결함 주입 방법을 이용한 소프트웨어 보안 취약점 검출

조병민*, 윤영민**, 최종천*, 조성제*, 유해영*

*단국대학교 정보컴퓨터과학과

**단국대학교 전산통계학과

Vulnerability Testing of Software using Fault Injection

Byoung-Min Cho*, Young-Min Yun**, Jong-Cheon Choi*, Seong-Je Cho*, Haeyoung Yoo*

*Dept. of information & computer science, Dankook University.

**Dept. of computer science & statistics, Dankook University.

요 약

최근 소프트웨어의 복잡도가 증가되어감에 따라 소프트웨어 취약점 검출에 대한 정형화된 방법과 자동화된 도구가 필요하게 되었다. 본 논문에서는 기존의 소프트웨어 테스트에서 고려되지 않았던 보안을 고려한 테스트라는 측면에서 자동화된 도구를 이용하여 소스가 없고 바이너리 코드만 있는 경우 결함 주입 기법을 통해 취약점 분석 방법을 보여주며, 윈도우즈 환경에서 사용되는 응용프로그램에 대한 상호 비교를 통해 향후 발생할 취약점에 대한 예방과 회피에 활용될 사례를 보여주고 있다.

I. 서론

컴퓨터가 개발되고 프로그램 되어진 소프트웨어가 사용되어지는 동안 소프트웨어에 대한 테스트와 검증에 대한 작업은 끊임없이 요구되어지고 발전해왔다. 최근 멀티미디어와 인터넷을 통한 네트워크적인 요소가 강화되고, 유비쿼터스 환경에서 사용자 중심의 정보화 흐름이 요구되어짐에 따라 소프트웨어에 대한 안전성이 더욱 절실하게 요구되어지고 있다.

특히 최근 인터넷을 통하여 전파되고 있는 웜이나 바이러스의 경우 소프트웨어 취약점을 통한 전파가 일반적인 경향으로 나타나고 있는 만큼 소프트웨어의 취약점에 대한 보다 빠른 탐지와 대응이 요구되어 진다.

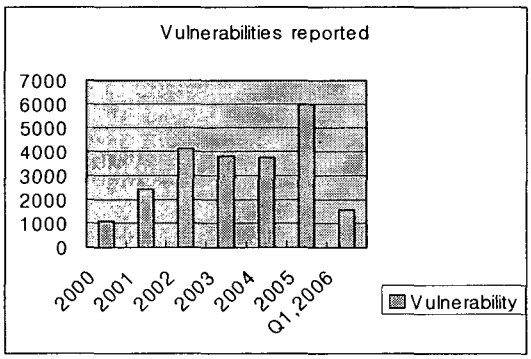
소프트웨어의 취약점에 기인한 막대한 손실과 보안관련 오류의 심각성이 커짐에 따라 소프트웨어 결함을 줄이려는 다양한 노력이 시도되고 있으며, 최근 소프트웨어의 규모증가와 그에 따른 분석의 복잡도 증가로 취약점 탐지를 자동화하는 도구를 개발하는 연구가 진행되고 있다. 이러한 연구노력으로는 프로그래머에 대한 체계적인 교육, 자동화된 프로그래밍 도구 개발, 품질보증 테스트, 그리고 소프트웨어의 취약점을 자동으로 분석하는 도구 개발 등이 있다. 본 논문에서는 결함주입 기법을 적용 가능한 소프트웨어 테스트 도구를 활용하여 기존의 소프트웨어 테스트에서 깊이 있게 다루어지지 않았던 소프트웨어 보안에 영향을 주는 취약점 검출 목록에 따른 테스트를 적용하고, 그 결과를 통해 소프트웨어 취약점에 대한 검출 사례를 보여준다.

본 논문의 구성은 다음과 같다. 2장에서는 보안취약점 현황 및 분석 방법을 설명하고, 3장에서는 윈도우즈에서 사용되는 응용프로그램을 대상으로 취약점 분석 사례를 재연하고 분석하였다. 4장에서 결론 및 향후 연구에 대해 기술한다.

II. 소프트웨어 취약점 분석

2.1 소프트웨어 취약점 분석 필요성

첨단 정보화 사회와 인터넷의 일반화, 컴퓨터 기술의 발달 등으로 소프트웨어의 개발 규모가 커지고 복잡해짐에 따라 잠재적인 보안취약점이 증가하고 있으며 이러한 보안결함을 조기에 발견할 수 있는 기술개발이 요구되고 있다. 소프트웨어 내부의 보안취약점은 소프트웨어 제작자의 주의와 테스트를 거친 이후에도 발견되고 있으며, 소프트웨어의 코드에 대한 보안취약점은 오라클과 같이 수년간에 걸쳐 테스트를 거친 상용 S/W의 경우에서도 계속 발견되고 있다. 최근 CERT의 발표를 보아도 소프트웨어 보안 취약점이 해마다 증가하고 있음을 (그림 2-2)에서 알 수 있으며, 그로 인한 피해도 계속 증가하고 있다[6].



(그림 2-2) 소프트웨어 보안취약점의 증가

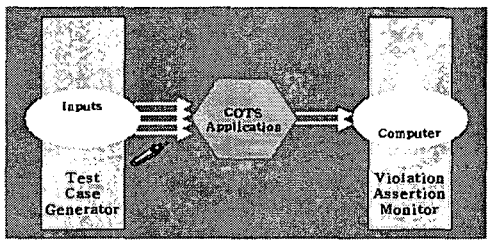
2.2 취약점 분석 기법

소프트웨어의 결함에 기인한 막대한 손실과 보안상의 오류의 심각성이 커짐에 따라 소프트웨어 결함을 줄이려는 다양한 노력이 시도되고 있다. 선진국 등에서는 소프트웨어를 직접 개발한 개발자들과 관련 전문가에 의해 소프트웨어 보안 취약점이 발견되어 알려지고 있으며, 최근 소프트웨어의 규모증가와 그에 따른 분석의 복잡도 증가로 취약점 탐지를 자동화하는 도구를 개발하는 연구를 진행하고 있다. 이러한 연구노력으로는 프로그래머에 대한 체계적인 교육, 자동화된 프로그래밍 도구 개발, 품질 보증 테스트(Quality Assurance Test), 그리고 소프트웨어의 취약점을 자동으로 분석하는 도구 개발 등이 있다. 이 중에서 취약점 자동 분석 도구는 접근 방법에 따라 개발단계에서 소스 코드에 적용하는 SA(Static Analysis. 정적 분석)[3], 실행중인 프로그램을 대상으로 하는 DA(Dynamic Analysis. 동적 분석)[4], SWIFI(Software Implemented Fault Injection. 결함주입)[5] 등으로 구분할 수 있다.

이런 개별 방법에 의한 보안취약점의 분석은 소프트웨어의 크기가 커짐에 따라 한계상황에 달하고 있다. 대규모 기계어 프로그램에 대한 보안취약점을 효과적으로 분석하는 방법과 도구를 개발할 필요가 있다. 이러한 분석 기법 중에서 본 논문에서는 결함주입 기법을 이용하여 분석 사례를 보여준다.

2.3 결함주입 기법

결함주입 기법이란 소프트웨어의 강건성을 테스트하는 한 방법으로 결함을 용용의 내부나 수행 환경에 삽입하고 소프트웨어가 삽입된 결함에 대해 어떻게 대처하는지 분석한다. 이 분석 내용을 통해 소프트웨어에 존재하는 에러를 제거하고 또한 결함을 감내할 수 있는 다양한 방법들을 추가하여 더욱 강건한(robust) 소프트웨어를 만들려는 시도이다. 결함주입 분석 방법에 대한 논리적인 구성도가 (그림 2-1)에 나타나 있다.



(그림 2-2) 결함주입에 관한 논리 구성도

본 논문에서는 결함주입 기법을 위한 도구로 Holodeck 이라는 소프트웨어 테스트 도구를 사용하였다. Holodeck은 테스터와 개발자가 결함주입 기법을 이용하여 메모리 부족, 유해한 파일, 부정한 레지스트리 데이터, 네트워크 패킷 오류 등 어플리케이션 시스템을 중단시킬 수 있는 시나리오를 시뮬레이션 할 수 있는 강력

한 도구이다[8].

III. 소프트웨어 취약점 검출

3.1 소프트웨어 취약점 검출

소프트웨어 취약점 분석 방법에는 2절에서 설명한 것과 같이 여러 단계와 기법으로 구분할 수 있다. 이러한 분석 기법을 활용하여 소프트웨어 취약점 검출이 가능하며, 효과적인 검출에 필요한 도구 및 기법의 개발이 필요하다. 이존의 소프트웨어 테스트는 명세중심의 소프트웨어 테스트가 주로 이루어 졌으며 이러한 테스트로는 알려지지 않은 취약점 분석에 많은 한계가 있으며, 본 논문에서는 보다 효율적인 소프트웨어 취약점 검출을 위해 기존의 소프트웨어 테스트 과정 중에서 특히 보안에 관련되어 중요하게 수행되어야할 테스트로 정의한 19가지 검출리스트를 가지고 테스트를 수행하였다[7]. 이러한 보안 취약점 검출리스트는 크게 4가지 부문으로 나눌 수 있는데 검출단계에 따라 시스템의존성, 사용자인터페이스, 설계단계, 구현단계로 구분하여 보안 취약점 검출을 수행할 수 있으며 본 논문에서는 이중 시스템 의존성에 대한 부분에 대한 검출을 수행하였다. 이를 위해 윈도우즈에서 많이 사용되는 응용프로그램인 훈글2004와 MS워드2003에 대한 취약점 검출을 수행하여 존재하는 취약점에 대한 재연과 두 프로그램 간에 발생한 취약점에 대한 상호 비교를 통하여 취약점 검출에 대한 예를 보여주고 있다.

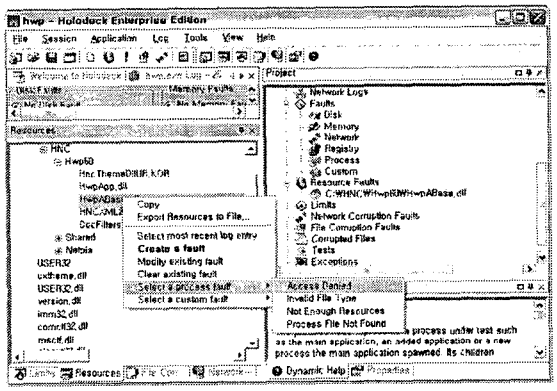
3.2 소프트웨어 취약점 재연

3.1절에서 이야기한 검출리스트에 대하여 본 논문에서는 시스템 의존성 부분에 대한 검출 방법을 적용하여 라이브러리에 대한 접근제한, 파일 엑세스 제한과 손상으로 인한 비정상적인 엑세스 등을 이용하여 훈글과 MS워드에 대한 테스트를 수행하였으며, 테스트 결과로 나타난 취약점에 대한 재연과 두 프로그램 간에 발생한 취약점에 대한 결과를 통해 상호 비교를 수행 하였다.

소프트웨어 취약점 재연을 위해 본 논문에서는 결합주입 기법을 위한 도구로 Holodeck라는 소프트웨어 테스트 도구를 이용하여 취약점에 대한 재연을 보여주고 있다.

3.3 소프트웨어 취약점 분석 결과

3.2절에서 보여준 소프트웨어 취약점 검출 방법 및 재연 기술을 통하여 시스템 의존성에 대한 훈글과 MS워드에 대한 취약점을 분석하여 상호 비교를 수행 하였다. (그림 3-1)은 훈글의 핵심 DLL 파일에 "Access denied" 공격을 수행한 결과이다.



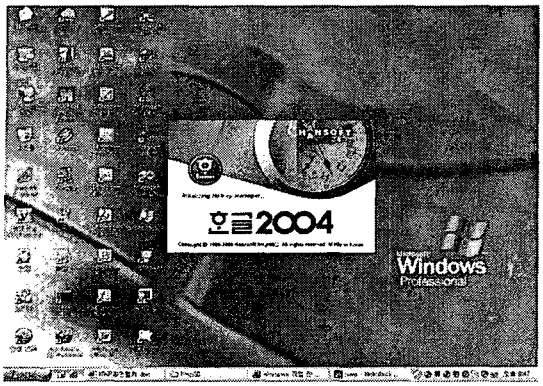
(그림 3-1) Access denied 공격

[표 3-1]은 DLL 파일별로 수행된 결과를 보여주고 있으며, 각각의 취약점이 19가지 취약점 리스트에 대하여 어느 부분에 해당하는 지를 보여주고 있다. 본 논문에서는 어플리케이션에 취약점 공격의 결과를 CRASH(비정상종료)와 DoS(응용의 교착상태)로 표현하고 있다.

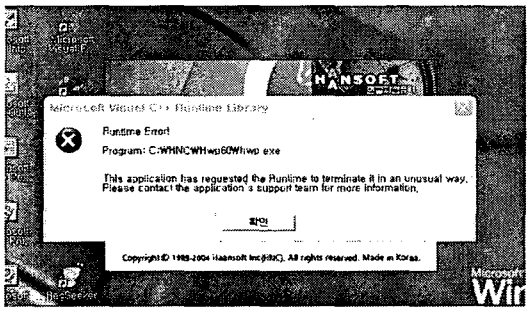
[표 3-1] 훈글 취약점 검출 결과

DLL명	취약점	검출 리스트
Hxxxxx2.DLL	CRASH	list 1,3,4
Hxxxxp.dll	CRASH	list 3,4
Hxxxxxe.dll	DoS	list 3,4

실험 결과 훈글의 경우 리스트 와 같이 3가지 라이브러리에서 취약점을 보였다. (그림 3-2)은 취약점 분석 도구에서 DoS의 재연을 보인다. 이때 프로그램은 시작 화면을 끝으로 어떠한 동작도 수행하지 않는다. 같은 맥락으로 해당 DLL파일들을 이름, 확장자, 경로와 같은 정보를 이용하여 의미 없는 파일로 변경시키는 방법으로 Hxxxxxe.dll 파일에 대하여 (그림 3-3) 와 같이 런타임 에러가 발생하는 것을 볼 수 있다.



(그림 3-2) DoS 재연 화면

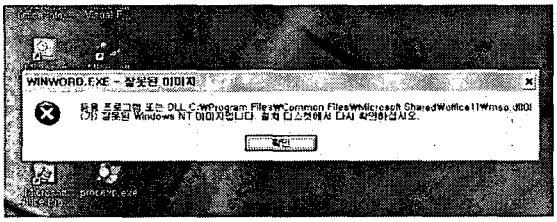


(그림 3-3) 훈글에서의 런타임 에러

반면 MS 워드의 경우 실험 결과 하나의 DLL파일에서만 [표 3-2]와 같이 취약점을 나타내었다.

[표 3-2] MS 워드 취약점 검출 결과

DLL명	취약점	검출 리스트
MXX.DLL	CRASH	list 3,4



(그림 3-4) MS 워드의 에러 처리

취약점을 유발 시킨 DLL파일을 훈글에서와 같이 의미없는 파일로 변경하고 MS 워드를 실행하였을 경우 (그림 3-4)에서 보이는 것처럼 훈글에서와는 달리 런타임 수행상의 에러가 아닌 명확한 에러처리와 문제가 있는 라이브러리 및 문제 해결 방법을 제시한다. 반면 분석 툴에서는 실행이 되지 않는 취약점을 확인할 수 있었다. 이러한 취약점들은 소프트웨어를 수행하는데 있어서 예기치 못한 결과를 유발하는 중대한 오류가 될 수 있으며, 앞의 사례에서 보듯이 동일한 기능의 소프트웨어라 하더라도 개발된 환경에 따라 발생하는 취약점의 위치와 결과가 달라지는 것을 볼 수 있다.

IV. 결론

본 논문에서는 보안 취약점 검출리스트에서 제공하는 검출 대상에 대하여 결함주입 기법을 활용한 취약점 검출 방법에 대한 사례를 보여주었다. 이러한 사례를 통하여 소프트웨어 개발자들은 유사기능의 프로그램에 대한 취약점 분석 결과를 자신이 개발하는 소프트웨어에 대한 취약점 점검의 토대로 활용이 가능하다. 또한 이러한 정형화된 검출 리스트를 활용할 경우 취약점에 대한 효율적인 검출이 가능하게 될 것이다. 향후 이러한 검출 방법에 대한 지속적인 연구를 통하여 자동화된 검출 도구의 개발이 요구된다.

[참고문헌]

[1] Matt Bishop, Computer Security : Art and Science, Addison-Wesley, 2003.

[2] William S., Operating Systems, Prentice-Hall, 2001.

[3] D. Evans and D. Larochelle. "Improving Security Using Extensible Lightweight Static Analysis", IEEE Software, Jan/Feb 2002.

[4] Lo R., Kerchen P., Crawford R., Ho W., Crossley J., Fink G., Levitt K., Olsson R. and Archer M., "Towards a Testbed for Malicious code detection", *COMPCON Spring '91. Digest of Papers.* San Francisco, CA, P 160-166, Feb.-Mar. 1991.

[5] Pete Broadwell and Emil Ong, "A Comparison of Static Analysis and Fault Injection Techniques for Developing Robust System Services", class project paper, May 2002.

[6] http://www.cert.org/stats/cert_stats.html#vulnerabilities

[7] James A. Whittaker, Herbert H. Thompson, How to Break Software SECURITY, Addison Wesley, 2003

[8] http://www.securityinnovation.com/