안티 패턴 기반의 안드로이드 애플리케이션 취약점 분석기법

이재용*, 최종석, 박상진, 신용태
*숭실대학교 컴퓨터학과
e-mail:jylee@icn.ssu.ac.kr

Vulnerability Analysis Scheme of Android Application based on Anti-patterns

Jae-Yong Lee*, Jong-Suk Choi, Sang-Jin Park, Young-Tae Sin
*Dept of Computer Science, Soong-Sil University

요 약

스마트폰이 대중화되면서 안드로이드 애플리케이션의 보안문제가 대두되고 있다. PC환경의 소프트웨어는 개발단계에서부터 시큐어코딩을 통해 안전성을 확보하고 있으나 안드로이드 애플리케이션의 경우는 연구가 더 필요한 상황이다. 본 논문에서는 CWE(Common Weakness Enumeration)의 취약점 분류 체계와 CAPEC(Common Attack Pattern Enumeration and Classification)의 공격 패턴 분류 체계와 CERT(Computer Emergency Response Team)의 취약점 발생 예방을 위한 정책들을 통해 안드로이드 애플리케이션의 최신화된 취약점 목록을 도출하고 카테고리별로 분류하여 취약점을 효율적으로 분석하는 기법을 제안한다.

1. 서론

최근 스마트폰의 대중화와 기존의 폐쇄적인 정책을 벗어나 소스코드를 오픈하는 개방적인 정책을 선택한 구글의 정책으로 인해 안드로이드 애플리케이션 마켓이 급성장하였다. 하지만 개방성으로 인해 악성코드의 유입이 쉬워졌고 단 기간에 걸쳐 개발된 안드로이드 애플리케이션은 기반이 되는 JAVA언어의 취약점까지 내포하고 있다.

본 논문에서는 안드로이드 애플리케이션의 취약점을 분석하기 위한 기법을 제안하고자 한다. 서론에 이어 2장에서는 취약점 분석을 위한 기법 및 분류체계에 대해 살펴보고 3장에서 제안하는 취약점 분석기법에 대해 기술한 후, 4장에서 결론을 맺는다.

2. 관련연구

2.1 안드로이드(Android)

안드로이드는 2008년 구글과 OHA(Open Handset Alliance)가 함께 제안한 오픈소스 플랫폼으로 운영체제, 미들웨어, 응용프로그램을 포함한다. 안드로이드는 오픈소스라는 개방성 및 JAVA를 통한 응용프로그램 개발과 같은 특징과 함께 제조사별로 차별성 있게 사용자 인터페이스를 제공할 수 있으며 컨텐츠 개발 및 배포가 가능한 특징이 있다.

2.2 시큐어코딩(Secure Coding)

최근 소프트웨어 자체에 내포된 보안취약점을 이용한 공격이 많이 발생하고 있으며 일반적인 보안장비로는 대응이 어려운 특성이 있다. 시큐어코딩이란 앞에서 언급한 사이버 공격을 예방하기 위하여 소프트웨어의 개발단계부터 보안취약점을 보완하여 공격을 차단하는 것으로 안전한 소프트웨어를 개발하기 위한 방법론이다.

2.3 소스코드 분석기법

<표 1> 동적 · 정적 분석

구분	동적분석 (Dynamic Analysis)	정적분석 (Static Analysis)
분석 방법	프로그램에 다양한 입력 값을 주며 실행 후 결과를 기반으로 분석	프로그램을 실행하지 않고 소스 코드의 내용을 기반으로 분석
장점	• 검출된 결함이 정확하며 회귀 테스트에 적합 • 결함 검출 시 입력 값을 바로 알 수 있으므로 결함 수정이 용이	• 테스트케이스를 작성하는 비용 없이 검사 가능 • 적은 비용으로 모든 실행 경로 검사 가능
단점	• 결함 검출 범위가 제한적 • 분석비용 및 구현복잡도가 높음	• 탐지오류(False Alram)의 발생 가능성

1 이 논문은 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단-차세대정보컴퓨팅기술개발사업의 지원을 받아 수행된 연구임(No.2012-0029927)
2 본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음(NIPA-2012-H0301-12-4008)

2.4 취약점 관리체계

MITRE는 미국 연방정부가 출연해 설립한 비영리 기관으로 지금은 세계를 넘나드는 대표적인 정보보안기관이다. 1999년 MITRE와 15개의 보안 관련 기관이 주도해 운영하는 CVE(Common Vulnerabilities and Exposures) database를 시작으로 CWE, CAPEC 등을 통해 다양한 자료를 제공하고 있다.

2.4.1 CWE(Common Weakness Enumeration)

세계 각국의 보안 업체에서 기준으로 사용되고 있는 취약점 분류 체계인 CWE는 1999년부터 MITRE를 중심으로 연구를 시작하여 2008년 9월 정식버전이 공개되어 계속 발전시켜나가고 있다.

CWE는 SQL인젝션, 크로스 사이트 스크립팅, 버퍼 오버플로우 등의 소프트웨어 취약점을 식별하기 위해 취약점 유형에 맞게 체계적으로 분류하여 제공하고 있다.

2.4.2 CAPEC(Common Attack Pattern Enumeration and Classification)

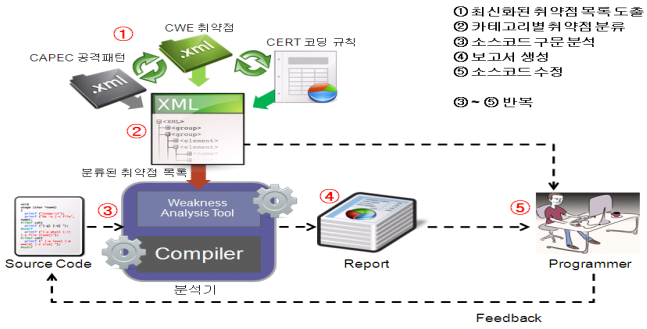
MITRE에서 CVE, CWE와 함께 제공하는 CAPEC은 보안취약점을 통한 공격 패턴의 형태와 실제 사례, 소프트웨어의 공격패턴에 대한 취약점 시험 및 검출 방법, 공격 패턴에 대한 대응방안 등을 정리 및 분류하여 제공하고 있다. CAPEC은 버전 1.0에서 시작해서 현재 버전 1.7까지 발전하였으며 버전 1.7에는 474개의 공격패턴이 기록되어 있다.

2.4.3 CERT(Computer Emergency Response Team)

CERT는 미국 연방정부의 자금을 지원받는 연구개발센터인 '소프트웨어 공학 연구소' 프로그램의 일부로 피치버그의 카네기 멜론 대학 내에 위치하고 있다. CERT는 worm으로부터 인터넷에 대한 공격이 발생한 직후인 1988년 11월 DARPA에 의해 만들어졌다.

오늘날 CERT는 보안상의 취약점에 초점을 맞추어 보안관련 사고처리 및 예방을 위한 정책수립과 코딩규칙 및 가이드의 표준화작업을 수행하고 있다.

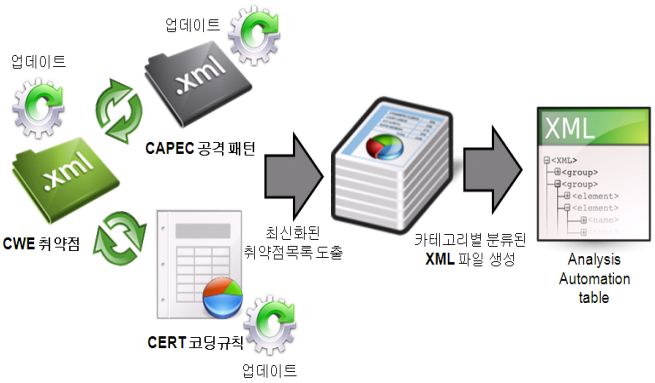
3. 안티 패턴 기반의 안드로이드 애플리케이션 취약점 분석기법



(그림 1) 취약점 분석 절차

3.1 취약점 목록 도출

취약점 분류체계인 CWE와 공격 패턴 분류체계인 CAPEC과 취약점 발생 예방 정책과 코딩 규칙 및 가이드의 표준인 CERT는 수시로 새로운 취약점 및 공격패턴 등의 정보들이 업데이트되며 빠르게 발전하고 있다. 한 시점에 머무르지 않고 추가된 정보를 반영하여 안드로이드 애플리케이션의 최신화된 보안 취약점 목록을 도출한다. (그림 2)는 최신화된 취약점 목록 도출 과정을 나타낸다.



(그림 2) 취약점 목록 도출 과정

가장 먼저 중심이 되는 CWE를 통해 발생가능한 공격패턴들을 CAPEC에서 도출하고 이러한 취약점 및 공격패턴들의 예빵을 위한 코딩규칙 및 해결방안과 함께 새로운 취약점들을 CERT와 관련하여 매팅되는 항목들을 도출한다. 도출된 취약점 목록들을 취약점 분석의 효율성을 높이기 위해 카테고리별로 분류 및 정리하여 제안하는 타입의 XML파일인 Analysis Automation Table을 생성한다. 이렇게 생성된 XML파일은 취약점을 찾기위해 소스코드 구문분석을 할 때 사용된다.

3.2 취약점 분류

취약점 분석의 효율성을 높이고 해당 공격유형에 대한 취약점 분석을 통하여 공격 예방이 용이하도록 도출된 취약점 목록을 카테고리 별로 체계적으로 분류한다. <표 2>는 제안하는 취약점 분류체계를 나타낸다.

<표 2> 취약점 분류체계

구분	내용	구분	내용
취약점 ID	취약점 고유 식별번호(C3-00)	관련 취약점	관련 취약점 정보 제공
취약점 명	취약점 명칭	관련 공격패턴	관련 공격패턴 정보 제공
분류	카테고리	해당 플랫폼	취약점 존재 가능 플랫폼
설명	취약점에 대한 상세한 설명	취약점 출처	도출한 취약점 출처
탐지방법	취약점 탐지 방법	업데이트 날짜	취약점 업데이트 날짜
해결방안	취약점 예방 및 완화 방법		

도출된 취약점에 고유 식별번호를 부여하여 구분하고 취약점을 탐지하기 위한 방법과 예방하기 위한 방법을 통해 취약점 발생을 최소화하며 관련된 CWE의 취약점과 CAPEC의 공격패턴을 함께 제공하여 연관 취약점 발생을 예방할 수 있다. 본 논문은 안드로이드에 국한되어 있지만 향후연구의 범위를 확장하기 위해 해당 플랫폼 항목을 제공하며 취약점의 업데이트날짜를 참고하여 필요에 따라 업데이트를 수행한다. 기존의 분류체계에서는 한부분에 대해서만 다루는 반면에 본 논문에서 제안하는 분류체계는 관련 취약점과 공격패턴 및 해결방안을 함께 제공한다.

취약점 분류 카테고리는 CWE/SANS의 가장 위험한 25가지 소프트웨어 오류를 참고하여 현재 가장 치명적인 공격유형에 따른 효율적인 취약점 분석을 통해 취약점 발생을 최소화시키며 공격을 예방할 수 있다.

3.3 취약점 분석

취약점 분석을 위해 소스코드를 파싱하여 취약점을 탐지하기 위한 문장을 추출하고 추출된 문장과 도출된 취약점 목록을 비교 분석하여 취약점을 탐지한다. (그림 3)은 취약점 분석 수행 과정의 의사코드를 나타낸다.

```
Weakness_Analysis_Function(src, ans_tab)
{
        src 파싱

        while 취약점 분석이 끝났는가?
                if 도출된 취약점 목록 ans_tab의 취약점 w가 검출되었는가?
                        w 검출 알림 표시
                end if
        end while
}
```

(그림 3) 취약점 분석 수행과정의 의사코드

취약점 분석함수는 분석대상 소스코드와 도출된 취약점 목록을 받아 분석을 수행한다. 임의의 취약점 w가 검출되면 프로그래머가 확인할 수 있도록 표시하여 소스코드 수정을 통해 취약점을 예방할 수 있다.

4. 결론

스마트폰의 대중화로 인한 안드로이드 애플리케이션 마켓의 급격한 성장은 새로운 보안문제를 야기하였다. 이러한 보안문제는 대부분 안드로이드 애플리케이션의 취약점으로부터 발생되고 있다.

본 논문에서는 안드로이드 애플리케이션의 취약점 분석을 효율적으로 하기위해 취약점 목록 도출의 최신화와 도출된 취약점 목록의 체계적인 분류를 통한 취약점 분석기법을 제안하고자 한다. 향후 안드로이드 애플리케이션에 국한되지 않고 다양한 플랫폼에 대해 더 효율성 높은 기법에 대한 연구를 진행함으로써 취약점 발생을 최소화하려 한다.

참고문헌

[1] 문일룡, 오세만, “모바일 애플리케이션을 위한 취약점 분석기의 설계 및 구현” Journal of Multimedia Society Vol. 14. No. 10. October 2011, pp. 1335-1347

[2] 최윤희, 최은만, “안티 패턴 기반의 정적분석을 이용한 안드로이드 어플리케이션 취약점 분석” 정보과학회논문지 컴퓨팅의 실제 및 레터 제18권 제4호 2012. 4, pp. 316-320

[3] CWE, http://cwe.mitre.org

[4] CAPEC, http://capec.mitre.org

[5] CERT, http://www.cert.org