JavaCC를 이용한 정적 분석 도구 구현

김병철O, 김창진*, 윤성철*, 한경숙*
O*한국산업기술대학교 컴퓨터공학부
e-mail: altorious153@gmail.comO, {dhgoak45, sungchul09}@naver.com*, khan@kpu.ac.kr*

Implement static analysis tool using JavaCC

Byeong-Cheol KimO, Chang-Jin Kim*, Sung-Chul Yoon*, Kyung-Sook Han*
O*Dept. of Computer Engineering, Korea-Polytechnic University

● 요 약 ●

본 논문에서는 JavaCC를 사용하여 보안약점을 찾아내는 도구를 구현한다. 이 도구는 구문 정보와 흐름 정보를 이용하여 프로그램 소스 코드에서 보안약점을 찾아 진단한다. 이를 통해 컴파일러 단계별 얻을 수 있는 정보와 관련 보안약점을 연계함으로써 보안약점에 대한 진단 방법과 분석 정보의 연계를 기대할 수 있다.

키워드: 보안약점(weakness), 정적 분석(static analysis), 구문 정보(syntax information), 흐름 정보(flow information)

I. Introduction

보안약점(weakness)은 공격자가 원하는 정보를 유출, 탈취 가능하거나 제어 흐름을 바꿀 수 있도록 하는 프로그램의 불완전한 부분을 의미한다. 보안약점을 공격자가 이용하여 실제 보안 사고가 발생하면 그 보안약점을 취약점(vulnerability)이라고 한다. 보안약점을 찾기 위한 방법이 많이 개발되고 있는데, 그 중 한 가지 방법이 정적 분석(static analysis)[1]이다. 정적 분석이란 프로그램의 실행 없이 소프트웨어를 분석하는 것을 말한다.

본 논문에서는 구문 정보(syntax information)와 흐름 정보(flow information)[2]를 이용하여 JavaCC[3]로 정적 분석 도구를 구현하였다.

본 논문의 2절에는 보안약점을 진단하기 위한 관련 연구에 대해 기술하고, 3절에서 CERT 사이트가 기술하고 있는 코딩 규칙 중 위험도가 높은 코딩 규칙을 진단 방법에 따라 분류한다. 4절에서 JavaCC로 구현된 내용을 설명하고, 5절에서 결론을 맺는다.

II. Related works

CERT[4]는 CWE[5]의 보안약점을 예방할 수 있는 코딩 규칙을 설명과 예제를 통해 기술하였다.

정적 분석 도구는 프로그램 소스코드를 검토하여 공격자로 부터 중요한 데이터나 정보에 대한 접근을 최소화 할 수 있도록 돕는다. 대표적인 예로 fortify SCA[6]와 sparrow[7]가 있다.

정적 분석 도구의 성능을 검증할 수 있도록 CWE에 기술된 보안약점을 토대로 다양한 코드로 작성된 테스트 모음이 존재한다. 대표적인 예로 Juliet 테스트 모음이 있다.[8]

또한 본 논문에서 사용한 JavaCC는 어휘 생성기와 구문 생성기를 포함하고 있고, 자바 언어를 기반으로 만들 수 있는 컴파일러 컴파일러이다.

III. Classify

CERT에 기술된 Java 관련 코딩 규칙 중 위험도가 높은 코딩 규칙을 선정하여, 진단 방법이 구문 정보 또는 흐름 정보인 것으로 분류하였다. Table 1.은 분류한 6개의 코딩 규칙을 보여준다.

구문 정보는 어휘 정보인 토큰을 이용하여 생성된 구문 트리(syntax tree) 상에서 트리 노드의 존재 유무나 위치를 의미한다.

흐름 정보는 제어 흐름(control flow)과 데이터 흐름(data flow) 두 가지 정보가 있다. 제어 흐름은 문장 사이의 실행 순서를 의미하고, 데이터 흐름은 제어 흐름을 기반으로 데이터가 직접적 또는 간접적으로 영향을 미치는 관계를 의미한다.

Table 1. Weakness and Diagnostic Information

Syntax Information	Flow Information
OBJ01-J. Limit accessibility of fields	IDS00-J. Prevent SQL injection
ERR07-J. Do not throw RuntimeException, Exception, or Throwable	IDS01-J. Normalize strings before validating them
ERR08-J. Do not catch NullPointerException or any of its ancestors	MSC03-J. Never hard code sensitive information

IV. Implement

본 절에서는 JavaCC를 기반으로 Table 1.의 보안약점을 진단하는 도구를 구현하였다.

구문 정보를 이용하는 경우 JavaCC에서 추상 구문 트리(abstract syntax tree)상의 구조를 분석하는 형태이다. 예를 들어 “Do not catch NullPointerException or any of its ancestors”에 대한 진단 방법으로 try-catch 구문 트리 상에서 catch의 자식 노드 값을 NullPointerException과 비교한다. 해당 예외와 같다면 코딩 규칙 위반으로 진단하고 코딩 규칙의 이름과 작성된 코드의 위치 정보를 출력한다.

흐름 정보를 이용하는 경우 JavaCC를 통해 얻은 구문 정보와 라인에 대한 정보를 취합하여 배열리스트에 저장한다. 이후 보안약점이 될 수 있는 함수들의 리스트를 저장하고, 배열 리스트를 탐색하면서 해당 함수 이전과 이후에 보안약점이 되는 함수 또는 구문을 찾는다. Fig. 1.은 흐름 정보를 이용한 진단 방법을 간단히 보여준다. 예를 들어 “Prevent SQL Injection”은 executeQuery 함수의 존재 여부를 확인 후 배열리스트에 저장된 해당 라인의 구문 정보를 이용하여 자식 노드에 있는 값이 일반적인 SQL문으로 쓰여 있으면 1차적으로 진단한다. 그렇지 않고 변수로 쓰여 있으면 저장되어 있는 라인 정보를 이용하여 해당 변수를 찾는다. 그 변수가 Statement 클래스로 선언되었다면 외부의 변수에 따라 SQL문이 생성되므로 2차적인 진단을 하고 구문 정보를 이용한 진단과 마찬가지로 코딩 규칙의 이름과 코드의 위치 정보를 출력한다.

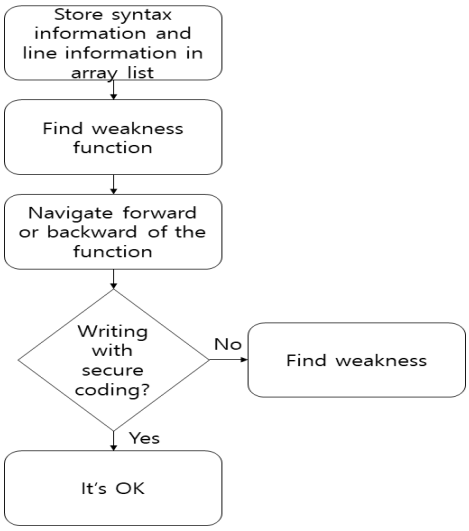

Fig. 1. Diagnostic Algorithm using Flow Information

V. Conclusions

본 논문에서는 보안약점을 진단할 때 사용하는 여러 가지 정보 중 구문 정보와 제어 흐름 정보를 이용하여 JavaCC로 보안약점을 진단하는 도구를 구현하였다. 이를 통해 각 컴파일러 단계에서 얻을 수 있는 정보와 보안약점을 연계함으로써 보안약점에 대한 진단 방법과 프로그램 분석 정보의 연계를 기대할 수 있다.

향후 다른 분석 정보를 이용하여 진단하는 도구를 구현하는 방법을 연구하는 방향으로 확장할 것이다.

REFERENCES

[1] SungMoon Hong, Seungcheol Shin, Kyung-Goo Doh, Detection of Security Vulnerability From the Knowledge-Base Representation of Source Code, Journal of The Korea Information Science Society pp.1618-1620, June 2014.

[2] Kyungsook Han, Damho Lee, Changwoo Pyo, Classification of Diagnostic Information and Analysis Methods for Weaknesses in C/C++ Programs, Journal of The Korea Society of Computer and Information Vol. 22 No. 3, pp. 81-88. March 2017.

[3] JavaCC, *https://javacc.org/*

[4] CERT, Computer Emergency Response Team, *https://wiki.sei.cmu.edu/confluence/*

[5] CWE, Common Weakness Enumeration, *https://cwe.mitre.org/*

[6] MICRO FOCUS Inc., *https://software.microfocus.com/*

[7] Sparrow Co., https://sparrowpasso.com/

[8] Juliet test-suite, *https://samate.nist.gov/SRD/testsuite.php/*