특징을 말한다. 프로그램으로부터 추출된 문자열을 비교하거나 체크섬을 비교하는 것 등이 버스마킹 기법으로 사용될 수 있다. 이 경우에는 추출된 문자열이나 체크섬이 버스마크가 된다. 기존 연구가 많이 있는 소스 표절 검사 기법과도 연관성이 있으나 버스마킹은 비교 대상이 프로그램 소스가 아니라 개발 후 배포되는 바이너리나 바이트코드라는 것이 큰 차이점이다.

* 본 연구는 첨단정보기술 연구센터를 통하여 과학재단의 지원을 받았음.

이 논문에서는 Windows PE(Portable Executable)[1] 파일의 임포트 테이블에 기반한 프로그램 버스마킹 기법을 제안한다. 모든 Windows PE 파일은 프로그램 실행에 필요한 API 이름을 임포트 테이블에 포함하고 있기 때문에 임포트 테이블에 포함된 API 정보를 추출하여 버스마크로 사용할 수 있다. 이 기법은 프로그램의 특성을 분별할 수 있을 정도로 많은 수의 API가 사용되는 Windows PE 파일에만 적용할 수 있다는 한계가 있고, Windows 프로그래밍에서 빈번하게 사용하는 범용의 API들에 대해서는 프로그램의 특징을 구별할 수 있는 신뢰도가 떨어진다는 문제점이 있다. 버스마크의 신뢰도를 높이기 위하여 범용의 API는 버스마크에서 제외시키고 프로그램의 특화된 API에 초점을 맞추어서 비교하는 방법을 제안한다.

본 논문에서 제안한 버스마킹 기법을 평가하기 위해서 다양한 카테고리의 Windows 프로그램에 대해서 실험을 한다. 신뢰도를 측정하기 위해서 같은 프로그램에 대해서 버전별로 비교를 하고, 프로그램의 분류에 따라서 유사한 카테고리와 다른 카테고리에 대해서 비교를 한다. 프로그램의 변환이나 난독화에도 견딜 수 있는 강인도(Resilience)를 평가하기 위해서 서로 다른 컴파일러를 사용하여 생성된 프로그램에 대해서 비교를 한다. 올바른 강인도 평가를 위해서는 난독화 도구로 변환을 하고 평가해야 하지만 실험 가능한 Windows PE 파일 난독화 도구를 구할 수 없는 현실에서 Tamada[5]의 논문과 유

사하게 여러 버전의 컴파일러를 선택하여 실험한다.

2. 관련 연구

소프트웨어 지적 재산권을 위반하는 방법은 소프트웨어의 불법 재판매, 무단 변경, 악의적인 역설계 등 다양한 방법으로 시도될 수 있다. 이를 보호하는 한 방법으로 소프트웨어 워터마킹은 소프트웨어의 제작자를 증명할 수 있는 방법으로 가장 잘 알려진 방법이다. 그러나 워터마킹은 소프트웨어에 고유한 코드를 내장시켜야 하기 때문에, 항상 적용 가능한 방법은 아니다.

소프트웨어 버스마크(birthmark)[3,4,5,6,7,8,9]는 프로그램의 식별에 사용될 수 있는 프로그램 내부에 나타나는 고유의 특성을 의미한다. 소프트웨어 버스마크에서는 부가적인 고유 코드를 삽입할 필요가 없이 프로그램의 여러 가지 특성들을 요약하는 방법을 이용한다. 버스마크는 프로그램의 제작자를 증명하거나 불법적으로 재사용되었음을 명시하지는 못하지만, 워터마크가 적용될 수 없는 환경에서 프로그램의 유사성을 비교하거나 특성을 분석하는데 사용될 수 있다. 또한 워터마크와 함께 사용되면 악의적인 공격에 의해서 워터마킹이 손상되었을 경우에도 워터마킹보다 상대적으로 강인한 버스마킹 기법을 적용하여 보완이 가능하다.

[3, 4]에서는 프로그램의 도용을 확인하는 정적 소프트웨어 버스마크를 처음으로 제안하였다. 이 방법은 자바에서 특징지어지는 네 가지 특성을 분석할 수 있는 자바 클래스 파일에 적용가능하다. 이 특성들은 필드 변수에 사용된 상수 값들, 메쏘드 호출의 시퀀스, 상속 구조, 사용된 클래스들의 값을 소프트웨어 버스마크로 이용한다.

[5]에서는 동적 버스마크를 소개하고, Windows 프로그램에 대한 시스템 콜의 추적에 기반한 버스마크 방법을 제안했다. 소프트웨어 실행 동안에 호출되는 API 함수의 시퀀스와 횟수를 버스마크로 이용한다. 이 방법은 프로그램을 직접 실행시키고 종료되기 전까지의 API 호출을 후킹 하는 기법이기 때문에 사용자 상호 작용이 없이 동작하는 프로그램에서만 사용가능하다는 단점이 있다.

[6]에서는 전체 프로그램 패스(whole program path) 버스마크라는 또 다른 동적 버스마크를 제안하였다. 이 방법은 인스트럭션 시퀀스를 통해서 프로그램의 버스마크를 구성하였다.

k-gram 버스마크[7]는 프로그래밍 언어에 관계없이 적용 가능한 정적 버스마크이다. k-gram은 문자, 단어, opcode 등의 연속된 k 개의 부분 문자열(substring)을 의미한다. k-gram 버스마크에서는 실행 프로그램에 사용되는 opcode의 k-gram들의 집합을 버스마크로 사용한다.

[8]에서는 자바 프로그램에 대해서 실행 중의 API 호출 시퀀스의 집합을 이용하는 동적 버스마크 방법이다. 이 방법은 [5]와 동일하게 사용자와의 상호 작용이 없이 동작하는 프로그램에만 적용할 수 있다는 한계가 있다.

[9]에서는 실행 프로그램의 함수별 API 호출 집합을 이용한 정적 버스마크를 제안하였다. 이 방법을 프로그램에서 함수의 API 사용 구조를 비교함으로서 프로그램 사이의 유사성을 비교하였다.

지금까지 연구된 버스마킹 기법은 주로 자바 바이트코드에 대한 내용이었다. 자바 바이트코드는 Windows PE 파일과 비교하여 구조가 간단하기 때문에 여러 가지 분석 도구들이 많이 개발되어 왔고 버스마킹으로 사용할 수 있는 프로그램의 특성을 추출하는 것도 어렵지 않다. 반면 Windows PE 파일은 구조가 복잡하여 정적인 분석을 통한 특성의 추출이 어렵다. 지금까지 발표된 정적인 Windows PE 파일 버스마킹 기법은 논문[9]에서 제안된 방법이 유일하다. 이 방법의 단점은 분석의 전단부로 IDAPro 프로그램을 사용하기 때문에 분석의 시간이 오래 걸리고 다수의 프로그램을 대상으로 하는 배치 형태의 분석에 적합하지 않다는 것이다. 본 논문에서 제안하는 임포트 테이블 기반의 버스마킹 기법은 분석 방법이 간단하기 때문에 다수의 프로그램에 대해서 빠른 결과를 얻는 방법으로 선택될 수 있다.

3. 임포트 테이블에 기반한 소프트웨어 버스마크

3.1 소프트웨어 버스마크

Tamada와 Myles는 copy 관계를 이용하여 소프트웨어 버스마크를 정의했다. 다음은 버스마크에 대한 Myles[7]의 정의이다.

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 부가적인 정보 없이 p 자신으로부터 얻는다.
조건 2. 프로그램 p와 q가 서로 copy 관계에 있다면 f(p) = f(q) 이다.

조건 1은 워터마킹과 버스마킹이 구분되는 특징이라고 할 수 있다. 워터마킹은 배포 전에 부가적인 정보가 삽입된 경우에 대해서 적용이 가능한 방법이지만, 버스마킹은 프로그램이 원래 가지고 있던 고유한 특징을 추출하는 방법이다.

조건 2는 copy된 두 프로그램의 경우에 프로그램의 특성이 남게 되고, 이런 경우 같은 버스마크가 추출된다는 것을 나타낸다. 여기서 copy라는 개념은 완전한 복제의 개념 외에도 프로그램 변환을 포함한다. 동일한 프로그램은 최적화나 난독화와 같은 프로그램 변환을 거치더라도 같은 버스마크가 추출되어야 한다. 조건 2의 역은 성립하지 않는다. 버스마크가 서로 같다고 해서 두 프로그램이 서로 copy라는 결론을 내릴 수 없다. 따라서 버스마킹 기법은 프로그램의 복제나 도용을 확인하는 과정에서 보조적인 수단으로 사용될 수 있지만 절대적인 증거로서 사용되지는 못한다.

다음 두 가지 property는 Tamada와 Myles가 제시한 버스마크가 만족시켜야 하는 평가 기준을 나타낸다.

Property 1 (신뢰도)
같은 기능을 하는 두 프로그램 p와 q에 대해서, p와 q가

독립적으로 개발되었을 때, f(p)≠ f(q)을 만족해야 한다.

Property 2 (강인도)
프로그램 p'이 프로그램 p로부터 변환되었다고 할 때, f(p) = f(p')을 만족해야 한다.

Property 1은 버스마킹 기법의 신뢰도를 나타낸다. 두 프로그램이 비록 동일한 기능을 하는 프로그램이라고 하더라도 독립적으로 개발된 프로그램이라면 두 프로그램에서 추출된 버스마크는 같지 않아야 신뢰도가 높다고 평가할 수 있다.
Property 2는 버스마킹 기법의 강인도를 나타낸다. 컴파일러 최적화나 난독화와 같은 프로그램 변환 기법을 적용하기 전과 후의 버스마크는 같아야 강인도가 높다고 평가할 수 있다.

3.2 임포트 테이블 기반 버스마크

Windows PE 파일은 임포트 테이블에 프로그램이 사용하는 DLL(Dynamic Link Library)과 API 함수 이름을 포함하고 있다. API 함수는 기능별로 그래픽, 멀티미디어, 네트워크, 보안과 같은 다양한 종류로 나눌 수 있는데, Windows 프로그램은 API 함수 호출을 통해서만 Windows OS가 제공하는 기능을 수행할 수 있기 때문에 다른 함수로 교체하기 어렵다는 특징이 있다. 따라서 프로그램이 사용하고 있는 API 함수는 그 프로그램의 특성을 반영하는 버스마크로서 사용될 수 있다. 본 논문에서는 Windows PE 파일의 임포트 테이블에서 추출할 수 있는 API 함수 목록을 버스마크로 사용하였다.

3.2.1 Windows PE 파일의 구조

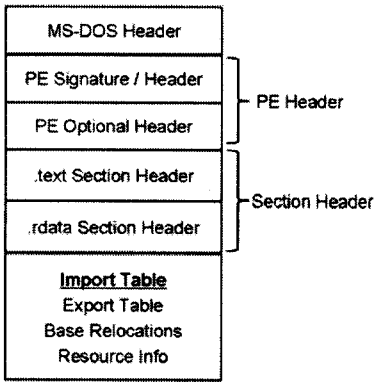

그림 1 Windows PE 파일의 구조

그림 1은 Windows PE 파일의 구조를 나타낸다. 프로그램이 사용하는 API 함수 리스트는 임포트 테이블에 포함되어있다. Section Header 정보로부터 임포트 테이블의 위치를 확인하고 임포트 테이블에서 API 함수 리스트를 읽어낼 수 있다.

3.2.2 유사도(Similarity)와 포함관계(Containment) 계산

본 논문에서 제안하는 버스마킹 기법을 실험적으로 평가하기 위해서는 프로그램의 유사도와 포함관계에 대한 정의가 필요하다.

정의 2 (프로그램 유사도)
두 프로그램 A와 B에 대해서 S(A)와 S(B)를 각각 프로그램 A와 B의 임포트 테이블에 포함된 API의 집합이라고 할 때 두 프로그램의 유사도는 다음과 같이 정의한다.

$$Similarity(A,B) = \frac{n(S(A) \cap S(B))}{n(S(A) \cup S(B))}$$

프로그램의 유사도는 두 프로그램이 임포트 테이블에 포함하고 있는 모든 API에 대해서 공통된 API의 비율을 계산한 값이다. 유사도는 0과 1 사이의 값을 가지며, 두 프로그램이 완전하게 같은 API 집합을 포함하고 있을 경우 1이 된다.

정의 3 (프로그램 포함관계)
두 프로그램의 A와 B사이의 포함관계는 다음과 같이 정의한다.

$$Containment(A,B) = \frac{n(S(A) \cap S(B))}{min(n(S(A)), n(S(B)))}$$

프로그램 포함관계는 두 프로그램 중 적은 개수의 API를 포함한 프로그램을 기준으로 공통된 API의 비율을 계산한 값이다. 포함관계는 0과 1사이의 값을 가지며, 한 프로그램의 API 집합이 모두 다른 프로그램의 부분집합으로 포함될 경우에 1이 된다.

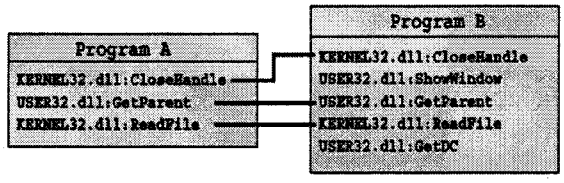

그림 2 예제 프로그램 A, B

그림 2의 예제 프로그램 A, B에 대해서 유사도와 포함관계를 정의 2와 정의 3에 의해서 계산해보면 다음과 같다.

유사도 = 3/5 = 0.6
포함관계 = 3 / min(3,5) = 1

두 프로그램의 유사도는 0.6이지만 포함관계로 보면 프로그램 A가 B에 완전하게 포함되기 때문에 1이 된다.

4. 실험 및 평가

본 논문에서 제안한 임포트 테이블 기반 버스마킹 시스템은 Windows 환경에서 C++ 프로그래밍언어로 구현되었다. 프로그램의 동작은 다음과 같다. 먼저 두 개의 Windows PE 파일을 입력으로 받아서 각 프로그램이 임포트 테이블에 포함하고 있는 API 리스트를 추출해낸다. 그리고 두 API 리스트를 이용하여 유사도와 포함관계를 계산하여 출력한다.

임포트 테이블 기반 버스마킹 기법을 평가하기 위해서 다양한 카테고리의 Windows 프로그램에 대해서 실험을 하였다. 신뢰도를 측정하기 위해서 같은 프로그램에 대해서 버전별로 비교를 하였고, 프로그램의 분류에 따라서 유사한 카테고리와 다른 카테고리에 대해서 비교를 하였다. 프로그램의 변환이나 난독화에도 견딜 수 있는 강인도를 평가하기 위해서 서로 다른 컴파일러를 사용하여 생성된 프로그램에 대해서 비교를 하였다.

신뢰도에 대한 실험을 위해서 사용된 예제 프로그램은 표 1과 같다.

표 1 신뢰도에 대한 실험에 사용된 예제 프로그램

카테고리	예제 프로그램	버전	번호
문서 편집기	UltraEdit	7.0 / 7.2	1
	EditPlus	2.0 / 2.1	2
FTP 클라이언트	FileZilla	2.2.14 / 2.2.26	3
	CuteFTP32	3.5.4 / 4.0.19	4
터미널 프로그램	Putty	0.56 / 0.58	5
	SecureCRT	5.5.0 / 5.5.1	6
P2P 프로그램	Donkeyhote	2.40 / 2.54	7
	Emule	0.45b / 0.47c	8
그래픽 도구	ACDSee	4.01 / 4.02	9
	xnView	1.21 / 1.25a	10
MP3 플레이어	Winamp	5.23 / 5.35	11
	Foobar2000	0.9.1 / 0.9.4	12
동영상 플레이어	GOM Player	2.0.0 / 2.1.6	13
	Adrenalin	2.1 / 2.2	14
CD 라이터	CDRWin	3.8 / 3.9	15
	DVDCopy	2.2.6 / 2.5.1	16
다운로드 매니저	Flashget	1.6.5 / 1.7.2	17
	NetTransport	2.3.0 / 2.4.1	18
CD 에뮬레이터	Daemon	4.3.0 / 4.9.0	19
	CD Space	4.1 / 5.0	20

임포트 테이블에 포함된 API에 기반한 버스마킹 기법의 유효성을 검증해보기 위해서 API 함수 분포를 조사해보았다. 그림 3은 예제 프로그램에 포함된 API 함수의 분포를 나타낸다. 40개의 예제 프로그램으로부터 총 1925개의 함수가 추출되었는데 모든 프로그램에서 예외 없이 추출된 범용 API의 개수는 7개였고, 1개의프로그램에서만 추출된 특화된 API 개수는 175개 였다.

그림 3에서 최고 값은 2개의 프로그램에서 추출된 API 개수로 678개가 추출되었으며 최저 값은 39개의 프로그램에서 추출된 API의 개수로 1개가 추출되었다.

40개의 예제 프로그램에서 모두 추출될 정도로 빈번하게 사용되는 범용 API의 경우는 버스마크로 사용될 경우 프로그램의 신뢰도를 떨어뜨리는 작용을 하기 때문에 이에 대한 고려가 필요하다. 본 논문에서는 예제 프로그램에서 사용된 API 1925개 중에서 90%를 커버하는 개수인 1732개를 버스마크로 사용하고 나머지 10%에 해당하는 범용 API는 라이브러리 파일로 만들어서 유사도와 포함관계를 계산할 때 제외하도록 하였다.

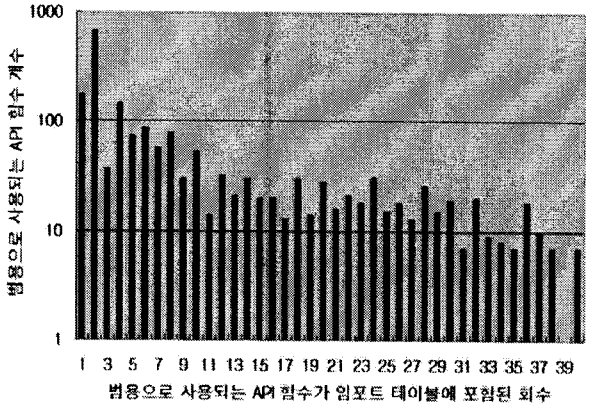

그림 3 예제 프로그램에 포함된 API 함수 분포

4.1 신뢰도 평가

그림 4는 같은 프로그램에 대한 버전별 비교 결과를 보여준다. 3번 프로그램인 FileZilla와 20번 프로그램인 CD Space에서 0.6이하의 값이 계산되었고 나머지 프로그램에서는 모두 0.7 이상의 값이 계산되었다. 같은 프로그램이라도 버전사이에 업데이트가 많을 경우에는 유사도와 포함관계 값이 낮아질 수 있다.

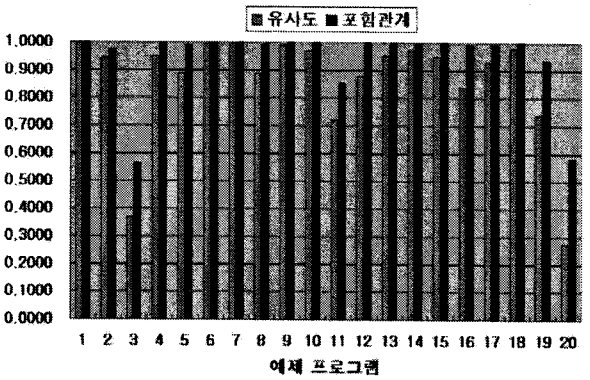

그림 4 같은 프로그램에 대한 버전별 비교

그림 5는 유사한 카테고리의 다른 프로그램에 대한 비교 결과를 보여준다. 유사한 카테고리의 프로그램들은 동일한 API를 사용할 확률이 높기 때문에 유사도와 포함관계가 높아질 수 있다. 7번 프로그램 Donkeyhote와 8번 프로그램 Emule은 유사도와 포함관계 값이 모두 0.8 이상으로 계산되었다. 그 이유는 Donkeyhote가 Emule의 P2P모듈을 그대로 사용하고 GUI를 수정한 프로그램이기 때문이다. 1번과 2번의 에디터 프로그램과 13번과 14번의 동영상 플레이어도 유사도가 0.5 이상으로 계산되었다. 유사한 카테고리일 뿐만 아니라 프로그램의 기

능도 매우 유사하기 때문에 높은 값이 계산된 것으로 평가할 수 있다.

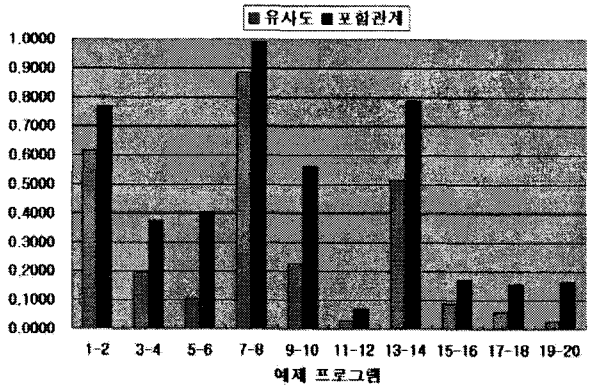

그림 5 유사한 카테고리의 프로그램 비교

그림 6은 서로 다른 카테고리의 프로그램을 비교한 결과를 보여준다. 서로 다른 카테고리일 경우 다른 종류의 API를 호출할 가능성이 높기 때문에 유사도와 포함관계가 낮은 값을 가진다.

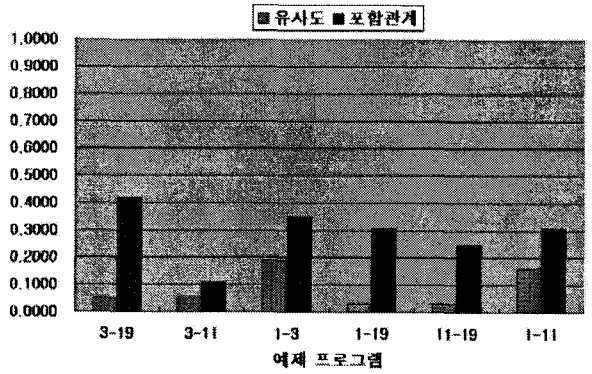

그림 6 다른 카테고리의 프로그램 비교

4.2 강인도 평가

강인도에 대한 실험을 위한 예제 프로그램으로는 오픈 소스 hex 편집기인 frhed 1.1.0버전[10]과 MP3 Tag 편집기인 Super Tag Editor 2.02버전[11](이하 STE)을 사용하였다. frhed는 순수 Win32 API로 작성된 프로그램이고 STE는 MFC 라이브러리를 사용하여 작성된 프로그램이다. 컴파일러는 Visual C++6.0, Visual C++ .NET 2003, Visual C++ .NET 2005를 사용하였다.
서론에서도 언급했듯이 올바른 강인도 평가를 위해서는 난독화 도구로 변환을 해야 하지만 실험 가능한 Windows PE 파일 난독화 도구를 구할 수 없는 현실에서 Tamada[5]의 논문과 유사하게 여러 컴파일러를 선택하여 실험하였다.
표 2는 다양한 버전의 컴파일러로 생성된 frhed 프로그램의 정보를 보여준다. 파일 크기는 컴파일러와 옵션 변화에 따라서 차이가 크지만 API 개수는 거의 변화가 없다는 것을 알 수 있다.

표 2 컴파일러 버전별 frhed 프로그램 비교

컴파일러	컴파일 옵션	파일 크기 (bytes)	총 API 개수	범용 API 제외 개수
VC++ 6.0	디버그	409,668	303	171
	릴리즈	317,952	290	160
VC++ .NET 2003	디버그	446,464	311	179
	릴리즈	331,776	297	168
VC++ .NET 2005	디버그	716,800	320	185
	릴리즈	377,344	302	169

표 3은 frhed 프로그램의 강인도 측정 결과를 보여준다. 컴파일러나 옵션 변화에 관계없이 유사도와 포함관계가 모두 0.8 이상의 값이 계산되었다. 프로그램이 사용하는 API는 컴파일러나 옵션 변화에 강인한 것을 확인할 수 있다.

표 3 frhed 프로그램의 강인도 측정 결과

컴파일러	컴파일 옵션		VC++ 6.0		VC++ .NET 2003		VC++ .NET 2005	
			디버그	릴리즈	디버그	릴리즈	디버그	릴리즈
VC++ 6.0	디버그	유사	1.000	0.9244	0.9553	0.9045	0.9037	0.8785
		포함	1.000	0.9938	1.0000	0.9583	0.9883	0.9408
	릴리즈	유사	-	1.000	0.8833	0.9524	0.8351	0.9128
		포함	-	1.000	0.9938	1.0000	0.9812	0.9812
VC++ .NET 2003	디버그	유사	-	-	1.000	0.9278	0.9259	0.8610
		포함	-	-	1.000	0.9940	0.9777	0.9527
	릴리즈	유사	-	-	-	1.000	0.8579	0.9148
		포함	-	-	-	1.000	0.9702	0.9583
VC++ .NET 2005	디버그	유사	-	-	-	-	1.000	0.9032
		포함	-	-	-	-	1.000	0.9941
	릴리즈	유사	-	-	-	-	-	1.000
		포함	-	-	-	-	-	1.000

표 4는 다양한 버전의 컴파일러로 생성된 STE 프로그램의 정보를 보여준다. 파일 크기는 컴파일러와 옵션 변화에 따라서 차이가 크고, API 개수는 디버그 버전과 릴리즈 버전사이에 차이가 크다. 그 이유는 STE가 MFC 라이브러리 기반으로 작성되었기 때문에 디버그 모드로 컴파일 할 경우 MFC에 대한 디버그 함수가 추가로 사용되기 때문이다.

표 4 컴파일러 버전별 STE 프로그램 비교

컴파일러	컴파일 옵션	파일 크기 (bytes)	총 API 개수	범용 API 제외 개수
VC++ 6.0	디버그	1,929,304	686	496
	릴리즈	716,800	414	234
VC++ .NET 2003	디버그	1,908,736	762	570
	릴리즈	770,048	431	247
VC++ .NET 2005	디버그	3,260,416	725	534
	릴리즈	880,640	419	235

표 5 Super Tag Editor 프로그램의 강인도 측정 결과

컴파일러	컴파일 옵션		VC++ 6.0		VC++ .NET 2003		VC++ .NET 2005	
			디버그	릴리즈	디버그	릴리즈	디버그	릴리즈
VC++ 6.0	디버그	유사	1.000	0.4688	0.8316	0.4455	0.7577	0.4004
		포함	1.000	0.9957	0.9758	0.9271	0.8952	0.8894
	릴리즈	유사	-	1.000	0.3934	0.8643	0.3714	0.7632
		포함	-	1.000	0.9701	0.9530	0.8889	0.8675
VC++ .NET 2003	디버그	유사	-	-	1.000	0.4308	0.8872	0.3927
		포함	-	-	1.000	0.9960	0.9719	0.9660
	릴리즈	유사	-	-	-	1.000	0.4072	0.8755
		포함	-	-	-	1.000	0.9150	0.9574
VC++ .NET 2005	디버그	유사	-	-	-	-	1.000	0.4374
		포함	-	-	-	-	1.000	0.9957
	릴리즈	유사	-	-	-	-	-	1.000
		포함	-	-	-	-	-	1.000

표 5는 STE 프로그램의 강인도 측정 결과를 보여준다. 디버그 모드와 릴리즈 모드에서 유사도가 0.3정도로 떨어지는 경우가 있는 것을 확인할 수 있다. 디버그 모드에서 추가된 MFC 디버그 함수 때문이다. 포함관계를 보면 릴리즈 모드의 API가 디버그 모드의 API에 대부분 포함이 되기 때문에 0.8 이상의 값이 계산되는 것을 확인할 수 있다.

5. 결론

소프트웨어 버스마크는 프로그램을 식별하는데 사용될 수 있는 프로그램의 고유한 특징을 말한다. 본 논문에서는 Windows PE 파일에 사용되는 API에 대한 정보를 이용한 버스마크 방법을 제안하였다. 프로그램에서 사용되는 API들은 임포트 테이블로부터 추출하였고, 추출된 API들의 관계를 비교하는 방법을 사용하였다. 버스마크의 신뢰도를 높이기 위한 방법으로 대부분의 Windows 프로그램에서 사용되는 범용 API는 버스마크에서 제외시키고 프로그램 사이의 특성을 나타낼 수 있는 특화된 API에 초점을 맞추어서 비교하는 방법을 사용한다.

본 논문에서 제안한 버스마킹 기법을 평가하기 위해서 다양한 카테고리의 Windows 프로그램에 대해서 실험을 하였다. 신뢰도를 측정하기 위해서 같은 프로그램에 대해서 버전별로 비교를 하였고, 프로그램의 분류에 따라서 유사한 카테고리와 다른 카테고리에 대해서 비교를 하였다. 프로그램의 변환이나 난독화에도 견딜 수 있는 강인도(Resilience)를 평가하기 위해서 서로 다른 컴파일러를 사용하여 생성된 프로그램에 대해서 비교를 하였다. 실험 결과에서 본 논문에서 제안하는 버스마크가 프로그램의 특징을 충분히 표현하고 있음을 보여준다.

본 논문에서 제안한 방법은 임포트 테이블에서 추출된 API를 사용하였기 때문에, API가 프로그램의 특성을 반영하기 힘든 범용의 API가 대부분을 차지하거나, API가 많이 사용되지 않은 프로그램에 대해서는 충분한 변별력을 보이지 못한다는 단점이 있다.

향후 연구 과제로 다른 버스마킹 시스템과의 신뢰도 및 강인도 평가 및 Windows PE 파일에 제한되지 않고 다양한 운영체제 및 프로그래밍 언어로 작성된 실행 프로그램에 대한 방법으로의 확장을 고려하고 있다.

참고 문헌

[1] Microsoft Portable Executable and Common Object File Format Specification. Revision 8.0. Microsoft Corporation

[2] Using BinDiff for Code theft detection. http://www.sabre-security.com/products/Code Theft.pdf

[3] H. Tamada, M. Nakamura, A. Monden, and K. Matsumoto. Design and evaluation of birthmarks for detecting theft of java programs. In Proc. IASTED International Conference on Software Engineering (IASTEDSE2004), pages 569-575, Feb 2004.

[4] 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

[5] Haruaki Tamada and Keiji Okamoto. Dynamic software birthmarks to detect the theft of windows applications. In Proc. Int. Symposium on Future Software Technology 2004, 2004. Xi-an, China.

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

[7] 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.

[8] David Schuler and Valentin Dallmeier. Detecting Software Theft with API Call Sequence Sets. Workshop Software Reengineering (WSR 2006), Bad-Honnef, Germany.

[9] Seokwoo Choi, Heewan Park, Hyun-il Lim, and Taisook Han. A Static Birthmark of Binary Executables Based on API call Structure. 12th Annual Asian Computing Science Conference (ASIAN 2007), Carnegie Mellon University Qatar Campus, Doha, Qatar.

[10] Raihan Kibria. frhed - free hex editor v1.1.0. http://www.codeproject.com/tools/frhed.asp

[11] Mercury's Software Lab. - Super Tag Editor v2.02. http://www2s.wisnet.ne.jp/~mercury