23-1-R04

소스코드 보안취약성 자동진단도구 개발 사례

Development of a Source Code Security Vulnerability Analyzer

권현준 · 김유경* · 김현하* · 도경구* · 신승철 · 안준선*** · 이욱세* · 이은영**** · 한환수*****

지티원 · 한양대학교* · 한국기술교육대학교** · 한국항공대학교*** ·
동덕여자대학교**** · 성균관대학교*****

hjkweon@gtone.co.kr · {yukoung, hhkim, doh, oukseh}@hanyang.ac.kr · scshin@kut.ac.kr,
jsahn@kau.ac.kr · elee@dongduk.ac.kr · hhan@skku.edu

요 약

본 논문은 2009년도 행전안전부의 주관 및 지원 하에 한국인터넷진흥원을 중심으로 (주)지티원, (주)파수닷컴, 한국정보보호학회 소프트웨어보안연구회에서 개발한 소스코드 보안취약성 자동진단 도구 개발 사례를 소개한다. 본 과제와 관련하여 규칙 기반 취약성 분석엔진이 개발되었고, 보안 취약성 데이터베이스가 구축되었다. 개발된 소프트웨어는 전자정보시스템 취약성 분석에 적용됨으로써 국가 정보시스템의 보안 안전성 강화에 활용될 것이다.

1. 서 론

컴퓨팅 시스템의 보안문제의 해결에 있어서 소프트웨어 소스코드 수준의 보안성 강화를 통한 근원적인 접근법에 관심이 증가하고 있다. 최근 들어 응용프로그램의 취약성을 사용한 공격의 빈도가 사이버 위협의 주요 원인이 되고 있는 것으로 보고되고 있으며, 거의 모든 종류의 정보서비스가 불특정 다수가 직접 접근할 수 있는 인터넷을 통해 제공되는 상황에서 소프트웨어 소스코드 수준의 보안성 강화가 중요한 관심사로 등장하고 있다. 또한 2009년 7월 7일 국내에서 발생한 대규모 DDoS(분산서비스거부) 공격과 같은 소프트웨어 시스템에 대한 공격 피해사례가 전 세계적으로도 증가하는 추세이며, 이러한 사례의 근원적인 원인도 대부분 소프트웨어 소스코드의 보안 취약성에서 찾을 수 있는 것으로 밝혀지고 있다.

코드리뷰를 통하여 소스코드의 버그 또는 보안 취약성을 찾아내는 작업이 이루어지고 있지만, 개인의 숙련도에 의해서 품질이 좌우되고 경제적이지도 못하므로 자동으로 찾아주는 도구가 필요하다. 또한 다양한 보안 취약성 각각에 대하여 검출 방법과 공격 방법에 대한 방지책을 일일이 마련하여야 하므로 취약성에 대한 다양한 정보를 구축하는 작업 또한 필수적이다.

이와 관련하여 미국에서 fortify와 같은 보안 취약성 도구가 개발되어 산업현장에서 이미 활용되고 있으며[1] 국내적으로는 메모리 오류를 자동으로 검출해주는 정적분석시스템인 Sparrow가 파수닷컴(http://www.fasoo.com)에 의해서 상용화되었고[2], 보안 취약성을 야기하는 소스

코드 내의 구문 패턴과 흐름 패턴을 규칙 명세에 기반하여 검출하는 CodePrism이 지티원((http://www.gtone.co.kr)[3]에 의하여 개발된 바 있다.

이러한 필요성에 부응하여 정부에서도 행정안전부를 중심으로 전자정부 소프트웨어 개발 시에 보안 취약성에 대한 검사를 의무화하는 정책을 추진하고 있다. 이러한정책의 시행을 위해서는 현재까지 알려진 보안 취약성과 검사 방법을 데이터베이스화하고, 이를 바탕으로 소프트웨어를 자동 검사하는 도구를 개발해야 한다. 뿐만 아니라 보안 취약성의 지속적인 갱신 관리체계 구축, 검사의 의무화를 위한 법률 제정, 소프트웨어 개발 시에 취약성이 발생하지 않도록 개발하는 방법(안전한 코딩 표준; secure coding standard)의 보급과 교육 등도 병행되어야 한다. 본 논문은 2009년도 행전안전부의 주관 및 지원 하에 한국인터넷진흥원을 중심으로 (주)지티원, (주)파수닷컴, 한국정보보호학회 소프트웨어보안연구회에서 개발한 소스코드 보안취약성 자동진단 도구의 개발 사례를 소개하고자 한다. 본 과제와 관련하여 규칙 기반 취약성 분석엔진이 개발되었고, 분석엔진과 프로그램 개발자, 검수자 등이 사용할 수 있는 보안 취약성 데이터베이스가 구축되었다.

본 논문의 전체적인 구조는 다음과 같다. 2장에서 분석도구의 전체적인 구조와 분석엔진에 대하여 기술하고 3장에서는 취약성 데이터베이스의 구축에 대하여 기술한다. 4장에서는 개발된 시스템의 활용을 위한 진단 체계에 대하여 설명하며, 5장에서 결론으로 맺는다.

2. 소스코드 취약성 분석 엔진

2.1 취약성 진단도구의 전체 구조

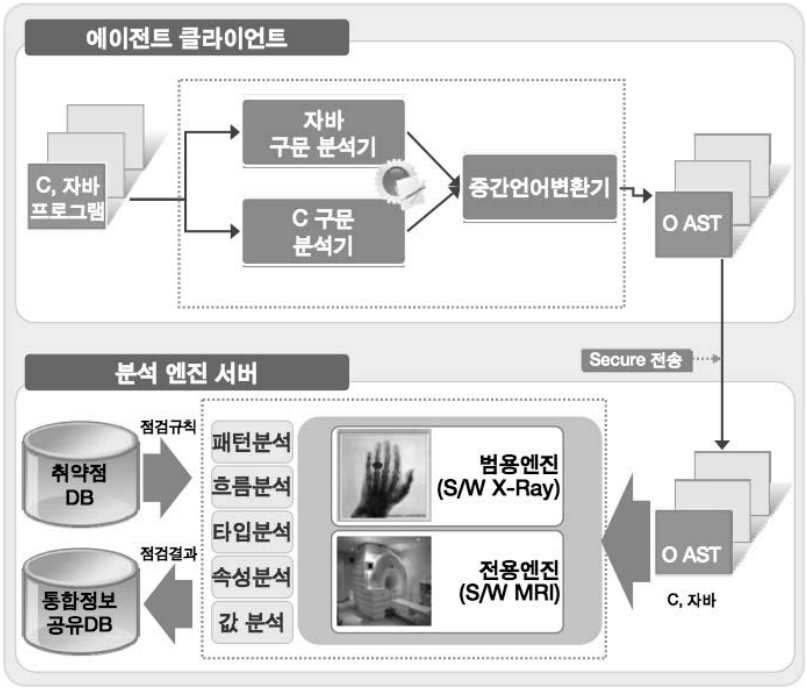
소스코드 취약성 자동진단도구는 [그림 1]과 같이 클라이언트에서 수행되는 1차 분석과 분석엔진서버에서 수행되는 2차 분석으로 나누어진다. 1차 분석에서는 C와 Java 소스를 파싱한 후, 분석에 필요한 구문 및 의미 정보를 망라해서 가지고 있는 중간코드로 변환한다. 소스코드 유출의 우려를 불식시키기 위해 클라이언트 쪽에서 중간코드로 변환하고 암호화하여 분석엔진서버로 전송한다. 중간코드는 유출이 된다 하더라도 형식 및 저장방식이 비공개이므로 중간코드에서 소스코드의 복원은 원천적으로 불가능하다. 2차 분석은 각 취약성의 특성 및 난이도에 따라서 두 가지 방법으로 나누어서 분석한다. 범용엔진은 단순한 구문 패턴이나 흐름 패턴으로 표현할 수 있는 다양한 취약성을 자체개발한 규칙명세언어로 기술하여 목록을 구축하고 이 목록에 수록된 패턴과 일치하는 취약한 부분을 일괄적으로 찾아낸다. 전용엔진은 단순한 패턴으로 표현할 수 없는 취약성에 대해서, 취약성 별로 고안된 분석기가 별도로 구현된 진단 엔진이다. 특정 취약성에 대해서 탐지의 정밀도를 높일 필요가 있는 경우에 전용엔진으로 진단하기도 한다.

개발된 소스코드 취약성 분석엔진은 현재 총 259개의 취약성을 진단할 수 있는데, 이 중에서 Java 취약성은 163개, C 취약성 96개이다. Java의 경우 119개의 취약성이 범용엔진으로, 44개 취약성이 전용엔진으로 진단되고, C의 경우 64개 취약성이 범용엔진으로, 32개 취약성이 전용엔진으로 진단된다.

2.2 범용 분석엔진

범용엔진은 일정한 구문 또는 흐름 패턴으로 취약성을 찾을 수 있는 분석엔진이다. 구문 또는 흐름 패턴은 규칙명세언어 RDL(Rule Description Language)로 기술할 수 있으며, 이 언어로 기술된 취약성 패턴취약성 패턴은 취약성 DB로 유지되고 수시로 갱신할 수 있다.

범용엔진은 진단 대상이 되는 소스코드의 중간코드와 RDL로 기술된 검사규칙목록을 입력으



[그림 1] 소스코드 취약성 분석 엔진

로 받아서, 취약성을 탐지한다. 중간코드는 생성 과정에서 모아둔 타입정보와 같은 프로그램의 기본적인 요소들을 가지고 있으며, 이러한 정보들은 분석에서 사용된다. 상수분석이나 널값여부 분석 등 간단한 수준의 값분석은 성능을 위해서 미리 수행되며, 단일패턴 분석기는 입력받은 규칙명세목록을 단일패턴단위로 쪼개서 각 단일패턴들의 위치와 부가정보들을 모은다. 흐름분석을 위해서 중간코드는 흐름그래프로 변환되고, 흐름패턴 분석기는 단일패턴들의 지점들과 흐름그래프로 흐름패턴으로 명시된 단일패턴들 사이의 관계를 분석하여 취약성 진단 결과를 생성한다.

2.3 전용 분석엔진

전용엔진은 패턴을 RDL로 기술할 수 없거나 정밀분석이 요구되는 경우 취약성별로 개별적으로 구축되는 분석엔진이다. 일반적으로 분석엔진의 정밀도를 높이기 위해서는 복잡한 분석을 감수해야 하고 따라서 분석시간이 많이 걸린다. 따라서 정밀도와 분석시간 사이에 적절한 조정이 필요하다. C의 경우 전용분석으로 처리된 32개의 취약성 중에서 18개 취약성은 값 분석을 자세히 수행하는 Sparrow 엔진을 기초로 제작되어서 정밀도가 이미 상당히 높다. C의 나머지 14개 취약성과 Java의 44개 취약성은 아직 정밀도 향상의 여지가 많이 남아있으며, 공학적이며 경제적인 측면에서 균형유지를 위한 적절한 의사결정이 필요하다.

3. 보안 취약성 데이터베이스

3.1 전체 구조

취약성 데이터베이스는 기본 취약성 관련 정보인 취약성 테이블, 언어별 관련 취약성 정보와 검출 규칙을 정리한 취약성 규칙 테이블 및 취약성과 관련한 공격 패턴을 정리한 공격패턴 테이

블로 구성된다. 본 취약성 데이터베이스는 CWE, CVE, CAPEC 등의 국외 관련 연구 내용[5,6,7,8,9,10]과 행안부 발주 주요 소프트웨어의 특성 및 업계 관련 제품의 성능 등을 참고하여 구축되었으며 xml 형식으로 정리되었다.

취약성 테이블은 취약성 식별자, 설명, 중요도, 관련 언어, 분류, 관련 공격 패턴, 참고문헌 등의 정보로 이루어져 있으며 KCWE(Korea Common Weakness Enu- meration) 일련번호를 부여하였다. 취약성 규칙 테이블은 현재 C 및 Java와 관련된 취약성 각각에 대하여 취약성 규칙 식별자, 취약성 예제 및 수정 예제와 관련설명, 취약성 검출 규칙 및 규칙 설명 등으로 이루어져 해당 언어의 프로그램 개발 시 참고할 수 있다. 또한 취약성 검출 규칙은 RDL로 기술되어, 프로그램 취약성 자동 분석 엔진의 입력으로 사용된다. 공격 패턴 테이블은 공격 패턴 식별자, 공격 패턴 제목, 관련 공격 도구, 공격 패턴에 대한 취약성 테스트 방법 및 취약성 회피 방법 등의 정보를 담고 있다. 취약성 테이블의 각각의 취약성에 대해서는 해당하는 언어별 취약성 규칙이 1:n으로 연결되며, 취약성과 공격 패턴에 대해서는 n:n의 관계로 연관된다.

3.2 보안 취약성 분류

취약성 데이터베이스를 구축함에 있어서 본 연구에서는 취약성들을 원인 및 특성에 따라 분류하여 구축 작업을 수행하였다. 본 절에서는 이러한 분류에 따라 분석 대상이 되는 보안 취약성에 대하여 간단히 설명한다.

3.2.1 검증되지 않은 값의 사용(Data Handling)

본 취약성 분류는 검증되지 않은 부적절한 값이 보안에 민감한 작업에 사용됨으로써 발생하는 취약성을 말한다. 부적절한 값이란 외부의 불특정 입력이나, 범위를 벗어나는 주소값 및 첨자(index) 값, 각각의 자료형(types)의 범위를 벗어나는 값 등이 되며, 민감한 작업이란 메모리 접근, 명령어 수행, 경로값을 사용한 파일(file) 접근, 데이터베이스 접근, 동적 웹페이지의 생성, 형식 문자열(format string)을 사용하는 입출력문 등을 들 수 있다.

본 취약성 분류에 속하는 대표적인 취약성으로는 SQL 삽입 공격(SQL Injection Attack, KCWE-89) 취약성, 사이트 교차접속 스크립트 공격 취약성(Cross Site Scripting, KCWE-79), 메모리 버퍼 범위 조건 위반(KCWE-119) 등이 있다.

3.2.2 API 사용 오류 (API Abuse)

API 사용 오류란 개발환경에서 제공하는 시스템 라이브러리 중 자체에 보안취약성이 있는 것을 사용하거나, 라이브러리에 대한 이해 부족으로 잘못 사용하여 보안취약성을 발생시키는 것을 말한다.

대표적인 API 사용 오류로는 위험하다고 알려진 함수 사용 (KCWE-242), 함수 결과 검사 부재(Unchecked Return Value, KCWE-252), super.finalize()를 호출하지 않는 finalize() 메소드(KCWE-568) 등이 있다.

3.2.3 보안 관련 오류(Security Features)

보안성과 관련된 오류란 패스워드 관리를 담당하는 프로그램이나 암호화 등을 이용하는 과정에서 발생한 프로그램 상의 오류가 시스템 전체를 취약하게 만드는 것을 말한다. 여기에는 사용자 암호를 정하지 않고 사용하는 경우나, 프로그램 내부에서 문자열 상수로 패스워드를 저장하고 그대로 사용하는 경우, 혹은 이미 취약성이 알려진 암호화 기법을 사용하는 경우 등이 포함된다. 보안성과 관련된 오류들은 오류가 발생하

면 사용자 아이디나 패스워드 등이 노출되면서 시스템 전체의 기밀성이나 무결성이 침해되는 경우가 발생할 수 있다. 이 때문에 패스워드를 상수의 형태로 노출하거나 잘못된 암호화 알고리즘을 사용하는 오류는 발생빈도는 낮더라도 발생하는 경우, 침해의 결과는 치명적일 수 있다.

대표적인 보안성 관련 오류로는 코드에 고정된 패스워드(Hard-coded Password, KCWE-259), 취약한 암호화 알고리즘의 사용(KCWE-327), 주석문 안에 포함된 패스워드(KCWE-9302), 적절하지 않은 난수 값의 사용(KCWE-330), 보안 속성이 결여된 HTTPS 세션으로 보내지는 민감한 내용의 쿠키 (KCWE-614), 익명의 LDAP 바인딩(KCWE-9311) 등이 있다.

3.2.4 시간과 상태 관련 오류(Time and State)

시간과 상태에 대한 오류란 프로그램의 동작 과정에서 시간적 개념을 포함한 개념(프로세스, 혹은 쓰레드 등)이나 시스템 상태에 대한 정보(자원 잠금이나 세션 정보)에 관련된 오류를 말한다.

대표적인 시간과 상태에 관련된 오류로는 외부에서 제한 없이 접근 가능한 잠금(KCWE-412), 중복 검사된 잠금(Double-Checked Locking, KCWE-609), 세션 고착(Session Fixation, KCWE-384) 등이 있다.

3.2.5 에러 처리(Error Handling)

프로그램 수행 중에 발생할 수 있는 에러를 부적절하게 처리할 경우 발생할 수 있는 소프트웨어 취약성을 의미한다. 이러한 소프트웨어의 취약성을 공격자가 인지할 경우, 다양한 조작된 입력 값을 주어 프로그램이 에러를 내도록 하여 다른 공격에서 필요한 프로그램의 내부 작동 구조나 내부 정보를 습득할 수 있으며 제대로 에러 상황을 처리하지 않아 프로그램을 크래쉬 시킬 수도 있다.

여기에 속하는 취약성으로는 액션 없는 오류 조건 탐지(KCWE-390), 미검사된 에러 조건(KCWE-391), NullPointer- Exception을 통한 Null 포인터 사용 포착(KCWE-395), 포괄적 예외를 사용한 catch 선언(KCWE-396), 포괄적 예외를 사용한 throws 선언(KCWE-397), Finally 블록 내에서의 리턴(KCWE-584) 등이 있다.

3.2.6 코드 품질이 낮음을 표시하는 요소 (Indicator of Poor Code Quality)

작성된 프로그램에는 직접적으로 결점이나 보안과 관련된 취약성을 나타내지는 않으나 코드가 충분한 주의를 기울여 개발되고 관리되었는지를 표시하는 요소들이 있다. 즉, 프로그램 코드가 너무 복잡하여 관리하기 힘들거나 다른 시스템에 이식하기 힘들도록 되어있다든지 충분한 주의를 기울여 작성되지 않았음을 나타내는 요소가 있다면 이 프로그램에는 안전성을 위협할 취약성들이 코드 안에 숨겨져 있을 가능성이 높다. 이와 같이 코드의 품질을 나타내는 주요 취약성들로는 타입 불일치(KCWE-195, KCWE-9613, KCWE-9614, KCWE-9617), 메모리 누수(KCWE-401), 자원의 부적절한 반환(KCWE-404), 메모리 중복 반환(Double Free, KCWE- 411), Null 포인터 참조(KCWE-476), 더 이상 지원되지 않는 함수의 사용(Use of Obsolete Functions, KCWE-477), 사용되지 않는 코드(Dead Code, Unused Field/Method, KCWE-561, KCWE-9609, KCWE-9610), 중복된 Null 검사 (KCWE-9606), 쓰레드 조기 종료(Premature Thread Termination, KWCE-9620), 포맷 스트링 에러(Format String Error, KCWE-9607, KCWE-9622) 등이 있다.

3.2.7 불충분한 캡슐화(Insufficient Encapsulation)

캡슐화는 데이터와 연산에 대한 강력한 울타리를 치는 것인데, 이것이 실패하면 웹 브라우저 같은 경우에 이동 코드(mobile code)가 다른 이동 코드에 의해서 의도와 다르게 사용될 수 있으며, 서버의 경우에는 검증된 데이터와 검증되지 않은 데이터를 구분하지 못하게 되거나, 허용되지 않는 사용자들간의 데이터 누출이 가능해진다.

본 취약성 분류는 주로 Java 언어와 관련되며, 클래스 이름으로 클래스를 비교하기(KCWE-486), 세션 간의 데이터 누출(Data Leak between Sessions, KCWE-488), 디버깅용 코드 남겨두기(KCWE-489), 공개 메소드가 비밀 배열타입 필드를 반환하기(Private Array-Typed Field Returned From a Public Method, KCWE-495), 신뢰 경계 위배(Trust Boundary Violation, KCWE-501), super.clone을 사용하지 않는 clone 메소드 (KCWE-580) 등이 있다.

3.3 보안 취약성의 중요도 평가

취약성의 위험성을 나타내는 중요도를 계산하여 영향력을 평가하고, 문제 해결에 대한 우선순위 결정을 지원하기 위해 취약성 데이터베이스 내의 보안 취약성들에 대한 중요도 평가체계를 마련하였다. 이러한 작업은 기존의 CVE, CWE, CAPEC 등의 취약성 평가와의 관련성 및 전자정부 시스템 관련 각종 보안성 측정 방법과 연계를 고려하여 설계되었다.

정보보호란 데이터 또는 시스템에 대한 고의적이거나 실수에 의한 불법적인 공개(노출), 변조, 파괴 및 지체로부터의 보호를 의미한다. 즉, 데이터 및 시스템의 기밀성, 무결성, 가용성을 확보하는 것이며, 이들은 보안의 핵심 원칙들로서, 어떤 한 요소가 손상을 받게 되는 경우, 그 조직의 지속적인 존재 여부까지 위협할 수 있다. 따라서 보안 취약성 중요도는 기밀성, 무결성, 가용성을 기준으로 평가가 이루어진다.

기밀성, 무결성, 가용성 평가 메트릭은 3점 척도로서, Complete=1, Partial=0.7, None=0 값으로 정의된다. 각 기준에 대한 가중치 산정은 보안 영향력의 비중을 표현하기 위해 3가지 기준이 모두 균등한 경우 0.333으로, 각 기준에 비중을 둔 경우는 비중을 둔 요소의 가중치는 0.5, 나머지 두 요소의 가중치는 0.25로 할당하였다.

보안 취약성 항목의 중요도는 보안에 영향을 주게 되는 위험성(Severity)을 나타내며, 다음과 같이 S에 의해 계산된다. 계산된 S값의 범위는 0과 5 사이의 값으로 각 취약성 항목은 값에 따라 VeryHigh, High, Medium, Low, VeryLow의 중요도를 갖게 된다.

$$S = 5*(C*w_C) + (I*w_I) + (A*w_A)$$

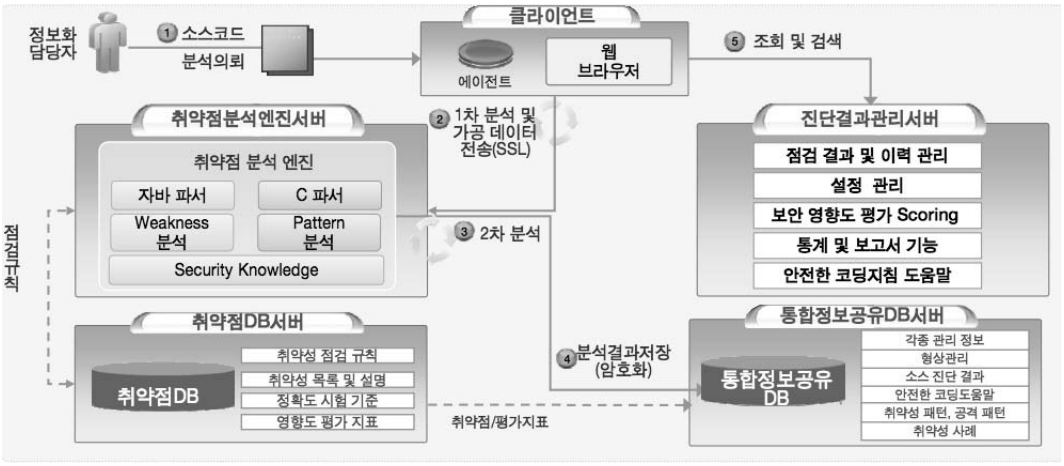
이렇게 계산된 중요도와 공격가능성 메트릭을 이용하여, 각 취약성들이 갖는 보안 영향력 지표 SI(Security Index)를 정의하였다. 취약성 항목별 보안 영향도는 소스코드 취약성의 보안 위험 관리에 대한 지표로 사용될 수 있다. 소스코드 취약성 항목별 보안 영향도 값이 큰 순서로 관리해야 하며, 평가 대상 프로그램에서 검출된 취약성들 가운데 영향도가 높은 취약성들이 존재한다면, 전체 프로그램의 보안 위험수준에 위협요소가 될 가능성이 높아지게 될 것이다.

4. 진단 체계

개발된 진단도구를 사용하여 실제 정부 발주 소프트웨어에 취약성 검사를 적용하는 체계는 [그림 2]와 같다. 전체적인 진단체계의 수행 과정을 설명하면 다음과 같다.

1) 소스코드 분석의뢰

- 정부의 정보화담당자는 클라이언트 에이전



[그림 2] 소스코드 취약성 자동진단 체계

트를 통해서 개발한 소스코드의 분석을 의뢰한다.

2) 1차 분석 및 가공 데이터 전송(SSL)
- 정보화담당자의 클라이언트 에이전트는 분석의뢰 대상 소스코드를 중간코드로 변환한다.
- 변환된 중간코드와 1차분석 도중에 가공된 데이터를 암호화하여 취약성분석엔진서버로 보낸다.

3) 2차 분석
- 취약성분석엔진에 1차분석의 결과가 전달되면 취약성DB로부터 분석에 요청된 최신의 점검규칙들을 가져온다.
- 취약성 분석엔진에서 변환된 분석의뢰 대상 중간코드와 점검규칙을 입력으로 받아서 취약성 분석을 수행한다.

4) 분석결과저장 (암호화)
- 분석된 중간코드의 진단결과를 암호화하여 통합정보공유DB서버로 전송한다.
- 통합정보공유DB서버는 전달된 진단결과를 저장 및 관리한다.

5) 조회 및 검색
- 클라이언트는 진단결과관리서버를 통해 분석이 완료된 코드들의 진단 결과, 보안영향도 평가지표에 따른 평가 성적표, 취약성이 없는 안전한 코딩 제안 등을 조회할 수 있다.

5. 결 론

2009년 행정안전부가 주관하고 한국인터넷진흥원이 수행한 “정보시스템 보안강화체계 구축” 사업은 정보화 프로세스에서 보안을 고려한 관리체계 구축을 통해 운영하기 전에 취약성 사전 제거로 보안 사고를 예방하고 이에 선도적으로 대응할 목적으로 시행한 사업이다. 이 사업의 결과로 개발된 소스코드 취약성 자동진단도구는 전자정부시스템에서 보안결함 발생을 최소화하는데 기여하게 될 것이며, 궁극적으로 국가 소프트웨어 보안수준 향상을 위한 안전한 소프트웨어 개발체계를 정립하는데 초석이 될 것이다.

앞으로 해야 할 과제로 다양한 언어에 대한 확장 작업과 추후로 드러나는 취약성에도 탄력적으로 대응하기 위한 체계 구축을 수행할 예정이며, 진단도구의 정밀도 향상을 위한 분석 기술 개발을 수행할 예정이다.

참 고 문 헌

[1] Fortify, http://www.fortify.com
[2] Sparrow, http://www.spa-arrow.com
[3] 코드프리즘, http://www.gtone.co.kr
[4] Hyunha Kim, Tae-Hyoung Choi, Seung-Cheol Jung, Oukseh Lee, Kyung-Goo Doh, Soo-Yong Lee, "Rule-based Source-code Analysis for Detection of Security Vulnerability", WISA2009:The 10th International Workshop on Information Security Applications, Busan, South Korea, August 25~27, 2009
[5] CWSS - Common Weakness Scoring System, http://cwe.mitre.org/cwss/
[6] Common Vulnerabilities and Exposures, http://cve.mitre.org
[7] Common Vulnerability Scoring System, http://www.first.org/cvss/
[8] CAPEC : Comon Attack Pattern Enumeration and Classification, http://capec.mitre.org/
[9] 2009 CWE/SANS Top 25 Most Dangerous Programming Errors, http://cwe.mitre.org/top25/
[10] Common Weakness Enumeration, http://cwe.mitre.org

권 현 준

1986 서울대학교 자연과학대학 수학과 학사
1990~1998 펜타컴퓨터 Senior 소프트웨어 엔지니어
1998~2008 아이티플러스 CTO
2008~현재 지티원 정보통신연구소 개발1본부장
관심분야: 소프트웨어 공학, 소프트웨어 구조분석, 애플리케이션 보안

김 유 경

1991 숙명여자대학교 수학과(학사)
1994 숙명여자대학교 전산학과 (석사)
2001 숙명여자대학교 컴퓨터과학과 (박사)
2001~2005 숙명여자대학교 정보과학부 컴퓨터과학전공 초빙교수
2005~2006 University of California Davis, 박사후연구원
2006~현재 한양대학교 공학대학 컴퓨터공학전공 연구교수
관심분야: 소프트웨어 품질평가, 웹서비스 신뢰성 평가, SOA 모델링

김 현 하

2003 한양대학교 전자컴퓨터공학부 (학사)
2005 한양대학교 컴퓨터공학과 (석사)
2006~현재 한양대학교 컴퓨터 공학과 (박사과정수료)
관심분야 : 프로그램분석, 소프트웨어보안

도 경 구

1980 한양대학교 산업공학과(학사)
1987 Iowa State University, Computer Science(석사)
1992 Kansas State University, Computer Science(박사)
1993~1995 University of Aizu 교수
2005~2006 University of California Davis 방문교수
1995~현재 한양대학교 ERICA캠퍼스 교수
관심분야: 프로그래밍언어, 프로그램분석, 소프트웨어 보안, 소프트웨어공학

신 승 철

1987 인하대학교 전산학과 (학사)
1989 인하대학교 전산학과 (석사)
1996 인하대학교 전산학과 (박사)
1999~2000 Kansas State University 박사후연구원
1996~2006 동양대학교 컴퓨터공학부 교수
2006~현재 한국기술교육대학교 컴퓨터공학부 교수
관심분야: 프로그램 분석 및 검증, 소프트웨어 보안, 수리논리

안 준 선

1992 서울대학교 계산통계학과(학사)
1994 KAIST 전산학과 (석사)
2000 KAIST 전산학과 (박사)
2000~2001 KAIST 프로그램분석 시스템연구단 연구원
2001~현재 한국항공대학교 항공전자 및 정보통신공학부 교수
관심분야: 프로그램 분석, 소프트웨어 보안, 유비쿼터스 컴퓨팅, 프로그램 병렬화

이 욱 세

1995 KAIST 전산학과 (학사)
1997 KAIST 전산학과 (석사)
2003 KAIST 전산학과 (박사)
2003~2004 서울대학교 컴퓨터 공학과 박사후연구원
2004~현재 한양대학교 컴퓨터공학과 교수
관심분야: 프로그램 분석, 포인터 분석, 프로그램 검증, 타입 시스템

이 은 영

1996년 고려대학교 전산과학과 (학사)
1998년 고려대학교 전산과학과 (석사)
2004년 Princeton University, U.S.A. (박사)
2005년~현재 동덕여자대학교 컴퓨터학과 조교수
관심분야: 프로그래밍언어, 소프트웨어 보안, 컴파일러

한 환 수

1993 서울대학교 컴퓨터공학과 (학사)
1995 서울대학교 컴퓨터공학과 (석사)
2001 University of Maryland, Computer Science (박사)
2001~2002 Intel, Senior Engineer
2003~2008 KAIST 전산학과 교수
2008~현재 성균관대학교 정보통신공학부 교수
관심분야: 컴파일러, 컴퓨터구조, 멀티코어 컴퓨팅