새로운 언어 설계의 지침을 위한 C 시큐어 코딩 규칙 분류

김연어, 우 균
부산대학교 전자전기컴퓨터공학과
e-mail:{yeoneo, woogyun}@pusan.ac.kr

Categorizing C Secure Coding Rules for a Design Guideline of a New Language

Yeoneo Kim, Gyun Woo
Dept. of Electrical and Computer Engineering, Pusan National University

요 약

현대 사회에서 정보보안은 무엇보다 중요한 요소로 자리 잡고 있다. 시큐어 코딩은 정보보안 기법의 하나로 보안 취약점을 원천적으로 차단하여 보안 비용을 획기적으로 줄이는 방법이다. 하지만 기존 시큐어 코딩 가이드는 C나 Java와 같은 특정 언어에 대한 가이드만 제공하고 있다. 이 논문에서는 다양한 언어에서도 기존의 시큐어 코딩 가이드를 활용할 수 있도록 언어적 특징을 기반으로 시큐어 코딩 가이드를 재분류하고자 한다. 이를 위해 이 논문에서는 많은 언어의 기반이 되는 C 언어의 시큐어 코딩 가이드 중 안전행정부에서 발표한 C 시큐어 코딩 가이드를 이용하여 재분류 작업을 수행하였다. 그 결과 총 58개의 취약점 중 언어와 관련이 있는 취약점은 19개로 약 33%가 프로그래밍 언어와 관련 있는 것을 확인하였다. 또한, 제안 방법의 내용 중 언어적 특성쪽의 취약성을 모두 해결할 수 있도록 문법을 설계한다면 C 언어보다 보안성이 높은 언어를 설계할 수 있다.

1. 서론

최근 우리 사회는 개인 정보 유출과 같은 정보 보안의 문제로 큰 몸살을 앓고 있다. 특히 최근 신용 카드 3사의 허술한 보안관리로 인한 개인 정보 유출은 사회적, 경제적 측면에서 큰 손해를 입었으며, 아직 많은 사람이 이야기하고 있는 사건이다[1]. 이처럼 현대 사회에서는 개인의 정보를 안전하게 지키는 것이 중요한 문제로 떠오르고 있다.

가트너의 발표에 의하면 대부분의 보안 침해 사건의 약 75%는 응용 프로그램의 취약점을 이용하여 발생하고 있다[2]. 특히 보안상의 버그는 약 50%가 프로그램 코드 수준에서 발생하고 있다[3]. 이처럼 대부분의 프로그램 취약점은 프로그램 작성단계에서 생긴다. 그러므로 프로그램 설계나 코드 작성 단계에서 보안을 고려해야 안전한 프로그램을 작성할 수 있다. 최근 이러한 문제점을 해결하기 위해 시큐어 코딩(secure coding)이라는 개념이 등장하였다.

시큐어 코딩은 코드 작성 시 프로그램 작성 단계에서 문제가 될 수 있는 코드를 미리 예방하여 보안 침해 공격을 막는 방법이다[4,5]. 이 방법은 국내외에서 새로운 보안 수단의 하나로 주목받고 있으며 기존의 보안이 뚫린 다음 해결하는 방법이나 네트워크 시설에 투자하는 방법에 비해 비용을 많이 절감할 수 있다. 이에 발맞춰 최근 정부에서는 공공기관에 납품하는 20억 이상의 소프트웨어의 경우 안전행정부에서 제안하는 시큐어 코딩 지침을 준수하도록 하고 있다[6,7].

하지만 현존하는 모든 프로그래밍 언어에 대해서 시큐어 코딩 가이드를 준수하는 것은 어려운 일이다. 수많은 프로그래밍 언어가 사용되고 있으며, 지금 현재도 새로운 프로그래밍 언어는 계속 개발되고 있지만 이에 대응하는 시큐어 코딩을 습득하는 것은 어렵다. 또한, 모든 프로그래밍 언어에 대해 시큐어 코딩 가이드가 존재하지 않은 사실도 문제점이다.

이 논문에서는 시큐어 코딩 가이드를 언어적 특성을 이용하여 새로이 분류하고자 한다. 기존 시큐어 코딩 가이드는 기능적 특성을 이용하여 취약점을 분류하고 있기 때문에 새로운 언어에 대한 시큐어 코딩 가이드를 작성하게 되면 가이드 내용 일부가 겹치게 되는 경우가 발생할 수 있다. 그러므로 이 논문에서는 많은 언어의 기반이 되는 C 언어를 대상으로 하며, 안전행정부에서 발표한 C 시큐어 코딩 가이드를 언어적 특성으로 분류하고자 한다[6].

이를 위해 이 논문에서는 2장에서 관련 연구로 시큐어 코딩에 대해 살펴본다. 그리고 3장에서는 연구 배경 및 기존 C 시큐어 코딩 가이드가 어떤 방식으로 취약점을 분류하고 있는지 살펴본다. 4장에서는 언어적 특징을 이용해서 취약점을 분류하는 방법과 실제 C 시큐어 코딩 가이드의 내용을 제안하는 방법으로 분류해본다. 마지막으로 5장에서 결론을 맺는다.

이 논문은 2013년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(NRF-2013R1A1A4A01006704)

2. 관련 연구

현재 많은 프로그램은 위험에 노출된 상태로 서비스되고 있다. 이는 최근 소프트웨어 제공 형태가 단독 실행에서 네트워크를 통한 서비스를 제공하는 방식으로 변경되었기 때문이다. 따라서 프로그램은 ㅜ항상 악의적인 사용자의 공격을 고려하고 있어야 한다. 즉 악의적인 사용자로부터 프로그램이나 개인 정보를 보호하는 것이 무엇보다 중요한 과제가 되었다. 하지만 네트워크나 인증을 통하여 철저히 정보를 지키는 것에도 한계가 존재한다. 또한, 약 75%의 보안 침해 사건이 응용 프로그램 수준의 취약점에 의해 발생하기 때문에 응용 프로그램 수준에서 보안을 지키는 것이 중요하다. 그래서 이를 어떻게 해결할지가 중요한 이슈로 떠오르고 있으며, 이를 해결하는 방법으로 최근 제시된 것이 시큐어 코딩이다.

시큐어 코딩은 프로그램 개발 과정 중 소스코드를 작성하는 단계에서 취약점을 제거하는 방법이다[6,7]. 시큐어 코딩은 입력 검증부터 잘못된 API 사용까지 프로그래밍 과정 중 발생할 수 있는 다양한 취약점을 해결하는 방법과 안전한 프로그램 작성 방법을 다루고 있다. 시큐어 코딩은 개발 단계에서 보안 취약점을 원천적으로 차단하는 방법이기 때문에 네트워크 보안이나 사고 후 사고를 수습하는 다른 방법에 비해 비용을 절약할 수 있다.

국내외에서는 이러한 장점 때문에 시큐어 코딩에 대해 활발히 연구가 이루어지고 있다. 대표적으로 국외의 경우 소프트웨어 보안의 중요성을 인지하고 CERT(Computer Emergency Response Team), CWE(Common Weakness Enumeration)와 같은 기관에서 소프트웨어 보안 취약점 수집하고 이를 해결하는 방안을 제시하고 있다[9,10]. 그리고 국내의 경우 2009년부터 전자정부서비스 개발단계에서 시큐어 코딩이 고려되고 있으며, 2012년에서는 안전행정부에서 정부 납품 소프트웨어에서 시큐어 코딩 규칙을 준수하도록 하는 법안을 고시하였다[7].

3. 연구 배경 및 기존 분류 방법

3.1 연구 배경

현재 우리가 주로 사용하고 있는 C 언어나 Java와 같은 대부분의 프로그래밍 언어는 보안을 고려하지 않고 설계된 언어이다. 초창기의 프로그래밍 언어는 정확한 결과와 빠른 성능, 그리고 편리한 기능에 초점에 맞추어져 있었기 때문에 악의적인 사용자로부터 공격은 고려하지 않고 설계되었다. 하지만 개인 정보가 중요한 사회가 되면서 이 정보를 다루는 소프트웨어의 보안은 중요한 쟁점이 되었지만 프로그래밍 언어는 이러한 쟁점을 따라가지 못하고 있다. 이 때문에 언어적 수준에서 보안을 막는 방법으로 시큐어 코딩 가이드가 개발되었다.

하지만 시큐어 코딩 가이드는 많은 사람이 사용하는 특정 언어로만 제공되고 있다. 대표적인 예로 안정행정부에서는 C 언어와 Java를 대상으로 한 시큐어 코딩 가이드만 제공하고 있다[6,8]. 따라서 프로그램을 개발할 때 반드시 시큐어 코딩 가이드를 지켜야 한다면 이는 개발 언어의 제약사항이 될 것이다.

그러므로 다른 언어에서 시큐어 코딩 가이드의 코딩 규칙을 적용하기 위해서는 (1) 다른 언어용 시큐어 코딩 가이드가 마련될 때까지 기다리거나 (2) 기존 시큐어 코딩 가이드의 기법을 바탕으로 다른 언어의 코딩 방법을 수정하는 것을 생각해 볼 수 있다. 이 중에서 선택 (2)를 추구하려면, 기존 언어의 시큐어 코딩 가이드를 분석하여 비언어적 규칙을 추출해내야 할 것이다.

또한, 시큐어 코딩 가이드를 언어적 특성을 이용하여 분류한다면 새로운 언어를 설계하는 과정에 사용할 수 있다. 보통 새로운 언어를 설계하는 경우 기반이 되는 언어를 하나 정하여 설계하는 경우가 많다. 이 때 기반이 되는 언어의 시큐어 코딩 가이드가 언어적 특성으로 분류되어 있다면 언어 설계에 활용하여 안전한 프로그래밍 언어를 설계할 수 있다.

3.2 기존 분류 방법

기존의 시큐어 코딩 가이드에서는 취약점을 기능적 특성을 이용하여 분류하고 있다. 예를 들어 SQL 삽입 공격과 같은 취약점은 입력이 잘못되어 발생하는 문제이기 때문에 입력 검증으로 분류하고 있다. 그리고 메모리를 잘못 해제하여 발생하는 문제는 코드의 문제이기 때문에 코드 오류로 분류하고 있다. 즉 기존의 시큐어 코딩 분류는 기능적 특성을 기반으로 나누고 있으며 이 논문에서 대상으로 하는 C 시큐어 코딩 가이드의 취약점 분류 현황은 표 1과 같다.

표 1 C 언어 시큐어 코딩 가이드 취약점 분류표. 안전행정부에서 발표한 시큐어 코딩 가이드에 수록된 취약점 분류표로 총 58개의 취약점을 수록하고 있으며 7개의 분류로 나누어져 있다.

분류 기준	취약점수	분류 기준
입력 데이터 검증 및 표현	19	프로그램의 입력 데이터가 잘못되어 취약한 경우
보안기능	17	인증과 관련된 정보를 처리할 때 취약점이 발생하는 경우
시간 및 상태	3	시간적 개념이나 시스템 상태정보 처리에서 취약점이 발생하는 경우
에러 처리	3	에러 처리 시 취약점이 발생하는 경우
코드 오류	9	프로그램이 복잡하여 취약점이 발생하는 경우
캡슐화	2	의도하지 않은 데이터나 불필요한 데이터로 인해 취약점이 발생하는 경우
API 오용	5	과도한 API 사용이나 취약점이 있다고 알려진 API를 사용하는 경우

기존 C 시큐어 코딩 가이드는 표 1과 같이 기능적 특성을 7가지 기준으로 분류하고 있다. 취약점이 발생하는 형태나 기능에 따라 입력 데이터나 에러 처리와 같이 분류하고 있다. 이러한 기능적 분류는 이미 작성된 프로그램에서 보안 취약점을 찾아서 해결하기에는 유용하지만 하나의 언어에만 적용 가능하다는 한계가 있다. 따라서 다양한 프로그래밍 언어에 직접 적용하기 어려우며 적용한다고 하더라도 고려해야 할 사항이 많아지는 단점이 있다.

4. 언어 설계를 위한 분류 방법

이 논문에서는 기능적 특성에 따라 분류된 C 시큐어 코딩 가이드가 아닌 언어적 특성에 따라 C 시큐어 코딩 가이드규칙을 분류하고자 한다. 제안 방법이 C 시큐어 코딩을 고른 이유는 현재 많이 사용되고 있는 많은 프로그래밍 언어는 C 언어를 기반으로 하고 있기 때문에 C 시큐어 코딩 가이드가 가장 널리 활용될 수 있다고 판단되기 때문이다. 제안하는 방법은 표 2와 같이 취약점을 분류한다.

표 2 언어 설계를 위한 C 시큐어 코딩 분류 방법. 새로운 시큐어 코딩 분류 방법은 언어적 특징과 플랫폼, 라이브러리, 로직으로 나뉘며 언어적 특징은 세부적으로 다시 3가지 항목으로 나누어진다.

대분류	세부분류	특징
언어적	예외 처리	예외 처리를 통해 해결될 수 있는 취약점
	타입	타입 변환 규칙이나 특수 한정자를 통해 해결할 수 있는 취약점
	포인터	포인터를 통해 발생하는 취약점
플랫폼		시스템 API와 같은 플랫폼 취약 함수나 플랫폼 정보 누출을 통해 발생하는 취약점
라이브러리		API 사용 시 보안에 취약한 방법으로 호출거나 위험하다고 알려진 API를 사용하는 취약점
로직		알고리즘과 같은 프로그램 로직 문제로 발생할 수 있는 취약점

다른 분류 중 플랫폼은 플랫폼에 의존적인 시스템 API나 시스템 정보를 프로그램 내부에서 사용하지만 잘못된 사용이나 유저의 권한을 확인하지 않고 시스템 정보를 전달하는 것과 같은 문제점을 의미한다. 그리고 라이브러리의 경우 사용이 제한된 라이브러리는 아니지만, 라이브러리 호출을 잘못하여 보안 정보가 유출되거나 취약점이 생기는 경우를 의미한다. 마지막으로 로직은 라이브러리나 플랫폼의 취약점은 아니지만, 프로그래머가 프로그램을 작성 시 로직을 잘못 작성하여 발생하는 문제점을 의미한다. 이러한 표 2의 분류 기준을 이용해 C 시큐어 코딩 가이드를 재분류하면 표 3과 같이 분류된다.

표 3 언어적 특성을 이용한 C 시큐어 코딩의 분류. C 시큐어 코딩 가이드의 내용을 언어적 특성을 이용해 분류한 결과로 총 58개의 취약점을 특징에 맞게 분류하였다.

대분류	세부분류	기존 규칙	규칙수
언어적	예외 처리	입력데이터 검증(7), 에러 처리(1)	8
	타입	입력데이터 검증(2), 보안 기능(4), 코드 오류(2)	8
	포인터	코드 오류(3)	3
플랫폼		입력데이터 검증(4), 캡슐화(1)	5
라이브러리		입력데이터 검증(6), 보안 기능(3), API 오용(4)	13
로직		보안 기능(10), 시간 및 상태(3), 에러 처리(2), 코드 오류(4), 캡슐화(1), API 오용(4)	25

안전행정부에서 발표한 C 시큐어 코딩 가이드의 취약점 58개를 제안하는 분류방법으로 재분류한 결과, 표 3과 같은 결과를 도출할 수 있었다. 언어적 특성으로 처리 가능한 취약점은 19개가 발견되었다. 예외 처리의 경우 총 8개의 취약점이 포함되었다. 예외 처리는 대표적인 예로 배열 처리에서 인덱스가 배열 크기보다 크거나 음수가 되는 예가 있다. 이 분류는 예외 처리 구문을 지원하는 Java나 C#에서는 문제가 되지 않지만, C에서는 문제로 발생하는 부분이 포함되었다.

타입의 경우 총 8개의 취약점이 포함되었다. 타입의 대표적인 예는 묵시적 타입 변환이 발생하여 원래의 값에서 값이 변경되거나 정수와 문자 간의 변경과 같이 의도치 않은 값으로 변경되어 문제가 발생하는 경우이다. 이 경우는 묵시적 타입 변환이나 타입 변환이 명확한 언어에서는 문제점이 발생하지는 않지만, C 언어에서는 문제로 발생하는 경우이다.

포인터의 경우 총 3개의 취약점이 포함되었다. 포인터의 대표적인 예로 함수의 반환 값으로 스택 변수의 주솟값을 반환하거나, 널 포인터를 참조하여 발생하는 문제가 대표적이다. 이러한 문제는 포인터를 지원하지 않는 언어에서는 발생하지 않기 때문에 언어적 특성으로 분류하였다.

플랫폼의 경우 총 5개의 취약점이 포함되었다. 대표적으로 외부 입력값을 검증하지 않고 시스템 자원에 접근하는 문제인 자원 삽입 공격이 있다. 이러한 문제는 언어에 종속적인 문제가 아니므로 플랫폼으로 따로 분류한 경우이다.

라이브러리의 경우 총 13개의 취약점이 포함되었다. 라이브러리의 대표적인 예는 검증되지 않은 외부 입력으로 동적 웹 페이지나 스크립트를 실행하여 문제가 발생하는 크로스사이트 스크립트(Cross-Site Scripting, XSS)가 있다. 라이브러리의 경우 동일한 라이브러리를 여러 언어에서 제공하는 경우도 있기 때문에 언어에 종속적인 문제가 아니므로 따로 분류하였다.

마지막으로 로직의 경우 총 25개의 취약점이 포함되었

다. 로직의 대표적인 예는 재귀 함수에서 재귀 지점이 제대로 제어되지 않는 경우나 무한 자원 할당과 같은 문제가 있다. 로직의 경우 언어적 특성도 아니며, 플랫폼, 라이브러리 사용의 문제가 아닌 프로그래머가 프로그램을 작성하면서 범하기 쉬운 로직 문제로 많은 취약점이 여기에 속하게 된다.

전체 분류 결과 총 58개의 취약점 중 언어적 특성으로 발생하는 취약점은 19개로 측정되었으며, 플랫폼은 5개, 라이브러리는 10개, 로직은 25개로 재분류되었다. 이 분류 결과 C 언어의 컴파일러에서 언어적 문제점을 지원한다면 약 33%의 취약점을 해결할 수 있다. 그리고 시큐어 코딩 가이드가 제공되지 않는 언어에 대해 새로이 작성할 경우 언어적 특성 외의 취약점은 다른 언어에서도 다시 사용할 수 있을 것으로 판단된다.

5. 토의

C 언어에서 포인터와 관련된 취약점을 해결하기 위해서 포인터 문법을 제거한다면 포인터와 관련된 취약점은 해결될 수 있다. C 언어에서 포인터를 제거하게 된다면 메모리 해제 오류나 널 포인터 참조와 같은 실행 중 치명적인 오류는 사라지게 된다. 하지만 C 언어에서 포인터를 사용하지 못한다면 C 언어의 표현력이 제한받게 되는 문제가 있다. 그러므로 C 언어에서 포인터를 제거하기 위해서는 비슷한 표현을 할 수 있는 문법과 힙 영역의 메모리를 관리할 수 있는 수단이 추가돼야 C 언어에서 포인터를 제거하여 취약점을 제거할 수 있을 것으로 판단된다.

또한, C 언어에서 예외 처리와 관련된 취약점을 해결하기 위해서는 Java나 C++와 같은 try catch 예외 처리 문법을 추가해야 한다. 이러한 예외 처리 문법이 추가되어야 배열의 인덱스 오류 취약점이나 오버플로우 문제도 해결할 수 있을것으로 판단된다. C 언어의 경우에는 Java와 같이 Exception 클래스를 상속받아 예외 처리를 하지 못하기 때문에 비트 마스킹을 이용하여 예외를 처리해야 할 것으로 생각된다.

제안 방법은 C 언어를 기반으로 한 새로운 프로그래밍 언어 설계에도 도움이 된다. 제안 분류 기준 중 비언어적인 특성은 언어에서 해결할 수 없지만, 언어적인 특성은 언어에서 특정 문법을 지원할 수 있다면 거의 다 해결할 수 있다. 즉 제안 방법을 이용하여 언어를 설계한다면 기존 C언어 대비 33% 이상의 언어의 보안성이 높아진다고 볼 수 있다.

6. 결론

이 논문에서는 기존의 C 시큐어 코딩 가이드를 언어적 특성을 기반으로 새로이 재분류하였다. 이를 위해 이 논문에서는 언어적 특성 3가지를 제안하고 취약점을 분류하였으며, 이외의 취약점으로 플랫폼, 라이브러리, 로직 방법을 제안하고 취약점을 분류하였다. 그리고 C 시큐어 코딩 가이드의 58개의 취약점 중 약 33%의 취약점이 언어적 특성에 의해 발생하는 취약점임을 확인하였다.

향후 연구로서 새로이 분류한 C 시큐어 코딩 가이드를 이용하여 C#이나 Go언어와 같은 새로운 언어에 대한 시큐어 코딩 가이드를 작성해볼 예정에 있다. 그리고 현재는 C 시큐어 코딩 가이드를 대상으로 언어적 특성을 기반으로 분류하였지만 향후 Java 시큐어 코딩 가이드도 같은 기준으로 분류해볼 예정이다. 그리고 두 가이드를 비교해볼 예정이다.

참고문헌

[1] 강은성, “사상최대 1억명 카드정보 유출 ‘파문’,” 디지털타임스, 2014.

[2] Xie B., Zhang Q., “Application-layer Anomaly Detection Based on Application-layer Protocols' Keywords,” 2012 2nd International Conference on ICCSNT, IEEE, pp. 2131-2135, 2012.

[3] Shahriar H. and Zulkernine M., “Mitigating program security vulnerabilities: Approaches and challenges,” ACM Computing Surveys 44(3), Article 11, pp. 1-46, 2012.

[4] McGraw G, *Software security: building security in*, Addison-Wesley Professional, 2006.

[5] Viega J., McGraw G., Building *secure software: how to avoid security problems the right way*, Pearson Education, 2001.

[6] 이경진, 전자정부 SW 개발 운영자를 위한 C 시큐어 코딩 가이드, 안전행정부, 2012.

[7] 김경애, “시큐어코딩 법적의무화 체크유형 7가지,” 보안뉴스, 2014.

[8] 이경진, 전자정부 SW 개발 운영자를 위한 Java 시큐어코딩_가이드, 안전행정부, 2012.

[9] Carnegie Mellon University, “Secure Coding | The CERT Division,” https://www.cert.org/secure-coding, 2014년 03월 20일 방문.

[10] MITRE, “CWE - Common Weakness Enumeration.” https://cwe.mitre.org, 2014년 03월 20일 방문.