안전한 코딩을 위한 정적 C 코드 분석기 개발

류두진 성시원 김덕헌 한익주
한국산업기술대학교 컴퓨터공학과
e-mail : {rdj850*,sangza40,ahasma,ijhan}@kpu.ac.kr

An Implementation of Static C – Code Analyzer for Secure Coding

Doo-Jin Ryu Si-Won Sung Deok-Heon Kim Ik-Joo Han
Dept. of Computer Engineering, Korea Polytechnic University

요 약

최근 Application 의 취약성을 악용한 해커들의 시스템 공격 사례가 증가하고 있다. 본 논문에서 다루는 코드 분석기는 이러한 해커의 공격을 사전에 차단하기 위해 사용자로부터 입력받은 Application 의 소스 코드가 사전에 탑재해 놓은 일련의 보안 규칙(Security Rule)을 제대로 준수하는지의 여부를 어휘 분석(Lexical Analysis)과 구문 분석(Semantic Analysis)을 통해 판별해 낸다. 본 코드 분석기는 미국 카네기멜론대학(CMU) 산하의 인터넷 해킹 보안 기구인 CERT 에서 제시하는 규칙을 그대로 적용하여 분석 결과의 정확도와 객관성을 높였으며, 이 분석기를 통해 프로그래머가 신뢰도와 보안성이 높은 소프트웨어를 개발할 수 있도록 하였다.

1. 서 론

해커의 공격에 의한 기업 및 관공서의 피해가 갈수록 심각해지고 있다. 최근에는 각종 안티바이러스 프로그램의 방어에 의해 웜(Worm)이나 바이러스(Virus)에 의한 피해 사례는 줄어들고 있는 반면, 피싱 및 내부 침입, 변조 시도 등 시스템의 취약점을 적극적으로 공격하는 방식에 의한 피해 사례는 대폭 증가된 것으로 보고된 바 있다[1]. 특히 Application 의 취약점을 이용한 공격이 전체 공격의 75%를 차지했다[2].

한편, 시스템이 해커의 공격에 의해 파괴되었다면 그 문제를 수습하기 위해 시스템 관리자들은 적지 않은 시간과 비용을 소비해야만 한다. 일반적으로 소프트웨어 내부적 문제가 발생했을 때 복구 및 유지보수에 드는 비용은 그 문제를 설계 및 구현 단계에서 미리 방어했을 때 드는 비용의 2 배 이상이라고 알려져 있다[3].

본 연구를 통해 개발된 분석기는 소프트웨어 구현 과정에서 소스 코드 내에 향후 보안에 위배가 될 수 있는 이상 징후가 있는지의 여부를 분석 및 감지하는 역할을 수행한다. 본 분석기를 통해 소프트웨어의 취약성을 최소화하여 해커의 공격을 사전에 차단할 수 있고, 따라서 시스템의 사후 관리에 드는 시간과 비용을 절약할 수 있게 된다.

본 연구에서는 프로그래머가 사용하기 편리하면서도 정확한 결과를 제시하고 시스템 자원을 효율적으로 사용할 수 있는 분석기를 개발하는 것을 목표로 한다. 또한 정적 분석기(Static Analyzer)로서 보안 규칙이 사전에 탑재되어 있어야 하는데, 이 보안 규칙은 미국 카네기멜론대학(CMU)이 후원하고 있는 세계 최대의 인터넷 침해 대응센터 CERT 의 규칙을 그대로 적용하였다. CERT 에서는 해커의 침입 가능성을 제거한 소프트웨어 개발 문법을 7 개 영역(전처리기, 선언, 연산, 문자열, 예외처리, 입출력, 배열 및 포인터)에 걸쳐 자세하게 명시하고 있다[4].

본 연구에서는 CERT 에서 제시한 보안 규칙을 중요도에 따라 분류, 가장 심각한 문제가 발생할 수 있는 사항을 선별하여 우선적으로 분석기에 내장하였다.

분석기는 내장된 규칙을 기반으로 입력된 소스 코드의 어휘 분석(Lexical Analysis) 및 구문 분석(Semantic Analysis)을 수행하고, 그 결과를 log 리포트로서 출력하게 된다.

2. 관련 연구

2.1 보안 코딩 (Secure Coding)

보안 코딩은 소프트웨어의 개발 단계부터 소스 코드의 보안 취약점을 고려해 프로그래밍하는 기법이다. 마이크로소프트 등 글로벌 소프트웨어 회사들은 보안 개발 라이프사이클(Security Development Lifecycle)에 기반해 보안 취약점을 최소화한 소프트웨어를 개발하고 있지만, 국내 도입은 아직 미비한 상황이다.

최근 행정안전부는 보안 코딩 도입을 위해 전자정부법과 정보 시스템의 효율적 도입 및 운영 등에서 적용 근거를 검토하고 있다.

한편 미국 정부는 전자정부의 정보보호를 위해 피즈마(FISMA)법을 제정, 보안 코딩을 의무화하고 있다. 실제로 마이크로소프트의 경우 보안 코딩 기법을 도입해 개발한 윈도 비스타(Windows Vista)가 XP 에 비해 보안 위협을 45% 감소시켰고 소프트웨어 오류 수정에 드는 비용도 크게 줄었다고 보도한 바 있다[5].

이처럼 보안 코딩 기법에 대한 연구는 소프트웨어 개발을 위한 새로운 영역으로 발전해 나가고 있다.

2.2 코드 분석기 (Code Analyzer)

코드 분석기는 프로그래머가 개발한 코드에 대해 특정한 문제가 존재하는지의 여부를 판별하는 소프트웨어 검사 도구의 일종이다.

현재 개발되어 상용화되고 있는 정적 코드 분석기는 포티파이 SCA(Fortify Source Code Anaysis) 4.0, 클록워크(Klockwork) K7.5, 온스 랩스(Once Labs) 4.1 이 대표적이다. 이들 분석기는 각기 상이한 표준과 문법을 채택하고 있으며, 특히 소스 코드가 지니고 있는 논리적 취약성을 발견하지 못한다는 문제점이 있다[6].

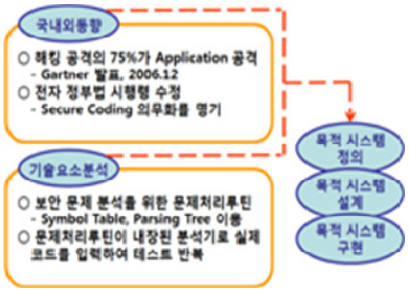
코드 분석기 개발 분야는 지금도 연구가 활발하게 진행되고 있으며, 특히 각 개발사마다 소프트웨어의 설계상 또는 논리적 취약성을 자동으로 분석해내기 위한 노력이 계속되고 있다.

3. 개발환경

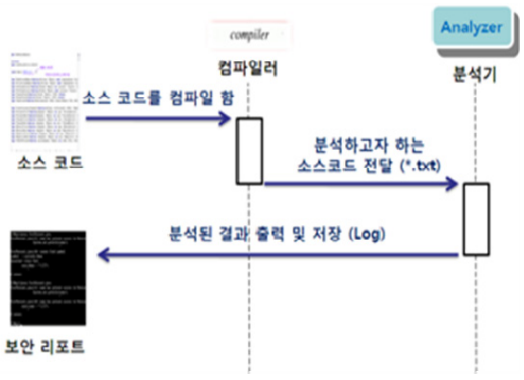
본 논문에서 제시한 코드 분석기는 Unix 와 Linux 운영체제 상에서 동작하는 어휘 분석기 생성기(LEX, Lexical Analyzer Generator) 및 파서 생성기(Parser Generator)를 이용하여 구현하였다.

분석기에 적용된 C 의 문법은 BNF 형식으로 명시된 C95 표준을 적용하였으며[7], 보안 규칙은 CERT 가 제시하고 있는 문법을 준수하였다.

[그림 1]은 보안 코딩에 대한 국내외 동향 및 기술 요소 분석 내용을 도식화하였고, [그림 2]는 코드 분석기의 시퀀스 다이어그램을 나타낸 것이다.



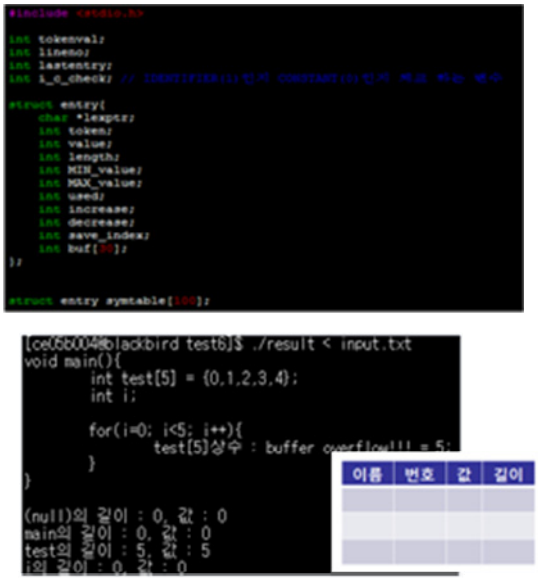
(그림 1) 국내외 동향 및 기술요소 분석



(그림 2) 시퀀스 다이어 그램

4. Symbol Table 과 Semantic Analysis

심벌 테이블(Symbol Table)이란 심벌(또는 명칭)들에 관한 정보를 관리하고 운영하는 데 사용되는 자료구조이다[8]. 본 코드 분석기에서는 심벌 테이블이 어휘 및 구문 분석 단계에서 소스 코드에 대한 정보(Attribute)를 수집하고, 그 정보가 분석기에 내장된 보안 규칙과 일치하는지의 여부를 판독하기 위한 용도로 사용되었다. [그림 3]은 분석기에 사용된 심벌 테이블의 형태와 실제로 정보가 어떻게 저장되고 있는지를 보여주고 있다.

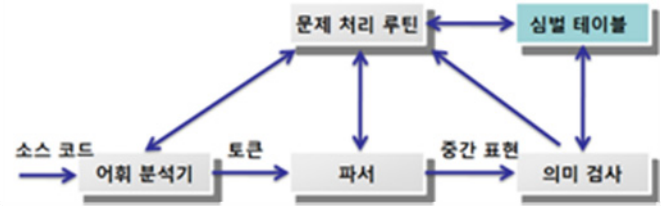


(그림 3) 심볼테이블에 저장되는 정보

심벌 테이블의 효율적인 구성은 전체적인 분석 시간에 많은 영향을 준다. 본 분석기는 보안성 검사가 플래그(Flag) 방식으로 이루어져 테이블의 크기가 비교적 작다. 따라서 데이터 삽입이 동적, 순차적으로 이루어지고 선형 검색(Linear Search)을 수행하는 비정렬 방식의 심벌 테이블을 구현하였다[9].

5. 문제처리 루틴 (Problem-Handling Routine)

심벌 테이블에 저장된 정보는 문제처리 루틴을 통해 내장된 보안 규칙과 비교된다. 문제처리 루틴은 보안 위배 항목 발생 여부를 심벌 테이블에 반영시키고, 그 문제를 log 리포트로 출력하는 기능을 수행한다. [그림 4]는 분석기 내에서의 문제처리 루틴의 기능을 보여준다.

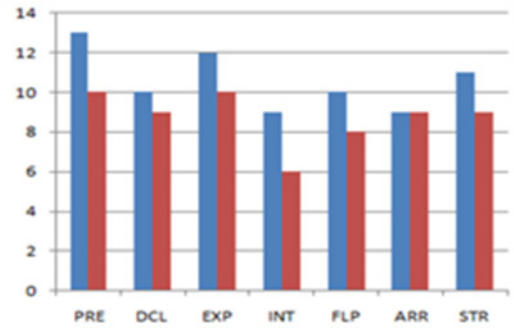


(그림 4) 문제 처리 루틴의 처리과정

문제처리 루틴은 분석기를 이루고 있는 LEX 와 YACC 내에 내장된 보안 규칙과 결합한 명령(Action Code)를 수행함으로써 동작한다. 즉, 문제처리 루틴은 입력받은 소스 코드로부터 보안에 위배되는 구문이 발견될 경우 처리해야 하는 일련의 함수들의 집합이다. 보안 문제가 토큰 단계(Token Phase)에서 발생하는 경우 액션 코드는 LEX(lex.yy.c)에 기술되며, CERT 에서는 주로 전처리 및 선언 규칙에 해당된다. 한편 보안 문제가 파서 단계(Parser Phase)에서 발생하는 경우 액션 코드는 YACC(y.tab.c)에서 기술되며, CERT 에서는 연산 등 주로 논리적인 규칙에 해당된다. [그림 4]에서 보는 것처럼 심벌 테이블은 코드 분석 전 단계에 걸쳐 문제처리 루틴과 유기적으로 연결되어 있으며, 보안 문제가 발견될 때마다 그 결과를 순차적으로 기록한다.

6. CERT 보안 적용 범위

본 논문의 코드 분석기는 CERT 가 제시한 7 개 영역별 위험도가 가장 높은 보안 규칙 74 개 항목 중에서 61 개 항목을 분석할 수 있다.



(그림 5) CERT 보안 규칙(좌),분석 가능한 규칙(우)

(그림 6) 코드 분석기의 기능

7. 결론 및 고찰

위에서 제시한 것처럼 본 논문에서는 CERT 가 가장 심각한 문제를 발생시킬 수 있다고 지목한 주요 리스트 중 60%가 구현되어 있다. 또한 향후 보안 규칙 추가 및 변경에 따라 언제든지 확장이 가능하도록 설계되어 있다.

이러한 코드 분석기는 C 프로그래머를 대상으로 소스 코드에 보안 문제가 잠재하는지의 여부를 사전에 발견 및 조치하기 위한 Unix/Linux 용 개발 도구(Utility)로써 유용하게 사용될 것으로 기대된다. 또한 시스템 관리자는 코드 분석을 통해 견고하게 작성된 소프트웨어를 운용함으로써 해커의 공격을 차단, 시스템의 유지 보수에 소요되는 시간과 비용을 절감할 수 있을 것이다.

[참고문헌]

[1] 한국 인터넷침해 대응센터 (http://www.krcert.or.kr)
인터넷침해사고 동향 및 분석 월보 (2010.7)

[2] IT 리서치 Gartner (http://www.gartner.com)
해킹 유형 분석 결과 (2006.12)

[3] Ian Sommerville, 소프트웨어 공학 8 판, 보안공학 (2008.2)

[4] 카네기멜론대학 CERT (http://www.cert.org)
Secure Coding Standards (2010)

[5] ETNews (http://www.etnews.co.kr)
전자정부용 SW 보안 규격 제정 검토 (2009.10)

[6] DataNet (http://datanet.co.kr)
간편한 SW 개발, 만병통치약 아니다 (2007.11)

[7] C95 표준(BNF Form)
(http://www.csci.csusb.edu/dick/samples)

[8] 오세만, 컴파일러 입문 (개정판)
심벌 테이블의 기능 (2008.1)

[9] 오세만, 컴파일러 입문 (개정판)
심벌 테이블의 운영 (2008.1)