이벤트 구동 방식 프로그램을 위한 동적 테스트 도구의 설계

박지우O, 손윤식, 오세만
동국대학교-서울 컴퓨터공학과
{ jojaryong, sonbug, smoh } @dongguk.edu

Design of Dynamic Test Tool for Event-driven Programs

Jiwoo ParkO, Yunsik Son, Seman Oh
The Department of Computer Engineering, Dongguk University-Seoul

요 약

최근에는 스마트 폰의 저변이 확대되면서, 보안 취약성에 대한 많은 문제점이 새롭게 등장하고 있다. 스마트 폰 프로그램은 PC 환경에서 실행되는 프로그램과는 달리, 배포가 이루어진 이후에 소프트웨어 업데이트 등의 방법으로 보안 취약성을 제거하는 것이 매우 어려운 특징이 있다. 기존의 테스트 방법론은 스마트 폰의 특성에 대한 고려가 없기 때문에 스마트 폰을 위한 테스트 방법론과 함께 이를 위한 동적 테스트 도구에 대한 연구가 필요하다.

본 논문에서는 이벤트 구동 방식으로 동작하는 스마트 폰 프로그램의 특징을 고려한 동적 테스트 도구를 설계한다. 테스트 도구는 컴파일러 이론을 적용하여 체계적으로 설계 한다. 제안한 도구는 테스트 케이스 생성기와 테스팅 시스템으로 구성되며, 이벤트 구동 방식으로 동작하는 소프트웨어의 취약성 검출 자동화 도구로 활용할 수 있다.

1. 서 론

소프트웨어에 대한 사용자의 요구사항이 방대해지고 복잡해짐에 따라 소프트웨어 개발 및 테스트 방법론은 다양한 소프트웨어 취약성에 대한 고려가 이루어져야 한다. 특히, 오늘날의 소프트웨어는 인터넷 환경에서 데이터를 교환하기 때문에 입출력 데이터에 대한 신뢰성 보장이 매우 어렵다. 소프트웨어의 취약성은 심각한 경제적 손실을 발생시키는 소프트웨어 보안 침해사고의 직접적인 원인이 되어 왔다. 따라서 안전한 소프트웨어 개발을 위한 코딩 규약과 취약성 분석 도구에 대한 연구가 프로그래밍 언어적인 측면에서 활발하게 진행되고 있다.

최근에는 스마트 폰의 저변이 확대되면서, 보안 취약성에 대한 많은 문제점이 새롭게 등장하고 있다. 특히, 스마트 폰 프로그램은 PC 환경에서 실행되는 프로그램과는 달리, 배포가 이루어진 이후에 소프트웨어 업데이트 등의 방법으로 오류를 수정하는 것이 매우 어려운 특징이 있다.

기존의 테스트 방법론은 스마트 폰의 특성에 대한 고려가 없기 때문에 스마트 폰을 위한 테스트 방법론과 함께 스마트 폰 프로그램의 오류를 효과적으로 탐지하고 결과를 프로그래머에게 친숙한 형태로 제공하는 테스트 자동화 도구에 대한 연구가 필요하다.

본 논문에서 설계한 테스트 자동화 도구는 크게 테스트 케이스 생성기와 테스팅 시스템으로 이루어져 있으며, 입력으로 스마트 폰 콘텐츠를 받으며, 해당 콘텐츠를 이벤트 기반으로 테스트 할 수 있는 SSF(Stuffed Source File)와 테스트 정보를 생성한 후, 생성된 정보를 이용하여 테스트를 동적으로 수행한다.

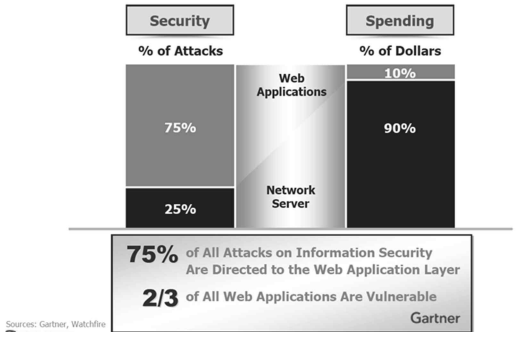
본 논문의 2장에서는 현재 활발하게 연구가 진행되고 있는 시큐어 코딩의 개념과 소스 코드의 취약성을 분석하는 자동화 도구에 대해 간략히 설명한다. 3장에서는 이벤트 구동 방식 프로그램의 취약성을 분석하고, 4장에서는 이를 위한 테스트 도구를 설계한다. 마지막으로 5장에서는 본 논문의 내용을 요약하고 향후 연구 방향에 대해 기술한다.

2. 관련 연구

2.1 시큐어 코딩

오늘날의 소프트웨어는 인터넷환경에서 데이터를 교환하므로 입출력 데이터에 대한 신뢰성을 확보하기 매우 어려우며, 임의의 침입자에게 악의적인 공격을 받을 가능성이 존재한다[1, 2]. 이러한 취약성은 심각한 경제적 손실을 발생시키는 소프트웨어 보안 침해사고의 직접적인 원인이 되어 왔다.

침해 사고의 예방을 위한 보안시스템은 네트워크 방화벽, 사용자 인증시스템 등이 대부분이지만 가트너의 보고서에 의하면 소프트웨어 보안 침해사고의 75%는 취약성을 내포하는 응용프로그램에 의해 발생되었다. 따라서 외부환경에 대한 보안시스템을 견고히 하는 것 보다 프로그래머가 견고한 소프트웨어 코드를 작성하는 것이 보안 수준을 향상시킬 수 있는 본질적이고 가장 효과적인 방법이다. 그러나 컴퓨터 시스템의 취약성을 줄이기 위한 대부분의 노력은 여전히 네트워크 서버에 치우쳐 있다.



[그림 1] Security and Spending Are Unbalanced(From Gartner, Watchfire)

최근에는 이러한 문제점을 인식하고 개발 단계에서부터 안전한 코드를 작성하는 시큐어 코딩에 대한 연구가 활발히 진행되고 있다. 특히, CWE[3]에서는 소스 코드 작성 단계에서 발생할 수 있는 다양한 취약성을 언어별로 분석하여 명시하고 있다. 또한, CERT[4]는 안전한 소스 코드를 작성하기 위한 시큐어 코딩 규칙을 정의하고 있다. 소프트웨어의 결함으로 인해 치명적인 문제가 발생할 수 있는 항공기, 자동차 등의 산업에서는 이미 JSF(Joint Strike Fighter), MISRA Coding Rule 등의 코딩 규약을 도입하여 양질의 소프트웨어 개발을 위한 지속적인 노력을 기울이고 있다.

2.2 소스 코드 취약성 분석 도구

MOPS[3]는 University of California at Berkeley에서 개발한 모델 검사기이다. MOPS는 보안 취약 요소를 프로퍼티로 정의하고, 유한 오토마타를 이용하여 정형화하였다. 따라서 적은 분석 비용으로 모델링된 취약성을 모두 검사할 수 있다. 그러나 자료 흐름 분석을 하지 않기 때문에 분석할 수 있는 취약성에 한계가 있다. Plum Hall의 Safe-Secure C/C++[4]는 컴파일러와 소프트웨어 분석 도구를 통합한 일종의 컴파일러이다. Safe-Secure C/C++는 버퍼 오버플로우 제거에만 초점을 맞추고 있다. 이 소프트웨어에 의해 만들어진 실행 프로그램은 버퍼 오버플로우를 100% 제거할 수 있으며, 일반적인 컴파일러에 의해 생성된 실행 파일보다 5% 이하의 성능 감소를 보인다고 주장하였다. Coverity의 Coverity Prevent[5]는 소스 코드에 대한 정적 분석 도구이다. Coverity Prevent는 전체 코드에서 발견된 취약점을 목록으로 나타낸다. 각각의 목록은 취약점이 발생한 소스 코드내의 위치와 취약점이 발생된 원인 등을 포함한다. Fortify SCA[6]는 Fortify에서 개발한 취약성 탐지 도구이다. Fortify 360은 C/C++, 자바 등 12개의 언어를 지원하며, 정적 분석과 동적 분석 기법을 사용하여 소스 코드의 취약성을 탐지한다. 발견된 취약성 정보는 통계 자료와 함께 사용자에게 제공된다.

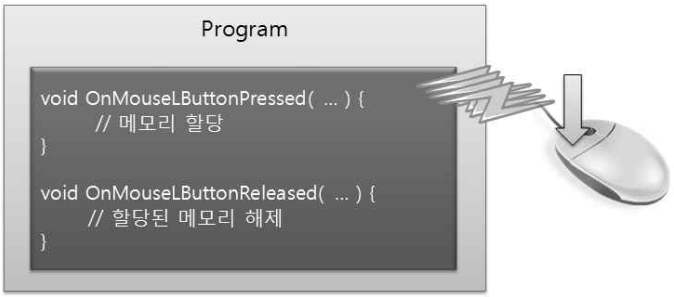
3. 이벤트 구동 방식 프로그램의 취약성 분석

이벤트 구동 방식으로 동작하는 프로그램의 취약성은 프로그래머가 예상하지 못한 이벤트와 데이터에 의해 노출된다. 이러한 프로그램의 취약성은 해커의 주요 공격 대상이 된다. 따라서 이벤트 구동 방식 프로그램의 견고성을 테스트 하기 위해서는 [표 1]과 같은 항목을 고려해야 한다.

[표 1] 이벤트 구동 방식 프로그램의 견고성 테스트

1. 프로그래머가 예상하지 못한 이벤트
2. 프로그램 내에서 발생할 수 없는 이벤트
3. 임의의 이벤트 시퀀스
4. 발생한 이벤트의 부적절한 데이터 전달

[그림 2]의 프로그램은 마우스 왼쪽 버튼을 누른 경우 메모리를 할당하고 버튼을 해제한 경우 할당된 메모리를 해제하는 예제이다.



[그림 2] 이벤트를 통한 프로그램 크러쉬 예

일반적으로 마우스 버튼을 누른 경우, 차후 마우스 버튼을 해제하는 이벤트가 항상 발생하는 것으로 간주할 수 있다. 그러나 임의의 공격자가 강제적으로 마우스 버튼 해제 이벤트 없이 마우스 버튼을 눌렀음을 알리는 이벤트를 보낼 경우, [그림 2]의 프로그램은 메모리와 관련된 취약성을 가지게 된다.

CWE와 CERT 등의 소프트웨어 취약성 분석 기관에서 분석한 이벤트와 관련된 프로그램의 취약성 항목은 각각 [표 2]와 [표 3]과 같다.

[표 2] CWE에서 제시한 이벤트 관련 취약성

CWE-360. Trust of System Event Data
CWE-422. Unprotected Windows Messaging Channel ('Shatter')
CWE-479. Unsafe Function Call from a Signal Handler
CWE-662. Insufficient Synchronization

[표 3] CERT에서 제시한 이벤트 관련 취약성

SIG00-C. Mask signals handled by noninterruptible signal handlers
SIG01-C. Understand implementation-specific details regarding signal handler persistence
SIG02-C. Avoid using signals to implement normal functionality
SIG04-C. Do not return from SIGFPE from inside a signal handler
SIG30-C. Call only asynchronous-safe functions within signal handlers
SIG31-C. Do not access or modify shared objects in signal handlers
SIG32-C. Do not call longjmp() from inside a signal handler
SIG33-C. Do not recursively invoke the raise() function
SIG34-C. Do not call signal() from within interruptible signal handlers
SIG35-C. Do not return from SIGSEGV, SIGILL, or SIGFPE signal handlers

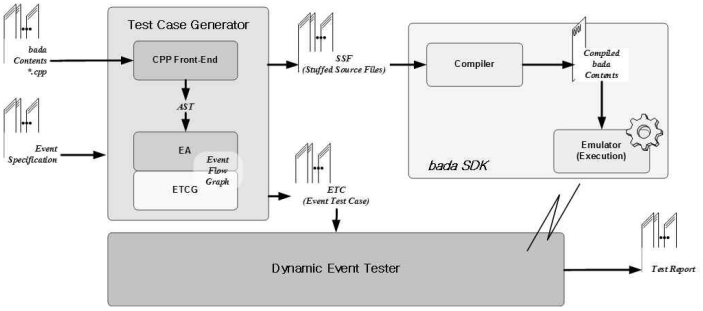
본 논문에서의 이벤트 구동 방식 프로그램을 위한 테스트 자동화 도구는 이벤트 구동 방식 프로그램의 특징과 관련된 예상 취약성과 CWE와 CERT의 이벤트 관련 취약성 분석 자료를 토대로 설계한다. [표 4]는 이벤트 구동 방식 프로그램에서 발생 가능한 취약성의 보여준다.

[표 4] 예상 취약성 리스트

번호	항 목
1	시스템 메시지 혹은 요구사항을 처리하지 못함
2	예상되는 범위를 벗어난 입력의 전달로 인한 애플리케이션의 부적절한 종료
3	성능에 크게 영향을 주는 API를 호출하는 이벤트의 악의적인 발생
4	침입자의 프로그램에 의해 생성된 이벤트로 인한 보안 문제
5	잘못된 시스템 이벤트 처리기 구현으로 인한 치명적인 시스템 오류 혹은 보안 문제 유발
6	노출되지 않은 시스템 기능을 직접적으로 접근함으로써 전체 시스템에 악영향을 줌
7	부적절한 이벤트와 데이터의 전달로 인해 애플리케이션이 의도하지 않은 상태로 변경됨
8	애플리케이션이 프로그래머의 의도와 다르게 동작하거나 오류를 유발함

4. 이벤트 구동 방식 프로그램을 위한 동적 테스트 도구

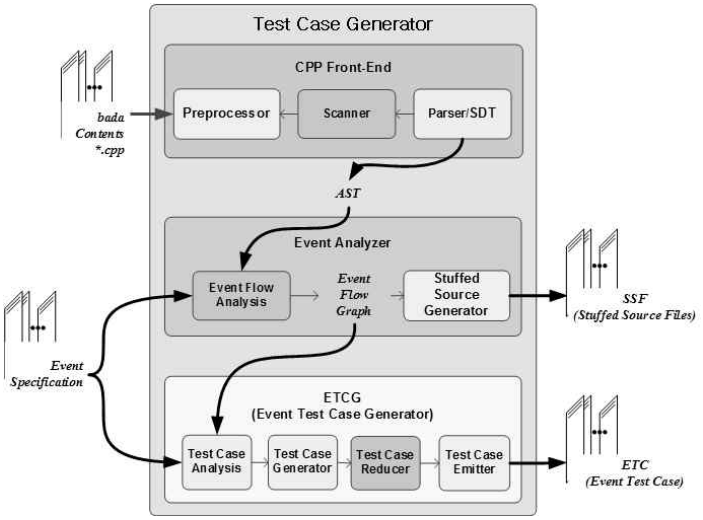
본 논문은 설계한 이벤트 구동 방식의 프로그램을 동적으로 테스트할 수 있는 테스트 자동화 도구는 테스트 케이스 생성기와 테스팅 시스템 2가지로 구성된다. 전자는 입력으로 bada[10] 플랫폼용 콘텐츠를 받으며, 해당 콘텐츠를 이벤트 기반으로 테스트 할 수 있는 SSF(Stuffed Source File)와 테스트 정보를 생성하고, 후자는 생성된 정보를 이용하여 입력 프로그램을 동적으로 테스트하고 테스트 보고서를 결과로 출력한다. [그림 3]은 테스트 자동화 도구의 구성도이다.



[그림 3] 테스트 자동화 도구 구성도

테스트 자동화 도구는 크게 테스트 케이스 생성기와 테스팅 시스템으로 이루어져 있으며, 입력으로 bada 플랫폼용 콘텐츠를 받으며, 해당 콘텐츠를 이벤트 기반으로 테스트 할 수 있는 SSF(Stuffed Source File)와 테스트 정보를 생성한 후, 생성된 정보를 이용하여 테스트를 동적으로 수행한다.

테스트 케이스 생성기의 구성도는 [그림 4]와 같으며, CPP전단부와 정적 분석을 이용한 이벤트 분석기(Event Analysis)와 이벤트 테스트 케이스 생성기(ETCG, Event Test Case Generator)로 구성된다.



[그림 4] 테스트 케이스 생성기 구성도

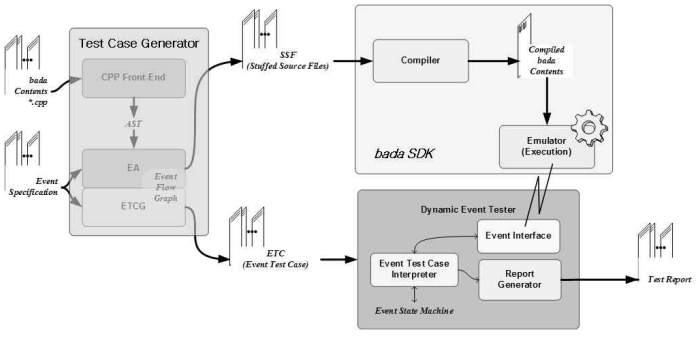
CPP 전단부는 Preprocessor, Scanner, Parser/SDT 모듈로 구성되어 있으며, 입력 CPP 소스 프로그램을 분석

에 용이한 AST 형태로 변환한다. Event Analysis는 Event Flow Analysis 모듈, Stuffed Source Generator로 구성되어 있으며, AST와 이벤트 명세서를 입력으로 받아 이벤트 흐름 그래프(EFG, Event Flow Graph)를 구성하고, 이를 이용하여 원시 콘텐츠에 테스트용 코드가 추가된 Stuffed Source File을 생성한다. ETCG(Event Test Case Generator)는 테스트 케이스 분석기, 테스트 케이스 생성 모듈과 테스트 케이스 경감기, 테스트 케이스 출력기로 구성되며, 이벤트 명세서와 Event Analysis에서 생성되는 EFG를 입력으로 받아 동적 테스트 케이스를 생성한다. 테스트 케이스 생성기는 [표 4]의 예상 취약성 항목별로 [표 5]의 검사 내용에 해당하는 테스트 케이스를 생성할 수 있다.

[표 5] 예상 취약성 분석을 위한 검사 내용

항목	관련 취약성
필수적인 시스템 이벤트가 모두 처리되는지 검사	1
발생된 이벤트와 입력 데이터에 대한 유효성 검사	2,4,6,7
사용자의 행위에 의해 호출 가능한 API 여부 검사	3
외부 애플리케이션에서의 접근 허용 여부를 검사하는 루틴 분석	4
시스템 이벤트 처리기의 재정의 유무 검사	5
플랫폼에서 제시한 구현 방법론 만족 여부	5
프로그래머에 의해 발생 가능한 이벤트인지 검사	6
이벤트가 발생하는 순서대로 정확히 구현되었는지 검사	8
특정(주로 시스템) 이벤트 처리기에서 처리되어야 할 항목 검사	5

테스팅 시스템 모델은 [그림 5]와 같으며, 테스트 케이스 생성기의 출력인 SSF와 이벤트 테스트 케이스를 이용하여 입력 콘텐츠에 대한 검사를 동적으로 수행한다. 테스팅 시스템은 동적 이벤트 테스터와 bada SDK에서 제공되는 컴파일러, 에뮬레이터로 구성된다.



[그림 5] 테스팅 시스템 모델

먼저, SFF는 기존의 bada SDK의 컴파일러를 이용하여 bada 플랫폼에서 실행 가능하도록 컴파일 되며, bada SDK의 에뮬레이터에서 실행된다. 다음으로, 동적 이벤트 테스터는 이벤트 분석 결과로 얻어진 DETC와 콘텐츠 실행에서 발생하는 다양한 정보 및 이벤트를 에뮬레이터로부터 전달받아 테스트를 동적으로 수행하고 테스트 보고서를 생성한다. 동적 테스트의 유형은 [표 6]과 같이 크게 3가지 형태로 구분할 수 있다.

[표 6] 동적 테스트 유형

1. 유효한 키 혹은 터치 스크린 이벤트
   (a) 프로그램의 기능 검사
2. 랜덤하게 발생된 이벤트
   (a) 프로그램의 견고성 검사
3. 프로그램 분석을 통해 생성된 이벤트
   (b) 데이터 흐름 그래프, 이벤트 흐름 그래프 분석
   (c) 프로그램의 취약성 및 예상치 못한 상황 집중 검사

동적으로 프로그램을 테스트하기 위해서는 기존의 버퍼-오버플로우 보호를 위한 동적 분석 기법[11]인 데이터 삽입, 모니터 코드 삽입, 메타데이터 태깅, 라이브러리 치환/랩핑 등을 응용하여 적용한다. 또한, 테스터와의 상호작용 코드를 콘텐츠에 삽입하여 에뮬레이터와의 데이터 교환에 대한 제약 사항을 극복한다.

5. 결론 및 향후연구

스마트 폰 프로그램의 버그 탐지는 대부분 고전적인 소프트웨어 테스트 방법론과 테스트 자동화 도구에 의존한다. 이러한 방법론은 이벤트 기반 프로그램의 특징을 고려하지 않았으며, 테스트 시나리오의 작성 작업과 함께 테스트 전문 인력이 요구된다. 본 논문에서 제안한 스마트 폰을 위한 동적 테스트 도구는 이벤트 기반 프로그램의 특성을 위주로 테스트를 수행한다. 또한, 소스 코드의 분석을 통한 테스트 케이스의 자동 생성, 동적 테스트 등 기존의 테스트 도구와 차별화된 기능을 갖는다.

본 도구는 소스코드를 분석하여 테스트 케이스를 자동으로 생성하고, 이를 동적 테스터에 적용하여 자동으로 테스팅을 진행할 수 있는 이벤트 구동 방식으로 동작하는 스마트 폰 프로그램을 위한 동적 테스트 도구를 개발하는 것이다. 본 논문에서 설계된 자동화 도구는 모바일 프로그램의 테스트를 효과적으로 수행할 수 있는 도구이다. 따라서 콘텐츠 테스팅 도구의 체계적이고 자동적인 구성 방법론을 이용하여 실무에 필요한 테스팅 기술의 도입을 쉽게 하며, 새로운 수익 모델을 창출할 수 있을 것으로 생각된다.

향후에는 제안된 동적 테스트 도구를 구현하고 실험 내용을 평가하는 연구가 필요하다. 또한, 다수의 스마트 폰 플랫폼에 적용하기 위해, 각각의 플랫폼을 분석하고 이러한 플랫폼의 특성을 명세할 수 있는 방법론에 대한 연구가 진행되어야 한다.

[참고문헌]

[1] Gary McGraw, Software Security, Addison-Wesley, February 2006.
[2] John Viega, Gary McGraw, Building Secure Software, Addison-Wesley, September 2001.
[3] Common Weakness Enumeration(CWE): A community-Developed Dictionary of Software Weakness Types, http://cwe.mitre.org/.
[4] J. McManus and D. Mohindra, The CERT Sun Microsystems Secure Coding Standard for Java, CERT, 2009.
[5] H Chen and D Wagner, "MOPS: an infrastructure for examining security properties of software," Proceedingsofthe9thACMConferenceonComputerand CommunicationsSecurity,pp.235-244,2002.
[6] Plum Hall, Inc. Overview of Safe-Secure Project: Safe-Secure C/C++, http://www.plumhall.com/SSCC_MP_071b.pdf, 2006.
[7] Coverity, Inc., Coverity Static Analysis, http://www.coverity.com/products/static-analysis.html, 2009.
[8] Fortify Software Inc., "Fortify Source Code Analysis(SCA)," http://www.fortify.com /products /sca.
[9] A.V.Aho, R.Sethi, J.D.Ulman, Compilers: Principles, Techniques, and Tools, Addison Wesley, 2007.
[10] Bada Developers, http://developer.bada.com/
[11] K. Piromsopa, R. Enbody, Survey of Buffer-Overflow Protection Technical Reports #MSU-CSE-06-3, Department of Computer Science and Engineering, Michigan State University, 2006.