2013 한국컴퓨터종합학술대회 튜토리얼

SW 개발보안(시큐어코딩) 제도에 대한 이해

2013. 6. 28(금)

평가검증팀장 강필용

kangpy@kisa.or.kr

KISA 한국인터넷진흥원
Korea Internet & Security Agency

발표 순서

I 추진 배경

II SW 개발보안 제도

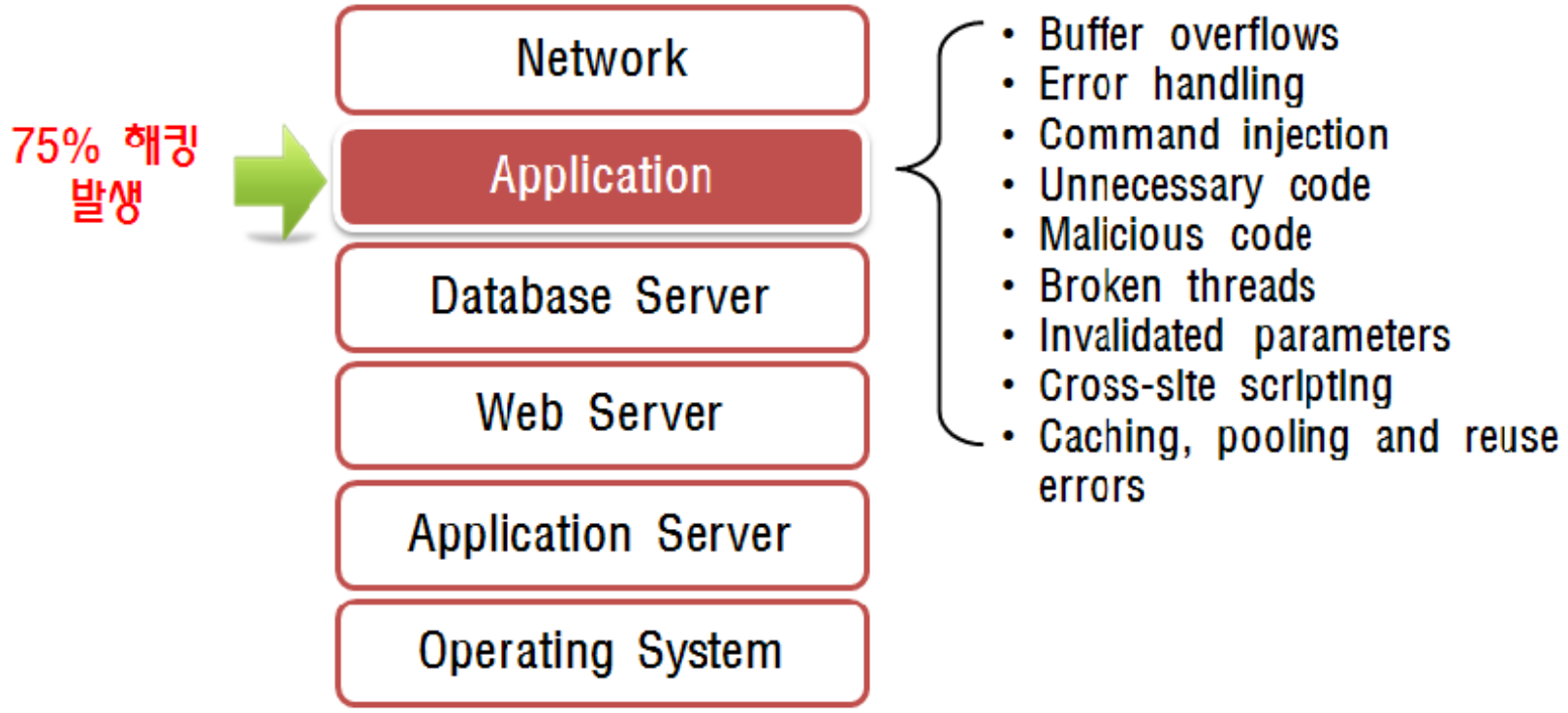
III SW 개발보안 진단

IV 지원현황 및 향후계획

I. 추진 배경

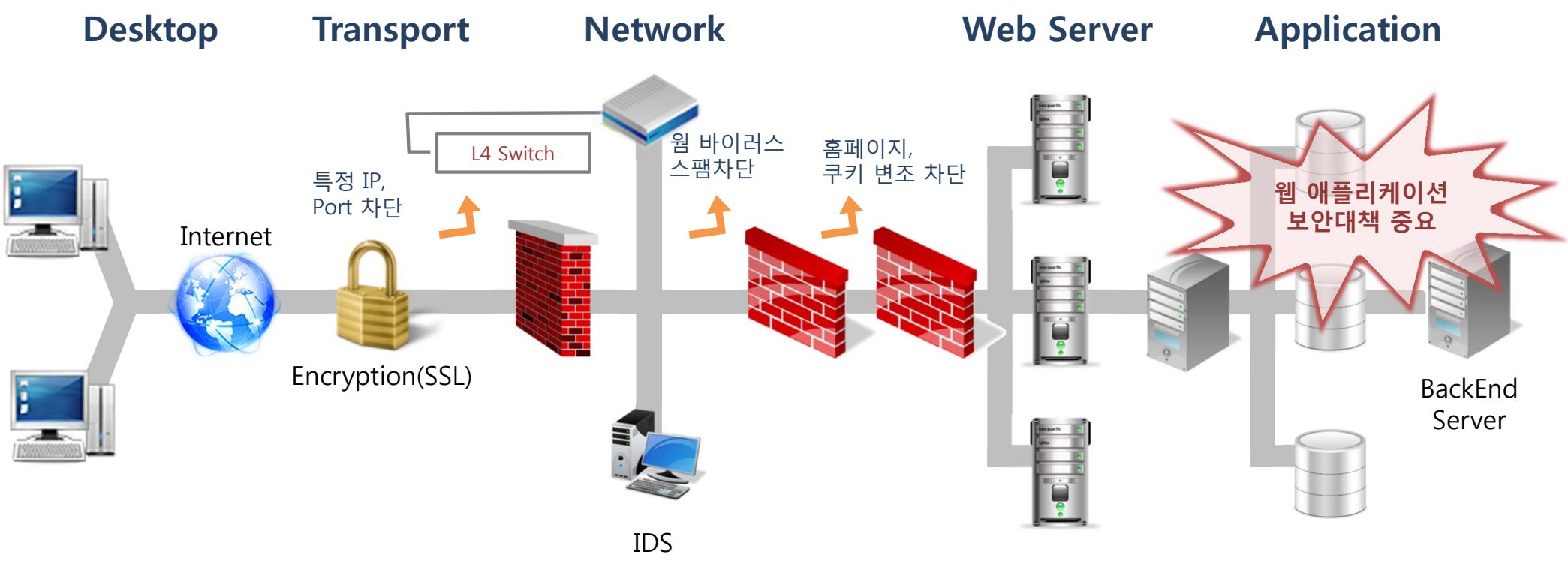
SW 보안강화 필요성 부각

- 상당수 **침해사고가 응용 SW에서 발생**하고 있음에도 관련 **보안투자는 미흡**
  - 응용 SW에 내재된 보안취약점을 악용, 계정탈취 · 정보유출 등 침해사고 유발



※ 출처 : 가트너, "Now is the time for security at Application Level"(2005. 12.)

[참고] 웹서비스 침해사고 발생빈도



	Transport	Network	Web Server	Application
발생 빈도	0.5% 미만	3%	10%	75%

※ 출처 : 가트너, "Now is the time for security at Application Level"(2005. 12.)

(참고) SW 취약점에 의한 침해사고 사례

SW 취약점에 의한
각종 피해 발생

공격개요	OO 사이트 해킹
취약점	XSS + Flash 취약점
피해	악성코드 배포
손실	일일 70만여명 감염

공격개요	OO증권 해킹
취약점	SQL Injection
피해	개인정보 유출
손실	개인정보 2만 6천여건

공격개요	OO 사이트 해킹
취약점	SQL Injection
피해	개인정보 유출 (1억여명)
손실	140억엔 손해

SW 개발보안 개념

SW 개발보안이란 ?

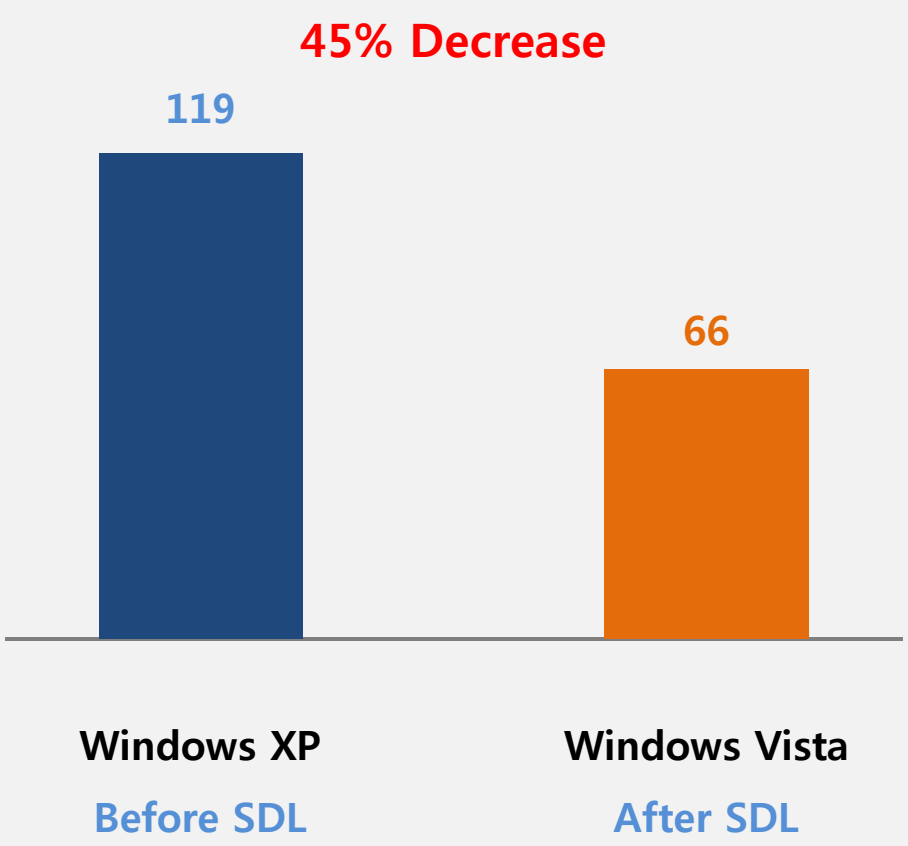
안전한 SW 개발을 위해 소스코드 등에 존재할 수 있는 잠재적인 보안취약점을 제거하고, 정보보호를 고려하여 SW를 설계 · 구현하는 등 SW 개발과정에서의 **일련의 보안활동**

요구사항	설계 및 디자인	구현	테스트
• 보안 요구사항 식별	• 위협원 도출 • 외부 인터페이스 식별 • 보안통제 수립	• 보안통제 구현 • 시큐어코딩 적용 (SW 개발보안가이드 참조 등)	
		• 소스코드 취약성 분석 (보안약점 진단 등)	• 실행코드 취약성 분석 (취약점 스캐닝, 모의해킹 등)

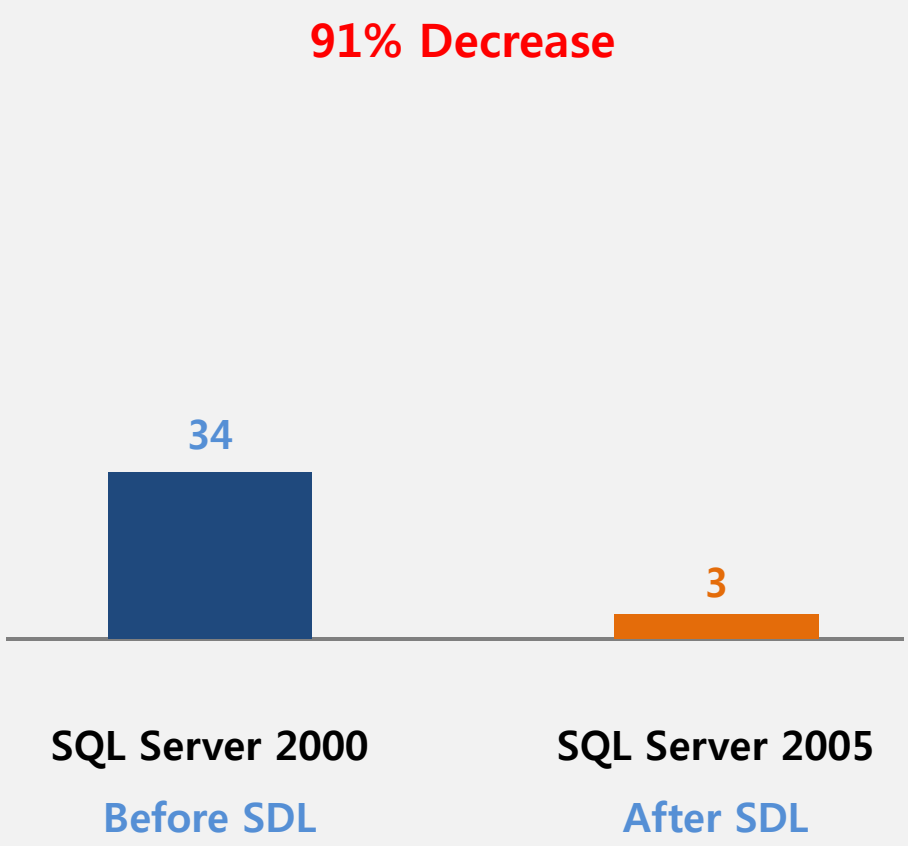
※ 현재는 SW 개발과정中 소스코드 구현단계를 중심으로 SW 개발보안 적용

(참고) MS SDL(Secure Development Lifecycle) 적용효과

[Vulnerabilities disclosed one year after release]



[Vulnerabilities disclosed three year after release]



※ http://www.microsoft.com/security/sdl

시큐어코딩 기반 개발보안 도입

- **보안취약점이 내포된 SW**는 해커의 공격목표가 되어 **심각한 보안위협 초래**
  - 사이버 침해사고의 **약 75%가 응용 프로그램(SW)의 취약점을 악용**(가트너, '05)
- 정보시스템 운영 이전 **개발단계**부터 보안성 고려 및 **잔존 취약점 제거 필요**
  - 운영단계에서의 취약점 제거 비용은 **개발단계보다 60~100배의 비용 소요**(IBM社 보고서)
- 사전 예방체계 강화를 위한 **SW 개발보안(시큐어코딩) 강화체계 도입** 추진
  - 시큐어코딩(Secure Coding) : SW **구현시 취약점을 배제**하기 위한 안전한 코딩 기법

선제적인 예방조치 강화를 위해 **SW 개발단계**부터 **취약점 진단 · 제거**를 강화

※ '10년 전자정부지원사업(10개) 적용 → '11년 23개 → '12년 33개, 개발보안 의무화(12월)

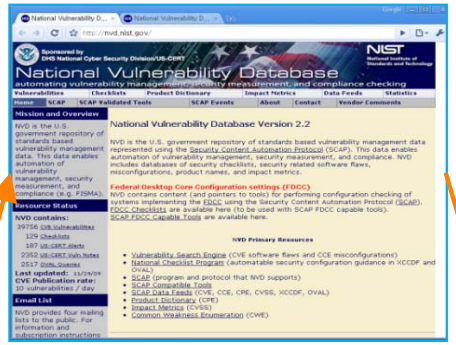
(참고) National Vulnerability Database

보안취약점 vs. 보안약점

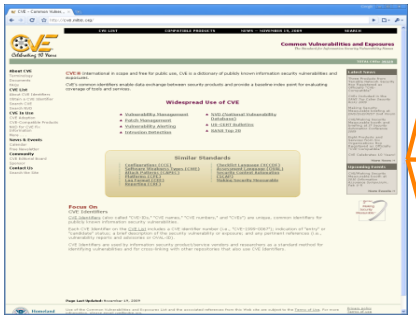
- **Vulnerability(보안취약점) : 해킹 등 실제 침해사고에 이용되는 SW Weakness**
  - SW 보안약점 중 해킹을 유발하는 실제 인스턴스를 의미
  - 개발단계에서 원인을 제거하거나, 구현 후 해커보다 먼저 찾아서 제거해야 함
  - CVE(Common Vulnerability Exposure)

- **Weakness(보안약점) : Vulnerability의 근본 원인이 되는 SW 결점, 오류 등**
  - 보안취약점은 SW 보안약점에 포함되며, 모든 보안약점이 보안취약점인 것은 아님
  - 시큐어코딩은 보안약점이 존재하지 않도록 개발하는 기법
  - CWE(Common Weakness Enumeration)

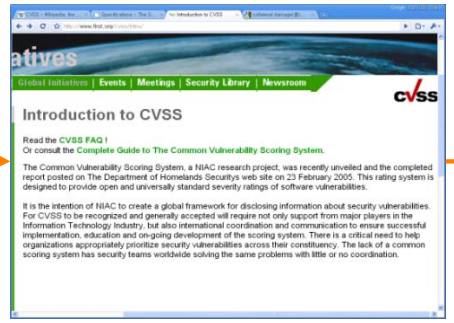
(참고) 보안취약점 및 보안약점 등 관계도



NVD.mitre.org

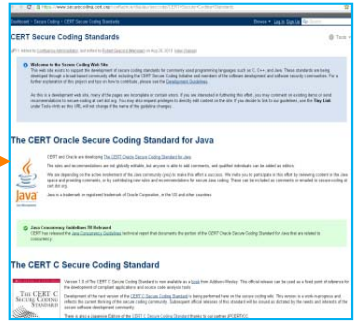


CVE.mitre.org



first.org/CVSS

CWE.mitre.org



SECURECODING. cert.org

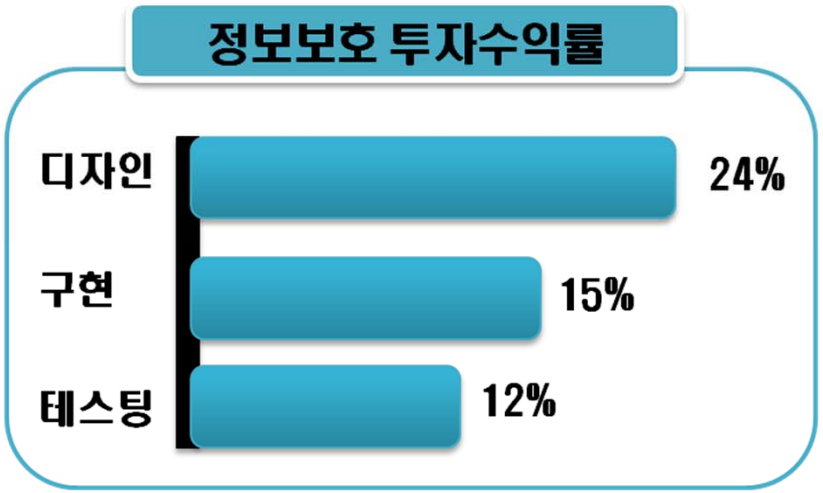
CAPEC.mitre.org

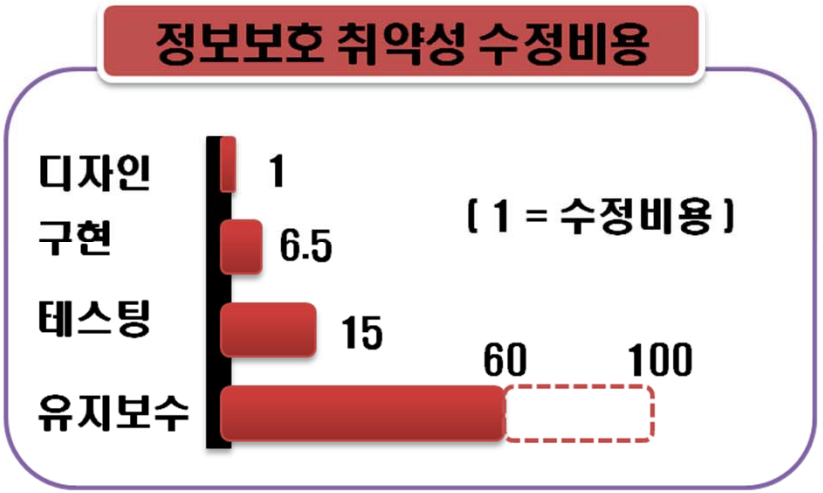
SW 개발보안 비용효과

< SW 개발단계별 결함 수정비용 분석 >

구분	설계단계	코딩단계	통합단계	베타제품	제품출시
설계과정 결함	1배	5배	10배	15배	30배
코딩과정 결함		1배	10배	20배	30배
통합과정 결함			1배	10배	20배

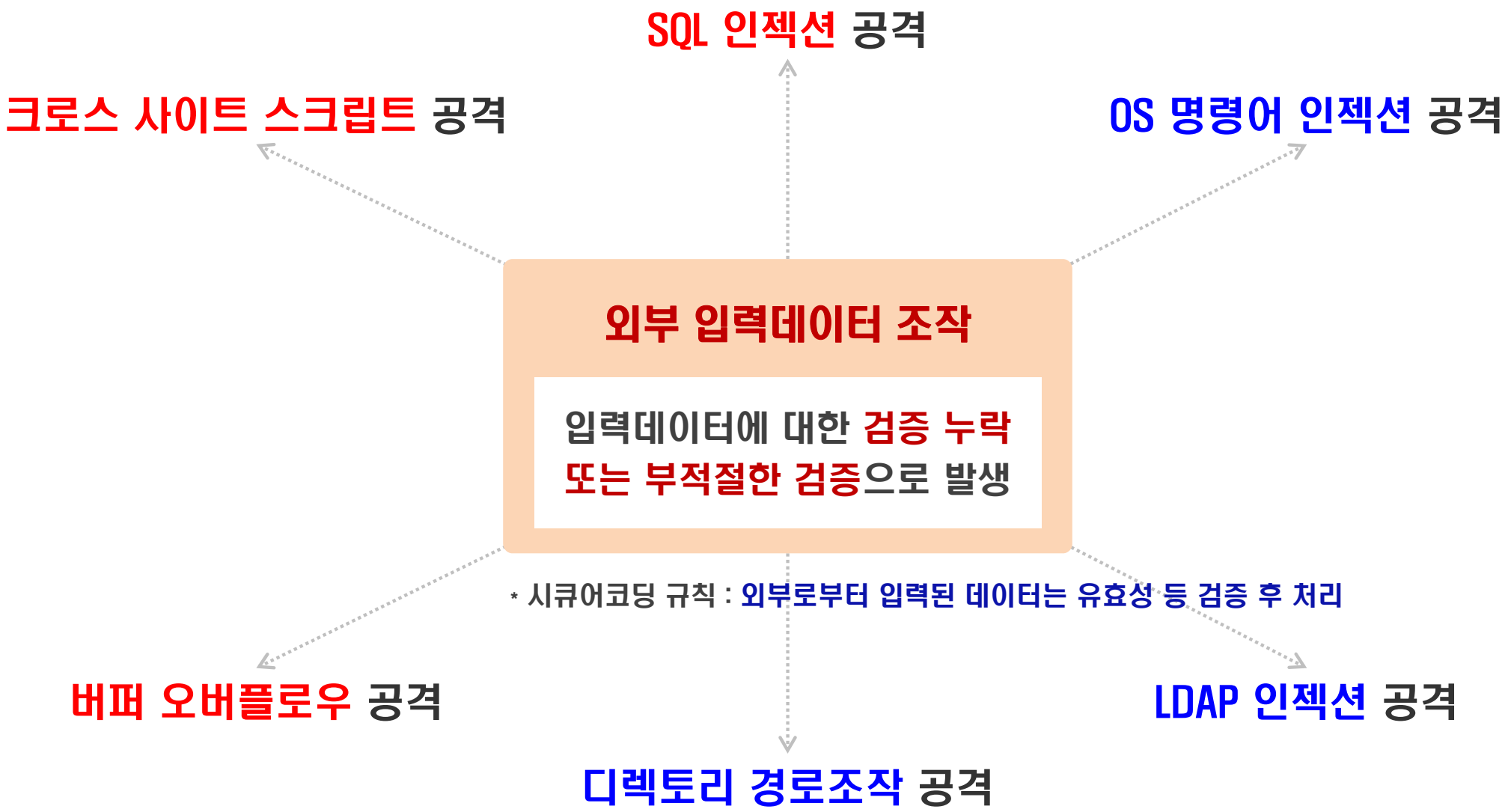
※ 출처 : The Economic Impacts of Inadequate Infrastructure for SW Testing(2002.5, NIST)





※ 출처 : IBM연구보고서

시큐어코딩 적용효과(예시)



※ 진단도구 적용 등 자동화 용이, 투자 대비 비용효과 우수

(참고) 2011 CWE/SANS Top 25 Most Dangerous SW Errors

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

* http://cwe.mitre.org/top25/

II. SW 개발보안 제도

제도 개요 : 대상 및 범위

정보시스템 개발단계에서 SW(소스코드) 보안약점 제거조치 의무화

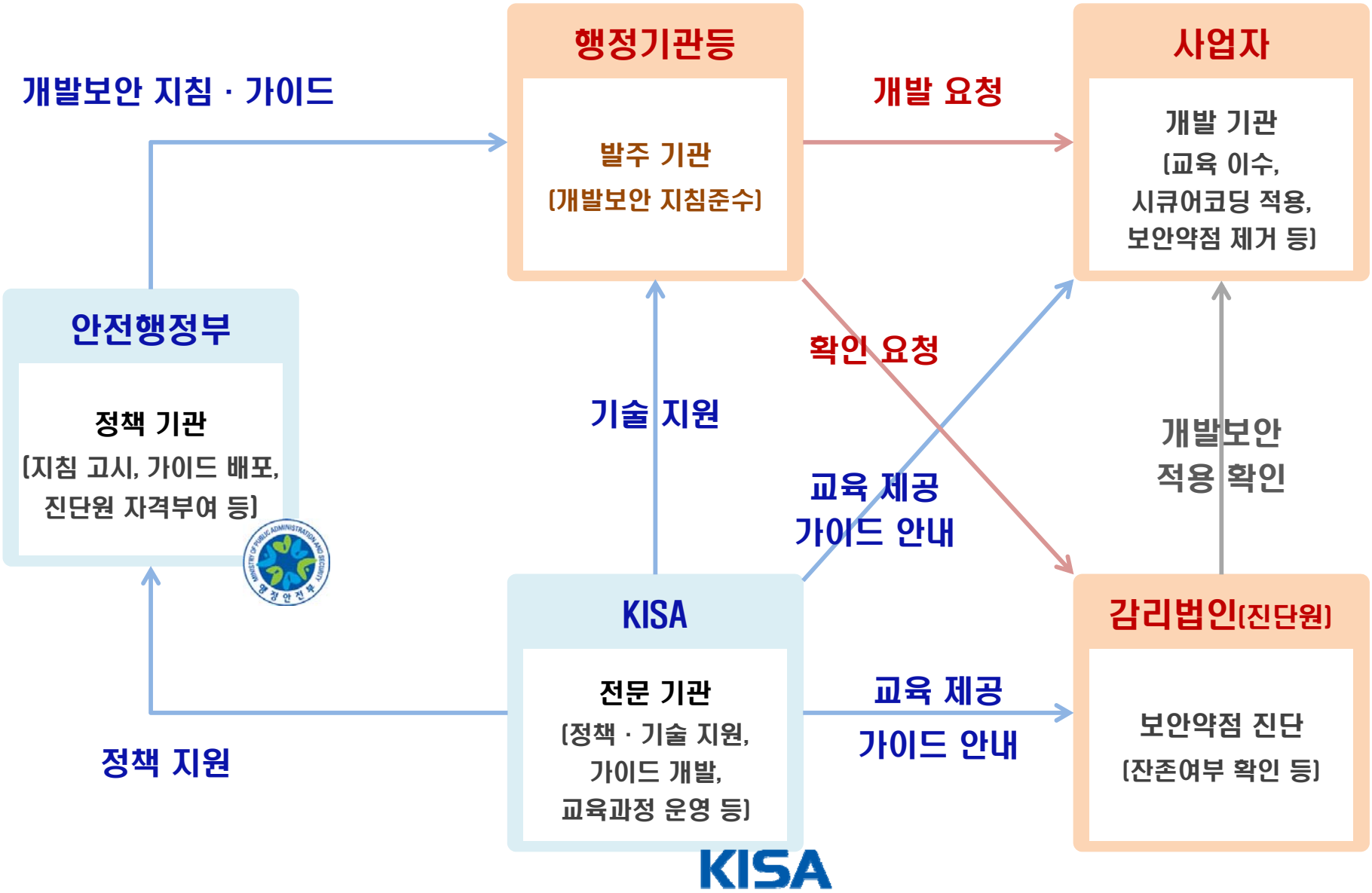
* 근거 : 정보시스템 구축 · 운영 지침(행정안전부고시 제2012.25호)

구분	내용	비고
대상	· 행정 · 공공기관 추진 **감리대상 정보화사업** ※ ('12.12월) 사업규모 40억원이상 → ('14.1월) 20억원이상 → ('15.1월) 감리대상 전체	단계적 확대
범위	· **소스코드**(신규개발 전체 또는 유지보수로 변경된 부분)	단, 상용SW 제외
기준 및 기타	· **SW 보안약점** 기준(SQL 삽입 등 43개 항목) ※ 정보시스템 구축·운영 지침 '별표3'	진단 기준
	· 감리법인이 **진단도구** 사용시, 국정원장이 인증한 도구를 사용 ※ 진단도구 : 소스코드상의 SW 보안약점을 찾기 위하여 사용하는 정적분석도구	'14.1월부터 적용
	· 감리법인이 SW 보안약점 진단시, **진단원**을 우선적으로 배치하도록 권고 ※ 진단원 : SW 보안약점 잔존여부 진단을 통해 조치방안 수립 · 조치결과 등 확인	진단원 활용

제도 개요 : 주체별 역할

구분	내용	비고
안행부	· 국정원과 협의하여 'SW 개발보안 가이드' 공지 · SW 보안약점 진단원 자격부여 및 요청시 진단원 자격유무 확인	KISA (정책, 기술지원)
행정기관등	· 제안요청서에 'SW 개발보안 적용' 명시 · 진단도구 사용여부, 개발절차 · 방법의 적절성 등 확인 및 평가에 반영 · SW 보안약점 진단 · 제거 결과 확인 ※ 감리법인(감리대상사업) 활용 또는 자체적(감리대상外사업)으로 수행	발주자 (공무원)
개발업체	· 개발인력 대상 SW 개발보안 관련 교육 실시 · SW 개발보안 가이드를 참조하여 개발 · 자체적으로 SW 보안약점 진단 및 제거	사업자 (개발자)
감리법인	· SW 보안약점 제거여부 진단 및 조치결과 확인 · SW 보안약점 진단시, 진단원 우선배치 · 진단도구 사용시, CC 인증(국정원)된 도구 사용('14.1월부터 적용)	감리원 (진단원)

SW 개발보안 체계



* 국가정보원 : 개발보안가이드 협의, 진단도구 CC인증('14.1월부터 인증된 도구사용 의무화)

SW 보안약점(지침 별표3)

유형	주요내용	개수(43)
입력 데이터 검증 및 표현	프로그램 입력 값에 대한 부적절한 검증 등으로 인해 발생할 수 있는 보안약점 (예) SQL 삽입, 자원삽입, 크로스사이트스크립트 등	14
보안기능	인증, 접근제어, 권한 관리 등을 적절하지 않게 구현시 발생할 수 있는 보안약점 (예) 부적절한 인가 허용, 중요정보 평문저장, 하드코드된 패스워드 등	16
시간 및 상태	멀티프로세스 동작환경에서 부적절한 시간 및 상태 관리로 발생할 수 있는 보안약점 (예) 경쟁조건(TOCTOU), 제어문을 사용하지 않는 재귀함수 등	2
에러처리	불충분한 에러 처리로 중요정보가 에러정보에 포함되어 발생할 수 있는 보안약점 (예) 오류상황 대응 부재, 오류메시지를 통한 정보노출 등	3
코드오류	개발자가 범할 수 있는 코딩오류로 인해 유발되는 보안약점 (예) 널 포인터 역참조, 부적절한 자원 해제 등	2
캡슐화	불충분한 캡슐화로 인가되지 않은 사용자에게 데이터가 노출될 수 있는 보안약점 (예) 제거되지 않고 남은 디버그 코드, 시스템 데이터 정보노출 등	5
API 오용	부적절하거나, 보안에 취약한 API 사용으로 발생할 수 있는 보안약점 (예) DNS lookup에 의존한 보안결정 등	1

※ **국외사례**(CWE, SANS Top25, OWASP Top10 등) **참조 및 시범사업 결과**(´10~´11년) **분석**

SW 보안약점 진단원 자격기준(지침 별표4)

(제1호) 자격기준

구분	자격기준	제출서류
기본요건	· 6년이상 소프트웨어 개발분야 업무를 수행한 자 · 3년이상 소프트웨어 보안약점 또는 보안취약점 진단·분석업무를 수행한 자	· 해당 업체 또는 기관에서 발급한 경력증명서 (업무내용 포함)
교육요건	· 기본요건을 만족하고 제2호의 기본교육을 이수한 자	

*** 자격 유지를 위해 1년마다 최신 취약점 항목 및 진단기술에 대한 보수교육을 수료해야 함**

(제2호) 진단원이 받아야 하는 교육

교육종류	교육내용	교육시기	교육시간
기본 교육	· 소프트웨어 개발보안 제도.기준 · 소프트웨어 보안약점 진단.제거기술 등 · 진단 수행능력의 배양을 위한 교육	진단원 자격을 취득 하고자 하는 경우	40시간
보수 교육	· 지침.기준 등 제도 변경사항 · 최신보안약점, 진단.제거기술 등 · 소프트웨어 개발보안 지식의 지속적인 습득 및 기술능력 유지를 위한 교육	진단원 자격을 유지 하고자 하는 경우	1년마다 8시간

III. SW 보안약점 진단

SW 보안성 분석 : 개요

- **보안기능 분석** : 중요 보안기능 구현여부 확인 등(개발문서 참조)
  - 예 : 사용자 인증 · 식별, 중요정보 암호저장 · 전송 등
- **보안 메커니즘 분석** : 안전한 알고리즘 적용여부 등(개발문서 참조)
- **취약점 분석** : 알려진 취약점 존재여부 확인(취약점 DB 등 활용)
  - 예 : SQL Injection, Buffer Overflow 등 중요 취약점 존재여부 확인 등
- **침투 시험** : 실제 해킹(또는 모의해킹)을 통해 취약여부 확인

SW 보안성 분석 : 테스트 유형

화이트박스 테스트

- ❖ 내부 구조 및 논리 위주 테스트
- ❖ 디버깅 및 단위 테스트 단계에서 수행

블랙박스 테스트

- ❖ 기능 및 I/O 위주 테스트
- ❖ 통합 및 인수 테스트 단계에서 수행

정적(static) 분석

- ❖ SW 실행없이 분석(소스코드 필요)
- ❖ SW 개발초기부터 분석 가능
- ❖ SW(소스코드) 전체를 진단

동적(dynamic) 분석

- ❖ SW 실행中 분석(실행코드 필요)
- ❖ SW 개발완료 후 분석 가능
- ❖ 기본적으로 샘플링 테스트 기반으로 수행

수동(manual) 분석

- ❖ 전문가가 수동으로 분석

자동(automatic) 분석

- ❖ 도구를 활용해 자동으로 분석

SW 보안성 분석 : 진단도구 기반 소스코드 분석

- 진단도구 기반 **자동화된 코드리뷰 지원**
- 매뉴얼 코드리뷰의 **단점을 보완** : 수백가지 보안문제를 **짧은 시간에 검토**
  - 평균 100~200 라인/시간 검토로 많은 시간 소요
  - 수백개 보안문제가 소스코드에 존재, 진단자 기억에만 의존은 한계
- 샘플링 테스트 기반의 동적 분석(모의해킹 등)에 비해 **소스코드 전체를 진단**
- **정 · 오탐 문제** 발생 및 **전문가 검토** 필수, 진단도구 신뢰성 중요

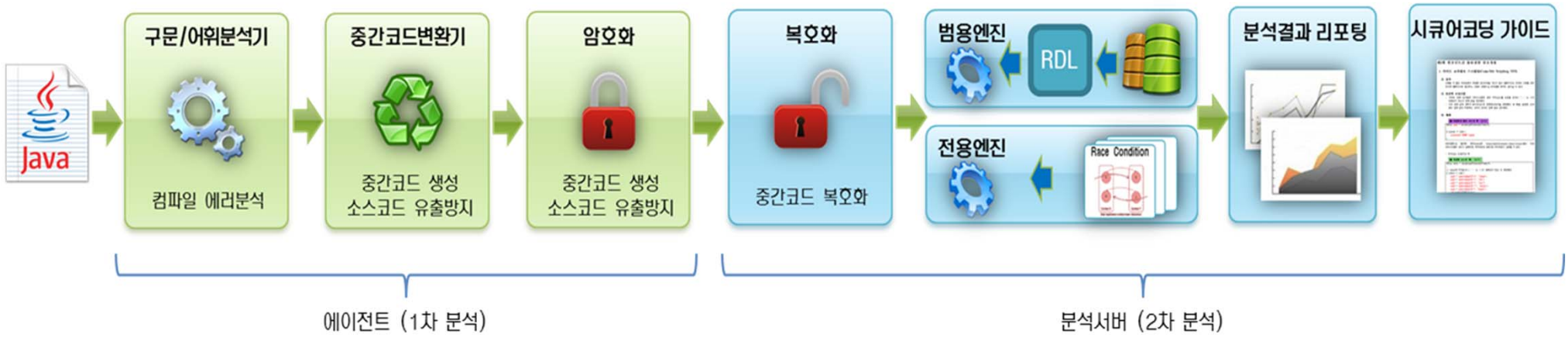
진단도구(예시) : SSEN v2.0

개요

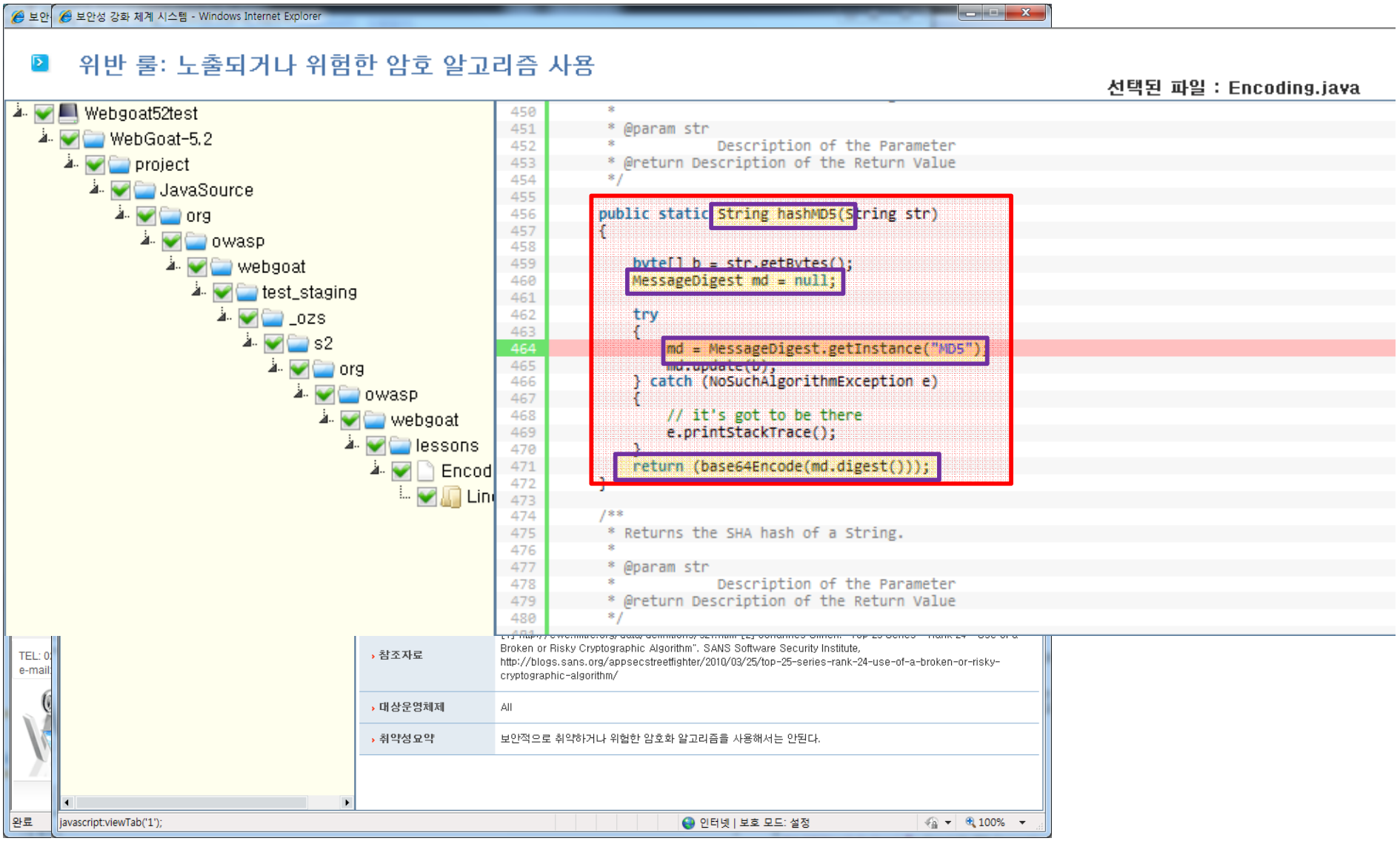
- **KISA 자체 개발도구('09~'11)**
  - ✓ Java, C, C++ 등 11개 언어에 대한 보안약점 진단 지원
  - ✓ 웹 기반의 진단결과 관리서비스 지원

소스코드 보안약점 진단

- 파일명과 소스 행 번호로 취약지점 제시
- 분석결과 레포트 자동 작성 및 시큐어코딩 예제 등 지원



(참고) SW 보안약점 진단(예시) : SSEN v2.0



사업단계별 개발보안 진단활동(예시)

진단계획(안) 수립 및 협의

- 개발보안 **진단기준 및 제거범위** 등 수립
  - 필수 보안약점(지침 별표3) 외 추가로 진단 · 제거할 항목 등
- **진단도구 적용방안** 수립 (가급적 개발자와 동일한 진단도구 적용)
  - 구현하려는 서비스 및 개발언어 등 특성을 고려, 적절한 보안약점 진단도구 선정
- SW 보안약점 **진단일정 수립**(최소 2회 진단)
  - 구현 후 **시험단계**에서 1차 진단 후, **검수단계**에서 조치결과 확인을 위한 2차 진단 수행
- 발주자 및 사업자 협의를 통해 **진단계획(안) 보완**
  - 원활한 진단활동을 위해 사전협의 필수

중간 점검(1차)

- **사전 교육과정 이수** 및 전달교육 실시여부 확인
  - 참여 개발인력의 20% 이상은 교육과정을 이수하고, 나머지 인력은 자체 전달교육 실시
- **개발보안가이드 준수여부** 확인
  - 적용 기준 및 지침 등 마련여부 및 참여 개발인력의 숙지여부 점검
- 사업자 자체 보안약점 **진단 · 제거 활동의 적정성** 확인
  - 보안약점 적용 및 보완조치 누락여부 등 점검
- 산출물(소스코드) 확인 → 진단도구 기반 **보안약점 존재여부 확인**
  - 보안약점 존재시 보완조치 요청 및 조치방안 컨설팅 수행

이행 점검(2차)

- 산출물(소스코드) 확인 → 진단도구 기반 **조치결과 확인**
  - 보안약점 존재시, 보완조치 요구 및 추가 진단일정 협의

- 원칙적으로 **필수 보안약점(43개)은 반드시 제거**여부 확인
  - 단, 사업자가 해킹 등에 악용될 소지가 없음을 증명할 수 있는 경우엔 협의 후 예외허용

개발보안 진단시 고려사항

- **소스코드 확인**
  - 신규 개발 또는 변경된 소스코드 누락여부 확인

- **진단규칙 확인**(튜닝) → 진단도구 기반 **보안약점 진단**
  - 사업자가 제공한 진단도구 활용시, 진단규칙 누락여부 등 사전확인 필수
  - 필요시, 동적 진단도구 및 개발문서(기능명세 및 설계서 등) 참조

- 정 · 오탐 판정 등 **진단결과 분석**
  - 보완조치가 요구되는 진단결과 등 정리

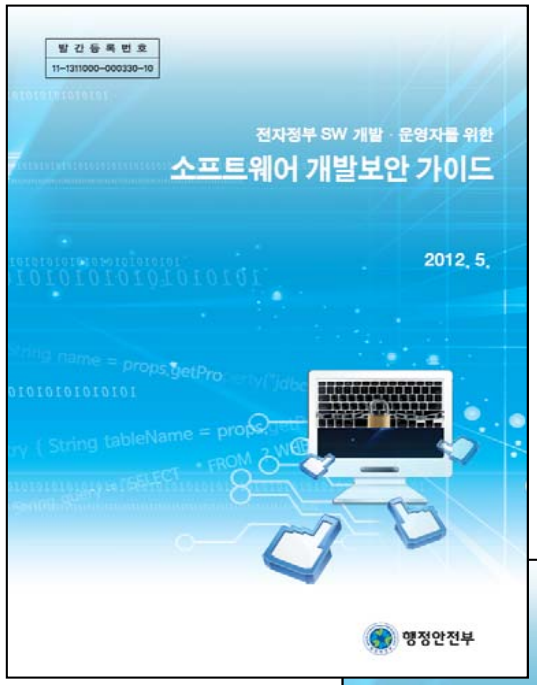
- **진단보고서 작성** 및 조치방안 **컨설팅** → 보완조치 **이행점검** 실시
  - 반드시 조치가 필요한 사항들 정리 및 보완조치 적절성 등 검토

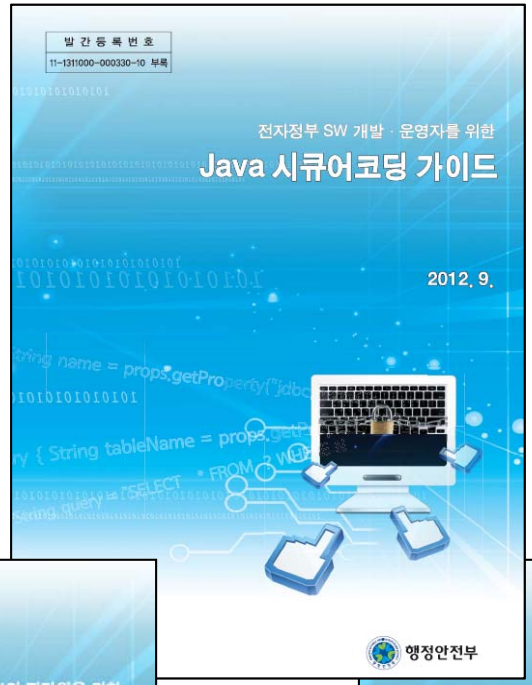
Ⅳ. 지원현황 향후계획

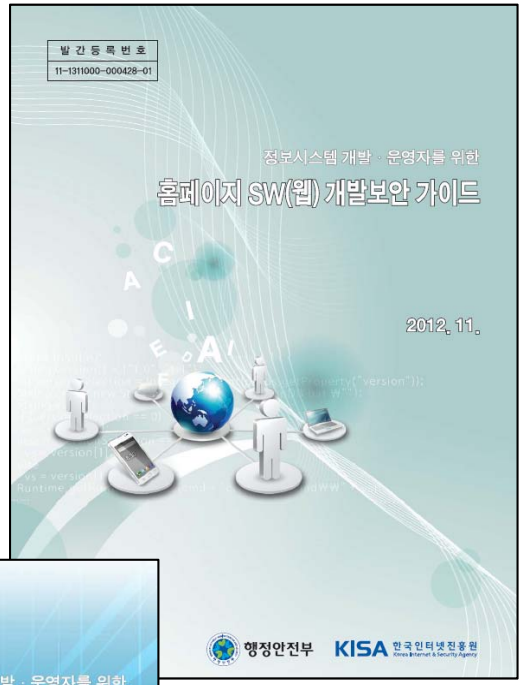
SW 개발보안 정착기반 조성

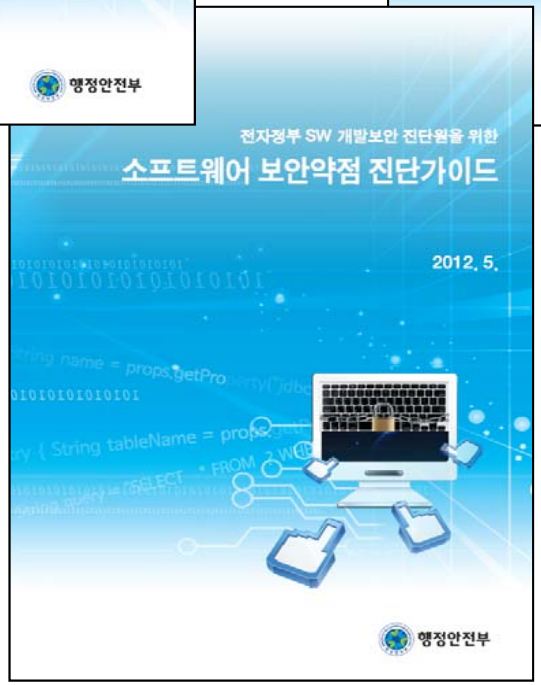
가이드 보급	● 개발자, 진단원 등을 위한 **보안약점 진단 · 제거방안 등 제공** ● 개발보안 가이드, 진단 가이드, Java · C 시큐어코딩 가이드 등 배포
교육과정 운영	● **참여주체 인식제고 및 역량강화**를 위한 교육과정 운영 ● 공무원(일반과정), 개발자(실무교육), 진단원('12년 **82명 양성**) 등
기술 지원	● 전자정부서비스 개발단계 **보안약점 진단 · 제거** 지원('12년 33건) ● 보안약점 **진단도구 시범검증** 및 기술지원('12~13년)
개발보안 연구센터 지원	● **전문연구기관**을 선정, SW 개발보안 기반기술 등 **연구지원** ● 국내외 동향분석, 필수 보안약점 도출 및 진단 · 제거기술 연구 등

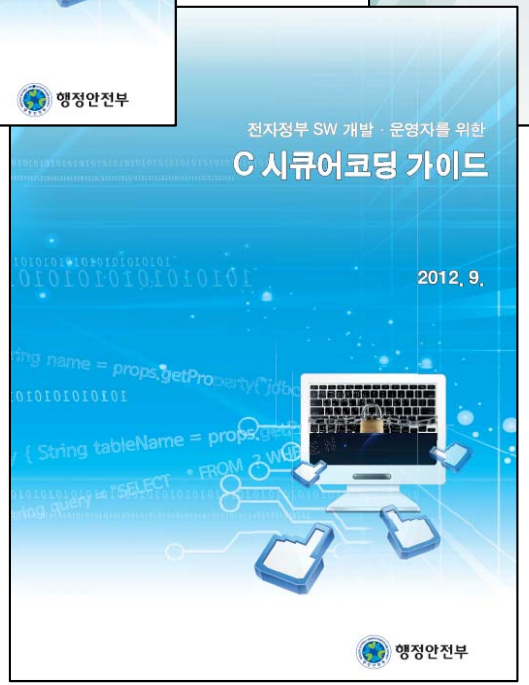
보안가이드 배포 및 교육과정 운영







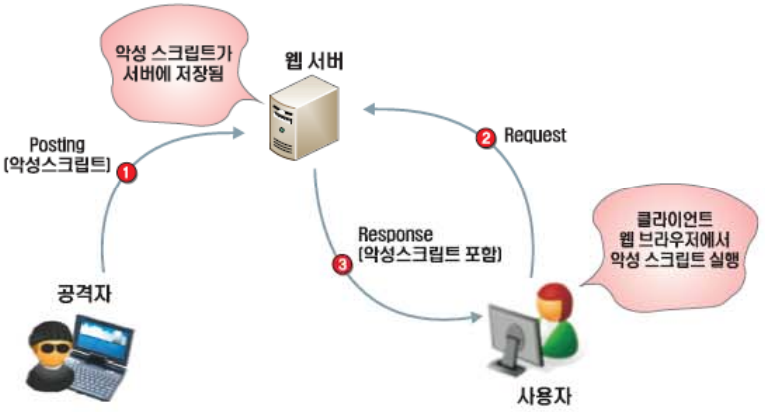




3. 크로스사이트 스크립트

가. 개요

웹 페이지에 악의적인 스크립트를 포함시켜 사용자 측에서 실행되게 유도할 수 있다. 예를 들어, <그림 3-3>과 같이 검증되지 않은 외부 입력이 동적 웹페이지 생성에 사용될 경우, 전송된 동적 웹페이지를 열람하는 접속자의 권한으로 부적절한 스크립트가 수행되어 정보유출 등의 공격을 유발할 수 있다.



<그림 3-3> 크로스사이트 스크립트

나. 보안대책

일반적인 경우에는 사용자가 문자열에 스크립트를 삽입하는 것을 막기 위해 사용자가 입력한 문자열에서 <, >, &, ", ' 등을 문자 변환 함수나 메소드를 사용하여 &lt, &gt, &amp, &quot로 치환한다. HTML 태그를 허용하는 게시판에서는 게시판에서 지원하는 HTML 태그의 리스트(White List)를 선정한 후, 해당 태그만 허용하는 방식을 적용한다.

다. 코드예제

파라미터(name)에 <script>alert (document.cookie);</script>와 같은 스크립트 코드가 입력되고, 이 값이 그대로 사용되면 공격자에게 피해자의 쿠키정보가 전송될 수 있다.

안전하지 않은 코드의 예 JAVA

```
1: <h1>XSS  Sample</h1>
2: <%
3:   String name = request.getParameter("name");
4: %>
5: <p>NAME:<%=name%></p>
```

외부 입력 문자열에서 replaceAll() 메소드를 사용하여 <, >, &, ", ' 같이 스크립트 생성에 사용되는 문자열을 &lt, &gt, &amp, &quot 등으로 변경하면, name에 악성코드가 포함되더라도 스크립트가 실행되지 않는다.

안전한 코드의 예 JAVA

```
1: <%
2:   String name = request.getParameter("name");
3: if ( name != null ) 4:  {
5:   name = name.replaceAll("<","&lt;");
6:   name = name.replaceAll(">","&gt;");
7:   name = name.replaceAll("&","&amp;");
8:   name = name.replaceAll("""","&quot;");
9: }
10: else { return; }
11: %>
```

보안약점 진단기술 수준제고

SW 보안약점 진단기술을 **민간에 기술이전**(25건, '12.9월)

대상	· SW 보안약점 **진단도구 개발업체** 및 **희망기관 등**
내용	· KISA 개발 SW 보안약점 진단도구 **진단규칙(Rule) 및 설명서(JAVA용)** ※ 진단규칙은 명세언어(RDL) 기반 범용규칙과 전용규칙(VSP)으로 구성 · SW 보안약점 진단도구 시범검증시 사용되는 **검증용 샘플코드** ※ 주요 보안약점 진단여부 확인을 위한 검증코드

보안약점 진단도구 시범검증 및 기술지원(7개 업체, '12.10~'13.3월)

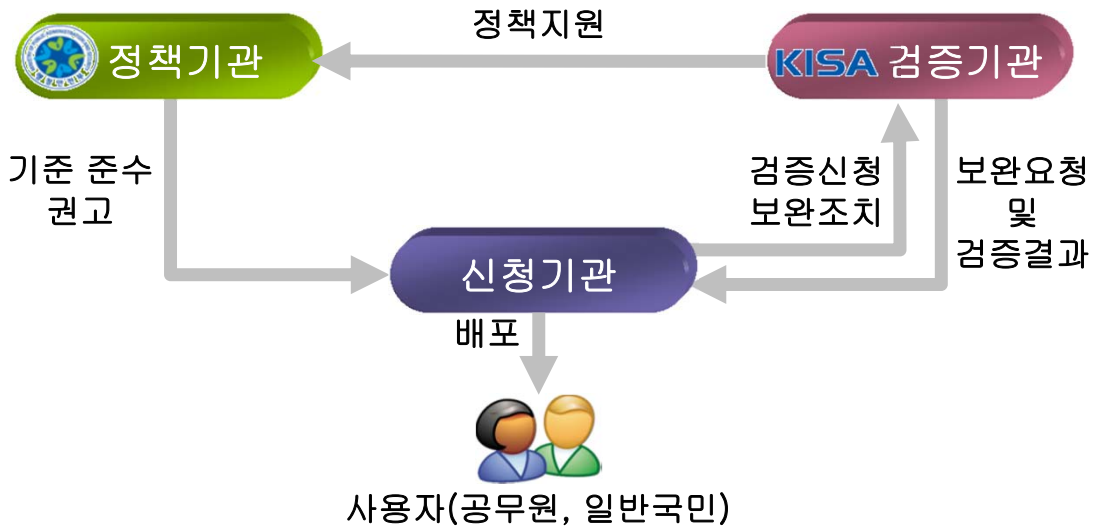
기준	**SW 보안약점(43개) 진단여부 및 수준(정확도 등)** 등 진단도구 신뢰성
지원 사항	(검증前) SW 보안약점 진단도구 **검증코드 및 진단규칙(예제 및 설명) 제공** (검증後) **보완사항 자문**, 업체 요청시 진단도구 **재검증**

[참고] 진단도구 검증코드

보안약점	검증코드 예시
SQL 삽입	**data = buffread.readLine();** … Boolean bResult = sqlstatement.execute("insert into users (status) values ('updated') where name='"+**data**+"'");
운영체제 명령어 삽입	**data = buffread.readLine();** osCommand = "/bin/ls "; … Process p = Runtime.getRuntime().exec(osCommand + **data**);
XPath 삽입	**data = buffread.readLine();** String [] tokens = **data.split("||")**; … String **uname** = tokens[0]; String **pword** = tokens[1]; … String **query** = "//users/user[name/text()='" + **uname** + "' and pass/text()='" + **pword** + "']" + "/secret/text()"; String secret = (String)**xp.evaluate**(**query**, inxml, XPathConstants.STRING);
하드코드된 패스워드	**data = "pass";** String **pw = data**; … conn2 = DriverManager.getConnection("data-url", "root", **pw**);

모바일 전자정부서비스 보안성 검증

모바일 전자정부서비스 **앱 보안성 검증센터 오픈**(2011.9월~, 390여건 검증)



검증 절차

< 준비 및 신청 >

- 제출물 준비

(제출물: 보안명세서, 실행앱, 소스코드)

- 검증신청

(공문, 신청서, 제출물)

검증 수요에 따라 검증대기 가능

< 보안성 검증 >

- 소스코드·기능보안성 점검
- 보안조치 및 조치내역 확인

최초 : 1주, 조치확인 : 1주이내

< 배포 및 서비스 >

- 각 기관에서 모바일 전자정부 지원센터에 개별신청

지원센터 : 2주(필요시, 2주 추가소요)

개발보안 적용단계 확대(1/2)

- 정보시스템 **개발단계**부터 보안성 고려 및 **SW 보안약점 진단 · 제거** 강화
  - (**개발**) 시큐어코딩 기반 SW 보안약점 배제 (**진단**) 실제 SW 보안약점 진단 · 제거여부 확인

개발사	교육수강	가이드 준수	자체 진단	보안약점 제거	유지보수
SW 생명주기	사전교육	분석/설계/구현	테스트	검수	운영
안행부 · KISA	개발보안 실무교육	개발보안 가이드	진단 가이드	보안약점 진단	관리자 보안교육

개발보안 적용단계 확대(2/2)

운영단계 SW에 개발보안 적용 시범검증 추진

- **운영중인 시스템** 대상 **개발보안 적용 시범사업** 추진('13년)
  * 운영시스템은 SW 수정 비용부담, 장애우려 등으로 인해 보안약점 진단 · 제거 곤란
- **사용자 인터페이스**와 관련된 웹 프로그램을 대상으로 보안약점 제거여부 확인

주요 내용

- 시스템 운영기관이 유지보수 사업을 통해 **홈페이지(웹) 취약점 제거**
  * 유지보수 계약에 SW 개발보안 적용(진단결과 보완조치 등) 반영
- 실무자가 활용할 수 있는 「홈페이지 SW(웹) 개발보안 가이드」 제작 · 배포
- 안행부(KISA)에서 홈페이지(웹) **소스코드 보안약점 진단** 및 조치방안 설명 등 지원
  * '13년 수요조사를 통해 신청접수(160건) 및 진단서비스 제공中

보안가이드 개선 및 교육과정 강화

- (보안가이드) 주요 개발언어별 **시큐어코딩 가이드 확대 · 개선**
  - 최신 침해사고 동향 및 대책 반영, 모바일 전자정부서비스 관련 **보안가이드(Objective C 등) 신규개발** 등

- (교육과정) 개발보안 인식제고 및 역량강화를 위한 **교육과정 확대 · 개선**
  - **일반 · 전문과정 확대**('13년 1,800명), **진단원 양성교육 확대**('12년 82명 → '13년 120명 등 200명 양성)

[참고] 개발보안 교육과정 운영현황

구분	교육과정	교육일/회수	인원	비고
일반	① SW 개발보안 일반과정 ※ 제도소개 및 담당자의 역할, 보안약점 이해 등	4시간 (16회) 매달 1회 이상 개설	800명	개발·운영 담당 공무원
전문	② SW 개발보안 개발자과정 ※ 제도소개, 보안약점 기준 및 개발보안 기법 등	1일 (15회) 상반기 집중개설	600명	정보시스템 개발자
	③ SW 개발보안 개발자 심화과정 ※ 보안약점별 개발보안 및 조치방법 실습 등	1일 (10회)	400명	
진단원	④ SW 보안약점 진단원 양성과정 ※ 보안약점 기준 및 진단기법·분석 실습 등	5일 (3회)	120명	자격해당자
	⑤ SW 보안약점 진단원 보수교육 ※ 변경된 제도 소개, 신규 보안약점 진단기법 등	1일 또는 2일(선택) (2회)	82명	SW 보안약점 진단원

※ 교육신청 기본요건(지침 별표4) : SW 개발(6년) 또는 보안약점 진단(3년) 경력자

맺음말

- 사업 착수단계에 발주자 · 사업자간 **사전협의 필수** → 진행中 이견 최소화
  - 진단항목 및 조치범위, 적용도구, 진단일정 등
- 개발자와 **동일한 진단도구 적용** (가급적 사업자가 기보유한 진단도구 활용)
  - 검수지연 등 방지를 위해서는 **참여주체간 사전협의 중요**
- 보안약점 진단도구 한계를 인식, **정적 · 동적 진단기법** 등 적절히 **적용**
  - 개발단계에서의 보안약점 진단도구 기반 진단 · 제거 활동은 **최소한의 보안조치**
- 감리원은 진단규칙 외에도 해당 정보시스템에 대한 **보안위협 분석결과 검토중요**
  - 서비스 및 플랫폼 등 특성을 고려한 보안기능(중요정보 암호화 저장/전송 등) 확인 등

감사합니다

해킹·스팸·개인정보침해 신고는 ☎118