Journal of Korea Multimedia Society Vol. 12, No. 9, September 2009(pp. 1309-1315)

실행 중인 바이너리 코드 추출 프로그램의 기능 확장 연구

장항배[†], 권혁준[††], 김양훈[†††], 김국보[††††]

요 약

본 연구에서는 일반 프로그램 실행 시에 발생할 수 있는 보안 결함이 의심되는 부분의 바이너리 코드를 추출하는 기술개발을 목표로 한다. 이를 위하여 분석 대상 실행프로그램에서 정규적으로 발생하는 예상 취약점 부분을 토대로 실제 수행하면서 동적 분석 및 조사하면서, 각종 로그 기록을 산출할 수 있는 기술을 개발하였다. 본 연구의 결과는 기업 및 기관의 보안관련 조직의 교육 자료로써 활용할 수 있을 뿐 만 아니라 외부 정보침입자로부터 해킹을 방지 할 수 있게 된다.

The Study on Improvement of the Program that Traces the Binary Codes in Execution

Hangbae Chang[†], Hyukjun Kwon[††], Yanghoon Kim[†††], Gukboh Kim[††††]

ABSTRACT

This research goal of developing and producing a tool that finds security weakness that may happen when a usual program is executed. The analyzing tool for security weakness has the major functions as follows. In case that a part of anticipated security weakness are in execution, it traces a machine language to a part in execution. And Monitoring System calls and DLL(API) calls when a program is in execution. The result of this study will enable to contribute to use as educational materials for security service in companies and related agencies and to prevent from hacking of external information invaders in the final analysis.

Key words: binary code trace(바이너리 코드추출), dissaembler(디스어셈블러), debug interrupt(디버깅 인터럽트), static analysis(정적분석), dynamic analysis(동적분석)

1. 서 론

일반적으로 프로그램에 존재하는 보안취약점은 보안취약성 또는 개발자의 실수에 의하여 발생한다. 보안취약성은 프로그램의 근본적인 문제점으로부터 파생되어 보안 사고를 일으킬 여지가 있는 성질의 것으로서, 해커가 보안취약성을 악용하면, 그것이 보안취약점으로 구체화되어 보안사고의 원인이 된다. 예를 들어, 잘 알려진 버퍼 오버프로우(Buffer Overflow)는 보안 취약성이며, 대부분의 프로그램에서 공통적으로 발생할 수 있는 하나의 property이다. 이러한 버퍼 오버프로우가 하나의 특정 프로그램에서 발견되어 보안문제의 소지가 되면, 이를 보안취약점이라 부른다. 이렇게 개발자 등은 동일한 프로그램을 사용하는 다른 사용자들이 그러한 보안 취약점을 점검(Scan)할 수 있는 수단을 제공하고 있다. 다른

※ 교신저자(Corresponding Author) : 장항배, 주소 : 경기도 포천시 선단동 산11-1(487-711), 전화 : 031)539-1752, FAX : 031)539-1750, E-mail : hbchang@daejin.ac.kr
접수일 : 2009년 3월 24일, 수정일 : 2009년 5월 18일
완료일 : 2009년 6월 3일
[†] 종신회원, 대진대학교 경영학과 조교수
[††] 연세대학교 정보대학원
(E-mail : junkwon@yonsei.ac.kr)
[†††] 준회원, 대진대학교 컴퓨터공학과 박사과정
(E-mail : kimyh7902@daejin.ac.kr)
[††††] 종신회원, 대진대학교 컴퓨터공학과 교수
(E-mail : kgb@daejin.ac.kr)

측면에서의 보안취약점은 프로그램 자체에 근본적인 문제점 등이 있는 것은 아니고, 개발자의 실수 등에 의해 발생하는 개개의 프로그램에 있어서의 보안상 문제점이다[1,2].

정보보호의 주요과정은 개개의 프로그램에 대하여 보안취약성의 분석(Vulnerability Analysis)을 통하여 구체적인 보안취약점(Vulnerability Instance)을 발견(Discovery)해 내며, 발견된 보안취약점을 점검(Scan)하여 이를 악용하는 보안 사고를 방지하는 것이다. 결과적으로 보안취약성의 분석을 통한 보안취약점의 발견이 보안사고 방지를 위해 가장 중요하고 근본적인 부분이라 할 수 있다. 하지만 보안취약성의 분석은 프로그램의 크기가 커짐에 따라 한계 상황에 달하고 있으며, 지수 적으로 증가하는 분석대상 영역의 폭증에 효과적으로 대처할 수 있는 새로운 방법과 도구를 끊임없이 개발하여야 한다. 취약성 분석기술은 프로그램에 존재하는 보안취약성을 분석하여 보안취약점을 발견하는 기술이다. 이 기술은 개발이 완료된 프로그램의 분석을 기본으로 하기 때문에 프로그램 역 공학 기술과도 관련 있으며, 프로그램 자체의 분석을 넘어 프로그램에 존재하는 제반의 문제점이 보안사고와 연관될 수 있는지를 실제의 보안 사고에 앞서 판단하는 심도 있는 예측을 하는 분야이기도 하다[3].

따라서 국외에서 개발된 상용 프로그램을 많이 사용하는 국가에서는 적어도 보안취약성 분석을 독자적으로 할 수 있는 기술을 필수적으로 보유하여야 한다. 그리고 그러한 분석기술에 의해 얻어진 심도 있는 보안관련 정보는 국가의 기간산업에 직접적이고 전격적인 영향을 줄 수 있으므로, 비영리 국가기관 등에서 모든 분석과정을 합리적으로 통제해 나아가야 할 것이다. 인터넷 대란의 경우도 블라스터 웜 바이러스가 특정 프로그램의 취약점을 이용하여 발생한 전국적인 재난이다. 이러한 문제의 재발을 막기 위해서는 트로이 목마(Trojan Horse), 논리 폭탄(Logical Bomb) 등의 악의적인 코드가 삽입되어 있거나, 운영체제나 통신, 데이터베이스 호출상의 알려진 취약점을 이용하여 시스템 호출을 하는 보안 결함을 지닌 프로그램을 검출하거나 테스트하는 것은 보안상 반드시 필요한 분석 분야이다[4].

하지만 소스코드를 확인할 수 없는 대부분의 실행 프로그램의 보안 결함을 검출 또는 테스트할 수 있는 도구는 현재 국내외적으로 존재하지 않는 실정이다. 이에 본 연구는 소스코드가 없는 실행 프로그램의 보안 결함이 예상되어지는 부분을 실행하면서, 보안 결함이 의심되는 부분의 바이너리 코드를 추출하는 도구의 개발을 목표로 한다.

2. 바이너리 코드 추출기

프로그램들의 취약점은 공격자로 하여금 시스템을 점령하는 통로로 이용되어지고 있다. 이러한 취약점 중 일부는 프로그램 개발 라이프 사이클 상에서 초기 부분인 설계 단계에서 발생한 아주 깊이 위치하여 알아차리기 어려운 것도 존재하며, 또한 일부는 단순한 코딩상의 부주의로 인하여 발생하기도 한다.

결과적으로 보안취약성의 분석을 통한 보안취약점의 발견이 보안사고 방지를 위해 가장 중요하고 근본적인 부분이라 할 수 있다. 하지만 보안취약성의 분석은 프로그램의 크기가 커짐에 따라 한계 상황에 달하고 있으며, 지수 적으로 증가하는 분석 대상 영역의 폭증에 효과적으로 대처할 수 있는 새로운 방법과 도구를 개발하여야 할 필요가 증대되고 있다. 그리고 매달 발견되는 보안 취약성 개수가 바이러스 출현 개수보다 2002년부터 추월하기 시작한다는 보고는 향후에 시간이 갈수록 인위적으로 작성한 바이러스보다 무심코 잘못 코딩한 취약성으로 인한 피해가 더욱 커질 것이라는 것을 충분히 예상되어 질 수 있다. 윈도우 시스템의 취약성 상위 프로그램들을 살펴보면 Internet Information Service, MS SQL과 같이 일반적으로 널리 많이 사용하는 프로그램이 1, 2위를 차지하고 있다. 이러한 상용 프로그램의 경우 소스 코드가 없는 상태에서 보안 취약점을 분석하여야만 한다는 것이 큰 문제로 작용한다.

본 연구에서 개발하고자하는 바이너리 추출기는 이러한 소스 코드가 없는 상용 프로그램의 보안 취약점을 분석하기 위한 도구이다. 즉 분석 대상 실행 프로그램의 정규화 된 예상 취약점 부분을 토대로 대상 프로그램을 실제 수행하면서 동적 분석 및 조사하며, 이때 각종 로그 기록을 산출할 수 있는 도구개발을 목표로 한다.

3. 선행연구

어플리케이션 프로그램의 보안 취약점을 분석하기 위한 목적으로 바이너리 코드 추출기를 개발하는 경우는 별로 알려진 바가 없다. 오히려, 게임이나 각종 프로그램의 크래킹이나 해킹을 위하여 바이너리 코드 추출이 가능한 디버거들이 몇몇 개인들을 위주로 개발이 되고 있는 실정이다.

해커나 크래커가 사용하는 프로그램 분석도구는 주로 프로그램의 PE 정보를 변경하는 도구, 디버깅 도구, 역 어셈블러 도구 등이다. 이러한 도구들을 이용하여 취약성이 있는 의심되는 부분이나 허점을 바이너리 코드 수준에서 분석하여 공격을 시도하는 것이다[5,6].

일반적으로 사용하는 디버거 프로그램으로는 Microsoft사의 'WinDebug DbgWin(Microsoft Platform SDK에 포함)'과 Compuware사의 'SoftICE' 등이 있다. SoftICE와 같은 디버깅 도구는 Intel CPU에서 제공하는 in-circuit emulator(ICE) Mode를 사용한 디버깅 도구로서 가장 저수준에서 가장 강력한 디버깅 기능을 제공하고 있다.

대부분의 상용화된 디버깅 툴들은 그림 1처럼 디버깅 과정 중에 디스어셈블링 기능을 동시에 보유하고 있다.

4. 동적 분석 시스템 개발

4.1 디버그 인터럽트를 이용한 코드추적

실행 시 특정 조건에서 바이너리 코드를 추출하기 위하여 본 연구에서는 80x86 호환 CPU에서 제공하는 디버그 인터럽트인 int3을 사용하였다.

그림 2처럼 디버거 프로그램인 바이너리 추출기는 분석 대상 프로그램인 디버거를 실행하기 전에 시작점 규칙과 종료 점 규칙을 설정한 후 디버깅 프로그램을 실행하고, 각각의 디버그 이벤트에 대하여 루프를 반복하며 실행 추적을 수행하게 된다.

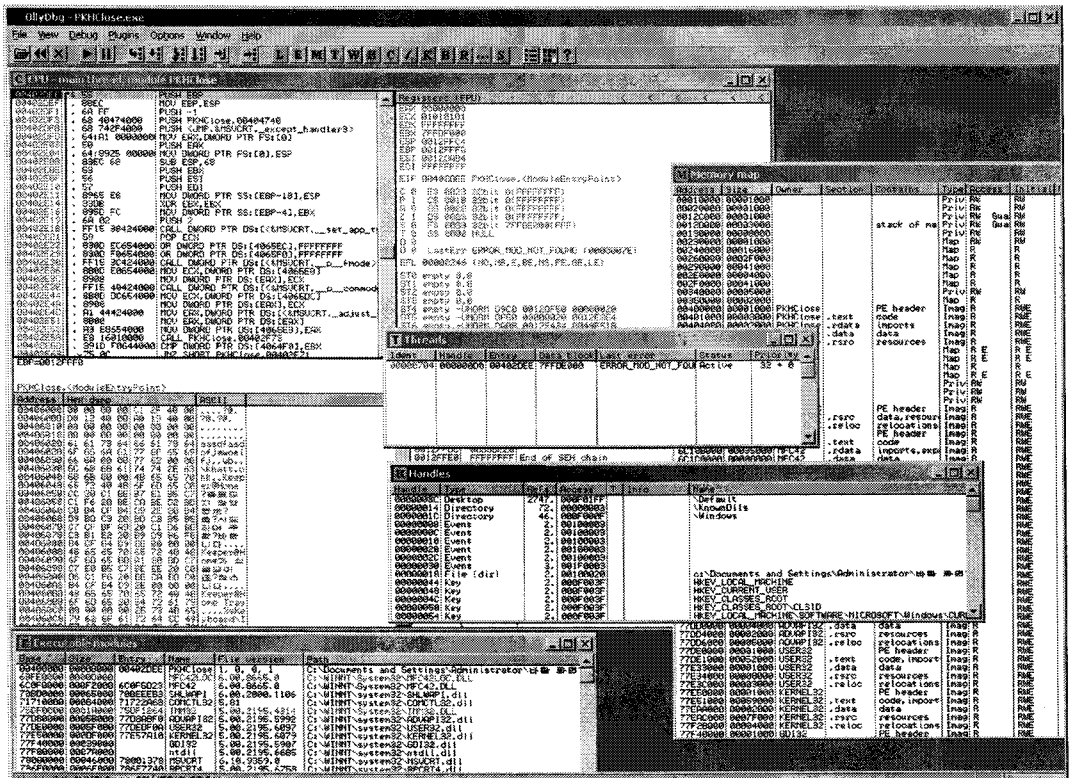


그림 1. Oleh Yuschuk의 'OllyDbg'

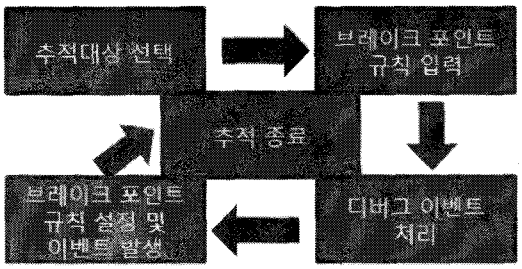


그림 2. 디버그 인터럽트를 이용한 코드 추적 프로세스

4.2 API 후킹을 통한 코드추적

실행 중인 프로그램의 모든 실행 영영에 대하여 바이너리 코드를 추출한다는 것은 매우 부하가 큰 작업이며, 또한 분석이 어려운 작업이다. 그러므로 취약성이 의심이 가는 부분(예를 들면, 패스워드 내용 처리 부분, TCP/IP 접속을 처리하는 부분)만을 지정하여 그 시작규칙을 정하여 실행 바이너리 코드를 추출하여 분석하는 것이 가장 좋은 방법이다.

특정 윈도우 메시지에 대한 메시지 handler의 내용을 추적하기 위해서는 윈도우 어플리케이션이 윈도우 메시지를 가져와 처리하는 API 함수인 Dispatch Message API 함수를 후킹하여, 후킹 한 함수 내에서 윈도우 시스템에서 전달된 윈도우 메시지와 추적을 하고자하는 특정 윈도우 메시지와 비교하여 조건에 일치할 경우에 인터럽트 3번을 수행하여 디버거로 수행을 전달하는 방법을 사용한다. 디버거로 제어가 돌아오면 디버거에서는 사용자에게 로깅을 수행할 것인지를 물어보게 된다.

5. 분석 절차

5.1 Disassemble

그림 3과 4처럼 동적 분석 시스템은 disassemble 메뉴를 통해 프로그램을 실행해서 실행파일의 disassemble 된 파일(실행파일이름.dis)을 생성한다. 메뉴를 선택하면 disassemble 할 파일을 선택하는 파일 다이얼로그가 나타나고 여기서 실행파일을 선택하면, 바로 disassemble이 수행된다. disassemble이 끝나면 실행파일이름.dis 파일이 생성된다. disassemble된 파일열기 메뉴를 선택하면 확장자가 "dis"인 파일을 선택하는 파일 다이얼로그가 나타나고, 여기서 disassemble 된 파일을 선택하면 파일의

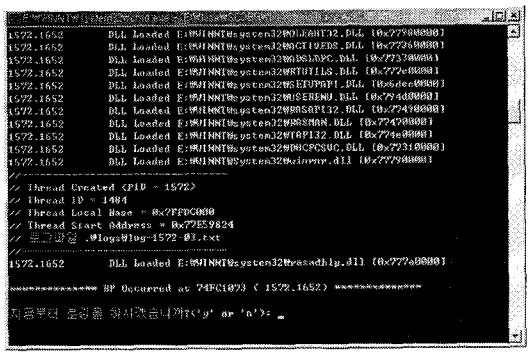


그림 3. API에 의한 동적 분석 시스템 실행 화면

그림 4. 메시지에 의한 동적 분석 시스템 실행화면

내용이 화면 좌측에 표시되게 된다. "동적 분석 모듈 호출" 메뉴를 선택하면 동적 분석을 수행할 파일을 선택하는 파일 다이얼로그가 나타나고, 파일을 선택하면, RS(Rule Start)와 RE(Rule End)를 선택하는 다이얼로그가 나타난다. 여기서 모드별로 "핫키", "API", "Message"를 선택한 후 해당하는 RS 와 RE를 설정한 후 확인을 누르면 동적 분석 모듈이 호출된다.

5.2 동적 분석 시스템의 추적로그

그림 5는 동적 분석 시스템에 의해 생성된 로그파일을 정적 분석 모듈의 오른쪽 화면 출력하는 기능이다. 정적 분석 시스템의 "로그파일 열기" 메뉴를 선택하면 로그파일을 선택하는 파일다이얼로그가 뜬다. 거기서 로그파일을 선택하면, 파일을 열 수 있다. 이때 주의사항은 동적 분석 모듈에 의해 생성된 "txt"파일만을 열어야한다는 것이다. 다른 "txt"파일을 열 경우 프로그램 오류를 일으킬 수 있다.

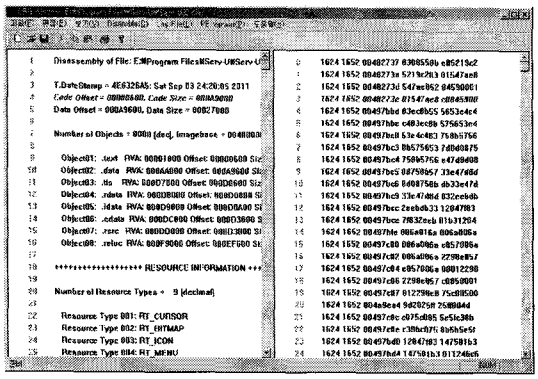


그림 5. 추적 로그 출력 화면

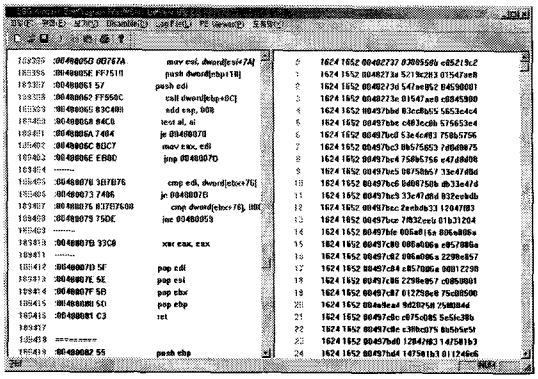


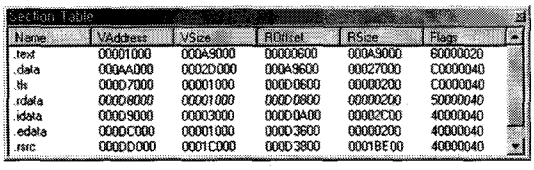
그림 6. mapping 수행 화면

5.3 추적 로그와 disassemble 된 파일의 mapping

disassemble된 파일 과 추적 로그를 같이 연 상태에서 "mapping"메뉴를 선택하면, 그림 6처럼 로그파일의 실행부위를 disassemble된 파일 화면에서 파랑색으로 나타나고, 해당 위치로 자동으로 이동하게 된다.

5.4 분석 다이얼로그

mapping을 수행한 후 "분석 다이얼로그"를 선택하면, 분석 다이얼로그가 나타난다. 분석 다이얼로그에는 현재 disassemble된 파일의 영역과, 맵핑된 영역을 다른 색으로 구분하여 보여주게 된다. 표시된 영역에 마우스를 클릭하면, 메인화면의 좌측 윈도우의 disassemble된 파일의 영역에서 클릭한 위치로 자동으로 이동하게 된다. 이때 disassemble된 파일의 영역을 찾아가는 데는 0.1%의 오차범위를 갖게 된다.



Name	VAddress	VSize	RAWOffset	RSize	Flags
.text	00001000	000A9000	00000600	000A9000	60000020
.data	000AA000	0002D000	000A9600	00027000	C0000040
.tls	000D7000	00001000	000D0600	00000200	C0000040
.rdata	000D8000	00001000	000D0800	00000200	50000040
.idata	000D9000	00003000	000D0A00	00002C00	40000040
.edata	000DC000	00001000	000D3600	00000200	40000040
.rsrc	000DD000	0001C000	000D3800	0001BE00	40000040

그림 7. Portable Execution Viewer Section Table 화면

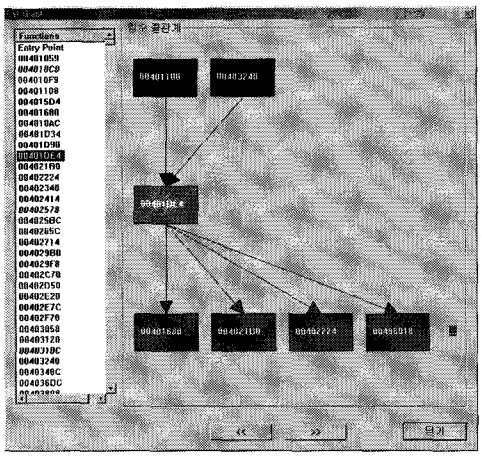


그림 8. Call Graph

5.5 PE View

메뉴의 "Portable Execution View"를 선택하면, 그림 7처럼 분석할 프로그램을 선택하는 메뉴가 나타나고, 파일을 선택하면 Portable Execution 정보를 나타내는 다이얼로그가 출력되게 된다. 여기서는 프로그램의 Entry Point, Size, Section Table, Import 정보 등을 볼 수 있다.

5.6 Call Graph 표시

disassemble된 파일이 열려있는 상태에서 메뉴의 "Call Graph"를 선택하면 그림 8처럼 현재 열려있는 disassemble된 파일에 대한 함수 Call 관계를 보여주는 다이얼로그 나타난다. 좌측의 함수들이 나타나는 리스트에서 함수를 선택하면 오른쪽에 그 함수를 호출하는 함수와 그 함수가 호출하는 함수들이 나타난다. 여기에서 보고 싶은 함수를 더블클릭하면 해당 함수에 해당하는 정보를 바로 볼 수 있다.

6. 결 론

정보 통신 기술의 발달로 정보시스템 사용은 점차

늘어나고 있으나, 인터넷과 같은 정보 통신망에 대한 취약성 및 위험도 동시에 가중됨에 따라 정보보호에 대한 중요성이 크게 부각되고 있다. 실제로 정보 유출, 파괴, 위·변조 등과 같은 컴퓨터 범죄 및 해킹이 급증하고 있으며, 바이러스 감염, 서비스 방해, 불건전 정보 유통 등과 같은 정보화 역기능이 확산됨에 따라 정보보호를 위한 보안 운영체제에 대한 개념은 개인 및 기업 활동을 포함한 사회전반에 있어서 중요한 이슈로 떠오르고 있다.

특히, 최근 들어 매월 발견되는 보안 취약성 개수는 바이러스 출현의 개수를 이미 추월하기 시작하고 있으며, 인위적으로 작성한 바이러스보다 무심코 잘못 코딩한 취약성으로 인한 피해가 더욱 커질 수 있는 상황에 이르게 된 것이다. 실제로 과거에 발생한 인터넷 대란의 원인은 블라스터 웜 바이러스(Blaster Warm Virus)가 특정 프로그램의 취약점을 공격하면서 발생한 결과이다.

하지만 소스코드를 확인할 수 없는 대부분의 실행 프로그램 보안 결함을 검출 또는 테스트 할 수 있는 도구가 현재 국내외적으로 존재하지 않고 있다. 이러한 상황에서 본 연구의 결과로 얻어지는 소스 코드가 없는 프로그램을 실행하면서 보안 결함이 예상되어지는 부분을 찾을 수 있는 도구 개발은 큰 의미를 갖는다. 본 연구에서는 선행연구에서 연구 개발한 도구에서 구현된 추적 시작점, 종료 점 규칙의 설정 기능, 멀티 threading 환경에서의 추적 등의 기능을 확장하여 일반적인 상용 프로그램에 대하여 실행 시의 바이너리 코드 추출 기능을 원활히 수행하는 도구를 개발하였다.

거대한 양의 기계어 코드로 구성되어 있는 일반 상용 프로그램에서 보안 취약점으로 의심되는 부분을 찾아가기 쉽게 하기 위하여 시작점 규칙(Rule Start)을 특정 API가 호출되는 시점, 특정 윈도우 메시지 처리 시점, 핫키 사용 시점 등으로 규정할 수 있도록 기능을 확장하였다. 또한 멀티 threading 실행 환경을 갖는 어플리케이션도 원활히 시작점 규칙을 통하여 동일하게 추적이 가능하도록 하였다. 개발된 도구의 성능 평가 결과, 일반 상용 프로그램은 물론 멀티threading을 사용하는 FTP 프로그램에서도 시작점 규칙을 통하여 분석을 원하는 위치에서 실행 바이너리 코드를 원활히 추출할 수가 있었으며, 정적 분석 모듈과 연계하여, 실제 로깅 된 정보가 disassemble 된 파일에서 어느 위치에 있는지도 쉽게 알아볼 수 있었다. 또한 disassemble된 파일의 분석을 통한 함수의 그래프 기능을 통해 프로그램에서 사용하는 함수들의 call 관계를 쉽게 판단할 수 있었다.

참 고 문 헌

[1] Bishop M. and Bailey D., 1996, "A Critical Analysis of Vulnerability Taxonomies," *Technical Report CSE-96-11, Dept. of Computer Science*, University of California at Davis.

[2] Brian Marick, 1990, "A Survey of Software Fault Surveys," *Technical Report UIUCDCS-R-90-1651*, University of Illinois at Urbana-Champaign.

[3] Eugene H. Spafford, "Common System Vulnerabilities," *Proceedings of the Workshop on Future Directions in Computer Misuse and Anomaly Detection*, 1992.

[4] Howard JD., 1997, "An Analysis of Security Incidents on the Internet," *Ph. D Thesis, Carnegie Mellon University.*

[5] Lee,Y. H. and Hwang, D. J., "Design and Implementation of Agent Based Dynamic Digital Rights Management," *Journal of Information Processing Association*, D. Vol. 8D, No. 5, pp. 613-622, 2001

[6] Otwell, K. and B. Aldridge, "The Role of Vulnerability in Risk Management," *IEEE Proceedings of the 5th Annual Computer Security Applicant Conference*, pp. 32-38, 1989.

장 항 배

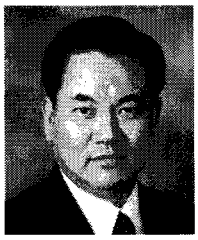
2001년 3월~2006년 2월 연세대학교 정보시스템 박사

2007년 3월~현재 대진대학교 경영학과 조교수

관심분야 : 산업보안, u 비즈니스 전략, 정보화(정보보호) 수준 및 성과평가

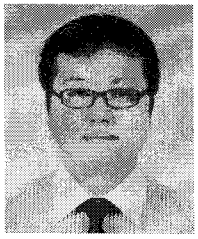
권 혁 준

2007년 3월~현재 연세대학교 정보대학원 박사과정
관심분야 : 산업보안, 지식관리 시스템, 정보시스템 관리 Social Computing



김 국 보

1990년~1993년 부경대학교 교수
1993년~현재 대진대학교 컴퓨터공학과 교수
관심분야 : 소프트웨어공학, 시스템 분석 및 설계, e-Biz 시스템



김 양 훈

2007년 3월~현재 대진대학교 컴퓨터공학과 박사과정
관심분야 : 소프트웨어 공학, 정보보안, IT Governance