논문 2012-2-3

윈도우용 이진실행 프로그램 식별을 위한 정적 버스마크

최종천*, 한용만**, 조성제***, 유해영***

A Static Birthmark for Identifying Windows Binary Executables

Jongcheon Choi*, Yongman Han**, Seong-je Cho***, Haeyoung Yoo***

요 약

애플이나 구글 등의 영향으로 소프트웨어 산업에 대한 관심이 커지고 있지만, 국내 소프트웨어 산업은 아직 선진국의 수준에 미치지 못하고 있다. 또한, 계속되는 소프트웨어 불법복제로 인한 경제적인 피해가 만연되어 있다. 특히, 웹하드(또는 Online Service Provider)와 같은 파일 공유사이트를 통해 소프트웨어 불법 유통이 자유롭게 이루어지는 것이 국내 소프트웨어 산업 발전을 방해하는 주요 원인으로 지적되고 있다. 현재 불법 유통되는 대부분의 상용 소프트웨어는 마이크로소프트사의 윈도우 기반의 이진 실행파일인 경우가 많다. 이에 본 연구에서는 윈도우에서 실행되는 실행 파일들을 대상으로 정적 버스마크를 추출하여, 이 버스마크 기반으로 윈도우 소프트웨어를 식별하여 불법 유통을 탐지하는 방안을 제안한다.

한글키워드 : 이진실행 프로그램, 정적 버스마크

1. 서 론

소프트웨어(Software, SW)가 자동차·조선·항공·정보가전·의료 등의 산업제품에 탑재되고, 금융·전자정부 등의 서비스 실현에 기여하면서 SW 산업의 범위가 더욱 확대되고 있다. 이처럼 산업과 기술이 융합되고 SW 산업의 서비스화 추세가 맞물리면서 SW 적용 범위는 지속적으로 확대될 전망이다. SW관련 산업 범위가 확대되고 기존산업·새로운 서비스에 SW가 더 많이 적용되는 것에 반해, 소프트웨어 불법복제와 도용으

* 단국대학교 정보컴퓨터과학과
** 단국대학교 컴퓨터과학과
***단국대학교 소프트웨어학과
교신저자:조성제(email:sjcho@dankook.ac.kr)
접수일자: 2012.11.16 수정완료: 2012.12.22.

※ 본 연구는 문화체육관광부 및 한국저작권위원회의 2012년도 저작권기술개발사업의 연구결과로 수행되었음.

로 발생하는 피해규모는 갈수록 증가되고 있다.

소프트웨어 저작권 보호 단체인 사무용소프트웨어연합(Business Software Alliance, BSA)가 발표한 '2011년도 세계 소프트웨어 불법복제 현황 보고서'에 따르면, 2011년 소프트웨어 불법복제(piracy)로 인한 전 세계 피해규모는 634억 달러로 나타났다[1]. 국내의 경우 같은 해 약 8,900억 원으로 집계되어 불법복제 피해 규모가 역대 최대를 기록하였다. 이러한 피해 규모는 전반적인 불법복제 비율이 과거보다 감소했어도 전체 소프트웨어 산업의 규모가 커져감으로 피해도 함께 늘어난 것으로 국내외의 소프트웨어 불법복제 피해규모는 더욱 늘어날 것으로 전망되고 있다[1].

소프트웨어 불법복제는 비즈니스나 개인사용을 목적으로 소프트웨어 프로그램을 불법 복제·배포 또는 불법 재생산하는 것이다. 전 세계적으로 소프트웨어 불법복제로 인한 피해가 심각하기 때문에, 소프트웨어 개발자들은 자신의 프로그램을 불법복제로부터 보호하기 위해 SW 워터마킹(Watermarking)/핑거프린팅(Fingerprinting), 암호화, 패킹(Packing), 안티-디버깅(Anti-debugging), 난독화(Obfuscation) 등 많은 노력을 기울이고 있다.

소프트웨어의 불법복제 및 유통은 많은 재산상의 피해만이 아니라 소프트웨어 개발자들의 개발의욕과 진취적인 발전을 가로막는 주요한 사회적 요소가 되고 있다. 또한, 최근 소프트웨어만이 아닌 불법 저작물을 유통시키는 것에 대한 형사처벌 강화로 따라 소비자 개개인의 무지로 발생한 정신적, 물질적 고통이 증가되고 있다. 이러한 점에서 불법 유통되는 소프트웨어를 중간과정에서 탐지하여 소비자에게 저작권에 인식을 강화시키고 소프트웨어 산업을 보호하는 효과를 얻을 수 있다.

기존의 소프트웨어 불법복제 기법은 개발자가 스스로 불법복제를 입증하기 위한 기법을 적용하고 확인할 수 있는 도구를 제공해야 하는 불편함이 존재하였다. 본 논문에서는 날로 다양해지는 유통경로와 불법유통을 탐지해야하는 대상 소프트웨어가 증가되는 상황에서 소프트웨어가 개발된 당시에 가지고 있는 고유한 정보를 활용하여 소프트웨어를 식별하여 불법 유통을 탐지하는 방법을 제시한다. 즉, 마이크로소프트사의 윈도우 환경에서, SW마다 고유한 정보인 정적 버스마크(Static birthmark) 기반으로 바이너리 코드로 유통되는 애플리케이션들을 식별하여 불법 유통을 탐지하는 기법을 보인다.

이러한 기법은 개발사의 별도의 노력이 없이 소프트웨어 탐지를 가능하게 하며 유통경로 상에서 추가적인 도구나 비용 없이 탐지 및 차단에 대한 구성을 가능하게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대한 내용을 살펴보고 3장에서는 IAT에서 추출한 정적 버스마크(Static birthmark)를 기반으로 소프트웨어 탐지기법에 대해 제안하고 4장에서는 결론을 정리하였다.

2. 관련 연구

2.1 윈도우 실행코드

마이크로소프트 윈도우 운영체제는 실행 가능한 파일형식으로 PE (Portable Executable) 파일 포맷을 지원하는데, 이는 윈도우 운영체제 로더가 실행 코드를 관리하는데 필요한 정보를 정의한 데이터 구조이다. PE파일 포맷은 링크를 위한 DLL (Dynamic-link library)과 API (Application programming interface)의 Import 및 Export 테이블 등을 포함하고 있으며 프로그램이 실행되면 운영체제 로더가 이를 분석하여 필요한 DLL을

메모리에 함께 로드하게 된다. 이를 위해 PE 파일 포맷에는 DLL에서 사용될 API의 이름과 실행되면서 할당되는 주소가 저장되는 테이블을 가지고 있는데, 이러한 테이블을 IAT (Import Address Table)이라고 한다. 바이너리 실행파일 자체에서 추출하는 방법은 PE 파일 분석을 통해 IAT를 찾고 IAT로부터 포함된 API 리스트를 추출할 수 있다.

또한, 추출된 API 리스트와 할당된 주소를 이용하여 바이너리 실행파일의 코드영역을 디스어셈블(Disassemble)하여 실제 해당 API가 호출된 위치와 횟수를 확인할 수 있다.

2.2 소프트웨어 버스마크

소프트웨어 버스마크(Birthmark)는 소프트웨어가 개발된 상태에서 가지는 고유한 정보를 나타내는 것으로 소프트웨어 도용이나 복제에 대한 많은 연구들이 진행되어 왔으며 최근 윈도우에서 소스코드 없는 바이너리코드에 대해 소프트웨어 유사성을 탐지하는 기법으로 바이너리 코드에서 사용되는 API를 고유정보(Feature)로 하는 버스마크 추출 방법에 대한 다양한 연구가 진행되고 있다.

API를 고유정보로 활용하여 정적 또는 동적분석을 통한 유사성 비교에 대한 연구가 다양하게 진행되고 있다 [2,3,4,5].

이러한 API분석은 컴파일과정에서 변화되는 변수나 사용자 작성 함수정보와 달리 고유정보를 유지하고 있어 정적, 동적 분석 모두에 활용 가능한 정보이며 API가 프로그램의 시퀀스 상에 나타나는 고유정보라는 점을 활용한 것으로 복잡하고 성능이 떨어지는 Call-graph를 대체할 수 있는 방안으로 연구되고 있다.

기존의 버스마크에 대한 연구는 리버싱이 용이한 자바를 중심으로 연구가 진행되어 왔으며 최근 윈도우 바이너리 코드에 대한 연구가 활성화 되고 있는 추세이다. 기존의 버스마크는 주로 실행코드에 포함되어 있는 문자열에 대한 분석[10]이나 API 순차(Sequence) [11,12]에 초점이 맞추어 있었기 때문에 문자열의 경우 위변조가 쉽고 API 순차는 추출하기 위한 노력이 많이 필요하고 빠른 비교가 쉽지 않다는 단점이 존재하였다.

2.3 해시 기법

해시(Hash) 기법은 입력 값에 대응하는 단방향 압축 값을 생성하는 기법으로 암호화나 데이터베이스의 킷값을 생성하거나 무결성(Integrity)을 확인하는 목적으로 많이 사용된다. 해시함수(Hash function)는 일반적으로 압축성(입력에 대한 고정된 길이의 출력값)과 계산의 용이성(함수와 입력값에 대해 결과값을 계산하기 쉬움)을 가지고 있다[11].

해시 기법은 MD4, MD5, SHA1, SHA256, SHA512와 같은 메시지만을 이용한 기법과 HMAC-MD5, HMAC-SHA256, CBC-MAC과 같은 메시지와 킷값을 사용한 해시기법으로 구분되어 진다. 일반적으로 오픈소스로 제작된 소프트웨어 배포에서 소프트웨어 템퍼링(tampering, 변형, 변경, 또는 변조)을 방지할 목적으로 제공되고 있으며 <표 1>은 파일질라(FileZilla)에서 제공하는 해시값에 대한 예를 보여주고 있다.

<표 1> FileZilla에서 제공하는 파일정보

	FileZilla 3.5.3 Client
Size	4518720 bytes
SHA-512	0646c36997cbf57adb79cbc22dc4d0786a8db2a5aa5d9ae271c6e697170ffd659885f5e6bb2684be4bd72079646dde6fb797a5c9d1fed957f2c34d18052690f7

이러한 해시기법은 소프트웨어 도용에 대한 탐지에 활용될 수 있으나 중복 값이 발생할 가능성이 존재하고 개발자가 올려둔 정보를 별도 수집하는 방안이 제공되어져야 하는 단점이 있다.

3. IAT기반 SW 탐지

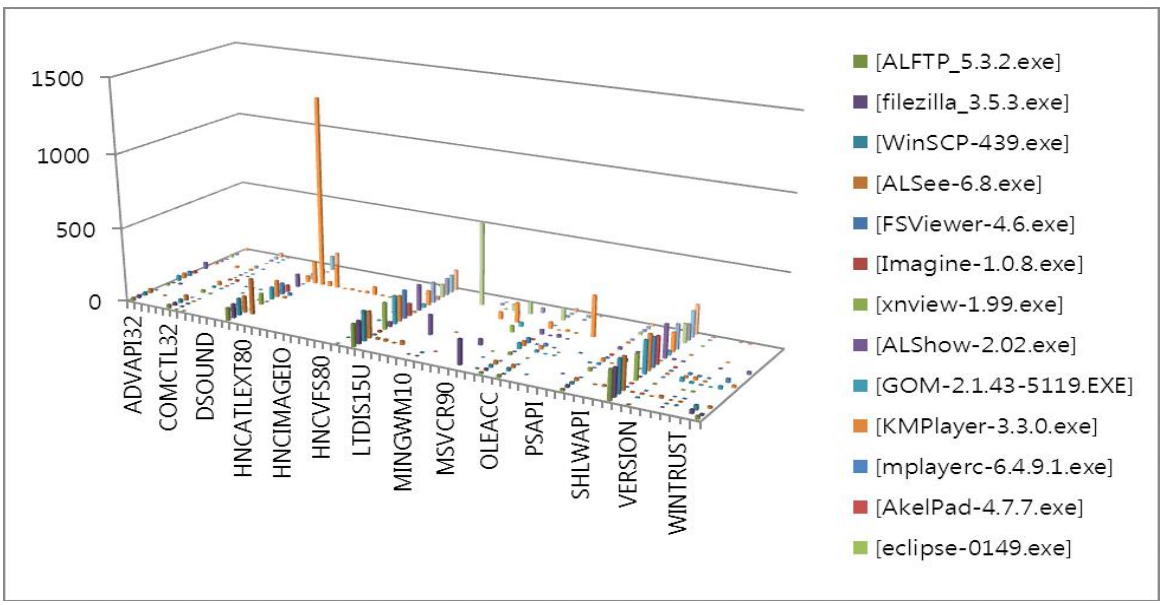
소프트웨어 워터마크는 기본적으로 새로운 형태의 데이터(코드)가 삽입되어야 한다. 이러한 방식은 개발자에 의해 삽입이 되던가 기존에 개발된 코드에 대한 변형이 불가피하다는 문제점을 가지고 있다.

따라서 본 논문에서는 윈도우기반의 소프트웨어에서 개발자의 도움 없이 SW코드가 가지고 있는 고유정보만을 추출하여 소프트웨어 탐지가 가능한 기법으로 버스마크 기법을 활용하여 웹하드를 통해 불법적으로 유통되는 소프트웨어 탐지에 활용할 수 있는 방안을 제안한다.

3.1 IAT기반 실행코드 분석

본 연구는 소스코드가 없는 마이크로 윈도우 바이너리 코드를 대상으로 정적 분석 기법을 활용하여 기존의 연구에서 다루어지지 않았던 웹하드(또는 불건전 OSP 등의 불법 저작물 유통 인터넷 사이트) 환경에서 유통되는 소프트웨어에 대해 API Frequency (Call count)를 이용하여 탐지하는 기법을 제안하며 이러한 기법은 동적분석이 어려운 소프트웨어 유통환경에서 쉽고 빠르게 적용 가능한 기법으로 유용하게 활용될 것이다.

본 연구에서 제안하는 버스마크는 개발자의 도움 없이 생성이 가능한 정보이므로 소프트웨어 저작권 분쟁에서 중앙집권적인 대처가 수월하다



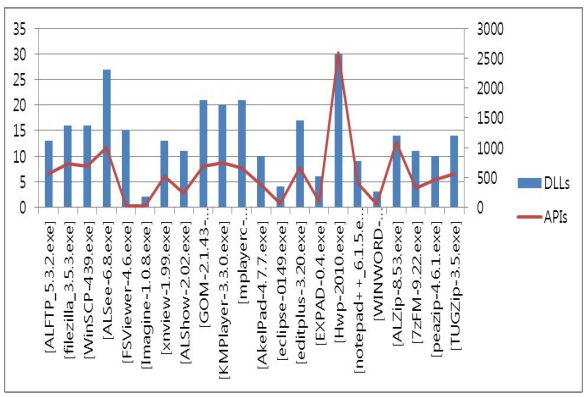
[그림 1] IAT기반 DLL 및 API 분석

는 장점도 가지고 있다.

IAT기반으로 소프트웨어 고유정보를 분석한 결과 [그림 1]에서 나타난 것과 같이 각각의 소프트웨어가 가지고 있는 DLL과 API정보가 상이하게 나타나는 것을 확인할 수 있다.

이러한 점은 IAT기반의 버스마크가 소프트웨어의 고유정보로 활용이 가능하다는 점을 보여주며 본 실험에서는 IAT를 기반으로 확보할 수 있는 정보(DLL 개수, DLL 명칭, DLL에 포함된 API 개수, API 명칭)와 IAT를 기반으로 추출한 API정보를 기반으로 실행코드에서 API가 호출된 횟수를 버스마크 정보로 활용하여 실험을 진행한 결과 중복 없이 대상 소프트웨어를 탐지하는 것이 가능했다.

본 연구는 이러한 기법을 이용하여 각 소프트웨어가 가지고 있는 DLL과 API 개수를 Imports table을 기반으로 분석한 결과 [그림 2]와 같은 형태로 나타났다. DLL의 개수와 참조된 API의 개수가 상호 연관성을 가지는 것을 보여준다.

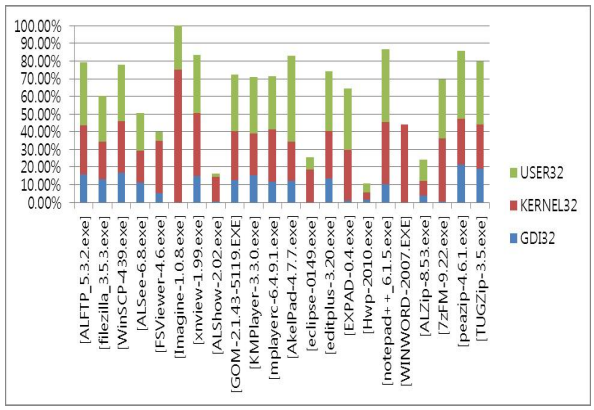


[그림 2] 실험대상 프로그램 DLL과 API개수 비교

분석된 내용을 보면 [그림 3]과 같이 다수의 프로그램들이 USER32.DLL, KERNEL32.DLL, GDI32.DLL을 높은 비율로 포함하고 있다. 특히 ALShow와 Eclipse, HWP, 그리고 ALZip 이 네 가지를 제외한 나머지 18개의 프로그램들은 3개의 DLL이 전체 API의 40% 이상 차지하고 있다. ALShow와 HWP와 ALZip의 경우 상용프로그램으로 회사 자체로 사용하는 DLL에서 참조하는 API의 비율이 높게 구성하고 있어 차지하는 비율이 낮게 나왔다. 또한, Eclipse의 경우는 오픈소스이면서 자체로 갖고 있는 API의 전체 개수가 59개로 다른 프로그램에 비해서 API숫자가 적고 자체 API의 비율이 높아 그래프상 낮은 비율을 보이고 있다. 한편, Imagine프로그램은 USER32.DLL과 KENRNEL32로만 구성되어 있어 100%의 비중을 보이고 있다. 이러한 점에 기반하여 일반적인 윈도우 프로그램에서 3개의 DLL이 자주 많이 사용된다는 것을 추정할 수 있다.

이 3개의 DLL은 각각 아래 [표 2]와 같은 역할을 하고 있다.

따라서 KERNEL32.dll, USER32.dll, GDI32.dll은 여러 윈도우 프로그램에서 자주 호출되어 특징분석에 대한 가중치(weight)가 낮다고 볼 수 있다. 이러한 점을 감안하여 본 논문에서 제안하는 소프트웨어 필터링 기법은 위 3개의 DLL에 대해 특징정보 비교 시 후위에 배치하는 방법을 제안하였다.



[그림 3] 주요 DLL이 차지하는 비율

<표 2> 주요 DLL 설명

DLL	Description	API
Kernel32.dll	메모리관리, 파일 입출력 등 윈도우 커널이 제공하는 모든 작업처리	LoadLibrary, GetCurrentProcess, ExitProcess, TerminateProcess, etc
User32.dll	모든 윈도우 로직을 조작하기 위한 사용자 인터페이스 제공	MessageBoxA, CreateWindowExA, SetCapture, SendMessageA, etc
GDI32.dll	그래픽 장치 인터페이스 화면 및 프린터에 텍스트와 그래픽 출력	Bitblt, TextOutA, SetROP2, LineTo, SetTextColor, etc

3.2 버스마크기반 탐지기법

버스마크(Birthmark) 또는 특징정보(Feature)는 프로그램 생성 시 가지는 고유정보로 소스코드가 바이너리 실행코드로 변환될 때 변수명, 함수명과 같은 프로그램의 특성을 나타내는 식별 가능한 많은 정보들이 변환되어 바이너리 실행파일만 주어졌을 경우 해당 프로그램에 대한 식별이 어렵게 된다. 이러한 점 때문에 많은 연구에서 바이너리로 변환되어도 변하지 않는 IAT를 기반으로 API에 대한 정보를 활용한 버스마크 추출에 대한 연구를 진행하고 있다.

이러한 연구들은 주로 소프트웨어 도용이나 포렌식에 이용되고 있으며 본 연구는 소프트웨어의 불법 유통이나 사용을 감지하기 위한 소프트웨어 탐지에 사용하고자 한다.

따라서 본 연구에서는 윈도우 실행파일이 가지고 있는 IAT기반의 DLL과 API정보와 해당 API가 실제 코드에서 호출되는 횟수를 기반으로 하는 Birthmark(IATB – Import Address Table Birthmark)를 추출 하여 비교의 효율성을 고려하여 다음의 4단계의 비교를 통해 비교대상의 수를 늘려가는 방식의 소프트웨어 감지기법을 제안한다.

Step 1. DLL 개수 비교

테이블에 적재된 DLL의 개수를 비교하고 동일한 개수의 DLL이 존재하는 경우만 다음 단계로 진입하고 적재된 DLL의 개수가 다를 경우 비교대상에서 제외한다.

Step 2. DLL 이름 비교

동일한 개수의 DLL이 적재되어 있는 경우 DLL의 이름을 하나씩 비교하여 모든 개수의 이름이 동일한 경우 다음 단계로 진입한다. 이 때 각 DLL이 참조하고 있는 API의 개수를 함께 비교하여 필터링 기능을 강화하는 효과가 있다.

Step 3. 사용 빈도가 높은 DLL을 제외한 API 비교

DLL 리스트까지 비교하여 추출된 리스트를 대상으로 개별 DLL에서 참조하고 있는 API 명칭과 해당 API가 코드에서 호출된 횟수를 비교하여 동일한 값을 가지고 있는 대상을 추출한다. 이때 IAT 분석으로 사용빈도가 높다고 판단된 KERNEL32.dll, USER32.dll, GDI32.dll은 비교 대상에서 제외하였다.

정의 1. 사용빈도 높은 DLL

KERNEL32.dll, USER32.dll, GDI32.dll은 실험대상 평균 62.2%의 높은 비율로 사용되고 있고 거의 대부분의 프로그램에서 사용되고 있는 사용빈도가 높은 DLL들이다. 이러한 점을 감안하여 해당 DLL에 대한 가중치는 낮게 두어 1차 비교에서 제외하고 비교한 결과를 통해 우선적인 동일 소프트웨어 여부를 판단한다.

Step 4. 사용 빈도 높은 DLL의 API 비교

3단계까지 비교에서도 하나의 소프트웨어가 검출되지 않을 경우 남아있는 빈도수 높은 DLL이 참조하고 있는 API의 명칭과 호출 횟수를 비교하여 동일 소프트웨어 여부를 판단하였다.

예외 1. 적재된 DLL개수가 제외대상 미만인 경우

이 경우는 사용된 API의 총량이 적다고 판단하였고 제외 대상의 수가 더 많기 때문에 단계 3,4를 동시에 수행하였다.

예외 2. 제외 대상 DLL만 존재하는 경우

이 경우는 단계2에서 DLL 명칭과 개수를 비교하기 때문에 미리 계산된 값에 의해 대상 DLL의 존재여부를 파악할 수 있다. 이러한 점을 이용하여 제외 대상 DLL만 존재하는 경우 단계 3을 건너뛰어 바로 단계 4를 수행하게 한다.

4. 결 론

본 논문에서는 소프트웨어 저작권 보호를 위해 정적 버스마크 기반으로 웹하드를 통해 유통되는 불법 소프트웨어에 대한 탐지기법을 제안하였다. 스마트 폰의 등장으로 모바일 환경에서 사용되는 소프트웨어의 저작권 보호에 대한 필요성은 더욱 절실히 요청되고 있으며, 다양한 경로의 불법 유통에 대한 탐지기법에 대한 연구가 진행되어야 한다.

소프트웨어 불법 유통을 탐지하고 유통경로를 탐지하여 손해에 대한 배상과 유출자에 대한 처벌을 가능하게 하는 기술에 대한 연구는 지속적으로 이루어져 왔다. 하지만, 수많은 개발사와 다양한 개발환경에 적합한 탐지기법에 연구는 아직 많이 부족하다.

본 논문에서 제안하는 버스마크기법을 이용한다면 개발사에 의존하지 않고 실행코드만으로 버스마크 추출하여 소프트웨어 탐지가 가능하므로 저작권 보호를 강화하는데 기여하게 될 것이다.

본 논문의 연구결과는 국내 소프트웨어 불법 복제를 줄일 수 있는 불법 유통 탐지기법 연구 및 관련 기술 개발에 일조할 수 있다고 기대한다. 더불어, 향후 다양한 윈도우 애플리케이션들을 대상으로 실험을 확대하여, 본 논문에서 제안한 기법들을 체계적으로 검증하고자 한다.

참 고 문 헌

[1] BSA, "2011 BSA GLOBAL SOFTWARE PIRACY STUDY", 2012. 5. 15. http://www.bsa.org/korea.

[2] 박희완, 한태숙, "소프트웨어 몽타주: 디지털 포렌식 수사를 위한 유사 소프트웨어 탐지 대상의 필터링", 정보과학회논문지: 컴퓨팅의 실제 및 레터, 제16권 제4호, 2010. 04.

[3] 최석우, 조우영, 한태숙, "Windows 프로그램 도용 탐지를 위한 기능 단위 동적 API 버스마크", 정보과학회논문지: 소프트웨어 및 응용, 제36권 제9호, 2009. 09.

[4] Xinran Wang, Yoon-Chan Jhi, Sencun Zhu, and Peng Liu, "Detecting Software Theft via System Call Based Birthmarks", Annual Computer Security Applications Conference (ACSAC '09.), 2009.

[5] Haruaki Tamada, Keiji Okamoto, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto, "Dynamic Software Birthmarks to Detect the Theft of Windows Applications", In Proc. International Symposium on Future Software Technology 2004 (ISFST 2004), October 2004.

[6] Ginger Myles, and Christian Collberg, "Detecting Software Theft via Whole Program Path Birthmarks", Information Security Conference, September, 2004.

[7] David Schuler and Valentin Dallmeier, “Detecting Software Theft with API Call Sequence Sets”, In Proceedings of the 8th Workshop Software Reengineering, May 2006.
[8] Ginger Myles, “Software Theft Detection Through Program Identification”, PhD thesis, Department of Computer Science, The University of Arizona, 2006.
[9] Microsoft, “Microsoft Portable Executable and Common Object File Format Specification”, Revision 8.2, 2010. 09.
[10] http://filezilla-project.org/
[11] 이만영, 현대 암호학 및 응용, 생능, 2002

저 자 소 개

최 종 천

<주관심분야 : >

한 용 만

<주관심분야 : >

조 성 제

1989년 서울대학교 컴퓨터공학과 (공학사)
1991년 서울대학교 컴퓨터공학과 (공학석사)
1996년 서울대학교 컴퓨터공학과 (공학박사)
2001년 미국 University of California, Irvine 객원연구원
2009년 미국 University of Cincinnati 객원연구원
1997년 3월~현재 : 단국대학교 소프트웨어학과 교수

<주관심분야 : 컴퓨터보안, 소프트웨어 보증, 시스템소프트웨어, 실시간스케줄링, 임베디드 소프트웨어 등>

유 해 영

<주관심분야 : >