Journal of Korea Multimedia Society Vol. 14, No. 10, October 2011(pp. 1335-1347)

모바일 애플리케이션을 위한 취약점 분석기의 설계 및 구현

문일룡†, 오세만††

요 약

최근 모바일 애플리케이션의 보급과 사용은 급속도로 확장되고 있으며, 이 과정에서 모바일 애플리케이션의 보안이 새로운 문제로 대두되고 있다. 일반적인 소프트웨어의 안전성은 시큐어 코딩을 통해 개발 단계에서부터 검증까지 체계적으로 이루어지고 있으나 모바일 애플리케이션의 경우는 아직 연구가 미흡한 실정이다. 본 논문에서는 모바일 애플리케이션에 특화된 취약점 항목을 도출하고 이를 기반으로 취약점을 분석할 수 있는 취약점 분석기를 설계하고 구현한다. 취약점 목록은 CWE(Common Weakness Enumeration)와 CERT(Computer Emergency Response Team)를 기반으로 모바일 애플리케이션의 특징인 이벤트 구동방식을 한정하여 도출하였으며, 분석 도구는 동적 테스트를 통하여 애플리케이션 소스 내에 취약점이 존재하는지 검사한다. 또한 도출된 취약점 목록은 모바일 애플리케이션을 작성하는 프로그래머의 지침서로 활용 될 수 있다.

Design and Implementation of A Weakness Analyzer for Mobile Applications

Ilyong Mun†, Seman Oh††

ABSTRACT

The dissemination and use of mobile applications have been rapidly expanding these days. And in such a situation, the security of mobile applications has emerged as a new issue. Although the safety of general software such as desktop and enterprise software is systematically achieved from the development phase to the verification phase through secure coding, there have been not sufficient studies on the safety of mobile applications yet. This paper deals with deriving weakness enumeration specialized in mobile applications and implementing a tool that can automatically analyze the derived weakness. Deriving the weakness enumeration can be achieved based on CWE(Common Weakness Enumeration) and CERT(Computer Emergency Response Team) relating to the event-driven method that is generally used in developing mobile applications. The analysis tool uses the dynamic tests to check whether there are specified vulnerabilities in the source code of mobile applications. Moreover, the derived vulnerability could be used as a guidebook for programmers to develop mobile applications.

Key words: Pogramming Language(프로그래밍 언어), Secure Coding(시큐어 코딩), Weakness Analyzer(취약점 분석기), Event-Driven Programming(이벤트 구동방식 프로그래밍)

※ 교신저자(Corresponding Author) : 오세만, 주소 : 서울시 중구 필동3가 동국대학교 원흥관 E408, 전화 : 02)2260-3342, FAX : 02)2265-8742, E-mail : smoh@dgu.edu
접수일 : 2011년 7월 1일, 수정일 : 2011년 8월 30일
완료일 : 2011년 9월 7일

† 정회원, (주)어니컴
(E-mail : aqua@onycom.com)
†† 정회원, 동국대학교 컴퓨터공학과
(E-mail : smoh@dgu.edu)

1. 서 론

일반적으로, 오늘날의 소프트웨어는 인터넷 환경에서 데이터를 교환하기 때문에 입출력 데이터에 대한 신뢰성 보장이 매우 어렵다. 이러한 소프트웨어의 취약점은 심각한 경제적 손실을 발생시키는 소프트웨어 보안 침해사고의 직접적인 원인이 되어 왔다. 현재까지는 소프트웨어 보안 침해 사고를 방어하기 위해, 방화벽, 사용자 인증 시스템 등의 보안 시스템에 초점을 맞추어 연구되어 왔다. 그러나 통계자료에 의하면 이러한 보안 침해 사고의 75%는 소프트웨어의 취약점으로부터 기인했다[1]. 따라서 부가적인 보안 시스템을 통해 침해 사고를 방어하는 것보다, 안전한 소프트웨어 개발을 위한 코딩 규약과 취약점 분석 도구를 통해 소프트웨어 보안 침해 사고를 사전에 예방하는 것이 매우 중요하다.

컴퓨팅 환경은 대형 컴퓨터에서 퍼스널 컴퓨터(PC)로 변화해왔으며, 최근에는 포스트 PC(Post PC)로서 스마트 폰, 타블렛 PC, 디지털 TV(DTV) 등이 각광받는 시대가 되었다. 특히, 최근에는 스마트 폰의 저변이 확대되면서, 포스트 PC로서의 스마트 폰의 사용 비중이 증가하고 있는 현실에 있다. 이러한 상황에 직면하면서, 오픈 마켓을 통해 제공되는 애플리케이션의 수가 기하급수적으로 증가하고 있다. 이러한 애플리케이션은 개방형 모바일 플랫폼을 탑재된 단말기의 확산과 함께 다양한 경로로 배포되고 있다. 따라서 모바일 애플리케이션의 보안 취약점에 대한 많은 문제점이 새롭게 등장하고 있다.

스마트 폰 애플리케이션은 PC 환경에서 실행되는 프로그램과는 달리, 배포가 이루어진 이후에 소프트웨어 업데이트 등의 방법으로 오류를 수정하는 것이 매우 어려운 특징이 있다. 특히, 기존의 시큐어 코딩 및 취약점 분석 방법론은 이러한 스마트 폰과 모바일 애플리케이션의 특성에 대한 고려가 없기 때문에 모바일 애플리케이션의 취약점을 분석하기에는 많은 제약이 따른다. 특히, 취약점 측면에서 살펴보면, 일반적인 소프트웨어의 취약점은 이미 제시된 다양한 취약점과 이를 검사할 수 있는 도구를 통해 대부분 제거될 수 있다. 그러나 모바일 애플리케이션은 기존의 도구를 활용하여 취약점을 제거한 후에도 이러한 도구들이 검사하지 않는 이벤트 구동 방식과 관련된 취약점, 모바일 애플리케이션의 실행 플랫폼에 의존적인 취약점 등이 존재할 가능성이 있다.

본 논문에서는 이러한 문제점을 해결하기 위하여 모바일 애플리케이션의 특징을 반영한 취약점을 도출하고 도출된 취약점을 탐지하기 위한 취약점 분석기를 설계 및 구현한다. 이를 위해, 먼저 모바일 애플리케이션을 위한 취약점 목록을 도출한다. 취약점 목록은 CWE, CERT에서 분석하고 관리하고 있는 취약점 목록을 이벤트 구동방식 프로그램[2]이라는 관점에서 접근하여 추출하고 또한 프로그래머를 위한 애플리케이션 개발 안내서로부터 도출한다.

다음으로, 도출된 취약점을 적용하여 모바일 애플리케이션의 소스 상에 존재하는 취약점을 분석할 수 있는 도구를 설계하고 구현한다. 이를 위해 도출된 취약점 목록을 검사하기 위한 정형화된 모델을 설계한다. 설계된 모델은 bada 플랫폼과 C++ 언어를 대상으로 적용하여 취약점 분석 도구를 구현한다.

취약점 분석 모델을 통해 설계한 분석 도구는 크게 테스트 시나리오 생성기와 동적 테스트 시스템으로 이루어져 있으며, 입력으로 bada 애플리케이션의 소스를 받는다. 테스트 시나리오 생성기는 소스 코드를 분석하여 정적 분석으로 검출 가능한 취약점을 탐지하고, 해당 애플리케이션을 이벤트 기반으로 테스트할 수 있는 SSF(Stubbed Source File)와 테스트 시나리오를 생성한다. 동적 테스트 시스템은 테스트 시나리오 생성기에서 출력된 테스트 시나리오를 입력으로 받아, bada 시뮬레이터와 정보를 송수신 하는 방법으로 동적 검사를 수행한다. 테스트 시나리오 생성기와 동적 테스트 시스템에 의해 검출된 취약점은 최종적으로 XML 파일 형식의 보고서로 출력된다. 구현된 취약점 분석 도구는 실험을 통해 애플리케이션 그룹별로 검출된 취약점의 수, 소스 코드 라인당 취약점의 수 등에 대한 통계자료를 제시하고 검증한다.

2. 관련연구

2.1 시큐어 코딩

오늘날의 소프트웨어는 인터넷환경에서 데이터를 교환하므로 입출력 데이터에 대한 신뢰성을 확보하기 매우 어려우며, 임의의 침입자에게 악의적인 공격을 받을 가능성이 존재한다[3,4]. 이러한 취약점은 심각한 경제적 손실을 발생시키는 소프트웨어 보안 침해사고의 직접적인 원인으로 작용되어 왔다.

침해 사고의 예방을 위한 보안시스템은 네트워크 방화벽, 사용자 인증시스템 등이 대부분이지만 가트너 보고서[1]에 의하면 소프트웨어 보안 침해사고의 75%는 취약점을 내포하는 응용프로그램에 의해 발생되었다. 따라서 외부환경에 대한 보안시스템을 견고히 하는 것보다 프로그래머가 견고한 소프트웨어 코드를 작성하는 것이 보안 수준을 향상시킬 수 있는 본질적이고 가장 효과적인 방법이다. 그러나 컴퓨터 시스템의 취약점을 줄이기 위한 대부분의 노력은 여전히 네트워크 서버에 치우쳐 있다. 그림 1은 이러한 내용을 잘 설명하고 있다.

최근에는 이러한 문제점을 인식하고 개발 단계에서부터 안전한 코드를 작성하는 시큐어 코딩에 대한 연구가 활발히 진행되고 있다. 특히, 미국토안보부(The U.S. Department of Homeland Security)에서 관리하고 있는 CWE[5]는 소프트웨어의 취약점을 프로그래머가 쉽게 접근할 수 있도록 사전식으로 분류하여 제공하고 있다. 또한, 수집된 취약점 항목을 뷰, 카테고리, 취약점, 복합요소를 기준으로 분류하여 살펴볼 수 있는 특징을 가지고 있다. 또한, 25개의 가장 위험한 프로그래밍 오류[6]를 제시하고 있다. CERT[7]는 카네기 멜론 대학교의 소프트웨어 공학 연구소에서 관리하고 있으며, 코딩 규칙 및 가이드의 표준화 작업을 수행하고 있다. CERT는 시큐어 코딩 표준을 프로그래밍 언어별로 구분하여 제공하기 때문에 개발 언어에 해당하는 시큐어 코딩 표준을 선별적으로 학습하기 용이하도록 구성되어 있다. 소프트웨어의 결함으로 인해 치명적인 문제가 발생할 수 있는 항공기, 자동차 등의 산업에서는 이미 JSF (Joint Strike Fighter)[8], 미즈라 코딩 규칙(MISRA coding rule)[9] 등의 코딩 규약을 도입하여 양질의 소프트웨어 개발을 위한 지속적인 노력을 기울이고 있다.

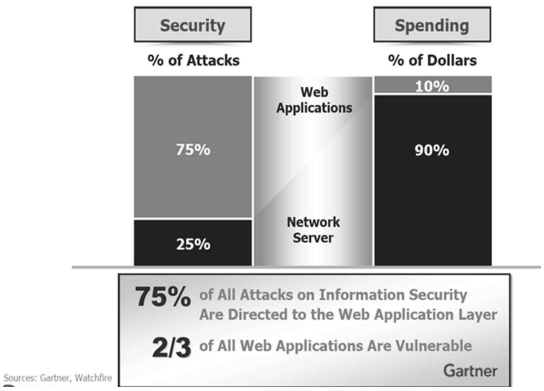

그림 1. Security and Spending Are Unbalanced (From Gartner, Watchfire)[1]

2.2 소스코드 취약점 분석 기법 및 도구

소스코드 분석도구의 분석 기법으로는 정적 분석[10]과 동적 분석[11]으로 나눌 수 있다. 정적 분석은 프로그램을 실제로 실행해 보지 않고 분석하는 방법으로 적은 비용으로 모든 실행 패스를 고려할 수 있는 장점이 있지만 과탐 발생 가능성이 큰 단점이 있다. 동적 분석은 프로그램을 실행하면서 분석하는 방법으로 높은 정확도를 가진 반면, 분석 비용이 크고 구현 복잡도가 높으며 모든 실행 패스를 고려할 수 없는 단점이 있다. 다시 말해서, 정적 분석은 과탐의 문제점이 있으며 역으로 동적 분석은 미탐의 문제점을 안고 있다.

현재 많이 사용되는 소스코드 취약점 분석기는 MOPS, Safe-Secure C/C++, Coverity Prevent, Fortify 360 등이 있다.

MOPS[12]는 UC Berkeley에서 개발한 모델 검사기이다. MOPS는 보안 취약 요소를 프로퍼티로 정의하고, 유한 오토마타를 이용하여 정형화하였다. 따라서 적은 분석 비용으로 모델링된 취약점을 모두 검사할 수 있다. 그러나 자료 흐름 분석을 하지 않기 때문에 분석할 수 있는 취약점에 한계가 있다.

Plum Hall의 Safe-Secure C/C++[13]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[14]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[15]는 Fortify에서 개발한 취약점 탐지 도구이다. Fortify 360은 C/C++, 자바 등 12개의 언어를 지원하며, 정적 분석과 동적 분석 기법을 사

용하여 소스 코드의 취약점을 탐지한다. 발견된 취약점 정보는 통계 자료와 함께 사용자에게 제공된다.

파수닷컴의 Sparrow[16]는 프로그램의 실행 의미 분석을 통해 버퍼 오버런, 메모리 누수 등의 치명적인 메모리 오류를 검출해 주는 실행의미 기반 프로그램 오류 자동 분석기이다. 제공되는 분석 정보는 분석된 오류에 대한 분석 실행 시간, 오류 발생 경로, 메모리 상태 등이 있다.

코딩 단계에서 취약점이 발생하는 가장 주요한 원인중의 하나는 잘못된 방식으로 API를 사용하는 경우이다. 이와 관련하여, 접근위반, 임계영역, 포맷 스트링 등 7개의 그룹과 연계된 API를 정리하고 이에 특화된 분석을 수행하는 자동화 분석 도구에 대한 연구[17]도 진행된 바 있다.

2.3 bada 플랫폼

bada[18]는 삼성전자에서 개발한 스마트 폰을 위한 플랫폼이다. bada 플랫폼의 장점은 바다폰이 가진 다양한 기능을 쉽게 제어할 수 있는 API를 제공하는 것이다. bada 플랫폼에서 동작하는 애플리케이션은 기본적으로 그림 2와 같이 *OnAppInitializing*, *OnAppTerminating*, *OnForeground*, *OnBackground*로 구성된 4개의 상태전이 사이클[19]을 갖도록 구성되어 있다.

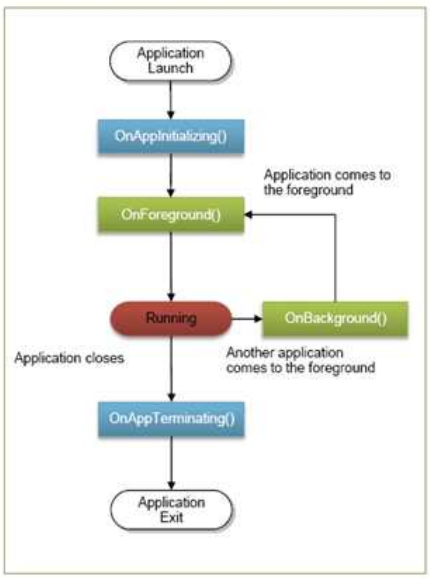

그림 2. bada 애플리케이션의 생명 주기

상태전이에 따른 각각의 이벤트를 처리하기 위해서는 Application 클래스의 가상함수를 활용할 수 있다. OnAppInitializing()은 애플리케이션 생성시 UI를 초기화 하는 부분이며, 데이터 초기화 또는 이전 상태로의 복원을 위한 코드를 작성하는 부분이다. OnAppTerminating()은 애플리케이션이 종료될 때는 호출되며, 할당된 리소스들을 해제한다. OnForeground()는 애플리케이션이 처음 생성되면 자동으로 호출 되며, 일반적으로 OnBackground()에서 멈추었던 프로세스를 재개하는 과정이 수행된다. OnBackground()에서는 연속적으로 실행되는 프로세스의 동작 등을 멈추는 과정이 수행 된다.

3. 모바일 애플리케이션의 취약점

3.1 취약점 목록 도출 방법

최근의 모바일 애플리케이션은 전화 및 모바일 기기라는 특성을 효과적으로 반영하여 사용자와의 상호작용하기 위해 이벤트 구동 방식으로 동작한다. 본 논문에서는 기존의 취약점 항목에서 반영하지 못한 모바일 애플리케이션의 취약점을 이벤트 구동 방식 프로그램의 관점에서 분석하고 도출한다. 그림 3은 이러한 방법을 그림으로 나타낸 것이다.

취약점 목록을 도출하기 위해서는 개발자의 풍부한 개발 경험을 도태로 알려진 취약한 코드 패턴을 수집하고 이를 분석해야한다. 그러나 이러한 패턴의 수집은 많은 시간이 소모되며, 검증 과정을 거쳐야 하는 문제점이 있다. 따라서 모바일 애플리케이션의 특성을 반영한 취약점은 CWE, CERT에서 관리하고 있는 이미 검증된 취약점 목록을 기반으로 도출하는 것이 효율적이다. 따라서 CWE와 CERT에서 이벤트에 관련된 취약점 항목을 추출하고, 삼성에서 제공하

그림 3. 취약점 목록 도출 방법

는 bada 개발 가이드의 내용을 기반으로 취약점 패턴을 수집한 후, 이를 토대로 모바일 애플리케이션에서 발생 가능한 취약점 항목을 도출한다.

3.2 취약점 분류

도출된 취약점은 언어 독립적인 취약점, 언어 의존적인 취약점, 그리고 플랫폼 의존적인 취약점으로 분류할 수 있다. 그림 4는 취약점 분류에 따른 시큐어 프로그램(Secure Program) 제작의 순환도를 나타낸다.

언어 독립적인 취약점은 이름 명명 규칙 등의 보편적 코딩 규칙이며, 언어 의존적인 취약점은 C++ 언어의 취약점으로 bada 애플리케이션 개발을 위한 언어의 취약점이다. 플랫폼 의존적인 취약점은 bada 플랫폼에서 제공하는 기능으로 인해 발생 할 수 있는 취약점과 실행 환경인 스마트폰의 특징을 고려한 취약점 그리고 플랫폼 자체적인 취약점을 갖고 있다.

언어 독립적인 취약점은 CWE와 CERT 그리고 코딩 관습(Coding Convention)으로부터 도출할 수 있고 언어 의존적인 취약점은 CWE와 CERT 그리고 애플리케이션의 종류별 특징 및 취약점이 존재하는 패턴으로부터 수집한다. 또한 플랫폼 의존적인 취약점은 bada 개발 가이드, 개발자의 경험을 토대로 알려진 취약점 그리고 새로운 취약점을 도출하게 된다.

또한 애플리케이션 개발 과정에서 취약점이 없는 프로그램을 개발하기 위해서는 그림 4에서 보여주듯이 시큐어 코딩을 위한 프로그래머의 지침서에 따라 작성한 후, 다양한 테스트가 필요하다. 테스트 작업은 소스 코드 분석을 통한 코드 검증과 시나리오를 통한 프로그램 안정성 검증으로 진행되며, 이러한 테스트 과정을 거친 후 최종적으로 시큐어한 프로그램이 제작된다.

3.3 이벤트 구동방식 프로그램의 취약점

이벤트 구동 방식으로 동작하는 프로그램은 프로그래머가 예상하지 못한 이벤트와 데이터에 의해 취약점이 발생하며, 이러한 프로그램의 취약점은 해커의 주요 공격 대상이 된다. 따라서 이벤트 구동 방식 프로그램의 견고성을 테스트하기 위해서는 표 1과 같은 항목을 고려해야 한다.

CWE와 CERT 등의 소프트웨어 취약점 분석 기관에서 분석한 내용을 기반으로 분류한 이벤트 관련 취약점 항목은 각각 표 2와 같다.

CWE에서는 4개 항목에서 이벤트와 밀접하게 연관된 취약점 항목을 찾을 수 있었으며, CERT에서는 C언어를 위한 시큐어 코딩 가이드의 시그널 관련 카테고리인 SIG에서 10개 항목을 도출할 수 있었다. 이러한 취약점 항목과 모바일 기기의 특성을 고려하여 종합한 결과 이벤트 구동 방식의 프로그램을 작성

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

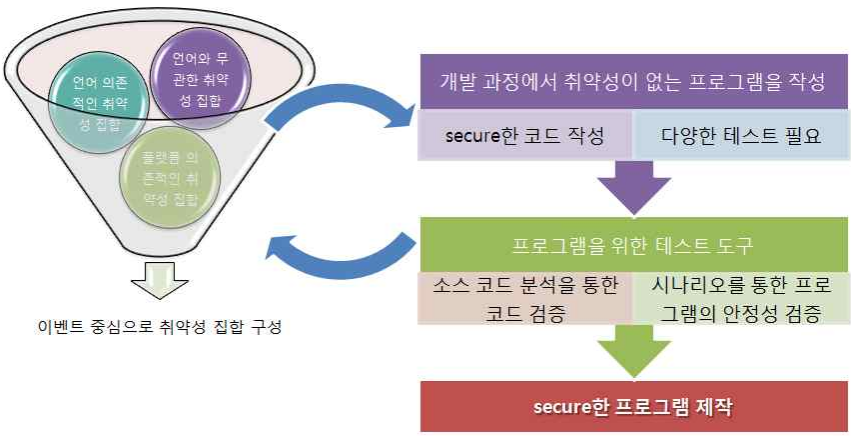

그림 4. 취약점 분류에 따른 시큐어 프로그램 제작 순환도

표 2. CWE와 CERT 기준 이벤트 관련 취약점

기 관	번 호	항 목
CWE (4개)	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
CERT (10개)	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

할 때, 이벤트 분류에 따라 발생 할 수 있는 주요 예상 취약점은 표 3과 같다.

표 3. 이벤트 분류에 따라 발생할 수 있는 주요 예상 취약점

1. 시스템 메시지 혹은 요구사항을 처리하지 못함.
2. 예상되는 범위를 벗어난 입력의 전달로 인한 프로그램의 부적적한 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 프로그램이 의도하지 않은 상태로 변경됨.
8. 프로그램이 프로그래머의 의도와 다르게 동작하거나 오류를 유발함.

3.4 도출된 취약점 목록

이벤트 구동 프로그램과 관련하여 총 23개의 취약점을 도출하였으며, 취약점 목록을 취약점 분류에 따라 구분하면 언어 독립적 취약점, 언어 의존적 취약점, 플랫폼 의존적인 취약점으로 구분할 수 있다. 그림 5는 취약점 도출 기준에 따른 취약점 목록을 도표로 표현한 것이다.

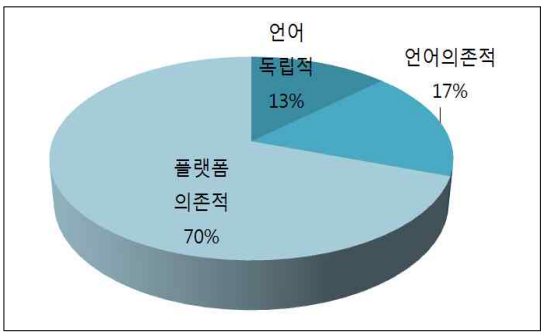

그림 5. 취약점 도출 기준에 따른 취약점 목록

언어 독립적 취약점은 3개의 목록을 도출하였으며, 애플리케이션의 정적 분석을 통해 검출이 가능하다. 언어 의존적 취약점은 총 4개의 취약점이 있으며, 주로 하이브리드 방식으로 검사를 수행한다. 총 16개의 플랫폼 의존적 취약점은 대부분 동적 분석과 하이브리드 분석 방법을 통하여 분석이 가능하다. 본 논문에서는 이벤트 구동방식 프로그램과 모바일 기기의 특성을 반영한 취약점에 초점을 맞추었기 때문에 플랫폼 의존적인 취약점이 많은 비중을 차지한다.

도출된 취약점 목록을 분석 방법에 따라 분류하면 다음과 같다. 23개 취약점 중 17개는 정적 분석 방법을 통하여 분석할 수 있으며, 3개는 동적 분석 방법을 통하여 분석할 수 있다. 그리고 동적 분석과 정적 분

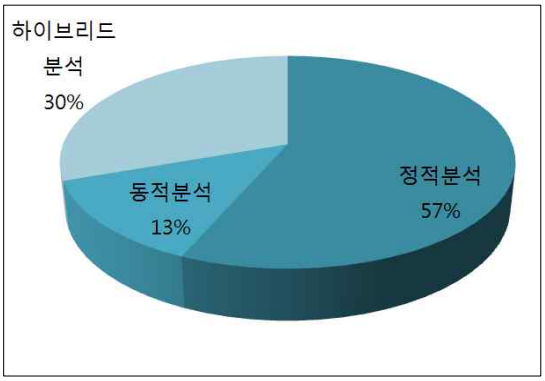

그림 6. 취약점 분석 방법에 따른 취약점 목록

석을 동시에 수행하는 하이브리드 분석 방법을 통하여 7개를 분석할 수 있으며, 그림 6은 취약점 분석 방법에 따른 취약점 목록을 도표로 표현한 것이다.

취약점의 의미에 따라 취약점 목록을 8가지 형태로 구분할 수 있으며, 그림 7은 8개의 그룹에 따라 취약점 목록을 분류한 도표이다. 도표에서 나타난 것과 같이 가장 많은 취약점을 포함하고 있는 그룹은 리소스와 전력소모이다. 모바일 기기에서 배터리와 미디어 등의 리소스는 극히 제한된 자원이기 때문에 이와 관련된 취약점이 발생하는 것을 반드시 사전에 차단해야 한다.

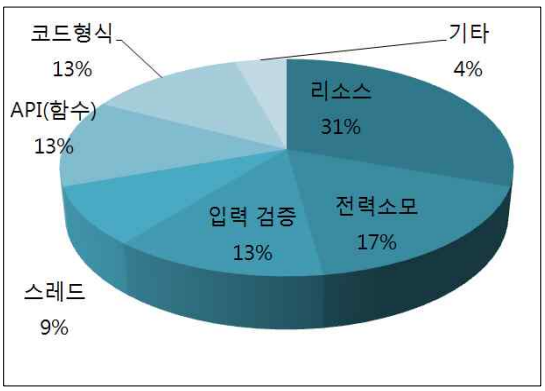

그림 7. 의미에 따른 취약점의 분류

4. 취약점 분석기의 설계 및 구현

4.1 취약점 분석기의 설계

본 절에서는 도출된 취약점 목록을 토대로 모바일 애플리케이션에 존재하는 취약점을 검사할 수 있는 취약점 분석기를 설계하고 구현한다. 이벤트 관련 취약점은 실행 플랫폼과 밀접한 관련을 가지고 있으며, 본 논문에서는 도출한 취약점 목록을 검증하기 위한 대상으로 삼성전자에서 개발한 bada 플랫폼을 사용하고, 대상언어는 C++, 그리고 이벤트 구동 방식의 모바일 애플리케이션을 대상으로 한다.

취약점 분석 방법으로는 정적 분석 13개, 동적 분석 3개, 하이브리드 분석 7개를 분석 방법별로 분류하여 그림 8과 같은 취약점 분석기 모델을 구축한다.

CWE, CERT, 개발자 설명서로부터 도출된 취약점은 활용도를 높이기 위해 XML 형태로 기술하며, 개발자 안내서로 사용할 수 있도록 체계적으로 문서화 한다. 취약점 분석 도구는 모바일 애플리케이션의 취약점을 검출하고 결과를 XML 형식의 보고서로 생성한다. 개발자는 결과 보고서와 취약점 목록을 참고하여 애플리케이션의 취약점을 수정할 수 있으며, 수정하는 과정을 반복적으로 수행함으로써 취약점을 프로그래밍 과정에서 차단할 수 있다.

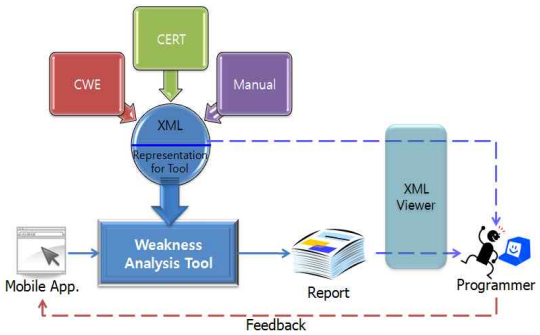

그림 8. 취약점 분석기 모델

4.2 취약점 분석기의 구현

모바일 애플리케이션을 위한 취약점 분석기는 Test Scenario Generator(TSG)와 Dynamic Event Tester(DET) 2가지로 구성된다. 그림 9는 취약점 분석기의 구현 모델을 그림으로 표현한 것이다.

TSG는 입력으로 bada 애플리케이션을 받으며, 정적 분석으로 추출 가능한 취약점을 검출한다. 또한, 동적 분석과 하이브리드 분석을 수행하기 위한 SSF(Stubbed Source File)와 테스트 정보를 생성한다. DET는 생성된 정보를 이용하여 애플리케이션을 동적 및 하이브리드 방법으로 분석하고 취약점 검출 결과를 XML 형태로 생성한다.

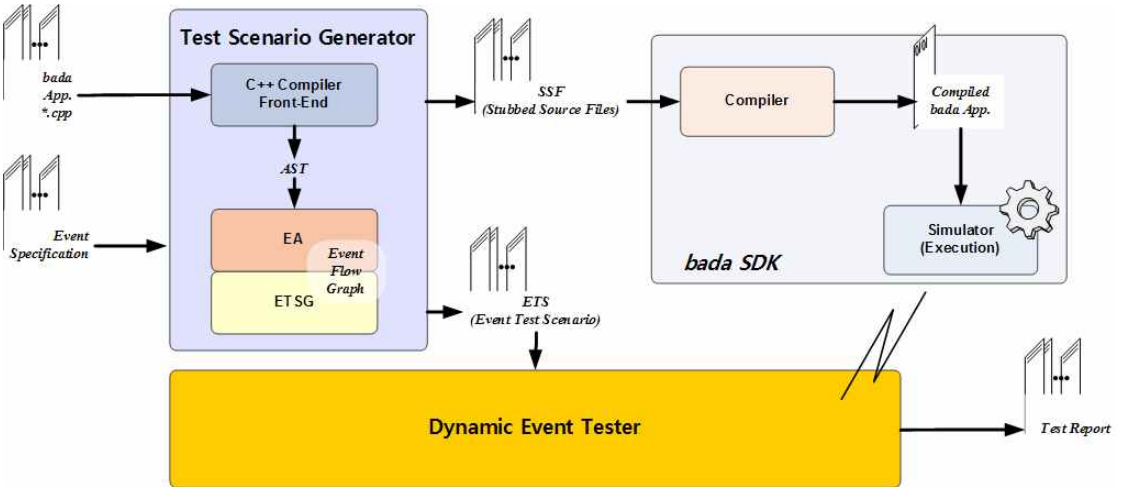

그림 9. 취약점 분석기의 구현 모델

4.2.1 Test Scenario Generator

TSG는 C++ 컴파일러 전단부(C++ Compiler Front-End)와 정적 분석을 이용한 이벤트 분석기(Event Analyzer)와 이벤트 테스트 시나리오 생성기(ETSG, Event Test Scenario Generator)로 구성된다. TSG의 구성도는 그림 10과 같다.

C++ 컴파일러 전단부는 전처리기, 스캐너, 파서 모듈로 구성되어 있으며, 애플리케이션의 소스 파일을 입력받아 소스 코드를 파싱하여 분석에 용이한 AST(Abstract Syntax Tree)를 생성하여 출력한다.

이벤트 분석기는 EFA(Event Flow Analyzer) 모듈과 SSG(Stubbed Source Generator)로 구성되어 있으며, C++ 컴파일러 전단부에서 생성된 AST와 대상 플랫폼의 이벤트 명세서를 입력받아 이벤트 흐름

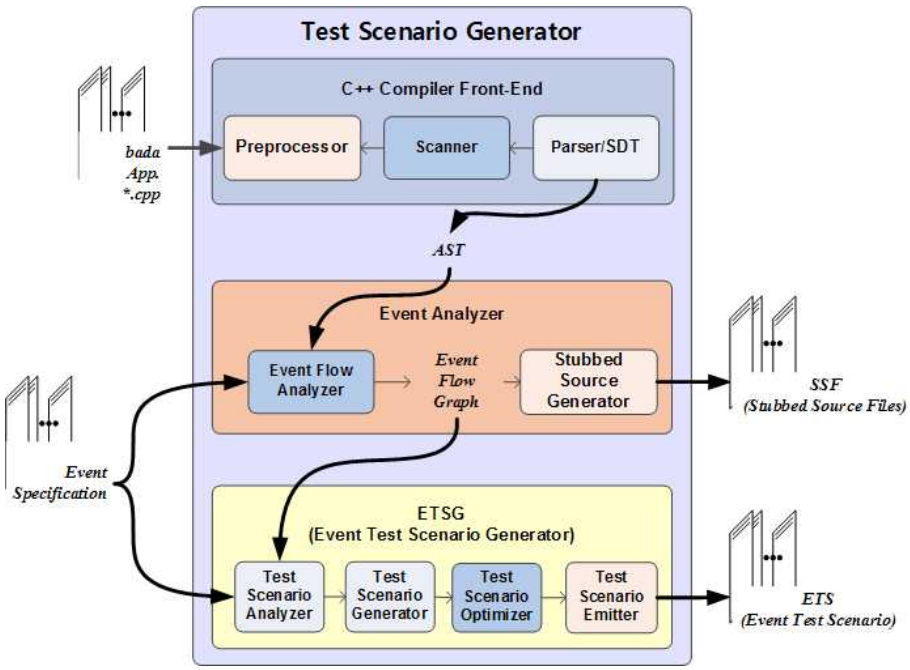

그림 10. 테스트 시나리오 생성기 구성도

그래프(EFG, Event Flow Graph)를 구성하고, 이를 이용하여 원시 애플리케이션 소스 프로그램에 테스트용 코드가 추가된 SSF(Stubbed Source File)를 생성한다.

ETSG(Event Test Scenario Generator)는 테스트 시나리오 분석기, 테스트 시나리오 생성 모듈과 테스트 시나리오 최적화기, 테스트 시나리오 출력기로 구성되며, 대상 플랫폼의 이벤트 명세서와 이벤트 분석기에서 생성되는 EFG를 입력으로 받아 DET의 입력으로 사용되는 ETS(Event Test Scenario)를 생성한다. TSG는 예상 취약점 항목별로 검사 내용에 해당하는 테스트 시나리오를 생성한다.

TSG에서 bada 애플리케이션의 취약점을 분석하고 동적 테스트를 위한 이벤트 시나리오를 생성하는 과정을 주요 호출 시퀀스로 표현하면 그림 11과 같다.

bada 애플리케이션은 C++ 컴파일러 전단부를 통해 AST의 형태로 변환된다. *ReaderforAnalysis Information* 클래스의 read *() 함수는 분석에 필요한 정보를 읽어 EDATable에 저장을 하고, *ASTTracer* 클래스의 traceTree() 함수는 collect *() 함수를 호출하여 EDATable에 AST 탐색 정보를 저장한다. 분석에 필요한 모든 정보가 EDATable에 수집되면 *EventAnalyzer* 클래스의 doAnalysis() 함수가 취약점 분석을 수행한다. doAnalysis() 함수는 각각의 취약점마다 개별적으로 존재한다. 분석 과정에서 정적 분석이 가능한 취약점을 검출하면, *ReportGenerator* 클래스의 wirte *() 함수를 호출하여 취약점 보고서를 생성한다. 동적 테스트가 요구되는 취약점인 경우 *CodeInjector* 클래스의 wirteStubCode() 함수를 통해 bada 애플리케이션 소스프로그램을 변환하여 Stubbed bada 애플리케이션 소스 프로그램을 생성한다.

4.2.2 Dynamic Event Tester

TSG의 출력인 SSF와 ETS를 이용하여 애플리케이션 소스에 대한 테스트를 동적 및 하이브리드 분석으로 수행하는 DET는 크게 3개의 모듈로 구성되어 있으며, 구성도는 그림 12와 같다.

ETS 인터프리터는 이벤트 인터페이스를 통해 bada 시뮬레이터와 데이터를 송수신하면서 테스트 시나리오를 수행한다. 테스트 시나리오 수행 중에 취약점에 관련된 정보를 수신하며, 이를 Report Generator에 전달한다. 이벤트 인터페이스는 DET와 bada 시뮬레이터 사이의 인터페이스 역할을 수행하며, 주로 데이터 송수신과 관련된 작업을 처리한다. Report Generator는 ETS 인터프리터로부터 전달받은 데이터를 가공하여 XML 형식의 테스트 보고서를 생성한다.

DET에서 bada 애플리케이션의 취약점을 동적으로 테스트하고 취약점 보고서를 생성하는 과정을 주요 호출 시퀀스로 표현하면 그림 13과 같다.

ScenarioManager 클래스의 parseScenario() 함수는 입력 파일인 Test Event Scenario의 유효성을 검사하고 트리 형태로 변환한다. loadScenario() 함수는 트리 형태로 구성된 시나리오를 이벤트 단위로 메모리에 로드한 후, analyzeScenario() 함수에서 시나리오를 분석하여 동적 테스트를 위한 이벤트 큐에 이벤트 시퀀스를 저장한다.

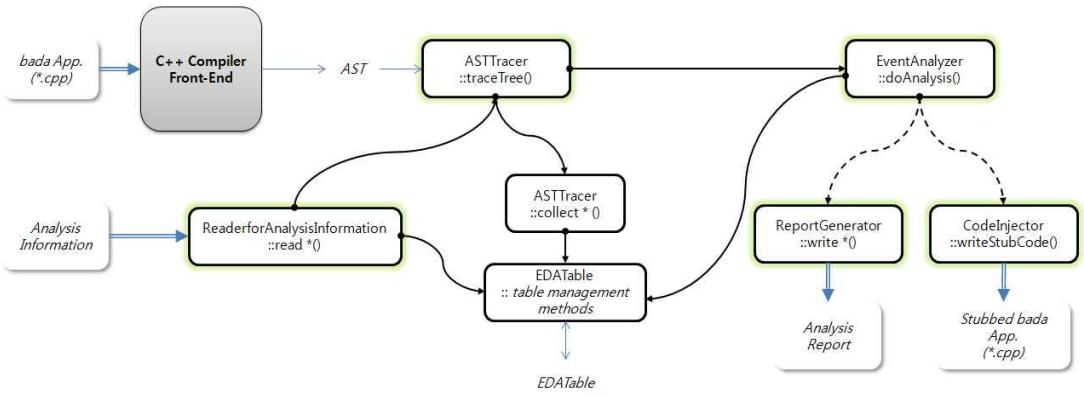

그림 11. TSG의 주요 호출 시퀀스

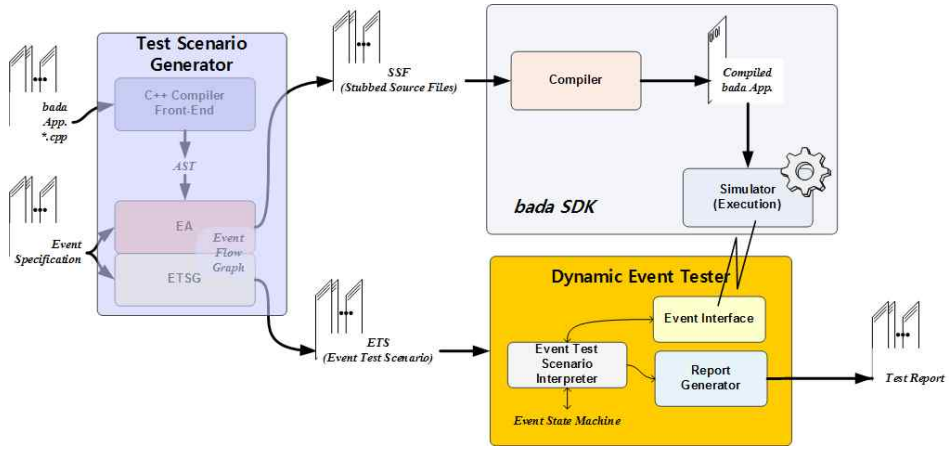

그림 12. DET의 구성도

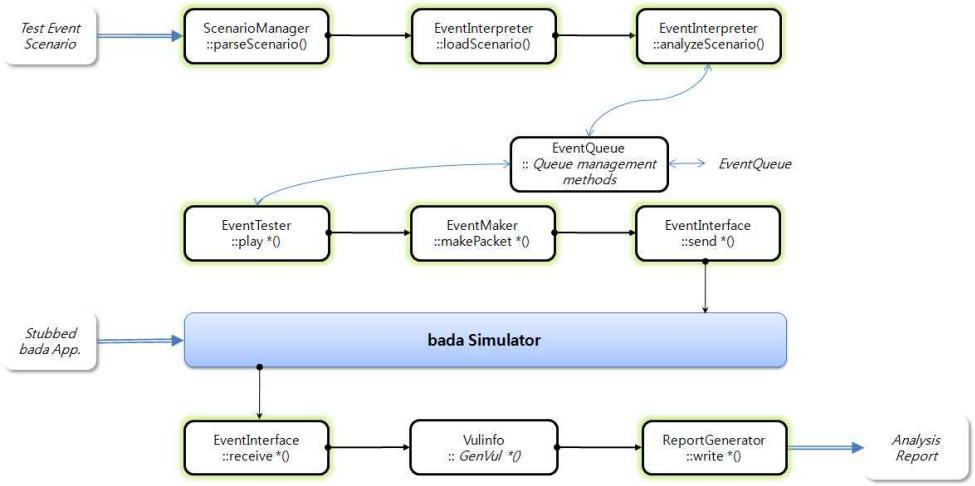

그림 13. DET의 주요 호출 시퀀스

테스트를 진행하기 위해 사용자가 액션을 취하게 되면, *EventTester* 클래스의 play *() 함수가 호출되어 이벤트 시퀀스를 bada 시뮬레이터에 전달한다. 이 과정에서 *EventMaker* 클래스의 makePacket *() 함수는 이벤트 시퀀스를 bada 시뮬레이터에서 해석 가능한 패킷 형태로 변환하고 send *() 함수는 실제로 시뮬레이터에 패킷을 송신하는 작업을 수행한다.

EventInterface 클래스의 receive *()는 테스트 과정에서 bada 시뮬레이터로부터 전달된 동적 정보를 수신하는 함수이다. 수신된 동적 정보는 프로그램의 취약점 정보이다. 따라서 GenVul *() 함수를 호출하여 취약점 객체를 생성하고 *ReportGenerator* 클래스의 write *() 함수를 통해 취약점 보고서를 생성한다.

5. 실험 및 결과

본 논문에서 제안한 모바일 애플리케이션을 위한 취약점 분석기의 모델에 대한 실험을 위해서는 대상 애플리케이션과 플랫폼을 결정해야 한다. 실험은 삼성전자에서 개발한 bada 플랫폼의 이벤트 관련 취약점 목록을 바탕으로 C++ 언어로 작성된 이벤트 구동 방식의 유틸리티 애플리케이션을 대상으로 하였다.

실험에서 사용한 애플리케이션은 bada SDK에서 제공하는 42개의 샘플 프로그램이며, 이는 개발자를 위한 가이드 형태로 제공하는 애플리케이션이기 때문에 다양한 기능과 이벤트에 대한 실험을 진행할 수 있는 특징을 가진다. 표 4는 샘플 프로그램을 기능

표 4. 샘플 프로그램의 분류

카테고리	이 름	LOC(Lines of Code)	카테고리	이 름	LOC
User Interface	AnimationApp	1,294	Graphics	DrawingBoard	1,313
	BasicApp	1,201		GlesCube	1,115
	FlashControl	963		GlesCube11	4,361
SNS	BuddyManager	2,705	Sensor	FaceRecognizer	1,316
	Contacts	3,950		HapticPlayer	841
	LifelogViewer	2,030		MetalDetector	840
	PrivacyManager	2,772		MotionDetector	336
	ProfileManager	3,137		RockPaperScissors	875
	Scheduler	3,782		TiltableImageViewer	426
	SimpleTwitter	2,262		WeatherMap	872
	SnsAuthManager	2,956	Map	LandmarkManager	2,007
Multi Media	CameraCapture	3,436		LocationManager	2,028
	MediaPlayer	2,636		MapControl	1,549
	VoiceRecorder	1,230		MapViewer	1,083
				Navigator	1,942
Graphics/Sensor/Map	Myplace	26,425			

에 따라 분류한 것이다.

42개의 샘플 프로그램을 사용하여 이벤트 구동 방식 프로그램을 위한 취약점 분석 도구를 검증하였다. 애플리케이션의 주요 그룹별로 대표적인 샘플 애플리케이션을 선택하여 취약점 분석을 수행한 결과를 요약하면 표 5와 같다.

여기서, 4번째 카테고리는 상용 애플리케이션으로 bada 플랫폼에서 제공하는 샘플에 비해 취약점이 많이 나타남을 확인할 수 있다. 그 이유는 샘플 애플리케이션은 개발자의 개발 효용성을 높이기 위한 모범적인 프로그램으로 안정적 검증을 거친데 비해 상용 애플리케이션은 많은 기능을 사용하고 프로그래머가 임의로 작성한 프로그램이기 때문으로 판단된다. 소스 코드의 크기에 따른 취약점 발생 빈도를 도표화하면 그림 14와 같다.

표 5. 주요 그룹별 샘플 애플리케이션의 취약점 통계자료

카테고리	취약점 수	취약점의 수 /LOC(K)	발생 비율
User Interface	28	23	2.3%
SNS	36	9	0.9%
Multimedia	19	7	0.7%
Graphics/Sensor/Map	7,335	277	27.7%

그림 14에서 알 수 있듯이 소스 코드의 크기가 1만 라인이상의 애플리케이션에서 취약점 발생 빈도가 기하급수적으로 증가함을 보이고 있다. 이러한 취약점 중에 이벤트 처리기의 기술 형식을 위반한 사례가 많은 비중을 차지하였고, 이는 bada SDK 개발사와의 의견 교류를 통해 수정 및 보완해야 할 필요성이 있다. 그림 15는 기존의 분석 도구와 분석 방법, 분석 내용, 분석 비용, 지원 언어에 대해 요약한 것이다.

본 논문에서는 모바일 애플리케이션의 특징을 반영한 취약점 목록을 도출하고 해당 취약점에 초점을 맞추어 분석을 수행하는 분석 도구를 제안하였다. 표

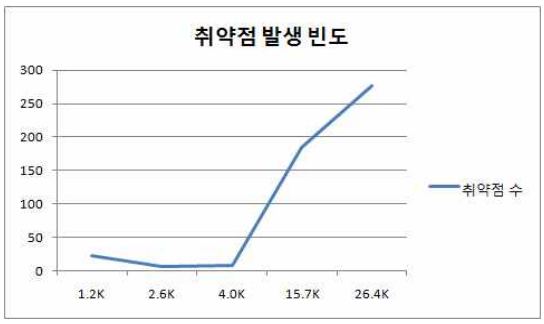

그림 14 소스 코드 크기에 따른 취약점 발생 빈도

표 6. 기존의 분석 도구 vs. 제안한 도구

	분석 방법	분석내용	분석비용	지원언어
MOPS	Static	Modeled Weakness	low	C
S-S C/C++	Static	Buffer Overflow	low	C/C++
Prevent	Static	API usage errors, Buffer overflow, Dangling stack references, Null pointer dereferences 등 16 그룹	mid	C/C++, COM, WIN32
Fortify 360	Static, Dynamic	OWASP TOP 10, CWE	high	C/C++, Java 등 12개
Sparrow	Static	Memory Error	low	C/C++
[17]의 도구	Static	WIN API Weakness	low	C++
제안한 도구	Static, Dynamic, Hybrid	Weakness for Mobile Application (bada apps.)	high	C/C++

6에서 나타난 바와 같이 각각의 도구는 서로 다른 취약점을 분석하기 때문에 취약점이 없는 견고한 프로그램을 개발하기 위해서는 한 가지 분석 도구에만 의존하지 말고 다양한 분석 도구를 활용해야한다. 따라서 제안한 도구는 애플리케이션의 전체 취약점 중 모바일 애플리케이션의 특징을 고려한 취약점을 분석하는데 기여할 수 있다.

6. 결론 및 향후 연구

최근 스마트 폰의 보급률 증가에 따라 기존에 고려하지 않았던 새로운 형태의 보안 침해 사고가 발생할 가능성이 있다. 보안 침해 사고의 예방을 위한 보안시스템은 네트워크 방화벽, 사용자 인증시스템 등에 치중되어 있는 반면, 소프트웨어 보안 침해사고는 가트너의 보고서에서 언급된 바와 같이 취약점을 내포하는 응용프로그램에 의해 발생되는 비율이 75%에 달한다. 따라서, 애플리케이션의 취약점을 사전에 검사하고 예방할 수 있다면 소프트웨어의 보안 침해 사고 발생률을 현저히 감소시킬 수 있다.

현재까지 응용프로그램을 대상으로 취약점을 탐지하는 방법은 대부분 고전적인 소프트웨어 테스트 방법론[20]과 테스트 자동화 도구에 의존한다. 이러한 방법론은 최근에 활성화되고 있는 모바일 애플리케이션의 특징인 이벤트 구동 방식 프로그램의 특징을 고려하지 않았으며, 테스트 시나리오의 작성에 필요한 전문 인력이 요구된다. 따라서 프로그램의 분석에 많은 시간과 노력이 소모되며, 시나리오의 질에 따라 프로그램 내에 존재하는 문제점의 발견 여부가 결정된다.

본 논문에서는 이벤트 구동 기반의 프로그램에서 발생할 수 있는 취약점 목록을 CWE, CERT의 취약점 목록과 시큐어 코딩 가이드(Secure Coding Guide)를 기준으로 도출하였으며 이를 기반으로 모바일 애플리케이션에 존재하는 취약점 검출하기 위한 동적 테스트 도구를 구현하였다. 구현된 동적 테스트 도구는 이벤트 기반 프로그램과 모바일 애플리케이션의 특성을 반영하여 테스트를 수행하며, 테스트 시나리오를 자동 생성하고 모니터 코드의 삽입 등을 이용하여 동적 분석을 수행하는 특징을 가지고 있다.

또한, 모바일 애플리케이션에 존재할 수 있는 취약점 목록을 제안하였으며, 이를 통해 취약점 분석 모델을 정립하고 모바일 애플리케이션의 테스팅 도구를 체계적이고 정형화된 방법으로 구성할 수 있게 되었다. 본 논문에서는 bada 플랫폼과 C++ 언어를 대상으로 적용한 취약점 분석 도구의 구현을 통해 제안한 모델을 검증할 수 있었다. 취약점 목록과 취약점 분석 도구는 개발 초기 단계에서 취약점을 사전에 제거함으로써 테스트와 유지보수의 비용을 크게 절감시킬 수 있으며, 오픈 마켓에서 제공하는 애플리케이션의 신뢰성 향상에 기여할 수 있다.

향후 연구로는 본 연구에서 도출한 취약점 목록을 추가 및 분류하고 오탐률과 검사 시간 측면에서 취약점 분석 도구의 성능을 개선하기 위한 연구가 필요하다. 또한, 모바일 애플리케이션을 위한 취약점 분석 도구의 멀티 플랫폼 지원 방안에 대한 연구가 진행되어야 하며, 소스 파일이 제공되지 않는 애플리케이션을 검증하기 위해 실행 파일을 통한 취약점 분석 기술에 대한 연구도 요구된다.

참 고 문 헌

[1] Gartner, Nov 2005, http://gartner.com

[2] A. B. Tucker and R. E. Noonan, *Programming Languages: Principles and Paradigms*, McGraw Hill, 2007.

[3] Gary McGraw, *Software Security*, Addison-Wesley, February 2006.

[4] John Viega and Gary McGraw, *Building Secure Software*, Addison-Wesley, September 2001.

[5] Common Weakness Enumeration(CWE), A Community-Developed Dictionary of Software Weakness Types, http://cwe.mitre.org.

[6] Richard Ford and Michael Howard, "Improving Software Security by Eliminating the CWE Top 25 Vulnerabilities," IEEE Security & Privacy, Vol.7, Issue 3, pp. 68-71, 2009.

[7] J. McManus and D. Mohindra, *The CERT Sun Microsystems Secure Coding Standard for Java*, CERT, 2009.

[8] Lockheed Martin Corporation, *Joint Strike Fighter: Air Vehicle C++ Coding Standards for The System Development and Demonstration Program*, 2005.

[9] MISRA, *Guidelines for The Use Of The C Language in Vehicle Based Software*, 1998.

[10] Y. W. Huang, F. Yu, C. Hang, C. H. Tsai, D. T. Lee, and S. Y. Kuo, "Securing Web Application Code by Static Analysis and Runtime Protection," Proceedings of the 13th Conference on World Wide Web, pp. 40-52, 2004.

[11] A.V. Aho, R. Sethi, and J. D. Ulman, *Compilers: Principles, Techniques, and Tools*, Addison Wesley, 2007.

[12] H. Chen and D. Wagner, "MOPS: an Infrastructure for Examining Security Properties of Software," Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 235-244, 2002.

[13] Plum Hall Inc., Overview of Safe-Secure Project: Safe-Secure C/C++, http://www.plumhall.com/SSCC_MP_071b.pdf.

[14] Coverity Inc., Coverity Static Analysis, http://www.coverity.com/products/static-analysis.html.

[15] Fortify Software Inc., Fortify Source Code Analysis(SCA), http://www.fortify.com/products/sca.

[16] Fasoo.com, About Sparrow, http://www.sparrow.com/.

[17] 하경휘, 김상영, 최진우, 우종우, 김홍철, 박상서, "안전한 소스코드 작성을 위한 자동화 분석 도구의 개발," 한국멀티미디어학회 추계학술발표대회논문집, pp. 980-983, 2003.

[18] Samsung Electronics, bada Developers, http://developer.bada.com.

[19] Ben Morris, Manfred Bortenschlager, Cheng Luo, Michelle Sommerville, and Jon Lansdell, *Introduction to bada: A Developer's Guide*, Wiley, 2010.

[20] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill, 2009.

문 일 룡

1995년 2월 충북대학교 컴퓨터공학과(학사)
2000년 8월 동국대학교 산업과학과(석사)
2007년 9월~현재 동국대학교 컴퓨터공학과(박사과정)
2004년 3월~현재 (주)어니컴 대표이사
관심분야 : 시큐어 코딩, 프로그래밍 언어, 모바일 컴퓨팅

오 세 만

1985년 3월~현재 동국대학교 컴퓨터공학과 교수
1993년 3월~1999년 2월 동국대학교 컴퓨터 공학과 대학원 학과장
2001년 11월~2003년 11월 한국정보과학회 프로그래밍 언어연구회 위원장
2004년 6월~2005년 12월 한국정보처리학회 게임연구회 위원장
관심분야 : 프로그래밍 언어, 컴파일러, 시큐어 코딩, 모바일 컴퓨팅