논문 2014-2-2

윈도우 실행파일을 위한 모듈 기반 유사도 측정 시스템

김예솔*, 조상욱*, 박성수**, 채동규***, 조성제*, 한환수**, 김상욱***, 한상철****

A Module-based System for Measuring Similarity of Windows Executable Files

Yesol Kim*, Sangwook Cho*, Seongasoo Park**, Dong-Kyu Chae***, Seong-je Cho*, Hwansoo Han**, Sangwook Kim***, Sangchul Han****

요 약

본 논문에서는 마이크로소프트 윈도우 실행 프로그램들 간의 유사도를 측정하여 비교하는 효과적인 시스템을 제안한다. 웹 인터페이스 상에서 사용자가 비교하고자 하는 두 개의 실행파일을 입력하면, 시스템은 두 실행파일 간의 유사도를 측정하여 보여준다. 현재, 네 개의 유사도 비교 알고리즘(모듈)을 구현하였으며, 사용자는 각 알고리즘별로 유사도 측정 결과를 볼 수 있다. 또한, 네 개의 알고리즘들을 통합하여 유사도를 비교할 수도 있는데, 이 경우 각 알고리즘마다 가중치를 설정할 수 있다. 각 유사도 비교 알고리즘들을 모듈화하여 구현하였기 때문에, 알고리즘의 개선이나 새로운 알고리즘의 추가가 용이하다.

Abstract

This paper presents an effective system to measure and compare similarity between Microsoft Windows executable files. When a user inputs two target binary programs in the web interface, the system computes the similarity between them and shows the results. We currently implement four algorithms (modules) for measuring similarity of Windows executable files, and a user can obtain the similarity computed using each algorithm. Our system also provides an integrated similarity measure that quantifies the similarity between two files using the four algorithms at the same time. For the integrated similarity measure, a user can assign a weight for each algorithm. Since we develop each algorithm as an extensible module, it is easy to improve an algorithm or add a new one.

한글키워드 : 소프트웨어 유사도, 윈도우 실행파일, 모듈, 가중치

※ 이 논문은 2014년 한국소프트웨어감정평가학회 추계학술발표대회에서 발표된 '윈도우 바이너리 실행파일을 위한 통합 유사도 비교 시스템' 논문을 확장한 것임.

* 단국대학교 컴퓨터학과
** 성균관대학교 정보통신대학
*** 한양대학교 컴퓨터소프트웨어학과
**** 건국대학교 컴퓨터공학과
† 교신저자 : 조성제(Email: sjcho@dankook.ac.kr)

* 본 연구는 문화체육관광부 및 한국저작권위원회의 2014년도 저작권기술개발사업의 연구결과 및 미래창조과학부 및 한국인터넷진흥원의 "2014년도 고용계약형 정보보호 석사과정 사업"의 연구 결과로 수행되었음 (과제번호 H2101-14-1001)
접수일자 : 2014.12.29 수정완료 : 2014.12.30

1. 서 론

소프트웨어 산업이 발전하면서 소프트웨어의 불법복제 및 도용으로 인해 발생하는 피해 규모가 갈수록 증가하고 있다. 이에 따른 저작권 침해 피해를 막기 위해 소프트웨어의 유사도를 비교하는 방법에 대한 연구가 활발히 이루어지고 있으며 크게 소스코드 기반의 유사도 비교법과 바이너리 기반의 유사도 비교법이 있다.

소스코드 기반의 유사도 비교법은 바이너리 기반의 유사도 비교법에 비해 판단이 쉽고 명확하게 유사한 부분이 드러난다는 장점이 있지만 기업들은 소프트웨어의 소스코드를 제공하려 하지 않기 때문에 이를 통한 유사도 비교로 저작권 침해 여부를 가늠하기가 어렵다.

반면 바이너리 기반의 유사도 비교법은 강인하고 신뢰성이 있는 특징정보를 사용할 경우 소스코드 기반의 유사도 비교법에 준하는 비교 정확도를 보일 수 있으며, 난독화 등을 통해 소스코드를 분석하기 어렵게 만든 프로그램도 힘들이지 않고 분석할 수 있는 장점이 있다. 그러나 컴파일러 최적화나 이기종 컴파일러 등을 고려할 때, 강인하고 신뢰성이 있는 특징정보를 추출하는 것이 어렵다.

위에서 언급한 바와 같이 바이너리 기반의 유사도 비교 기법의 성능은 특징정보가 좌우하며, 다양한 특징정보에 대한 연구가 진행되고 있다. 각 특징정보는 고유의 장단점이 존재하기 때문에 같은 비교 대상에 대하여 특징정보에 따른 유사도가 다를 수 있다.

본 논문에서는 다양한 바이너리 코드 유사도 비교 기법을 통합한 유사도 비교 시스템을 제시한다. 이 시스템은 사용할 유사도 비교 기법에 대한 선택을 제공하여 여러 기법 간의 비교가 가능하며, 웹서비스를 이용하여 높은 접근성을 제공한다. 그리고 여러 유사도 비교 기법을 선택하고 각 기법마다 가중치를 적용하여 유사도를 측정할 수 있게 함으로써 신뢰성을 증가시켰다.

본 논문의 구성은 다음과 같다. 2장에서는 통합 유사도 비교 시스템에서 사용한 유사도 비교 기법에 대해 살펴본다. 3장에서는 통합 유사도 비교 시스템에 대해 설명하고, 4장에서는 실험 및 실험 결과를 통해 본 논문에서 제시하는 비교 시스템을 평가한다. 마지막으로 5장에서 결론 및 향후 연구에 대한 내용을 정리한다.

2. 관련 연구

2.1 API 빈도 기반 유사도 비교 기법 [3]

마이크로소프트 윈도우 PE (Portable Executable) 파일 포맷 내 IAT (Import Address Table)에서 API (Application Programming Interface) 리스트를 추출한다. API 리스트를 이용하여 코드영역을 분석해 API의 빈도를 추출하여 cosine 유사도를 계산하여 프로그램 간의 유사도를 산출한다.

API 빈도 기반 유사도 비교 방법은 API 시퀀스 기반 비교 방법보다 코드 난독화(Obfuscation)나 패킹에 대해 강인하다. 실험을 통해 의미를 유지한 변형 (sematics-preserving transformation)에 대해 강인함을 확인하였다.

2.2 문자열 기반 PE 유사도 비교 기법 [4]

PE 포맷을 분석하여 rdata 섹션과 data 섹션에서 문자열을 추출하여 Jaccard나 n-gram 등을 이용하여 프로그램 간 유사도를 측정하였다.

실험을 통해 몇몇 에러 메시지나 컴파일러 의존적인 문자열이 포함 되는 경우를 제외하고 효율적으로 유사도 비교에 사용될 수 있음을 확인하였다.

2.3 Static Major-Path Birthmark 기법 [5]

바이너리 정적 분석을 이용하여 주요 경로의 API 시퀀스를 버스마크로 사용한다. 추출한 API 시퀀스는 서열정렬 알고리즘인 Smith Waterman 알고리즘을 이용하여 비교한다.

신뢰성을 위해 프로그램의 여러 버전과 강인성을 위해 오픈소스를 다양한 옵션으로 컴파일하여 비교하였다.

2.4 정적 API 호출 빈도 기반 유사도 비교 기법[6]

바이너리에서 디스어셈블과정을 통하여 콜그래프를 추출한다. 추출한 콜그래프에서 API 빈도를 계산한 뒤 IDF (Inverse Document Frequency)를 이용하여 가중치를 적용한다. 가중치를 적용한 API 호출 빈도를 버스마크로 사용하여 유사도를 비교한다.

원본 프로그램의 하위버전을 비교하는 실험을 통해 가중치가 적용된 기법의 정확성과 효율성을 검증하였다.

2.5 소스코드 기반 유사도 비교 기법 [7]

MOSS (Measure Of Software Similarity)는 소스코드를 기반으로 유사도를 비교하는 대표적인 도구로, 비교 대상 소스코드를 입력하면 웹페이지를 통해 유사도를 보여준다. 실행파일 기반 유사도 비교 기법을 검증하기 위한 비교 기준으로 사용될 수 있다.

3. 통합 유사도 비교 시스템

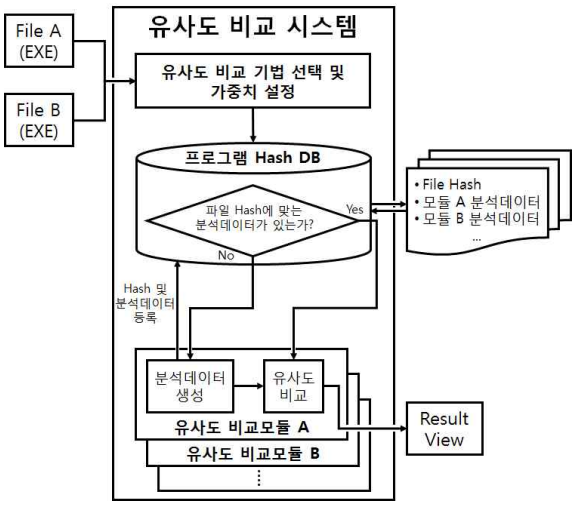
3.1 시스템 개요

본 시스템은 여러 유사도 비교 기법들을 하나로 통합하여 한 번에 비교할 수 있도록 만들어졌다. 또한 새로운 유사도 비교 기법이 추가되거나 기존의 기법이 강화 또는 수정되었을 때 전체 시스템을 수정하지 않고도 추가, 변경할 수 있도록 각 기법들을 모듈화 하였다.

또한 여러 유사도 비교 기법을 선택하여 각 비교 기법마다 가중치를 적용할 수 있게 함으로써 유사도 측정의 신뢰도를 높였으며, 분석데이터를 Hash와 함께 DB에 저장하여 같은 파일이 여러 번 분석과정을 거치는 것을 방지하여 시스템의 부하를 줄였다.

시스템 구축에 사용된 서버 제원은 Intel(R) Core(TM) i5-4670 CPU, RAM 8GB, Window 7 64bit를 사용하였다. 웹서버는 Apache Tomcat 7.0, DB는 MySQL 5.6.21을 사용하였다.

[그림 1] 가중치 적용 통합 유사도 비교 시스템 구조도



[그림 2] 가중치 적용 통합 유사도 비교 시스템 구조도

3.2 시스템 구조 및 구현

본 시스템에 접속하게 되면 [그림 1]과 같이 대상파일 입력과 유사도 비교 기법 선택할 수 있는 페이지가 로딩된다. 대상 프로그램을 선택하고, 원하는 유사도 비교 기법들을 선택하고 각 비교 기법마다 가중치를 입력한 후 전송한 후 유사도 비교를 진행하게 된다. 각 모듈별 유사도 비교 수행이 완료되면 모듈마다 설정된 가중치 값으로 배율이 적용되고 이들의 합으로 최종 유사도를 산출하여 웹페이지로 결과를 보여주게 된다.

3.2.1 가중치 적용

기존에 구현하였던 기법 외에 다른 유사도 비교기법을 추가하였다. 사용자는 각 비교 기법마다 가중치를 적용하여 대상 프로그램의 특성에 맞는 유사도 비교를 할 수 있다. 또한 적합한 가중치를 적용하여 정확도와 신뢰성을 높일 수 있다.

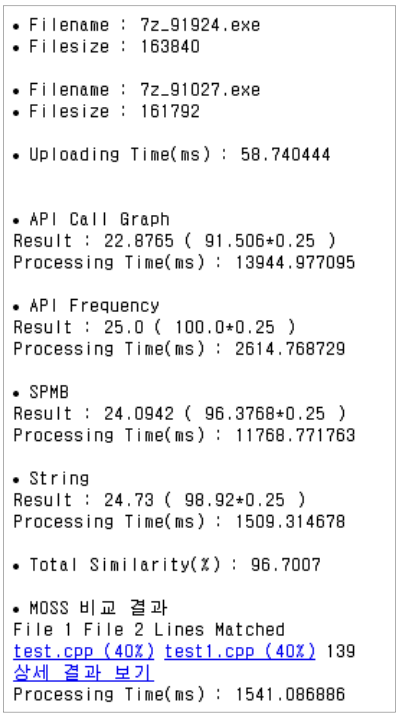
3.2.2 Hash값 기반 분석결과 DB 관리

실행파일의 비교를 위해 서버에 파일을 올려 비교기법을 수행하게 되면 매번 비교에 필요한 데이터를 추출하는 시간으로 인해 타임 오버헤드가 발생한다.

이를 줄이기 위해 서버에서는 수신 받은 파일의 md5 해시 값을 계산하여 프로그램 DB를 조회한다. DB상에 해시 값이 존재할 경우 이미 분석되어있는 데이터를 사용하여 유사도 비교를 진행하고, 해시 값이 존재하지 않을 경우 비교모듈에서 필요한 분석데이터를 생성한 후 DB에 해시 값을 등록한다.

3.2.3 비교 기준 (MOSS)

비교 대상의 소스코드가 존재할 경우 MOSS의 결과와 비교할 수 있도록 구현하였다. 간단하게 유사도 결과와 세부 결과를 볼 수 있는 링크를 연결하였다.



[그림 3] 모듈 기반 유사도 비교 시스템 결과화면

4. 실험

본 시스템을 실행한 결과는 [그림 3]에서 확인할 수 있다. 전체 모듈에 대해 각각 가중치를 적용하여 통합된 결과를 보여주고, MOSS를 이용하여 소스코드에 대한 유사도를 비교할 수 있도록 하였다.

본 시스템의 DB hash저장에 대한 효율성을 검증하기 위해 10개의 프로그램을 대상으로 각각 두 번 씩 프로그램을 비교하는데 걸리는 시간을 측정하였다. 첫 번째 등록 시에는 분석시간이 그대로 소요되고, 두 번째 등록 시에는 DB에 hash가 등록되어있기 때문에 이미 분석된 파일을 이용한다.

[표 1] 신규 및 해시값 존재 시 처리 시간 비교

프로그램	버전		신규	hash 존재	비교
7zip	910	921	9217.62	2629.41	0.285259
AIMP3	3.10	3.20	21375.20	5695.00	0.266430
coreftp	13	21	69042.31	16681.17	0.241608
editplus	3.20	3.70	66597.93	18139.05	0.272367
FileZilla	3.3.2	3.4.0	351823.77	73639.13	0.209307
FTPbox	2.5.3.1	2.5.4.1	2451.77	847.42	0.345636
mplayer	170	176	404732.17	75439.63	0.186394
notepad++	6.1.5	6.3	42810.86	11657.15	0.272294
vlc	2.0.5	2.1.5	3076.33	1415.80	0.460224
winamp	521	555	23604.24	7065.85	0.299347

프로그램 크기에 따라 모듈 처리 시간이 상이하지만, 신규 프로그램의 경우 대비 hash가 존재할 경우 처리 시간이 1/2~1/5 수준으로 오버헤드가 감소함을 확인할 수 있었다.

5. 결 론

본 시스템은 기존에 구현되었던 모듈 별 유사도 제공에서 여러 개 모듈의 가중치를 적용하여 통합적인 유사도 비교 결과를 보여줄 수 있도록 기능을 수정하였다. 효율성을 위해 프로그램 업로드 시 hash를 비교하여 생성된 분석 데이터를 사용하도록 DB를 추가하였다. 또한 신뢰성을 위한 비교지표로 소스코드 기반의 유사도 비교방법(MOSS)을 추가하여 비교 가능하도록 구현하였다.

하지만 본 시스템에 구현되어 있는 유사도 비교 기법들은 모두 각 기법에서 사용되는 비교 기준에 의해 측정한 유사도만 제공하므로 표절 등의 근거로 사용되기 힘들다. 또한 가중치를 임의로 적용하기 때문에 이를 보완하기 위해 각 모듈의 가중치 적용 결과 패턴을 연구하여 적절한 가중치를 자동으로 제안하는 기능을 연구 중이다.

참 고 문 헌

[1] A System for Detecting Software Plagiarism: MOSS (a Measure Of Software Similarity), http://theory.stanford.edu/~aiken/moss/

[2] Ginger Marie Myles, Christian S. Collberg, “Software theft detection through program identification”, University of Arizona, Tucson, AZ, 2006

[3] Haruaki Tamada, Masahide Nakamura, Akito Monden, Ken-ichi Matsumoto, “Java Birthmarks - Detecting the Software Theft -”, IEICE Transactions on Information and Systems, 88-D(9) 2148-2158, 2005.09

[3] Dongjin Kim, Yongman Han, Seong-je Cho, Haeyoung Yoo, Jinwoon Woo, Yunmook Nah, Minkyu Park, Lawrence

Chung, "Measuring similarity of windows applications using static and dynamic birthmarks", Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp.1628-1633, 2013

[4] Yesol Kim, Jeongoh Moon, Dongjin Kim, Younsik Jeong, Seong-je Cho, Minkyu Park, Sangchul Han, "A Static Birthmark of Windows Binary Executables Based on Strings", Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013

[5] 박성수, 한환수, "정적 주요 경로 API 시퀀스를 이용한 소프트웨어 유사성 검사", 정보과학회논문지, 2014

[6] Dong-Kyu Chae, Jiwoon Ha, Sang-Chul Lee, Sang-Wook Kim, Gyun Woo, "Software Plagiarism Detection via the Static API Call Frequency Birthmark", ACM Symposium on Applied Computing(SAC), 2013

[7] A system for detecting software plagiarism - MOSS. [online] http://theory.stanford.edu/ ~aiken/moss/.

저 자 소 개

김 예 솔

2014년 단국대학교 컴퓨터학과(공학사)
2014 - 현재 단국대학교 소프트웨어보안 석사 과정
<주관심분야 : 소프트웨어 유사도, 소프트웨어보안, 시스템 보안>

조 상 욱

2014년 단국대학교 컴퓨터학과(공학사)
2014 - 현재 단국대학교 소프트웨어보안 석사 과정
<주관심분야 : 운영체제, 시스템보안>

박 성 수

2011년 한국산업기술대학교 컴퓨터공학(학사)
2013년 성균관대학교 전자전기컴퓨터공학(공학석사)
2013년 - 현재 성균관대학교 전자전기컴퓨터공학 박사과정
<주관심분야 : 컴파일러 및 최적화, 소프트웨어 유사도, 프로그램 분석>

채 동 규

2012년 한양대학교 컴퓨터전공(공학사)
2012년 - 현재 한양대학교 컴퓨터소프트웨어학과 석박통합과정
<주관심분야 : 데이터마이닝, 소프트웨어>

저 자 소 개

조 성 제

1989년 서울대학교 컴퓨터공학과(공학사)
1991년 서울대학교 컴퓨터공학과(공학석사)
1996년 서울대학교 컴퓨터공학과(공학박사)
2001년 미국 University of Califonia, Irvine 객원 연구원
2009년 미국 University of Califonia 객원연구원
1997년 3월 - 현재 단국대학교 소프트웨어학과/컴퓨터학과 교수
<주관심분야 : 컴퓨터보안 (취약점 탐지 및 분석, 악성코드 분석), 스마트폰 보안, 소프트웨어 지적재산권 보호, 소프트웨어 보증>

한 상 철

1998년 연세대학교 컴퓨터과학과(공학사)
2000년 서울대학교 컴퓨터공학과(공학석사)
2007년 서울대학교 컴퓨터공학과(공학박사)
2008 - 현재 건국대학교 컴퓨터공학과 교수
<주관심분야 : 실시간 시스템, 스케쥴링 알고리즘>

한 환 수

1993년 서울대학교 컴퓨터공학과(공학사)
1995년 서울대학교 컴퓨터공학과(공학석사)
2001년 미국 University of Maryland 전산학(박사)
2001년 - 2002년 Intel 책임연구원
2003년 - 2008년 KAIST 전산학과 교수
2008년 - 현재 성균관대학교 정보통신대학 교수
<주관심분야 : 컴파일러 및 아키텍쳐, 고성능 컴퓨팅>

김 상 욱

1989년 서울대학교 컴퓨터공학과(공학사)
1991년 KAIST 전산학과(석사)
1994년 KAIST 전산학과(박사)
1997년 - 1991년 Stanford University, Computer Science Department, 방문 연구원
1994년 - 1995년 KAIST 정보전자 연구소 전문 연구원
1995년 - 2000년 IBM T.J. Watson Research Center, Post-Doc.
1995년 - 2003년 강원대학교 정보통신공학과 부교수
2003년 - 현재 한양대학교 정보통신대학 정보통신학부 교수
2009년 - 2010년 Carnegie Mellon University, Visiting Scholar
<주관심분야 : 데이터베이스 시스템, 저장시스템, 트랜잭션 관리, 데이터 마이닝, 멀티미디어 정보 검색, 공간 데이터 베이스/GIS, 주기억장치 데이터베이스, 이동 객체 데이터베이스/텔레매틱스, 사회 연결망 분석, 웹데이터 분석>